Discovery of Design Methodologies for the Integration of Multi-disciplinary Design Problems

by

Cirrus Shakeri

B.S. (University of Tehran, Iran) 1989
M.S. (University of Tehran, Iran) 1994

A Dissertation Submitted to the Faculty of the

WORCESTER POLYTECHNIC INSTITUTE

in partial satisfaction of the requirements for the

Degree of Doctor of Philosophy

in

Mechanical Engineering

by

Fall, 1998

Approved (in alphabetical order):

Professor David C. Brown, Advisor

Professor Michael A. Demetriou, Member

Professor Allen H. Hoffman, Graduate Committee Representative

Dr. Susan E. Lander, Member

Professor Mohammad N. Noori, Advisor

Professor John M. Sullivan, Jr., Member

Discovery of Design Methodologies for the

Integration of Multi-disciplinary Design Problems

Abstract

Discovery of Design Methodologies for the Integration of Multi-disciplinary Design Problems

by

Cirrus Shakeri

Doctor of Philosophy in Mechanical Engineering

Worcester Polytechnic Institute

In order to succeed in today's global, competitive market, manufacturing industries need continuous improvements in their multi-disciplinary design processes. These improvements should result in expending fewer resources on the design process while achieving better quality and more environmentally friendly products. The current approach for improving design processes is mostly based on intuitive observations followed by incremental changes to the existing methodologies. However, today's fast-paced world needs rapid incorporation of new technologies and methods into design methodologies. Recent advances in the application of Artificial Intelligence to design-Multi-agent Design Systems in particular-provide an opportunity to accomplish this goal. The inter-disciplinary collaboration between Computer Science and Engineering Design provides the means to develop systematic and holistic approaches for constructing superior design methodologies.

An innovative approach has been developed that is based on simulating the design process using a multi-agent system that mimics the behavior of the design team. The multi-

agent system implements a knowledge-based model of design in which highly specialized knowledge from expert sources is applied to synthesize a design. The multi-agent system activates the pieces of design knowledge when they become applicable. The use of knowledge by agents is recorded by tracing the steps that the agents have taken during a design project. Many traces are generated by solving a large number of design projects that differ in their requirements. A set of design methodologies is constructed by using inductive learning techniques to generalize the traces generated. These methodologies then can be used to guide design teams through future design projects.

Acknowledgments

I would like to express my deepest gratitude to my advisors Professor Mohammad Noori and Professor David Brown. Professor Brown has provided the key technical insights, has critically read through several drafts of this work and has made himself readily accessible. He has made many direct contributions to this work and I have very much enjoyed and benefited from the intellectual discussions that we had during this time. Professor Noori has provided a far-reaching vision of the future of inter-disciplinary research and the financial support for this work. I would like to thank him for his personal support, trust, and understanding during the difficult times in my doctoral education journey. My thesis owes much to this unique combination of advisors.

I would like to thank my entire committee, Professor Michael A. Demetriou, Allen H. Hoffman, Professor John M. Sullivan, and particularly Dr. Susan E. Lander who provided invaluable feedback and guidance during this work.

I cannot miss this chance to thank my friends whom their support and care helped me temper the obstacles I faced during these years. Especially, I owe the deepest affection and gratitude to Zahed and Debbie Sheikholeslami for their constant support during this undertaking. I would like to thank them for their unconditional friendship and inspiration. My friend Siamak Najafi provided me with the technical and moral support. I would like to thank him for all the overtime hours that he put on making the systems ready and available for my work.

I would not have been able to have done this work without the support of the Mechanical Engineering Department at WPI. I am indebted to the faculty, staff, and graduate students of both Mechanical and Computer Science departments for their support and help throughout my four years at WPI. Especially, I would like to thank Barbara Edilberti, the secretary of the graduate program at the Mechanical Engineering Department, for her constant support during these years.

I would also like to acknowledge the inputs that the members of AIDG (AI in Design Research Group) at the Computer Science Department and CLPSI (Center for Loss Prevention and Structural Integrity) at the Mechanical Engineering Department provided. CLPSI has generously provided the financial support for the last year of this work.

Finally, I would like to dedicate this dissertation to my parents, my sisters Sima and Sudabeh, my brother in law Ata, and my niece and nephew Mina and Amir Reza. Their profound and unconditional love and belief in me lit the path for me throughout my life. This dissertation is the fruit of my labor and their love.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

The context of this research is the *simulation of complex processes* in order to synthesize, analyze, or modify their emergent behavior. Specifically, this research addresses issues within the broad context of multi-disciplinary design. Multi-disciplinary design entails participation of different disciplines in the design process. Examples of multi-disciplinary design are design of aircraft, automobiles, robots, and buildings.

The general scope of this project is the multi-disciplinary design of engineered systems. This work extends the concept of analysis-by-simulation to the area of engineering design research. Analyzing the behavior of physical systems in engineering applications by computer simulation using mathematical models has been a powerful tool in engineering, reducing costs and time in comparison to physical prototyping and experimentation.

In this work the same concept is applied to the design process instead of the design product. A computational model in the form of a knowledge-based multi-agent system is built that simulates the design process. By running the simulation under different conditions, and examining the performance, detailed understanding of the design process is gained. As for simulations of physical systems, the computational model of the design process is a simplified one in which the design activities that are usually carried out by humans are performed by software agents in a slightly simplified manner. We have developed these ideas using the multi-disciplinary domain of robot arm design.

This work is a new approach to multi-disciplinary design based on *integration* of different disciplines. Integration, however, makes an already complex design process even more complicated. To overcome this complexity, computer programs are developed based on multi-agent systems technique. The program simulates examples of multi-disciplinary design processes while applying integration principles on the problem. Based on the traces produced by the program, some candidate design methodologies are extracted.

This research is an inter-disciplinary exploration between engineering design and computer science. The positive side of an inter-disciplinary exploration is taking advantage of the power of the different disciplines. However, the inter-disciplinary aspect makes the research more difficult to start and also to present to discipline-based people.

This introductory chapter is an overview of the thesis. In the next section we give an example of the design methodology that has been discovered by the approach proposed in this dissertation. This will help to better understand the more abstract and formal discussions that follow in the rest of this chapter. We then give the motivation for pursuing this research, followed by a formal problem statement. Next we formulate the goal and objectives, give the significance of this research, review the approach and implementation, and finally describe the outcomes of and potential applications for this work.

1.2 A Methodology Discovered

Figure 1-1 shows an example of a design problem that a design team might encounter in the field of robotics. The design problem is defined by a set of specifications that the robot is required to have (items 1 to 4 in Figure 1-1) and a list of constraints on the rest of specifications of the robot.

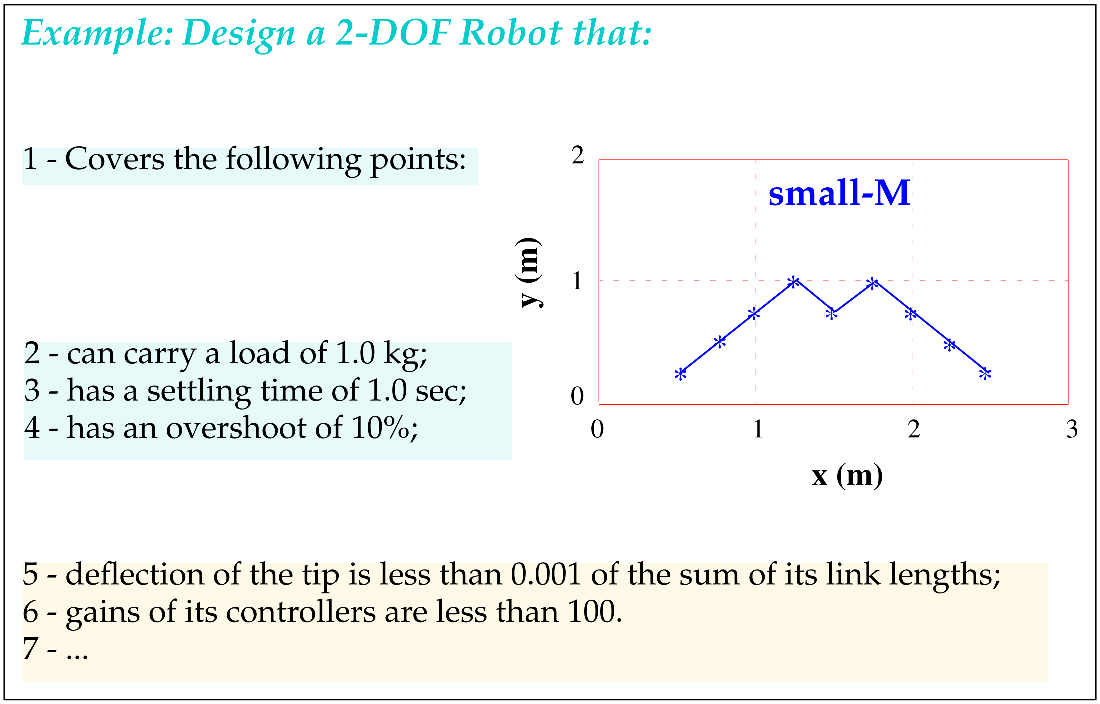


Figure 1-1. An Example of the Design of a 2-DOF Planar Robot.

The question that the design team is facing is: What methodology should we use for designing a robot with specifications shown in Figure 1-1? The design methodology provides an answer to the question of: *How should we conduct the design process?* Some of the questions that originate from the general question are as follows:

- What design methods to use?
- In what order should the design methods be used?
- When should the members of the team stop to exchange the partial designs?
- How should the members of the team evaluate the partial designs?

- How should the members of the team cooperate?
- How should the members of the team do things concurrently?
- etc.

Figure 1-2 shows a methodology discovered based on the approach proposed in this dissertation. This methodology can not only answer the above questions, it also facilitates the integration of different points-of-view in the design. Integration of different points-of-view speeds up the design process and reduces the resources required to conduct the design.

In the rest of this chapter we provide an overview of this dissertation in a more formal way by answering the 'why', 'what', and 'how' questions regarding the approach proposed.

Methodology

- **choose** the location of the base of the robot: “left or below midway of the workspace length”
- **choose** the material: “steel stainless AISI 302 annealed”
- **select** the shape of the cross section of the link: “hollow round”
- **choose** the structural safety factor: “3”

  - **do** the design and proceed to the next step

- **choose** the link 2 to link 1 length ratio: “0.5”

  - **do** the design and proceed to the next step

- **pick** the configuration of the arm: “left-handed”
- **select** the ratio of the cross section dimension of the link to minimum required by stress analysis: ″4″—if it fails select “3”

  - **do** the design and proceed to the next step

- **find** the accessible region: use Equation 2-4
- **find** the deflection of the tip: use Equation 2-14
- **choose** the type of controller: “PD”

  - **do** the design and finish the process.

Figure 1-2. A Methodology Discovered.

1.3 Motivation

The motivation for this work is *the need for better ways of doing design* in today's manufacturing companies. For many companies this need is the matter of being able to compete and thus, to survive, in today's fast-paced world. Methods and tools are needed that systematically generate better design methodologies with the same speed as new technologies are emerging. Improving design processes based merely on ad-hoc approaches and intuition are no longer adequate. New methods and techniques from the area of Artificial Intelligence in Design are at the stage of maturity where they can provide better alternatives for improving the design methodologies.

The following is a summary of the motivations for conducting this research:

- **Need for Continuous Improvement.** In order to succeed in today's global, competitive market, companies need continuous improvements in their design processes. These improvements should result in expending fewer resources on the design process while achieving better quality and more environmentally friendly products.

- **Need for Rapid Incorporation of New Technologies.** New technologies (e.g., new materials, manufacturing processes, etc.) are emerging into design products increasingly quickly. These new technologies can not only improve the quality of the products, they also can provide better ways of conducting the design process. In this situation we need to incorporate the new technologies and methods into design methodologies as quickly as they appear.

- **Need for Integration.** Integration of multi-disciplinary design is a means to enhance the quality of the design, reduce the cost and the time to market and incorporate environmental considerations into the design of the product. Integration reduces the number of failures and backtracking by facilitating information sharing and thus saves resources. On the other hand, integration provides collaboration between different participants that, as a result, enhances the quality of the design.

- **Need for Design Assistant Tools.** There is a need for design assistant tools that can help designers understand the big picture. It is becoming harder to improve the system performance of engineering devices based merely on advances in individual disciplines. In other words, improvements in individual disciplines alone are not sufficient to affect the improvements in products and processes needed in the future. To achieve higher quality, system-oriented, holistic, multi-disciplinary approaches to the design of engineered systems are needed that consume less resources [NSF 96]. Therefore, design research must produce a scientific foundation for the development of classes of new design methodologies and tools that will address the need for system integration.

- **Need for Concurrency in Design.** "It is well known that concurrent decision making is an important and very desirable component of modern design methodology" [Badhrinath 96]. A concurrent strategy, in contrast to a sequential strategy, carries out some of the problem-solving activities in parallel to each other. As a result, the design process speeds up, because the participants in the design do not have to wait in a line if they can make a contribution.

- **Recent Advancements in Artificial Intelligence in Design.** Recent advances in the application of Artificial Intelligence to design—Multi-agent Design Systems in particular—provide an opportunity to build superior design methodologies. Theories and techniques from Artificial Intelligence that have become available recently enable engineering design researchers to take advantage of the computational power of computers in solving their problems.

1.4 Problem

The problem is the following:

> "*There are no systematic approaches to building design methodologies for integrating different disciplines in multi-disciplinary design so that they collaborate in both contributing to the common goals of the design and sharing resources*".

The following factors contribute to the difficulty of the problem:

- The current approaches for improving design processes are mostly based on intuitive observations followed by incremental changes to the existing methodologies. The current practices of multi-disciplinary design are based on ad hoc strategies for handling the complexities that multiple points-of-view bring to the design process. These techniques solve the problem of complexity at the expense of giving up the potential advantages of diversity. The common methodologies for multi-disciplinary design are based on compromising between different disciplines rather than collaborating between them.

These methodologies do not use a systematic, holistic approach to the problem of multi-disciplinary design and thus these approaches to multi-disciplinary design are not as efficient and effective as they could be.

- "One aspect of design that is usually neglected is that design knowledge is constantly evolving" [Reich 91]. "Designers not only must cope with a complex task, but they must track the evolution of a domain. In this situation, designers determine whether new knowledge is related to the body of existing knowledge, or whether the new knowledge reflects a more fundamental change in technology. The latter may have a large effect on their problem solving behavior" [Reich 91].
- The number of specialists is increasing, while the number of generalists, capable of doing system integration, is decreasing. At the same time, the knowledge burden on the designer keeps increasing as more materials and more options become available [NSF 96].

1.5 Goal and Objectives

The goal of this research is the following:

> *"To synthesize design methodologies for rapid product development, thus reducing time-to-market."*

To achieve the above goal we need to develop approaches and build tools that produce such design methodologies. Since design methodologies are about how to carry out the design process, the emphasis of our research is on improving the process of design rather than the product. In the context of multi-disciplinary design, we are especially interested in design methodologies for integration of different disciplines as that is the key factor in achieving superior design methodologies.

The objectives are to develop design methodologies that provide common knowledge representation schemes and common communication protocols, facilitate design knowledge sharing among participants, provide a cooperative strategy among designers, and provide comprehensive mechanisms for conflict discovery and resolution.

By accomplishing these objectives we systematically overcome the complexities that different points-of-view bring into multi-disciplinary design. Beside that we will be able to take advantage of diverse points-of-view to enhance the quality of design products. Thus, what we are looking for is a twofold solution. First, we wish to overcome the difficulties that participation of different disciplines causes in the design process. Second, we wish to take advantage of the diversity of the participants to design superior products while consuming less resources.

1.6 Approach

We propose a new approach to the problem of producing better design methodologies for multi-disciplinary design based on the integration of different disciplines. The discipline-sequential approach, while poor, is relatively simple. Integration tends to make the design process more complicated. To overcome this complexity, a computer system has been developed representing a knowledge-based model of design in order to automate the simulation of the design process.

The innovative approach is based on simulating the design process using a multi-agent system that mimics the behavior of the design team. The multi-agent system implements a knowledge-based model of design in which highly specialized knowledge from expert sources are applied to synthesize a design [Lander 97]. The multi-agent system activates the pieces of design knowledge when they become applicable. The use of knowledge by agents is recorded by tracing the steps that the agents have taken during a design project.

Many traces are generated by solving a large number of design projects that differ in their requirements. A set of design methodologies is constructed by using inductive learning techniques to generalize the traces generated. These methodologies then can be used to guide design teams through future design projects.

The multi-agent system simulates examples of multi-disciplinary design processes while applying integration principles to the problem. These include common design knowledge representation schemes and common communication mechanisms; design knowledge sharing among participants; cooperative problem-solving strategies among participants; simultaneous design process where possible; and comprehensive mechanisms for conflict discovery and resolution.

The large segments of discipline-specific knowledge are broken into small pieces and are represented in the system by agents. Agent activation is triggered in an opportunistic manner and is unaffected by discipline boundaries. Agents might participate in the design process sequentially or in parallel.

The traces of the agent activations (i.e., knowledge use) during the course of the design process are recorded. The recorded traces consist of orderly patterns of different design tasks that have led to the solution. Some candidate design methodologies are extracted by generalizing the patterns using clustering and inductive learning techniques.

Some of these candidates will be reinforced by solving more examples and accepted as design methodologies for that particular class of problems.

1.7 Significance

The following is a summary of the significant aspects of this research:

- **Radical changes to the design practice for complex systems.** Using methodologies developed by the system allows effective and efficient practices to be used from the start of a project instead of being learned from experience. These new methodologies are radically different from the sequential, discipline-based ones.

- **Reduces time-to-market and saves resources.** To be able to compete, companies not only need continuous improvements in the quality of their products, but they also need to improve the performance of their design and manufacturing processes in order to reduce the cost and the time-to-market. Integration facilitates information sharing among multiple and often contradictory points-of-view. As a result, the number of failures and the amount of backtracking in the design process is reduced, thus saving resources and shortening design time. Integration also provides collaboration between different participants that, as a result, enhances the quality of the design.

- **Incorporates new technologies systematically and quickly.** Agent-based systems allow the addition or deletion of agents. Thus, new knowledge can be added, and old knowledge removed rapidly. Running the system with the new set of agents will result in new traces and thus new and different methodologies. This provides a way to systematically incorporate the new design knowledge into the problem solving process. A system that discovers design methodologies can constantly be fed with the new design knowledge, hence producing design methodologies that are based on the latest technologies.

- **Design process can be biased toward more environmentally friendly products**. As the alternative methods that are built into each agent are tried in a preferential order, and as each method tends to contribute differently towards the final properties of the design, it is possible to bias the design process towards particular properties.

- **Attacks the problem of integration in multi-disciplinary design.** The number of specialists is increasing, while the number of generalists, capable of doing system integration, is decreasing. Also the knowledge burden on the designer keeps increasing due to more materials and more options [NSF 96]. Thus, it is becoming harder to develop methodologies for the integration of multiple disciplines in design.

- **Allows designers to break out of disciplinary confines.** An increasingly specialized technological environment tends to force designers to concentrate on some disciplines more than others. This research allows designers to see the whole design problem.

- **Applies computers to new areas of engineering design.** Computers have mostly been used to support the manipulation and analysis of design product information. This work focuses on the design process, an aspect that has not benefited from computers very much.

- **Incorporates new software methods.** Simulation of design processes based on a multi-agent paradigm is a new area of research that has a high potential for practical as well as theoretical impact on the design of products. The use of multi-agent systems technology is growing rapidly with the development of Java-based systems and agent access across the world-wide web.

- **Incorporates judgement and experience.** As Sobolewski [Sobolewski 96] has stated "System integration, many consider, is an ill-structured problem (the term ill-structured problem is used here to denote a problem that does not have an explicit, clearly defined algorithmic solution). No specific rules have to be followed when doing integration; integration depends totally upon the environment to be integrated. Experienced designers deal with system integration using judgement and experience. Knowledge-based programming technology offers a methodology to tackle these ill-structured integration and design problems".

- **Inter-disciplinary Research.** This work benefits from inter-disciplinary contribution from the state-of-the-art in both Artificial Intelligence and Engineering Design.

- **Most impact in engineering design research.** According to NSF's report on Research Opportunities in Engineering Design [NSF 96], "research areas that will have greatest impact on engineering design over the next 10 years are: Collaborative Design Tools and Techniques, Perspective Models/Methods, System Integration Infrastructure/Tools, and Design Information Support Systems". This work covers all of these areas of research and hence is expected to have a strong impact.

1.8 Outcome and Potential Applications

The potential outcome of this research will be superior design methodologies that facilitate integration and collaboration between different disciplines, conduct design tasks concurrently, and apply to a wide range of design problems. Such methodologies consume fewer resources at design time and provide better quality for the product.

The result of this research will be a generic approach and a set of tools that can be used to synthesize new methodologies for multi-disciplinary design, as well as to analyze and refine current methodologies. These design methodologies will be specialized to collaborate and share resources during the design process. As a result, the time-to-market and the design budget are significantly reduced while the quality of the product is enhanced. This increases profitability and enhances the impact of manufacturing industries on the market.

This approach is specifically aimed at multi-disciplinary design situations where large gains can be achieved by integrated methodologies. In addition, current methodologies can be analyzed for flaws and bottlenecks, and necessary refinements made. New methodologies can be customized so that they are biased toward specific objectives such as manufacturability or being environmentally friendly. By applying this approach the response time for the incorporation of new technologies in design processes will be reduced. Methodologies can be refined as soon as a change occurs in the market or in the organization of the company.

Simulating complex processes using a multi-agent system that approximates the process in order to discover and learn better ways of conducting those processes can be advantageous in many areas such as: Collaborative Product Development, Process Control, Supply Chain Management, Shop Floor Scheduling, and Enterprise Integration. In all of the these areas, the process is the result of an emergent behavior originating from the interaction between the components of the system. Due to the complexity of the behavior, intuition, common sense, and experience are not enough to discover better ways of conducting the process.

In addition, in the areas mentioned the behavior of the components change constantly over time and components are added or removed. For instance, new design approaches may be added to the designers as a result of technological advancements (e.g. new materials). In the supply chain management example, new suppliers may come to the market. New manufacturing methods or machine tools may become available and as a result better scheduling may become possible. Therefore, there is a need for a way to control this dynamism and eventually take advantage of it. Simulating these processes using a multi-agent system is the answer to such a need. One scenario is to let the system run constantly in the background searching for better ways of conducting the design process, scheduling the shop floor jobs, or managing the supply chain.

The design simulation approach, via a multi-agent system, can not only be used as an analysis tool, but also for sensitivity studies in which quantitative and qualitative measurements are formed to show the effect of inputs (requirements/constraints) on outputs (the product attributes). This type of sensitivity analysis is very valuable in areas for which analytical or computational models cannot be built for the physics of the problem.

1.9 Outline of the Dissertation

This section provides an overview of how the rest of the dissertation is organized.

- Chapter 2: While the approach that we propose and develop will be generic, we describe and implement it in the context of robot arm design. This not only provides us with better understanding of the issues, but it also lets us avoid too general or abstract discussions that are difficult to understand. Robot design is a good example of multi-disciplinary design.
- Chapter 3 is an in-depth study of the problem of multi-disciplinary design. We will review the major issues of multi-disciplinary design including integration of different disciplines and concurrency among them. We will then propose a set of strategies that will be incorporated into the approach for synthesizing methodologies for integration in multi-disciplinary design.
- Having described the problem and robot design as an example of the type of problems we are dealing with, in Chapter 4 we describe the knowledge-based model of design. The ingredients of knowledge-based design as well as the strategies that are proposed to be incorporated in the model will be discussed in Chapter 4.
- Chapter 5 will describe the multi-agent system paradigm. The framework that will be presented for the multi-agent design system is a generic architecture that is applicable to all parametric design problems. However, to avoid too much abstract discussion we present the framework in the context of robot design.

- Chapter 6 describes how we are planning to use the multi-agent design system and the results of the experiments conducted to generate design methodologies. This part of the approach is mostly based on machine learning and clustering techniques.

- Chapter 7 describes the building blocks of the multi-agent system for design of a robot arm called *Robot Designer* (**RD**). This chapter is a bridge between the chapters on general approach and implementation of the system. RD will be used to conduct experiments simulating the design process of a robot arm.

- Chapter 8 describes the implementation issues and the contributions of this dissertation that concern building an automated design system based on a multi-agent paradigm.

- Chapter 9 is devoted to 'Design of Experiments', that is how we should set up the experiments so that, while the key features of the problem are covered, the number of experiments are limited to a manageable number.

- Chapter 10 consists of an overview of the collected data from the experiments and then presents the results of processing the data using the approach described in Chapter 6 in order to synthesize the design methodologies.

- Chapter 11 closes the loop by revisiting the goal to see if it has been reached. It summarizes the results of this dissertation and makes some conclusions based on the results presented in Chapter 10. Finally we discuss similar problems that can be tackled with the approach developed in this dissertation in order to analyze or synthesize their emergent behavior.

2 Design of a 2-DOF Robot

2.1 Introduction

A robot can be defined as a technological system, able to replace or assist human in carrying out a variety of physical tasks [L'Hote 83, p. 9]. The official definition for a robot as formulated by the Robotic Industries Association is as the following [Holzbock 86, p. x]:

> "A robot is a programmable, multifunctional manipulator designed to move material, parts, tools, or specialized devices, through variable programmed motions for the performance of a variety of tasks".

The principal functions that a fixed (i.e., not mobile) robot can perform are the following [L'Hote 83, page 11]:

- Handling: loading and unloading, storing.
- Transformation: painting, coating, drilling, machining, filing, buffing, bending, stamping, etc.
- Assembly or Dismantling.
- Fixing: gluing, welding, soldering, riveting.
- Measuring: collecting quantitative information on the structure of the object.

In the following we define some of the terms used in robot design:

- Linkage: The linkage is compromised of links and joints that provide the movement for the end effector of the robot. It also transfers the force required for performing the task. By far the most common robotic joints are the simple hinge joint (or revolute) and the linear sliding joint (or prismatic) [Andeen 88, page 3.5].

- End Effector: The mechanical device that is attached to the end of the linkage to actually do the task is called the end effector. Some examples of an end effector include: grippers and suction pads for gripping, nozzles and torches for arc welding [L'Hote 83, page 12].

- Degrees of Freedom (DOF): The number of degrees of freedom of a robot is equal to the sum of the DOF of the joints. DOF of a joint is the number of independent variables needed to describe the state of the joint. The states of revolute and prismatic joints are expressed by an angle and a displacement respectively. If one desires to have six DOF for a part, it follows that the manipulator holding the part must have at least six DOF. For specialized tasks where arbitrary position and orientation are not required, fewer than six degrees of freedom can be used. The advantage of fewer DOF is decreased cost and complexity of the robot [Andeen 88, page 3.2].

- Workspace: The set of points in space reachable by the end effector is called the workspace of the robot. If the workspace is the set of points on a plane the robot is called a planar robot. A planar robot needs not more than 3 DOF in order to reach to any point in the plane with arbitrary orientation. A planar robot with 2 DOF can reach to any point in the plane but the orientation of the end effector cannot be arbitrary.

- Workload: The maximum load (weight) that the robot can carry is called the workload of the robot.
- Agility: Agility is a qualitative property of a robot that is the "effective speed of execution of prescribed motions" [Rivin 88, page 2].
- Accuracy: This is the accuracy of the robot in positioning the end effector in various DOF. Accuracy should not be mistaken with repeatability, as that is the ability of the robot to repeat the same accuracy over and over again.
- Stiffness: No robot arm is completely rigid. A force or torque on the end effector will always produce some deflection or rotation. More stiffness for the arm reduces the deflection of the end effector.

Many of the above parameters are *interrelated*. Maximum workload, speed, stiffness, and accuracy, might depend on the point of the workspace at which they are being measured [Rivin 88, page 2].

2.2 Robot Design

The manipulator (i.e., robot arm) design process has traditionally been an evolutionary, empirical process. "That is the various industrial manipulator manufacturers have modified their products over time, based on their performance in the workspace" [Depkovich 89]. As a consequence, little knowledge about the potential interactions between disciplines has been accumulated.

Moreover, design methodology for robots is influenced by this fact that there are very large segments of knowledge bounded by different disciplines [Tsai 89]. Therefore, it is difficult to cut these segments of knowledge into pieces so that they can be used in parallel with knowledge from other disciplines. In addition, because the knowledge in different disciplines has been developed independently, it tends to have built-in and not global goals. As a consequence of all of these factors, there is no design methodology that integrates robot design, and it is very difficult to build one.

"Early manipulators were designed by a single individual or small design team (fewer than five engineers). The design procedures were not rigorous or well organized; designers relied on intuition, experience in other fields, and trial and error. As experience with manipulators has increased, design choices have become better defined; in many cases a superior choice has become a standard. Emphasis is increasingly being placed on sophisticated improvements requiring larger, *multi-disciplinary* teams and more structured procedures. A characteristic of the sophisticated design process is increased use of engineering prediction through mathematical modeling, prior to hardware experimentation. The approach to robot design is becoming more like that used in the design of a aircraft, automobiles, computers, and other complex electromechanical systems" [Andeen 88, page 3.1].

Robot design has some characteristics that make it a good candidate domain for implementing the ideas proposed in this thesis:

- Robot design is an example of multi-disciplinary design with participation of Mechanical, Mathematics, Controls, Electrical, and Computer disciplines.
- Size of the problem can be controlled by varying the number of degrees of freedom while the complexity of interaction between disciplines (i.e., being multi-disciplinary) can still be preserved.

- It will be easier to prove the *scalability* of the proposed approach because the major difference between the robot design problem that we will consider in this thesis and more complex robots lies in the complexity of the design methods and interfaces rather than in introducing more disciplines.
- The relationships between the requirements and constraints in robot design is complicated even for a small design problem such as the one that we are to consider.
- Not all design knowledge for robot design is in the form of equations. As a result, we can verify the feasibility of the approach for design methods other than algorithmic methods (see Algorithmic Approach on page 53).
- There is no systematic approach for generating methodologies for the design of robots. Therefore, the area of robot design can gain the most benefit from any advancement in generating better design methodologies.
- Robotics is an area that continues to benefit the most from state-of-the-art technology. It is very crucial that the new technologies be incorporated into the mainstream of the design process as fast as possible. The approach that we are proposing provides a way to do that. Consequently, robot design is a good candidate for implementing the proposed approach.

2.3 Design of a 2-DOF Robot

The type of robot that we consider for implementing the proposed approach is a planar robot with revolute joints and two degrees of freedom (2-DOF). Reducing the number of degrees of freedom to two reduces the computational effort needed in one design project.

As it was explained in the previous section the complexity of interactions between multiple disciplines is preserved in a 2-DOF robot. As a result, the design methodologies that integrate disciplines for a 2-DOF robot can provide useful guidelines to integrate disciplines in robots with higher DOF.

For a planar manipulator, two DOF are necessary for the arbitrary positioning of an object [Rivin 88, page 35]. Because the type of the joints is revolute, there might be more than one combination of joint angles for the arm to reach to a given position.

The disciplines that we consider for the design of a 2-DOF robot are: Kinematics, Structural Mechanics, Dynamics, and Controls.

2.3.1 Design Parameters

The following is a list of design parameters for a planar 2-DOF robot as shown in Figure 2-1. The names in the parentheses will be used later in the development of the knowledge-based computer program that automates the design of the robot.

- *Operational Plane*: This is the orientation of the plane in which workspace of the robot lies in. The orientation of the operational plane affects the structural as well as control design of the robot. In a vertical plane the tension and bending effects on the links vary as the end effector moves within the workspace. In a horizontal plane, on the other hand, tension (or compression) effects are replaced by torsional effects that vary depending on the position of the end effector. The bending effects will stay the same in a horizontal operational plane. Also, in a vertical plane the control system has to compensate for the effect of the acceleration due to gravity. (`operational_plane`).

- *Workspace*: The desired workspace may be defined by a set of points that the end effector of the robot has to reach. A good design covers all the desired points while keeping the area of the accessible region small, Figure 2-1. (`workspace`, `areaaccessible_region_area`).

- *Location of the Base*: The location of the base of the robot relative to the points in the workspace affects the length of the links of the robot and the area of the accessible region. (`base_location`).

- *Link Lengths*: Link lengths influence the design in all three disciplines: kinematics, structural design, and controls. They have direct effect on the size and shape of the workspace, the stress level and deflection of the links, as well as the performance of the control system. (`link1_length`, `link2_length`).

- *Joint Angles*: A robot can reach to a point in the workspace by rotating its joints. Rotation of the joints moves the end effector to the desired point. The minimum and maximum values of the joint angles determine the shape and size of the workspace. (`theta1_array`, `theta2_array`, `theta1_min`, `theta1_max`, `theta2_min`, `theta2_max`).

- *Workload*: Workload is the maximum load that the robot should be able to carry. This value changes for different points in the workspace. That is, when the robot is fully stretched it can carry smaller loads due to the larger deflections and the requirement for more actuating power. We assume that the workload is expressed for this worst case. (`workload`).

- *Link Cross Section Shape and Dimensions*: These two design parameters determine the shape and the size of the cross section of the links. Some shapes might be preferable from manufacturability point-of-view while others might provide more stiffness. (`link_cross_sectional_shape`, `link1_cross_section_dimension`, `link2_cross_section_dimension`, `link1_cross_section_thickness`, `link2_cross_section_thickness`).

- *Material Properties*: The type of material that is used for manufacturing the links affects the structural design as well as the control design. As in all other mechanical devices the most cost effective material with a low density, plus high strength and stiffness properties is desirable. (`material_name`, `material_mass_density`, `material_yield_stress`, `material_elasticity_modulus`).

- *Deflection*: Deflection of the links directly affects the accuracy of the robot in positioning the end effector. We design the robot for the worst deflection, when the arm is fully stretched. (`tip_deflection`).

- *Structural Safety Factor*: The structural safety factor reflects various uncertainties in the design of the structure. It accounts for the difference between the published and actual data for the material's properties as well as any overloading of the structure beyond the nominal values. (`structural_safety_factor`).

- *Control System's Performance*: The speed and accuracy of the control system in positioning the end effector are expressed in terms of settling time and the overshoot. The settling time is the time required for the robot to reach to a destination point and stay within an acceptable range of that point, i.e., acceptable oscillation. The settling time is related to the largest time constant of the control system [Ogata 97, page 151]. The maximum overshoot is the maximum peak in the response of the robot beyond the destination point. The maximum overshoot directly indicates the relative stability of the control system [Ogata 97, page 151]. (`settling_time`, `maximum_overshoot`)

- *Controller Gains*: The gains of the controller show the effort required by the control system to meet the desired speed and accuracy performance. Large control gains mean larger actuators and more accurate sensors that increase the cost of the system. (`proportional_gain1`, `derivative_gain1`, `proportional_gain2`, `derivative_gain2`).

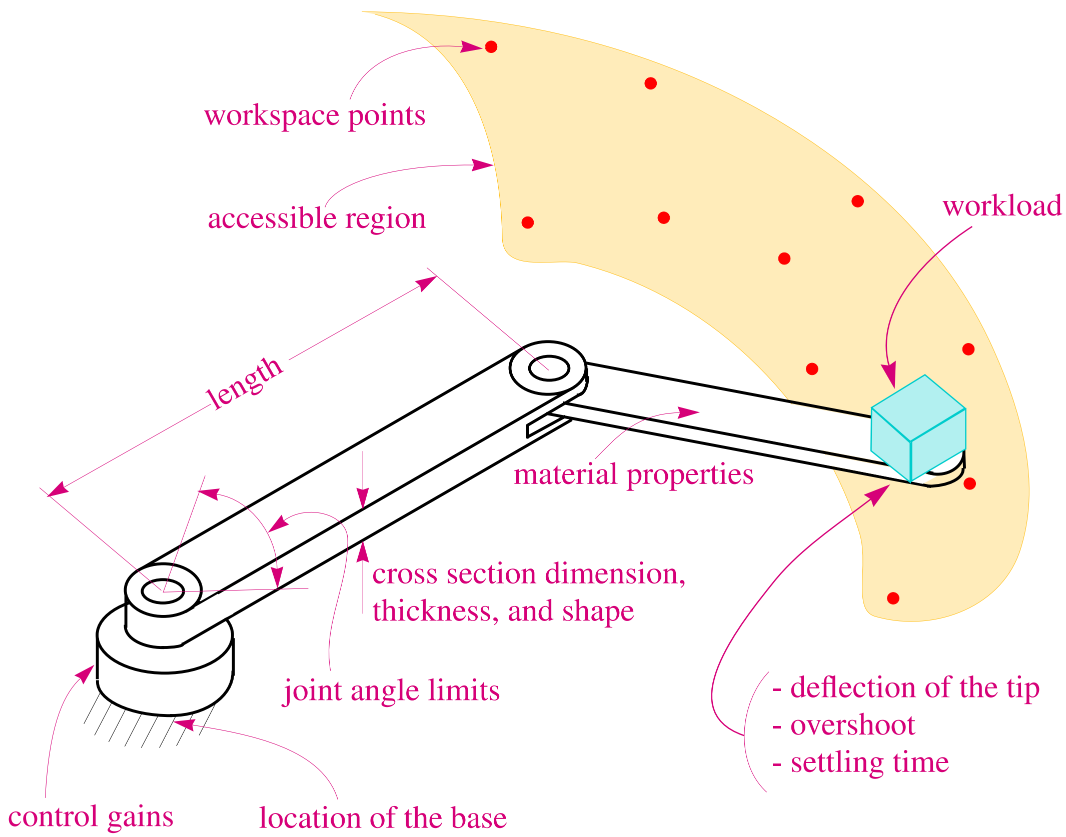


Figure 2-1. Design Parameters of a 2-DOF Planar Robot.

2.4 Kinematics

The kinematics of a robot arm deals with positions, velocities, and acceleration of the manipulator links [Rivin 88, page 34]. "Different kinematic designs means different selection of kinds of joints and lengths of links that compromise the manipulator. The choice of kinematic design is probably the most important choice in the design procedure, and the set of choices is very large. Yet there are only a few guiding principles" [Andeen 88, page 3.2].

In this section we formulate the design procedures that are needed for synthesizing a 2-DOF planar robot with revolute joints. The kinematic parameters that we consider for kinematic synthesize of the robot are:

- the location of the base of the robot, that is, the location of the point to which the first link of the robot is hinged,
- the lengths of the links of the robot, and
- the joint angles.

2.4.1 Kinematic Design Algorithms

We assume that the set of desired points in the workspace that the robot should reach are given. Using the algorithms in [Tsai 81] we can find the location of the base of the robot, the length of the links, and the joint angles for each desired point in the workspace.

The algorithms for calculating the joint angles and the accessible region of a 2-DOF planar robots, as described in [Tsai 81], are only applicable to situations where the first joint angle is between zero and π. The algorithms work for problems in which the workspace points are in the first and second quadrants relative to the base of robot and with reference to a vertical line. However, for points in the third and fourth quadrants the algorithms produce wrong answers for the joint angles. The reason is that the *arc cosine* function always produces a value between zero and π and that leaves out the angles in the third and fourth quadrants.

The following equations are used in [Tsai 81] to find the joint angles for a point (x_i, y_i) within the workspace. The base of the robot is at (x_b, y_b) and the length of the links are

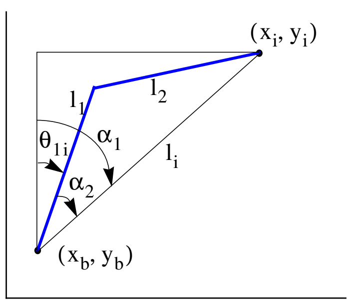
l_1 and l_2 respectively. As a convention we assume that the positive direction for measuring angles is clockwise.

$$l_i = \sqrt{(x_i - x_b)^2 - (y_i - y_b)^2} \qquad \textbf{(2-1)}$$

$$\theta_{1_i} = \operatorname{acos}\left(\frac{y_i - y_b}{l_i}\right) - \operatorname{acos}\left(\frac{l_i^2 + l_1^2 - l_2^2}{2 l_1 l_i}\right) \qquad \textbf{(2-2)}$$

$$\theta_{2_i} = \operatorname{acos}\left(\frac{l_i^2 - (l_1^2 + l_2^2)}{2 l_1 l_2}\right) \qquad \textbf{(2-3)}$$

The above equations can be derived assuming that the point (x_i, y_i) is located in the first quadrant with respect to the base of the robot, (x_b, y_b), as is shown in Figure 2-2:

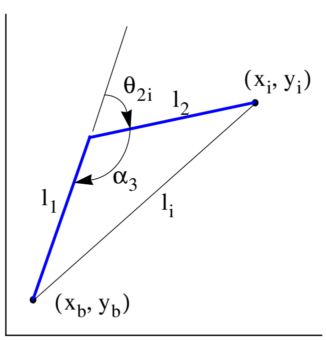


$$l_i = \sqrt{(x_i - x_b)^2 - (y_i - y_b)^2} \qquad \cos\alpha_1 = \frac{y_i - y_b}{l_i}$$

$$l_2^2 = l_i^2 + l_1^2 - 2 l_1 l_i \cos\alpha_2 \qquad \cos\alpha_2 = \frac{l_i^2 + l_1^2 - l_2^2}{2 l_1 l_i}$$

$$\theta_{1_i} = \alpha_1 - \alpha_2$$

$$\theta_{1_i} = \operatorname{acos}\left(\frac{y_i - y_b}{l_i}\right) - \operatorname{acos}\left(\frac{l_i^2 + l_1^2 - l_2^2}{2 l_1 l_i}\right)$$



$$l_i^2 = l_1^2 + l_2^2 - 2 l_1 l_2 \cos\alpha_3$$

$$\cos\alpha_3 = -\left(\frac{l_i^2 - (l_1^2 + l_2^2)}{2 l_1 l_2}\right)$$

$$\cos\theta_{2_i} = \cos(\pi - \alpha_3) = -\cos\alpha_3 = \left(\frac{l_i^2 - (l_1^2 + l_2^2)}{2 l_1 l_2}\right)$$

$$\theta_{2_i} = \operatorname{acos}\left(\frac{l_i^2 - (l_1^2 + l_2^2)}{2 l_1 l_2}\right)$$

Figure 2-2. Kinematics Design Equations.

Equations (2-1) to (2-3) define one set of answer to the joint angles. However, there is a second set of answers for joint angles that is obtained by mirroring the configuration of the arm with respect to l_i. That is, the robot can reach the same point with two different configurations. Therefore, as it is shown in the following figure, there are two solutions to the problem of finding θ_{1_i} in all four quadrants, the first one is $\alpha_1 - \alpha_2$ and the second one is $\alpha_1 + \alpha_2$.

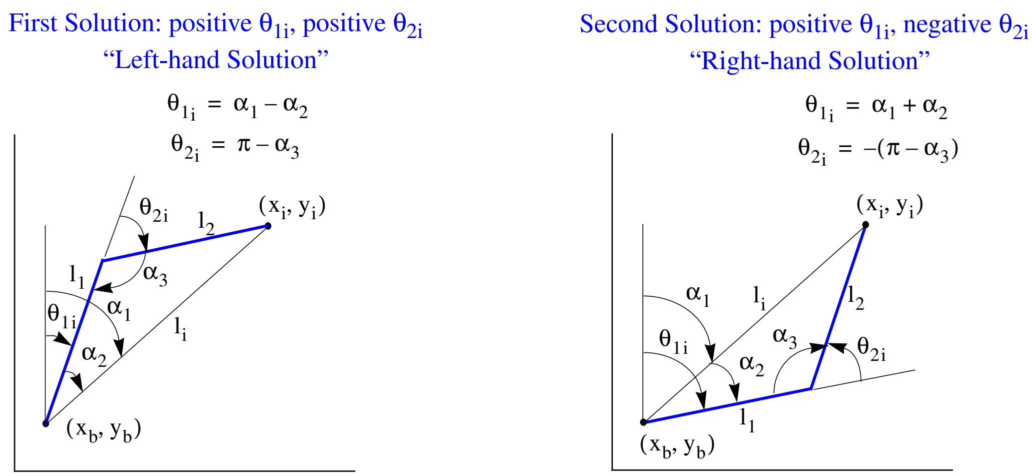


Figure 2-3. Two Solutions for Kinematics Design.

As we will see in Chapter 7, having two possible solutions for kinematics provides us with an opportunity to define two different design approaches. If the first design approach that correspond to "left-hand solution" fails to satisfy design constraints, the second design approach will be tried that may resolve the constraint violation.

Also, please note that it is possible to combine these two solutions for a set of points in the workspace so that some of the points are reached by joint angles from the first solution and the rest from the second solution. This mixture of solutions may result in a different accessible region area than an only “left-hand solution” or an only “right-hand solution”.

2.4.2 Calculating Accessible Region Area

To make the above equations applicable to the third and fourth quadrants as well we had to modify them. The modification of the equations is discussed in Appendix A. In the following we present the equations that are applicable to the general case.

The accessible region of a 2-DOF planar robot can be calculated based on link lengths, the angle swept by the first link (θ_{1i} angles), and the maximum and minimum of $\cos\theta_{2i}$ angles as shown in Figure 2-4:

$$A = l_1 l_2 \theta_{1,\,sweep}[(\cos\theta_{2i})_{max} - (\cos\theta_{2i})_{min}]$$

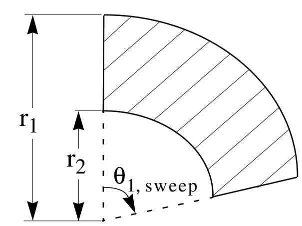


Figure 2-4. Calculating the Accessible Region of A 2-DOF Robot.

2.5 Structural Mechanics

Structural design of a robot deals with the stiffness and the deflection of the structural components (e.g., the links), as well as the natural frequency and damping characteristics of the robot’s structure. In this thesis we only consider the static aspects of structural design of a robot, stress analysis and deflection of the links. In addition, we assume the following:

- The operational plane that the arm moves is horizontal,

- Masses of the links are concentrated at the end of the links,
- Links are tubular with square or circular cross sections,
- Only the bending of the arm is considered in structural design of the robot.

2.5.1 Structural Design

We design the links based on the stress analysis and then check for the deflection to be within the allowed range.

2.5.1.1 Stress Analysis

The bending stress of the link can be calculated from Equation (2-4):

$$\sigma_b = \frac{M\frac{d}{2}}{I} \tag{2-4}$$

where σ_b is the bending stress, M is the bending moment, d is the dimension of the cross section of the link, and I is the moment of inertia of the cross section that can be calculated from Equation (2-5):

$$I = c_1 d^4[1-(1-2r)^4] \tag{2-5}$$

$$c_1 = \frac{\pi}{64} \text{ : for circular links,} \quad c_1 = \frac{1}{12} \text{ : for square links}$$

$$r = \frac{t}{d}$$

After substitution of (2-5) into (2-4)we will have:

$$\sigma_b = \frac{M\frac{d}{2}}{c_1 d^4[1-(1-2r)^4]} \tag{2-6}$$

The maximum bending moments for each link can be calculated from Equations (2-7):

$$\begin{aligned} M_2 &= (m_2 + m_\iota)gl_2 \\ M_1 &= (m_1 + m_2 + m_\iota)gl_1 + (m_2 + m_\iota)gl_2 \end{aligned} \tag{2-7}$$

where m_1, m_2, and m_ι are the mass of link 1, mass of link 2, and the workload, respectively. The masses of the first and second links can be calculated from:

$$m = c_2[r(1-r)]d^2\rho l \tag{2-8}$$

$c_2 = \pi$: for circular links, $\quad c_2 = 4$: for square links

where ρ is the mass density of the material of the link. Substitution of Equations (2-7) and (2-8) in Equation (2-6) gives:

$$\begin{aligned} {d_2}^3 &= A + B{d_2}^2 \\ {d_1}^3 &= C + D{d_1}^2 \end{aligned} \tag{2-9}$$

where:

$$A = \frac{m_\iota g l_2}{\{2c_1\sigma_{all}[1-(1-2r)^4]\}}$$

$$B = \frac{\rho l_2^2 c_2 r(1-r)g}{\{2c_1\sigma_{all}[1-(1-2r)^4]\}} \tag{2-10}$$

and,

$$C = \frac{(m_2 + m_t)(l_1 + l_2)g}{\{2c_1\sigma_{all}[1 - (1 - 2r)^4]\}}$$

$$D = \frac{\rho l_1^2 c_2 r(1 - r)g}{\{2c_1\sigma_{all}[1 - (1 - 2r)^4]\}} \qquad \textbf{(2-11)}$$

and,

$$\sigma_{all} = \frac{\sigma_y}{n} \qquad \textbf{(2-12)}$$

where σ_{all} is the allowable stress, σ_y is the yield strength of the material, and n is the structural safety factor. We can find a closed form solution to Equations (2-11) as follows:

if: $\quad d^3 = \alpha + \beta d^2$

$$d = \gamma + \frac{\beta^2}{9\alpha} + \frac{\beta}{3} \quad \text{and,} \quad \gamma = \left(\sqrt{\frac{\alpha\beta^3}{27} + \frac{\alpha^2}{4}} + \frac{\beta^3}{27} + \frac{a}{2}\right)^{1/3} \qquad \textbf{(2-13)}$$

Please note that the dimension of the first link cannot be determined without knowing about the dimension of the second link. The first link carries the mass of the second link too and the mass of the second link depends on the dimension of the cross section of the link, Equation (2-8).

2.5.1.2 Deflection Analysis

In this section we calculate the deflection of the tip of the robot arm assuming all other structural parameters are known. The deflection of the tip is composed of the deflection of the second link plus deflection of the first link due to the bending of the links plus the rigid body rotation of the second link due to the deflection of the first link, Equation (2-14):

$$\delta_{tip} = \delta_{bending,\,1} + \delta_{bending,\,2} + \delta_{rigid,\,1,\,2} \tag{2-14}$$

$$\delta_{bending,\,2} = \frac{(m_2 + m_\iota)gl_2^3}{3E_2I_2} \tag{2-15}$$

$$\delta_{bending,\,1} = \frac{(m_1 + m_2 + m_\iota)gl_1^3}{3E_1I_1} + \frac{(m_2 + m_\iota)gl_2l_1^2}{2E_1I_1} \tag{2-16}$$

In Equations (2-15) and (2-16) E is the modulus of elasticity of the material and I is the moment of inertia of the cross section of the link as calculated in Equation (2-5).

$$\delta_{rigid,\,1,\,2} = \left[\frac{(m_1 + m_2 + m_\iota)gl_1^2}{2E_1I_1} + \frac{(m_2 + m_\iota)gl_1l_2}{E_1I_1}\right] \times l_2 \tag{2-17}$$

2.6 Dynamics and Controls

The dynamic equations of a 2-DOF planar robot can be derived based on the procedure in [Craig 86, page 173]. Ignoring the nonlinear terms we will have Equations (2-18):

$$\begin{aligned} \tau_1 &= m_2l_2^2(\ddot{\theta}_1 + \ddot{\theta}_2) + m_2l_1l_2(\cos\theta_2)(2\ddot{\theta}_1 + \ddot{\theta}_2) + (m_1 + m_2)l_1^2\ddot{\theta}_1 \\ \tau_2 &= m_2l_2^2(\ddot{\theta}_1 + \ddot{\theta}_2) + m_2l_1l_2(\cos\theta_2)\ddot{\theta}_1 \end{aligned} \tag{2-18}$$

where τ_1 and τ_2 are the torques applied to the first and second joints by the actuators. θ_1 and θ_2 are the first and second joint angles. The transfer functions of the system can be derived using the Laplace transform and assuming zero initial conditions as shown in Equations (2-19):

$$\frac{\dot{\theta}_1}{\tau_1} = \frac{1}{I_{eq_1}s}$$
$$\frac{\dot{\theta}_2}{\tau_2} = \frac{1}{I_{eq_2}s} \tag{2-19}$$

where I_{eq_1} is the equivalent moment of inertia of the arm as seen by the actuator of the first joint and I_{eq_2} is the moment of inertia of the second link seen by the actuator of the second joint that can be calculated using Equations (2-20):

$$I_{eq_1} = m_2 l_2^2 + 2m_2 l_1 l_2(\cos\theta_2) + (m_1 + m_2) l_1^2$$
$$I_{eq_2} = m_2 l_2^2 \tag{2-20}$$

2.6.1 Position Control Design

The type of controller that we consider for the robot is a position control scheme that is shown in Equation (2-21):

$$\tau = G(\theta) + K_p E - K_d \dot{\theta} \tag{2-21}$$

This controller does not force the manipulator to follow a trajectory, but moves the manipulator to a goal point along a path specified by the manipulator's dynamics, and then regulates the position there [Craig 86, page 15]. This is a controller of type PD (proportional plus derivative) that is shown in the block diagram of Figure 2-5.

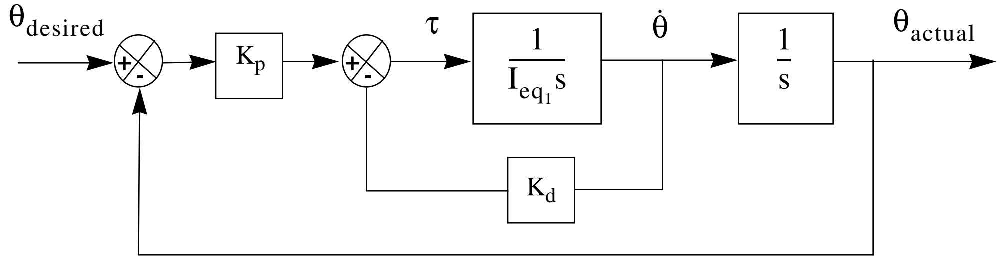


Figure 2-5. A PD Controller for the Robot.

The closed-loop transfer function can be found by simplifying the block diagram of Figure 2-5.

$$G(s) = \frac{\frac{K_p}{I_{eq}}}{s^2 + \frac{K_d}{I_{eq}}s + \frac{K_p}{I_{eq}}} \tag{2-22}$$

Assuming that ω_n is the desired natural frequency and ς is the desired damping ratio for the closed loop system, the controller gains can be found from Equations (2-23):

$$K_p = \omega_n^2 I_{eq}$$

$$K_d = 2\varsigma\omega_n I_{eq} \tag{2-23}$$

The closed loop system of Equation (2-22) is always stable because all the coefficients of the denominator are positive quantities. Therefore, the only constraint on the values of the gains is imposed by the size and power of the actuator and the resolution of the tachometer that measures the angular velocity of $\dot{\theta}$.

In this chapter we described the design knowledge that is used in kinematic design, structural design, and control design of a 2-DOF robot. Robot design is a good example of multi-disciplinary design. The design of 2-DOF robot has a small number of design parameters (compare to a 6-DOF industrial robot). However, it preserves the complexity of multiple disciplines in the design.

The next chapter is an in-depth study of the problem of multi-disciplinary design. We will review the major issues of multi-disciplinary design including integration of different disciplines and concurrency among them. We will then propose a set of strategies that will be incorporated into the approach for synthesizing methodologies for integration in multi-disciplinary design.

3 Multi-disciplinary Design

3.1 Introduction

Multi-disciplinary designs are very complex processes that consume a lot of time, money, expertise, information and other resources. Complexity originates from the diversity of disciplines that each possess a different point-of-view regarding the design problem. As a result, different disciplines adopt different and often contradictory goals and constraints, while they have to share resources such as budget, time, expertise, and information.

Although diversity is the source of complexity, it can be turned into a source of advantages. Having representation from different functional areas in multi-disciplinary teams is beneficial to the design [Dowlatshahi 97]. Diversity in disciplines brings multiple sources of knowledge, problem-solving techniques and expertise to the design process and also modularizes the design knowledge. As we will discuss in the next chapter, modularity in the design knowledge produces some problems by creating boundaries around segments of knowledge. However, at the same time, modularity helps to manage the vast amount of required knowledge by partitioning it into different fields of expertise.

In order to take advantage of diversity in multi-disciplinary design, different disciplines should collaborate with each other in adopting common goals, sharing resources, exchanging information, and resolving conflicts. “Engagement of different cooperative agents in the design problem solving can solve the design problem faster than either a single agent or the same group of agents working in isolation from each other. As a matter of fact, that cooperation leads to improvements in the performance of a group of individuals underlies the founding of the firm, the existence of scientific and professional communities, and the establishing of committees charged with solving particular problems" [Clearwater 92]. Such a collaboration strategy between different disciplines is called integration of multidisciplinary design.

3.2 Survey of Related Work

"Engineering design should always be thought of as a multi-disciplinary activity. Indeed, it must include consideration of all disciplines: it is *omnidisciplinary*" [Hazelrigg 96]. Recently, there has been increasing recognition that multi-disciplinary design is important. A large amount of very good research has been focused on Multi-disciplinary Design Optimization (MDO) [Sobieszczanski-Sobieski 96]. MDO tries to produce an effective product by recognizing and using appropriate combinations of parameters to be controlled and optimized by the designer.

MDO is based on mathematical modeling of the design problem in terms of objective functions and then their minimization. The problem is that a mathematical model for the design product is not available until the very end of the process, when the conceptual and embodiment design are complete. Additionally, for many cases a mathematical model cannot comprehensively include all design concerns and ignores those characteristics that cannot be mathematically modeled. In our approach, MDO techniques are employed whenever they are applicable.

The numerical methods for disciplinary and multi-disciplinary optimization would be considered as 'design methods' in this work. The contrast between our work and MDO is that we are concerned about moving towards an optimal design process while MDO is concerned about optimization of the product. Also, MDO needs access to a mathematical model of the system, while development of complete mathematical models for the type of problems we are working on is not possible.

While a key part of the MDO process is the use of appropriate decompositions, there has been less attention paid to the sequence of design activity that caused the process to arrive at the point where optimization can be done (see the discussion of decomposition in [Sobieszczanski-Sobieski 96]). Problem decompositions are influenced by dependencies between design decisions [Gebala 91] [Liu 94] [Kusiak 93] [Rogers 96]. Most existing research into decomposition assumes that problem decompositions are not affected by prior design decisions.

However, in multi-disciplinary design problems the values of design parameters may determine what design method will be employed, as methods may have applicability conditions. As different design methods may introduce different dependencies, dependency chains, and potentially problem decompositions, can be dynamically determined. This means that the sequencing of design tasks can also be dynamically determined.

Some approaches to the support of multi-disciplinary design problems provide some user interaction to help determine what task sequence will be used [Kroo 88] [Kroo 90] [Hale 96] [Wujek 96]. However, while Multi-disciplinary Design problems often require the user's investigation of design trade-offs, for each problem and related set of requirements, there are a number of common design task sequences (design flows) that are used. Such sequences form the basis of a design methodology for that problem or class of problems.

This work is also distinguished by the requirement that results of the discovery process be well integrated, and, if possible, concurrent. For that to happen, fine-grained tasks are needed, as opposed to the large grained tasks (often based on existing software) used by research such as [Hale 96] and [Woyak 95].

This research acknowledges that not all design knowledge is based on equations—some is qualitative, experiential, and heuristic. Our agent-based approach can accommodate such knowledge. Lander [Lander 97] provides a detailed review of this field, while other work on multi-agent systems in Concurrent Engineering is reported in a special issue of the CERA journal [Brown 96-b].

3.3 Characteristics of Multi-disciplinary Design

The following characteristics of multi-disciplinary design contribute to the problem of producing better design methodologies. They are the most important barriers to integration of different disciplines:

3.3.1 Different Points of View

The notion of 'points-of-view' is essential in multi-disciplinary design. It is due to the difference in points-of-view of multiple disciplines that makes multi-disciplinary design interesting and challenging.

3.3.2 Departmentalization of Disciplines Over Time

Another challenge is that different disciplines have developed their own terminology and conceptualizing of the world separately from other disciplines because of the historical facts. As a consequence, there are not many techniques for understanding and collaboration among different disciplines. There could be different views among members of a team but at the same time there could be well defined terminologies and conceptualizing that allows the members to collaborate with each other in achieving a common goal.

Different disciplines conceptualize and represent their knowledge differently from the others. Boundaries are built around disciplines with special internal languages and no means for communicating with the outside world. As a consequence, it becomes difficult for the participants to communicate their points-of-view, let alone collaborate with each other or resolve their conflicts. For example, consider the disciplines involved in the design of robot manipulators and the way they formulate the problem in terms of different concepts:

- *kinematics*: length, angle, coordinate system, velocity, acceleration;
- *mechanics*: strength, deflection, power transmission system;
- *dynamics*: force, torque, vibration;
- *controls*: stability, time constant, positioning accuracy.

3.3.3 Built-in Goals

Different disciplines tend to accumulate knowledge independently. As a result, they tend to have built-in goals that are often in conflict with global goals of the design. Ignoring the conflicts between local and global goals leaves the behavior of the system to the dynamics of self-design as determined by the structure of the system itself [Forrester 69]. For example, knowledge about the design of robots has been developed based on steady discipline-by-discipline contributions over time [Craig 86] [Andeen 88] [Rivin 88]. "The manipulator design process has traditionally been an evolutionary, empirical process. That is, the various industrial manipulator manufacturers have modified their products over time, based on their performance in the workspace" [Depkovich 89]. For instance, consider how the disciplines involved in robot manipulator design adopt different goals:

- *kinematics*: covering the workspace,
- *mechanics*: keep the deflection of the tip low,
- *dynamics*: keep the vibration low,
- *controls*: increase the speed and positioning accuracy.

3.3.4 Focused Expertise of the Disciplines

The points-of-view of different disciplines are sharply limited due to highly focused expertise in their fields. As a result, integration techniques that are solely based on disciplinary knowledge become fragile, because they fail to apply as soon as the conditions change slightly.

3.3.5 Need for Broad Range of Expertise

The required knowledge for doing multi-disciplinary design is distributed among different fields of science and engineering so that no single person is able to possess all the required expertise. "Large-scale engineering projects typically involve up to 300 different specialty design firms, suppliers, and contractors. Therefore, many different types of professionals must interact and communicate with one another, which in many cases can result in conflicts” [Pena-Mora 95]. As a consequence, a broad range of expertise needs to be combined in order to develop an integrated design methodology.

3.3.6 Disciplinary Design in Big Chunks

Disciplinary designs are processed in large segments that make integration very difficult because they hide valuable information that is necessary for integration (such as decisions that may lead to conflicts) from the rest of participants. Also, considering the iterative nature of design, it is costly and time consuming to repeat disciplinary designs in big chunks. This is because in every iteration the designers have to redo the big chunks of disciplinary design in their entirety.

3.3.7 Complexity of Interactions

The interactions between different disciplines are complex because they are multifaceted, meaning that multiple disciplines might be interested in one parameter at the same time. Also, the number of interactions is very large and they may change depending on the type of the problem.

3.3.8 Large Number of Iterations

Departmentalization increases the number of conflicts between disciplines hence increasing the number of iterations required for finding a solution. Large number of iterations consumes more time and other resources in the design process.

3.3.9 Counter-Intuitive Behavior

"It has become clear that complex systems are counter-intuitive, that is they give indications that suggest corrective action which will often be ineffective or even adverse in its results" [Forrester 69, p. 1]. Multi-disciplinary designs are a type of complex systems with counter-intuitive behavior. "Intuition fails to hold true when the constraints become active; it is then that the real interaction among design groups occurs" [Wujek 96, p. 370]. Therefore, integration in multi-disciplinary design cannot be done based on intuitive approaches and comprehensive studies are needed in order to develop solutions to the integration problem.

3.4 Integration in Multi-disciplinary Design

Integration is important in multi-disciplinary design because different disciplines have different requirements and constraints to satisfy. In the robot design problem, for instance, kinematics is required to cover the desired workspace while minimizing the accessible region. On the other hand the controls discipline is required to minimize the rise time and the overshoot while keeping the control gains within a practical range.

On the surface these requirements seem independent of each other. However, the sets of design parameters that are affected by the requirements or affect the constraints may have common elements. Integration means bringing all of the disciplines that are affected by shared design parameters together in order to negotiate and assign values to shared parameters.

Integration makes it possible for different disciplines to participate simultaneously in the process of assigning values to these shared parameters. As a result, any possible conflict on the assigned value is discovered and resolved immediately. Besides, with simultaneous participation of disciplines there will not be any “lead discipline” that has earlier participation in the design process. “Lead disciplines” tend to dominate the decision making and therefore to favor their own requirements. This simultaneity in decision making is called concurrency.

Integration in multi-disciplinary design becomes even more complicated when there are *inter-disciplinary constraints* in the design process that should be satisfied. Inter-disciplinary constraints are a subset of design constraints that cannot be evaluated by a single discipline independent from other disciplines. For instance, the total cost or weight of a robot cannot be evaluated by a single discipline in contrast to the accessible region of the robot arm and the time constant of the control system that can be evaluated independently by kinematics and controls disciplines respectively.

Similarly, the set of design parameters can be categorized into inter-disciplinary and disciplinary. For instance, the length of the links in the robot design example are inter-disciplinary design parameters because they affect constraints in both kinematics and controls disciplines. On the other hand, the modulus of elasticity of the material used for the links is only of concern to structural design, hence it is a disciplinary parameter.

In this chapter we reviewed the major issues of multi-disciplinary design including integration of different disciplines and concurrency among them. In the next chapter we describe the knowledge-based model of design, its ingredients, as well as the strategies that are proposed to be incorporated in the model.

4 Knowledge-Based Design

"Knowledge-based systems provide a means by which different design methods and methodologies can be efficiently studied, compared, and evaluated. Though knowledge-based systems have not yet been used in this way except in ad-hoc fashion, they will be. When the design research community finally gets the time, freedom, support, courage, and vision (there exist plenty of intellect) to attempt development of an empirical science of engineering design, then the knowledge required, and different methods and methodologies, will be subject to study via knowledge-based systems" [Dixon 95].

"Curiously, there is resistance to the knowledge revolution, especially in engineering. This resistance comes from universities and research organizations that do not have good mechanisms for evaluating new disciplines, such as information technology. Often, engineering faculty who evaluate new disciplines base their judgments on comparisons with evaluating traditional, engineering science-based methodologies. When these analogies do not align, judgments tend to be harsher than otherwise necessary" [Sriram 98].

4.1 Introduction

“Design is a process that constructs a description of an artifact, process or instrument, that satisfies a (possibly informal) functional specification, meets certain performance criteria and resource limitations, is realizable, and satisfies criteria such as simplicity, testability, manufacturability and reusability. The design process itself may also be subject to certain restrictions such as time, human power, and cost” [Sriram 98].

The process of designing an artifact can be usefully viewed as a search of a multi-dimensional space of possible designs. The dimensions of such a space are the parameters that describe the artifact (e.g., the properties of the individual parts and the structural relationship between the parts) [Mittal 92]. Therefore, each point in such multi-dimensional space is a possible design and designing is to search for a point that satisfies the constraints.

Design is an ill-structured problem that demands a wide variety of knowledge sources, such as heuristic knowledge, qualitative knowledge, and quantitative knowledge. Engineering design involves a large number of components and the interaction of multiple technologies. Design decisions (e.g., selecting the components included in the product) are made in a multi-stage, iterative, and collaborative process, starting from customer requirements, through conceptual design, to detailed design. As a result, considerable communication and coordination is required between participants with varied domain specific backgrounds. Hence engineering design is a knowledge-intensive collaborative process [Sriram 98].

4.2 Design Process

"A design *process* is the series of activities by which the information about the designed object is changed from one information state to another. That is, a design process solves, or resolves, a design problem" [Dixon 95]. To improve the design process we need prescriptions that advocate how design should be done in particular circumstances [Dixon 87].

Models of design process can be categorized into three categories: descriptive, prescriptive, and computational. The descriptive models simply describe the sequence of activities that typically occur in designing. Prescriptive models attempt to prescribe a better or more appropriate pattern of activities [Cross 89, page 19]. A computational model expresses a method by which a computer may perform the design process [Dixon 87].

4.2.1 Models of Design Process

4.2.1.1 Knowledge-based Design

"Engineering design can only be knowledge-based, or else it is guesswork.... And if an intellectual task is knowledge-based, then the knowledge employed can be explicitly identified, organized, codified, studied, and experimented with using knowledge-based computer systems as an experimental apparatus" [Dixon 95].

A knowledge-based design paradigm applies highly specialized knowledge from expert sources to the synthesis or refinement of a design or a design process [Lander 97]. The development of knowledge-based systems for design, especially of mechanical systems, is increasing. The expectation is that these computer systems can improve the quality of design and shorten the design time [Mittal 92].

A knowledge-based view of design is more appealing to human designers than other methods such as optimization. Another advantage is that certain tasks are easier to be modeled in a knowledge-based systems than using a mathematical model[Coyne 90, page 30].

4.2.1.2 Systems Science Approach

In the systems approach the need for a close study of the problem environment is emphasized. Such a study takes place before specifying design requirements. Kannapan [92] describes four different theories that are used to model a system in Systems Science. The following are the models for design that would result from those theories:

- Black Box Theory: This models the design as the mapping of the requirements to the design descriptions with respect to the environment.
- State Theory: This models the design process by a vector of characteristic attributes representing the internal state of the process. The design process will be transition of current state to the next one.
- Component Integration Theory: This models the design by decomposing it into components whose input-output mappings are known. The behavior of the system is derived from the interaction between these components.
- Decision Theory: In this approach all activities of modeling, design, and analysis are modeled as decision making activities. Each decision is made in a systematic way, taking into account dependencies between actions [Kannapan 92].

4.2.1.3 Problem Solving Approach

The problem solving approach seeks to reduce a design problem to sub-problems recursively until sub-problems solutions are directly known. Subproblem interdependencies are formulated as constraints to be satisfied [Kannapan 92].

Due to the inter-dependencies between subproblems, a search process is needed to determine in what order the subproblems should be solved. “The search is formulated in terms of problem states: Given an initial state, the attributes of a goal state, and a set of state change operators, problem solving involves determination of a sequence of operators that transform the initial state to a goal state. The path of search may be controlled and constrained in several ways (e.g., breadth first, depth first, heuristic, dependency-directed backtracking)” [Kannapan 92].

4.2.1.4 Algorithmic Approach

“The algorithmic approach views design as a finite deterministic process. Cases where the entire design process is algorithmic are rare. However, parts of most design processes are algorithmic, especially where the emphasis is on numerical analysis and optimization. Optimization techniques apply where design problems can be formulated in the standard mathematical form of objective functions and constraint equations” [Kannapan 92].

4.2.1.5 Axiomatic Approach

“The key concepts of axiomatic design are: the existence of domains, the characteristic vectors within the domains that can be decomposed into hierarchies through zigzagging between the domains, and the design axioms. The design world of the axiomatic approach is made up of domains. There are four domains: the customer domain, the functional domain, the physical domain, and the process domain. Axioms are general principles or self-evident truths that cannot be derived or proven to be true except that there are no counter-examples or exceptions" [Suh 95].

Suh identifies two axioms by examining the ubiquitous, common elements present in good product, process, or system designs [Suh 95]:

- **Axiom 1:** The Independence Axiom: The independence of *Functional Requirements* (FR) must be always maintained, where FRs are defined as the minimum number of independent requirements that characterize the design goals.
- **Axiom 2:** The Information Axiom: Among those designs that satisfy the Independence Axiom, the design that has the highest probability of success is the best design. The design with minimum information content has the highest probability of success.

4.2.2 Classes of Design

Many different classifications have been proposed for design including: Preliminary, Conceptual, Functional, Innovative, Creative, Routine, Embodiment, Parametric, Detailed, Redesign, Non-routine, and Configuration [Brown 96-c].

Brown and Chandrasekaran have proposed the following three classes for design [Brown 89, page 32]:

- Class 1 Design (Creative Design): In this class of design neither the knowledge sources nor the problem-solving strategies are known in advance. The average designer in industry will rarely, if ever, do class 1 design. This type of design often leads to a major invention or completely new products.

- Class 2 Design (Innovative Design): What makes this type of design class 2 and not class 1 is that the knowledge sources can be identified in advance, but the problem-solving strategies cannot. This type of design will require different types of problem-solvers in cooperation and will certainly include some planning.

- Class 3 Design (Routine Design): The choices at each point in the design process may be simple, but that does not imply that the design process itself is simple, or that the components so designed must be simple. “We feel that a significant portion of design activity falls into this class” [Brown 96-c].

The main point, which is often overlooked, is summarized in Table 4-1, [Brown 96-c]:

Table 4-1. Classes of Design.

Class	Knowledge Source	Problem-Solving Strategies
Creative	Not Known	Not Known
Innovative	Known	Not Known
Routine	Known	Known

Design processes can be classified along a different axis that describes what sort of decisions are being made. “One end of the axis represents conceptual design and the other end represents parametric design. This axis shows the abstractness of the decisions being made, and reflects the notion that more constraints are added to the solution as the design activity progresses. For many design problems, the Conceptual-Parametric axis represents the flow of time during the design activity, with earlier decisions falling toward the left and later decisions falling toward the right” [Brown 96-c].

Consider the space that would result from the two orthogonal axes described above for classes of design, Figure 4-1.

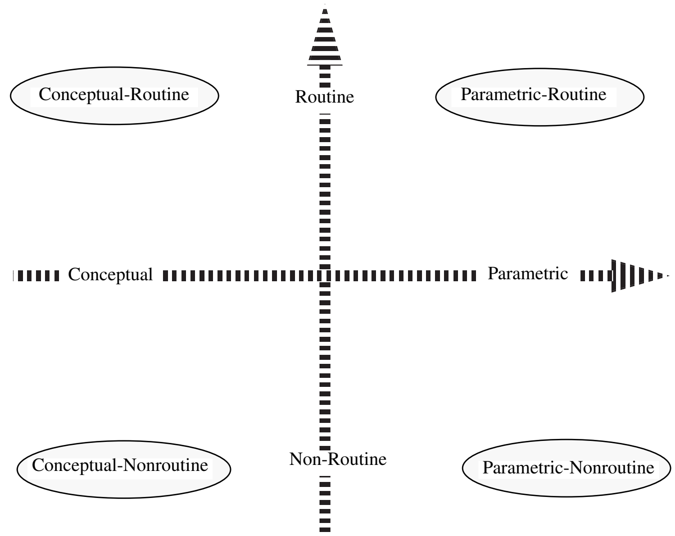


Figure 4-1. Classification of Designs.

A description of each four resultant categories can be found in [Brown 96-c]. The type that we are interested in is the Parametric-Nonroutine class. This is the class of the robot design problem that we have considered in Chapter 2 for the implementation of the proposed approach.

"At the Parametric-Nonroutine point the designer is deciding values for parameters (parametric), and does not have any well-formed approach to making them". The designer does not know about how to go about deciding the values of parameters. "This would result in non-routine behavior, such as analyzing the dependencies between the parameters in order to determine the appropriate methods or equations" [Brown 96-c].

The robot design problem presented in Chapter 2 is parametric because the designer is deciding values for parameters. The problem is non-routine because substantially different design knowledge might be used in different design problems. The following characteristics make the aforementioned design problem non-routine [Brown 89, p. 33]:

- The problem is decomposed into subproblems. Design of a new robot does not involve new discoveries about decomposition: the structure of the robot is well known.
- Robotics is a field that constantly undergoes major technological changes, and routine methods of design for some of the robot's components may no longer be applicable.
- The failure analysis is quite complex (see "Factors Contributing to the Complexity of Backtracking" on page 146). To recover from failure, the designer might engage in a complex dependency-directed backtracking process.

4.3 Design Methodology

The definition of "methodology" in Webster's Dictionary is:

> *A body of methods, rules, and postulates employed by a discipline: a particular procedure or set of procedures; the analysis of the principles or procedures of inquiry in a particular field.*

A design methodology is a scheme for organizing reasoning steps and domain knowledge to construct a solution [Dasgupta 1989]. It provides both a conceptual framework for organizing design knowledge and a strategy for applying that knowledge [Sobolewski 96].

"A design *methodology* is a prescription for a process intended to solve a specified design problem type. A design *method* is a procedure for implementing a step in a methodology" [Dixon 95].

Design methodology is different from the *Design* itself. Design is primarily concerned with the question of '*what* to design' to satisfy some specified need. Design methodology, however, is primarily concerned with the question of '*how* to design'. Good methodologies allow us to better model, teach and aid/automate the finding of solutions to '*what* to design'. "Design methodology is thus a vehicle for the evolution of design activity from an art or skill to a science. Insofar as design activity (as in industrial product design) is the natural testing ground for design methodology, we propose that the term *engineering design research* or *research in engineering design* be reserved for research in design methodology where the object is not a product but the knowledge of how to design products" [Kannapan 92].

The solution to the integration problem, whatever it will be, is to be represented in the form of a group of design methodologies. A design methodology is a scheme for organizing reasoning steps and domain knowledge to construct a solution. It provides both a conceptual framework for organizing design knowledge and a strategy for applying that knowledge [Sobolewski 96] (also refer to Appendix A). Therefore, the original problem of integration reduces to *development of design methodologies for integration of multi-disciplinary design problems*. A complete design methodology for integration provides knowledge for:

- how to evaluate partial designs from all participants' points-of-view,
- what to do next, considering proposals from all participants,
- how to resolve conflicts by negotiating with all participants,

- common representation schemes for design knowledge and common communication mechanisms for interactions,
- how to facilitate design knowledge sharing among participants,
- a cooperative strategy among participants (via control),
- how to conduct the design process in a simultaneous manner,
- comprehensive mechanisms for conflict discovery and resolution.

In this project the problem of robot manipulator design is considered as an example of a multi-disciplinary design process. Hence, the problem to be investigated reduces further to *development of design methodologies for integration of different disciplines in robot manipulator design.*

Design methodology is the answer to the question of "how to design" [Kannapan 92]. A design methodology provides methods for decomposing design problem into sub-problems, ordering the design tasks, generating partial designs, composing more complete designs from partial designs, evaluating partial designs, and discovering and resolving conflicts.

A design methodology is a problem-solving model at an abstract level. "A problem-solving model is a scheme for organizing reasoning steps and domain knowledge to construct a solution to a problem. A problem-solving model provides both a conceptual framework for organizing knowledge and a strategy for applying that knowledge" [Sobolewski 96].

Design methodology provides the design process knowledge that is the knowledge about how to carry out the design process to advance the design situation towards a solution [MacCallum 89].

4.3.1 Better Design Methodology

A better design methodology has at least the following properties:

- takes less time, causes fewer failures;
- produces better designs (better quality, simpler designs);
- works for a wide range of design requirements;
- integrates different disciplines;
- conducts design in a concurrent fashion;
- consumes less resources: time, money, expertise;
- requires less information (see Axiom 2 in page 54).

4.4 Design Methods

A *method* is defined in the Webster dictionary as the following:

- a procedure or process for attaining an object;
- a systematic procedure, technique, or mode of inquiry employed by or proper to a particular discipline or art;
- a systematic plan followed in presenting material for instruction;
- a way, technique, or process of or for doing something;

- a body of skills or techniques;
- a discipline that deals with the principles and techniques of scientific inquiry;
- orderly arrangement, development, or classification: plan;
- the habitual practice of orderliness and regularity.

"Design methods are any procedures, techniques, aids or 'tools' for designing. They represent a number of distinct kinds of activities that the designer might use and combine into an overall design process" [Cross 89, page 33]. All design methods have two principal features in common: One is that design methods formalize certain procedures of design, the other is that design methods externalize design thinking [Cross 89, page 36].

"Design goals have different design methods associated with them, which specify alternative ways to make decisions about the design parameters of the goal. These methods capture the knowledge about the possible values of properties of components, as well as knowledge about the behavior of components. The role of the design methods is then to generate partial designs" [Mittal 92].

Cross [Cross 89, pp. 34-36] has categorized design methods to the following types: methods for exploring design situations, methods for searching for ideas, methods for exploring problem structure, and methods of evaluation.

We propose the following classification of design methods:

- algorithmic versus non-algorithmic
- theory-base versus experience based
- iterative versus explicit

- analysis versus synthesis

A design method might be a combination of different types of methods such as a generator method plus an evaluation method. In addition, a design method might include different methods of the same type, such as different generator methods based on different technologies, etc.

4.4.1 Granularity of Design Methods.

Smaller design methods have many benefits over large ones. There are some criteria for a small design method. The first is that a small design method is the one that makes fewer decisions. A decision is assigning a value to a design parameter. Additionally, a small design method uses less external information (e.g., fewer number of design parameters as input), and produces fewer numbers of design parameters as output.

If there are other methods that use the outputs of a large design method, they should wait until the whole sequence of calculations for that method is finished. This prevents the other methods from having immediate access to those outputs. If any of the outputs violate a design constraint, it will not be revealed until the whole chain of parameters are produced from that method.

For instance, in kinematic design of a 2-DOF robot, after assigning values to “base location” and “link lengths” parameters, there is a well defined method that finds the joint angles and then calculates the accessible area by robot. We break this method into two smaller methods: The first method finds the joint angles. The second methods uses the angles generated by the first method to calculate the accessible area.

In this domain there are two advantages of having smaller methods:

1. A constraint violation on the joint angles is discovered immediately after the first method has finished. That is, the calculation of the accessible region has not delayed the discovery of the constraint violation. At this point a re-design process starts and continues until a set of values is found for the angles that satisfy the constraints. In contrast to the large design method the calculation of accessible region is not included in the process of re-design hence, time is saved.
2. Other design methods that use the joint angles as their input parameters can start right after the first method has finished. That is, they do not have to wait until the calculation for the accessible region is finished too. This enhances concurrency between the design methods hence, speeding up the design process.

4.4.2 Design Approach

Within each design method we may define multiple design approaches. "There might be several kinds of artifacts, based on different technologies, that can exhibit the same functinality" [Mittal 92]. So, for an example, one approach to designing a controller system for a robot might be to use a PD (proportional plus derivative) controller, while another approach would be to use a SVF (state vector feedback) design for the controller. Both approaches can be used to design a controller for the robot.

A Design Method is a unit of procedural (operational) knowledge about how to produce values for some design parameters. A design method contains the knowledge about what approaches can be used to produce values for design parameters. These approaches might be ordered based on their priority over the others. A design approach might contain equations, look-up tables, heuristic rules, optimization algorithms, etc. that actually produce the values.

4.5 Knowledge-based Design Systems

"Knowledge-based systems are a special class of computer programs that purport to perform, or to assist humans in performing, specified intellectual tasks. In order to distinguish a knowledge-based system from other kinds of computer programs for design, it is helpful to think of knowledge in a computer program as being either *explicitly* or *implicitly* represented." [Dixon 95].

Knowledge-based systems are very difficult to develop and need more investment than regular computer programs. But once developed, they are more general—that is, they can solve a wider range of problems [Dixon 95].

In this dissertation a knowledge-based model of design is adopted in order to implement the proposed strategies for design process integration.

The knowledge-based model of design will be used to build a system that simulates the design process. The system activates design methods when they become applicable, uses small design methods, facilitates information sharing, implements control techniques for promoting collaboration, and gives more priority to design tasks that lead to fewer possible conflicts. We will explain these strategies in Section "Strategies for a Knowledge-based Design System" on page 66.

The system conducts the design process autonomously. By recording the steps that the system has taken during the design process, some partial methodologies are constructed using an inductive learning technique. These partially developed methodologies are then reinforced by solving more design problems. Later these methodologies will be categorized based on different sets of design requirements.

Figure 4-2 shows how the proposed approach works for our test domain, the design of a robot arm. There are three different disciplines (i.e., kinematics, structure, and controls) involved in the design process. Design methods in each discipline are broken up into small methods such that each one of them has its own inputs, outputs, and constraints.

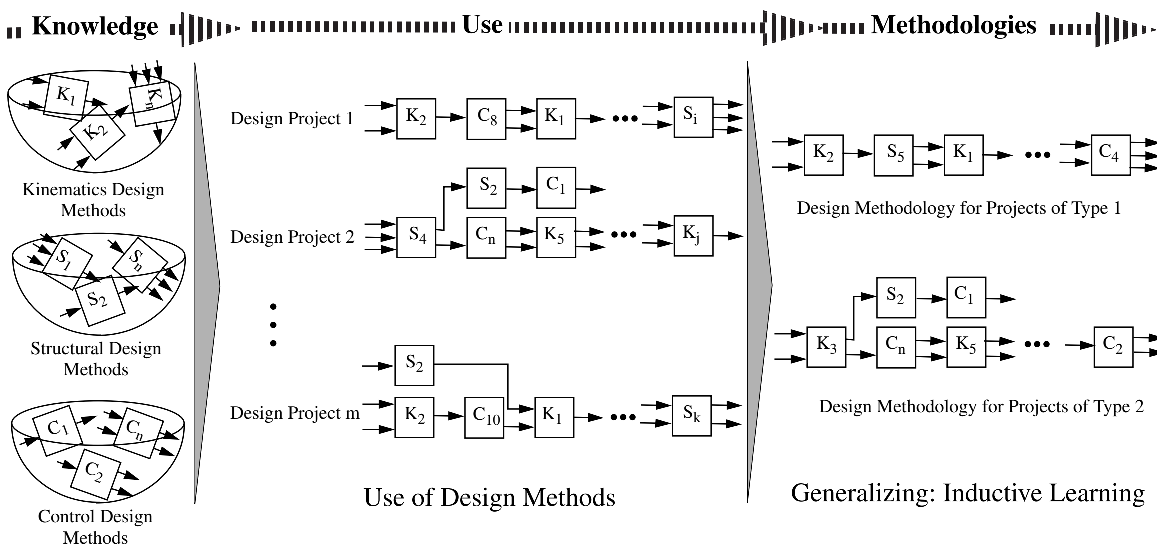


Figure 4-2. Knowledge-based Approach to Generating Design Methodologies.

A design project in Figure 4-2 is a design problem that differs from other problems in its requirements and constraints. As a result, design methods that become applicable during the design process might be different for different projects. However, there will be some similar patterns in activating design methods in different projects.

The similar patterns in activating design methods are extracted and related to the group of projects that later on will be categorized. To reinforce and further develop the similar patterns into general methodologies, the system solves many examples by perturbing the design requirements within the given range. We can close the loop by feeding the methodologies generated back to the system so that the system will follow them in new but similar design problems. In this process some of the candidate methodologies developed will be strengthened while some will be weakened and dropped from further development. At the end there will be a finite number of design methodologies for different types of design problems.

To implement the approach proposed a knowledge-based design tool based on a multi-agent architecture was developed that simulates the design process. “Design can be modeled as a cooperative multi-agent problem solving task where different agents possess different knowledge and evaluation criteria” [Sycara 90]. The multi-agent paradigm intuitively captures the concept of deep, modular expertise that is at the heart of knowledge-based design [Lander 97].

4.6 Strategies for a Knowledge-based Design System

The following sections describe the main problem-solving strategies that are to be followed in order to produce design methodologies using a knowledge-based approach.

4.6.1 Small Design Knowledge

In order to integrate different disciplines, one main strategy is to cut the big segments of design knowledge accumulated in different disciplines into pieces. In this work the design knowledge is represented in the form of design methods. A design method is a body of orderly procedures for accomplishing various design tasks (e.g., design synthesis, design selection, and design evaluation). Therefore cutting the design knowledge into pieces, in our approach, corresponds to breaking design methods into smaller methods. Smaller design methods means smaller number of decisions made in each method, shorter time is spent in a method, and less amount of information is produced as a result of executing that method. Smaller design methods are simpler and consume less resources.

There are some important advantages gained from smaller design methods including the following:

- smaller methods are more reusable, as a result methods can be applied in situations that were not explicitly anticipated at the development time [Lander 94, p. 4];
- repetition of design methods becomes faster;
- conflicts are discovered sooner because the produced information becomes available faster;
- resolving conflicts becomes easier because it is easier to track down the sources of conflict;
- shuffling the methods around, in order to change the order of their execution becomes easier;
- smaller design methods allow for going back and forth between different disciplines such that the border between different disciplines vanishes in favor of integration;

Breaking up big design methods into small methods is a type of more generic activity in problem-solving called decomposition. “One way to reduce the complexity of a large scale design project is to apply decomposition” [Kusiak 93]. Figure 4-3 shows how smaller design methods might remove the disciplinary boundaries and hence help in integration of different disciplines.

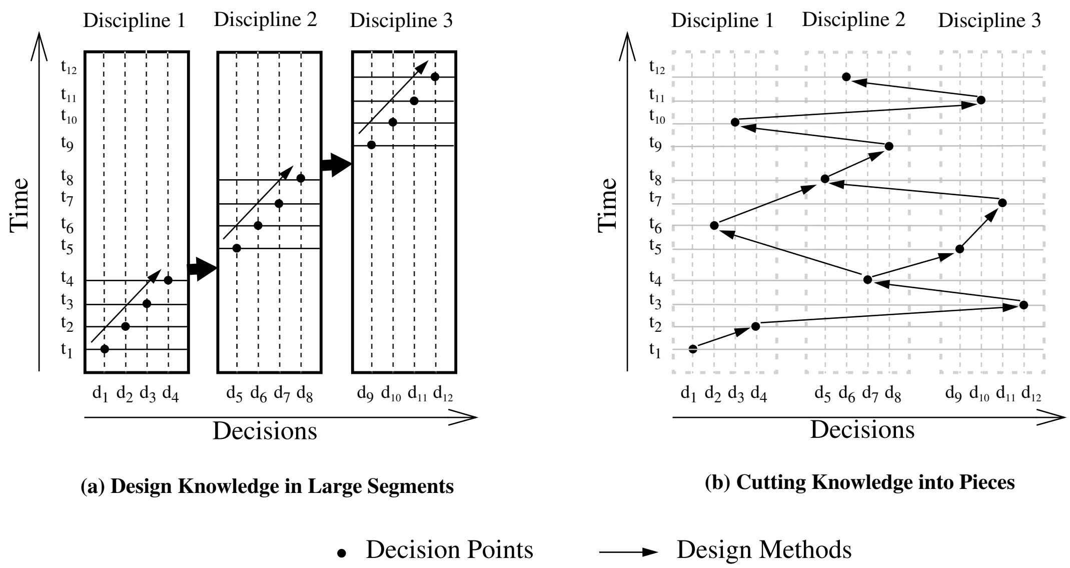


Figure 4-3. Integration by Breaking up the Knowledge into Smaller Segments.

Decomposition of the design knowledge is a knowledge intensive process—that is, it needs an extensive amount of domain knowledge. Breaking the design knowledge into smaller pieces might also require extensive knowledge engineering in order to get the smaller pieces of design knowledge in the right format. We have provided a few guidelines for how to break the design knowledge into smaller pieces in the last chapter. The guidelines are based on the factors that cause the design process to speed up.

4.6.2 Opportunistic Problem Solving

An opportunistic problem solving strategy is chosen to facilitate integration and contribution of different disciplines in the design process. "The key idea is that decisions are made as required and if possible. It moves the attention opportunistically between subproblems and avoids over-specifying local decisions" [Huang 93, page 102].

The opportunistic strategy for allowing different disciplines to contribute to the design is in contrast to executing a sequence of design tasks that is fixed *a priori*. "The specialization of design expertise suggests a future where teams form ad hoc collaborations dynamically and flexibly, according to the most *opportunistic* connections" [Wellman 95]. The opportunistic approach is necessary to achieve integration because of the complexity of the interactions among the disciplines. Also, an opportunistic approach is necessary if we are to take advantage of the diversity of different disciplines, because every participant should get a fair chance to contribute to the goals of the design process so that all points-of-view are explored and tried out.

Figure 4-4 shows the concept of an opportunistic strategy in design. In the beginning there are some design requirements that provide the required inputs so that some of the design methods can be executed and provide more information. Some other design methods get the opportunity to run after the first round of methods provide input for them. This process continues until the specification required for realizing the product is complete. Of course the process will not be as straightforward as was explained because, with the same available information, there may be many applicable design methods that could run or there may be none. Some of the information produced by the design methods may violate some of the constraints. Producing the same information by more than one design method causes conflict. There may be situations in which the design methods get stuck in a loop by providing no new information. These are the issues that will be addressed in the next two sections.

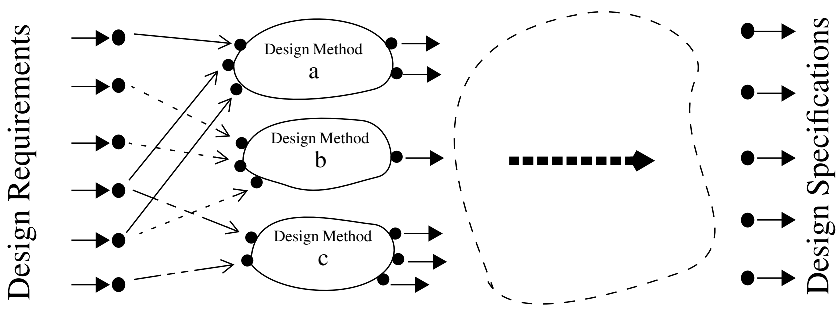


Figure 4-4. The Opportunistic Strategy in Activating Design Methods.

4.6.3 Cooperative Problem-Solving

Cooperative problem solving is in contrast to competitive situations. "In competitive conflict situations each party has solely their own benefit in mind and has no interest in achieving a globally optimal situation if such a solution provides them no added personal benefit. In cooperative situations, the parties are united by the superordinate goal of achieving a globally optimal solution, which often requires sacrificing personal benefit in the interest of increased global benefit" [Klein 91].

A cooperative strategy provides mechanisms by which different participants adopt the same goals. Implementation of the cooperative strategy in a multi-disciplinary design process results in favoring the common goals of the design over local goals. As a result of such a strategy different disciplines spend their diverse resources in the same direction. The cooperative strategy can be extended further such that different disciplines become considerate of the other disciplines' constraints when they propose their solutions. In this work the cooperative problem-solving strategy is implemented in the control mechanism of the design process as it is described in Section 6.

4.6.4 Least Commitment

Least commitment means deferring the decisions that constrain future choices for as long as possible [Jackson 90, p. 252]. That is, decisions should not be made arbitrarily or prematurely but postponed until there is enough information available [Huang 93, page 101]. The least commitment strategy, as opposed to early commitment strategy, is suitable for multi-disciplinary situations in which there is strong coupling between subproblems.

A least commitment strategy reduces the number of conflicts, because it avoids committing to decisions that are made based on incomplete information. In the absence of a least commitment strategy, decisions may be made as soon as they can be, even if incomplete, arbitrary, or less trusted information is used. As a consequence, there is more chance for conflicts to occur in the future, because such information may turn out to be invalid [Huang 93, page 102].

The least commitment strategy is implemented in the design process by giving priority to the design methods that use the least information (i.e., use fewer inputs) to produce the most new information (i.e., not producing information that already exists). The chance of using incomplete, arbitrary, or less reliable information is lower for the design methods that use the least information.

On the other hand, producing the most new information increases the chance for the next round of design methods to run based on least commitment strategy. As a result, the dependency of each design method and thus, the dependency of the produced information on the current available information is minimized. There are some benefits in adopting such a strategy, including the following:

- When the existing information is updated, some of the design methods that rely on that information as input have to be executed again. The methods that use the least information have more chance to be unaffected by these updates. This results in a faster design cycle.

- Because of postponing decisions, the least commitment strategy slows down the decision making process. However, giving priority to the design methods that produce the most new information speeds up the design process, because it increases the chance for the other design methods to run by providing them with their inputs.

- The design methods that produce the least amount of repetitive information reduce the number of updates. Making fewer updates reduce the design cycle. Sometimes the same piece of information (e.g., value of a design parameter) is produced by more than one different design method in the same or different design cycles. If the repetitive information produced in this way is not consistent, a conflict occurs. Consequently producing the least amount of repetitive information may reduce the number of conflicts too.

4.6.5 Inductive Learning

"Learning is the improvement of performance in some environment through the acquisition of knowledge resulting from experience in that environment" [Langley 96, p.5]. Learning is thus vital to construction of superior design methodologies. To develop categories of design methodologies that are specialized in collaboration and sharing of resources, an inductive learning strategy is adopted. Inductive learning is learning by examples and the form of inductive learning that is employed in this work is called *descriptive generaliza-*

tion. "In descriptive generalization, one is given a set of instances which belong to a particular class, and the task is to derive the most parsimonious description which applies to each member of that class" [Jackson 90, p. 433]. The idea is to record the traces of the design process during producing design solutions. Descriptive generalization can begin by collecting enough examples of design traces and produce classes of design methodologies. In the course of improving the produced design methodologies, solving more examples can reinforce the learned methodologies.

4.6.6 Means-Ends-Analysis

Means-ends-analysis is one of the general mechanisms used to direct a search process [Kannapan 92]. In means-ends-analysis each operation should reduce the difference between the current state and the goal state [Jackson 90, p. 68]. In the context of design, means-ends-analysis drives the design process in a direction that is the shortest distance toward the goal.

Means-ends-analysis can be implemented by choosing design methods that, compared to other methods, push the current state closer to the final state. The final state of the design is the state that has complete description of the design product while all the constraints are satisfied.

4.6.7 Concurrency

"It is well known that concurrent decision making is an important and very desirable component of modern design methodology" [Badhrinath 96]. A concurrent strategy, in contrast to a sequential strategy, carries out some of the problem-solving activities in parallel to each other. Concurrent design is the main theme of the well-established concurrent engineering field. Concurrency in design gives freedom to all participants to contribute to the current state of the design in parallel. In a concurrent design process, design knowledge is accumulated from all design participants during the design process [Brown 93]. As a result, the design process speeds up, because the participants in the design do not have to wait in a line if they can make a contribution.

The most common strategy to overcome the complexities of multi-disciplinary design is sequential design, in which different disciplines take part in the design process sequentially. In sequential design, information sharing between different disciplines is limited to the interfaces between disciplines [Levitt 91]. As a result, conflicts between disciplines are not discovered until they are very expensive to resolve, because their resolution may need to destroy the partial designs generated by the previous discipline.

“In sequential design, a tentative design synthesis is developed by one designer, often acknowledged as the lead discipline designer, which address some of the key performance specifications and constraints for the artifact” [Levitt 91]. Having a lead discipline that makes and explores some of the key decisions reduces the number of conflicts. The other disciplines conform to the decisions made by the lead discipline. However, that may prevent them from producing their best solutions. In a lead-discipline approach a single point-of-view dominates the decision making process and therefore constraints from that discipline are favored. This produces a lower quality design product and increases the number of iterations required to reach an answer.

4.7 Design Dependencies

Dependencies provide the knowledge for the ordering of the design tasks: one part of a design methodology. They also provide a source of decomposition knowledge. However, in our approach we don't need decomposition knowledge as we are cutting the design knowledge into small pieces and we activate them as they become relevant.

In addition to design dependencies that are extractable from design methods, there are other types of dependencies based on analytical or statistical studies. For example, there is a dependency between the end point deflection and first natural frequency of a robot arm [Christain 89]. This type of dependency is mostly based on first principles and can provide valuable information for developing a design methodology.

"Lack of knowledge of dependencies, due to lack of a global model, is an underlying cause of conflict" [Brown 96]. Therefore, dependencies help to develop design methodologies that produce fewer conflicts.

The dependency knowledge that we are extracting is based on the current available design knowledge in the form of design methods, rather than on first principles. The types of dependencies that two design methods might have are as follows:

- *Completely Independent*: Two design methods are completely independent if they neither share input nor output, Figure 4-5 (a).
- *Loosely Independent*: Two design methods are loosely independent if they only share one or more of their inputs, Figure 4-5 (b).
- *Reading Dependency*: Two design methods have reading dependency if at least one of them uses the other one's output as an input, Figure 4-5 (c).

- *Conflict*: Two design methods have ‘conflict’ type of dependency if both produce the same output, Figure 4-5 (d).

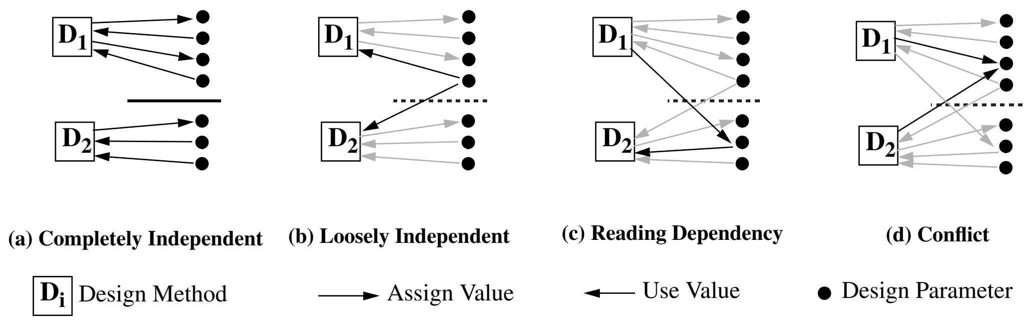


Figure 4-5. The Type of Relationship between Design Methods

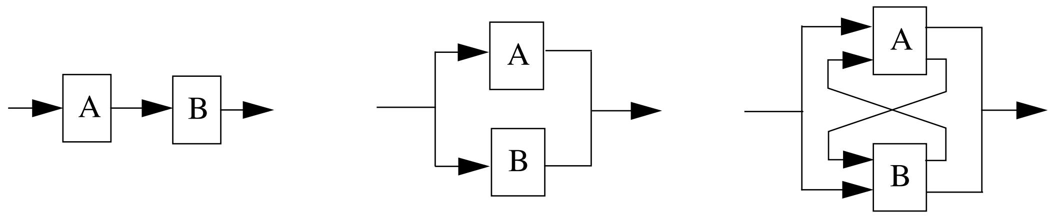
4.7.1 Sequencing Design Tasks

Eppinger proposes the following categorization of sequencing between design tasks [Eppinger 90]:

- Dependent Tasks: If task B simply requires the output of task A (or vice-versa), then the two tasks are dependent and are typically done in series.

- Independent Tasks: Tasks A and B are entirely independent if they can be performed simultaneously with no interaction between the designers.

- Interdependent (Coupled) Tasks: If task A needs information from task B, and also task B requires knowledge of task A’s results, then the two tasks are interdependent.



(a) **(b)** **(c)**

Figure 4-6. Sequencing of Design Tasks. After [Eppinger 90].

4.7.2 Dependencies and Decomposition

The knowledge-based system will be able to discover the dependencies between different design methods on the fly. It also will be able to rearrange the design methods so that the maximum concurrency happens between them. In this section the mechanisms used to dynamically discover the dependencies are described that are based on the work by Kusiak *et al* [93]. The results of implementing such mechanisms in discovering dependencies are presented in "Dependency Graph" on page 212.

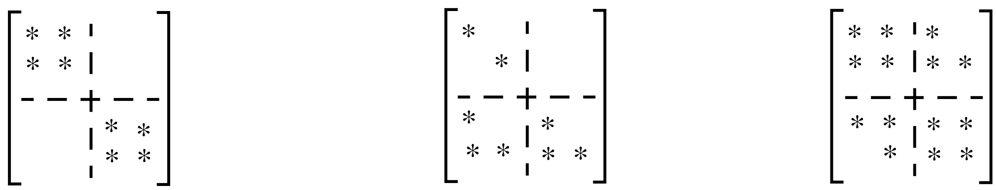
Kusiak has proposed three types of decompositions and for each decomposition type a type of incidence matrix is introduced [Kusiak 93]:

- decomposition of module (component) - activity matrix
- decomposition of procedure (formula) - parameter (variable) matrix
- decomposition of activity (variable) - activity (variable) matrix

The organized matrices can be categorized as follows:

- uncoupled matrix: an incidence matrix is uncoupled if its rows and columns can be ordered in such a way that the matrix separates into mutually exclusive submatrices,

- an incidence matrix is decoupled if it can be rearranged in a triangular form,
- an incidence matrix is coupled if it is not decomposable.



(a) uncoupled matrix (b) decoupled matrix (c) coupled matrix

Figure 4-7. Categories of Organized matrices. After [Kusiak 93].

Since the design activities associated with the upper left corner submatrix in Figure 4-7 (a) are independent of the activities corresponding to the lower right corner submatrix, they can be performed simultaneously. The activities in the decoupled matrix, Figure 4-7 (b), are dependent so that they can be performed partially in parallel and in part sequentially. The degree to which some tasks can be performed in parallel depends on sparsity of the matrix. In a coupled matrix, the activities are strongly interdependent which may occur in concurrent engineering. They may reflect an iterative nature of design or a negotiating process.

4.7.2.1 Decomposition of Module-Activity Matrix

A product or system can be decomposed into subsystems and these in turn into modules or components. Design of each module involves a set of design activities. Similar or identical activities may be performed in the design of different modules.

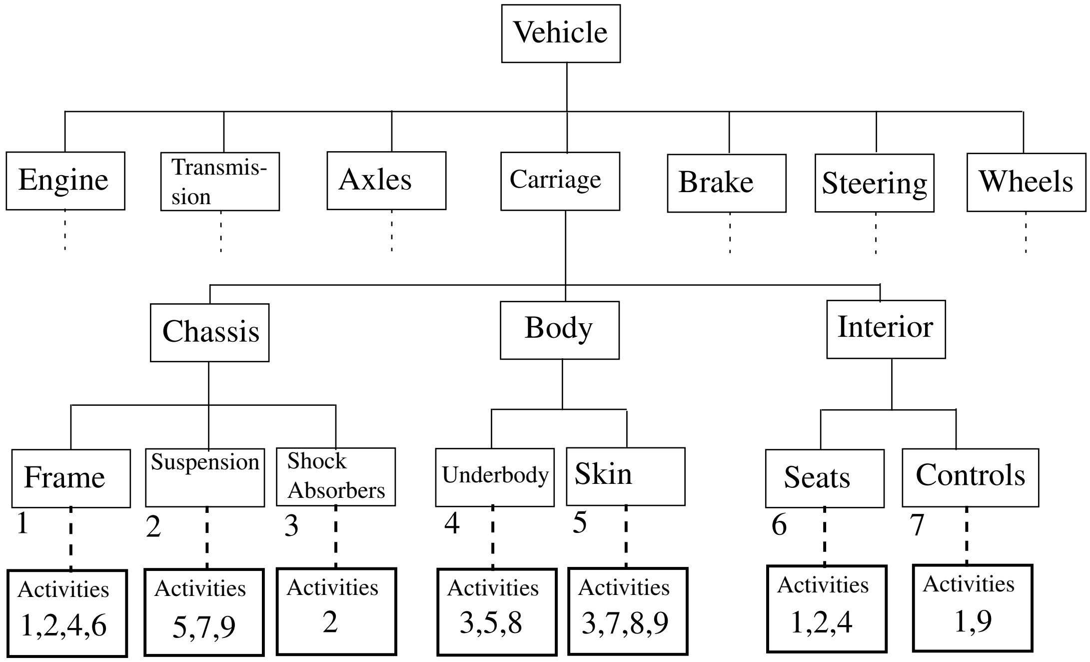


Figure 4-8. Decomposition of Module-Activity Matrix for a Vehicle. After [Kusiak 93].

The interactions between modules and activities can be represented as the module-activity incidence matrix:

Activities

Modules	1	2	3	4	5	6	7	8	9
1	*	*		*		*			
2					*		*		*
3		*							
4			*		*			*	
5			*				*	*	*
6	*	*		*					
7	*								*

Figure 4-9. Module-activity Incidence Matrix. After [Kusiak 93].

Using the clustering algorithm presented in [Kusiak and Cheng 90] the above incidence matrix can be rearrange as follows:

Modules \ Activities	1	2	4	6	3	8	5	7	9
1	*	*	*	*					
3		*							
6	*	*	*						
7	*								*
4					*	*	*		
2							*	*	*
5					*	*		*	*

Figure 4-10. Rearranged Module-activity Incidence Matrix. After [Kusiak 93].

Clustering of activities involved in the design process allows one to determine a potential group of activities that might be scheduled in parallel.

4.7.3 Building the Dependency Graph

We would like to build the dependency graph dynamically as the design process proceeds. It is dynamic in the sense that, if based on the current state of the design some design methods become applicable, the system picks those methods. The dependency graph would not include the design methods that did not participate in the process. Additionally, in general, the place of each design method may not be fixed in the dependency graph—i.e., the order in which design methods become applicable may change based on the state of the design (that itself depends on what design requirements were provided).

If we could ignore the dynamic aspect of the problem, the dependency graph could be built at the beginning of the design process and all the design methods could be indexed up front so that their order and depth in the dependency graph is fixed. For that implementation, there would be no need to check what design method is applicable in each cycle of design and the process of backtracking becomes considerably simpler.

To compromise, we adopt an approach in which the dependency graph is built incrementally as new design methods become applicable. We assume, however, that when the dependency graph is built it won't be changed, therefore we can index the designers with their suppliers and consumers as well as their depth in the dependency graph.

The best time to update the dependency graph during each cycle of design is after all applicable designers are determined. This happens in design as opposed to re-design actions. As a result, the dependency graph will have all applicable design methods to this point, should backtracking become necessary. The dependency graph is complete at the end of a successful design process.

4.8 Conflict Resolution

Two types of conflicts can be identified in design: *Domain Level versus Control Level* conflicts. "Domain level conflicts concern conflicting recommendations about the actual form of the design, while the control level conflicts concern conflicting recommendations about the direction the design process should take in trying to create a design" [Klein 91].

According to Klein [91] the relevant literature on conflict resolution can be grouped into three categories:

- *Development-Time Conflict Resolution:* Systems of this type require that potential conflicts be "compiled" out of them by virtue of exhaustive investigation when they are developed.

- *Knowledge-Poor Run-Time Conflict Resolution:* In this approach conflicts are allowed to be asserted by the design agents as the system runs, and then resolved by some kind of conflict resolution component, e.g., backtracking.
- *General Conflict Resolution:* Work in this class come closest to providing conflict resolution expertise with first-class status. Such systems provide categories of conflicts and have associate solutions or conflict resolution methods.

The conflict resolution approach that we have taken in this thesis is a combination of the first two methods in the above list. We prevent conflicts by collecting all the knowledge that is relevant to a design parameter in one agent. Following the strategy of "Small Design Knowledge" on page 66 we break the big segments of knowledge into small pieces so that all the pieces that decide about one particular parameter can be collected together. As a result, there will be more than one way to decide about the value of a particular design parameter.

In this chapter we described the knowledge-based model of design and its ingredients. In the next chapter we will describe the multi-agent system paradigm. The framework that will be presented for the multi-agent design system is a generic architecture that is applicable to all parametric design problems. However, to avoid too many abstract discussions we present the framework in the context of robot design.

5 Multi-Agent Systems (MAS)

5.1 Introduction

Software agents and multi-agent systems are relatively new technologies and as a result are not well defined. "In fact one of the most hotly debated issues in the agent research community is the definition of *agency*" [Lander 97]. This chapter reviews some of the literature related to agents and multi-agent systems, and specifically multi-agent systems for design. At the end of the chapter we propose a framework for a multi-agent design system. The following paragraphs summarize some of the definitions given for agents and multi-agent systems in the literature.

An *agent* is a self-contained problem solving system capable of autonomous, reactive, pro-active, social behavior. It is a powerful abstraction tool for managing the complexity of software systems [Wooldridge 95] [Franklin 96] [Wooldridge 97]. A *multi-agent system* is "a system composed of multiple interacting agents, where each agent is a coarse-grained computational system in its own right" [Wooldridge 98].

Agents are distinct, distributed, often autonomous computational processes that are made aware of their environment, continually monitoring the state of their world, choosing an appropriate action and reacting to changed conditions of this world. These processes can be seen in many important computer applications such as planning, cooperating robotics, process control, manufacturing, distributed sensing, avionics, *collaborative design*, and health care and diagnostics [Torsun 95, page 401].

In this dissertation we adopt the notion of an agent as an abstraction tool for conceptualizing, designing, and implementing the knowledge-based design approach that was proposed in the Chapter 4.

5.2 Characteristics of Multi-agent Systems

Multi-agent systems (MAS) have some characteristics that make them an attractive solution to many complex problems including multi-disciplinary design. Torsun [Torsun 95, page 402] summarizes these characteristics as follows:

- MAS are distributed systems and distribution is a useful approach to controlling complexity. Large and complex systems (e.g., design systems) can be decomposed into multiple cooperating agents such that control can be decentralised and rendered easier to deal with.
- Interactions and cooperation are a natural approach for many large evolutionary systems. These systems are subject to continuous change and extension. MAS facilitate the design and implementation of such systems.
- MAS increase reliability and robustness. MAS normally have some degree of redundancy in that more than one agent can solve the same task or the same knowledge is known by several agents. As a result, the system becomes more robust against the breakdown of some agents.

- "*MAS provide insight and understanding about information processing phenomena occuring in the real world. Research into computational methods that take the social interaction between the agents themselves and with their environment may shed light on how activities and actions are achieved in the face of enormous complexity*" [Torsun 95, page 402]. This advantage is the central point of using MAS for implementing the knowledge-based system that is to shed light on the complexity of the design processes by simulating the real world process.
- Due to the parallelism, the MAS approach is potentially more efficient. Several agents might be working simultanaously and asynchronously on their tasks. This allows one to investigate the effect of concurrency on the complexity of the process. Also, implementing the tasks that are inherently concurrent becomes much easier in MAS.

5.3 Developing MAS

In this section we review some of the techniques and methods for developing intelligent agents and multi-agent systems that we used for this dissertation.

5.3.1 Message Sequence Chart (MSC)

"Message Sequence Charts (MSCs) are a widespread means for the visualization of selected system runs (traces) within communication systems. A main advantage of an MSC is its clear graphical layout which immediately gives an intuitive understanding of the described system behavior" [Rudolph 96]. MSC language constructs are as follows [Rudolph 96]:

- *instance*. In the graphical representation, instances are shown by vertical lines or, alternatively, by columns. Along each vertical instance axis a total ordering of the described communication events is assumed. In Figure 5-1 `Coordinator`, `Designer`, `DesignDatabase`, and `DesignState` are instances.

- *message*. The message flow is represented by arrows which may be horizontal or with a downward slope with respect to the direction of the arrow to indicate the flow of time. In addition, the horizontal arrow lines may be bent to admit message overtaking or crossing. In Figure 5-1, `ask(design)` and `sorry(cause)` are examples of message element.

- *environment*. The system environment is graphically represented by the frame symbol which forms the boundary of an MSC diagram. In Figure 5-1 the environment sends the message of `ask(start)` to the `Coordinator` instance.

- *action*. Actions describe an internal activity of an instance. An action is graphically represented by a rectangle containing arbitrary text. In Figure 5-1 instance `Designer` performs the action of `ConstraintChecking`.

- *timer set*. The setting of a timer is represented by an hour-glass connected with the instance axis by a (bent) line symbol.

- *timer reset*. The reset symbol is represented by a cross (X), connected with the instance axis by a (bent) line symbol.

- *time-out*. Time-out is described by an arrow which is connected to the hour-glass symbol.

- *instance creation*. The create symbol is a dashed arrow which may be associated with textual parameters. A create arrow originates from a parent instance and points at the instance head of the child instance. In Figure 5-1 the `Coordinator` instance creates a `Designer` instance.

- *instance stop*. An instance can terminate by executing a process stop event. Execution of a process stop is allowed only as last event in the description of an instance. The termination of an instance is graphically represented by a stop symbol in form of a cross at the end of the instance axis. In Figure 5-1 the instance `Designer` has stopped after sending the message of `sorry(cause)`.

- *condition*. Conditions can be used to emphasize important states within an MSC or for the composition and decomposition of MSCs. Conditions are represented by hexagons covering the instances involved. In Figure 5-1 the condition `wait` shows the state of instances `Coordinator` and `DesignState`.

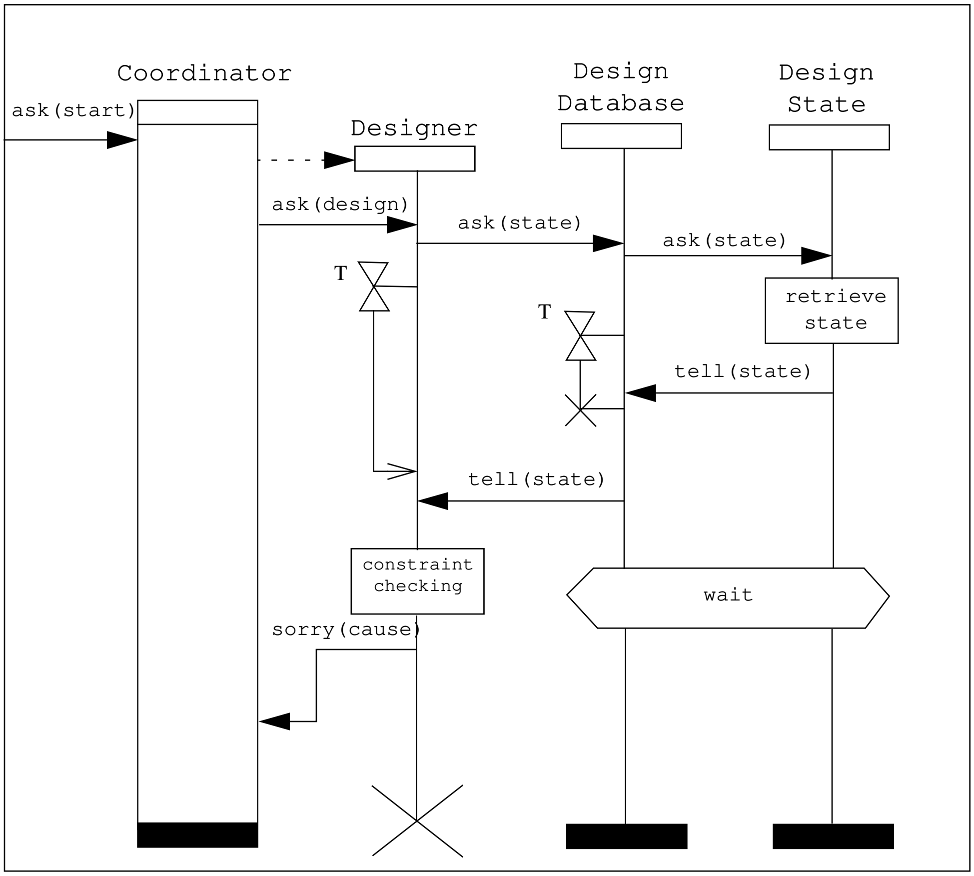


Figure 5-1. Basic Elements of MSC language

5.3.2 Model Development Cycle

The following is a model development cycle for multi-agent systems suggested by Iglesias *et al* [96]:

- Describe the prototypical scenarios between agents. These scenarios can be a further development of the scenarios determined in the conceptualisation phase for the use cases. The scenarios are described using message sequence charts (MSCs) [Rudolph 96]. An alternative representation is event trace diagrams. During this first stage, we will consider that every conversation consists of only one single interaction and the possible answer. The objective at this stage of development is to establish the set of conversations (channels) between agents.

- Represent the events (interchanged messages) between agents in event flow diagrams (also called service charts). These diagrams collect the relationships between the agents via services.
- Model the time of each interaction. An expertise model (EM) can help us to define the interchanged knowledge structures. EM models the problem solving knowledge used by an agent to perform a task.
- Model each interaction specifying speech-acts as inputs/outputs of message events.
- Each state can be further refined. If the state represent a knowledge task, the inference templates of the CommonKADS library [Breuker 94] are very useful. While decomposing a state, it can be decomposed in different agents, and the complete process should be repeated.
- Analyse each interaction and determine its synchronisation: synchronous, asynchronous or future.
- Determine the receivers of each service request: individual or group and if a coordination protocol like contract-net is desired [Smith 80]. This can be represented in SDL using the names of the agents or group names in the explicit addressing facility.
- Determine if a cooperation protocol is needed for each conversation. The reasons for using a cooperation protocol can be among others [Durfee 89]:
    - Increasing task completion through parallelism.

- Increasing the set or scope of achievable tasks by sharing resources (information expertise, physical devices, etc.).
- Increasing the likelihood of completing tasks by undertaking duplicate tasks.
- Decreasing the interference between tasks by avoiding harmful interactions.
- Resolving conflicts via negotiation protocols. Usually these conflicts need to be assisted by a human agent. The coordination model (CoM) is used for modelling the negotiation language category (protocol, primitives, semanticas and object structure) and, partly, the negotiation process category (procedure and behaviour) according to the classification of negotiation categories by [Müller 96].

5.4 Multi-agent Design Systems (MADS)

"Design applications that incorporate software agents are multi-agent design systems (MADS)" [Lander 97]. MADS technology offers an appealing framework for situations where multiple disciplines are participating in the design process by combining diverse sources and types of information and reasoning [Lander 97]. "The constant evolution of standards, technologies, and a dynamic marketplace demands a high degree of adaptability in both design expertise and in the process of applying that expertise. The need for diverse, highly sophisticated, and rapidly changing skills and knowledge makes the multiagent paradigm particularly appropriate for knowledge-based design" [Lander 97].

There are many issues that should be considered when developing a MADS system. In the following we review some of the issues raised by Lander [97], and describe the way that we will be handling those issues. This will be the foundation for proposing a framework for our multi-agent design system.

5.4.1 Interoperability

The first issue to consider when developing a MADS is interoperability. Interoperability means that the agents should be able to talk to each other. The factors that affect interoperability are:

- **Agent Heterogeneity.** One source of heterogeneity is the multi-disciplinary nature of the design problem. The designer agents are heterogeneous in that they each have a different point-of-view of the design problem. The other source of heterogeneity is the difference in the various tasks that should be done in the system including design, coordination, conflict resolution, methodology discovery, and so on.

- **Implementation.** The difference in implementation of different agents might reduce the interoperability of the system. As we have built the whole design system from scratch, there is no heterogeneity due to differences in implementation, language, or operating systems.

- **Representation.** Finding one representation language and model for all the designer agents, may not be trivial. Agents will be designed in order to understand and process the other agents messages. Also, some agents rely on other agent's services. For example, the agent that does coordination relies on the agent that provides the dependency knowledge.

- **Interaction.** Having a common protocol for interacting enhances the interoperability between the agents. In our system one communication protocol and language is used for all the agents, therefore in interacting with each other the agents do not need to translate the messages. However, there is a need for translating the physical concepts for designer agents in different disciplines. The approach is to find a common denominator for all disciplines and then translate all other quantities using the common ones. For example in the robot arm design (see Chapter 2), the controller designer uses the concept of moment of inertia. This is not used by the kinematic or structural designers, but it is related to length and mass which is understandable by both designers.

5.4.2 Information Flow

Information flow among agents is particularly important in design systems. Agents should be able to access the right information at the right time to prevent conflicts in the early stages. Some factors that affect the information flow among agents are described in the following:

- **Task Dependencies.** Task dependencies are mostly due to the fact that some agents are users of the services that the other agents provide. The agents ought to work closely together because of the task dependencies so that integration of different disciplines can be accomplished. As we discussed before, information sharing is one of the key factors for achieving integration, therefore the agents are designed to share information easily by sending and receiving messages.

- **Automation.** Automation refers to how much human intervention is expected in facilitating the information flow among agents. In our system information flows automatically from one agent to another with no human intervention.

- **Sharing Information.** Each agent in a MADS has public information that it should share with other agents, Sharing information is critical for realizing integration among different disciplines in the design. In our system there will be a shared repository for shared data and designs. Some database agents are responsible for gathering, storing, and providing different types of shared knowledge.

- **Routing Information.** Routing is the issue of what information should be sent to which agent and, perhaps, when. In our system the knowledge for routing information is partly based on the architecture and is partly embedded in the agents themselves. There are some agents whose job is mainly or exclusively routing the information, such as database agents. Agents have the knowledge on where to send the information that they produce.

5.4.3 Adaptability

Adaptability is another issue in MADS. A MADS is adaptable if new knowledge can be added and old knowledge removed without affecting the integrity of the whole system. The multi-agent system that we have developed is adaptable as we can add and delete designer agents. New technologies can be incorporated into the designer agents in the form of new design approaches. The internal structure of each agent can change as long as the services that are needed by other agents are provided.

5.4.4 Concurrency

Whenever possible, agent activities should be concurrent. Concurrency is one of the major enhancements to the design process. “However, concurrent activity introduces a number of issues in how to maintain enough consistency in data and scheduling across the agent set to enable effective performance. For example, questions to ask include how important is it for all agents to have up-to-date information?” [Lander 97].

The following sections describe the factors that affect concurrency in MADS.

5.4.4.1 Consistency

Consistency of shared data among different components is a very well known issue in systems with concurrent processes. Synchronizing the tasks that might run at the same time is a solution to make sure the data is consistent among the components. To solve the problem of consistency in the system we run the design process in cycles of *consistency*. The approach is to run the design process in cycles of *run-analysis-update* as is shown in Figure 5-2. In the ‘run’ part of the cycle the ‘designing’ takes place, in the ‘analysis’ part it checks the constraints, and in the ‘update’ part it combines the results with partial designs.

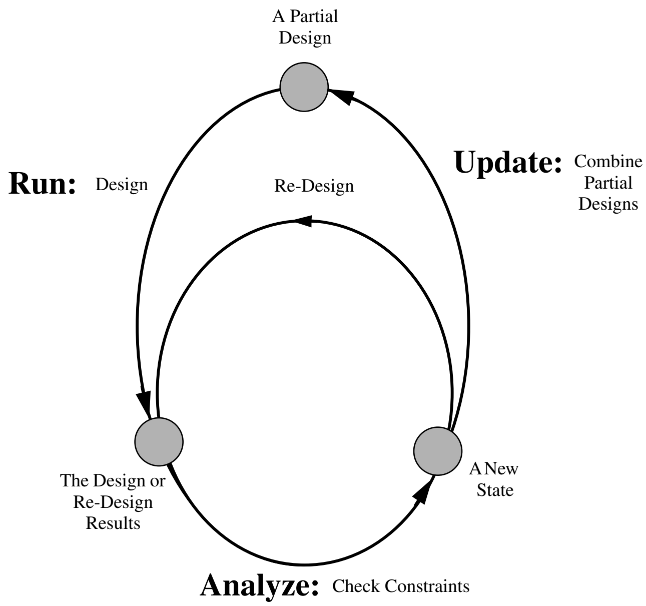


Figure 5-2. The Design Cycle.

The design cycles have to be small compare to the whole design process in order to cope with the opportunistic strategy (see "Opportunistic Problem Solving" on page 68). One of the factors that helps to obtain small cycles is to have small design methods (see "Small Design Knowledge" on page 66).

The 'run' part of each cycle may include designing with multiple agents at the same time. Also, in each designer agent several design tasks might take place simultaneously. Similarly, the 'analysis' and 'update' parts of each cycle may be done concurrently as long as synchronization of different updates is taken into account.

Implementing cycles of consistency converts the continuous design process to a discrete process; smaller cycles makes the process closer to an actual design process. All agents will have a consistent view of the design state after each update is complete.

Figure 5-2 shows that when the results of constraint checking are not successful, a 're-design' happens in which the system backtracks to the previous decisions and changes them. The system fails to find a successful design if all the possibilities are tried but still some of the constraints are still not satisfied.

5.4.4.2 Information Update

Each agent sends an appropriate message upon producing any piece of new information that the others might want to know. This message can be in the form of a notification to another agent that will decide how to use the new information (e.g., the results of designer agents) or it can be in the form of asking for an insertion in the shared data repository (usually done by coordinator type of agents).

5.4.4.3 Event Notifications

Most of the notification events are propagated by specific agents that have the knowledge of how to handle important notification events, i.e., which agents should be notified of what event. These agents are the coordinator type of agents. However, it is the responsibility of all the other agents to notify these agents of the important events. Most event notifications are concerned with updating values for design parameters. Two groups of agents are interested in these updates: first, those that can run because they now have enough information to run; Second, those whose previous designs depended on updated parameters, and who therefore should re-design.

5.4.4.4 Update Intervals

Different design activities were grouped in Figure 5-2 into three categories: run, analysis, and update. None of these activities get interrupted because of any possible new changes in the environment. As long as the cycles of run-analysis-update are small there is no problem in postponing propagation of changes to the end of each cycle.

5.4.4.5 Merging Multiple Partial Designs

A partial design is a subset of the design parameters with values proposed for them by designers. If there is no overlap between the proposed and previously constructed partial designs the new partial design will be the union of the two sets. In the case of overlap between different partial designs (i.e., if both have assigned values to the same design parameter) the following algorithm is employed:

- If the assigned values are exactly the same (for discrete parameters) one of them is accepted and the partial designs are merged.
- If the difference between assigned values is within an accepted range then based on the type of the design parameter an average, the minimum, or the maximum value is used in the merge.
- If the difference is big, then a conflict resolution agent is notified to resolve the conflict and return one value back so that the partial designs can be merged.

5.4.5 Strategic Control

Strategic control yet is another issue in MADS that deals with *how* the knowledge should be used. The global strategy for the control of the design process in our system is based on synthesis of new designs, analysis, exploration of the design space, and evaluation of alternative designs. The global strategy for the control of the design process will not include retrieving existing designs, constraint propagation, or analogical reasoning.

Information about the control strategies for the internal behavior of the agents are solely stored in their local databases. Global control strategies are in part realized in the architecture of the system and the rest is implemented in the form of special agents whose service is to apply the strategies. The internal states of agents are stored in their own databases, while the global design state is stored in a special agent called `DesignState`.

A major part of the responsibility for strategic control lies in the coordinator agents. The strategic control will embed the strategies (already listed in "Strategies for a Knowledge-based Design System" on page 66) in the system to effectively realize the global strategy. They are:

- Small design methods: This is realized by building small designer agents in terms of the execution time and the number of decisions that they make.
- Opportunistic strategy: All the designer agents that are able to contribute to the current state of design are notified and allowed to do so.
- Cooperation: The designer agents are asked to contribute to design in the direction of common goals of the design. The flow of information is facilitated and accomplished through message passing between agents, and in fact there are special agents whose service is solely providing information. Conflicts are resolved in the favor of the common goals of the design.
- Least commitment: The priority is given to those designer agents that use the least information to produce the most information.

- Inductive Learning: The inductive learning technique that we will use requires that all the data to be present at the outset. As a result, the learning part (i.e., learning design methodologies) will be done off-line.
- Means-Ends-Analysis: The coordinator agents prepare agendas based on what action should take place in order to close the gap between the current design state and the final state.
- Concurrency: Multiple designer agents get the chance to run concurrently.

5.4.5.1 Interactions

Some part of the interactions are planned and some are reactive. Implementation of the opportunistic strategy causes reactive interactions. Implementation of cycles of *run-analysis-update* embeds some planned interactions. The system runs without the intervention of a user, hence there will be no user-controlled interactions. Interactions are peer-to-peer as opposed to client/server.

5.5 A Proposed Framework for MADS

Figure 5-3 shows the proposed framework for the MADS based on the discussion in the previous section.

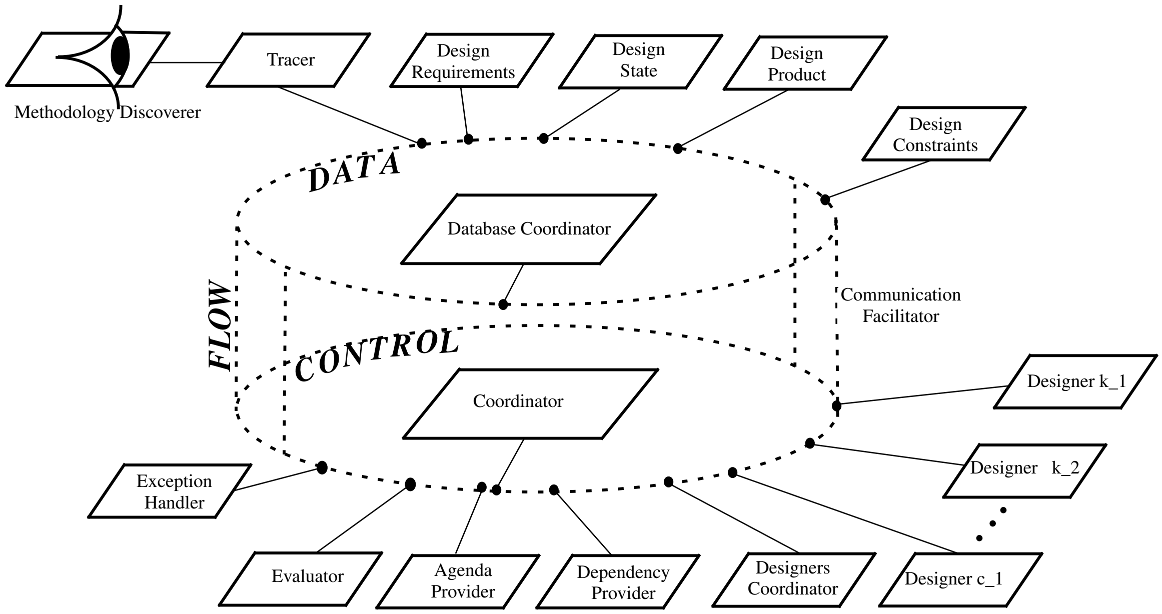


Figure 5-3. The Architecture of the Multi-agent Design System

There are three different layers in the system: *Data*, *Control*, and *Flow*. The data layer contains the design requirements and design constraints defined by the user at the beginning of each design project. The data layer also contains the state of the design process at any moment and the description of the product as it evolves during the process. Database agents update data and answer the queries of the other agents. A coordinator agent manages the consistency of the data between different database agents and synchronizes the updates and queries. Figure 5-3 shows how different agents are responsible for gathering, storing, and providing different types of shared knowledge. These agents are `DesignState`, `DesignRequirements`, `DesignProduct`, `Tracer`, `DesignConstraints`, and finally `DatabaseCoordinator`, responsible for gathering data and distributing it among the aforementioned agents to store it.

The control layer contains the design knowledge as well as the knowledge for how to use the design knowledge. In Figure 5-3 each `Designer_m_n` agent is responsible for carrying a specific design method *n* in discipline *m* (`k` for kinematics, `s` for structural, and `c` for control design of a robot arm).

The rest of the agents in the control layer are responsible for coordination and carrying out generic design tasks such as evaluation of the partial designs. They discover and provide the dependency between designers, and provide an agenda for various design tasks such as backtracking.

The flow layer of the system contains a mechanism for communication among agents based on sending and receiving messages. This mechanism consists of a registry and a message passing protocol. Each message has its own thread for processing, that not only provides concurrency between agents, but also it allows each agent to handle multiple messages simultaneously.

5.5.1 Agent Dependencies

In this section we describe the dependencies between the different agents in the system as is shown in Figure 5-3. In the following we refer to task dependencies between agents that means how one agent needs service from other agents to be able to do its job.

- The most dependent agent is `Coordinator` that uses most of services and the least dependent agents are database agents that only provide service to others.
- The `AgendaProvider` depends on `DependencyProvider`, that is the planning task depends on the dependency providing task in order to plan the sequence of design actions that should be taken by `Coordinator`.

- The `DependencyProvider` depends on the `Designers` to get the domain data dependency information based on what designers supply the input to each designer. This information will be then combined by `DependencyProvider` to build the complete design dependency graph (see Figure 10-1 on page 214).
- The `Coordinator` agent depends on `AgendaProvider` to execute the backtracking when a constraint is violated.
- All of the designer agents depend on most of the database agents such as: `DesignState` to check if they can run at the current state, and `DesignRequirements` to get the requirements.
- The `Evaluator` agent needs the information provided by `DesignConstraints` to evaluate the partial design.

5.5.2 Information Routing

`DesignDatabase` and `Coordinator` are the agents that are heavily involved in information routing. All agents know where they should ask for what service or where to send their outputs. Designer agents, for instance, know that they should send their partial designs to the `Coordinator` for evaluation, composition or further processing. One reason the `Coordinator` and `DesignDatabase` agents have been given more centralized role in the architecture (Figure 5-3) is to facilitate information routing by possessing some of the routing knowledge. The benefit is to let the other agents concentrate on their main job and also make the changes easier.

In this chapter we reviewed the area of multi-agent systems in general and multi-agent systems for design in particular. We also proposed a framework for a multi-agent design system based on aspects of MADS. In the next chapter we will show how is it possible to discover design methodologies from the trace of a multi-agent design system. We then review the techniques and methods of machine learning that can be used to automate the discovery of methodologies.

6 Discovering Methodologies

6.1 Introduction

In this chapter we discuss how the multi-agent design system (MADS) will be used to discover methodologies. First we describe how the system generates different designs for different problems. We then discuss how the system might take different steps and use different knowledge. Finally, the methods and techniques that are to be used to analyze the results generated are explained.

First we explain some of the terms that will be used throughout this and the following chapters.

6.1.1 Design Problem

A *design problem* is defined with a set of requirements and constraints. The set of requirements and constraints is often called *design specifications* [Pahl 88, p. 51]. The solution to the design problem is the *design product*. A design product is described by the set of *design descriptions* [Coyne 90, p. 71].

6.1.2 Design Project

A design project contains the specifications of the requirements and constraints (i.e., design specifications), the description of the design process, and finally the description of the product (i.e., design descriptions). The reason we introduce the concept of design project is to encapsulate all the information about the design problem, design process, and design product in one term.

Therefore, a design project contains a design problem and more. Later in this chapter and in the following chapters the phrase “the system solved a design project” means that a design problem was given as a set of requirements and constraints, a design process took place, and a design product was generated as the result of the process.

In this dissertation the most important factors that distinguish different projects from each other are requirements, constraints, and the steps that were taken during the design process.

6.1.3 Design Path

Each design method might possess more than one way of doing design (see “Design Approach” on page 63). That is, each designer agent may have different approaches for generating its output design parameters. For generating a design, a combination of different design approaches from different designers are used. If the generated design does not satisfy the constraints, another combination of design approaches is tried. It is like the design process takes different paths through agents to generate different candidate solutions for the same set of requirements. The candidate solutions that satisfy the set of constraints are the acceptable designs.

Figure 6-1 shows how selecting different design approaches produces different design paths. A path can be represented by the sequence of approach indices that were used, e.g., `1,1,1,2`. An alternative would be to index the sequence of indices, e.g., call that path `Path 2`. When a constraint is violated, designer agents systematically check all other possible design paths by varying their design approaches.

The knowledge about how designer agents are dependent on each other is used to select those paths that have a chance of resolving the constraint violation. They are executed while the rest will be pruned. This reduces the time and effort needed to find the path that generates a successful design (i.e., the design that satisfies all the constraints). This technique is known as dependency-directed backtracking. We will discuss the use of dependency knowledge for dependency-directed backtracking in “Dependency Graph for Design of a 2 DOF Robot” on page 214.

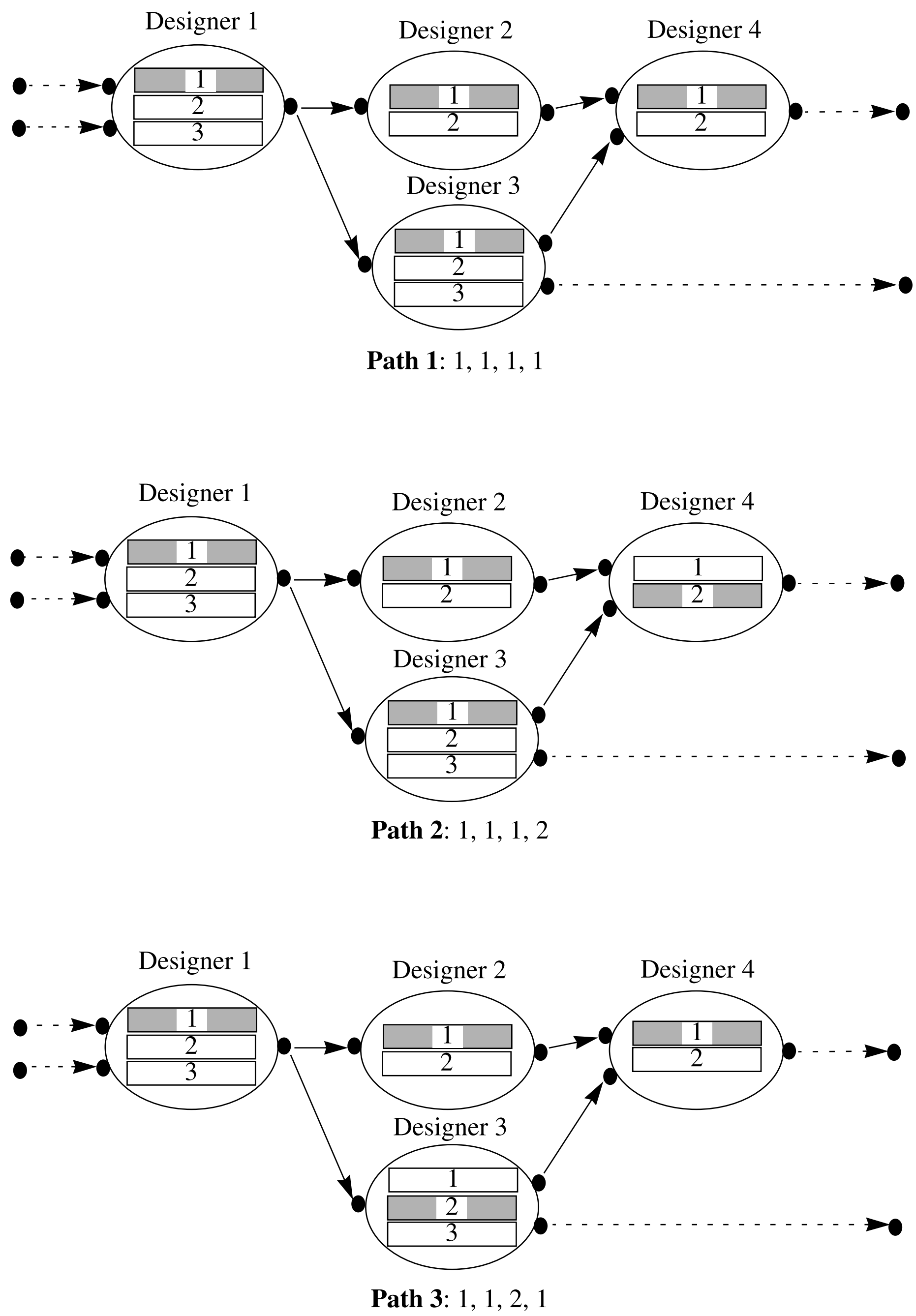


Figure 6-1. Different Design Paths.

In a design project the system may take different paths until a path generates a successful design. On the other hand, any change in requirements and constraints might force the system to take a different path than the other projects in order to achieve a successful design. That is why paths for different projects might be different.

6.1.4 Traces

A trace is the record of whatever actions the systems takes. Different types of traces are generated by the system (“Traces Produced by RD” on page 197). The type of the trace that we use for generating methodologies is the *trace of the design path* as opposed to say, the trace of design solutions. The ‘trace of the design path’ is the trace of the design approaches that the system has used, while the ‘trace of the design solutions’ is comprised of the values generated for the design parameters. Throughout this and the following chapters we often refer to this type of trace as “the trace of the system” or “the trace”. The traces of the system are represented by the same method that design paths are represented, e.g., the sequence of design approach indices or the index of the sequences.

6.1.5 Clusters

A cluster is a group of entities that are similar, i.e., have common features. Forming clusters of entities and describing the common features is a way to generalize the set of entities and create a new entity that represents all the members of the group. In this dissertation we are interested in clusters of traces and clusters of problems.

6.1.6 Requirements versus Constraints

Requirements and constraints both specify which solution in the design space is an acceptable design. In this dissertation, however, we have made a distinction between design requirements and design constraints. The set of requirements specify what are the properties of an acceptable design, therefore, the set of requirements are given as a set of parameter-value pairs. The set of constraints, on the other hand, describe what should not be violated. A constraint specifies a relation between a set of design parameters [Mittal 92].

Therefore, we might find points within the design space that satisfy the requirements but are not acceptable because they are outside the solution space (i.e., violating constraints). The method that we have used in the knowledge-based system is to use the requirements to generate the solution points in the design space. We then check to see if those points are inside the acceptable solution space by checking them against design constraints.

For example, some requirements were defined for a 2-DOF robot: a workspace that is stretched 5 meters, carry 1 kg workload, have 1 second settling time, and 10% maximum allowable overshoot. The required settling time and overshoot is quite tight for such a long workspace. So very high control gains were expected. The system started creating designs with gains as high as 300, while the maximum allowable gains were set at 100. The generated designs were clearly satisfying the requirements but because they were not within the acceptable solution space they were all rejected.

6.2 Mapping Problem Space to Design Space

The multi-agent design system maps the space of requirements (i.e., problem space) to the space of traces and then to the space of designs (i.e., design products), (Figure 6-2). A "space" is the reference set that contains all the members. Throughout this dissertation we often refer to this mapping as "design process follows a trace", i.e., to generate a design the design process follows the path shown by the trace of the system.

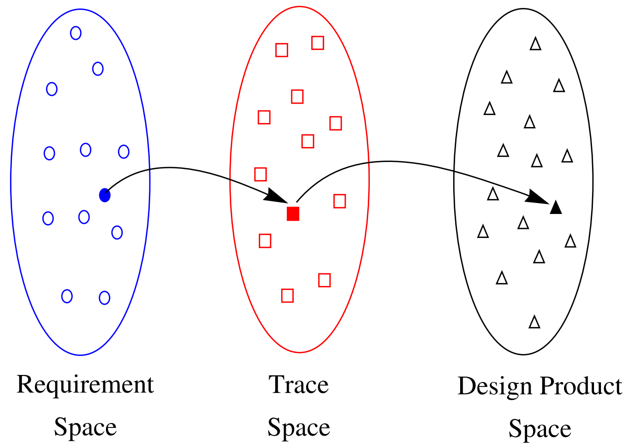


Figure 6-2. Mapping from Requirements to Designs.

It is the set of constraints that guides the mapping from requirement space to trace space and then to the design product space. For a new design project the system takes the first available trace that is formed by combining the default approaches of each designer. The system progresses through the design process checking for constraint violation at the end of each design cycle. If there is a constraint violation the system backs up and chooses a different path. As a result, for the same set of requirements, a different set of constraints may force the system to take a different path and produce a different design (Figure 6-3).

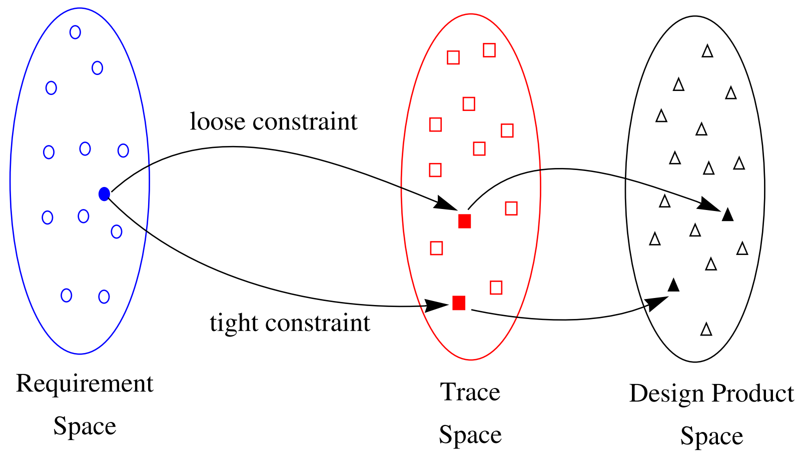


Figure 6-3. Different Constraints Produces Different Designs and Traces.

Also, it is possible that the same trace gets used in more than one project to produce a successful design (i.e., a design that satisfies all the constraints), (Figure 6-4). In fact, this is what we are hoping to happen so that we can group the design projects that used the same trace together. If a set of design projects can be grouped together based on some common characteristics we can formulate some guidelines on how to conduct the design process for similar projects. The set of these guidelines will eventually lead to the formation of methodologies. If similar projects take similar paths we can generalize both the set of projects and the set of traces, so that the methodologies generated can be applicable to a wider range of problems.

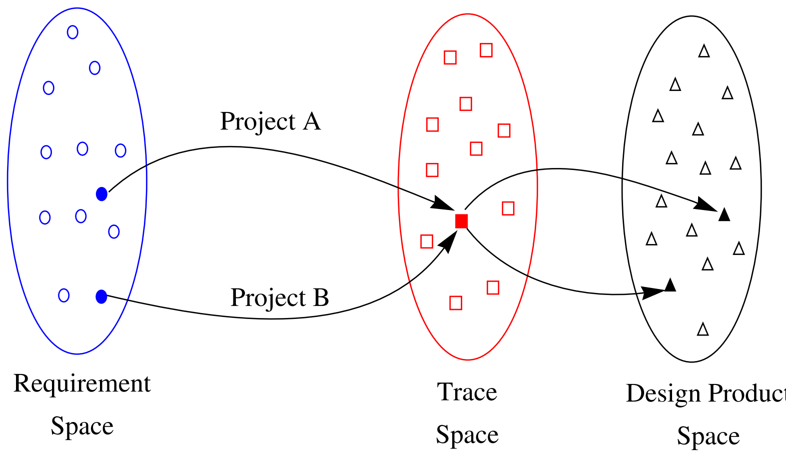


Figure 6-4. Same Trace Gets Used in More than One Project.

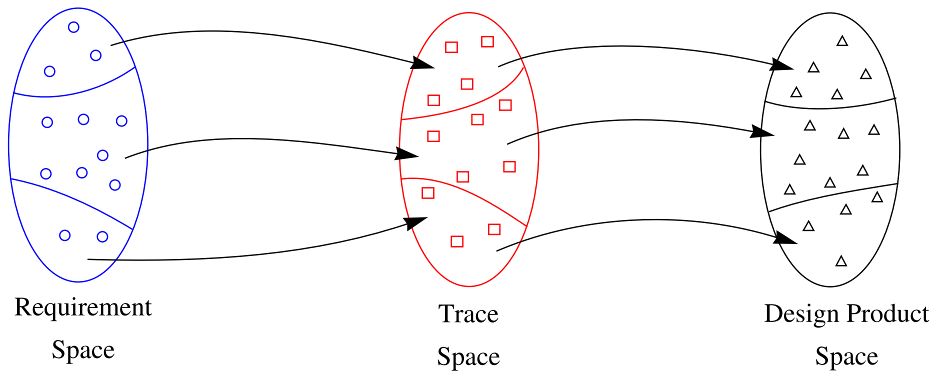
The best case that could happen in mapping the requirement space to the design space is that all the problems could be mapped to the solutions using only one trace. This, however, is unlikely to happen except perhaps in very simplified and single discipline design problems. The methodology that is generated in such an ideal case will include all the different situations included within the requirement space. It also will be very simple and precise in what design approaches should be followed in those situations. We will be looking for mappings from clusters of projects to clusters of traces for successful designs.

In reality we expect to see many traces are used to map the requirement space to design space. The following scenarios may happen in mapping the requirement space to the design space, Figure 6-5:

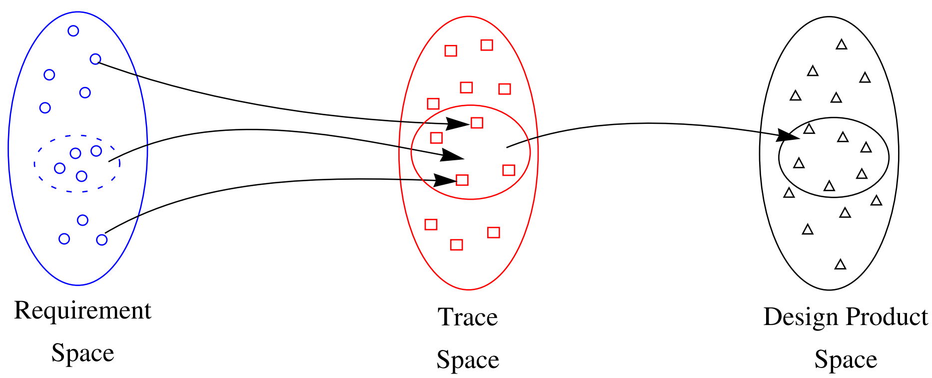
- **Case 1:** Each cluster of requirements is mapped to the design space by exactly one cluster of traces.
- **Case 2:** A cluster of projects plus some exceptions not included in that cluster are mapped to the design space by exactly one cluster of traces.

- **Case 3:** A cluster of projects is mapped to the design space by one trace cluster plus some exception traces that do not fit in the cluster.

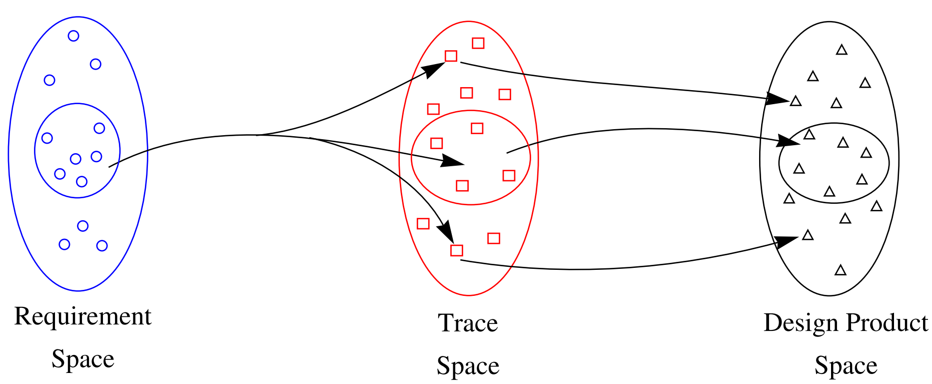
Other cases might happen that basically are a mixture of the above cases. However, with respect to generating the methodologies the most desirable cases are cases 1 to 3. The reason is that the above cases have the least exceptions, therefore the generalization becomes much cleaner and will cover more situations. In the next section we describe the techniques and methods that can be used for this generalization process.



Case 1: Exact Match between Clusters



Case 2: Partial Match Includes Exceptions



Case 3: Partial Match Includes Exceptions

Figure 6-5. Different Scenarios in Mapping Requirements to Designs.

6.3 Machine Learning

The methods and techniques of Machine Learning can be used to automate the process of extracting the design methodologies. These methods will provide the means to form the clusters of projects and clusters of traces and then finding some relationship between these two type of clusters. In this section we review some of the techniques and methods from the area of Machine Learning that can be used to extract the methodologies from traces of the system.

The use of Machine Learning methods in support of design has been well documented [Duffy 97]. Depending on what is included in the traces, and its representation, we can take advantage of work that has been done on clustering and on induced finite-state transition networks, inductive learning for state-space search, or flexible macro-operators [Langley 96, pp. 258, 304, 348].

The source of information available for automatic learning can be a collection of case histories. "The reason that the study of case histories is easier than manual expert elicitation is that experts can provide solved problems more readily than they can articulate their knowledge explicitly" [Rich 91].

The idea of extracting design methodologies from traces is very close to the subject of *concept formation* in Machine Learning. "Concept formation is the task of automatically inferring the general definition of some concept, given examples labeled as members or nonmembers of the concept" [Mitchell 97, p. 21]. *Supervised concept learning* and *unsupervised concept learning* are the two primary classes of machine learning techniques that comprise the *inductive* approach [Quinlan 93, p. 2]. Reich and Fenves in [Reich 91] have proposed to use the available information from experience or simulation (e.g., a set of design projects) to generate concept hierarchies or production rules. These hierarchies or rules are then used to predict the design product description for a new set of design requirements [Reich 91].

The difference between what Reich and Fenves have proposed and what we are intending to do is that we would like to predict the *design process* description from the new requirements rather than the *product* itself. This is because our objective is to improve the process of design—that is we would like to learn from the results of the system about the quality of the process rather than the quality of the product.

Consequently, mapping from the space of design requirements to the space of the traces of the system is more important than mapping from the space of traces to the space of design products (see Figure 6-2). The only aspect of the design product that we consider in the process of methodology generation is whether it satisfies the constraints (i.e., a successful design) or not. This conclusion helps us to decide what information should be included in the traces in order to be able to extract design methodologies from them.

There is an issue of how to relate classes of similar design requirements to classes of similar traces—that is, there is no guarantee that after classifying requirements and traces there will be a one-to-one map from design requirement classes to trace classes. One method to solve this issue is to merge the requirements and design approaches into one trace and do the classification in such a combined trace. The result would be classes of design methodologies that have the corresponding requirements embedded in them. In fact this is the way that classification has been described in Machine Learning literature (e.g., [Rich 91]). That is, the training set is described by a list of attribute-value pairs that include both design specifications (in our case requirements) and design description attributes (in our case design approaches).

Merging the set of requirements into the set of design approaches and producing a combined trace for clustering them can be used to automate the process of clustering all together. However, in this dissertation we will generate separate sets of requirement clusters and trace clusters. The reason is that we would like to explicitly reveal the relationship between these two sets. Clustering of the combined trace will hide this relationship (see “Requirements versus Constraints” on page 108).

6.3.1 Supervised Learning versus Unsupervised Learning

“Supervised concept learning creates knowledge structures that support the task of classifying new objects into predefined classes. In the case of design, examples are represented by a list of specification property value pairs and are classified into a set of classes that can represent a single design descriptor” [Rich 91]. Arciszewski, Mustafa, and Ziarko in [Arciszewlski 87] use a supervised method to differentiate between feasible and infeasible designs. The goal of acquired rule set is to predict whether or not a given combination of design description values is feasible. McLaughlin and Gero in [McLaughlin 87] present a similar approach. Instead of differentiating between feasible and infeasible designs, their task is to characterize designs that are optimal. These two approaches essentially extract evaluation knowledge rather than synthesis knowledge [Rich 91].

There are two approaches based on supervised learning that can perform synthesis. First, specification properties can be used to generate a classification over the set of designs (e.g., with ‘optimal’ and ‘inferior’ as labels). Concept descriptions in terms of the design properties can then be used to characterize subsets of the training data that were distinguished by their specification properties. This process captures a many-to-one mapping between designs and classes of specifications. In this process, new specifications are classified (e.g., via a decision tree) into a subset of designs (e.g., leaves of a decision tree). The pattern associated with this subset is forwarded as the synthesized design. Unfortunately, it appears that this strategy may yield patterns that are too general for practical purposes [Rich 91].

Supervised concept learning techniques are inadequate as a means of capturing synthesis knowledge. The reason is that synthesis involves a many-to-many mapping from requirement space to design space. Supervised concept learning, however, requires separate many-to-one mappings. Such separation causes information to be lost, since the requirements in the set are not dependent [Rich 91].

"Learning paradigms that are concerned with many-to-many mappings are unsupervised. The principle idea is that specifications and solutions (i.e., design descriptions) are correlated; specific combinations of specification properties give rise to corresponding combinations of design description properties that satisfy these specifications" [Rich 91].

In this dissertation we use unsupervised learning methods to find correlations between the requirement space and trace space (as opposed to the design space). A clustering based on this correspondence allows the retrieval of an appropriate trace given a new set of requirements that is similar to an existing one. The next section describes a clustering algorithm for this purpose.

6.3.2 Agglomerative Formation of Concept Hierarchies

This section is a summary of an approach for concept formation called Agglomerative Formation of Concept Hierarchies (ACH) [Langley 96, pp. 212-217]. We will use a slightly changed version of the ACH algorithm proposed by Langley in order to cluster the traces. Other clustering techniques might also be appropriate [Fisher 91], but they have not been investigated.

"ACH constructs concept hierarchies in an *agglomerative* manner, grouping instances into successively larger clusters. Although one can run such methods on supervised training data, they are typically used on unsupervised learning tasks. Also, they are nearly always nonincremental in nature, requiring that instances be present at the outset" [Langley 96, page 212].

The ACH algorithm receives a set of training cases (e.g., design traces) and a matrix that specifies all pairwise distances between the instances. ACH finds the closest pair of entries A and B, which may be observed instances or, later, clusters of instances. The algorithm combines the two entries into a new cluster C, storing A and B as its children in the hierarchy and generating an intentional description for C.

The methods for constructing an intentional description for C include: numerical averaging, storing the instances themselves, or generating a logical, threshold, or probabilistic summary [Langley 96, page 213]. In this dissertation the description for C would be the union of the two traces combined that is a logical summary of different traces (see Table 10-3 on page 217).

Next the algorithm checks to see if any entries remain to be incorporated. If not, ACH halts, returning the entire hierarchy it has generated along the way. If entries remain, it removes all pairs containing A and B (since they are now covered by C) and calculates all pairwise distance between C and the remaining entries. ACH then calls itself recursively on the new set of pairs, combining the closest pair of entries, adding a new node to the hierarchy, and so forth, until it has combined all entries into a single taxonomic structure.

The distance metric that is used to measure the similarity of two design traces is a city block measure (i.e., Hamming distance, [Langley 96, page 214]) that contributes 1 for each mismatched design approach and 0 for each matched approach. In calculating the distance between two cluster of traces the distance metric contributes 0 if the set of approaches accumulated in one cluster is a subset of the other cluster's approach set and 1 otherwise.

The ACH algorithm explained above constructs a binary concept hierarchy with exactly two children for each nonterminal node. We have modified the algorithm to find the *k* nearest entries on each recursion, thus generating a branchier tree. In each recursion the algorithm finds the minimum distance between non-identical clusters and find *k* neighbors that are within a circle that has a diameter equal to twice of the minimum distance. Using this method the algorithm finds a central tendency for each cluster and the computed distance between the clusters.

One alternative for the distance metric is to use Euclidean distance between traces or clusters, assuming that a trace or a cluster is a point in an n-dimensional Euclidean space, where n is the number of designers (hence, number of approaches in each trace). But compare to Hamming distance, the Euclidean distance can not capture the similarity of the traces with the same accuracy.

Euclidean distance, however, can be a measure of the goodness of the trace. If we measure the distance of a trace from the origin, it shows how much it has been affected by less desirable approaches. Less desirable design approaches are at the end of the list and have a higher approach index—that is they are farther from origin (of the n-dimensional space of traces) than traces with more desirable approaches.

6.4 Representation of Methodologies

Rules and decision trees are two representation methods that we propose for design methodologies. In representing the design methodologies by rules, the “IF” part of the rule specifies the characteristics of the cluster of design projects to which that methodology applies. The “THEN” block of the rule provides the guidelines on how the methodology has to be followed. The “THEN” part includes the description of the design approaches in the cluster of traces that mapped the cluster of design projects to the successful designs. An advantage of representing the design methodologies using rules is that the representation can be very close to “English”. This has a better chance of being understood by human designers and increases their likelihood of acceptance.

We could have a decision tree that classifies the points in the design requirement space. “In general, decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances. Each path from the tree’s root to a leaf corresponds to a conjunction of attribute tests, and the tree itself to a disjunction of these conjunctions” [Mitchell 97, page 53]. At the leaves of the tree we assign the design methodology that works for that particular group of design requirements. Extension of this idea would be to have different candidate methodologies assigned to a leaf that each are biased toward a specific aspect of design process, such as: “fastest”, “least expensive”, “best product quality”, “best manufacturable product”, etc.

If the clusters of traces are scattered all over the trace space or if there are many exceptions in the clusters of projects or cluster of traces, decision trees are a better solution. In Chapter 10 based on the distribution of the traces we will decide whether to use rules or decision trees for representing the methodologies generated.

In the next chapter we collect all the building blocks needed for constructing a multi-agent design system for the design of a 2-DOF robot. We will combine the results of Chapter 2 (design of a 2-DOF robot) and Chapter 5 (multi-agent design systems) to build a system called *Robot Designer* (RD).

7 Robot Designer (RD)

7.1 Introduction

In this chapter we put together all the building blocks needed to develop a multi-agent system that designs a robot. We call the system ***RD***, Robot Designer. This chapter is mostly based on the discussion in Chapter 2 about how to design a 2-DOF robot and Chapter 5 regarding multi-agent design systems. In the first half part of this chapter we present the results of breaking the design knowledge into small pieces. Each piece will eventually be embedded in one designer agent—that is, for each design method there will be a corresponding designer agent. As a result, in this chapter and the next chapters we might use the terms design methods and designer agents interchangeably. In the second half we represent the structure of each agent as well as algorithms and flowcharts for the system.

7.2 Design Methods for Robot Design

In "Design Methods" on page 60 we discussed what a design method is. In this section we present the design methods for the design of a 2-DOF robot. These methods will be based on the results of Chapter 2. For each design method we might have multiple approaches (see "Design Path" on page 105). To implement the strategy of small design methods we break the large design methods at decision points. Decision points in a design method are when a value is assigned to a design parameter.

The names that we use for the design methods are comprised of two parts: one of the letters of K, S, or C (standing for Kinematics, Structural Mechanics, and Controls disciplines) separated by a dash '-' from the index of that method in that discipline. For instance, K-1 means the first design method in kinematics. The ordering of the design methods in a discipline has no meaning.

In the following discussion we start with kinematic design methods, followed by structural, and then control design methods.

7.2.1 Kinematic Design Methods

The kinematic design decides where to put the base of the robot relative to the workspace, the length of the links, the joint angles, and the area of the accessible region. As a result, four kinematic design methods are introduced in this section.

7.2.1.1 Design Method K-1

Design method K-1 decides about the location of the base of the robot. A schematic diagram of the inputs, outputs and the design approaches of design method K-1 is shown in Figure 7-1. A description of each design approach follows the figure.

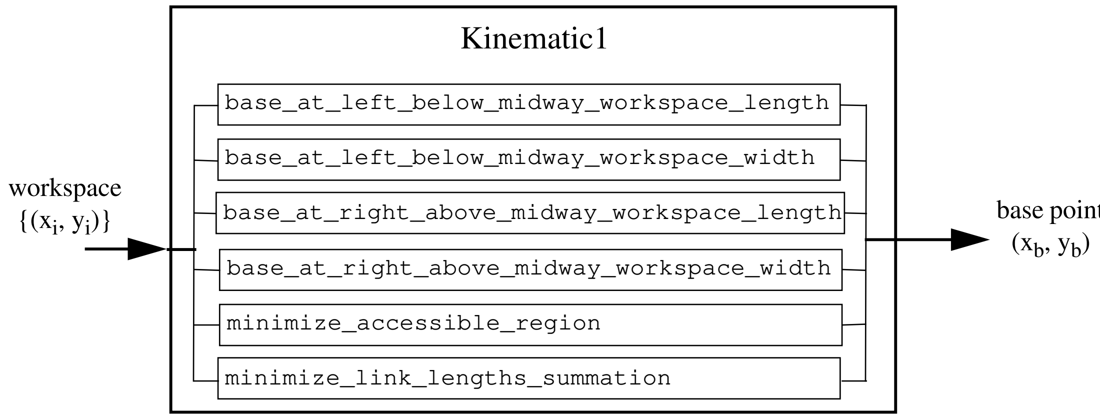


Figure 7-1. Kinematic Design Method 1

The description of the design approaches for Design Method K-1 are as following:

1. `base_at_left_below_midway_workspace_length`: puts the base of the robot at the left or below the midway of the length of the rectangle that circumscribes the workspace points. If the rectangle is vertical the base would be to the left of it and if it is horizontal the base would be below it (point 1 in Figure 7-2).

2. `base_at_left_below_midway_workspace_width`: point 2 in Figure 7-2.

3. `base_at_right_above_midway_workspace_length`: point 3 in Figure 7-2.

4. `base_at_right_above_midway_workspace_width`: point 4 in Figure 7-2.

5. `minimize_accessible_region`: this approach puts the base of the robot in a point so that the accessible region (Figure 2-1 on page 28) by the robot is minimized (point 5 in Figure 7-2). The minimization algorithm is "Downhill Simplex Method in Multidimensions" based on [Press 89].

6. `minimize_link_lengths_summation`: this approach finds a location for the base of the robot so that the sum of link 1 and link 2 lengths is minimized (point 6 in Figure 7-2).

While the last two approaches may help in satisfying tight constraints on structural and control performance of the robot, they are more expensive in terms of computational efforts and more time consuming. Besides, as it can be seen from Figure 7-2 (point 5 or 6), minimizing the accessible region or link lengths usually needs to put the base of the robot in the middle of the workspace, where large joint angles are needed in order to cover the whole workspace.

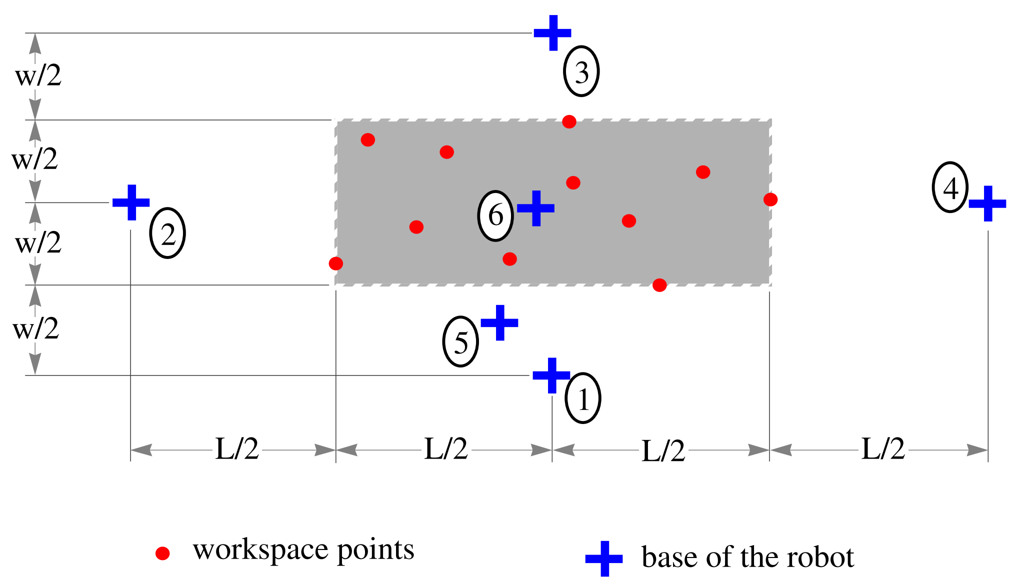


Figure 7-2. Different Locations for the Base of the Robot.

7.2.1.2 Design Method K-2

Design method K-2 decides about the ratio of the length of the second link to the length of the first link.

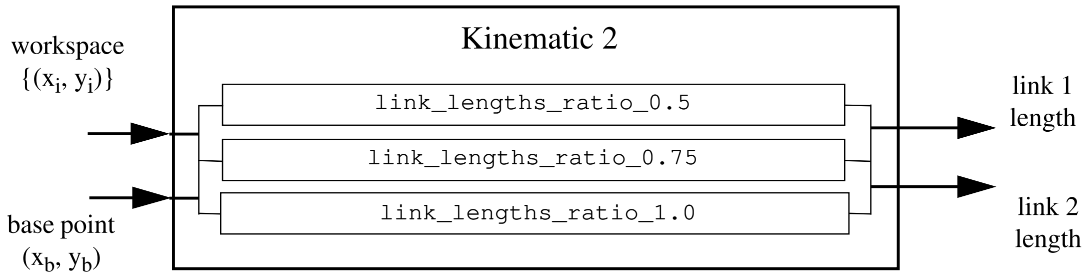


Figure 7-3. Kinematic Design Method 2

Different design approaches for Design Method 2 are different in the ratio of the length of the link 2 to link 1. These approaches assign a length to the second link that is half, three quarters or equal to link 1, respectively.

7.2.1.3 Design Method K-3

Design method K-3 decides about the configuration of the arm. There can be two different configurations: left-handed and right-handed (see Figure 2-3 on page 31).

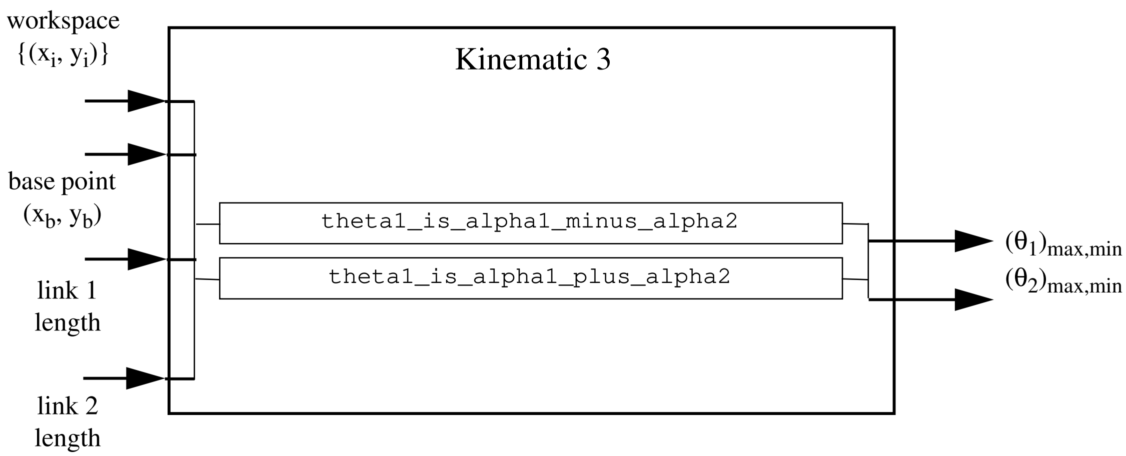


Figure 7-4. Kinematic Design Method 3

7.2.1.4 Design Method K-4

Design method K-4 calculates the area of accessible region by the robot.

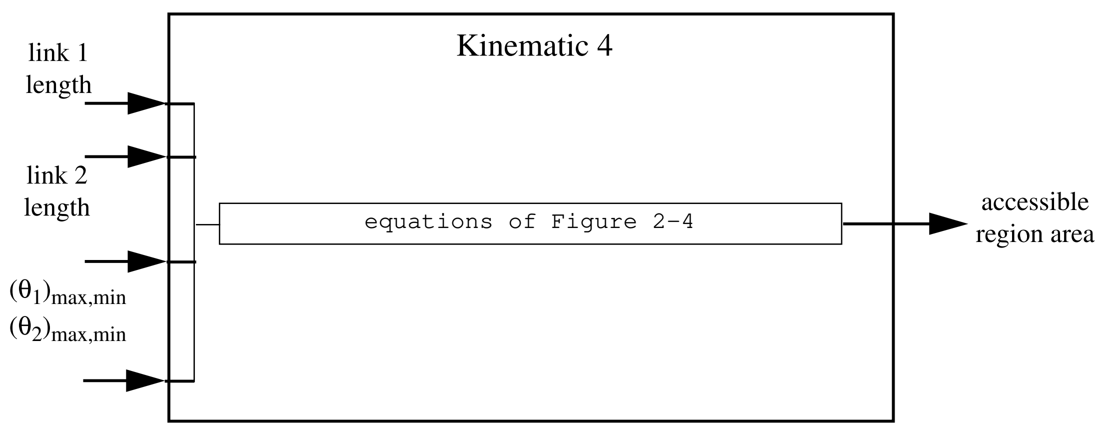


Figure 7-5. Kinematic Design Method 4

7.2.2 Structural Design Methods

The structural design methods decide about the material for the structure, the safety factor, the shape and dimensions of the cross section of the links and finally the deflection of the tip of the arm. As a result, five structural design methods are introduced in this section.

7.2.2.1 Design Method S-1

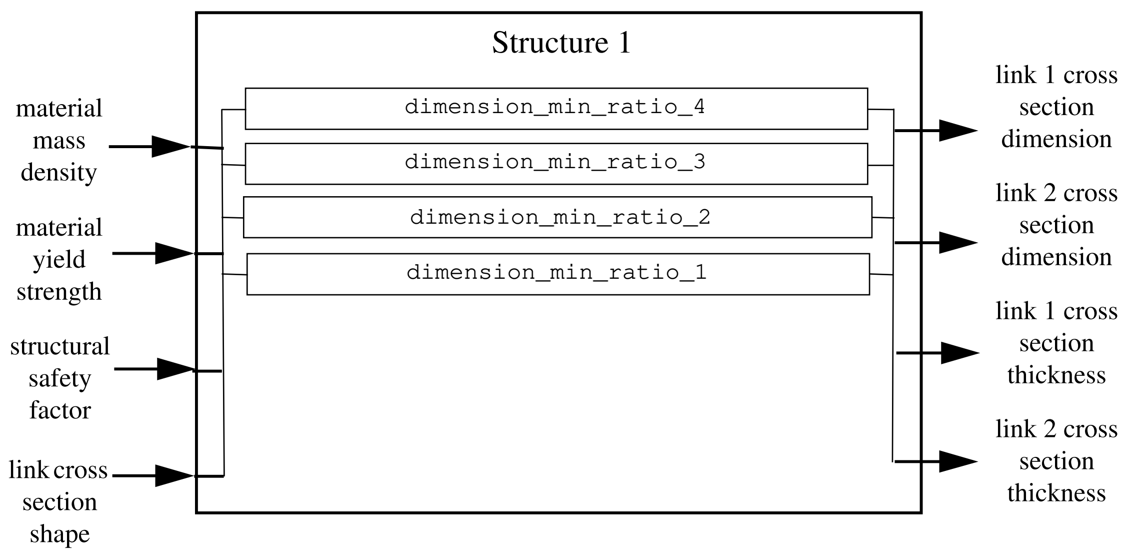


Figure 7-6. Structural Design Method 1

The design approaches are based on what should be the ratio of the dimension of the cross section of the link to the minimum dimension required by stress analysis. These ratios vary from 4 to 1.

7.2.2.2 Design Method S-2

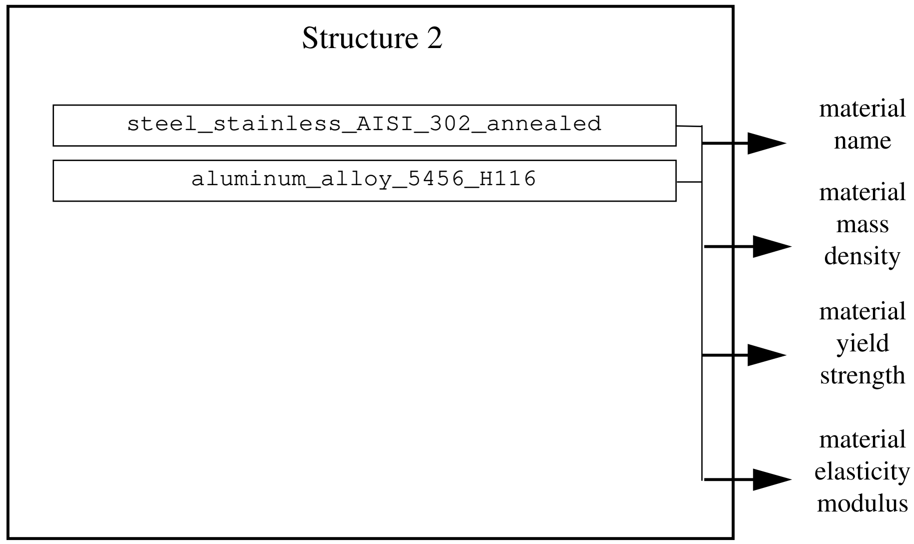


Figure 7-7. Structural Design Method 2

Design method S-2 decides what material should be used for the links. The available materials are steel and aluminum. This method is an example of a method that selects items from a catalog. A catalog look-up method does not need an input in the form a design parameter to use it in its calculations. It, however, may use domain knowledge about how to search for the best choice. The designer that will encapsulate design method S-2 will receive inputs in the form of a request for its service, i.e., a look-up in the catalog. In the case of backtracking, the designer will be asked to provide another alternative for the material.

7.2.2.3 Design Method S-3

Design method S-3 like the design method S-2 is a catalog type of design method in which the safety factor for structural design is looked up in a table. The safety factor is reduced in situations that the requirements are tough or the constraints are tight. Reducing the safety factor decreases the dimensions of the link, hence reducing the weight and moment of inertia of the links. However, reducing the safety factor increases the risk of structural failure especially due to overloading the robot.

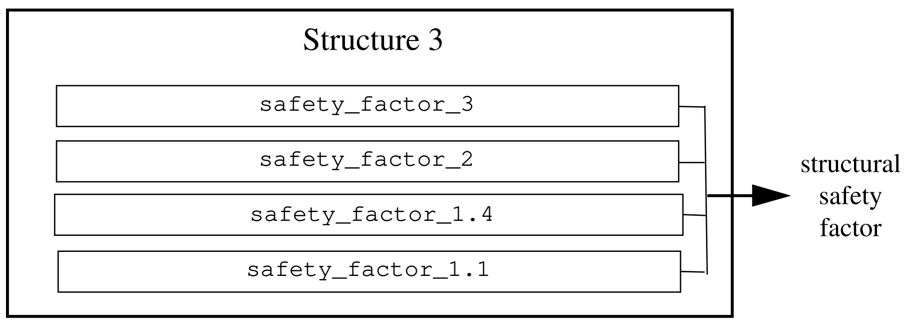


Figure 7-8. Structural Design Method 3

7.2.2.4 Design Method S-4

Design Method S-4 decides the shape of the cross section of the links. The choices are hollow round or hollow square shapes. While for the same weight the square shape has a higher stiffness [Rivin 88, p. 128], the circular shape is more suited for revolute joints and is less expensive.

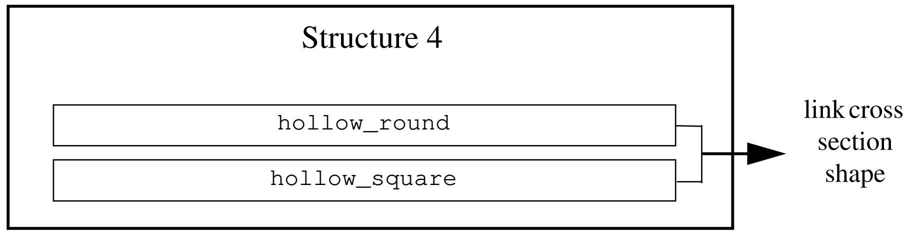


Figure 7-9. Structural Design Method 4

7.2.2.5 Design Method S-5

Design Method S-5 has one algorithmic design approach that uses Equation 2-14 on page 36 to calculate the deflection of the tip of the robot. The deflection calculated is for the worst case in which the arm is fully stretched and the maximum load is being carried.

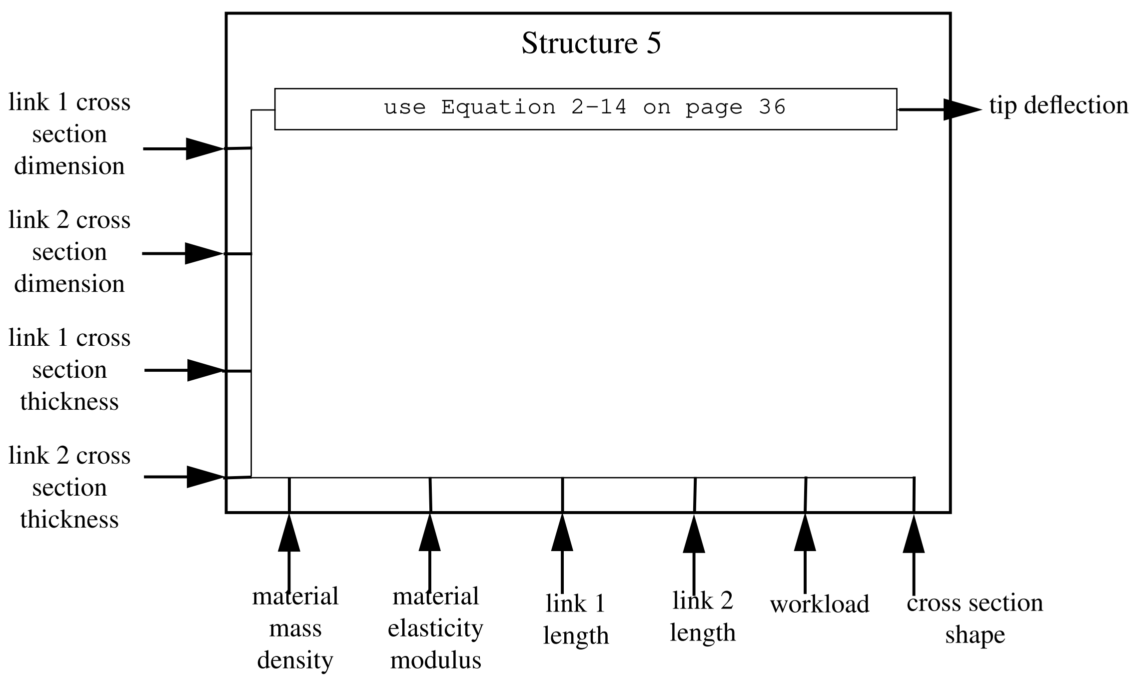


Figure 7-10. Structural Design Method 5

7.2.3 Control Design Methods

There is one control design method C-1 that decides the gains of a PD controller for each joint (see Figure 2-5 on page 38).

7.2.3.1 Design Method C-1

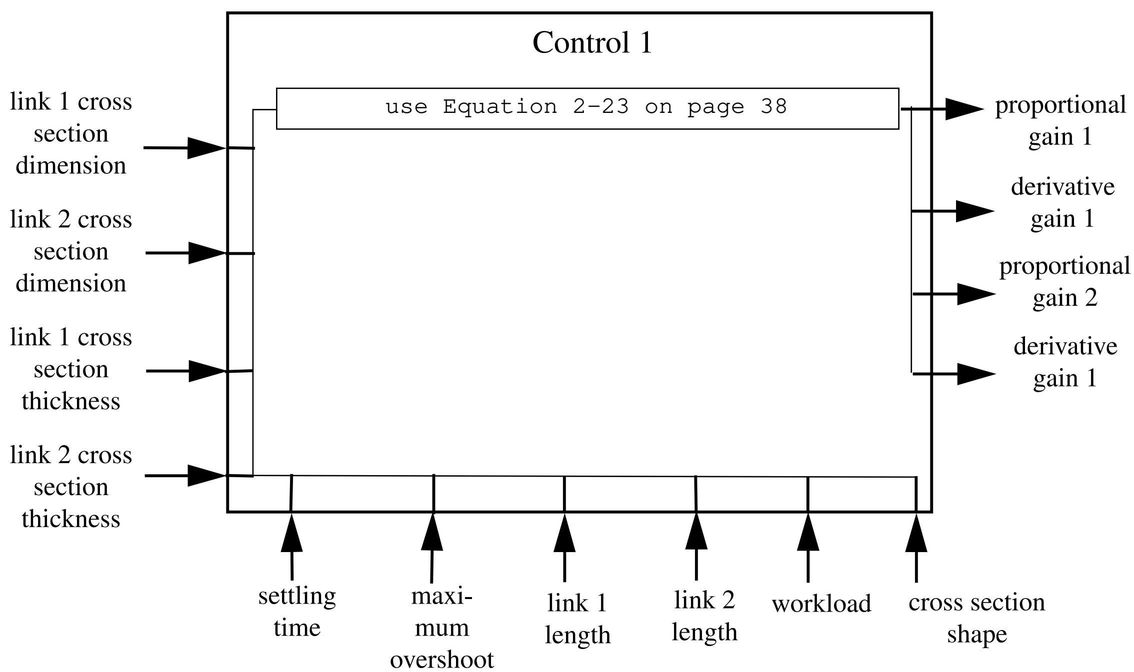


Figure 7-11. Control Design Method 1

7.3 Design Process Flowchart

The control of the flow of the design process is based on the design cycles that were discussed in "Concurrency" on page 94. A design cycle starts when the set of designers at a specific depth in dependency graph are asked to design. At the end of each design cycle the results of the design are checked against the constraints. If the results satisfy all the relevant constraints the corresponding design cycle is interpreted as successful otherwise it is labeled as unsuccessful. Figure 7-12 shows how the design process is moved through a design cycle.

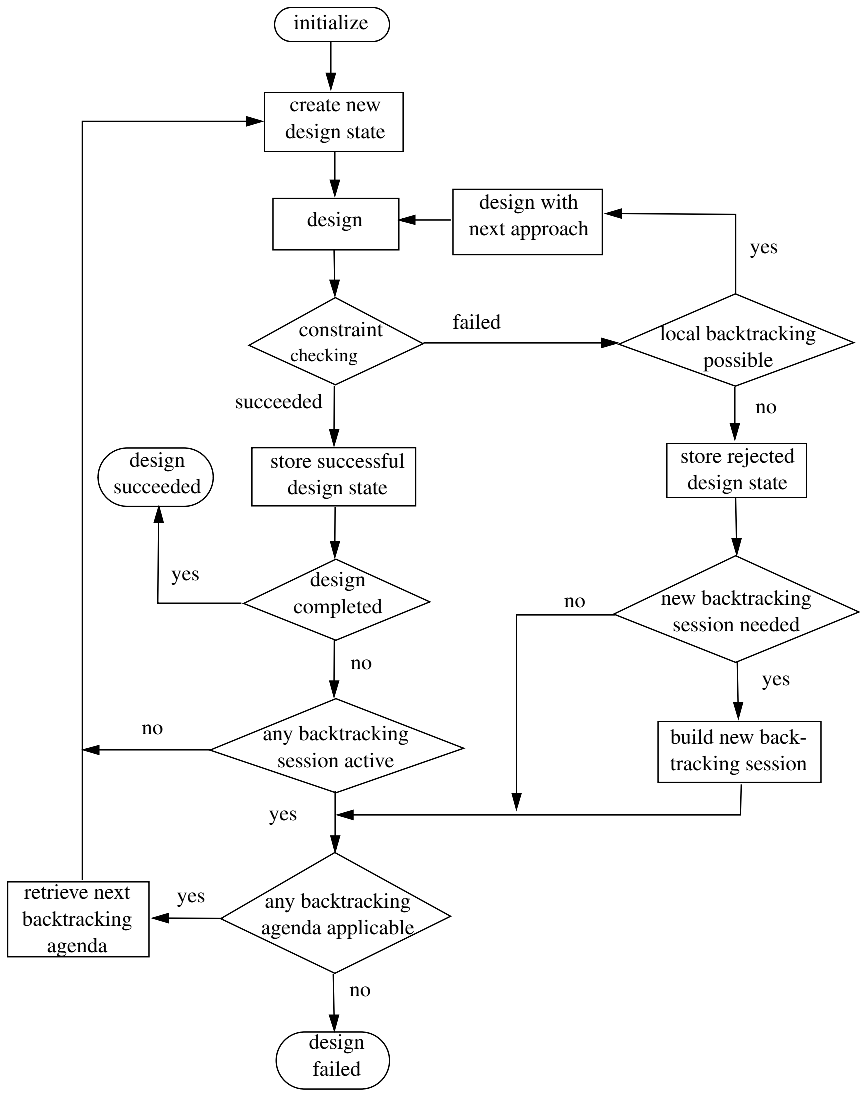


Figure 7-12. Flowchart of the Design Process.

Each design cycle happens at a certain depth in the dependency graph. In each depth the designers are independent from each other meaning that they do not use the others output. As a result designers in this depth are able to design concurrently. On the other hand design cycles help to manage the backtracking process. Different backtracking agenda are built for different design cycles. Backtracking sessions order the execution of design cycles during the course of backtracking. In section "Backtracking" on page 141 we will talk extensively about backtracking.

7.3.1 Dependency Graph vs. Cycle Tree, and Design Cycle vs. Design State

Figure 7-13 shows an example of a dependency graph that is dynamically generated during the design process. The design process starts with a set of design methods (implemented as designer agents) that can use the design requirements and generate a set of values for their output parameters (designers 1 to 3 in Figure 7-13). This set of designers form the first *row* of the dependency graph with *Depth 0* in the graph. Based on the input-output dependency between the design methods a new set of designer agents step forward and generate values for their output parameters. As a result of this process new rows are added to the graph in new depths until the design is complete.

A constraint violation on any of the generated values for the design parameters causes the process to backtrack to shallower depths of the graph to take a different path (see "Design Path" on page 105). We will discuss the backtracking process in more detail in "Backtracking" on page 141.

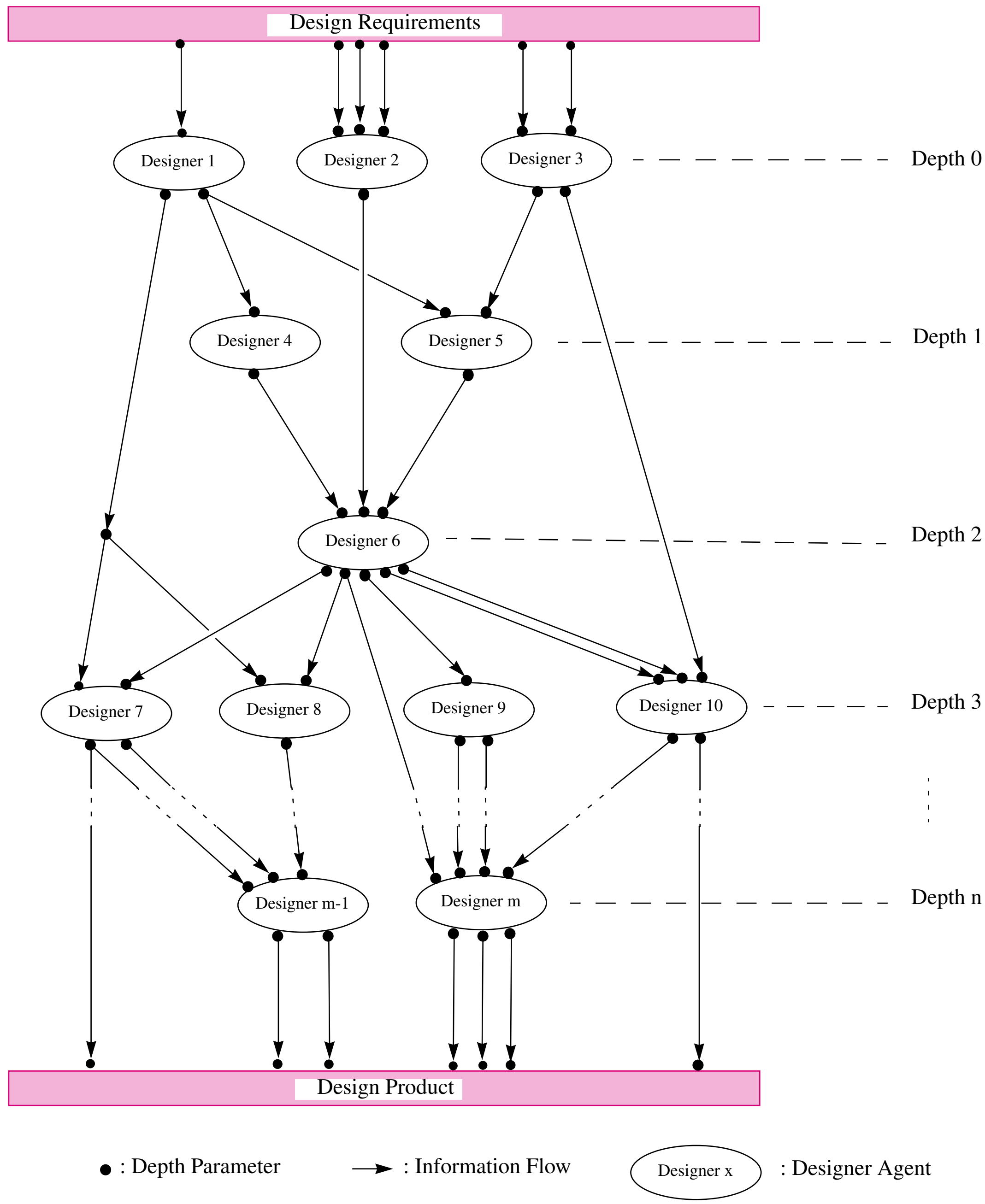


Figure 7-13. Dependency Graph

The design methods in a row of the dependency graph are executed in parallel because there is no dependency between them. All design agents that are in one row of the dependency graph do their design simultaneously in one design cycle (see "Consistency" on page 94). A new design state is initialized at the beginning of each design cycle and evolves during that cycle. A design state (except at Depth 0) is initialized to the last successful design state at one shallower depth and keeps a reference to the that design state as its parent. Each design state keeps a list of its children states (except the leaves). At the end of the cycle if all the relevant constraints are satisfied the corresponding design state is tagged as successful, otherwise it will be stored as an unsuccessful design state. The design process then backtracks to shallower depths in the dependency graph to try other alternative design approaches. Therefore, the design process does not pass through a depth in the dependency graph unless the design state at that depth is successful. During this back and forth process between different depths, design states are generated one after another in a sequence. However, based on their corresponding depth in the dependency graph the design states form a tree type of structure too that is called the *Cycle Tree*.

In **RD,** `DesignState` agent creates new design states, keeps track of design states and forms the cycle tree. The `DependencyProvider` agent is responsible for building the dependency graph and provides the depth of each designer agent in the tree to other agents. It is worthy to note that in general the dependency graph might change due to opportunistic participation of designer agents during the design process. That is, in different design iterations different designer agents might become applicable, hence generating a different dependency graph. However, if the dependency graph is not changed compared to previous iterations the design process will not be opportunistic, but based on an agenda generated by the backtracking mechanism. Following an agenda imposed by the system is necessary to make the backtracking process exhaustive.

7.3.2 Posing Design Goals

Coordinator agents are responsible for posing abstract goals, decomposing them into sub-goals, and requesting that other agents to achieve those sub-goals. Coordinator agents decide what the other agents should accomplish in order to eliminate any need for negotiation between different agents in the system. There are three coordinator agents in the system shown in Figure 5-3 on page 100: `Coordinator`, `DesignersCoordinator`, and `DatabaseCoordinator`.

The `Coordinator` agent has the most abstract goal in the design process, that is to achieve a design that satisfies the design requirements and constraints. Figure 5-2 on page 95 shows how the `Coordinator` conducts the design process in a loop until it finds either a satisfactory design or it fails to find a design that satisfies the requirements and constraints.

Inputs to a designer agent become available either by the user as design requirements or by other designers as their outputs. Designer agents use their first approach to generate a design unless there is a failure (i.e., constraint violation). When a failure occurs, designers re-design based on a backtracking agenda that is dictated by the `DesignersCoordinator` agent. The `DesignersCoordinator` agent prepares and enforces the re-designing agenda so that all possible combinations of design approaches are considered. In either case (with or without failure), design approaches are combined together in a sequence that starts to form a path from the designers at the root of the dependency graph to those in the leaves.

The number of possible paths is the product of the number of design approaches in all designers. Different paths are explored using a depth-first search algorithm. The system fails to produce a design if there is no path (i.e., no combination of design approaches) that satisfies all the design constraints.

7.4 Constraints

Design constraints define the criteria for acceptance or rejection of the partial designs that are generated by designer agents. A constraint between some variables can be implemented as a function that can return true or false based on the inputs to the function [Serrano 92].

Different types of constraints that are applicable to numeric or symbolic values can be defined in the system. A constraint is violated if its parameter's value is not a member of a set. The set of acceptable values might be pre-defined or dynamically change during the design process based of the change in the values of the design parameters. Design constraints may have been extracted from the design domain in order to satisfy physical constraints or to impose boundaries on some features of the product (e.g., cost, weight, etc.) that control the goodness of the design product.

7.4.1 Types of Constraints

There are two types of constraints at the top level:

- *Symbolic*: e.g., the selected material should be either "steel" or "aluminium".
- *Numeric*: e.g., $10 < \text{length} < 20$.

For numeric type of constraints we could have two subtypes:

- *Discrete*: e.g., thickness can only be one of these values: {0.1, 0.2, 0.4, 0.6, 0.8}.

- *Continuous*: e.g., 10< length < 20.

For the continuous-numeric type of constraints the following subtypes are proposed. The symbols in the parenthesis are used in the system to denote a specific type of constraint in various input or output files. Variable x is the argument of the constraint. Variables a and b might not be fixed. The following paragraphs discuss these in more detail.

- *Equality*: x = a, (=)
- *Inequality*: x != a, (!=)
- *Less*: x < a, (<)
- *Greater*: x > a, (>)
- *LessOrEqual*: x <= a, (<=)
- *GreaterOrEqual*: x >= a, (>=)
- *BoundedExclusive*: a < x < b, (<<)
- *BoundedInclusiveGreater*: a <= x < b, (<=<)
- *BoundedInclusiveLess*: a < x <= b, (<<=)
- *BoundedInclusive*: a <= x <= b, (<=<=)

The argument of a constraint (the variable whose current value is checked against the constraint) can be a design parameter, a function of design parameters (e.g., summation of link lengths or the weight of the product), a variable that can be calculated from design parameters plus maybe some other parameters (such as the cost of the product, its manufacturability, etc.) or a variable that specifies a specific characteristic of the design process (such as the time that has been spent on design).

Variables a, and b define the set of acceptable values or the boundaries of that set. These variables might be needed during run time. As an example of a constraint that needs to change its acceptable values on the fly, consider the constraint on dimensions of the cross section of the link of the robot. The diameter of the cross section of the link cannot be larger than a certain fraction of the same link length. But the length of the link may change in design iterations, that, as a result changes the upper bound of the constraint on the diameter. We define the set of acceptable values for this constraint by the maximum acceptable ratio of the diameter to the link length (e.g., 1/20).

7.5 Backtracking

Backtracking is a mechanism for recovering from failures by throwing away the recent results, going back to a previous state, and taking another path. In cases where design decisions lead to a constraint violations, "a problem solver needs the ability either to backtrack to correct bad decisions or to maintain parallel solutions corresponding to the alternatives at the stuck decision point. However, if alternative guesses exist at each point, and there are many such decision points on each solution path, a commitment to examine every possible combination of alternatives proves unwieldy" [Marcus 92]. The better approach is to change only those design decisions that affect the violated constraint: *Dependency-Directed Backtracking*.

We assume that each constraint applies only to one parameter (that is each constraint has only one argument) and each design parameter is produced by no more than one designer. Figure 7-14 shows part of a design process in which some design methods produce values for some design parameters based on some other parameters. Assume that one of the outputs of ‘Design Method 7’ violates a constraint. To satisfy this constraint the value of the design parameter should be changed. There are two possibilities to change the value of the parameter that caused constraint violation: 1) try another design approach in the design method 7, 2) change the inputs to Design Method 7 by trying other approaches in design methods that produced those inputs.

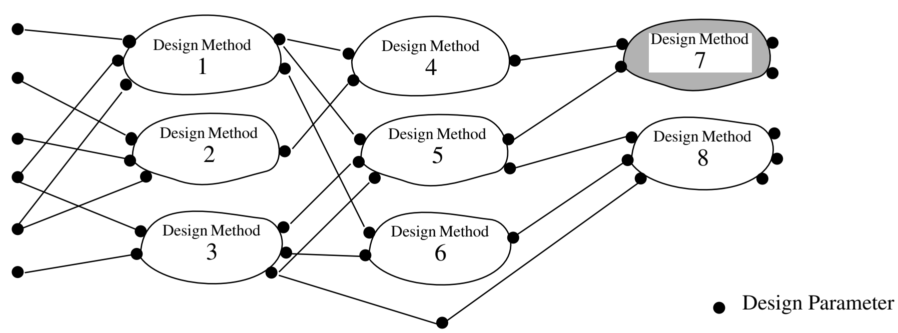


Figure 7-14. Design Methods Produce Values for Design Parameters

Figure 7-15 shows how the need to change the value of a design parameter in order to satisfy a violated constraint can propagate back to previous design parameters. This propagation obviously is a problem in the sense that it may cause other constraints, that were satisfied previously, to be violated. As a result we prefer to fix the violated constraints in a way that minimizes this propagation.

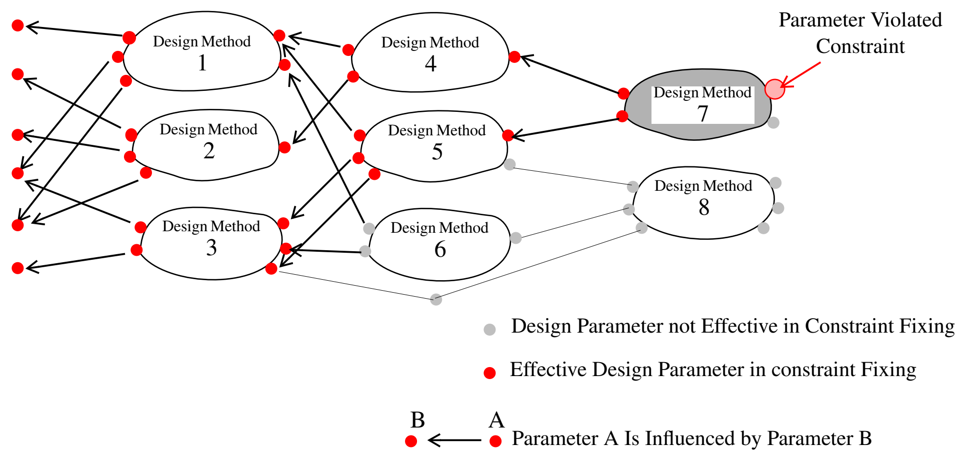


Figure 7-15. Possible Changes in Design Parameters for Fixing Constraint Violation

7.5.1 The Effect of Smaller Design Methods

It is worth noting that the smaller the design methods, the lesser the effect of propagation. One of the characteristics of small design methods is that their number of inputs and outputs is small. Consequently the dependencies between design methods are as direct as possible—that is, if the input to a design method does not really affect the output that is supposed to be changed, that parameter will not be considered for prospective change. Therefore one of the paths for propagating the changes is eliminated. This is shown in Figure 7-16.

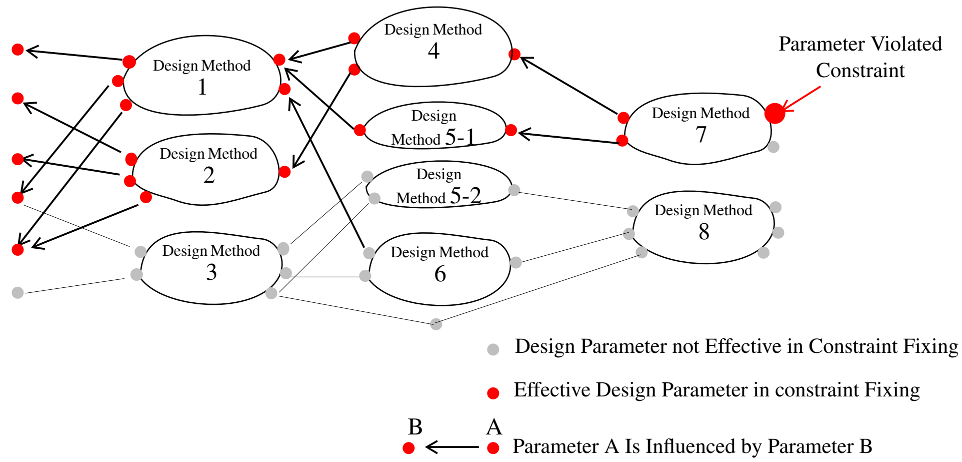


Figure 7-16. The Effect of Smaller Design Methods in Reducing Prospective Changes

The benefit of having small design methods becomes more obvious when we consider the situation in which fixing a violated constraint requires backtracking over two or more steps. For large design methods such changes propagate drastically down the stream and affect the parameters that were not even effective in constraint fixing. This type of propagation of changes may cause new constraint violations as is shown in Figure 7-17.

Figure 7-17 also shows that smaller design methods partition the set of design parameters into a larger number of subsets. As a result, the number of subsets that might be able to stay out of the changes increases. In Figure 7-17, the output of Design Method 5-2 might remain unaffected if the backtracking process does not change the inputs of that design method.

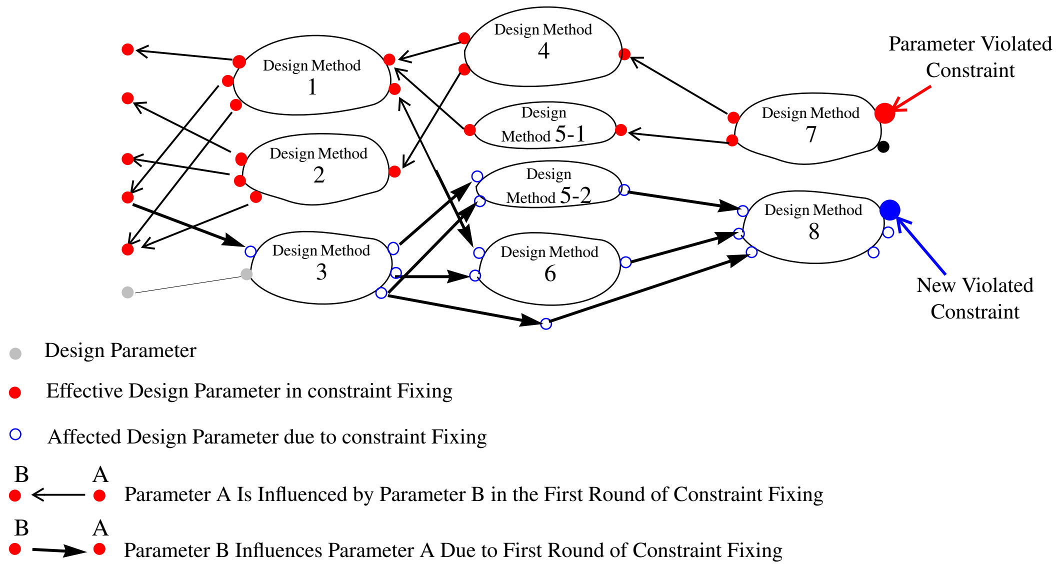


Figure 7-17. The Effect of Changes in Producing Possible New Constraint Violations

Now consider the case that Design Method 3 is broken into two smaller design methods. Figure 7-18 shows that the result is a considerable reduction in propagation of changes, hence reducing the danger of violating the parameters that were already satisfying the constraints. This reduction is a result of the Design Method 3-2 becoming independent of the changes that could happen in order to fix the violated constraint. In Figure 7-17 Design Method 3 would have to re-design if its input parameter was changed in order to affect the violated constraint. In Figure 7-18, however, part of the Design Method 3 that could stay away from changes, i.e., Design Method 3-2 will not be affected.

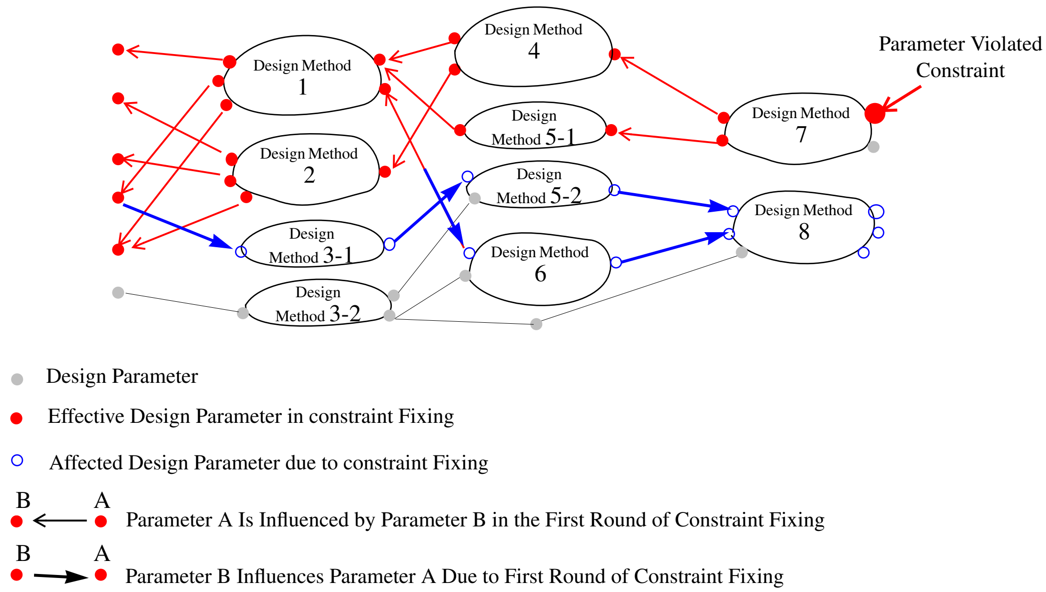


Figure 7-18. The Effect of Smaller Design Methods in Reducing

7.5.2 Factors Contributing to the Complexity of Backtracking

To be able to make progress in the design process (producing values for unassigned design parameters) all the active constraints should be satisfied in each design cycle. If there is more than one violated constraint in a specific design cycle all should be fixed before moving forward. One factor that contributes a great deal in making backtracking a complicated process is when possible changes for fixing two or more violated constraint happen at different levels. Consider the scenario of Figure 7-19 in which two constraints are violated. The first violated constraint can be fixed by employing a different design approach by Design Method 4. The second violated constraint can be fixed by Design Method 1. The problem is that this change affects Design Method 4 too that as a result might neutralize the attempt to fix the first constraint. Of course, there is no way to predict the effect of these changes on the constraints until they are actually executed. However, this type of situation can be detected before doing the re-design. It can be handled in a way that as a first attempt Design Method 1 re-designs and if it fails to change the status of Violated Constraint 1, Design Method 4 re-designs. If re-design by Design Method 1 fails to fix Violated Constraint 2, it cannot be accepted as a way to fix Constraint 1.

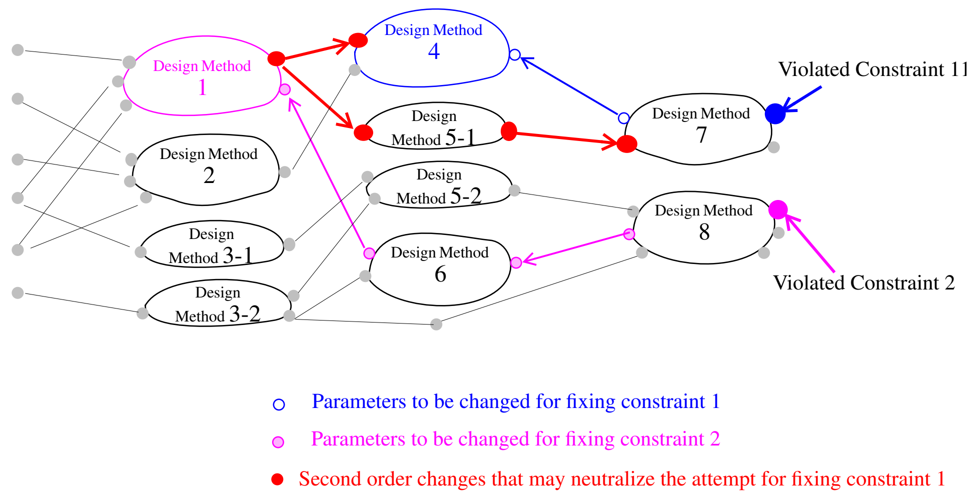


Figure 7-19. Factors Contributing to the Complexity of Backtracking

Backtracking can become a very complicated process if there is more than one input to each violated constraint. Especially, when one of the inputs to a constraint depends on the other inputs to the same constraint. Because attempting to change one of the inputs automatically changes the dependent input too. As a result these changes may neutralize each other so that the constraint stays unchanged (not satisfied) even though its inputs have changed.

Another complicated situations happens if the multiple inputs to a constraint are not produced in the same cycle of design. Assume that there are three violated constraints. Examining their input parameters and the designers that produce those parameters, reveals that for re-design in the first designer the system should backtrack one step, for the second designer two steps, and for the third designer three steps. It is obvious that the system should backtrack three steps to try to re-design. Then it should check to see if this re-design will affect the two designers' outputs that are input to the other two violated constraints. We prefer to re-check the other two violated constraints without trying other approaches to produce different values because the approaches are ordered based on their cost or optimality.

In this chapter we described the building blocks of the multi-agent system for design of a robot arm called Robot Designer (**RD**). RD will be used to conduct experiments simulating the design process of a robot arm. This chapter was based on chapters 2 and 5 and will act as a bridge between the previous chapters on general approach and the following chapters on implementation of the system. The next chapter describes the implementation issues and the contributions of this dissertation that concerns building an automated design system based on a multi-agent paradigm.

8 Implementation

8.1 Introduction

In this chapter we describe the software development stage for **RD**, the multi-agent design system for 2-DOF robots that we described in the previous chapter. Multi-agent systems are complex software systems that take a lot of effort and time to develop. The major reason for the difficulties of building multi-agent systems in general is that the area is new. More and more agent building shells and tools are becoming available as more experience is gained in the area.

We use the techniques from object oriented programming (OOP) to design and then implement the system. OOP is the natural choice for developing multi-agent systems due to many similar characteristics of objects (from OOP point-of-view) and agents [Shoham 93]. The implementation is done in Java, an object-oriented programming language which has some advantages over similar languages. Some of these advantages include a fast development cycle, a rich API (Application Programming Language), platform independence, and that it is multi-threaded.

The objects in **RD** can be categorized into two types: agent and non-agent objects. Agent objects are those that inherit from a superclass, naturally called `Agent`, that contains the generic components of an agent. Non-agent objects are mostly special data structures that model a concept whether design related or agent related. Some of the non-agent classes are: `Message`, `DesignParameter`, `BacktrackingSession`, `Event`, `Constraint`, `DesignCase` and many more. These objects bundle related pieces of information and may be passed between agents to transfer information.

The following sections describe the techniques developed and the experience gained during the implementation stage of **RD**. These techniques may be useful in building future multi-agent systems for design to reduce the effort and cost of development.

8.2 Agents in RD

Figure 5-3 on page 100 shows the agents that comprise **RD**.

8.2.1 Structure of an Agent

An agent is composed of some generic components for accomplishing common tasks (e.g., communication) and some specialized components for achieving its specific goals. The following are generic components of each agent:

1. *Message composer*: composes a message that is to be sent to one or more agents. Message composer receives the name of the receiver agent(s), a performative, and the message content.

2. *Message sender*: puts the messages that should be sent in a queue, finds the receiver of each message, and dispatches the messages to the receiver agents.

3. *Message receiver*: receives the messages from other agents.

4. *Message processor*: based on the type of the message received, it sends it to the right processing procedure to be processed.

5. *Observable*: sends notifications about internal events to other interested components of the same agent. For instance, when a message is processed the message processor dispatches a `message_processed` event that is of interest to the Logger. The Logger then makes a log of the processed message. Here 'Message Processor' acts like an observable and the 'Logger' is an observer.

6. *Logger*: records various internal events of an agent in different log files. Logger is also responsible for cleaning up when the agent is no longer needed.

8.2.2 `Agent` Object

All the agents in **RD** are derived from `Agent` object that provides the basis for the communication mechanism and other generic tasks. An agent that is processing messages has the `running` status and when it does not have anything to do has the `stand_by` status. The following is a list of the generic tasks that each agent inherits from the `Agent` object:

- Registering. Each agent has to register with the `Registrar` in order to be able to send and receive messages. Each agent registers by giving its name and an identification number to the `Registrar`.

- Composing Messages. An inner object of `Agent` called `MessageComposer` is in charge of creating messages. It can accept requests for different types and formats of messages with overloaded functions (i.e., a set of functions with the same name and functionality but different arguments). It assigns a unique identification number to each message.

- Sending Messages. `MessageSender` is another inner object of `Agent` that has two message buffers: `toBeSentMessages` and `sentMessages`. It finds the address of the receiver agent and puts the message into the received buffer of the receiver agent. An agent may send more than one message simultaneously therefore, the message is first put into the `toBeSentMessages` queue and when it is actually dispatched it is transfered to the `sentMessages` buffer. A report then is given in the interface of the agent and a log is made of the sent message.

- Receiving Messages. `MessageReceiver`, an inner object of the `Agent,` has a message buffer for the received messages. It also creates a new thread of execution for each message received and starts that thread. "The term thread is shorthand for thread of control, and a thread of control is a section of code executed independently of other threads of control within a single program" [Oaks 97]. A log is made when a message is received by an agent.

- Processing Messages. `MessageProcessor`, the inner object that starts processing the received messages, has the following message buffers for each stage of processing: `processingMessages`, `pendingMessages`, `processedMessages`, `ignoredMessages`. If the message is not a generic message, i.e., it cannot be handled in the `Agent` object itself, it dispatches it to the agent-specific message handlers. Examples of generic tasks are `achieve_show` (meaning: show your interface) and `achieve_clean_up` (meaning: stop processing messages, make logs of the current events and terminate yourself).

- Suspending and Resuming. The tasks that need a service from another agent are suspended and then resumed upon the receipt of the response. Each task that needs a response from another agent, assigns a maximum time that it will be waiting for the answer. If the response takes more than the maximum time, the agent sends an exception message to `ExceptionHandler` agent and terminates execution of that task. This mechanism can easily be modified so that the agent sends its request to another agent that might be able to provide its assistance. This is a way to prevent the system from halting due to non-responding or slow-responding agents.

- Creating logs of the activities of the agent. These activities include message passing process and other agent-specific events.

- Handling Exceptions. Whenever an exception happens the agent prepares a message about the exception and sends it to the `ExceptionHandler` agent for further processing.

- Clean-up. The most generic clean-up tasks such as making a log of the agent's activities is done in the agent object. The clean-up is required when the agent is asked to wrap up its activities and terminate itself.

Each agent has a generic interface through which it gives reports about what it is doing at the current moment. Figure 8-1 shows the interface of `Designer-K-1`. The agent is in `stand_by` status, the top portion of the window shows the recent messages that the agent has received, the middle section shows what activities the agent is currently doing, and the bottom section shows the messages that the agent has been sent. The only agent that accepts inputs from the user is `DesignRequirements` agent, through which the user enters the requirements and the parameters for constraints. If this information is provided through input files, the `DesignRequirements` agent is not used.

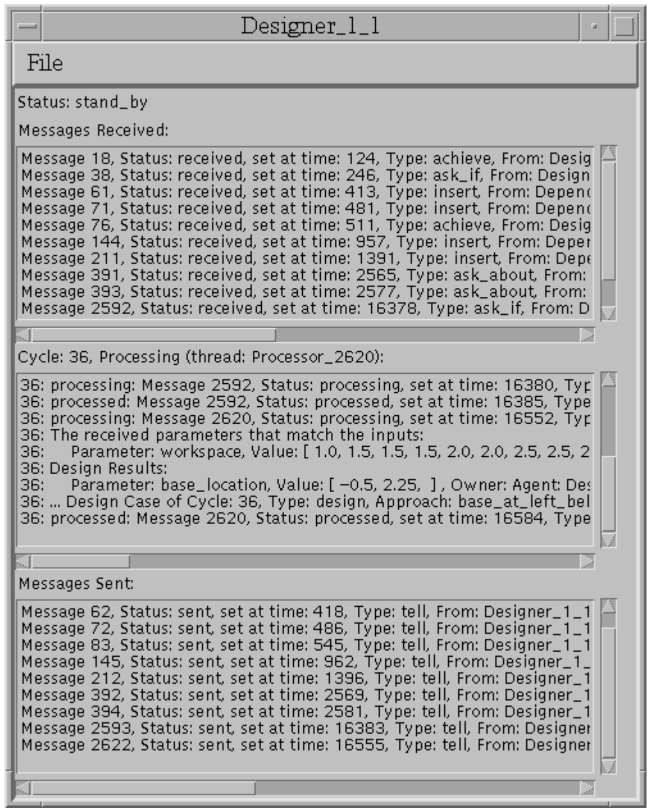


Figure 8-1. Interface of an Agent.

In the following paragraphs we review the implementation of some of the most important agents in **RD**.

8.2.3 `Coordinator` Agent

`Coordinator` is the agent in charge of controlling the design process at the highest level of abstraction. Its goal is to find a design that satisfies the requirements and the constraints.

It delegates the tasks of *designing*, *evaluating* partial designs, *backtracking*, and *updating* the design state to other agents. The `Coordinator`'s tasks are the following:

- **Initializing.** Initialization includes creating other agents, creating the first design cycle, sending initialization requests to the other two coordinator agents that are `Designer-sCoordinator` and `DatabaseCoordinator`.

- **Managing.** That is to conduct the design process by assigning some abstract tasks to other agents. Based on the information that other agents send to `Coordinator` (i.e., as a result of carrying the tasks that were assigned to them by `Coordinator`), it decides who should do what in the next step. This is a management job in which `Coordinator` makes other agents work while it oversees the whole process and makes high level decisions. The following are the tasks that `Coordinator` assigns to other agents:

    - Design. There are two situations in which `Coordinator` asks `DesignersCoordinator` to carry another cycle of design: in the first design cycle and whenever the previous design cycle has been successful in satisfying constraints.

    - Evaluating. Upon receiving the results of the "Design" request `Coordinator` asks the `Evaluator` to check the results from the current design cycle against the constraints.

- Backtracking. If the results fail to satisfy all the relevant constraints, `Coordinator` decides to backtrack to the previous decisions in order to fix this failure. Please note that, by backtracking to the previous decision making points, part of the result is destroyed. As a result, the current partial design is always consistent with the constraints.
- Re-design. When the setup for backtracking is complete the `Coordinator` requests a "re-design" from the `DesignersCoordinator`. If no more backtracking is possible, the `Coordinator` decides to terminate the design process. The difference between a 'design' and a 're-design'" request is that in the design case the designer agents participate in the process in an opportunistic manner. In 're-design', however, the process rolls back to a specific depth in the dependency graph in which some specific designer agents are asked to re-design. After the backtracking process is complete the normal 'design' process resumes. This dependency-directed backtracking prevents the designer agents that won't have any effect on violated constraints from participating in the design process, thus saving time.
- Updating. If the evaluation result is successful the `Coordinator` asks the `DatabaseCoordinator` to update the current partial design by adding the new results.

• **Termination.** The `Coordinator` terminates the design process in any of the following cases:

- When the design is complete.

- When no answer can be found that satisfies the requirements and the constraints.
- When an exception happens in the design process that is of type 'error' (as opposed to 'warning' exceptions).

The termination process includes the following steps:

- Asking the `Tracer` agent to record the information about the last design cycle and other information about the process (e.g., termination time, duration of the process, the total number of messages exchanged, the memory spent, etc.).
- Asking all other agents to clean up and terminate their pending tasks, if any.

8.2.4 `DesignersCoordinator` Agent

The `DesignersCoordinator` agent is the manager of designer agents. It works closely with both the `Coordinator` and the designer agents. It re-routes the design requests from the `Coordinator` to designers. The `DesignersCoordinator` is also in charge of managing the backtracking process among designer agents. `DesignersCo-ordinator` keeps the following lists that are updated at the beginning or during a design cycle:

- `designersWhoShouldBeAskedIfCanDesign` is the list of designers that is set at the beginning of each design cycle and includes designer agents that in this cycle should be considered for designing. This list does not include those designer agents that have already designed and thus are at the shallower depths of the dependency graph.

- `designersWhoCanDesign` is the subset of `designersWhoShouldBeAsked-IfCanDesign` list that in response to question of if they can design have answered positively. That is, these designers had all the necessary information for designing.

- `designersWhoCannotDesign` is the subset of the `designersWhoShould-BeAskedIfCanDesign` list that cannot design because they do not have enough information to start designing.

- `designersWhoAreChosenToDesign` is a subset of `designersWhoCanDe-sign` list that are actually chosen to design. In the case of backtracking some of designers that can design might not affect the violated constraints and as a result there is no point in repeating the previous designs that will not be included in `designer-sWhoAreChosenToDesign`.

- `designersWhoAreNotChosenToDesign` is the list of those designers that could design but could not resolve the constraint violation. Notice that these are designers whose inputs has not changed since the last time they designed. Therefore, while they do not have any effect on the violated constraints, the others do not have any effect on their inputs.

- `designersWhoAreDesigning` is the list of designers that at this moment are designing.

- `designersWhoDesigned` is a list that gets gradually filled during the course of a design cycle with those designers that finish designing. When the lists of `designer-sWhoDesigned` and `designersWhoAreChosenToDesign` become the same `DesignersCoordinator` informs `Coordinator` of the results that the designer agents have generated.

Having found the `designersWhoCanDesign`, the `DesignersCoordinator` agent adds one more level to the dependency graph. Therefore, the dependency graph is dynamically built as the process of design continues. The dependency graph is used in the process of backtracking to prepare a backtracking agenda.

The `DesignersCoordinator` plays a major role in conducting and managing the backtracking process. It creates new backtracking sessions and updates the backtracking agenda for the current backtracking session. It also prevents the same paths in different sessions from being repeated. For this purpose the `DesignersCoordinator` agent keeps a log of previous paths that were tried for all backtracking sessions. If the backtracking process switches to another session the backtracking agenda is compared with the log to make sure that the proposed path has not been taken before.

Adding a new designer agent to RD is as easy as adding its name to the list of designers in `DesignersCoordinator`. An instance of the new designer is created by `DesignersCoordinator` at the beginning of the process. During the process the new designer along with all other designer agents participate in the design. The participation of designer agents in the design is encouraged by the `DesignersCoordinator` by sending various messages to them.

8.2.5 Designer Agents

Designer agents implement the design methods that were introduced in "Design Methods for Robot Design" on page 123. Designer agents implement the design approaches of the corresponding design method in the form of procedures and functions. Design approaches are prioritized based on the preferences in the domain or based on general criteria such as cost, execution time, information needed, etc.

The implementation can be enhanced so that the ordering of design approaches can even change dynamically based on the results of the system while running. For instance, if the system detects that an iterative approach is taking more time, it can re-arrange the approaches so that the approach is used as the last resort.

Adding design approaches to designer agents is very easy and includes the following steps:

- the procedure that implements the design approach is added to the agent,
- any extra design parameter that might be needed for the new design approach is added to the list of input parameters, and
- the name of the new approach is added to the list of available design approaches considering the right order.

The major service that designer agents provide is, naturally, doing design that is to receive some design parameters as input and generate values for some other design parameters. A designer agent accepts requests for doing design in the following ways:

- design with the first approach,
- design with the current approach,

- design with the next approach,
- design with a specific approach,

During the backtracking process designer agents are asked to follow the backtracking agenda that dictates what design approach they should use. The reason for introducing the concept of a backtracking agenda is to make sure that every possible path (i.e., resulting from combining different design approaches, see “Design Path” on page 105) has been examined. The agent in charge of preparing and executing the backtracking agenda is the `DesignersCoordinator` agent. The `DesignersCoordinator` agent sends messages based on the backtracking agenda using different forms of design request service from the above list.

Design requests do not have to come necessarily from the `DesignersCoordinator` agent. Any agent that sends an appropriate design request message will receive the response from designer agents. After an agent requests a design service, the designer agent sends a request to `DesignState` agent for the current state of the design in order to use the most recent design parameters as input.

A designer agent keeps the history of its previous design cases. A design case for a specific designer is comprised of the set of input values, the set of output values, the design approach that was used, the design cycle that was created, and other bookkeeping details. The idea of keeping design cases for each designer agent is to be able to use the multi-agent design systems as a sensitivity analysis tool too. That is, each designer keeps track of how their output parameters changed as a result of change in the inputs or change in the design approaches.

Keeping the history of previous design cases might even be useful in saving time by avoiding repetition of same cases in different design projects. That is, if the designer agent receives a request for design that has been done before, it can retrieve the results from the design cases without repeating the calculations or design procedures. This is very close to the well known area of Case-Based Reasoning in design [Maher 95].

A designer agent keeps track of which designers have been supplying input to it and which agents have been consuming its outputs. The agent that is interested in this service is the `DependencyProvider` that puts pieces of information gathered from different designer agents together to build the dependency graph. The dependency graph is updated for each cycle of design to include the changes that might have happened in the list of designer agents that participated in the design process.

The other services that a designer agent provides are as follows:

- to look at the current design state and tell whether it can design or not,
- to report its current depth in the dependency graph, and
- to report the history of the design cases that they have done so far for off-line analysis.

The `Designer` object is the superclass of all objects that implement the designer agents. Some of the important attributes of the `Designer` object are as shown in Table 8-1:

Table 8-1. Attributes of `Designer` Object.

Attribute	Description
`inputs`	the list of input design parameters
`outputs`	the list of output design parameters
`designCases`	the list of previous design cases
`currentCase`	current design case
`designApproaches`	the list of design approaches

Table 8-1. Attributes of `Designer` Object.

Attribute	Description
`depthInDependencyGraph`	current depth in the dependency graph,
`inputSuppliers`	list of designers that supplied the input
`outputConsumers`	list of designers that consumed the output

8.3 Implementation of Messages

In the following we list the different types of messages that the system can handle. The different types of messages in the list are based on the KQML specifications (Knowledge Query and Manipulation Language) [Finin 93]. We have borrowed many ideas from the KQML specifications as well as Haddadi's work [95, Chapter 5] to design the messaging mechanism. However, the implementation is specific to our system. The reason is that the KQML specifications or the framework proposed by Haddadi are too general to be useful for direct implementation.

- `tell`. This is a "for your information" message in which the sender agent sends a statement to the receiver to express a fact about the contents of its knowledge base. The sender will not expect a reply to this message from the receiver. Examples of this type of message that have been used in **RD** are:
    - `(tell: design_results)`, that the `DesignersCoordinator` agent sends to the `Coordinator` agent to let it know about the results of the recent designs done by designer agents.
    - `(tell: constraints_satisfied)`, that the `DesignConstraints` agent sends to `Evaluator` agent to inform it that the design results satisfy the relevant constraints.

- `ask_about`. The sender agent asks the receiver about any relevant statement in the receiver's knowledge base. The sender agent expects a reply to this message from the receiver.

- `ask_if`. The sender agent asks the receiver if the statement is true in the receiver's knowledge base. The sender agent expects a reply to this message from the receiver.

- `achieve`. The sender agent makes a requests that the receiver accomplish a task. The receiver will inform the sender agent when the task is complete. The response message may include the results of the action too. If the receiver agent is not capable of doing the requested task it sends a `sorry` response back to the sender.

- `insert`. The sender agent asks the receiver agent to either add or replace the attached statement to/in its knowledge base. The sender agent does not expect any reply to this message.

- `sorry`. An agent sends a `sorry` message to another agent in response to a request that it cannot understand or process.

8.3.1 `Message` Object

Table 8-2 shows the attributes of the message object:

Table 8-2. Attributes of the `Message` object

Attribute	Description
`id`	a unique identification number
`sender`	agent sending the message
`receivers`	receiver agents
`performative`	the type of the message
`content`	of type MessageContent
`inReplyTo`	the `id` or `replyWith` of the original message if this is a reply message
`replyWith`	the `id` that should be used when responding to this message (used for messages that are to be broadcasted)
`status`	at any time one of the following status: created, to_be_sent, sent, received, processing, processed, pending, or ignored
`tag`	any additional information, e.g., why the message was sent
`timeCreated`	the time the message was created
`timeToBeSent`	the time the message was entered into the ToBeSent message buffer
`timeSent`	the time the message was actually dispatched
`timeReceived`	the time the message was received by the receiver agent
`timeProcessing`	the time the processing of the message started
`timeProcessed`	the time the processing of the message completed
`timePending`	the time the message was put in pending status
`timeIgnored`	the time the message was ignored

8.4 Implementation of Backtracking

A backtracking session starts when a set of violated constraints cannot be resolved. A backtracking session is uniquely defined by the set of designer agents that might be able to resolve the violation by changing their decisions. If during the backtracking process a new set of constraints are violated that might change the set of candidate designers for re-design, a new backtracking session replaces the old one. If the backtracking session is successful in resolving the set of violated constraints, it is terminated. There is a variable that holds the current backtracking session in the design process. The `DesignersCoordinator` agent and all designers have an attribute that refers to this variable. It is null for the `DesignersCoordinator` and the designers agents if the design is in regular 'forward-tracking' mode. It is also null for designers that do not participate in the current backtracking session.

A set of violated constraints always has a unique set of corresponding candidate re-designers. This is because as long as the dependency relationship between designers is not changed, the set of designers that affect violated constraints remains the same. As a result, when a set of violated constraints re-occurs, backtracking will continue from the point that it stopped last time. Otherwise an infinite loop may happen.

Backtracking designers are the set of designers that may be able to affect the set of violated constraints, hence resolving them. Backtracking designers affect the violated constraints by providing input to the designers that have violated their constraints. This effect can recursively propagate up to the designers that provide input for the first set of backtracking designers.

A backtracking session includes all designers that can potentially be backtracking designers. All the backtracking designers are arranged based on their depth in the dependency graph. There are two reason for this arrangement. First, in order to make the backtracking an exhaustive process, an untried design approach (alternative design decision) by a backtracking designer should be combined with all possible design approaches of the backtracking designers in the deeper levels of the dependency graph. Second, in order to make backtracking an exhaustive process, a new design approach (alternative design decision) by a backtracking designer should be combined with all possible design approaches of the backtracking designers in the same level in the dependency graph.

A backtracking process starts by creating a new backtracking session followed by identifying all backtracking designers and arranging them based on their depth in the dependency graph. All the designers that directly or indirectly (i.e., through intermediate designers) influence the violated constraint(s) are identified as backtracking designers. This information is extracted from the dependency graph. The new backtracking session is broadcasted to all backtracking designers. The first request for re-design is sent to one of the designers that is the deepest in the dependency graph. If there are several designers with the same depth, all the possible combinations of their design approaches is considered. For non-backtracking designers, their latest design case is used.

The question is why would we want to re-try those design approaches in a designer that have led to constraint violation before? The answer is that it is not the case that a design approach that has caused (directly) or contributed to (indirectly) to a constraint violation will violate that constraint again. The reason is that it is the produced value that determines constraint violation and the produced value depends on the input to the designer too and not just on the design approach. In other words, it could be the case that the inputs to the designer are not the same as before and as a result the design approach that caused constraint violation before will not cause that violation again. Change of input values to a designer happens because of a re-design in the designers that provide input to the mentioned designer (that are in the shallower levels of dependency graph).

Even for the designers in the shallowest level of dependency graph whose inputs have not changed, re-trying design approaches that were rejected before should be considered in backtracking process. The reason is that those approaches may have not been combined with all approaches in the designers downstream in the dependency graph. Therefore to be exhaustive, they should be tried again. The only situation in which re-trying rejected design approaches will lead to a re-rejection is when the designers whose inputs have not changed (those with the shallowest depth) directly violate their own constraints.

One interesting behavior that was observed during the run of **RD** was when changing between different backtracking sessions. Backtracking sessions have different sets of backtracking designers. Different sets of violated constraints might result in different corresponding candidate backtracking designers. Backtracking designers are a subset of designers that could affect the violated constraints, and who therefore are participating in alternating their design approaches.

In the course of a design process sometimes it happens that a backtracking session is replaced by a new session because a different set of constraints are violated. The process continues based on a backtracking agenda that is provided by the new session. Later during the process the old set of constraints are violated again and as a result the old session takes over the backtracking process. A mechanism is implemented that prevents different sessions from repeating the paths that have been tried unsuccessfully by other sessions.

The deeper a designer is in the dependency graph the more expensive it is from a backtracking point-of-view. A design case is composed of the values of the designer's input and output parameters and the design approach that the designer used to produce the outputs. During the backtracking process designers generate different design cases by receiving different inputs or using different approaches. The number of design cases that a designer produces depends on how many different sets of inputs it receives and how many design approaches it has. The number of input sets that a designer receives is equal to the product of the number of design approaches of the designers that affect this designer (i.e., affects its inputs) all the way up to the root of the dependency graph. In other word, the number of design cases that a designer generates depends on its depth in the dependency graph.

As a result, the designers that are deeper in the dependency graph take a bigger portion of the backtracking process. Therefore the effect of an expensive design method (i.e., with respect to design time) in the deeper levels of dependency graph is much higher. In one example, designers at depth zero of the dependency graph in **RD** each produced 64 design cases; at depth one, 205 cases; and in depths three and four, 820 cases. That is, designers at depth three and four, design twelve times more than those at depth zero.

Conclusions such as above, can be incorporated into a general design methodology that humans could follow in their design processes. Design methods that are deeper in the dependency graph should be selected to be as cheap as possible—that is the design methods that are more expensive should be moved to shallower depths in the dependency graph if possible.

The above discussion might lead us to an approach on how to break down the design knowledge into pieces: We should break the design knowledge in a way that the most expensive methods are located in the shallower depths of the dependency graph.

Figure 8-2 shows an example of a backtracking process between 10 designer agents that are arranged in a dependency graph (see “Dependency Graph vs. Cycle Tree, and Design Cycle vs. Design State” on page 135). The design process starts by making the design requirements available to the collection of designer agents. After the designers in a row of the dependency graph finish their design the constraints are checked. If any constraint is violated, the backtracking takes over and decides what backtracking agenda should be followed. The backtracking agenda includes the depth to which the design process should backtrack. A new design state is initialized at that depth and the process continues until either a solution is found that satisfies all the constraints or the system fails to find such a solution.

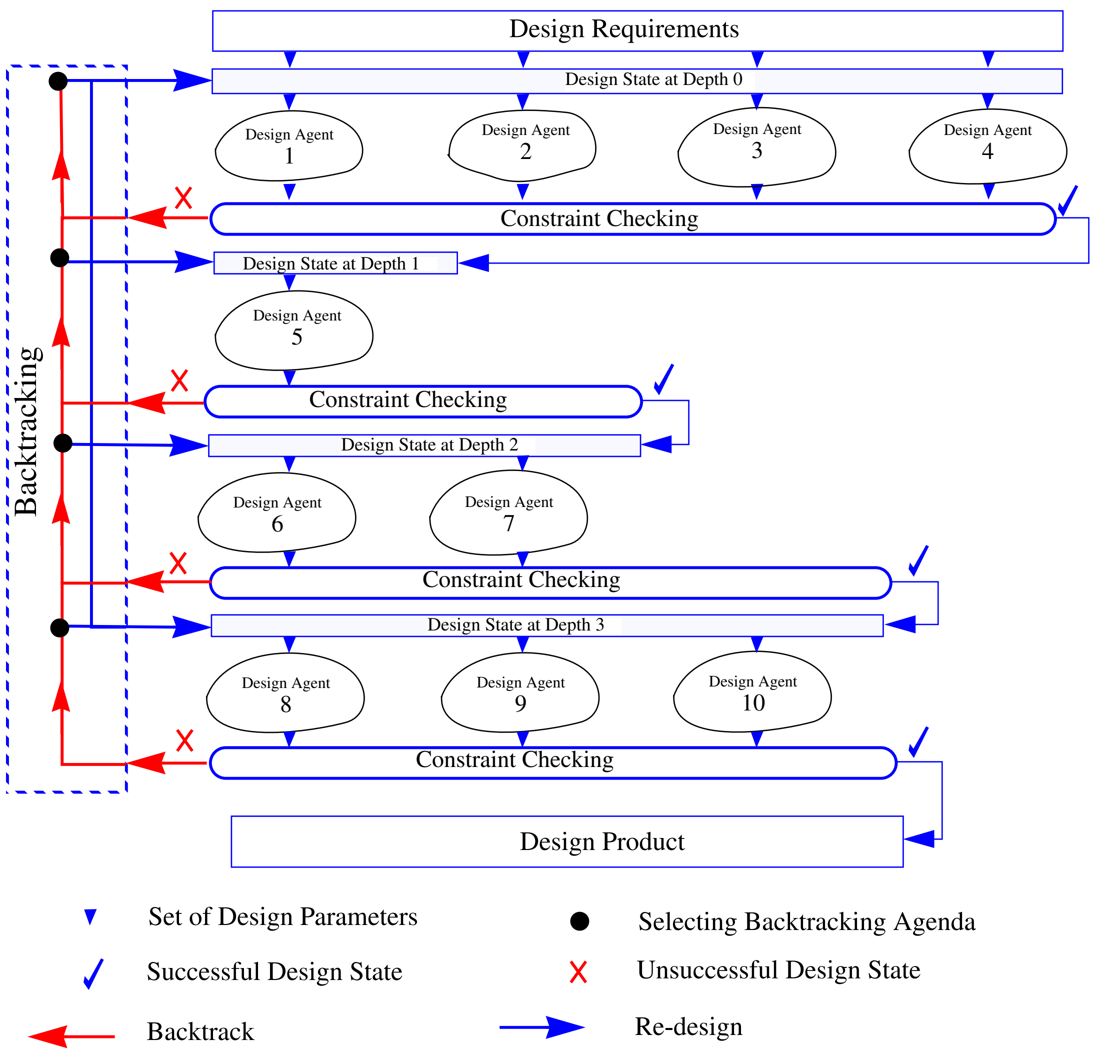


Figure 8-2. Flowchart of Backtracking Process.

8.4.1 An Algorithm for Backtracking

The algorithm for backtracking that is proposed in this section is the final version of a series of algorithms that evolved during the process of designing and implementing **RD**. The original algorithm was influenced a great deal by the backtracking mechanism of an expert system for designing elevators called VT ([Marcus 92]). Being exhaustive and fast are the two criteria that were used to modify the earlier versions of the backtracking algorithm in **RD**. Being general was later added to the list of criteria in order to make the algorithm applicable to situations that the set of designer agents that participate in the design process might change during the design process.

Assuming that each constraint applies only to one parameter (that is each constraint has one input) and each parameter is produced by no more than one designer, the following algorithm is proposed:

1. Reset the list of prospective re-designers;
2. For each violated constraint find the designer that can affect it and is the deepest in the dependency graph (that is the closest designer to the depth at which that constraint violation occurred). This rule is to reduce the propagation of the changes due to the re-design to other design parameters;
3. If there is at least one constraint that cannot be resolved by re-design (because there is no designer which can affect that constraint that has not exhaustively re-designed) then the design has failed;
4. Order all the designers in step (2) based on their depth in the dependency graph,

5. For the set of designers in (4) (ready for re-design) that have the lowest depth check if any of them affect the designers in (4) but at higher depths:
   - If there is such a designer, remove it from the list of (prospective re-designers),
   - If there is no such designer, move to the set of re-designers in the higher depth and repeat (5),
6. Roll back to the last design state in the depth of re-designers in (5) with lowest depth, create a new design state with its parent being the most recent design state at one level up,
7. Start re-design. For each designer downstream in the process check if it is supposed to re-design (that is if it is in the list of prospective re-designers).

In this chapter we described some of the implementation details of **RD**. This prepares us for the next chapter that discusses the experiments that we did using RD. In the next chapter we show how we defined the space of requirements and constraints. We also show the results of using RD in simulating the design process and provide some preliminary analysis of the results.

9 Experiments

In previous chapter we talked about the implementation details of the multi-agent design system for robot design: **RD**. In this chapter we describe the experiments that we did using **RD** to produce traces that in the next chapter will be used to generate design methodologies.

First we discuss how the range of design requirements and constraints were chosen. During this part we use the results of an extensive sensitivity analysis to see effect of different factors in the results of the system. We used the results of the sensitivity analysis also to verify the correctness of the system by checking to see if they conform to the first principles.

We then present the traces generated by the system and do a preliminary analysis to evaluate the opportunities for extracting methodologies from them.

9.1 Range of Requirements and Constraints

In this section we choose the range of values for the requirements and the constraints that defines the requirement space for the set of projects in the experiments. The range of requirements define the sub-set of requirement space that we select for experimenting with the system. Varying the constraints while keeping the requirements constant, brings the effect of constraints into the process of generating design methodologies (see Figure 6-3, "Different Constraints Produces Different Designs and Traces." on page 111).

In the experiments conducted, all the design parameters that serve as requirements were varied over different values. From the list of constraints, however we picked two of them to vary over a specified range. The reason we did not vary all the constraints is to reduce the number of projects that should be solved. Varying all the constraints each with two different values will produce a huge number of projects in the order of hundreds of thousands. Therefore, we needed to pick the most important constraints, i.e., the most sensitive constraints. Sensitive constraints are those where a change of their value will have the most dramatic effect on the different paths taken by the system. In the following section we present the results of a sensitivity analysis that we did with the help of **RD** to first choose the boundaries of requirements and secondly find the most sensitive constraints and their range.

9.1.1 Sensitivity Analysis

In order to be able to select the range of design requirements and constraints we need to get a picture of how design parameters change with changes in design requirements. That is, we need to do a sensitivity analysis in the domain in order to be able to determine the range of requirements that will lead to covering a large design space.

Similarly, we need to study the effect of design approaches in a designer agent on design parameters generated. Different design approaches should generate alternative solutions that are sufficiently apart from each other. The reason is that if different design approaches keep generating values that are almost in the same range, there is less chance that the violated constraints will be satisfied. Therefore, we need to make sure that picking different approaches (i.e., taking alternative paths) will actually generate different solutions.

For instance, as a result of sensitivity analysis we discovered that the design approaches in `Designer_S_1` are not generating sufficiently different designs. The design approaches of this designer were originally based on choosing different ratios for thickness to the dimension of the cross section of the link. Later in this chapter we will see that for ratios greater than 0.1 the change in the dimension of the link is negligible (Figure 9-5). In fact we found out that there is an optimum value for the ratio that is around 0.1. Consequently, we fixed the ratio of the thickness to dimension to 0.1. On the other hand we realized that the dimensions that are calculated based on stress analysis for the cross section of the links cause excessive deflection for the tip of the robot (with regard to the constraints). Therefore, we replaced the design approaches with the ones that are based on the ratio of the dimension of the cross section to the minimum required by the stress analysis.

For the sake of discussion we assume that the constant parameters of the constraints do not change from one project to another. Changes happen only to requirements. the requirements are: `workspace`, `workload`, `settling_time`, and `maximum_overshoot`. The last three of these parameters are scalar. Therefore the most straightforward method to vary them for different projects is to define a starting value, an increment, and the number of increments. The `workspace` requirement, however, cannot vary as simply as the others because it is a set of points in the plane. Some collective attributes could be related to the workspace points that are based on their distribution in the plane. A primary investigation of the design methods shows that the distribution of the points may not affect the result of design very much. What seems to affect the design more is how the points are stretched in the plane and the area that they cover. Therefore, we consider the two following collective attributes of the workspace based on the rectangle that includes the points:

- *aspect ratio*: that is the ratio of the width of the rectangle circumscribing the workspace, 'workspace rectangle', to its length assuming length is greater than width (see Figure 7-2, "Different Locations for the Base of the Robot." on page 126), and,
- *area*: the area of the workspace rectangle.

Now we are able to vary the workspace in the same manner as the scalar parameters by varying the above two attributes in a range.

The important point is how to pick the start and end points so that most of the design space is covered. That is, how to pick the limits on the requirements so that a large part of the trace space is covered, because we are interested in obtaining the largest set of traces that have led to some points in the solution space, without actually testing all possible requirements.

In the following sections we present some of the results of the sensitivity analysis that we did to find the proper range of values for requirements and to find the most sensitive constraints and their ranges.

9.1.2 Sensitivity Analysis on Control Gains

We found that the gain of the controller depends on many other design parameters. One of the design parameters that has the most effect on the gains is the dimension of the cross section of the links.

Equation 2-23 on page 38 shows that both proportional and derivative gains of the controller for each link is a linear function of I_{eq}, the moment of inertia of the links. On the other hand Equation 2-20 on page 37 shows that I_{eq} for the first link is a linear function of the masses of the first and second links, while I_{eq} for the second link depends only on the mass of its own. Finally, Equation 2-8 on page 34 shows that the mass of the links is a second order function of the dimension of the cross section of the link. Therefore we would expect to see a non-linear change in the gains versus dimensions of both links for gains of the controller for the first link. We also would expect to see that the gains for the second link does not depend on the dimension of the first link. Figures 9-1 to 9-4, that are based on the results of **RD**, verify the above conclusions about how the proportional and derivative gains for both links vary by changing the dimensions of the cross sections of the links.

Figures 9-1 to 9-4 show not only how the controller's gains change versus the dimension of the links, they also provide some candidate value for what the boundaries of the constraints on the gain should be. Kp1, Kd1, Kp2, and Kd2 are proportional and derivative gains for the first and the second links respectively. d1 and d2 are the dimension of the cross section of the first and the second links respectively. The figures show that among all the gains Kp1 has the highest probability of violating its constraint because it takes very large values.

To conduct the sensitivity analysis of Figures 9-1 to 9-4 the values of other parameters that would affect the gains of the controllers were kept constant. These parameters are: the link lengths (1.53m and 0.768m), material (steel), shape of the cross section (hollow round), workload (1 kg), settling time (3 sec), and overshoot (40%). These values are nominal—the way the gains change versus dimensions was not affected by changing these values.

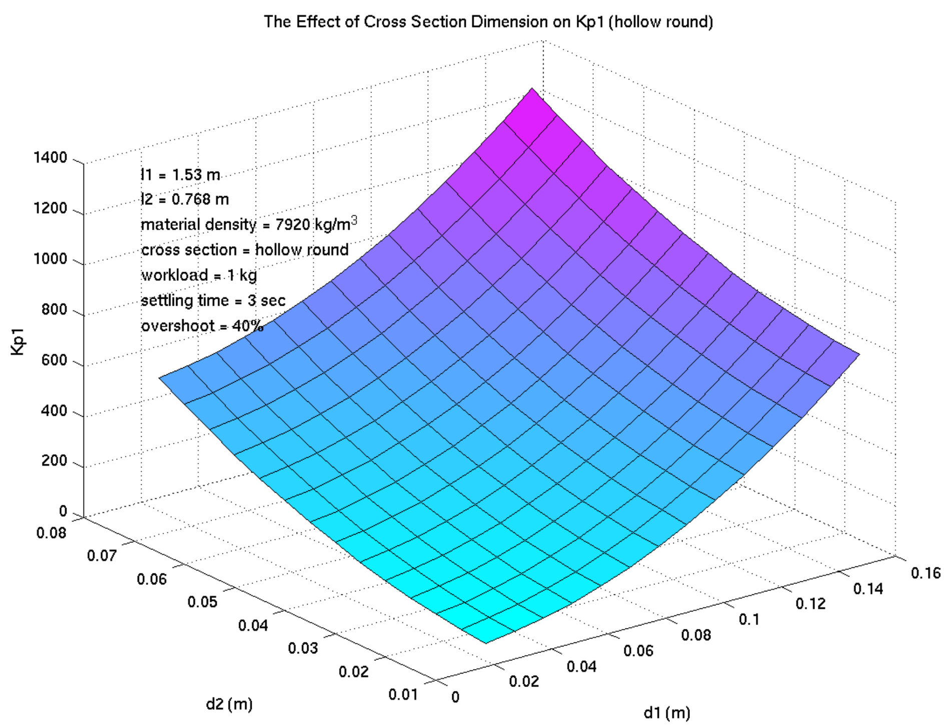


Figure 9-1. Sensitivity Analysis on Kp1 due to Changes in Cross Section Dimension.

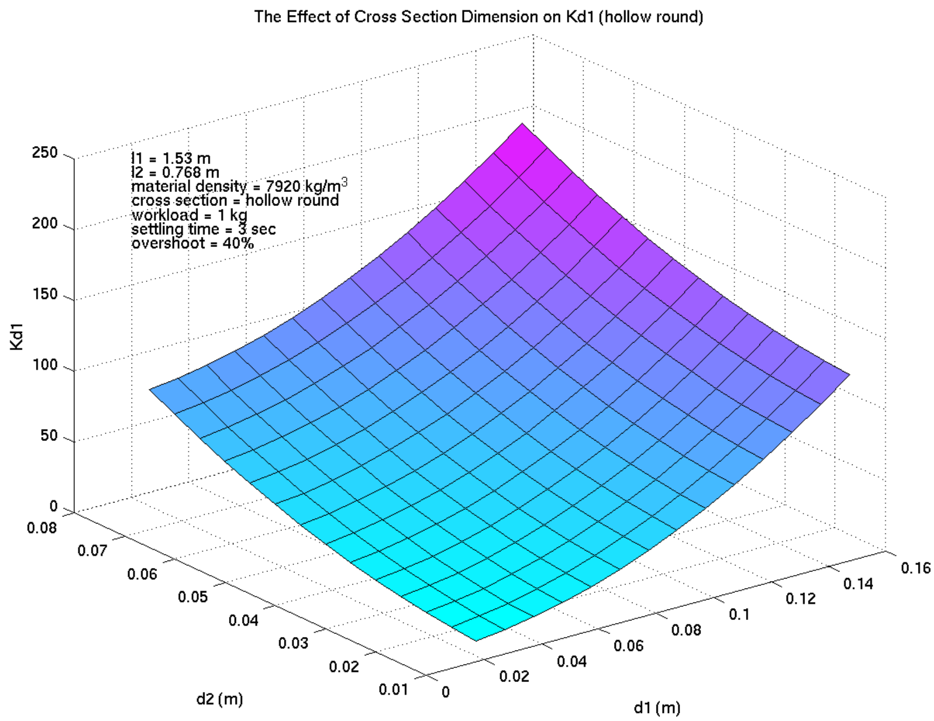


Figure 9-2. Sensitivity Analysis on Kd1 due to Changes in Cross Section Dimension.

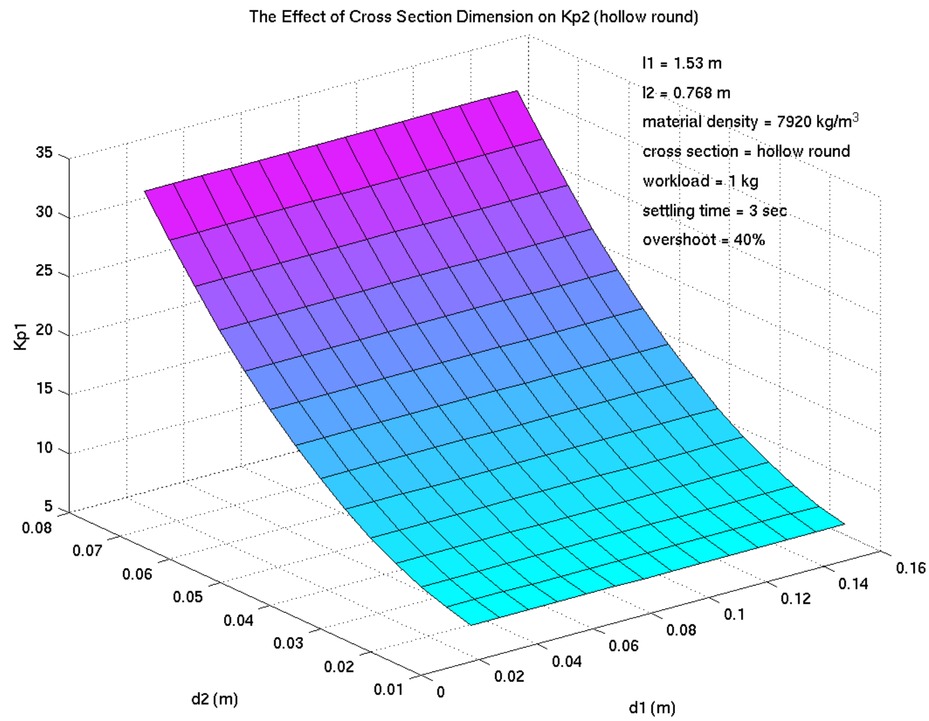


Figure 9-3. Sensitivity Analysis on Kp2 due to Changes in Cross Section Dimension.

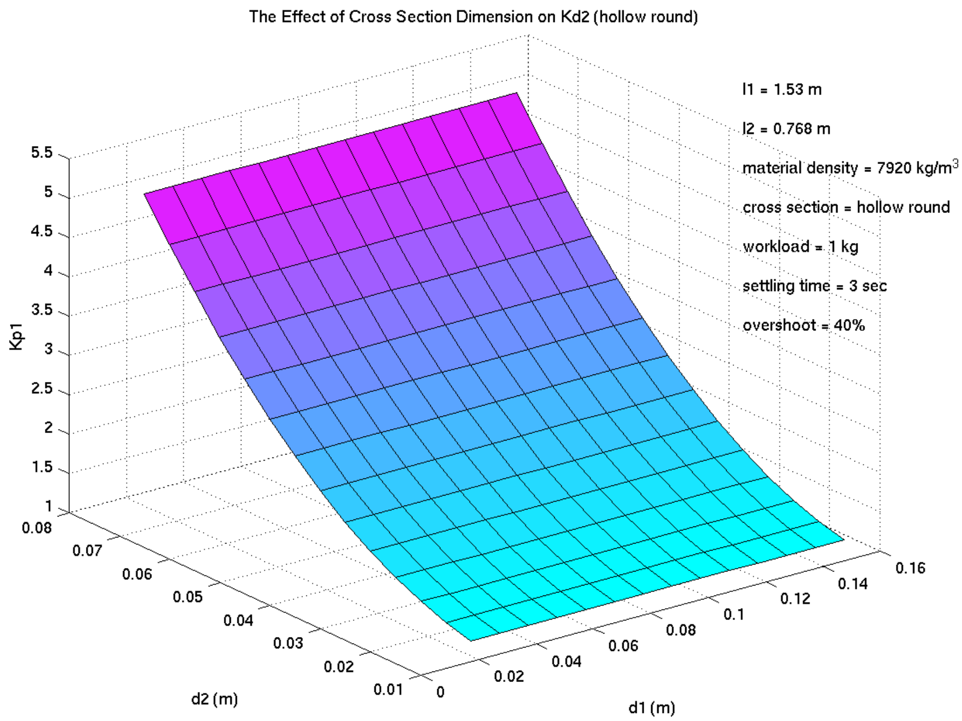


Figure 9-4. Sensitivity Analysis on Kd2 due to Changes in Cross Section Dimension.

Figure 9-1 to Figure 9-4 suggest that reducing the dimension of the cross section of the links reduces the controllers' gains. Reducing the cross section dimension of the links, however, will dramatically increase the maximum bending stress of the link (see Equation 2-6 on page 34). One way to keep the stress constant while reducing the dimension of the cross section (hence reducing control gains) is to increase the thickness of the cross section. Figure 9-5 shows how the dimension of the cross section is reduced by increasing the thickness/dimension ratio for a link with hollow round shape. Clearly there is an optimum value around 0.1 beyond which the increase in the thickness does not reduce the dimension but adds to the mass of the link. `Designer_S_1` uses this optimum value for structural design and finding the dimension and thickness of the cross section.

To conduct the sensitivity analysis of Figure 9-5 the values of other parameters that would affect the dimension of the cross section of the links were kept constant. These parameters are: the link lengths (1.53m and 0.768m), material (steel), safety factor (3), shape of the cross section (hollow round), and workload (1 kg). These values are nominal and do not affect the way the dimension changes versus the ratio.

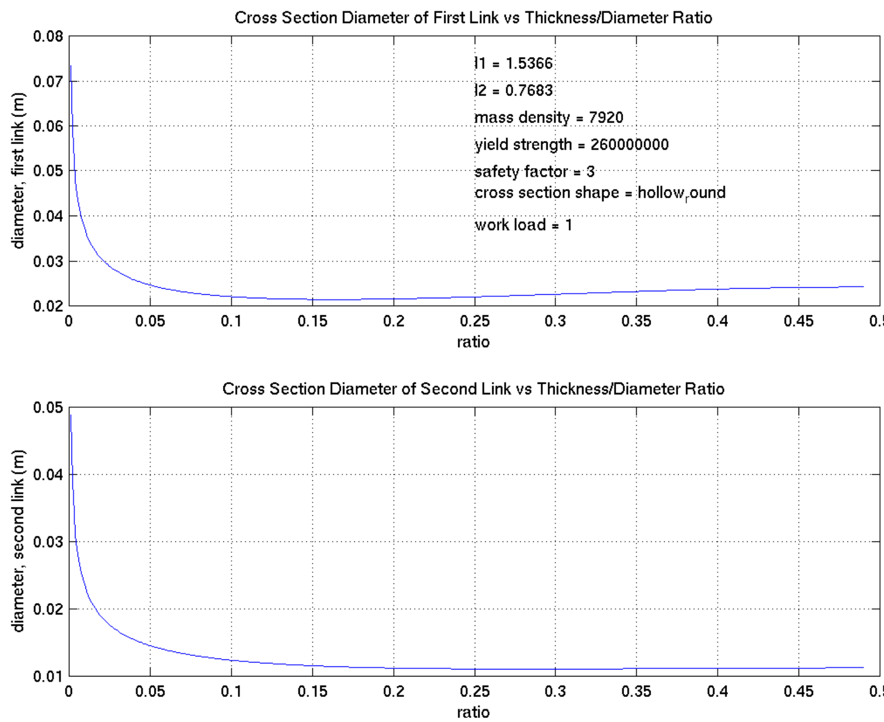


Figure 9-5. Sensitivity Analysis on Cross Section Dimension due to Thickness.

Figure 9-6 shows the same trend as in Figure 9-5 for a cross section with a hollow square shape. Therefore, we use the thickness to dimension ratio of 0.1 for both shapes.

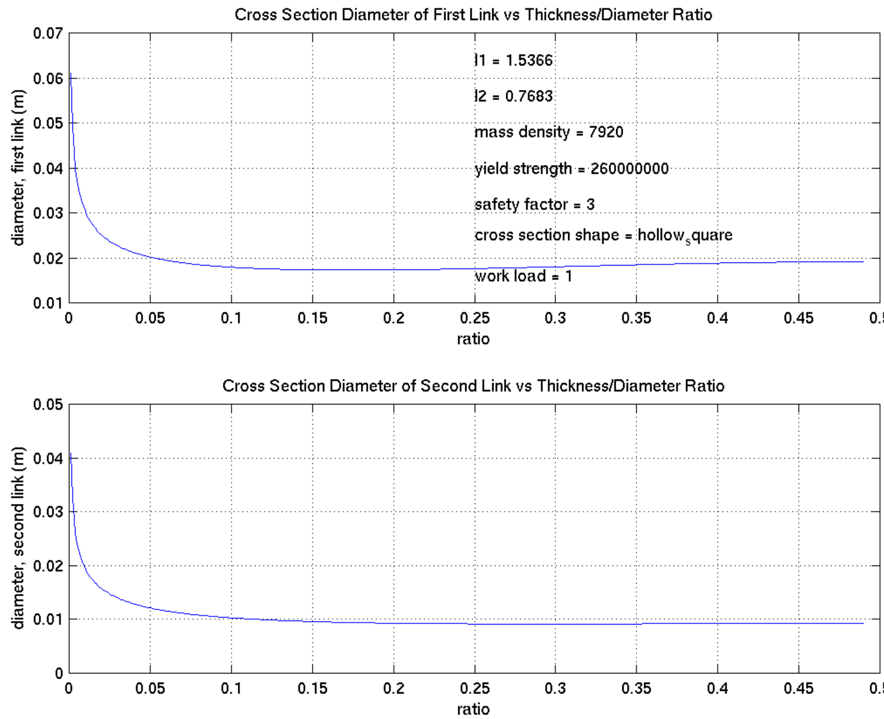


Figure 9-6. Sensitivity Analysis on a Hollow Square Cross Section Dimension.

Figure 9-7 shows that the dimension of the cross section of the links changes almost linearly with respect to the safety factor.

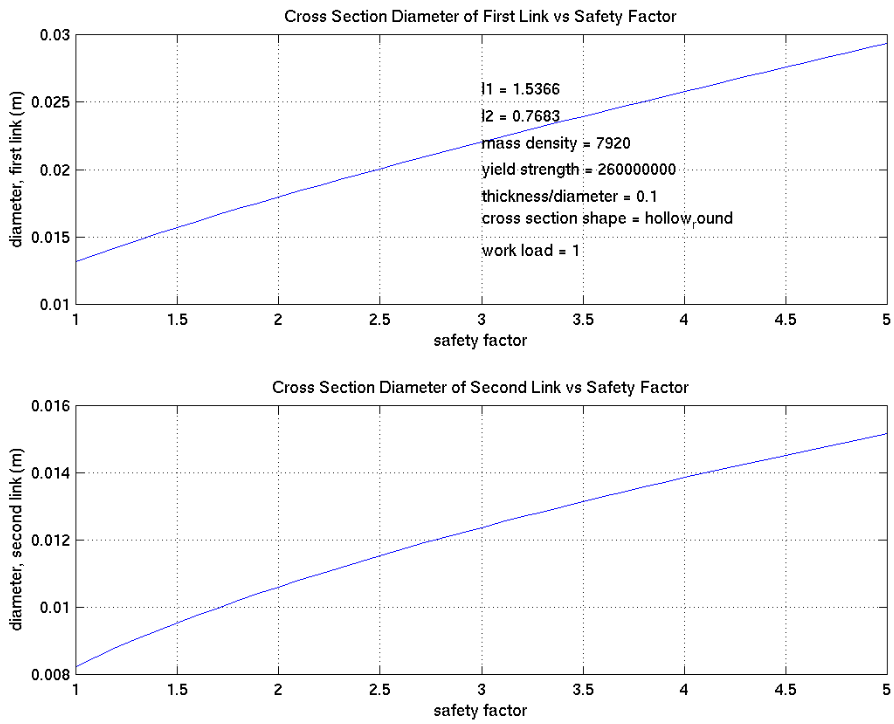


Figure 9-7. Sensitivity Analysis on Cross Section Dimension due to Safety Factor.

Figure 9-8 shows the effect of the dimension of the cross section on the deflection of the tip of the robot. It is clear that the deflection of the tip, similar to the proportional gain of the first controller, has a high chance of violating its constraint due to its exponential increase when the dimension of the cross section is reduced. Also, it was observed that for a large portion of the design space while the deflection of the tip grows to an unacceptable size, the stress level in the link remains in the safe range. One reason for this effect is that the cantilever configuration of the links produces large deflections. As a result, Designer_S_1 first picks the design approaches that produce large cross section dimensions for the links to avoid excessive deflection.

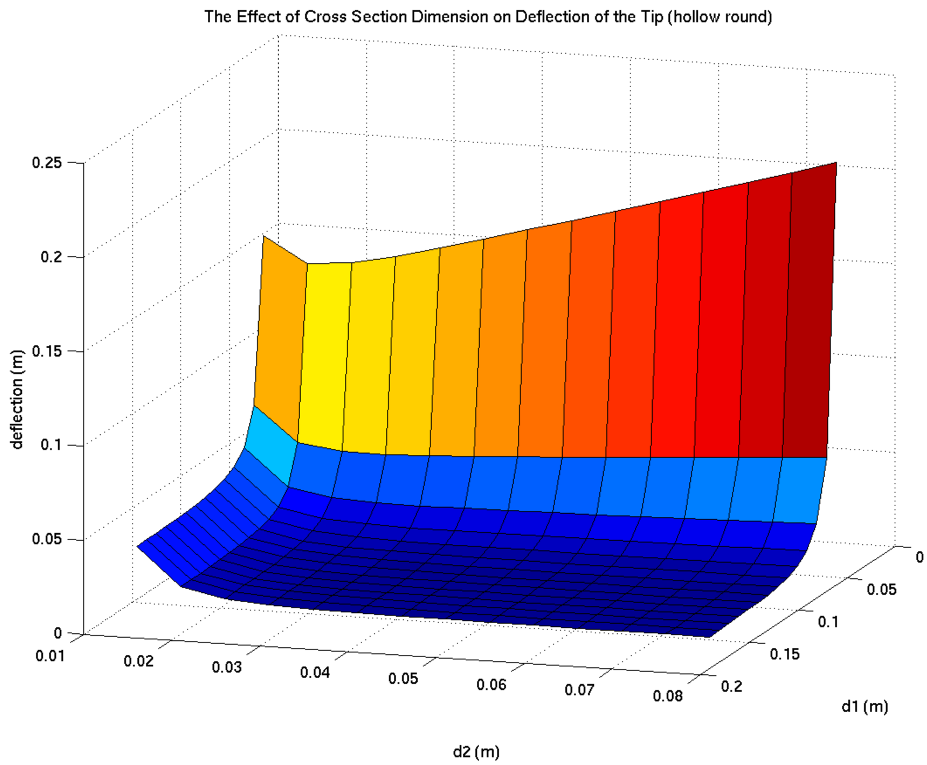


Figure 9-8. Sensitivity Analysis on Deflection of the Tip Due to Dimension of Cross Section.

Figure 9-9 shows the effect of the shape of the cross section on the deflection of the tip of the robot. Clearly a square cross section produces smaller deflection. The nominal values of the other parameters that affect the deflection of the tip are given in the graph and are the same for both round and square shapes.

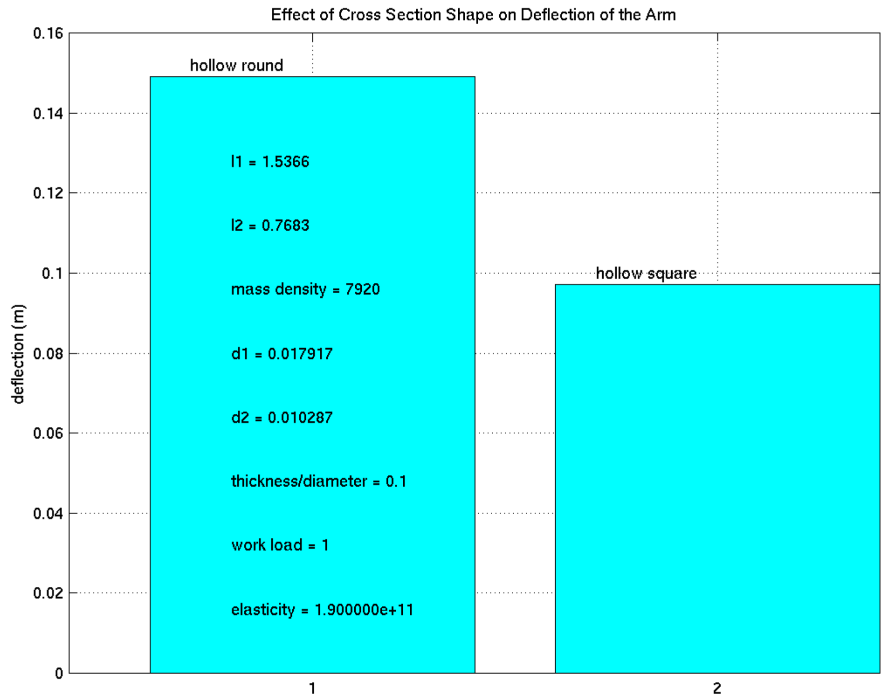


Figure 9-9. Sensitivity Analysis on Deflection of the Tip due to Cross Section Shape.

Figure 9-10 shows that links from steel have smaller deflections relative to links made from aluminum. Aluminum, however, is lighter, hence it produces smaller moments of inertia and smaller control gains. The values for the other parameters that affect the deflection of the tip were the same as shown in Figure 9-9 except that the shape of the cross section was fixed to circular and the material changed.

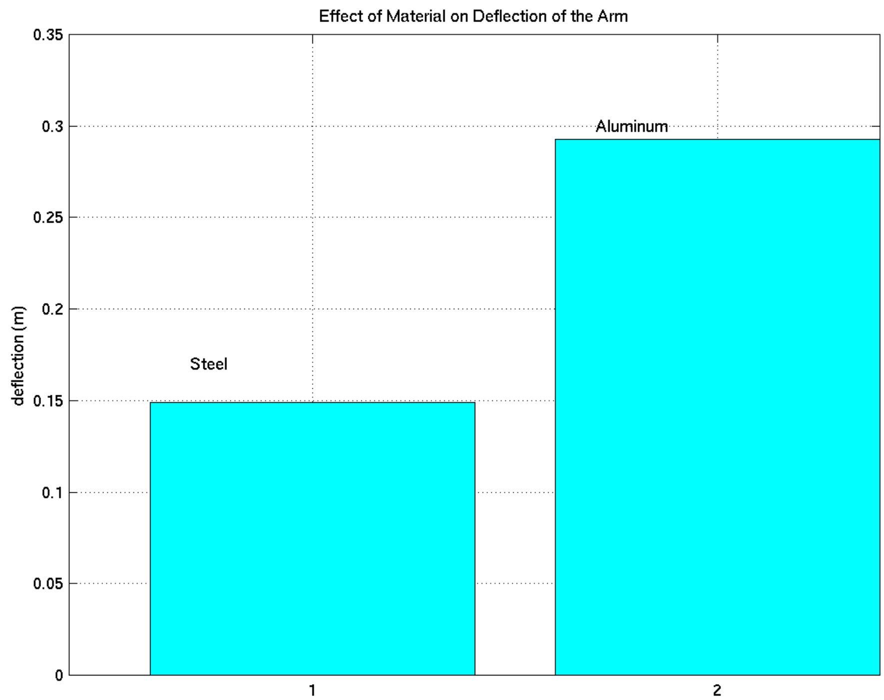


Figure 9-10. Sensitivity Analysis on Deflection of the Tip due to Material.

The most dramatic effect on the control gains is caused by the link lengths. Reducing the link lengths by half causes reduction in the gains for Kp1 from 1200 to 160, for Kd1 from 200 to 20, for Kp1 from 30 to 4.8, and for Kd2 from 4.8 to 0.75. The reason for this large effect is the multiple fold effect of the length on the moment of inertia of the links due to reduction of the mass and the length itself. As the result, minimization of the link lengths can become very effective in satisfying tight constraints. The minimization approaches, however, are very expensive in terms of the design time because of their iterative nature. Therefore, we push the design approaches in `Designer_K_2` that are based on minimization onto the end of the list of approaches. That is, we use the expensive approaches only after the less expensive design approaches cannot generate a satisfactory design.

The control gains decrease considerably by selecting aluminum over steel, assuming the same dimensions for the link. The reason is a much lower density for aluminum (2630kg/m3 compare to 7920kg/m3 for steel). The maximum Kp1 is reduced from 1200 to 450, for Kd1 from 200 to 80, for Kp2 from 30 to 15, and for Kd2 from 4.8 to 2.4 by switching to aluminum.

The control gains increase for a hollow square shape for cross section relative to hollow round shape, however, the magnitude of change is not as large as the other factors. For Kp1 from 1200 to 1500, for Kd1 from 200 to 280, for Kp2 from 30 to 37, and fro Kd2 from 4.8 to 7.5.

Many sensitivity analyses similar to what the above graphs show were conducted in order to find the appropriate range for design requirements. Table 9-1 shows the range of values for the requirements.

Table 9-1. Different Values for the Requirements.

`workspace` (see Table 9-2)	{"small-M", "small-L", "big-M", "big-L"}
`workload` (kg)	{1.0, 2.0, 3.0, 4.0, 5.0}
`settling_time` (sec)	{3.0, 2.0, 1.0}
`maximum_overshoot` (%)	{50, 40, 20, 10}

Table 9-2 shows the coordinate of the points in each workspace and Figure 9-11 shows the shape of the four different workspaces in the plane, hence the name chosen for them. In page 176 we described two collective attributes of the workspace, i.e., the aspect ratio and the area that are important. The 'M' and 'L' type of workspace have different aspect ratios. The size of the workspace, i.e., small or big takes the second attribute (area) into account. The combination of these two attributes for shape and size produces four different workspaces as shown in Figure 9-11.

Table 9-2. The Coordinates of the Points in Each Workspace (in meter).

small-M	x = [0.5 0.75 1.0 1.25 1.5 1.75 2.0 2.25 2.5] y = [0.25 0.5 0.75 1.0 0.75 1.0 0.75 0.5 0.25]
small-L	x = [0.5 0.75 0.75 0.75 1.0 1.0 1.25 1.25 1.25 1.5 1.5 1.75 1.75 2.0] y = [0.5 0.25 0.75 1.75 1.0 1.5 0.5 1.25 2.0 0.75 1.5 1.0 1.75 1.5]
big-M	x = [1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0] y = [0.5 1.0 1.5 2.0 1.5 2.0 1.5 1.0 0.5]
big-L	x = [1.0 1.5 1.5 1.5 2.0 2.0 2.5 2.5 2.5 3.0 3.0 3.5 3.5 4.0] y = [1.0 0.5 1.5 3.5 2.0 3.0 1.0 2.5 4.0 1.5 3.0 2.0 3.5 3.0]

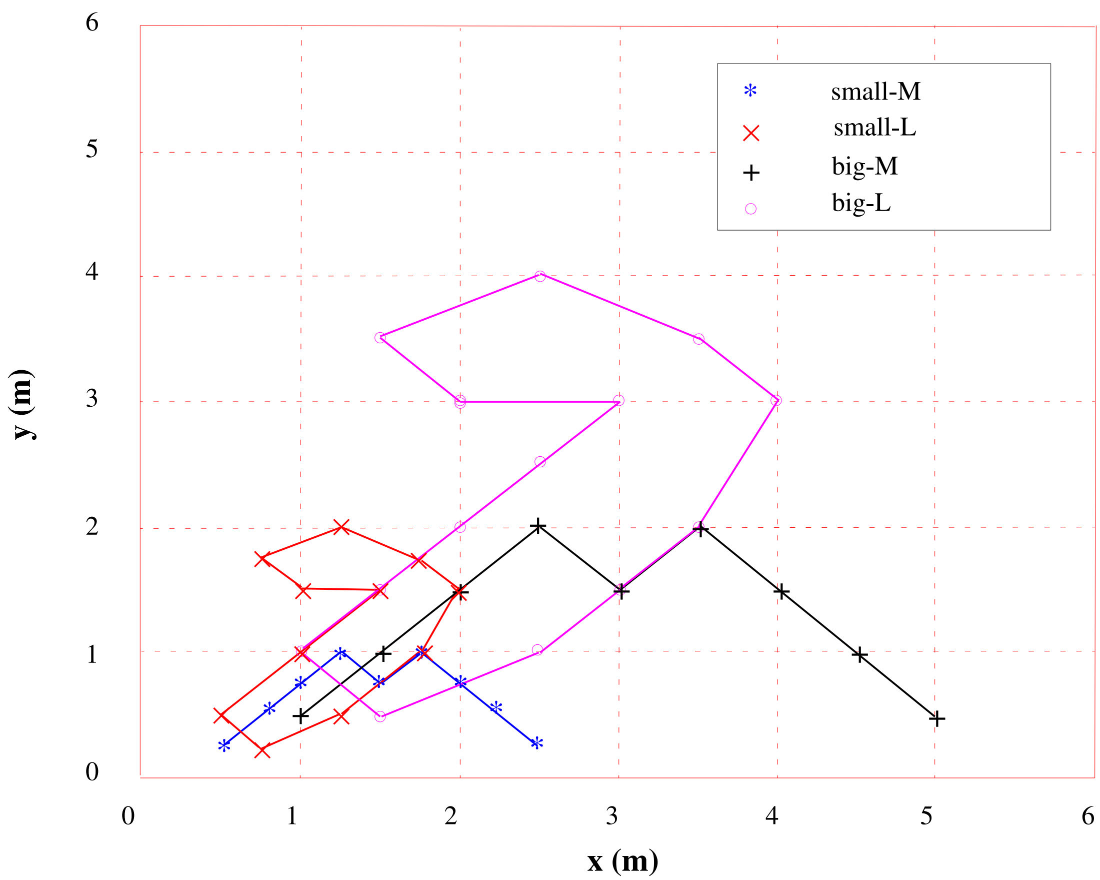


Figure 9-11. Different Workspace Used as Requirements.

The lower and the upper limits of the values selected for the workload, settling time, and overshoot are based on the feasibility of the controller's gains as well as the deflection of the tip. The variation of the values in each set is based on the sensitivity of the controller's gains and the deflection to each requirement. We were limited in the number of variations to keep the size of the design problems that need to be solved within a manageable range (e.g., 1000 problems). That is because the number of the problems is a function of the number of the variations of the requirements and constraints.

9.1.3 Finding Critical Constraints by Sensitivity Analysis

The result of an extensive sensitivity analysis on design parameters shows that the set of design parameters can be divided into three subsets:

1. The first subset includes the deflection of the arm and the control gains (especially K_{p1}). These are the design parameters that are effected by the design requirements the most and as a result are the most likely to violate the corresponding constraints.

2. The second subset of design parameters are less critical in violating their corresponding constraints. This subset includes accessible region area, and link lengths.

3. The third subset of design parameters that includes the rest of design parameters are the most likely to remain within their constraint ranges.

The sensitivity analyses also revealed that the constraints on the proportional gain of link 1 and the deflection of the tip are the most sensitive ones. Therefore, they are the two constraints that we chose to change for different projects. Changing a constraint here means to change the set of acceptable values by that constraint. That is, we would like to see how making a constraint tighter or looser affects the system in taking different paths (hence, generating different traces).

In “Constraints” on page 139 we described how the set of acceptable values for a constraint might be a function of design parameters that are changing during the design process, therefore, for such constraints the set of acceptable values change dynamically during the process. In these situations we describe the set of acceptable values by defining the values for the variables that are used to make up the acceptable set of values. For instance, the maximum acceptable ratio of the diameter to the link length is such a variable for the constraint on the maximum size of the dimension of the cross section of the link (see the example in page 141). Table 9-3 shows these variables and the set of values for them for constraints that were used in **RD**. As it can be seen only the constraints on the deflection of the tip and one of the gains is varied.

Table 9-3. Values for ‘Variables’ of the Constraints.

The Variable Name	The Set of Values
`minLink1Length`	{0.0}
`maxLink1LengthToWorkspaceLengthRatio`	{1.0}
`minLink2Length`	{0.0}
`maxLink2LengthToLink1LengthRatio`	{1.0}
`minTheta1Min`	{-3.141592653589793}
`maxTheta1Max`	{3.141592653589793}
`minTheta2Min`	{-3.141592653589793}
`maxTheta2Max`	{3.141592653589793}
`minLink1Dimension`	{0.0}
`maxLink1DimensionToLink1LengthRatio`	{0.1}
`minLink2Dimension`	{0.0}

Table 9-3. Values for ‘Variables’ of the Constraints.

The Variable Name	The Set of Values
`maxLink2DimensionToLink2LengthRatio`	{0.1}
`minLink1ThicknessToLink1DimensionRatio`	{0.05}
`maxLink1ThicknessToLink1DimensionRatio`	{0.25}
`minLink2ThicknessToLink2DimensionRatio`	{0.05}
`maxLink2ThicknessToLink2DimensionRatio`	{0.25}
`minAccessibleRegionArea`	{0.0}
`maxAccessibleRegionAreaToWork-spaceAreaRatio`	{1.0}
`minTipDeflection`	{0.0}
`maxTipDeflectionToLinkLengthsSumRatio`	{0.01, 0.001}
`minProportionalGain1`	{0.0}
`maxProportionalGain1`	{1000, 100}
`minDerivativeGain1`	{0.0}
`maxDerivativeGain1`	{200}
`minProportionalGain2`	{0.0}
`maxProportionalGain2`	{200}
`minDerivativeGain2`	{0.0}
`maxDerivativeGain2`	{200}

The combination of all the variations of design requirements and design constraints as shown in Table 9-1 and Table 9-3 generates 960 different projects:

$$2(\text{deflection}) \times 2(\text{gain}) \times 4(\text{workspace}) \times 5(\text{workload}) \times 3(\text{settling}-\text{time}) \times 4(\text{max}-\text{overshoot}) = 960$$

9.1.4 Categorizing Projects

The range of requirements or constraints can be partitioned into different subsets and assigned qualitative names. Figure 9-12 shows an example of such categorization. The reason we divided the requirements into five categories and the constraints into two categories is to cover all the values in the set of requirements and constraints shown in Table 9-1 and 9-3.

Combination of different situations with respect to the requirements and the constraints produces 10 different situations for the design projects only with respect to one requirement and one constraint. For instance, in 960 projects that were solved during the experiments 92% had at least one very tough requirement or one tight constraint (situations 5 to 10 in Figure 9-12).

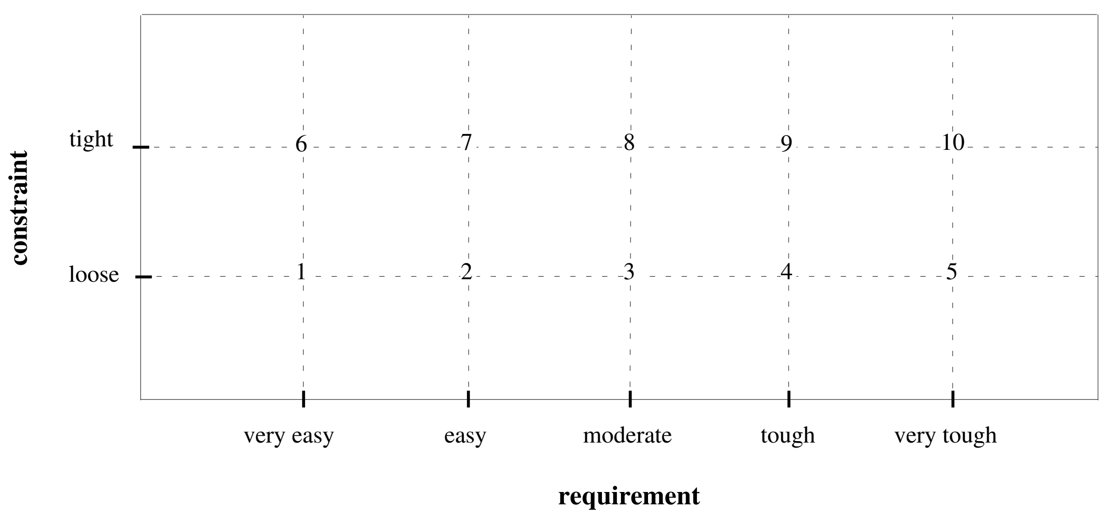


Figure 9-12. Categorizing Projects Based on Requirements and Constraints.

9.1.5 Effect of Design Approaches on Constraints

In this section we describe how the combination of some of the design approaches may affect design parameters leading to constraint violation or satisfaction. We give an example of a project that was not successful, i.e., none of the combinations of the design approaches could find a design that would satisfy all the constraints. In this example, the constraint that could not be satisfied was the constraint on the proportional gain of the controller.

In a run with the following requirements, we observed that the gain of controller (`proportional_gain_1`) would change between 105 to 16762. For this problem a solution was never found that would satisfy the constraint on the gain (the maximum allowable constraint on the gain was 100). This wide range of change was due to adopting different design approaches by designers. The results of a primary investigation showed that when `Designer_K_1` chooses to minimize the summation of link lengths and `Designer_S_2` chooses aluminum for the material (`aluminum_alloy_5456_H116`) the minimum value for the gain is found.

We discovered that a low control gain can be achieved by minimizing the link lengths and selecting a material that has a very high elasticity modulus to density ratio. This relationship can be derived from first principles. A short and light weight link has a low moment of inertia that needs less control effort hence lower control gains. Table 9-4 shows what design approaches led to the lowest control gains for a specific project.

Table 9-4. Design Approaches of the Lowest Control Gain

Proportional Gain 1	105.187	105.367	105.445	105.570	105.648	105.701	105.783	105.875	105.971
`Designer_K_1`	**minimize** the summation of link lengths								
`Designer_K_2`	choose the second link to be **0.5** of the first link								**0.75**
`Designer_K_3`	choose a **left-hand** configuration for the robot								
`Designer_K_4`	use the default approach for calculating accessible region area								

Table 9-4. Design Approaches of the Lowest Control Gain

<table>
<tr><th>Proportional Gain 1</th><th>105.187</th><th>105.367</th><th>105.445</th><th>105.570</th><th>105.648</th><th>105.701</th><th>105.783</th><th>105.875</th><th>105.971</th></tr>
<tr><td>`Designer_S_1`</td><td>**4**</td><td>**3**</td><td colspan="2">cross section = **4** times of minimum allowable</td><td>**3**</td><td>**2**</td><td>**3**</td><td>**4**</td><td>**4**</td></tr>
<tr><td>`Designer_S_2`</td><td colspan="9">use **aluminum** for the materials of the links</td></tr>
<tr><td>`Designer_S_3`</td><td colspan="3">safety factor = **1.1**</td><td>**1.4**</td><td colspan="2">**1.1**</td><td colspan="2">**1.4**</td><td>**1.1**</td></tr>
<tr><td>`Designer_S_4`</td><td colspan="2">**square** cross section</td><td>**circular**</td><td>**square**</td><td>**circular**</td><td colspan="2">**square**</td><td>**circular**</td><td>**square**</td></tr>
<tr><td>`Designer_S_5`</td><td colspan="9">use the default approach for calculating the deflection of the tip</td></tr>
<tr><td>`Designer_C_1`</td><td colspan="9">use the default approach for calculating the gains of the controller</td></tr>
</table>

Table 9-5 on the other hand shows what design approaches should be avoided when a low gain is required. `Designer_1_1` has used a design approach that puts the base of the robot to the left or below the width of the workspace rectangle. While this makes the sweep angles small, it generates long link lengths. `Designer_2_2` selects steel for the material (as opposed to aluminum) that makes the link heavy, and `Designer_2_3` selects a high safety factor that increases the dimensions of the cross section of the link. All of these approaches help to increase the moment of inertia of the link that on the other hand increases the gain.

One last observation based on the results of Table 9-5 is the contribution of `Designer_1_2` to the high gain. `Designer_1_2` is the designer that calculates the length of the links of the robot. It has three approaches that are ordered based on the ratio of the second link's length to the first link's length. The first approach sets the length of the second link as half of the first link, the second approach 0.75 and the last approach sets the link lengths equal to each other. As a result, to cover the same workspace the first and second approaches find a longer length for link 1 that at first does not seem to be in the favor of a low gain. But the fact is that the effect of the moment of inertia of the second link on the first link is larger than the effect of the first link's length. As a result, for a lower gain (in fact, for a better distribution of the values of the gains between link 1 and 2) it's better to make the second link shorter.

Table 9-5. Design Approaches for Highest Control Gain

Proportional Gain 1	9171	9618	9517	12541	12694	16762
Designer_K_1	put the base of the robot to the left or below midway width of the workspace rectangle					
Designer_K_2	link 2 **equal** to link 1		link 2 / link 1 = **0.75**	choose the second link to be **equal** to the first link		
Designer_K_3	choose a **left-hand** configuration for the robot					
Designer_K_4	use the default approach for calculating accessible region area					
Designer_S_1	**2**	**3**	cross section = **4** times of minimum allowable		**3**	**4**
Designer_S_2	use **steel** for the materials of the links					
Designer_S_3	safety factor **3**					
Designer_S_4	**circular** section	**square** section	**circular** section	**square** section	**circular** section	
Designer_S_5	use the default approach for calculating the deflection of the tip					
Designer_C_1	use the default approach for calculating the gains of the controller					

The requirements for the project that followed the approaches of Table 9-5 are shown in Figure 9-13:

```
workspace: x = [1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0]
           y = [0.5 1.0 1.5 2.0 1.5 2.0 1.5 1.0 0.5]
workload = 1.0 kg
settling_time = 1.0 sec
maximum_overshoot = 10.0%
```

Figure 9-13. Frequency of Successful Traces.

The above observations show two things:

1. Study of the behavior of the system in microscopic level (i.e., the steps that it takes to resolve a constraint violation) is rational and based on the first principles of physics.

2. At a macroscopic level the behavior of the system has some patterns that have emerged from its activities in the microscopic level. This is a promising sign of being able to discover design methodologies from the macroscopic behavior of the system.

9.2 Traces Produced by RD

We can define many different types of traces for the system, such as the trace of message handling, the trace that results in building the dependency graph, the trace of the design states, the trace of the system during the recovery from constraint violation, etc. The trace that we are interested in is the one that leads us to design methodology. Depending on how rich the design methodology is going to be (in terms of including more aspects or more details), we may have to look at various types of trace that the system produces. The most basic design methodology can be extracted from the trace of the design states.

The traces of **RD** are stored in four different files:

1. A file that stores the set of design requirements for each design project (each design project would be one example in the training set),

2. A file that stores the set of design constraints for each design project. Note that even if we keep the set of design constraints constant for different projects, the upper and lower boundaries of some of the constraints may change because they depend on the current value of some design parameters. The reason is that for some of the constraints we do not define the upper and lower boundaries explicitly, but in terms of the current value of a design parameter. For instance, we define the maximum acceptable value for the second link of the robot to be equal to the current value for the first link's length—that is, the second link is not allowed to be longer than the first link. Because the first link's

length changes during different cycles of design, the upper boundary for the constraint on the second link's length is dynamic. That should be considered when comparing different projects with similar traces for the requirements and design approaches.

3. A file that stores the sequence of design approach indices for designer agents. The order in which we store the design approaches in a sequence is based on the order that designers have in the dependency graph.

4. A file that contains the values of all design parameters for the project. If the design is successful, this is the design product description otherwise it is just the last set of values for the parameters as found by **RD**.

Each project will have a unique identification number that will be stored in each of the above trace files. Therefore we can relate traces of different type (requirements, constraints, approaches, design parameters) to each other via this identification number.

The following is summary of the information that different trace files can contain.

- the serial number of the trace,
- the design requirements and constraints for this trace,
- whether the design was successful or not,
- the values of design parameters for a successful design (i.e., description of the design product),
- the design dependency graph for designer agents,
- the sequence of design approaches in each designer agent that led to a successful design,

- the number of design cycles before reaching a successful design or the number of design cycles after exhausting all possible solutions for failure situations,
- the time and memory that were spent during this design project,
- the number of messages that were exchanged between agents,
- the number of sent, received, processed, and ignored messages for each agent (to detect bottlenecks in information exchange, overloaded agents, etc.)
- the time spent by each agent in processing its tasks especially the total time taken by each designer agent to design,
- the number of design cases (receiving inputs, generating outputs) done by each agent,
- the number of backtracking sessions that were created, their corresponding violated constraints, and the number of agendas in each backtracking sessions,
- if the solution was found while in a backtracking session, the session number, the agenda number, and the violated constraints of that session.

The above information could be used to analyze various aspects of the design process as well as the design product. For instance, we could draw the graph of a design parameter's values versus design cycles for a design project to show how they have changed over the period of the design process with respect to different design approaches. The boundaries of the design constraint on a specific design parameter value can be drawn too to show the best and worst cases in terms of satisfying the constraint.

We will prune the traces for the purpose of generating methodologies, so that they only contain the key information about the design process. The least amount of information that should be included in the traces are:

1. the serial number of the trace,
2. the sequence of design approaches in each designer agent that led to a successful design.

9.3 Distribution of Traces

The factors that are effective in distribution of traces with respect to problems are the selected range for requirements and the upper and lower limits of the constraints. The other factors are the design approaches and domain dependencies between design parameters.

Requirements influence the distribution of the traces in two different ways: First, the set of design parameters that are selected as requirements, and second, their values. Due to the domain dependencies, design requirements effect the rest of design parameters with varying strength. We have assumed that design requirements do not change during the design process. Therefore, the traces that are generated will be very dependent on which set of parameters have been selected as requirements. As a result, the generated traces will be concentrated around traces that are influenced by the set of requirements. Selecting a different set of design parameters will produce a set of traces with concentration in different parts of the trace space.

The upper and lower limits of the constraints directly affect the distribution of traces. This is because the violation of constraints cause different combinations of design approaches to be tried. Selection of the limits of most of the constraints is a domain dependent task and therefore seems to be ineffective in changing the distribution of the traces. However, the fact is that the limits of the constraints can dynamically change as a function of design parameter values. Therefore, they may become an important factor in the distribution of traces. In addition, some of the design constraint's limits are chosen based on the engineering judgement and the specific application in mind.

Selection of the actual design approaches directly affects the distribution of traces of approaches. If different design approaches generate design parameters' values that are considerably apart from each other, there is more chance that the corresponding violated constraints will be satisfied somewhere later in the process.

Please note that the 'design approaches' factor is not independent of 'requirements' and 'constraints' factors. Changing the set of design requirements makes a different set of parameters and their corresponding constraints critical. This might make the less effective approaches more effective with regard to those parameters and constraints.

Dependencies between design parameters determine whether choosing a different design approach can resolve the violated constraints. There are two difficulties, however, in using domain dependencies for directing the selection of one design approach over another: First, some of the dependencies are not monotonic. Second, the violated constraints often require contradictory changes in design approaches. In the following we will see that reduction of the deflection of the robot's arm is usually in contradiction with reducing control gains. The constraints on these two parameters are of critical type, meaning that they are effected by the requirements and are violated more often than the others.

9.4 Generating Traces

We used **RD** to solve a set of 960 design projects on the following machines:

1. SUN Ultra 5 Workstation, Running OS Solaris 2.5.1, CPU UltraSparc, 4 GB HDD, 64 MB RAM.
2. Digital 433au Workstation, Running Digital Unix 4.0D, Alphachip 433 MHZ CPU, 4 GB system Disk, 576 MB of RAM.

The run time varies from project to project from a few seconds to a couple of hours. The projects with easy requirements and loose constraints are solved very fast. This is, because there is a high chance that the solutions that are found early in the process satisfy all the constraints. For hard problems (i.e., tough requirements and tight constraints), however, the number of satisfying solutions are not many or do not exist at all. As a result, the system needs more time to search through all possible solutions. For the set of 960 projects that **RD** solved, the two machines were running for a period of around three weeks.

Figure 9-14 shows the frequency of successful projects compared to unsuccessful traces for the whole set of projects. The large number of unsuccessful traces relative to the total number of projects (38%) is due to the coarse grid that we have chosen on the requirement space. The number of projects with tough requirements and tight constraints is quite large compared to total number of projects. In fact 92% of the projects have at least one requirement that can be considered tough or one constraint that can be considered tight.

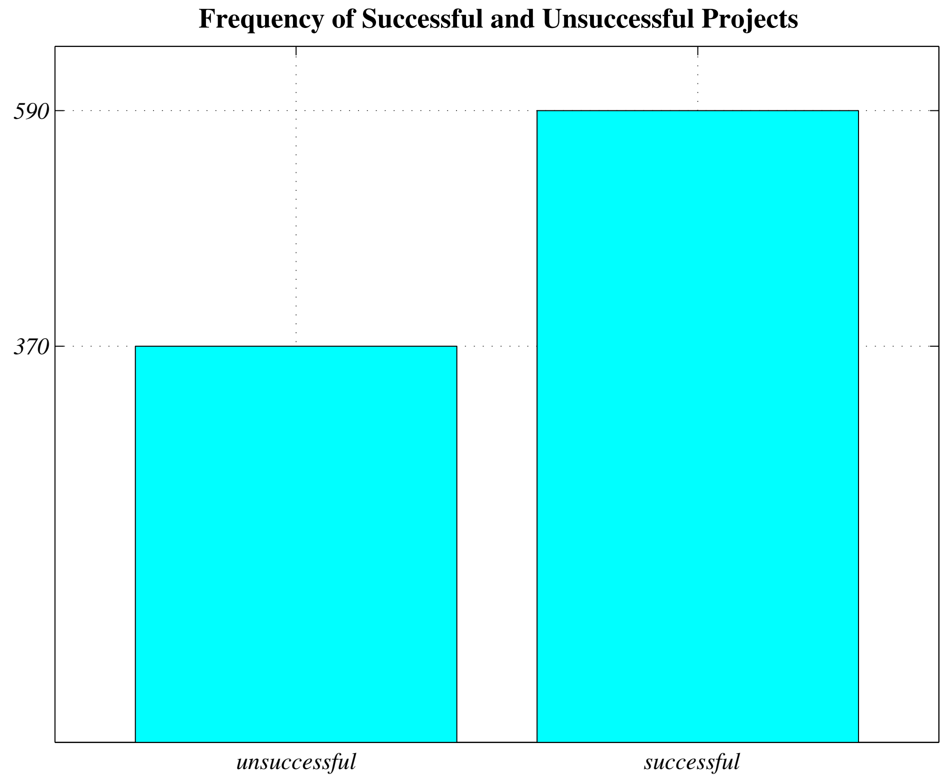


Figure 9-14. Frequency of Successful and Unsuccessful Projects.

Figure 9-15 shows how the successful and unsuccessful projects are distributed. The figure shows that a pattern is repeated that divides the projects into two parts at 480. These two parts correspond to the point where the constraint on deflection changes, that is for the first half of the projects the constraint on deflection is loose and for the second half it is tight. Toward the end of each half that the tough requirements combine with the tight constraint on the gain of the controller, the number of failed projects increases. The number of unsuccessful projects is clearly more in the second half because of the effect of the tight constraint of deflection.

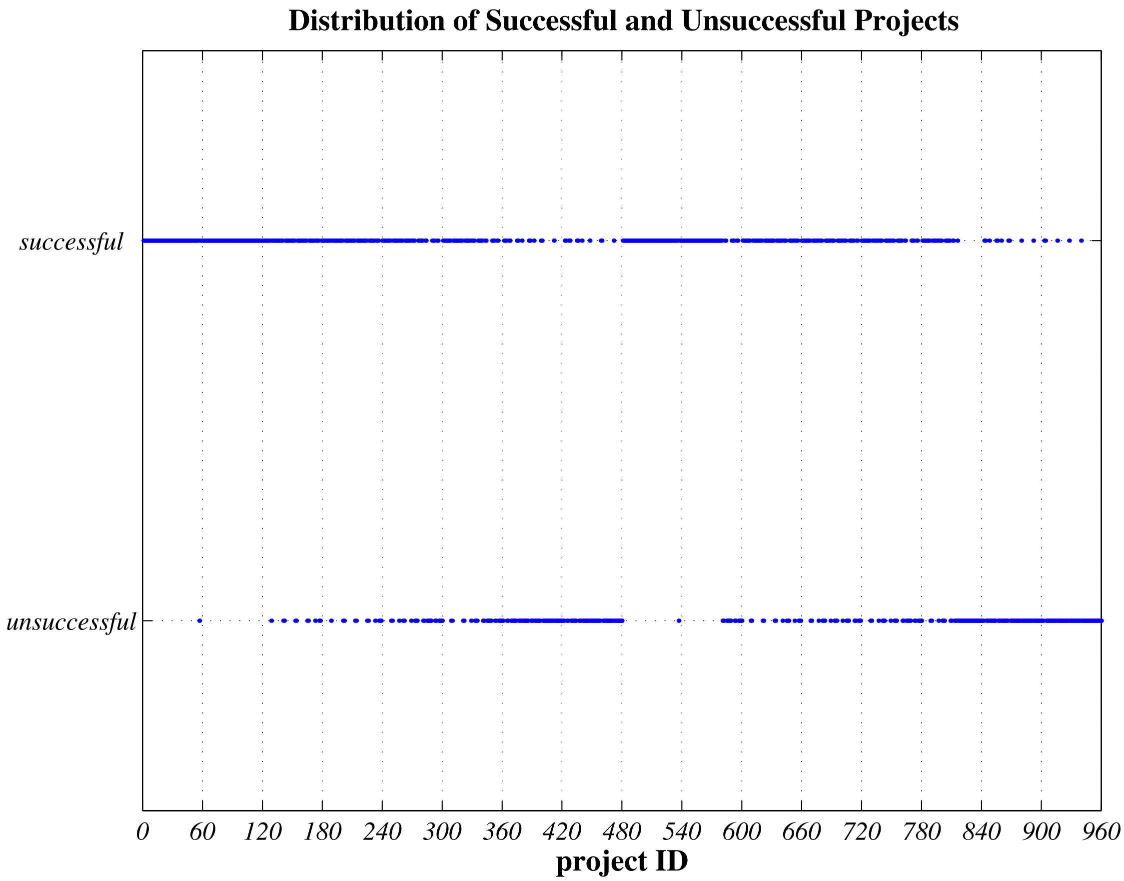


Figure 9-15. Distribution of Successful and Unsuccessful Projects.

Figure 9-16 shows how many projects followed a specific trace. The promising results is that the distribution of the traces is quite scattered—that is, many projects followed similar traces. Remember that the a specific trace index shows a unique combination of design approaches (see "Design Path" on page 105). Therefore, the total number of possible traces is the product of the number of design approaches of all the designer agents. For the experiments shown in this dissertation the total number of possible traces is 2304. Among all 2304 possible traces only 84 were followed to generate successful designs, i.e., less than 4%. Throughout this and the following chapters we refer to the set of traces that produced successful designs as successful traces. The low percentage of successful traces indicates that for each group of projects that followed a particular trace there is a unique combination of approaches leading to successful designs, hence there is a high chance that if similar projects follow the same trace they will succeed in generating a successful designs. As a result, the path followed by those projects can lead us to formulating a design methodology for the projects that followed that trace as well as projects that are similar to those projects.

We may even find traces in the set of successful traces that are close enough so that they can be clustered together form a generalized trace. A generalized trace covers all the projects that followed each of the traces incorporated in the generalized trace. In the next chapter we will extensively discuss clustering of traces into generalized traces.

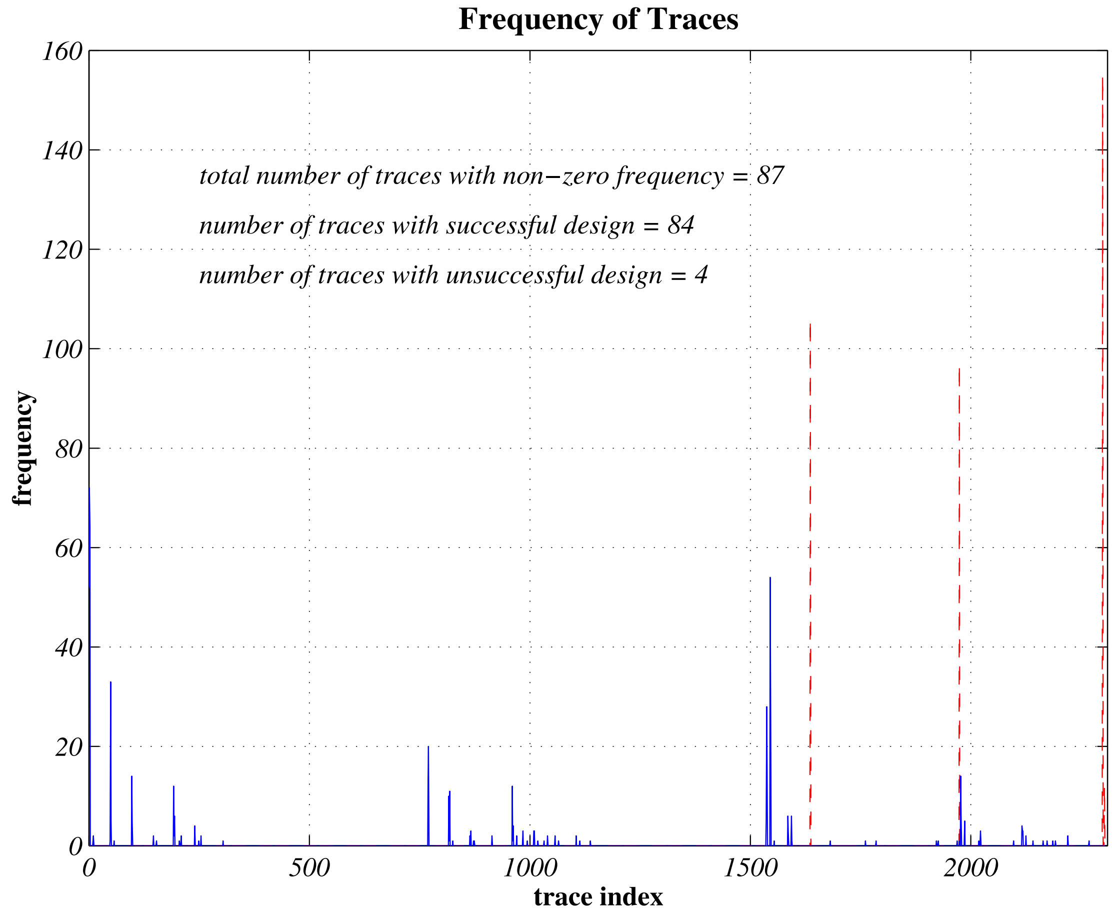


Figure 9-16. Frequency of Traces.

Figure 9-17 shows how the traces are distributed relative to the projects. It clearly shows that there are some patterns in projects taking one particular trace. We intentionally have set the grid in Figure 9-17 to match one of the dominate patterns that is happening every 60 projects. This pattern matches the change of workspace requirements: the first 60 projects have a ‘small-M’ type of requirement for the workspace. The next three subsets of projects in groups of 60 projects have a ‘small-L’, ‘big-M’, and ‘big-L’ type of workspace respectively (see Figure 9-11 on page 189).

Observing these patterns is another promising sign of being able to formulate design methodologies based on the traces of the system. These patterns show that the projects that followed the same traces have common features, e.g., they have the same requirements on workspace. As a results, we might be able to generalize the set of projects that followed the same trace based on their common features. Generalizing projects is another step toward formulating methodologies.

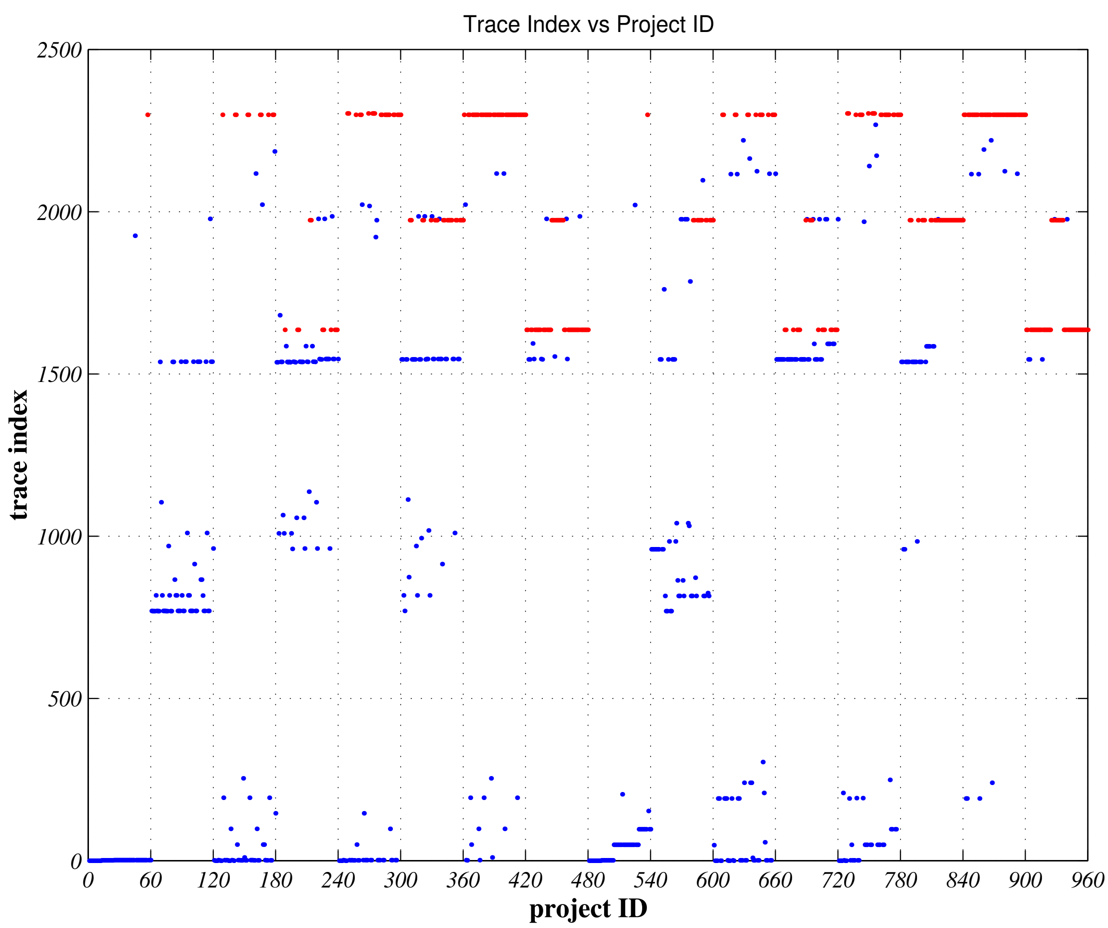


Figure 9-17. Traces versus Projects.

Figure 9-18 shows the frequency of all 84 successful traces. It is evident from this figure that a small number of traces have very high frequencies (say higher than 20). This is even more good news for being able to find design methodologies. A small number of traces with relatively high frequency shows that even without clustering traces together, we might be able to find methodologies that are based on those traces and still cover a large number of different situations (e.g., 70 different projects).

Moreover, existence of a small number of traces with high frequencies helps in clustering traces together. The high frequency traces can act like seeds for clustering—that is to absorb the traces with lower frequencies and form generalized traces with even more frequencies.

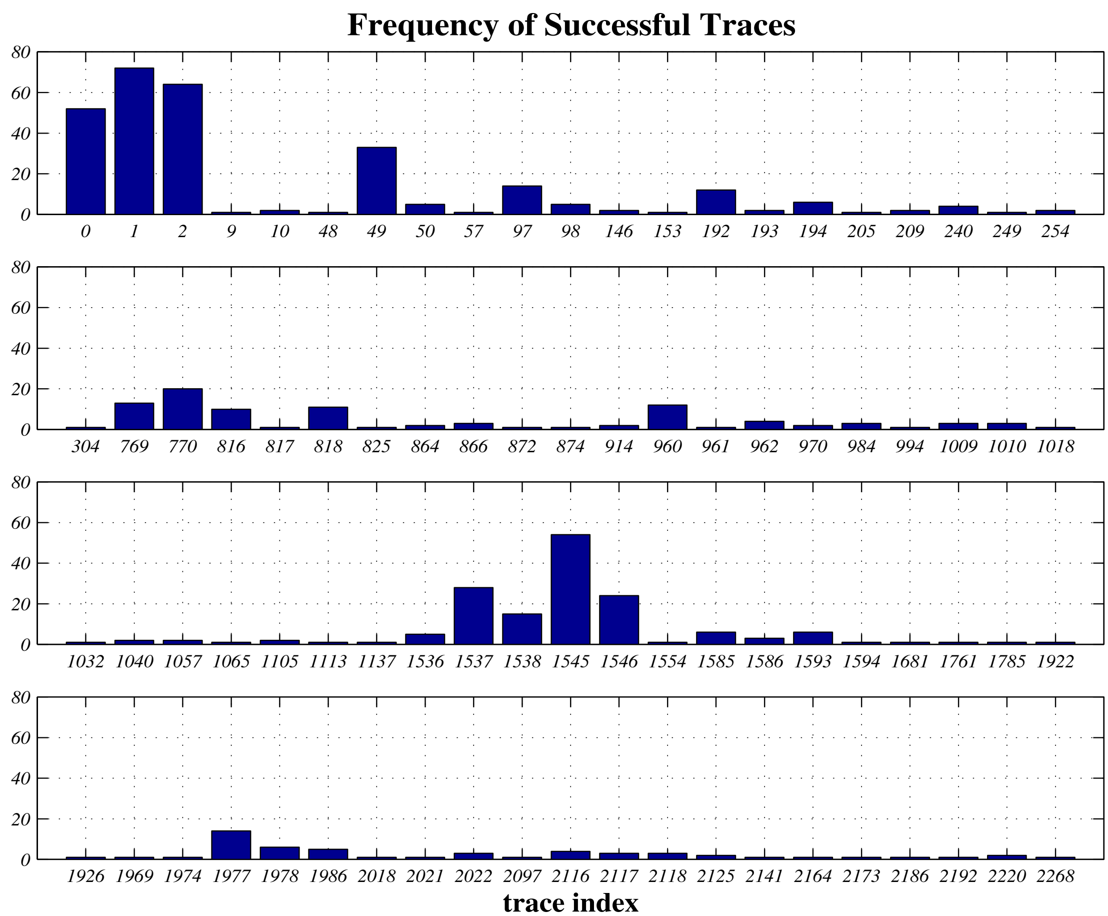


Figure 9-18. Frequency of Successful Traces.

Figure 9-19 proves that the traces that were followed by projects with common features have common features too. That is, there is a correlation between clusters of projects and clusters of traces. For instance, all the projects that had required 'small-M' for workspace have followed traces that either put the base of the robot below or to the left of the workspace rectangle or put the base at a location that minimizes the sum of link lengths. As we mentioned before, the existence of a correlation between clusters of projects and traces is a promising sign of being able to formulate methodologies.

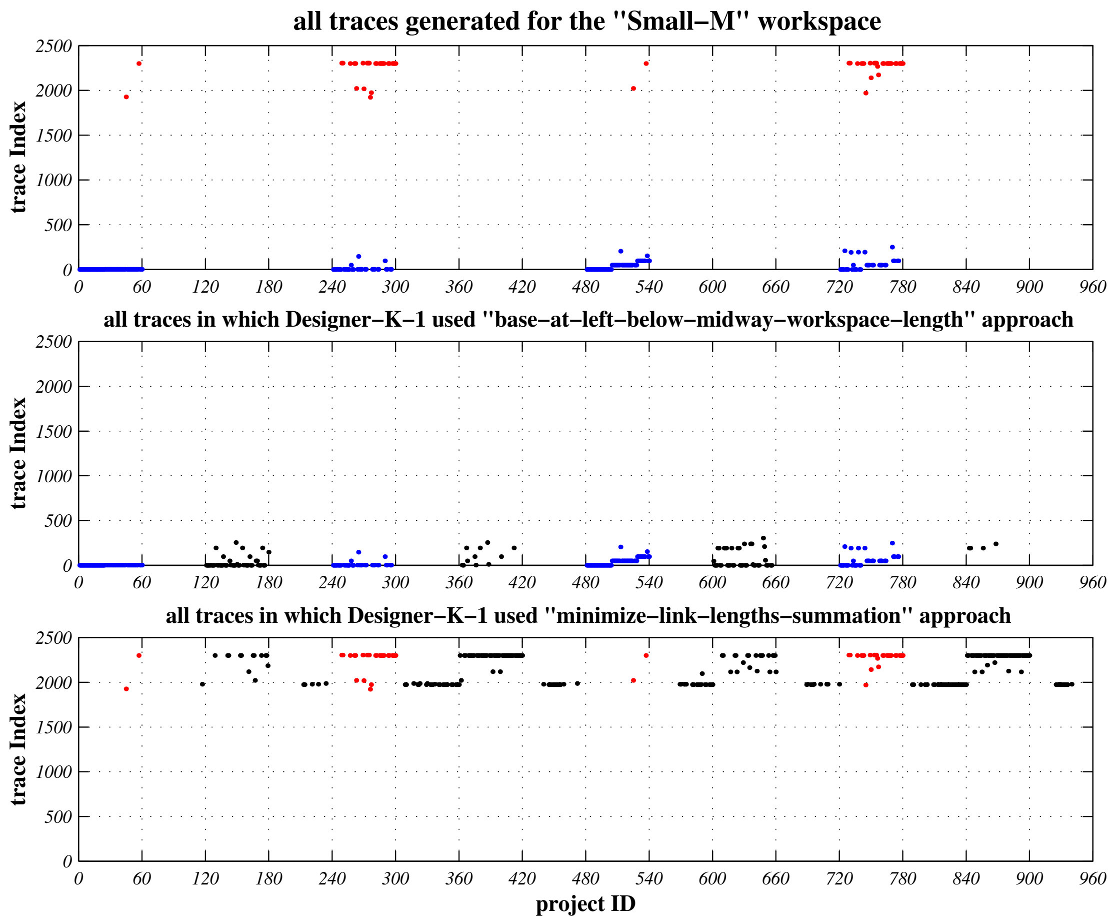


Figure 9-19. Correlation between Requirement Space and Trace Space.

In this chapter we described how **RD** was used to experiment with the design process. We presented the results of the experiments and showed that they are promising in being able to discover the design methodologies. In the next chapter we analyze these results in order to extract design methodologies.

10 Results

In this section we analyze the traces produced by the system in order to extract useful information from them. Some of this information such as dependency between the designers can be easily gathered from the trace of the system. More analysis, however, is needed for extracting the patterns that lead to the formation of methodologies. Also, traces need to be clustered based on their similarities in order to generalize the patterns. Generalized patterns are a way to produce methodologies that are applicable to a wide range of problems.

10.1 Summary of the Observations

The following summarizes the observations that we have made based on the traces generated by the system that were represented in Chapter 9.

1. The ratio of the traces taken by the system that led to a successful design compared to all possible traces is quite small: 84/2304 = 0.0365 (3.7%). This shows that the system has been able to successfully identify the small percentage of paths (i.e., 3.7%) that among all others lead to good designs.
2. The variation of traces with respect to projects clearly shows some patterns that is a good sign for being able to cluster groups of traces together.

3. In Figure 9-19 on page 209 there is a correlation between the patterns in groups of traces and groups of problems (i.e., requirements and constraints). This is a promising sign of being able to map clusters of problems to clusters of traces—a major step toward being able to index methodologies.

4. Frequency of the traces is not distributed evenly thus, the coverage of traces are different. This might lead to a way to evaluate the quality of different traces. Similarly, methodologies that are built using different traces might have different quality (see “Goodness of a Cluster of Traces” on page 217).

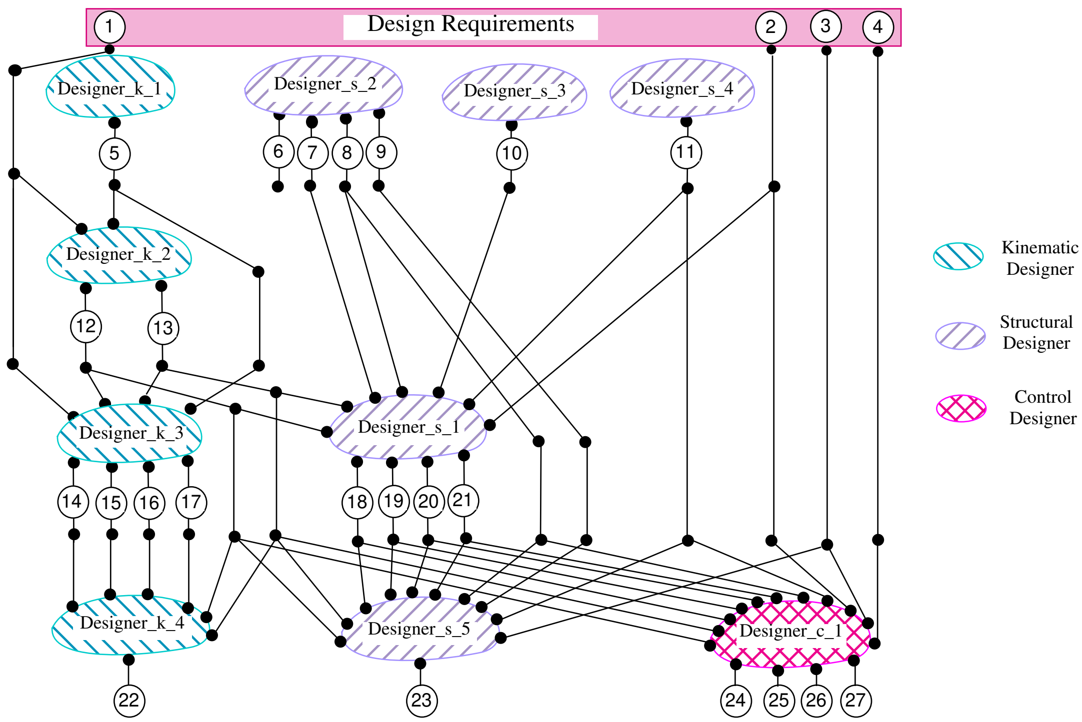
5. Due to preferential order of use of design approaches, traces are different regarding the desirability of their approaches. This might lead to another measure for the goodness of a methodology.

10.2 Dependency Graph

Discovering the dependency between designers is the first set of results that we extract from the traces of the system. Some designers need inputs that are generated by other designers, therefore, they have to wait until the designers that supply input to them finishes its job. Having small designers combined with an opportunistic strategy provides a way to discover dependencies among design parameters automatically. Additionally, the concurrency between designers in each design cycle can also be discovered by the system.

At the end of each run the `DependencyProvider` agent prints out the information about the dependency between designers. This information leads to formation of the dependency graph (see Figure 7-13 on page 136). The dependency graph resulted from the experiments is shown in Figure 10-1.

The dependency graph is formed dynamically during run time. In general the dependency graph changes its structure and its members during the course of a design process. The reason is that participation of some of the designers in the design might become unnecessary due to some decisions regarding how to conduct the design by designers in the up-stream. Similarly some new designers might be able to contribute to the design because of some early decisions. As a result, the way designers depend on each other might change, and that in turn changes the dependency graph.



Design Parameters:

(1) workspace (2) workload (3) settling_time (4) maximum_overshoot

(5) base_location

(6) material_name (7) material_mass_density (8) material_yield_stress (9) material_elasticity_modulus

(10) structural_safety_factor (11) link_cross_sectional_shape (12) link1_length (13) link2_length

(16) theta2_min (17) theta2_max (14) theta1_min (15) theta1_max

(18) link1_cross_section_dimension (19) link2_cross_section_dimension

(20) link1_cross_section_thickness (21) link2_cross_section_thickness

(22) accessible_region_area (23) tip_deflection

(24) proportional_gain1 (25) derivative_gain1 (26) proportional_gain2 (27) derivative_gain2

Figure 10-1. Dependency Graph for Design of a 2 DOF Robot

10.2.1 Discussion

The dependency graph of Figure 10-1 can be used during dependency-directed backtracking. When a constraint is violated, the design parameters and the designers affecting the constraint are identified. Based on which designers are responsible for generating those design parameters, a backtracking agenda is prepared and executed. Dependency-directed backtracking is more efficient in recovering from constraint violations.

Table 10-1 compares the two runs of the system for the same project one with dependency-directed and the other by exhaustive backtracking. It is clear that dependency-directed backtracking is a superior method in terms of spending time and resources to find a successful design.

Table 10-1. Project 161.

type of backtracking	trace index	number of cycles	time spent (hour)	memory size	number of events	number of messages
exhaustive	2118	5294	02:54:24	36921.0 K	2085879	341268
dependency	2118	2171	00:47:07	13304.0 K	867179	140069

The dependency graph of Figure 10-1 shows not only the way designers are dependent on each other, it also reveals that the following objectives for the design process have been achieved:

- **Integration.** From the mix of designers we can see that an integration of disciplines has happened.
- **Information Sharing.** The links in the dependency graph shows the information exchange between different disciplines and designers.

- **Collaboration.** Different designers from different disciplines collaborate with each other in order to push the design process ahead and generate a good design. Collaboration also happens during backtracking when based on the dependency data some of the designers may have to change their most desirable decisions in order for another designer be able to recover from a constraint violation.
- **Concurrency.** It is clear that the possible concurrency between different designers has been discovered.
- **Bottlenecks.** The dependency graph of Figure 10-1 shows that in the second row only Designer_K_2 can design. That is, while in the first row four designers and in the third row two were designing simultaneously, in the second row no concurrency can happen. As a result, the design process at this row is not as efficient as the other stages. Therefore, this is a bottleneck in the design process.

10.3 Clustering the Traces

In order to be able to extract design methodologies based on the traces generated by the system we need to cluster similar traces together. The similarity of two traces can be measured based on how their ingredient design approaches differ from each other. We will define a metric for measuring the distance between two traces in a mathematical form later in this section. The further two traces are from each other the less similar they will be.

A generalized trace will replace each cluster of similar traces. Later we will find correlation between each generalized trace and sub-sets of requirements and constraints. Suppose that the following five traces are clustered together:

Table 10-2. An Example Trace Cluster

trace index	Designer_K_1 approach	Designer_S_2 approach	Designer_S_3 approach	Designer_S_4 approach	Designer_K_2 approach	Designer_K_3 approach	Designer_S_1 approach
1926	5	0	0	0	0	1	2
1974	5	0	1	0	0	1	2
2018	5	0	2	0	0	0	2
2021	5	0	2	0	1	1	1
2022	5	0	2	0	0	1	2

A candidate generalized trace for the cluster of Table 10-2 is shown in Table 10-3:

Table 10-3. Generalized Trace for Trace Cluster of Table 10-2

generalized trace	Designer_K_1 approach	Designer_S_2 approach	Designer_S_3 approach	Designer_S_4 approach	Designer_K_2 approach	Designer_K_3 approach	Designer_S_1 approach
example	5	0	(0 | 1 | 2)	0	(0 | 1)	(0 | 1)	(1 | 2)

In some domains the collection of design approaches for each designer agent in the generalized trace (e.g., (0 | 1 | 2) for Designer_S_3) can be generalized based on the common features of those approaches. For instance, the collection of the design approaches might be generalized as catalog approaches or iterative approaches. In this dissertation we do not do such generalization.

10.3.1 Goodness of a Cluster of Traces

The goodness of a cluster of traces can be measured based on two factors:

1. The number of projects that followed the traces clustered: *coverage*

This is a measure of how many different design situations (i.e., projects) are covered by the traces clustered. We call this measure the *coverage* of the cluster. We can calculate the coverage of a cluster by adding the frequencies of the traces clustered. Methodologies that are generated from clusters with higher coverage are applicable to a wider range of design problems. As we discussed in “Better Design Methodology” on page 60 being applicable to a wider range of design problems is one of the factors that distinguishes superior design methodologies.

We calculate the coverage of a cluster by adding the coverage of traces it contains using Equation 10-1:

$$\text{coverage} = \left(\sum_{i=1}^{\text{traces}} p_i \right) \tag{10-1}$$

where, p_i is the number of projects covered by trace *i,* and “traces” is the number of traces in the cluster.

2. The number of variations of design approaches in the cluster: *uniformity*

Traces differ from each other because of the difference in the design approaches that designers have used. Clustering accumulates these different approaches in one place causing some loss information. That is, instead of being specific about what approach should be used for the situations covered (i.e., projects covered), the cluster suggests a collection of the possible approaches. We refer to this measure as the *uniformity* of the cluster. The most uniform cluster is the one that does not have any variation in its approaches. Two factors affect the uniformity of a cluster:

1. the number of designers that vary their approach in the cluster, and

2. the amount of variation for each designer that varies its approach.

We would like to give more credit to those clusters that have a smaller number of designers that vary their approach.

Based on the above factors we can assign a number to the uniformity of a cluster using Equation 10-2:

$$\text{uniformity} = \left(\sum_{j=1}^{\text{designers}} \frac{1}{a_j} \right) \tag{10-2}$$

where a_j is the number of approaches accumulated for designer j in the cluster, and "designers" is the number of designers in generalized trace.

The most uniform cluster is the one that does not have any variation in its approaches, i.e. all of the participant designers have used only one approach during different projects. For instance, if seven designers have participated in the design process the maximum possible uniformity factor would be 7, Equation 10-3.

$$\text{uniformity} = \left(\sum_{j=1}^{\text{designers}} \frac{1}{a_j} \right) = \sum_{j=1}^{7} \frac{1}{1} = 7 \tag{10-3}$$

Please note that Equation 10-2 incorporates both factors mentioned above. For instance, assuming that seven designers are involved a cluster with one designer varying its approach four times receives more credit than a cluster with two designers each varying their approaches two times, Equation 10-4.

$$\text{uniformity}_{one-four} = \left(\sum_{j=1}^{7} \frac{1}{a_j}\right) = \frac{1}{4} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} = 6.25$$

$$\text{uniformity}_{two-two-two} = \left(\sum_{j=1}^{7} \frac{1}{a_j}\right) = \frac{1}{2} + \frac{1}{1} + \frac{1}{1} + \frac{1}{2} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} = 6 \tag{10-4}$$

The goodness of a cluster of traces can be defined as the product of its coverage and its uniformity, Equation 10-5:

$$G_C = \left(\sum_{i=1}^{traces} p_i\right) \times \left(\sum_{j=1}^{designers} \frac{1}{a_j}\right) \tag{10-5}$$

The ideal situation is when all the projects have followed the same trace. That is, there will be one cluster containing only one trace that is applicable to all the projects. For the experiments that we have done here an ideal cluster would receive a goodness of 6720:

$$G_C = \left(\sum_{i=1}^{1} p_i\right) \times \left(\sum_{j=1}^{7} \frac{1}{a_j}\right) = 960 \times 7 = 6720 \tag{10-6}$$

However, the ideal situation is very unlikely to happen for real problems.

Table 10-4 gives two examples of the goodness of clusters:

Table 10-4. Goodness of Clusters.

Cluster	D_K_1	D_S_2	D_S_3	D_S_4	D_K_2	D_K_3	D_S_1	coverage	uniformity	G_C
cluster-1	5	0	(0 | 1 | 2)	0	(0 | 1)	(0 | 1)	(1 | 2)	50	4.83	241.5
cluster-2	5	0	(0 | 1)	0	0	1	(1 | 2)	35	6.00	210.0

Cluster-1 has received a higher score for goodness despite the fact that it is more uniform. That is, coverage has had more effect on the goodness of the cluster. Obviously, weight factors can be introduced into the equation that makes the goodness more biased toward the coverage or uniformity. In this dissertation we do not weight the coverage or the uniformity.

To be applicable to general situations the coverage and the uniformity of the cluster should be normalized. In general situations the number of designers may vary from problem to problem. Normalization is needed also when we want to compare the goodness of clusters from different sets of experiments and across different domains. The coverage can be normalized by dividing the resulting number by the total number of projects in the experiment. The uniformity can be normalized by dividing it by the maximum score for the uniformity of the cluster (equal to the number of participant designers).

Normalizing the goodness generates very small numbers that are hard to compare to each other. To increase the resolution we multiply the result by 1000. The general equation for calculating the normalized goodness of a cluster is shown in Equation 10-7:

$$G_{CN} = \frac{\left(\sum_{i=1}^{traces} p_i\right)}{T_p} \times \frac{\left(\sum_{j=1}^{designers} \frac{1}{a_j}\right)}{T_d} \times 1000 \qquad \textbf{(10-7)}$$

10.3.2 Cluster Tree

The result of clustering the traces can be represented in a tree structure [Langley 96, page 216]. At the leaves are the traces and the root node is the cluster that contains all the traces.

The nodes in the middle are the result of the clustering process with each having a parent and some children. A parent node is the cluster that contains all of its children clusters. This is because the cluster tree is constructed from the leaves towards the root (i.e., the children are clustered together to generate the parent cluster). However, naming a node as parent or child is merely based on the structure of the tree and not based on the process that builds the tree. A *row* in the cluster tree is composed of all clusters at the same depth.

The whole cluster tree for the traces generated during the experiments is included in Appendix B.

10.3.3 Naming Convention for the Clusters

The name of the clusters shows the location of the cluster in the cluster tree. The leaves of the tree are traces generated by RD and the root is a cluster that contains all traces. The name of a cluster is composed of two numbers separated by either an underscore '_' character or a dash '-'. The first number is the height of the cluster node in the cluster tree and the second number is the position of the node, from left to right, at that height.

For instance, Cluster 1-16 is the result of the first round of clustering traces (i.e., it is located one level above traces generated by RD) and is the 16th cluster in that level.

10.4 Evaluation of Clusters

In this section we evaluate the quality of the clusters generated by **RD** by looking at their coverage, uniformity, and goodness.

Figure 10-2 shows the coverage of the clusters. Five groups can be distinguished based on the range of coverage:

- Group1: with high coverage around 60%. These are clusters at a height of 16 or higher in the cluster tree.
- Group 2: with a coverage between 30% to 50%. These clusters start from clusters at height 1 and continues up to clusters at height 15 in the cluster tree.
- Group 3: with a coverage of around 10% that includes clusters from height 1 to height 15 in the cluster tree.
- Group 4: with coverage of 3 to 6% that includes clusters from height 0 up to height 15.
- Group 5: with coverage less than 3% that includes clusters at all heights except the cluster at the root of the tree. It is worthy to note that some of the clusters at high levels in the cluster tree have very small coverage. One might think of these clusters as the noise in the results and eliminate them from clustering process all together.

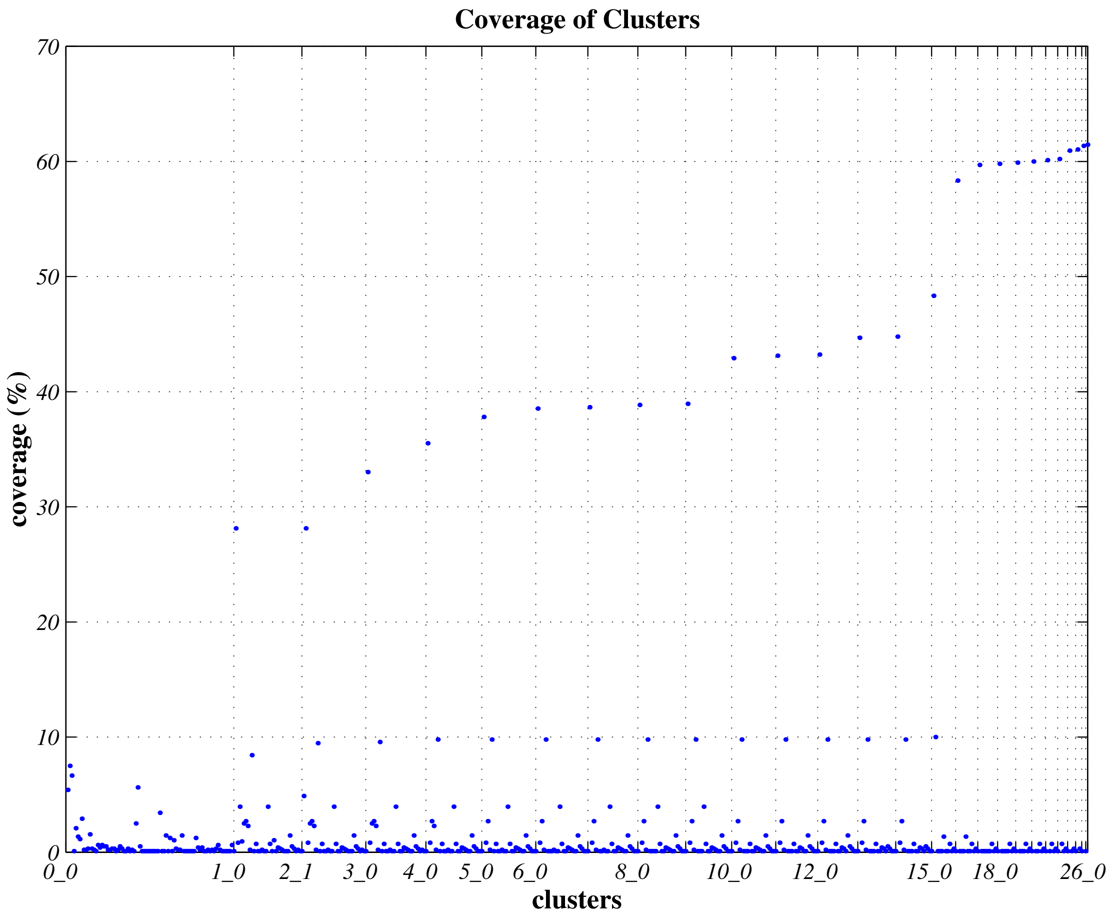


Figure 10-2. Coverage of Clusters Generated.

Figure 10-3 shows the uniformity of clusters. Similar to the coverage, we can distinguish five groups:

- Group 1: clusters with a uniformity less than 0.5 that start forming after the height of 15 in the cluster tree.
- Group 2: clusters with a uniformity around 0.6 that start from clusters with height 2 and continues up to height 15.
- Group 3: clusters with a uniformity around 0.8 that starts at height 1 and continues up to the clusters very close to the top of the cluster tree.

- Group 4: these are clusters with a uniformity around 0.9 that similar to Group 3 includes clusters from height 1 up to clusters very close to the root of the tree.
- Group 5: these are basically clusters with only one trace that have uniformity of 1.0. This group covers clusters from height zero up to one level below the root of the tree. In fact, as expected, the leaves of the tree, traces generated by **RD**, all have uniformity of 1.0.

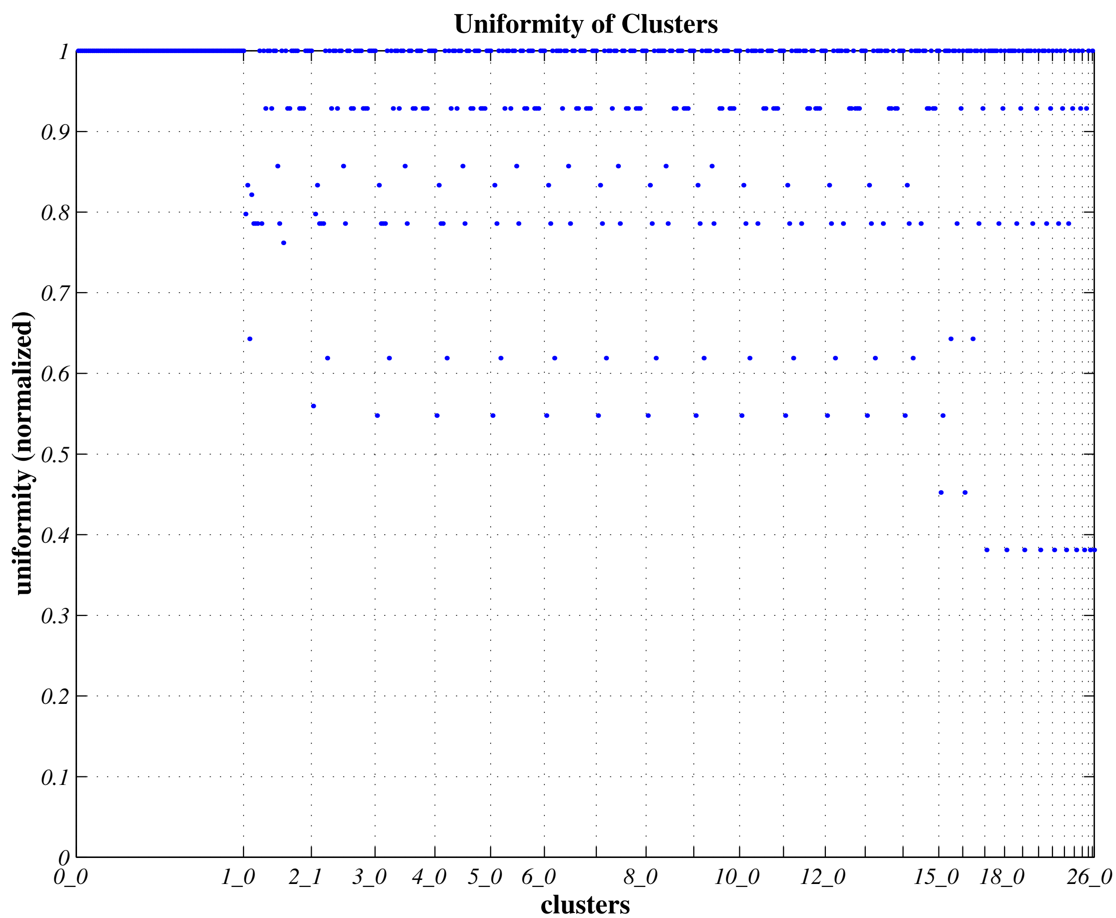


Figure 10-3. Uniformity of Clusters Generated.

Figure 10-4 shows the goodness of the clusters. From this figure it is clear that in each row of the cluster tree there is a cluster whose goodness is considerably higher than the others. These are clusters with a goodness higher than 180 in Figure 10-4. For the group of clusters up to height 15 there is one cluster with a goodness around 60 and another one with a goodness between 10 and 40. The last group of clusters have very small goodness and are distributed throughout the cluster tree.

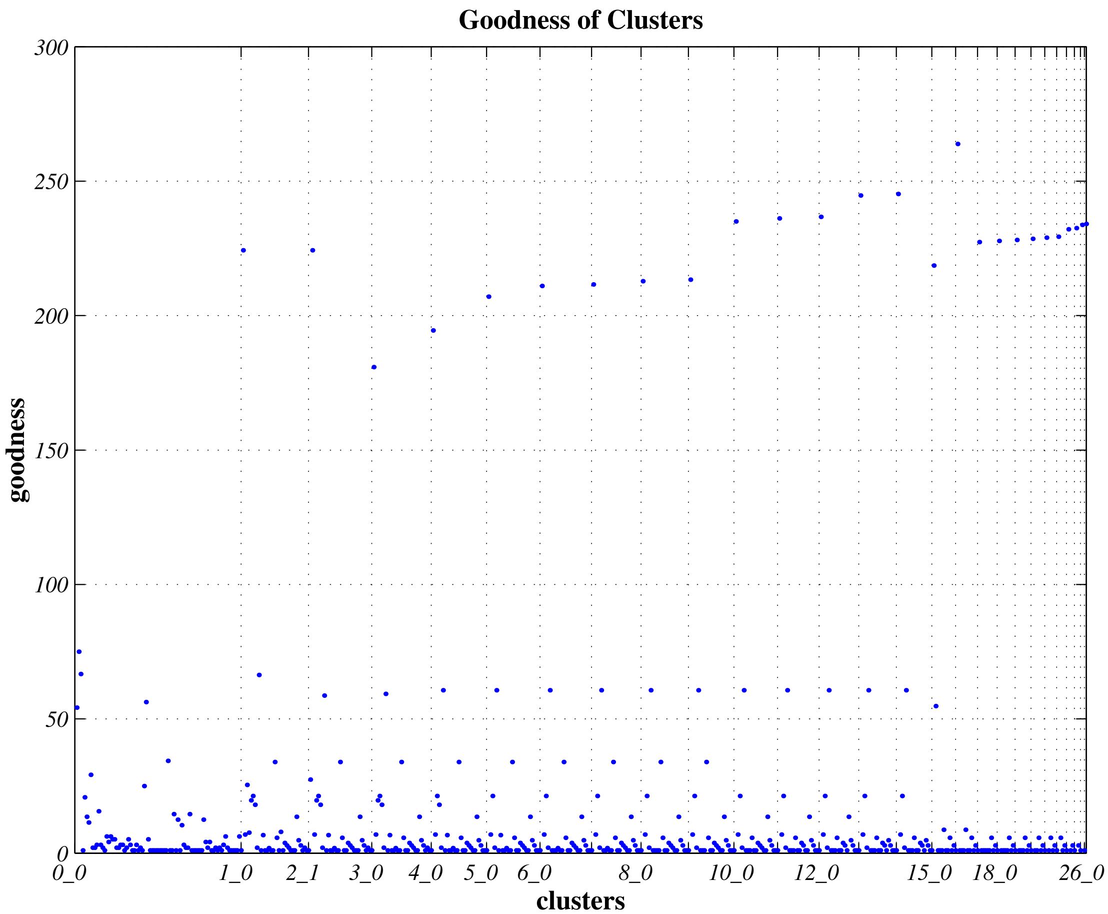


Figure 10-4. Goodness of Clusters Generated.

Table 10-5 shows the clusters with the highest goodness in ascending order of goodness:

Table 10-5. Clusters with highest goodness.

Cluster	Coverage	Uniformity	Goodness
3-0	33%	0.5476	180.8284
4-0	35%	0.5476	194.5188
5-0	37%	0.5476	207.0685
6-0	38%	0.5476	211.0615
7-0	38%	0.5476	211.6319
8-0	38%	0.5476	212.7728
9-0	38%	0.5476	213.3433
15-0	48%	0.4524	218.6508
1-0	28%	0.7976	224.3304
2-1	28%	0.7976	224.3304
17-0	59%	0.3810	227.3810
18-0	59%	0.3810	227.7778
19-0	59%	0.3810	228.1746
20-0	60%	0.3810	228.5714
21-0	60%	0.3810	228.9683
22-0	60%	0.3810	229.3651
23-0	60%	0.3810	232.1429
24-0	61%	0.3810	232.5397
25-0	61%	0.3810	233.7302
26-0	61%	0.3810	234.1270
10-0	42%	0.5476	235.0198
11-0	43%	0.5476	236.1607
12-0	43%	0.5476	236.7312
13-0	44%	0.5476	244.7173
14-0	44%	0.5476	245.2877
16-0	58%	0.4524	263.8889

Figure 10-5 enlarges the portion of the goodness graph that contains the clusters with the highest goodness. The cluster with the highest score is Cluster 16-0 that has a goodness of 264. This peak happens after a jump in the convergence of clusters from height 14 to height 15.

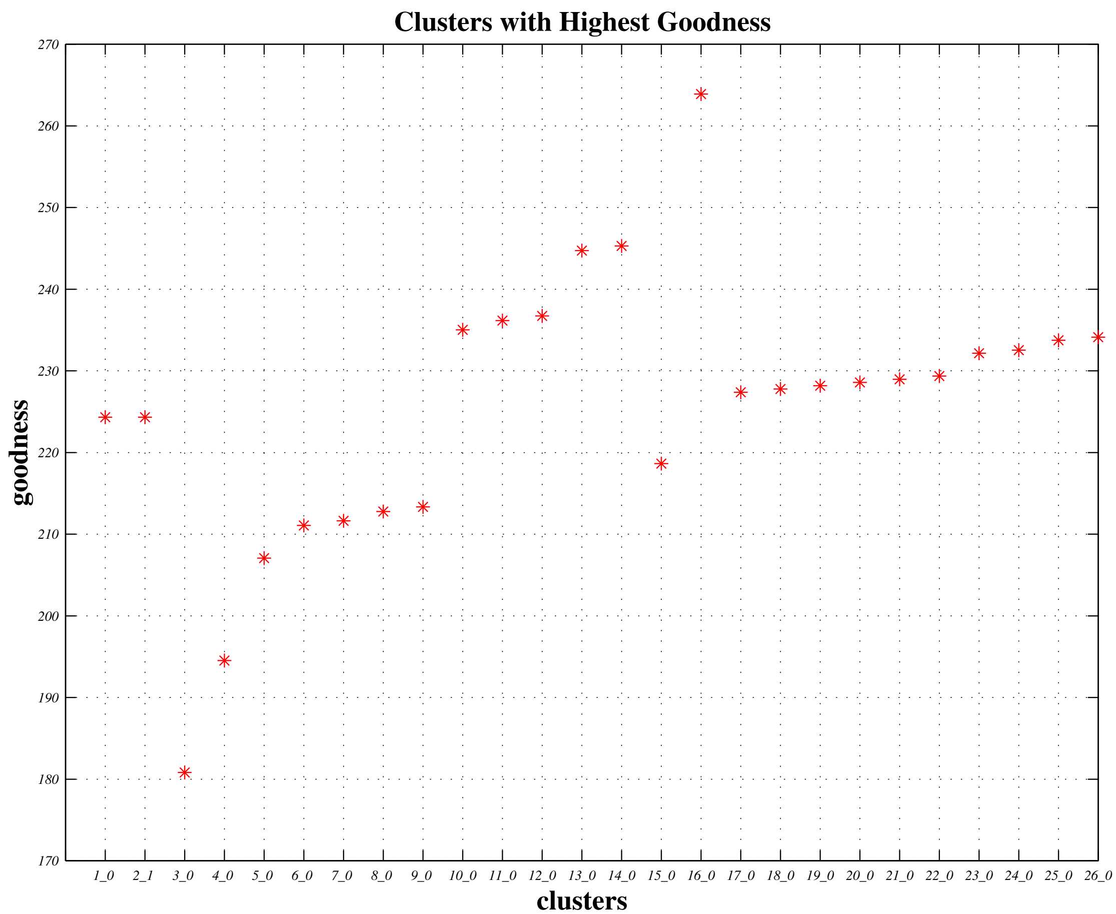


Figure 10-5. Clusters with Highest Goodness Measure.

10.5 Formulating Methodologies

To be able to formulate methodologies we need to find correlation between subsets of projects and clusters of traces. A correlation exist between a subset of projects and a specific trace or cluster of similar traces, if the projects have followed that trace or cluster of traces.

Selection of the final methodologies from the set of candidate methodologies is a trade-off process. In one end you have methodologies that are very detailed but are not applicable to many projects. On the other end there are methodologies that are more general (i.e., do not say as much about the details) but are applicable to a large number of projects.

The generality of the methodologies generated based on clusters of traces increases as the height of the cluster in the cluster tree increases (i.e., as additional similar clusters are combined into new clusters). On the other hand, as more clusters get grouped together the number of projects included grows and as a result increases the applicability of the cluster. However, to compare methodologies that are derived from clusters with the same height in the cluster tree we can use the goodness of the clusters.

In the following sections we analyze different traces and the projects that have caused them in order to find patterns which allow generalization of the observed correlation. We then analyze clusters of traces and their corresponding projects in order to formulate the a set of candidate methodologies.

10.6 Clustering the Problems

The purpose of this section is to cluster those problems that have followed the same trace into clusters of problems. That is, we want to study the similarity of problems that followed similar traces (for now only one trace and later clusters of traces). Similar problems are those that have the same value for a particular requirement or constraint (or a combination of requirements and constraints).

Similar to generalizing cluster of traces, the problems can be generalized too (as we will see in the following subsections). We have used the same clustering method that we used for traces to cluster the problems.

To demonstrate the clustering of problems that have followed the same trace we pick a small number of traces with the highest goodness. These are the traces that will have the most influence in generating the methodologies. We only have included the traces that have a goodness higher than 20 (equivalent to covering at least 20 projects) as are shown in Table 10-6. The clusters of the problems for each trace in Table 10-6 is given in Appendix C. In the following subsections we present the result of clustering projects that followed traces 0, 1, 2, 49, and 770.

Table 10-6. The Goodness of Traces with Highest Frequency.

Trace	frequency	coverage (%)	uniformity (normalized)	normalized goodness G_{CN}
1	72	7.50	1	75.0
2	64	6.67	1	66.7
1545	54	5.63	1	56.3
0	52	5.42	1	54.2
49	33	4.40	1	34.4
1537	28	2.92	1	29.2
1546	24	2.50	1	25.0
770	20	2.08	1	20.8

10.6.1 Trace 0

Trace 0 corresponds to the design approaches in Table 10-7. Note the lower the Approach Index the higher the preference is that the designer has for that approach, i.e., in this trace each designer has picked its "best" approach.

Table 10-7. Trace 0.

Designer	Approach Index	Approach
Kinematic 1	0	`base_at_left_below_midway_workspace_length`
Structural 2	0	`steel_stainless_AISI_302_annealed`
Structural 3	0	`safety_factor_3`
Structural 4	0	`hollow_round`
Kinematic 2	0	`link_lengths_ratio_0.5`
Kinematic 3	0	`theta1_is_alpha1_minus_alpha2`
Structural 1	0	`dimension_min_ratio_4`

Table 10-8 shows the set of generalized problems that followed Trace 0. All the problems that followed Trace 0 are limited to the workloads of 1.0 and 2.0 kg. Also, only type “M” of the workspace can be seen in the problems. That is, if we cluster the generalized problems of Table 10-8 (i.e., going up the cluster tree) a more generalized problem is obtained that can be characterized as having type “M” workspace and workload requirement of 2.0 kg or less.

Table 10-8. Generalized Problems that Followed Trace 0

Projects in the Cluster	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot (%)
1 to 12	0.01	1000	small-M	1	(3 | 2 | 1)	(50 | 40 | 20 | 10)
122 to 124, 127 to 128, 135 to 136, and 140	0.01	1000	big-M	(1 | 2)	(3 | 2)	(40 | 20 | 10)
241 to 248 except 245	0.01	100	small-M	1	(3 | 2)	(50 | 40 | 20 | 10)
481 to 492	0.001	1000	small-M	1	(3 | 2 | 1)	(50 | 40 | 20 | 10)

Table 10-8. Generalized Problems that Followed Trace 0

Projects in the Cluster	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot (%)
602 to 604, 607 to 608, 615 to 616, and 620	0.001	1000	big-M	(1 | 2)	(3 | 2)	(40 | 20 | 10)
721 to 728 except 725 and 726	0.001	100	small-M	1	(3 | 2)	(50 | 40 | 20 | 10)

10.6.2 Trace 1

Trace 1 corresponds to the approaches in Table 10-9.

Table 10-9. Trace 1.

Designer	Approach Index	Approach
Kinematic 1	0	base_at_left_below_midway_workspace_length
Structural 2	0	steel_stainless_AISI_302_annealed
Structural 3	0	safety_factor_3
Structural 4	0	hollow_round
Kinematic 2	0	link_lengths_ratio_0.5
Kinematic 3	0	theta1_is_alpha1_minus_alpha2
Structural 1	1	dimension_min_ratio_3

The projects that followed Trace 1 and the corresponding generalized problems are shown in Table 10-10.

Table 10-10. Generalized Problems that Followed Trace 1

Projects in the Cluster	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot (%)
13 to 24	0.01	1000	small-M	2	(3 | 2 | 1)	(50 | 40 | 20 | 10)
121, 126, 132, 134, 139, 146 to 148, 151 to 152, 159 to 160, 163 to 164, 171 to 172, 175 to 176	0.01	1000	big-M	(1 | 2 | 3 | 4 | 5)	(3 | 2 | 1)	(50 | 40 | 20 | 10)
246, 252, 254 to 256, 259 to 260	0.01	100	small-M	(1 | 2)	(3 | 2 | 1)	(40 | 20 | 10)
364	0.01	100	big-M	1	3	10
493 to 504	0.001	1000	small-M	2	(3 | 2 | 1)	(50 | 40 | 20 | 10)
614, 619, 626 to 628, 631 to 632, 639 to 640, 643 to 644, 651 to 652, 655 to 656	0.001	1000	big-M	(2 | 3 | 4 | 5)	(3 | 2)	(40 | 20 | 10)
726, 732, 734 to 736, 739 to 740	0.001	100	small-M	(1 | 2)	(3 | 2 | 1)	(40 | 20 | 10)

To generalize all the problems of Table 10-10 we observe that for all the projects that followed “Trace 1” only “M” type of workspaces were used. Also, if the constraints on the deflection and the gain both become tight, then the path of “Trace 1” is followed only for “small-M” type of problems.

Path of “Trace 1” is taken because Trace 0 could not generate a successful solution. The first approach of `Designer_S_1` in Trace 0 would generate too large cross sections for the links that violates the constraint on the acceptable ratio of the cross section dimension of the link to its length that was set to 0.1—that is the dimensions of the cross section of the links could not be larger than 0.1 of the link length.

Due to the violation of the constraint on the dimension of the cross section the system takes the next available path that corresponds to Trace 1. The following is the report generated by **RD** for Project 13, reporting the violation of the aforementioned constraint.

```
- unresolved constraints are:

constraint constraint_2_1_1 of type
numeric_continuous_b<x<=c:
0.0 < link1_cross_section_dimension (0.0859) <= 0.0768

constraint constraint_2_1_2 of type
numeric_continuous_b<x<=c: 0.0 <
link2_cross_section_dimension (0.0466) <= 0.0384
```

Figure 10-6. Constraint Violation in Project 13.

10.6.3 Trace 2

Trace 2 corresponds to the design approaches in Table 10-11. The difference between Trace 2 and the two previous traces is that the `Designer_S_1` had to reduce the dimension of the cross section of the link even further to find a satisfactory design.

Table 10-11. Trace 2.

Designer	Approach Index	Approach
Kinematic 1	0	`base_at_left_below_midway_workspace_length`
Structural 2	0	`steel_stainless_AISI_302_annealed`
Structural 3	0	`safety_factor_3`
Structural 4	0	`hollow_round`
Kinematic 2	0	`link_lengths_ratio_0.5`

Table 10-11. Trace 2.

Designer	Approach Index	Approach
Kinematic 3	0	`theta1_is_alpha1_minus_alpha2`
Structural 1	2	`dimension_min_ratio_2`

Table 10-12. Generalized Problems that Followed Trace 2

Projects in the Cluster	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot (%)
25 to 36	0.01	1000	small-M	3	(3 | 2 | 1)	(50 | 40 | 20 | 10)
37 to 48 except 45	0.01	1000	small-M	4	(3 | 2 | 1)	(50 | 40 | 20 | 10)
49 to 60 except 57	0.01	1000	small-M	5	(3 | 2 | 1)	(50 | 40 | 20 | 10)
125, 131, 133, 138, 144 to145, 156 to 157, 158, 170	0.01	1000	big-M	(1 | 2 | 3 | 4 | 5)	(3 | 2 | 1)	(50 | 40 | 20 | 10)
245, 251, 253, 264, 266 to 268, 271 to 272, 178 to 280, 283 to 284, 291 to 292, 295 to 296	0.01	100	small-M	(1 | 2 | 3 | 4 | 5)	(3 | 2 | 1)	(50 | 40 | 20 | 10)
363, 376	0.01	100	big-M	(1 | 2)	3	(20 | 10)

For all the projects that followed Trace 2 the constraint on the deflection is only limited to 0.01. Also, as in Trace 0 and Trace 1, only type M workspaces appear in the requirements.

10.6.4 Trace 49

Trace 49 corresponds to the design approaches in Table 10-13. Compare to the three previous traces that we studied so far, Trace 49 is less desirable because `Designer_S_3` had to reduce the safety factor from 3 to 2.

Table 10-13. Trace 49.

Designer	Approach Index	Approach
Kinematic 1	0	`base_at_left_below_midway_workspace_length`
Structural 2	0	`steel_stainless_AISI_302_annealed`
Structural 3	1	`safety_factor_2`
Structural 4	0	`hollow_round`
Kinematic 2	0	`link_lengths_ratio_0.5`
Kinematic 3	0	`theta1_is_alpha1_minus_alpha2`
Structural 1	1	`dimension_min_ratio_3`

Table 10-14. Generalized Problems that Followed Trace 49

Projects in the Cluster	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot (%)
505 to 516 except 513	0.001	1000	small-M	3	(3 | 2 | 1)	(50 | 40 | 20 | 10)
517 to 528 except 525	0.001	1000	small-M	4	(3 | 2 | 1)	(50 | 40 | 20 | 10)
733	0.001	100	small-M	2	3	50
746 to 748, 751 to 752	0.001	100	small-M	3	(3 | 2)	(50 | 40 | 20 | 10)
758 to 760, 763 to 764	0.001	100	small-M	4	(3 | 2)	(40 | 20 | 10)

Trace 49 has been followed only for the situations where the constraint on the deflection is tight (maximum 0.001 of the sum of the link lengths) and only for "small-M" type of workspace.

10.6.5 Trace 770

The design approaches that are followed in Trace 770 are shown in Table 10-15. In this trace compare to previous traces that we have studied so far, an important change has happened in the design approach of `Designer_K_1`. None of the approaches based on putting the base of the robot at the left or below the length as well as the width of the workspace rectangle have produced a satisfactory design. Putting the base of the robot at the right or above the workspace rectangle, however, has been successful. Later we will justify this decision by `Designer_K_1` based on the shape of the workspace in the requirements.

Table 10-15. Trace 770.

Designer	Approach Index	Approach
Kinematic 1	2	`base_at_right_above_midway_workspace_length`
Structural 2	0	`steel_stainless_AISI_302_annealed`
Structural 3	0	`safety_factor_3`
Structural 4	0	`hollow_round`
Kinematic 2	0	`link_lengths_ratio_0.5`
Kinematic 3	0	`theta1_is_alpha1_minus_alpha2`
Structural 1	2	`dimension_min_ratio_2`

Table 10-16. Generalized Problems that Followed Trace 770

Projects in the Cluster	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot (%)
61, 66, 72	0.01	1000	small-L	1	(3 | 2 | 1)	(50 | 40 | 10)
73 to 74	0.01	1000	small-L	2	3	(50 | 40)
86 to 88, 91 to 92	0.01	1000	small-L	3	(3 | 2)	(20 | 10)
98 to 100, 103 to 104	0.01	1000	small-L	4	(3 | 2)	(40 | 20 | 10)
111 to 112, 115 to 116	0.01	1000	small-L	5	(3 | 2)	(20 | 10)
304	0.01	100	small-L	1	3	10

For projects that followed the path of Trace 770 both constraints on the deflection of the tip and the gain of the controller are loose (except the last project in which the constraint on the gain become tight that is compensated by loosening the requirements on workload and settling time).

The interesting feature in all these projects is that all of them have the "small-L" type of workspace as their requirements. It is interesting because the shape of the workspace is not symmetric as "M" type, hence there is a difference in putting the base of the robot to the right or left (because the orientation of the rectangle that circumscribes the workspace is vertical) of the workspace. That is, the first approach of `Designer_K_1` that puts the base to the left of the workspace has failed to satisfy the constraints, whereas switching to the right of the workspace has recovered the constraint failure.

10.7 First Set of Clusters

In previous sections we studied the clustering of traces and demonstrated sample results of clustering their corresponding problems. We mentioned that to extract methodologies we need these two types of clusters, i.e., traces and problems. We also mentioned that extracting methodologies based on clusters of traces from different rows in the cluster tree is a trade-off between the methodology being too abstract but applicable to a large number of different situations or being detailed and applicable to a small number of situations. The approach for extracting the methodologies, however, is the same. To demonstrate the approach of extracting methodologies, in this section we use clusters at height 1 in the cluster tree. These clusters generate the most detailed methodologies (except for methodologies based on traces themselves).

The process of extracting the methodologies can be automated based on the approach proposed by Reich [91] and summarized in page 116. The idea is instead of clustering the traces and then clustering the problems that have followed each cluster, merge the traces and problems and cluster the resulted combined trace.

In this dissertation we extract the methodologies manually in order to explicitly show the correlations that exist between traces and problems. In order to do that we find patterns in a cluster of traces regarding the use of specific design approaches that correspond to a cluster of problems.

The first round of clustering traces has led to 34 clusters. The most populated clusters along with their goodness number based on Equation 10-5 are shown in Table 10-17.

Table 10-17. Goodness of First Level Clusters

Cluster	number of traces	coverage (%)	uniformity (normalized)	goodness
1-0	9	28.12	0.7971	224.2
1-8	4	8.43	0.7857	66.3
1-16	3	3.95	0.8571	33.9
1-2	8	3.95	0.6429	25.4
1-5	4	2.70	0.7857	21.3

Table 10-17. Goodness of First Level Clusters

Cluster	number of traces	coverage (%)	uniformity (normalized)	goodness
1-4	4	2.50	0.7857	19.6
1-6	4	2.29	0.7857	18.0

10.7.1 Cluster 1-0

The first cluster, Cluster 1-0, covers 270 projects and includes 9 traces. Roughly speaking Cluster 1-0 is the collection of projects with mostly loose constraints and requirements. Projects that have more tight requirements and constraints have used approaches 4 and 5 of the Designer_K_1 to reduce the length of the links.

Table 10-18. Cluster 1-0. Total 9 traces covering 270 projects.

Designer	Approach Index	Approach

Kinematic 1	0,
	2,
	4,
	5
		`base_at_left_below_midway_workspace_length`
		`base_at_right_above_midway_workspace_length`
		`minimize_accessible_region`
		`minimize_link_lengths_summation`

Structural 2	0	`steel_stainless_AISI_302_annealed`
Structural 3	0	`safety_factor_3`
Structural 4	0	`hollow_round`
Kinematic 2	0	`link_lengths_ratio_0.5`
Kinematic 3	0	`theta1_is_alpha1_minus_alpha2`

Structural 1	0,
	1,
	2
		`dimension_min_ratio_4`
		`dimension_min_ratio_3`
		`dimension_min_ratio_2`

For example the first project that has taken a different path than of the Cluster 1-0 is project 45 in which `Designer_K_3` has chosen a right-hand configuration for the robot arm as opposed to a left-hand configuration (see Figure 2-3 on page 31). Both projects in the neighborhood of project 45 followed the path shown in Cluster 1-0.

Let us look at the requirements and constraints for these three projects to see what has caused for project 45 to take a different route:

Table 10-19. Comparing the Requirements and Constraints for Projects 44, 45, 46.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
44	0.01	1000	small-M	4.0	2.0	10%
45	0.01	1000	small-M	4.0	1.0	50%
46	0.01	1000	small-M	4.0	1.0	40%

Project 45 has taken the trace 1926 while projects 44 and 46 both have followed the route of trace 2, Table 10-20.

Table 10-20. Traces taken by projects 44, 45, 46.

Designer	Approach Index	Approach
Kinematic 1	44, 46 **45**	`base_at_left_below_midway_workspace_length` **`minimize_link_lengths_summation`**
Structural 2	44, 45, 46	`steel_stainless_AISI_302_annealed`
Structural 3	44, 45, 46	`safety_factor_3`
Structural 4	44, 45, 46	`hollow_round`
Kinematic 2	44, 45, 46	`link_lengths_ratio_0.5`
Kinematic 3	44, 46 **45**	`theta1_is_alpha1_minus_alpha2` **`theta1_is_alpha1_plus_alpha2`**
Structural 1	44, 45, 46	`dimension_min_ratio_2`

Trace 1926 that is followed by Project 45 is clustered with eight other traces into Cluster 1-0 which due to its small population of projects and traces will be dropped from consideration for generating methodologies at this level.

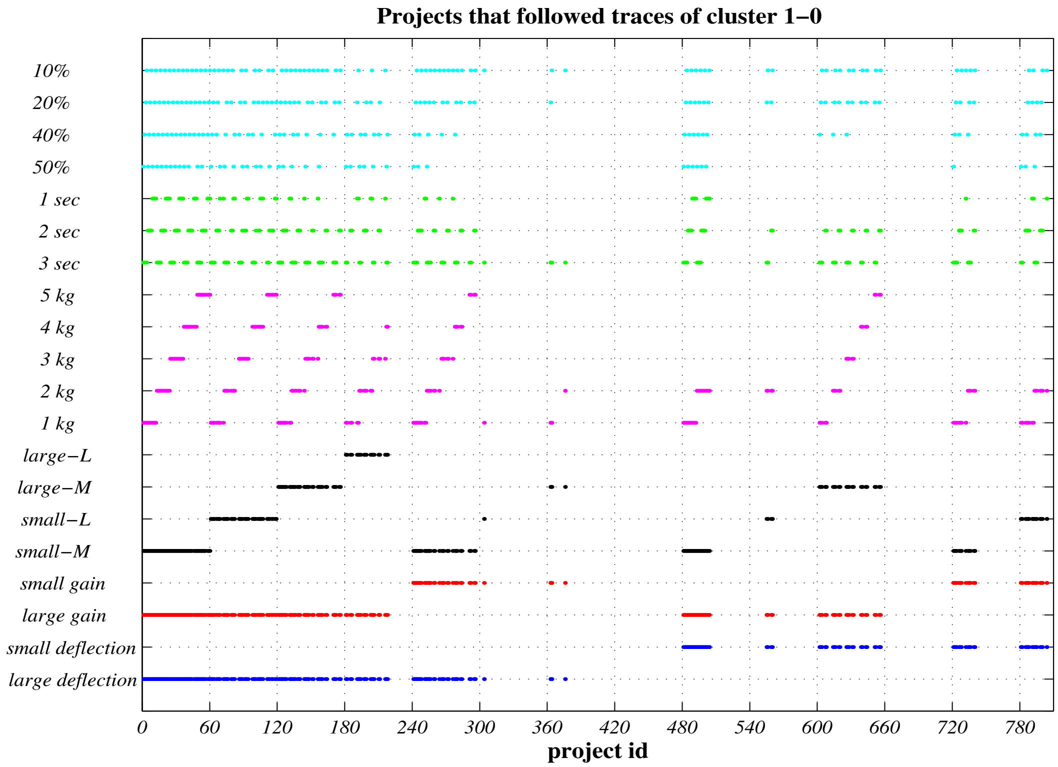


Figure 10-7. Distribution of Constraints and Requirements for Projects of Cluster 1-0

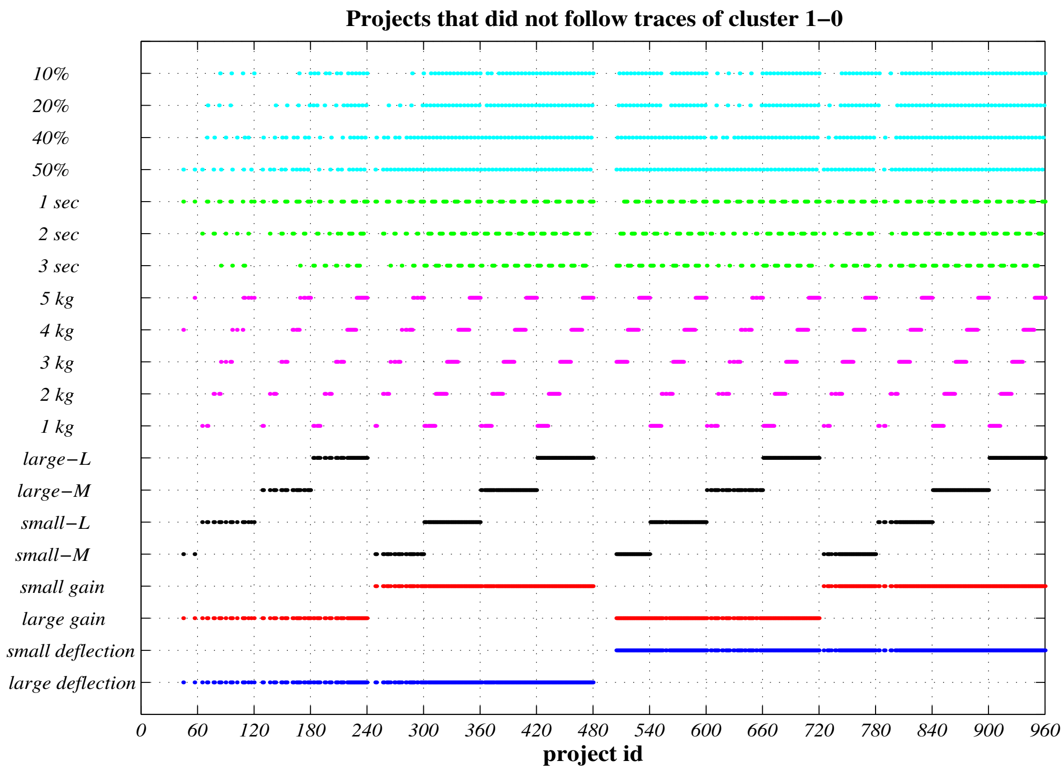
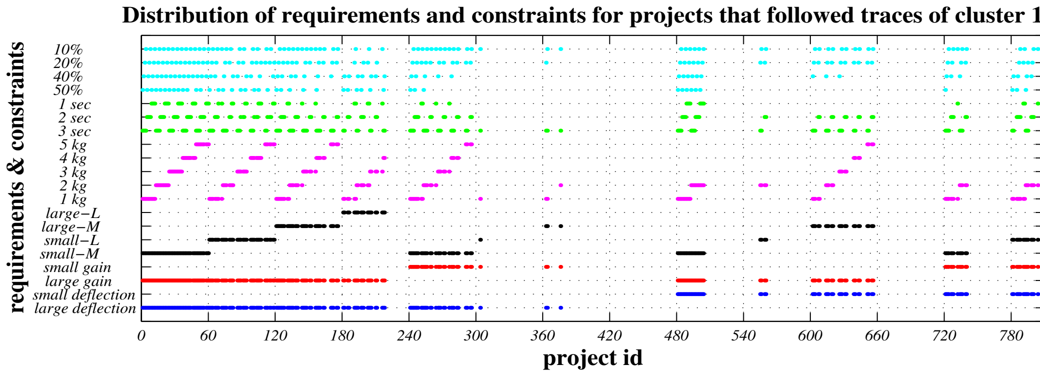


Figure 10-8. Constraints and Requirements for Projects that did not follow Cluster 1-0



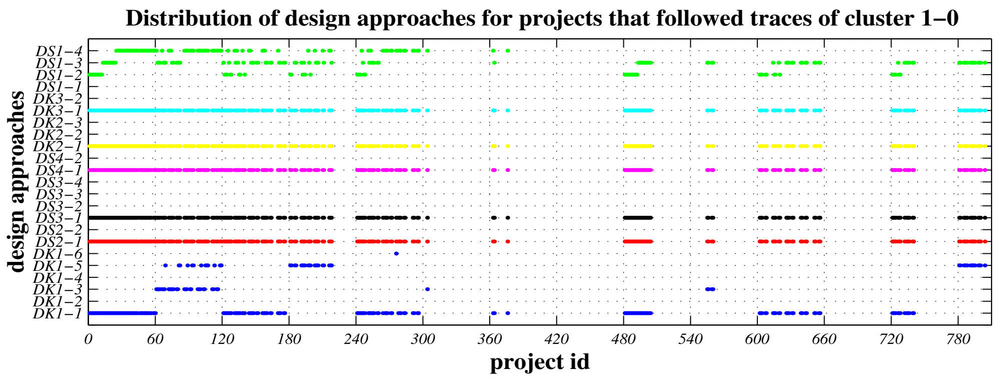


Figure 10-9. Comparing the trace of constraints and requirements with the trace of design approaches for Projects that followed Cluster 1-0

Based on the correlation between the change of requirements and constraints with the change of design approaches in Figure 10-9 the candidate methodology in Figure 10-10 is extracted:

METHODOLOGY 1-0

IF	• *constraints on deflection and the gain are loose,*		
	• *workspace is of type "small-M";*		
THEN	**IF**	• *requirement on the workload is easy, i.e., less than 1.0 kg;*	
	THEN	• *for designers use their first or default approaches.*	
	ELSE IF	• *requirement on the workload is in the range of 2.0 kg;*	
	THEN	• *use a dimension for the cross section that is not more than 3 times the minimum required dimension by stress criteria.*	
	ELSE IF	• *requirement on the workload is more than 2.0 kg;*	
	THEN	• *use a dimension for the cross section that is not more than 2 times the minimum required dimension by stress criteria.*	
ELSE IF	• *constraints on deflection and the gain are loose,*		
	• *but workspace is of type "small-L", and*		
	• *any workload requirement;.*		
THEN	• *put the base of the robot at the right or above the length of workspace—**if it fails** put it in a location that minimizes the accessible region,*		
	• *use a dimension for the cross section that is not more than 3 times the minimum required dimension by stress criteria—**if it fails** reduce the ratio to 2.*		

Figure 10-10. Methodology 1-0.

METHODOLOGY 1-0, continued

ELSE IF • *constraints on deflection and the gain are loose;*

THEN **IF** • *workspace is of type "big-M";*

THEN • *use a dimension ratio for the cross section equal to 4—**if it fails** reduce the ratio to 3—**if it fails** reduce it to 2,*

• *for all other designers use their first or default approaches.*

ELSE IF • *workspace is of type "big-L", and*

• *the workload requirement is not very tight (i.e., lass than 5.0 kg;*

THEN • *put the base of the robot in a location that minimizes the accessible region,*

• *use a dimension ratio for the cross section equal to 4—**if it fails** reduce the ratio to 3—**if it fails** reduce it to 2,*

• *for all other designers use their first or default approaches.*

ELSE IF • *constraints on deflection is loose, but on gain is tight,*

• *workspace is of type "small-M";*

THEN • *use a dimension ratio for the cross section equal to 4—**if it fails** reduce the ratio to 3—**if it fails** reduce it to 2,*

• *for designers use their first or default approaches.*

Figure 10-10. Methodology 1-0, continued.

METHODOLOGY 1-0, continued

ELSE IF
- *constraint on deflection is tight,*
- *constraint on the gain is loose,*
- *workspace is not of type "big-L";*

THEN
- *use a dimension ratio for the cross section equal to 4—**if it fails** reduce the ratio to 3,*
- *for workspace of type "small-L" put the base of the robot at the right or above the length of workspace, for type "M" workspaces put the base of the robot at the left or below the length of workspace,*
- *for all other designers use their first or default approaches.*

ELSE IF
- *constraints on deflection and the gain are both tight, and*
- *requirements on workload and settling time are rather "easy";*

THEN

  IF
  - *workspace is of type "small-M";*

  THEN
  - *use a dimension ratio for the cross section equal to 4—**if it fails** reduce the ratio to 3,*
  - *for all other designers use their first or default approaches.*

  ELSE IF
  - *workspace is of type "small-L";*

  THEN
  - *put the base of the robot in a location that minimizes the accessible region,*
  - *use a dimension ratio for the cross section equal to 3,*
  - *for all other designers use their first or default approaches.*

Figure 10-10. Methodology 1-0.

A part of Methodology 1-0 that might seem counter-intuitive is when the workload increases the methodology suggests reducing the dimension ratio of the cross section of the links. This is the ratio of the dimension of the cross section (RDCS) of the link to the minimum allowable dimension based on stress criteria.

The fact is that while the RDCS is reduced, the absolute value for the dimension of the cross section is increased. The reason is that the minimum allowable dimension by stress criteria is increased because of the increase in the workload. On the other hand, there is a constraint on the maximum ratio of the dimension of the link to its length (e.g., 0.1). Increase in the dimension of the link violates this constraint.

To recover from this constraint violation the system tries a smaller RDCS that reduces the dimension of the link to a point that it satisfies all the relevant constraints on dimension and maximum stress as well as the deflection of the links.

Figure 10-11 shows an example of the above situation that happens for Project 38 as it is reported by the system. The default path (index 0) that corresponds to RDCS equal to 4 is rejected because it violates the constraint on the size of the dimension of the cross section of both links. The second path that corresponds to RDCS equal to 3 is rejected too because it violates the dimension constraint on the second link. The third path, however, is successful in which RDCS is equal to 2 that satisfies all the constraints.

```
>> design state: 2 with parent ID: 1 at depth: 2 was created

  - index of rejected path: 0

  - unresolved constraints are:
    constraint constraint_2_1_1
0.0 < link1_cross_section_dimension (0.1036) <= 0.0768
    constraint constraint_2_1_2
0.0 < link2_cross_section_dimension (0.0585) <= 0.0384

  - index of rejected path: 1

  - unresolved constraints are:
    constraint constraint_2_1_2
0.0 < link2_cross_section_dimension (0.0439) <= 0.0384

>> design state: 3 with parent ID: 2 at depth: 3 was created

0.0 < link1_cross_section_dimension (0.0458) <= 0.0768
0.0 < link2_cross_section_dimension (0.0292) <= 0.0384
```

Figure 10-11. Failure Recovery by Reducing RDCS.

10.7.2 Cluster 1-8

Cluster 1-8 is the next cluster with the most population of projects that includes 4 traces covering 81 projects, Table 10-21.

Table 10-21. Cluster 1-8. Total 4 traces covering 81 projects.

Designer	Approach Index	Approach
Kinematic 1	0, 4	`base_at_left_below_midway_workspace_length` `minimize_accessible_region`
Structural 2	0	`steel_stainless_AISI_302_annealed`
Structural 3	0	`safety_factor_3`
Structural 4	0	`hollow_round`
Kinematic 2	1, 2	`link_lengths_ratio_0.75` `link_lengths_ratio_1.0`

Table 10-21. Cluster 1-8. Total 4 traces covering 81 projects.

Designer	Approach Index	Approach
Kinematic 3	0	`theta1_is_alpha1_minus_alpha2`
Structural 1	1, 2	`dimension_min_ratio_3` `dimension_min_ratio_2`

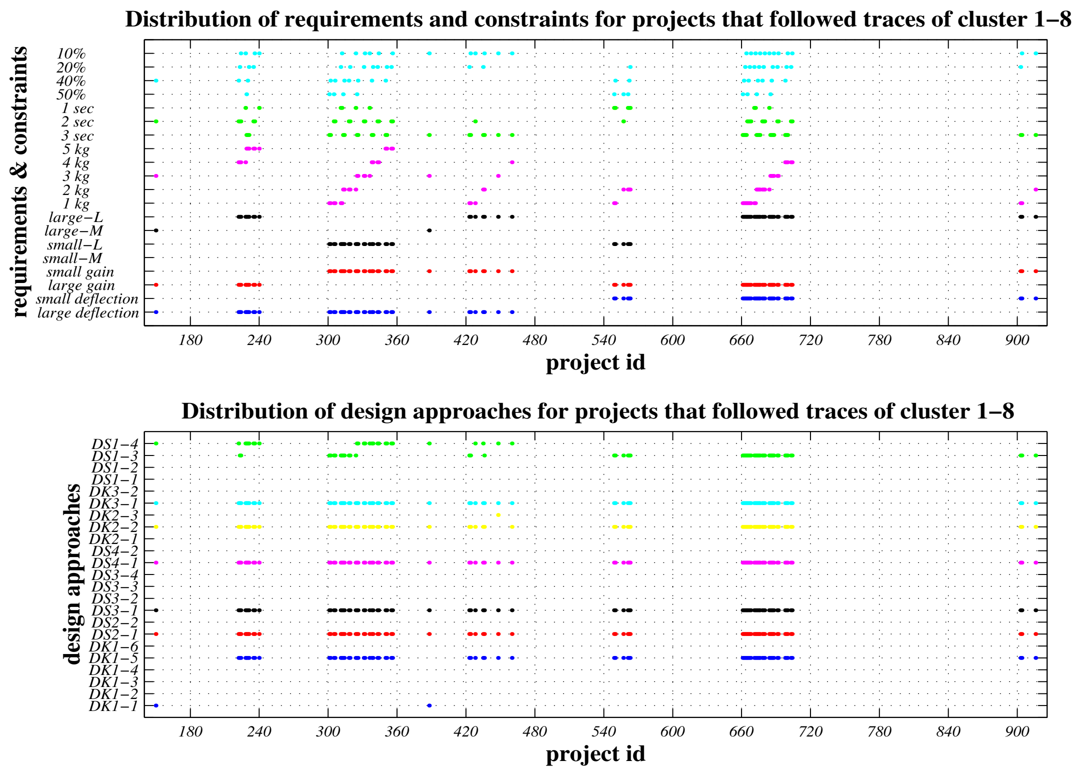


Figure 10-12. Comparing the trace of constraints and requirements with the trace of design approaches for Projects that followed Cluster 1-8.

Figure 10-12 shows that except two occasions (projects 150 and 388) `Designer_K_1` has put the base of the robot in a location that minimizes the accessible region area. Also, except in one occasion (Project 448) `Designer_1_2` has used a length for the second link that is 0.75 time of the first link.

This set of projects can be divided into two subsets based on the requirements on the workload. `Designer_S_1` has used a dimension that is twice the minimum dimension required by stress criteria for projects that have requirements of workload larger than 2.0 kg, settling time less than 2.0 seconds (most of the time), and overshoot less than 40% (most of the time). For this subset of projects the constraint on deflection is “loose”, i.e., equal to 0.01 of the sum of the link lengths.

For the second subset of projects, `Designer_S_1` switches back to the approach that requires the dimension of the cross section of the link to be 3 times the minimum required by the stress criteria. This happens when the requirement on the workload reduces to be not larger than 2.0 kg. For this subset of projects the constraint on deflection can be either “loose”, i.e., equal to 0.01 of the sum of the link lengths or tight, i.e., equal to 0.001 of the sum of the link lengths.

The above observations leads to formulation of Methodology 1-8 as is shown in Figure 10-13.

METHODOLOGY 1-8

IF	• *workspace is of type "L",*	
THEN	**IF**	• *constraint on deflection is "loose",* • *requirement on the workload is tough, i.e., 3.0 kg or more, and* • *settling time less than 2.0 seconds, and overshoot less than 40%;*
	THEN	• *use a cross section that is 2 times of the dimension based on stress analysis*
	ELSE IF	• *requirement on the workload is loose, i.e., 2.0 kg or less;*
	THEN	• *use a cross section that is 3 times the dimension based on stress analysis*
	• *put the base of the robot at a location that minimizes the accessible region area,* • *select the length of link 2 to the length of link 1 to be 0.75—**if it fails** choose them to be equal in length,* • *for all other designers use their first approach.*	

Figure 10-13. Methodology 1-8.

10.7.3 Cluster 1-16

Cluster 1-16 covers 38 projects and includes 3 traces, Table 10-22.:

Table 10-22. Cluster 1-16. Total 3 traces covering 38 projects.

Designer	Approach Index	Approach
Kinematic 1	0	`base_at_left_below_midway_workspace_length`
Structural 2	0, 1	`steel_stainless_AISI_302_annealed` `aluminum_alloy_5456_H116`
Structural 3	1	`safety_factor_2`
Structural 4	0	`hollow_round`
Kinematic 2	0	`link_lengths_ratio_0.5`
Kinematic 3	0	`theta1_is_alpha1_minus_alpha2`
Structural 1	0, 1	`dimension_min_ratio_4` `dimension_min_ratio_3`

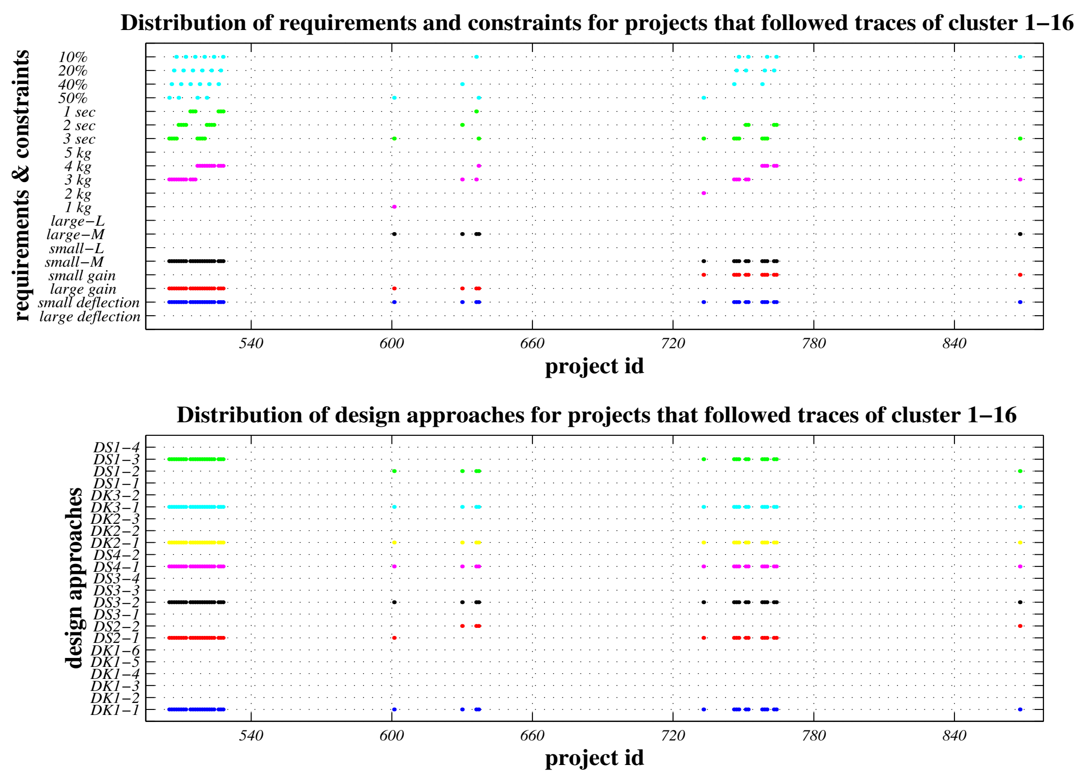


Figure 10-14. Comparing the trace of constraints and requirements with the trace of design approaches for Projects that followed Cluster 1-16.

Based on Figure 10-14 we can extract Methodology 1-16 that is shown in Figure 10-15.

METHODOLOGY 1-16

IF
- *constraint on deflection is "tight", and*
- *requirement on the workload is between 3.0 to 4.0 kg;*

THEN
- **IF**
  - *type of workspace is "small-M";*
- **THEN**
  - *use a cross section that is 3 times the dimension based on stress analysis,*
- **ELSE IF**
  - *type of workspace is "big-M";*
- **THEN**
  - *use a cross section that is 4 times the dimension based on stress analysis,*
- *use a safety factor of 2,*
- *for all other designers use their first approach.*

Figure 10-15. Methodology 1-16.

10.7.4 Cluster 1-2

Cluster 1-2 covers 38 projects and includes 8 traces, Table 10-23.

Table 10-23. Cluster 1-2. Total 8 traces covering 38 projects.

Designer	Approach Index	Approach
Kinematic 1	0, 2, 4	base_at_left_below_midway_workspace_lengthbase_ at_right_above_midway_workspace_length minimize_accessible_region
Structural 2	0, 1	steel_stainless_AISI_302_annealed aluminum_alloy_5456_H116
Structural 3	1, 2, 3	safety_factor_2 safety_factor_1.4 safety_factor_1.1
Structural 4	0	hollow_round
Kinematic 2	0	link_lengths_ratio_0.5
Kinematic 3	0	theta1_is_alpha1_minus_alpha2
Structural 1	0, 1, 2	dimension_min_ratio_4 dimension_min_ratio_3 dimension_min_ratio_2

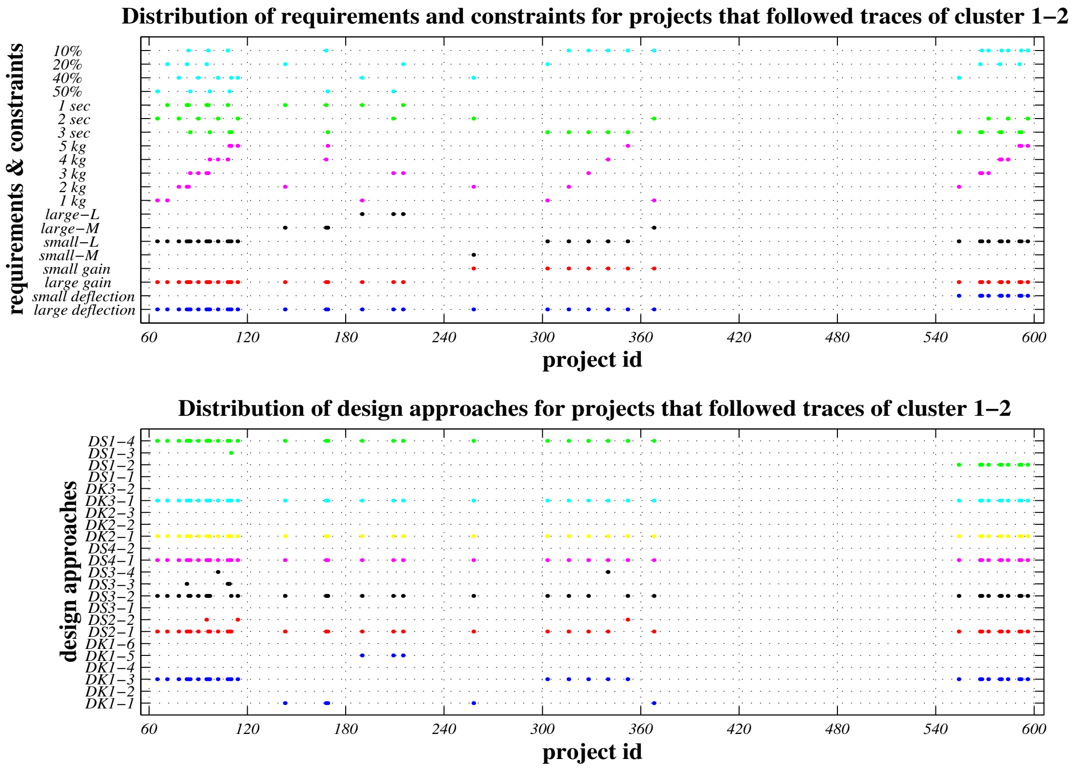


Figure 10-16. Comparing the trace of constraints and requirements with the trace of design approaches for Projects that followed Cluster 1-2.

Based on Figure 10-14 we can extract Methodology 1-16 that is shown in Figure 10-17.

METHODOLOGY 1-2

IF
- *constraint on deflection is "tight",*
- *constraint on gain is loose,*
- *type of workspace is "small-L",*
- *workload is 3.0 kg or higher,*
- *settling time is not less than 2.0 seconds, and*
- *overshoot is less than or equal to 20%;*

THEN
- *put the base of the robot to the right or above the length of the workspace,*
- *use a safety factor of 2,*

ELSE IF
- *constraints on deflection and gain both are "loose",*
- *type of workspace is "small-L",*

THEN
- *put the base of the robot to left or below the length of the workspace—**if it fails** put it to the right or above the length of the workspace—**if it fails** put it in a location that minimizes the accessible region area,*
- *use a safety factor of 2—if it fails reduce it to 1.4—**if it fails** reduce it to 1.1,*
- *use a cross section that is 2 times of the dimension based on stress analysis*

- *for the remaining designers use their first approach.*

Figure 10-17. Methodology 1-2.

10.7.5 Cluster 1-5

The design approaches that traces in Cluster 1-5 picked in producing successful designs are shown in Table 10-24. Notice that Designer_K_1, Designer_S_3, Designer_K_2, and Designer_S_1 in no case were able to achieve a successful design by their first approaches. Especially, Designer_K_1 had to use its expensive iterative approaches to minimize the accessible region or the link lengths.

Table 10-24. Cluster 1-5. Total 4 traces covering 26 projects.

Designer	Approach Index	Approach
Kinematic 1	4, 5	minimize_accessible_region minimize_link_lengths_summation
Structural 2	0	steel_stainless_AISI_302_annealed
Structural 3	1	safety_factor_2
Structural 4	0	hollow_round
Kinematic 2	1, 2	link_lengths_ratio_0.75 link_lengths_ratio_1.0
Kinematic 3	0	theta1_is_alpha1_minus_alpha2
Structural 1	1, 2	dimension_min_ratio_3 dimension_min_ratio_2

Cluster 1-5 includes only projects that have "L" type of workspace in their requirements. Also, except for Project 427 in all the other projects the requirement on the workload is rather tough—that is higher than 2.0 kg. Only for this project, Designer_K_1 has been able to find a successful design by minimizing the accessible region (approach minimize_accessible_region). For all other projects this designer had to use the more expensive iterative approach for minimizing the sum of link lengths (approach minimize_link_lengths_summation).

Another pattern that can be seen in Figure 10-18 is that as soon as the constraint on the deflection becomes “tight”, `Designer_S_1` switches its approach back to `dimension_min_ratio_3`.

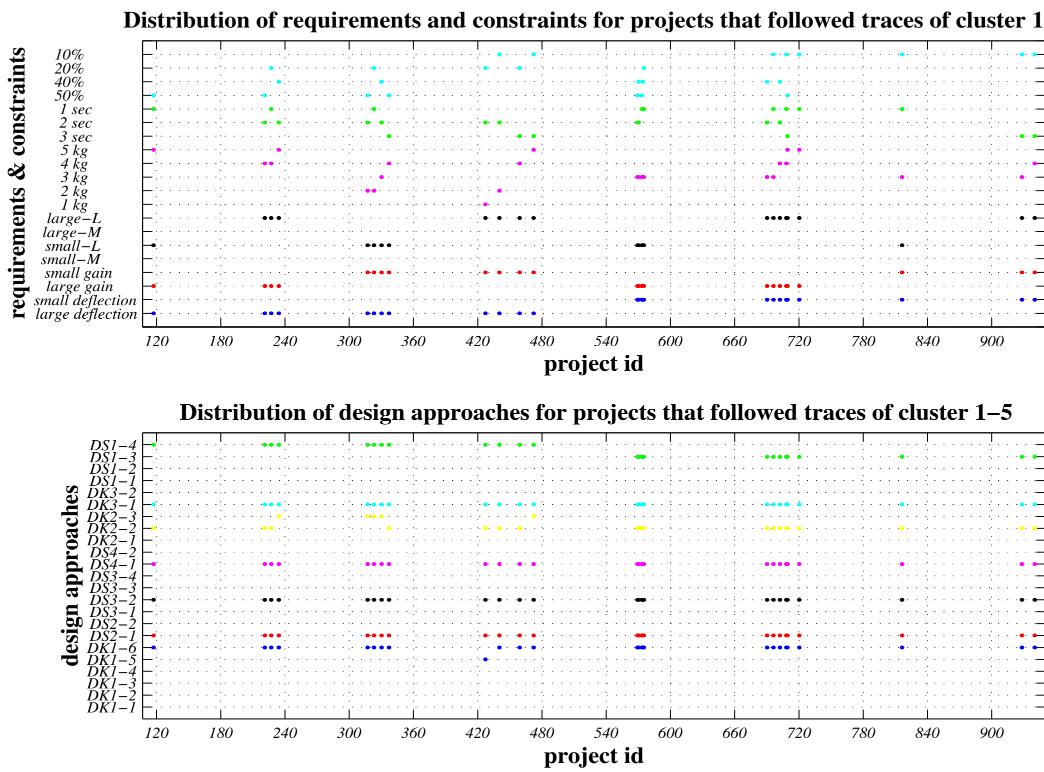


Figure 10-18. Comparing the trace of constraints and requirements with the trace of design approaches for Projects that followed Cluster 1-5.

Based on the above observations and Figure 10-18 we can extract Methodology 1-5 that is shown in Figure 10-19.

METHODOLOGY 1-5

IF
- *workspace is of type "L",*
- *requirement on the workload is rather tough—that is higher than 2.0 kg, and*
- *requirement on settling time is "not easy"—that is less than 3.0 seconds;*

THEN
- *always minimize the sum of the link lengths,*
- *always use a safety factor of 2,*
- *select the length of link 2 to the length of link 1 to be 0.75—if it fails choose them to be equal in length, and*

  IF
  - *constraint on deflection is tight*

  THEN
  - *use a cross section that is 3 times the dimension based on stress analysis;*

  ELSE
  - *use a cross section that is 2 times the dimension based on stress analysis;*

- *for all other designers use their first or default approaches.*

Figure 10-19. Methodology 1-5.

10.7.6 Cluster 1-4

The approaches that were chosen for Cluster 1-4 are shown in Table 10-31. In Cluster 1-4 `Designer_K_1` has used two of its basic approaches that puts the base of the robot along the length of the workspace. Also, `Designer_S_2` has used aluminum for the material instead of its first choice that is steel.

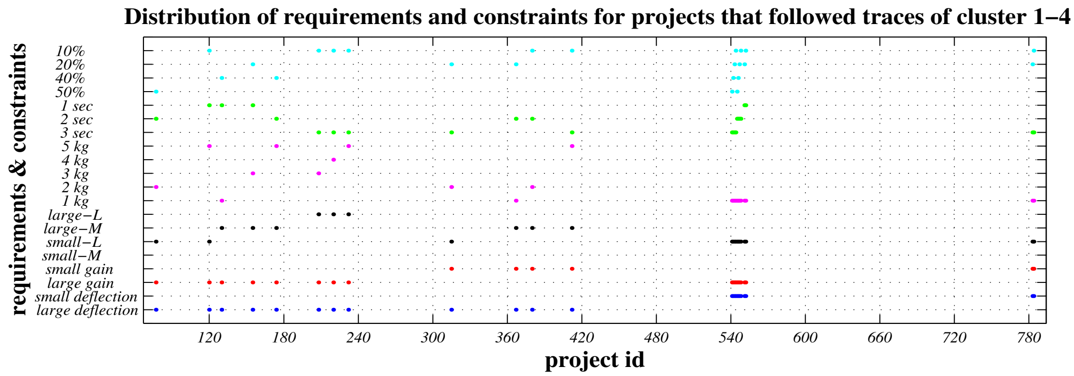
Table 10-25. Cluster 1-4. Total 4 traces covering 24 projects.

Designer	Approach Index	Approach
Kinematic 1	0, 2	base_at_left_below_midway_workspace_length base_at_right_above_midway_workspace_length
Structural 2	1	aluminum_alloy_5456_H116
Structural 3	0	safety_factor_3
Structural 4	0	hollow_round
Kinematic 2	0, 1	link_lengths_ratio_0.5 link_lengths_ratio_0.75
Kinematic 3	0	theta1_is_alpha1_minus_alpha2
Structural 1	0, 2	dimension_min_ratio_4 dimension_min_ratio_2

Figure 10-20 shows that there is a sub-group of projects in Cluster 1-4 with the following constraints and requirements:

1. “tight” constraints on deflection,
2. “loose” constraint on the gain of controller,
3. “small-L” type of workspace, and
4. “easy” requirement on the workload.

For the projects of this group, all designers have used their first approach except `Designer_K_1` that has used its third approach (that is putting the base of the robot at the above of the workspace length) and `Designer_S_2` that has used aluminum instead of steel.



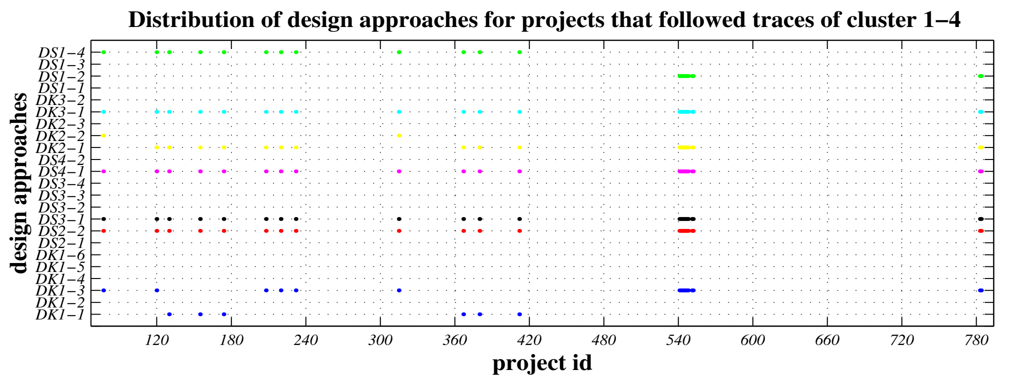


Figure 10-20. Comparing the trace of constraints and requirements with the trace of design approaches for Projects that followed Cluster 1-4.

The candidate methodology based on Cluster 1-4 is shown in Figure 10-21:

METHODOLOGY 1-4

IF

- *"tight" constraints on deflection,*
- *"loose" constraint on the gain of controller,*
- *"small-L" type of workspace, and*
- *"easy" requirement on the workload.*

THEN

- *first put the base of the robot at the top or to the right of work-space length—**if it fails** put the base below or to the left of the length of the workspace;*
- *always use aluminum;*
- *first choose the length of the second link to be half of the first link's length—**if it fails** choose a length ratio of 0.75;*
- *choose the dimension of the cross section to be either 2 or 4 times of the dimension based on stress analysis;*
- *for all other designers use their first or default approaches.*

Figure 10-21. Methodology 1-4.

10.7.7 Cluster 1-6

The design approaches used in Cluster 1-6 are shown in Table 10-26. One common characteristic of the projects that are included in Cluster 1-6 is that most of them have a "tough" requirement on workload (most of them 5.0 kg). They also, only include type "M" workspaces in their requirements and their requirements on settling time are moderate (mostly 2.0 sec).

Table 10-26. Cluster 1-4. Total 4 traces covering 22 projects.

Designer	Approach Index	Approach
Kinematic 1	0, 5	base_at_left_below_midway_workspace_length minimize_link_lengths_summation
Structural 2	0	steel_stainless_AISI_302_annealed
Structural 3	2, 3	safety_factor_1.4 safety_factor_1.1
Structural 4	0	hollow_round
Kinematic 2	0	link_lengths_ratio_0.5
Kinematic 3	0	theta1_is_alpha1_minus_alpha2
Structural 1	1, 2	dimension_min_ratio_3 dimension_min_ratio_2

Figure 10-23 reveals that Designer_K_1 has used its expensive approach of minimizing the sum of link lengths only once. Also, Designer_S_3 has used the safety factor of 1.1 only for two projects. Considering these three projects as exceptions, we can formulate a design methodology based on the common characteristics of the projects and traces of Cluster 1-6 as is shown in Figure 10-22:

METHODOLOGY 1-6

IF

- *the requirement on workload is "tough",*
- *the type of workspace is "M", and*
- *the requirement on settling time is moderate (i.e., 2.0 or 3.0 seconds;*

THEN

- *first put the base of the robot below or to the left of the length of the workspace—**if it fails** put the base in a location that minimizes the length of the links;*
- *first use a safety factor of 1.4—**if it fails** use a factor of 1.1;*
- *use a dimension for the cross section that is either 2 or 3 times the dimension based on stress analysis;*
- *for all other designers use their first or default approaches.*

Figure 10-22. Methodology 1-6.

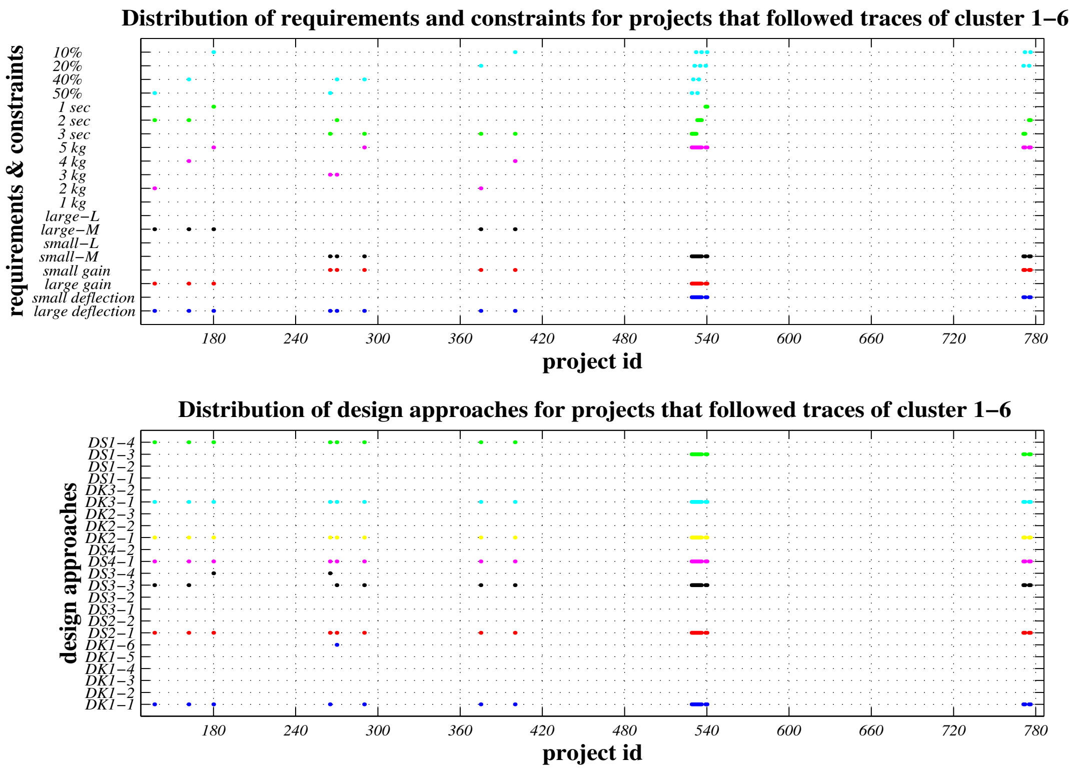


Figure 10-23. Comparing the trace of constraints and requirements with the trace of design approaches for Projects that followed Cluster 1-6.

10.8 Goodness of Methodologies

In “Formulating Methodologies” on page 228 we mentioned how the goodness of clusters can be used to compare different methodologies with each other. That is, based on how many different design situations a methodology is covering (coverage) and how many variations in the use of design approaches are needed (uniformity) the goodness of a methodology relative to other methodologies can be measured.

Another criteria, besides coverage and uniformity, that can be used to compare the goodness of methodologies is based on the desirability of the design approaches included in the methodologies. Design approaches are ordered in each designer based on their desirability such as their cost in the design process. For instance, iterative methods that do some sort of optimization are given less preference because they are time consuming. These types of approaches are tried only if other types of approaches (e.g., table lookup) that are faster do not produce successful designs.

By introducing a cost function that assigns higher cost to less desirable approaches we can compare different methodologies with respect to the desirability of their approaches. Choosing different weight factors for different approaches makes it possible to find methodologies that are biased toward some specific approaches. For instance, we may give more weight to approaches that produce more environmentally friendly products, e.g., optimize for fuel consumption. This places methodologies that include fuel optimization approaches at the top of the list.

Yet, another factor that affects the goodness of a methodology is the degree of parallelism in the methodology. This parallelism is directly related to the concurrency between designer agents as it is seen in the dependency graph, Figure 10-1. In general, the set of designer agents that participate in the design process might change from one project to another. As a result, the dependency graph and the degree of concurrency may change in different projects that will be reflected in the methodologies generated based on those projects.

10.9 Evaluation of Methodologies

The ultimate method for evaluation of the methodologies generated is to do a field test. That is, to introduce them to real design teams so that they could follow the methodologies in their design projects and report any improvement in:

- reducing the design time cycle,
- reducing conflicts,
- integrating different points-of-view,
- enhancing the parallelism between tasks,
- increasing information exchange, etc.

Testing the methodologies using real design teams, however, is very difficult. It is similar to, for instance, testing new techniques for seismic isolation on a full-scale multi-story building. Or, testing a new airfoil on an airplane before doing wind tunnel experimentations. In these two examples, like all other new technologies, the final approval comes from real life use of those technologies.

Field or real life tests of new approaches and techniques are very difficult, expensive, and highly risky. This is why these types of test are always done in the last stages. The approach proposed in this dissertation, generating design methodologies by simulation, and the design methodologies produced are not an exception to these rules. More research is required before the approach proposed in this dissertation can be field tested. The next chapter discusses the areas that this research should be investigated further.

The next evaluation method is to compare the methodologies generated with the current practice of design in a related company. However, access to information, with the right level of detail, on how a company conducts its design process is very difficult because of the proprietary issues. Resolving the practical issues of accessing proprietary information in a company that designs robots takes a long time and is not within the scope of a Ph.D. dissertation.

Yet another method to evaluate a new design methodology is to present it to an expert in the field and ask for his or her judgement. For multi-disciplinary type of design problems such an expert should be a generalist who can evaluate the methodology from a non-disciplinary point-of-view. Finding such people, however, is very difficult because they are usually in high demand, thus the amount of help that one might be able to receive from them is very limited.

In summary, similar to many other new approaches, more research needs to be done in extending and expanding the approach proposed in this dissertation before any of the above evaluation methods can be used (see “Extending to Other Domains” on page 291).

One important point is that the issue of evaluating design methodologies is not equivalent to the evaluation of this research work. The problem that we are tacking in this dissertation is that there are *no systematic approaches to building design methodologies.* Thus, the evaluation of this work would be evaluating the approach that we are proposing for generating methodologies. We discuss this issue in the next chapter.

11 Conclusions

In this final chapter we first review the problem, the proposed approach, and the results obtained. Next we revisit the goal that was set at the beginning of this dissertation. We then summarize the contributions of this dissertation and make some final conclusions. At the end we propose the directions in which this work can be extended and finally propose some other areas of research and practice that might benefit from the results of this dissertation.

11.1 Review of the Problem

In this section we review the problem that initiated the research and led to this dissertation. We then summarize the reasons that make the problem hard to solve and finally discuss the significance of solving it.

In Chapter 1 we stated the problem as the following:

> *"There are no systematic approaches to building design methodologies for integrating different disciplines in multi-disciplinary design so that they collaborate in both contributing to the common goals of the design and sharing resources".*

Multi-disciplinary design processes have three characteristics that make them very hard to integrate, Figure 11-1. Multi-disciplinary design processes are a type of distributed system (as opposed to centralized system) due to compartmentalization of the disciplines. Distributed systems are hard to integrate because of the problems of accessing information, the difficulty of communications, etc.

With respect to the representation of knowledge, multi-disciplinary design processes use heterogeneous knowledge sources. Heterogeneity can also be due to different level of abstraction and granularity of the knowledge. Integrating heterogeneous knowledge is difficult because of the lack of common languages and lack of common goals. Multi-disciplinary design processes are also very difficult to anticipate because the process emerges from the interactions of many components in the system. Being unpredictable is also a barrier to integration as the strategies for integration based on incremental or ad-hoc methods may not hold for all situations.

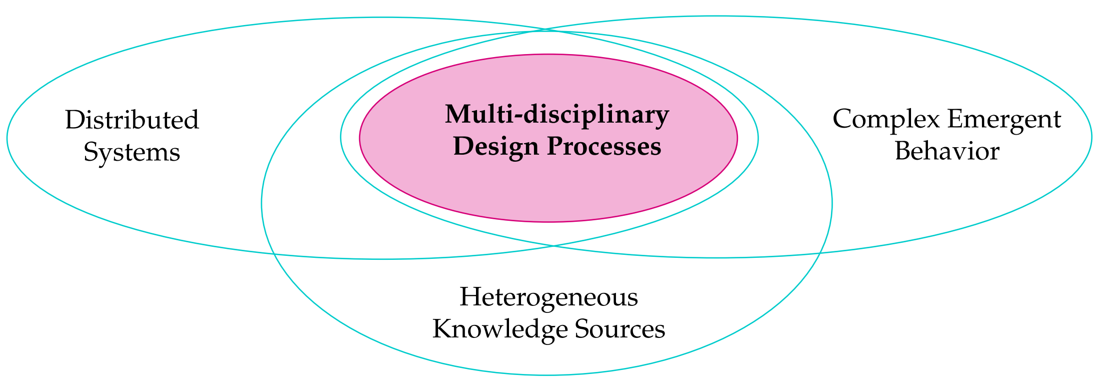


Figure 11-1. Multi-disciplinary Design Processes in Intersection of Three Hard Areas.

In Chapter 1 we gave an extensive list of motivations for the development of an approach that systematically discovers methodologies for multi-disciplinary design processes. In summary, the problem is worth solving because it has tremendous potential for enabling manufacturing companies to speed up their design processes. Another motivation is that, the recent advances in the area of artificial intelligence in design have provided powerful techniques and tools for solving the problem.

11.2 Revisiting the Goal

In Chapter 1 we stated the goal of this dissertation as the following:

> *"To synthesize design methodologies for rapid product development, thus reducing time-to-market."*

We then formed a hypothesis that we can improve multi-disciplinary design processes by simulation. That is, we can develop an approach based on simulation of the design process that systematically generates design methodologies. The design methodologies generated are superior because they breaking the boundaries between disciplines.

We proposed that the goal can be reached by a simulation of the design process based on a knowledge-based model of the design process. The idea is similar to the concept of analysis-by-simulation in engineering systems. Analyzing the behavior of physical systems in engineering applications by computer simulation using mathematical models has been a powerful tool in engineering, reducing costs and time in comparison to physical prototyping and experimentation.

Simulation of design processes is very difficult because of their complexity. However, to extract design methodologies we do not need to simulate the design process in full detail. Similar to the simulation of physical systems in the engineering analysis domain, appropriate simplification methods might be used to abstract the process and prune less important details. The key characteristics of the multi-disciplinary design process are captured in the knowledge-based model and are implemented in the simulation. The key characteristics of multi-disciplinary design processes that should be captured are shown in Figure 11-1 and include: distributed, heterogeneous knowledge, and emergent behavior.

Developing a system that actually implements the simulation has by no means been less challenging than the other parts of this dissertation. Recent advances in the area of multi-agent design systems helped us to overcome part of the difficulty of implementing a system that simulates the design process.

We implemented a multi-agent system for designing 2-DOF robots called **RD**. **RD** simulates a simplified knowledge-based model of the design process while preserving the characteristics of multi-disciplinary design processes as shown in Figure 11-1. The results from the system that were presented in chapters 9 and 10 show that **RD** is simulating the design process with quite a good accuracy.

11.3 Summary of the Results

In this section we summarize the results that **RD** produced.

1. **Sensitivity Analysis.** The system was used to study the effect of subsets of design parameters (i.e., requirements) on other design parameters. It was also used to investigate the effect of different design approaches on the design parameters generated. The sensitivity analysis let us to fine tune the system in terms of evaluating design approaches and/or re-ordering them.

2. **Dependency Graph.** The system builds the dependency graph on the fly. The dependency graph can reveal very important facts about the design process. It shows integration among designers from different disciplines, the degree of concurrency between designers, the bottlenecks in the design process, and many more facts.

3. **Traces.** The system produced a wide range of information regarding what steps it has taken in the design process and what resources have been used. These results can be used in many different ways to learn about how to improve the practice of design in real world. Our focus in this dissertation was to track the system's use off different design approaches in order to generate methodologies. Other types of conclusions can be made using the other types of traces. For instance, one can use traces of the system to study the effect of the design requirements, constraints, and approaches on the quality of the design.

4. **Methodologies.** Some design methodologies were generated based on first level clusters of the traces of the system. The higher the level of clusters the more abstract the methodologies will be. Therefore, there is a trade-off between containing enough details and covering a wide range of situations.

11.4 Evaluation of the Results

In this section we evaluate the results of the system to see if they conform with what we expected. We concentrate on the design methodologies generated and compare them with what we called better design methodologies in Chapter 4 (see "Better Design Methodology" on page 60). The following is the list of the features that better design methodologies should have. Each feature is followed by a brief discussion regarding whether the methodologies generated posses them or not.

- **Takes Less Time:** The methodologies generated are based on successful traces. Therefore there is a high chance that following those methodologies takes us directly to a solution that satisfies all the constraints along the way. That is, less iteration is needed and thus time is saved. Giving more priority to design approaches that takes less time makes the methodologies even more time effective. This is because the system first follows the traces which have higher priority design approaches.

- **Causes Fewer Failures:** The methodologies generated are based on traces of the system that have succeeded in producing a successful design and thus have less chance of encountering constraint violations.

- **Produces Better Designs:** Better designs are those with better quality or those that are simpler. We did not consider the quality of the product in producing traces. The reason is that this is an area of research that has been investigated thoroughly and many good approaches have been developed for optimizing design products, e.g., the methods from multi-disciplinary design optimization (MDO). However, one can incorporate the effect of the quality of the design product into traces generated by introducing global constraints that define the boundaries of near-optimal designs. This forces the system to find the design that satisfy those global constraints, hence producing better quality. For instance, introducing constraints that put limits on the maximum weight, cost, power consumption, etc. will produce designs that will have just those features (if there are any).

- **Works for a Wide Range of Design Requirements:** This feature has been incorporated into the mechanism that produces design methodologies. In "Goodness of a Cluster of Traces" on page 217 we described how the effect of the coverage of the traces are taken into account. In producing methodologies we picked the cluster of traces whose coverage combined with their uniformity was higher than others. This leads to methodologies that are applicable to a wide range of different situations.

- **Integrates Different Disciplines:** Figure 10-1 on page 214 clearly shows that different disciplines are integrated in the design process. In the first and third round, designers from kinematics and structural design and in the last round designers from all three disciplines have participated in the process of design. Integration of the disciplines is evident also from the methodologies generated. For instance, the mixture of design approaches from kinematic and structural design in the 'Methodology 1-0' (see Figure 10-10 on page 245) is an evidence of integration of disciplines.

- **Conducts Design Concurrently:** The dependency tree of Figure 10-1 on page 214 shows how multiple designers can design simultaneously. In Chapter 4 we discussed the strategies that are incorporated into the knowledge-based model of design (see "Strategies for a Knowledge-based Design System" on page 66). One of these strategies was to conduct the design process concurrently. Implementation of 'concurrency strategy' plus the 'opportunistic strategy' conducts the design process among different designers concurrently.

- **Consumes Less Resources:** Less resources, e.g., time and money, are expended in the design process due to less failure, integration, concurrency, etc., features that are incorporated into the design methodologies. Having access to design methodologies that are built systematically based on the latest available technology reduces the load on designers, especially the generalists that can now spend more time on more creative parts of the design process.

- **Requires Less Requirements and Information:** The methodologies generated are based on correlation between clusters of traces of the system and the requirements. Whenever possible we have pruned the less important features of the requirement set.

11.5 Outcome of the Research

The outcome of this research is an approach and a design system that can be used to generate design methodologies for non-routine parametric design problems. The summary of the approach is as follows:

1. **Type of Design.** Determine the type of the design process under consideration. If the type of the design process is parametric this approach can be used.

2. **Knowledge Acquisition.** Identify the design parameters that define the design product. Make a list of different disciplines involved in the design. Extract the design knowledge in each discipline including design methods, design constraints, domain preferences for using particular methods, etc.

3. **Small Design Methods.** Break the design knowledge into small segments. The rule is to break a method at the decision-making points. In parametric design, a decision point is when a value is assigned to a design parameter. For instance, when an equation is used, a table is looked up, a catalog is searched, an optimization procedure is con-

ducted, or a heuristic rule is used to find the value of a design parameter. Please note that the design parameter can be numeric, symbolic, scalar, vector, matrix or any other type as defined in the domain.

4. **Design Approaches.** Reorganize the design knowledge in small segments to form groups of related design approaches. Related design approaches are those that have similar inputs and outputs. Each group of related design approaches can be bundled together in the form of new design methods. Order the design approaches in each method so that those with higher preference are used first. The ordering of the design approaches can be based on the quality of the approach itself (e.g., fast, accurate, etc.) or based on its outcome (e.g., manufacturable, environmentally friendly, etc.).

5. **Designer Agents.** Insert each design method (i.e., group of reorganized related design approaches) in one designer agent.

6. **Multi-agent Design System.** Insert designer agents in the multi-agent design system. The multi-agent system that we have developed can be used as a shell into which designer agents can be added or removed. However, should a system be developed from scratch, the framework and the set of techniques and tools that was proposed in this dissertation can be used.

7. **Design Experiments.** Design of Experiment techniques enable one to gain a maximum amount of information in a minimum number of runs. Trade-offs as to the amount of information gained for the number of runs, are known before running the

experiments. Conducting sensitivity analyses such as those presented in Chapter 9 should be used to determine the appropriate range of variation for requirements as well as the sensitivity of the results to different constraints.

8. **Experiments.** Conduct the experiments by running the system for all combinations of requirements and constraints determined in 'Design of Experiments' stage. We referred to each unique combination of requirements and constraints as a *design problem* and to the problem plus its solution and other information about the traces of the system as a *design project*. Collect the information about how the system conducted the design process in the form of traces in order to be analyzed.

9. **Analysis of Traces.** Analyze the traces of the system in terms of the number of successful and unsuccessful projects, distribution of traces with respect to projects, discovering patterns among project that followed the same or similar traces, discovering correlation between clusters of projects and particular combination of design approaches, etc. This step can be automated using machine learning techniques such as concept formation using unsupervised learning techniques presented in Chapter 6.

10. **Generate Methodologies.** Generate the methodologies by finding the correlation between subsets of projects and subsets of traces. Methodologies can be represented in the form of rules, decision trees, or any other appropriate representation scheme.

11.6 Evaluation of the Outcome

In this section we evaluate the outcome of this research work by looking at some important aspects of using the approach proposed.

11.6.1 Return in Investment

There is a trade-off between resources saved due to using methodologies and the resources used to generate them. That is, on one hand we have to spend more resources (e.g., time, money, expertise, etc.) to generate methodologies with better quality. On the other hand, design methodologies with better quality save more resources (e.g., speed up the design). The return on the investment for generating methodologies increases by reducing the costs while increasing their use (Figure 11-2).

The factors that contribute to the use of the methodologies are as follow:

- **The Quality of the Methodologies.** Good quality methodologies get used more, hence they save more resources by guiding the design process. The quality of the methodologies generated can be enhanced by increasing their coverage, uniformity, desirability, and parallelism (see "Goodness of Methodologies" on page 266).
- **Existence of Current Methodologies.** In the domains that design methodologies with good quality do not exist, the use of the methodologies generated by our proposed approach will be higher. Multi-disciplinary design is an example of a situation in which the quality of the existing methodologies is poor. This is because the incremental and ad-hoc approach to improving the current methodologies is severely limited by the complexity of the design process. As a result, the methodologies that are embedding techniques for integrating different disciplines are superior and will get used more.

The factors that affect the cost of generating methodologies are the following:

- **Knowledge Acquisition and Knowledge Engineering.** Existence of methods and techniques for extracting the domain knowledge from the experts, published literature, etc., can dramatically reduce the cost of knowledge acquisition. Converting the acquired knowledge to the right form so that it can be used in the multi-agent design system can be time consuming and costly. Developing methods and tools that would assist in or automate this process is a step toward reducing the cost of generating methodologies. For instance, we proposed a rule for how to break the design knowledge into smaller pieces that can be used by the knowledge engineer to speed up the process (see "Small Design Methods" on page 277).

- **Building Multi-agent Design System (MADS).** The burden of building the multi-agent system that simulates the design process can be greatly reduced by developing multi-agent shells for design. We proposed a framework and set of techniques for building MADS that can be a starting point in developing such shell systems. Building the MADS using the shell will then be only the matter of pluging in the design knowledge.

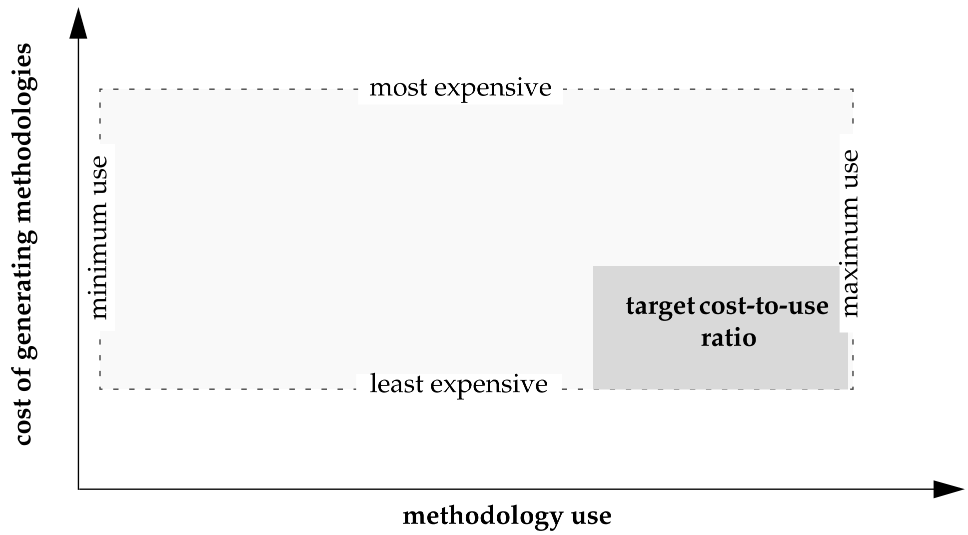


Figure 11-2. Return in Investment in Generating Methodologies.

11.6.2 Type of Design

The approach that we have proposed in this dissertation has been developed for non-routine, parametric types of design problems (obviously the approach is applicable to routine-parametric designs). Non-parametric types of design (e.g., configuration design, conceptual design, etc.) remain to be investigated. This limits the use of the approach. However, parametric designs cover a large portion of the design activities in practice.

11.6.3 Scalable

The methodology generation requires a lot of detailed investigation (in terms of knowledge acquisition) even for relatively simple problem/domain. One unanswered question is how this effort increases when the complexity/size of the problem scales up? We discussed that for extracting design methodologies we do not necessarily need a detailed model of the design process. However, to find the right level of detail and the required ingredients might not be straightforward.

11.6.4 Automated Extraction of Methodologies

In this dissertation we were able to automate the methodology generation to a high degree. We provided the reference to methods that can be used to automate the whole process (e.g., conception formation methods based on unsupervised learning techniques). It is not evident that these techniques scale up to real and more complex problems. There is clearly a need for a change of representation of the results from such automated techniques so that they can easily be understood by human users of the methodologies. Further investigation is needed to see whether this stage can be automated or not.

11.6.5 Quality of the Methodologies

The quality of methodologies has been verified only to the extent that they conform with the first principles of physics. For real-size problems such an evaluation approach may not be helpful because of the complexity of the problem. Formal methods of evaluation for methodologies are needed before the approach proposed in this dissertation can be used for real applications.

11.6.6 Quality of the Design

In the approach proposed the system accepts the first design that satisfies all the constraints. The methodologies are then generated based on the corresponding traces. The approach should be modified so that the system finds a near optimal design and methodologies are generated based on such results.

11.7 Contributions of the Research

The overall contribution of this work to engineering design research is to propose an approach and a tool for constructing better methodologies in parametric design. As a result of this research we are able to generate superior design methodologies that facilitate integration and collaboration between different disciplines, conduct design tasks concurrently, apply to a wide range of design problems, consume fewer resources at design time, and provide better quality for the product.

We divide the contributions of this research into the following categories: theoretical, experimental, implementation, and domain dependent.

11.7.1 Theoretical Contributions

Investigating and developing computational and knowledge-based models for the design process has provided us with some better insight into how a real design process might be conducted more efficiently.

For instance, how the design tasks should be delegated to the members of a design team so that little overlap happens in carrying out the tasks. Or, that to reduce the number of conflicts reorganize the teams so that all decisions about one design parameter are taken in one place, that is no two different designer decide about the same parameter. Also, by discovering the dependencies between designers, dependency-directed backtracking prevents the designers that won't have any effect on violated constraints from having to re-do their work.

A summary of the other theoretical contributions follows:

- Extending the technique of analysis by simulation to the area of analyzing design methodologies and in general to synthesizing the emergent behavior of complex systems.

- Decomposing the design process into small designers, coordinators, databases, and other utility agents (e.g., `ExceptionHandler`), as is done for **RD**, can be used in real design processes.

- Approximating the continuous design process with a discrete process comprising of cycles of run-analysis-update.

- The idea that conflict resolution between different participants in design can be aided by breaking the design knowledge into small pieces at the decision points (i.e., a decision about assigning values to design parameters). In these situations we will have a set of (atomic) design methods where some of them decide about the same parameter. Collect and bundle the design methods that are about the same design parameters into one design agent. In each agent, order and prioritize each method and let the agent decide what method it should use for assigning a value to the associated parameter. In this situation there is no conflict between designer agents in assigning values to design parameters. The same approach can be used for the control knowledge in which designer agents might not agree on what the order of the design tasks should be.

- Another contribution is to provide an approach for analyzing current methodologies for flaws and bottlenecks, and suggesting necessary refinements. New methodologies can be customized so that they are biased toward specific objectives such as manufacturability or "green design". By applying this approach the response time for the incorporation of new technologies or new design methods into design processes will be reduced. Methodologies can be refined as soon as a change occurs in the market or in the organization of the company.

- The algorithms we developed for backtracking lead us to an approach for how to break the design knowledge into pieces: We should break the design knowledge in a way that the most expensive methods are located in the shallower depths of the dependency graph (see page 169).

11.7.2 Experimental Contributions

The experiments conducted in this dissertation revealed some interesting results. The following is a summary of these results:

- An approach for discovering the dependency relations between design parameters is to use a multi-agent design system.

- The experimental studies reveal some valuable information regarding the sensitivity of the design to the design approaches as well as to various subsets of design parameters. The design process simulation approach can be used as an analysis tool, and for sensitivity studies in which quantitative and qualitative measurements are formed to show the effect of inputs (requirements/constraints) on outputs (the product attributes). This type of sensitivity analysis is very valuable when analytical or computational models cannot be built for the physics of the problem.

11.7.3 Implementation Contributions

Developing multi-agent design systems is very difficult and time consuming. The framework that we have proposed for implementation as well as the implementation decisions that we have made might be helpful in the future in developing new systems. The framework and other implementation decisions that we proposed are results of many iterations and failures in the process of developing the system.

Besides simulating the design process for the purpose of generating methodologies, the lessons learned during implementation of the system can be used to build design assisting tools that automate the whole or part of the design process.

The fact that the system has generated nearly 1000 projects and been running continuously for a long period proves the credibility of the proposed framework and implementation strategies. Besides, it shows that the system is quite robust and fault tolerant. These are very desirable features in design system especially if they are intended for real world as opposed to research purposes. The results from the implementation phase contributes to how to build such design systems.

The implementation of backtracking algorithms by introducing concepts such as backtracking session, backtracking agenda, etc. can be used in other systems other than design systems that need dependency backtracking mechanisms.

11.7.4 Contributions to Robot Design

The robot design area can benefit from the methodologies generated. Although the robot design problem that we chose in this dissertation is a simplified one, the complexity of the interaction between disciplines is preserved. Also, the results on how to break the design knowledge into pieces in each discipline involved might be used in real design situations for robot design.

11.8 Final Conclusion

The final conclusion is that the following hypothesis has been proved to be true:

> *Computers can provide us with better ways of doing design by discovering superior design methodologies that integrate different points-of-view of multiple disciplines in the design process.*

In this dissertation we showed that it is possible to use the computers to simulate the design process. We can then analyze the results of the simulation to synthesize design methodologies that have superior features. The approach that we have proposed has been developed based on parametric design problems. Applicability of the approach to other types of problems can be investigated.

11.9 Future Work

There are directions in which this research should be extended to increase the chance for the success of the proposed approach in real design practices. There are also ways in which this research can be extended to increase its scope. The following is a list of the areas we think this research should or can be extended to:

- Applying the approach to other types of design problems other than non-routine parametric design such as configuration design. This is a direction that increases the scope of the proposed approach. Obviously one important feature that should exist in the new types of design problems is that they should be able to be automated via a multi-agent system, even in a simplified version.

- Applying the approach to other multi-disciplinary domains such as automotive design, electronics packaging design, or building design. This would help to get better insight into the problem and enhance the generality of the approach.

- Discovering the rules for simplification of the process. We discussed that for being able to discover design methodologies based on simulation of the design process it is not necessary to model the design process in full detail. Similar to the simulation of physical phenomenon in engineering analysis we can simulate a simplified version of the design process and still get the design methodologies that are helpful for the real situations. The rules for these simplifications and approaches and tools should be developed for this purpose.

- Evaluation of the methodologies in real situations. The design methodologies developed by the approach proposed should be evaluated in the real design situations in which human designers use those methodologies and report on any enhancement in the design process. Also, the lesson we have learned in organizational side of the design process via agents can be implemented in design processes by human designers to see whether any improvements happen.

- The effect of scaling up the approach should be investigated. Larger design processes with more disciplines and design parameters must be considered for this approach to be implemented.
- Enrich the design methodologies. Design methodologies can contain more knowledge about not only the design approaches but also about control aspects of the design process too.
- Biased methodologies could be generated using this approach. Design methodologies that are biased toward manufacturability for example can be another area of further research.
- Change the order of approaches. It would be useful to investigate the effect of order of the design approaches in traces produced.
- Convert the tool to a sensitivity analysis tool or even an optimization tool. As we used the multi-agent design system in discovering sensitive parameters as well as their boundaries, more research can be done to convert the system to an optimization tool. Such an approach to optimization is especially attractive for domains that do not conform to a completely mathematical form allowing classical optimization tools to be used.
- Introduce new types of design approaches into the design process such as heuristic, statistical, and probabilistic approaches.

- Close the feedback loop around the system in which the methodologies generated by the system are fed back into it to further refine the methodology. That is, to force the system to use the methodologies generated for similar problems and measure the improvement.
- Investigate the effect of changing the resolution of the set of constraints and requirements. Using a concept similar to adaptive mesh generation in FEM (Finite Element Method) to find the best grid and/or find the areas of high gradient. Enhance the resolution of the constraint-requirement grid so that the system covers more design problems.
- Investigating the effect of the design knowledge used to produce correct designs on generating the right methodologies. That is, finding the answer to the following questions: What is the trade-off between the design quality, as generated by RD, and correct traces, hence correct methodologies? Can we remove some of the design knowledge from the system (knowing that it will produce inferior designs) but still obtain correct design methodologies? If the answer to the last question is positive, the knowledge engineering and implementation stages can be substantially simplified.

11.10 Extending to Other Domains

In this section we describe the research directions that examines the applicability of the proposed approach to problems other than discovering design methodologies.

Simulation of complex systems using agents can be a powerful tool in analyzing why those systems behave in a certain way. Especially, using simulation via multi-agent systems the complex systems can be designed to have some desired emergent behavior. Multi-disciplinary design processes are one example of complex systems whose emergent behavior is hard to anticipate. Other areas that are similar to the multi-disciplinary design process in having an emergent behavior are: 'supply chains in manufacturing enterprises' and 'shop floor job scheduling'.

This approach is directly adaptable to the problem of 'Supply Chain Optimization' in manufacturing enterprises. Suppliers and consumers can be modeled as agents that are negotiating with each other to sell and buy goods or services based on their needs as well as on their resource constraints. A multi-agent system that simulates this process can find the best supply chain for a company. The same approach can be applied to multi-enterprise supply chain management problems.

Shop Floor Job Scheduling problems in manufacturing industries is another area that can be attacked by this approach. Different machine tools and manufacturing activities will be modeled by different agents. A multi-agent system simulates the manufacturing process in the shop. The system generates different schedules for different jobs based on the requirements as well as the constraints on time, budget, and priority of the jobs. Generalizing the generated schedules guides how the future jobs should be scheduled.

Bibliography

Andeen 88 G. B. Andeen. *Robot Design Handbook*, McGraw-Hill Book Company, 1988.

Arciszewlski 87 T. Arciszewlski, M. Mustafa, and W. Ziarko. “A Methodology of Design Knowledge Acquisition for Use in Learning Expert Systems”, *International Journal of Man-Machine Studies*, 27, 1987, pp. 23-32.

Badhrinath 96 K. Badhrinath and J. R. Jagannatha Rao. “Modeling for Concurrent Design Using Game Theory Formulations”, *Concurrent Engineering: Research and Applications*, Vol. 4, Number 4, December 1996, pp. 389-399.

Breuker 94 J. Breuker and W. Van de Velde (Eds). *CommonKADS Library for Expertise Modeling*, IOS Press, 1994.

Brown 89 D. C. Brown and B. Chandrasekaran. *Design Problem Solving: Knowledge Structures and Control Strategies*, Research Notes in Artificial Intelligence Series, Morgan Kaufmann Publishers, Inc., 1989.

Brown 93 D. C. Brown & R. Douglas. “Concurrent Accumulation of knowledge: A View of CE”, *The Handbook of Concurrent Design and Manufacturing*, (Eds.) H. R. Parsaei & W. G. Sullivan, Chapman & Hall, 1993, pp. 402-412.

Brown 96-a D. C. Brown. “Modeling Conflicts Between Agents in a Design Context”, *ECAI'96 Workshop on Modeling Conflicts in AI*, European Conference on Artificial Intelligence, Budapest, Hungary.

Brown 96-b D. C. Brown, S. E. Lander and C. J. Petrie. "The Application of Multi-agent Systems to Concurrent Engineering", Editorial. *Concurrent Engineering: Research and Applications*, Special Issue on Multi-agent Systems in Concurrent Engineering, Technomic Publ., Vol. 4, No. 1, March 1996, pp.2-5.

Brown 96-c D. C. Brown. Routineness Revisited. *Mechanical Design: Theory and Methodology*, (Eds.) M. Waldron & K. Waldron, Springer-Verlag, 1996, pp. 195-208.

Brown 97 D. C. Brown and W. P. Birmingham. "Understanding the Nature of Design", *IEEE Expert: Intelligent Systems & their Applications*, March-April 1997, pp. 14-16.

Clearwater 92 S. H. Clearwater, B. A. Huberman, and T. Hogg. "Cooperative Problem Solving", in *Computation: The Micro and the Macro View*, B. Huberman (Ed.), World Scientific, 1992, pp. 33-70.

Coyne 90 R. D. Coyne, M. A. Rosenman, A. D. Radford, M. Balachandran, and J. S. Gero. *Knowledge-based Design Systems*, Addison-Wesley Publishing Company, 1990.

Craig 86 J. J. Craig. *Introduction to Robotics, Mechanics & Control*, Addison-Wesley Publishing Company, 1986.

Cross 89 N. Cross. *Engineering Design Methods*, John Wiley & Sons Ltd., 1989.

Dasgupta 89 S. Dasgupta. “The structure of Design Processes”, in *Advances in Computers*, Vol. 28, M. C. Yovits (Ed.), Academic Press Inc. 1989, pp. 1-67.

Depkovich 89 T. M. Depkovich and R. M. Stoughton. “A General Approach for Manipulator System Specification, Design, and Validation”, *IEEE International Conference on Robotics & Automation*, Vol. 3, 1989, pp. 1402-1407.

Dixon 87 J. R. Dixon. “On Research Methodology towards a Scientific Theory of Engineering Design”, *AI EDAM*, 1987, 1(3), pp. 145-157.

Dixon 95 J. R. Dixon. Knowledge-based Systems for Design, *Journal of Mechanical Design*, Vol. 117, June 1995, pp. 11-16.

Dowlatshahi 97 S. Dowlatshahi. “Design Optimization in Concurrent Engineering: A Team Approach”, *Concurrent Engineering: Research and Applications*, Volume 5, Number 2, June 1997, pp. 145-154.

Duffy 97 A. H. B. Duffy. “The ‘What’ and ‘How’ of Learning in Design”, *IEEE Expert: Intelligent Systems & Their Applications*, Vol. 12, No. 3, May-June 1997, pp. 71-76.

Durfee 89 E. H. Durfee, V. R. Lesser, and D. D. Corkill. “Trends in Cooperative Distributed Problem Solving”, *IEEE Transactions on Knowledge and Data Engineering*, 1989, 1(1).

Eppinger 90 S. D. Eppinger, D. E. Whitney, R. P. Smith, and D. A. “Gebala. Organizing the Tasks in Complex Design Projects”, *Proceedings of Design Theory and Methodology Conference*, DTM-90, Chicago, ASME 1990, pp. 39-46.

Finin 93 T. Finin, J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, D. McKay, J. McGuire, S. Shapiro, and C. Beck. "Specification of the KQML: Agent-Communication Language", June 1993, *http://www.csee.umbc.edu/kqml/kqmlspec.ps*

Fisher 91 D. H. Fisher, M. J. Pazzani, and P. Langley (Eds). *Concept Formation: Knowledge and Experience in Unsupervised Learning*, Morgan Kaufmann Publishers, Inc., 1991.

Forrester 69 J. W. Forrester. *Urban Dynamics*, MIT Press, 1969.

Franklin 96 S. Franklin and A. Graesser. "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents", *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, Springer-Verlag, 1996.

Gebala 91 D. G. Gebala & S. D. Eppinger. "Methods for Analyzing Design Procedures", *Proc. ASME Design Theory and Methodology Conference*: DTM'91, ASME DE-Vol. 31, 1991, pp. 227-233.

Haddadi 96 A. Haddadi. *Communication and Cooperation in Agent Systems, A Pragmatic Theory*, Lecture Notes in Artificial Intelligence, Springer Verlag, 1996.

Hale 96 M. A. Hale, J. I. Craig, F. Mistree and D. P. Schrage. "DREAMS and IMAGE: A Model and Computer Implementation for Concurrent, Life-Cycle Design of Complex Systems", *Concurrent Engineering: Research and Applications*, Vol. 4, No. 2, June 1996, pp. 171-186.

Hazelrigg 96 G. A. Hazelrigg. *Systems Engineering: An Approach to Information-Based Design*, Prentice-Hall, Inc., 1996.

Holzbock 86 W. G. Holzbock. *Robotic Technology, Principles and Practice*, Van Nostrand Reinhold Company Inc., 1986.

Huang 93 G. Q. Huang and J. A. Brandon. *Cooperating Expert Systems in Mechanical Systems*, John Wiley & Sons Inc., 1993.

Iglesias 96 C. A. Iglesias, M. Garijo, J. C. Gonzàlez, and J. R. Velasco. "A Methodological Proposal for Multiagent Systems Development extending CommonKADS", *Proceedings of Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop*: Shareable and reusable problem-solving methods, Banff, Alberta, Canada November 9-14, 1996, also: http://ksi.cpsc.ucalgary.ca:80/KAW/KAW96/iglesias/Iglesias.html.

Jackson 90 P. Jackson. *Introduction to Expert Systems*, Addison-Wesley, Inc., 1990.

Kannapan 92 S. M. Kannapan and K. M. Marshek. "Engineering Design Methodologies: A new Perspective", in *Intelligent Design and Manufacturing*, A. Kusiak (Ed.), 1992 John Wiley & Sons, Inc. pp. 3-38.

Kauffman 80 D. L. Kauffman Jr. *Systems One: An Introduction to Systems Thinking*, Future Systems, Inc. 1980.

Klein 91 M. Klein. "Supporting Conflict Resolution in Cooperative Design Systems", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No. 6, Nov./Dec. 1991, pp. 1379-1389.

Kroo 88 I. Kroo, M. Takai. "A Quasi-Procedural, Knowledge-Based System for Aircraft Design", *AIAA/AHS/ASEE Aircraft Design, Systems and Operations Meeting*, Atlanta, GA, AIAA-88-4428, September 1988.

Kroo 90 I. Kroo and M. Takai. "Aircraft Design Optimization Using a Quasi-Procedural Method and Expert System," *Multidisciplinary Design and Optimization Symposium*, Nov. 1990.

Kusiak 93 A. Kusiak. "Decomposition of the Design Process," *Journal of Mechanical Design*, December 1993, Vol. 115, pp. 687-695.

L'Hote 83 F. L'Hote, J. M. Kauffmann, P. Andre, and J. P. Taillard. *Robot Technology: Robot Components and Systems*, Printice-Hall, Inc., 1983.

Lander 92 S. E. Lander and V. R. Lesser; "Customizing Distributed Search Among Agents with Heterogeneous Knowledge," *Proceedings 5th International Symposium on AI Applications in Manufacturing and Robotics*, Cancun, Mex., Dec. 1992.

Lander 94 S. E. Lander. *Distributed Search and Conflict Management Among Reusable Heterogeneous Agents*, Ph.D. Dissertation, University of Massachusetts at Amherst, May 1994.

Lander 97 S. E. Lander. "Issues in Multi-agent Design Systems," *IEEE Expert: Intelligent Systems and their Applications*, March-April 1997, Vol. 12, No. 2, pp. 18-26.

Langley 96 P. Langley. *Elements of Machine Learning*, Morgan Kaufmann, Inc., 1996.

Levitt 91 R. E. Levitt, Y. Jin, and C. L. Dym. "Knowledge-Based Support for Management of Concurrent, Multidisciplinary Design", *AI EDAM*, Academic Press Ltd., Vol. 5, No. 2, 1991, pp. 77-95.

Liu 94 J. Liu and D. C. Brown. "Generating Design Decomposition Knowledge for Parametric Design Problems," *AID-94: Artifi-*

cial Intelligence in Design'94, (Eds.) J. S. Gero and F. Sudweeks, Kluwer Academic Publ., 1994, pp. 661-678.

MacCallum 89 K. MacCallum and Sue Green. "THESYS - Implementation of a Knowledge-Based Design System with Multiple Viewpoints," *Intelligent CAD Systems II: Implementation Issues*, V. Akman, P. J. W. ten Hagen, and P. J. Veerkamp (Eds.), Springer-Verlag, 1989, pp. 228-244.

Maher 95 M. L. Maher, M. B. Balachandran and D. M. Zhang. *Case-Based Reasoning in Design*, Lawrence Erlbaum Assoc., 1995.

Marcus 92 S. Marcus, J. Stout, and J. McDermott. "VT: An Expert Elevator Designer that Uses Knowledge-Based Backtracking", in *Artificial Intelligence in Engineering Design*, Academic Press, Inc., 1992, edited by: C. Tong and D. Sriram, Vol. I, pp. 317-355.

McLaughlin 87 S. McLaughlin and J. S. Gero. "Acquiring Expert Knowledge from Characterized Designs", *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing*, 1, 1987, pp. 73-87.

Mitchell 97 Tom M. Mitchell. *Machine Learning*, McGraw-Hill Companies, Inc., 1997.

Mittal 92 S. Mittal and A. Araya. "A Knowledge-based Framework for Design", in *Artificial Intelligence in Design*, (eds) C. Tong and R.D. Sriram, Academic Press, Inc., 1992, pp. 273-293.

Müller 96. H. J. Müller. "Multi-agent systems engineering", in *Second Knowledge Engineering Forum*, Karlsruhe 1996, page 213.

NSF 96 *Research Opportunities in Engineering Design*, NSF Strategic Planning Workshop Final Report, (ed.) J. Shah April 1996.

Oaks 97 S. Oaks and H. Wong. *Java Threads*, O'Reilly & Associates, Inc., 1997.

Ogata 97 K. Ogata. *Modern Control Engineering*, Prentice-Hall, Inc., 1997.

Pahl 88 G. Pahl and W Beitz. *Engineering Design: A Systematic Approach*, Springer-Verlag 1988.

Pena-Mora 95 F. Pena-Mora, R.D. Sriram, and R. Logcher. "Conflict Mitigation System for Collaborative Engineering", *AIEDAM* (1995), 9, pp. 101-124.

Press 89 W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes*, Cambridge University Press, 1989.

Quinlan 93 J.R. Quinlan. *Programs for Machine Learning*. Morgan Kaufmann Publishers, Inc., 1993.

Reich 91 Y. Reich and S.J. Fenves. "The Formation and Use of Abstract Concepts in Design", in *Concept Formation: Knowledge and Experience in Unsupervised Learning*, Edited by D. H. Fisher, M. J. Pazzani, and P. Langley. Morgan Kaufmann Publishers, Inc., 1991, pp. 323-353.

Rich 91 E. Rich and K. Knight. *Artificial Intelligence*, McGraw-Hill, Inc., 1991, second edition.

Rivin 88 E. I. Rivin. *Mechanical Design of Robots*, McGraw-Hill Book Company, 1988.

Rogers 96 J. L. Rogers, C. M. McCulley & C. L. Bloebaum. "Integrating a Genetic Algorithm Into a Knowledge-Based System for Ordering Complex Design Processes", *Fourth Int. Conf. on AI in Design '96*, Stanford, CA, June 24-27, 1996.

Rudolph 96 E. G. Rudolph, J. Grabowski, and P. Graubmann. "Tutorial on message sequence charts" (MSC), *Proceedings of FORTE/ PSTV'96 Conference*, also: *ftp://ftp.win.tue.nl/pub/techreports/ sjouke/msc/msc96tutorial.ps.Z.*

Senge 94 P. Senge, A. Kleiner, C. Roberts, R. Ross, and B. Smith. *The Fifth Discipline Fieldbook: Strategies for Building a Learning Organization*, Currency Publishing, 1994.

Serrano 92 Serrano, D., and Gossard, D., 1992. "Tools and Techniques for Conceptual Design", in *Artificial Intelligence in Design*, edited by C. Tong and D. Sriram, Vol I, Academic Press, 1992, page 71-116.

Shoham 93 Y. Shoham. "Agent-oriented Programming" *Artificial Intelligence*, Vol. 60, 1993, pp. 51-92.

Sieger 95 D. B. Sieger and A. B. Badiru. "A Performance Parameter Model for Design Integration," *Engineering Design and Automation*, Volume 1, Number 3, Fall 1995, pp. 137-148.

Smith 80 R. G. Smith. "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver", *IEEE Transaction on Computers*, C-29(12), 1980, pp. 1104-1113.

Sobieszczanski-Sobieski 96 J. Sobieszczanski-Sobieski and R. T. Haftka, "Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments", *34th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, AIAA Paper No. 96-0711, January 15-18, 1996, pp. 32.

Sobolewski 96 M. Sobolewski. “Multiagent Knowledge-Based Environment for Concurrent Engineering Applications”, *Concurrent Engineering: Research and Applications*, Vol. 4, No. 1, March 1996, pp. 89-97.

Sriram 98 R.D. Sriram. “Information Technology in Engineering Design”, Invited editorial, *ASCE Journal of Computing in Civil Engineering*, July 1998, vol. 12, issue 3, pp. 123-125,

Suh 95 N.P. Suh. “Axiomatic Design of Mechanical Systems”, *Journal of Mechanical Design*, Vol. 117, June 1995, pp. 2-10.

Sycara 90 K. Sycara. “Cooperative Negotiation in Concurrent Engineering Design”, *Cooperative Engineering Design*, Springer Verlag Publications, 1990.

Torsun 95 I. S. Torsun. *Foundations of Intelligent Knowledge-based Systems*, Academic Press, August 1995.

Tsai 81 Y. C. Tsai and A. H. Soni. “Accessible Region and Synthesis of Robot Arms”, *Journal of Mechanical Design*, October 1981, Vol. 103, pp. 803-811.

Tsai 89 M. J. Tsai, T. G. Wu, and L. Hong. “Expert System for Design of Industrial Robots - ESDIR”, *Advances in Design Automation*, Vol. 1, ASME 1989, pp. 315-324.

Wellman 95 M.P. Wellman. “A Computational Market Model for Distributed Configuration Design”, *AIEDAM* (1995), 9, pp. 125-133.

Wooldridge 95 M. Wooldridge and N.R. Jennings. “Intelligent agents: Theory and practice”, *The Knowledge Engineering Review*, 10(2):115-152, 1995.

Wooldridge 97 M. Wooldridge. “Agent-based software engineering”, *IEEE Transactions on Software Engineering*, 144(1): 26-37, 1997.

Wooldridge 98 M. Wooldridge and N. R. Jennings. “Pitfalls of Agent-Oriented Development”, in *Proceedings of the Second International Conference on Autonomous Agents*, 385-390. Minneapolis/St. Paul, MN. USA, May 9-13, 1998.

Woyak 95 S. Woyak, B. Malone and A. Myklebust, “An Architecture for Creating Engineering Applications: The Dynamic Integration System”, *Proc. ASME Computers in Engineering Conf.*, Boston, MA, September 1995.

Wujek 96 B. A. Wujek, J. E. Renaud, S. M. Batill, and J. B. Brockman. “Concurrent Subspace Optimization Using Design Variable Sharing in a Distributed Computing Environment,” *Concurrent Engineering: Research and Applications*, Volume 4, Number 4, December 1996, pp. 361-377.

Appendix A. Extention of the Kinematics Equations

A.1 Modification of Equations

In this appendix we modify the kinematic equations in order to make them applicable to points in all the four quadrants.

Calculation of the joint angles is based on the *acos* function. There are always two angles in the range of zero to 2π that have the same cosine value. As a result, in calculating the arc cosine of a number (between -1 and 1) we should have extra information to know which one of the two angles has produced that cosine value. In computer programming languages (e.g., Java) the *acos* function returns the arc cosine of an angle in the range of zero to π. Therefore, the problem arises when the angle is in fact between π and 2π and the program returns a value between zero and π.

To solve this ambiguity we find the quadrant to which the angle belongs and adjust the value produced by acos function if necessary. Referring to Figure A-2-3 we find the quadrant of α_1 angle by checking the location of (x_i, y_i) point relative to (x_b, y_b). Knowing that the α_2 angle will always be within the zero to π range (because it belongs to a triangle) and whether the first solution (i.e., $\alpha_1 - \alpha_2$) or the second solution (i.e., $\alpha_1 + \alpha_2$) is desired, we can adjust θ_{1i} accordingly so that its value is always between $-\pi$ and π.

The absolute value of θ_{2i} angle is always between zero and π (because it is an external angle of a triangle). It is positive for the first solution (i.e., $\alpha_1 - \alpha_2$) and negative for the second solution (i.e., $\alpha_1 + \alpha_2$).

Figure A-1 shows how the value of θ_{1i} may be adjusted after determining the quadrant to which α_1 belongs.

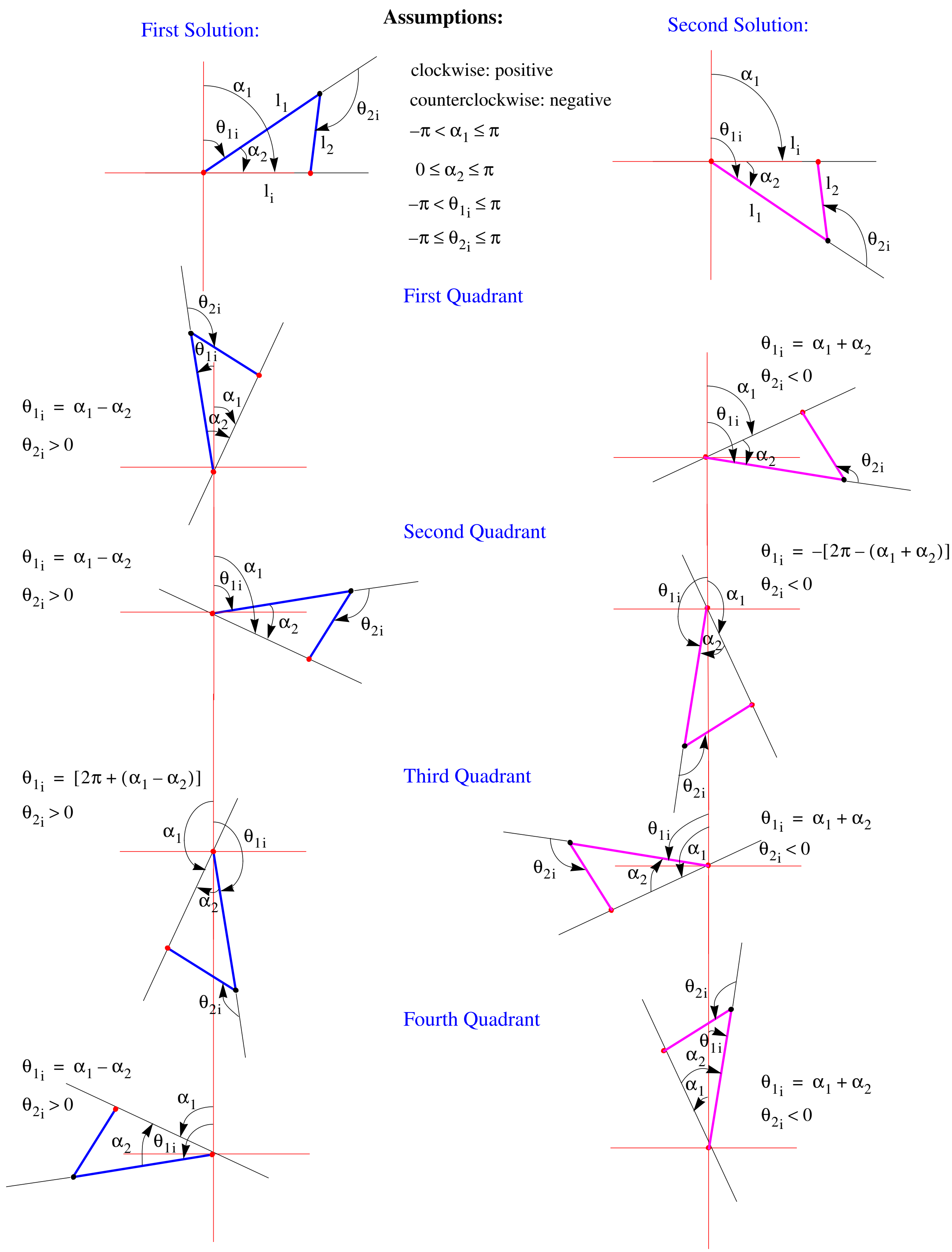


Figure A-1. Adjustment of θ_{1i} Angle.

A.1.1 Calculation of Accessible Region Area

The accessible region of a 2-DOF planar robot is a function of link lengths, the angle swept by the first link (θ_{1i} angles), and the maximum and minimum of $\cos\theta_{2i}$ angles:

$$A = \frac{1}{2}\theta_{1,\,sweep}(r_1^2 - r_2^2)$$

$$A = \frac{1}{2}\theta_{1,\,sweep}[l_1^2 + l_2^2 + 2l_1l_2(\cos\theta_{2i})_{max}] - [l_1^2 + l_2^2 + 2l_1l_2(\cos\theta_{2i})_{min}]$$

$$A = l_1l_2\theta_{1,\,sweep}[(\cos\theta_{2i})_{max} - (\cos\theta_{2i})_{min}]$$

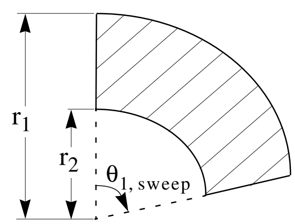


Figure A-2. Calculating the Accessible Region of A 2-DOF Robot.

While the above equations are general (i.e., they are applicable to any configuration that the robot may take), the equations that are derived in [Tsai 81] are only valid if the points in the workspace are located in the first and/or second quadrants. In [Tsai 81], $\theta_{1,sweep}$ is found by calculating $\theta_{1,max}$ - $\theta_{1,min}$. As it is shown in Figure A-3 $\theta_{1,max}$ and $\theta_{1,min}$ define two totally different accessible areas for the robot, while $\theta_{1,max}$ - $\theta_{1,min}$ produces the same sweep angle for both of those areas. Therefore, using the equation: $\theta_{1,sweep} = \theta_{1,max} - \theta_{1,min}$ may produce the wrong answer for the area of accessible region of the robot.

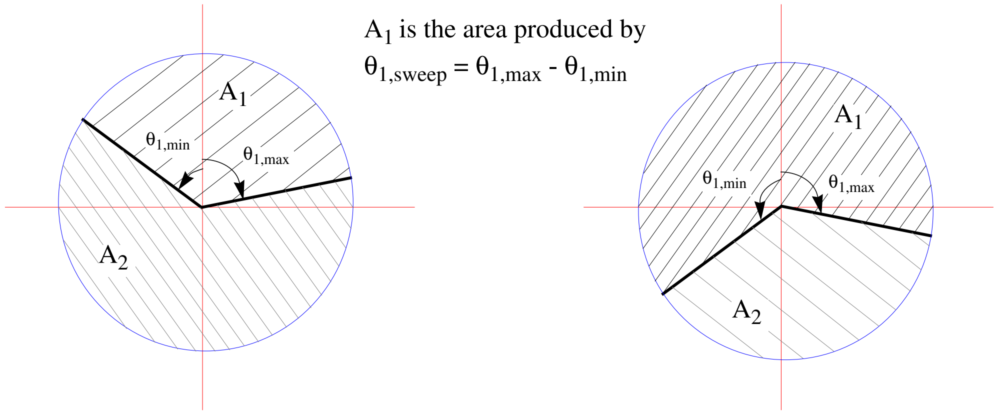


Figure A-3. Different Covered Areas for the Same $\theta_{1,max}$ - $\theta_{1,min}$.

To find the correct sweep angle for the first link of the robot we have to take the θ_{1i} angles that participate in creating the workspace. That is, we have to find the sum of the sweep angle for each quadrant that has been covered in the workspace and then add the result to the sum of gap angles between participating quadrants. However, the fact that there is a discontinuity in measuring angles when they pass from the second quadrant to the third quadrant, introduces an exception into the calculation of the sweep angle. Moreover, in calculating the gap angles between quadrants, there can be two answers for the sweep angle. These two issues are illustrated in Figure A-4:

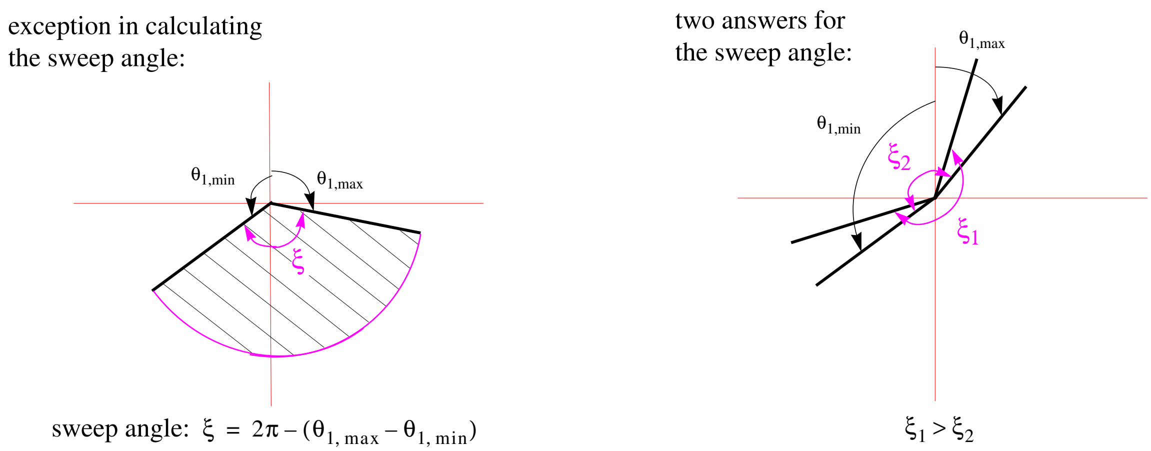


Figure A-4. Calculation of Sweep Angle.

As a result of the above issues we need to adjust the algorithm for calculating the sweep angle. To solve the first problem we first find in which quadrants the angles are distributed. We calculate the sweep angle for each quadrant separately and then add the gap angle between each pair of angles in two adjacent quadrants (or two non-adjacent quadrants if the two participating quadrants are not neighbors). Fifteen different situations may happen as it is shown in Figure A-5. As a convention we assume:

- angles ending on positive part of y axis belong to the First Quadrant
- angles ending on positive part of x axis belong to the Second Quadrant
- angles ending on negative part of y axis belong to the Third Quadrant
- angles ending on negative part of x axis belong to the Fourth Quadrant

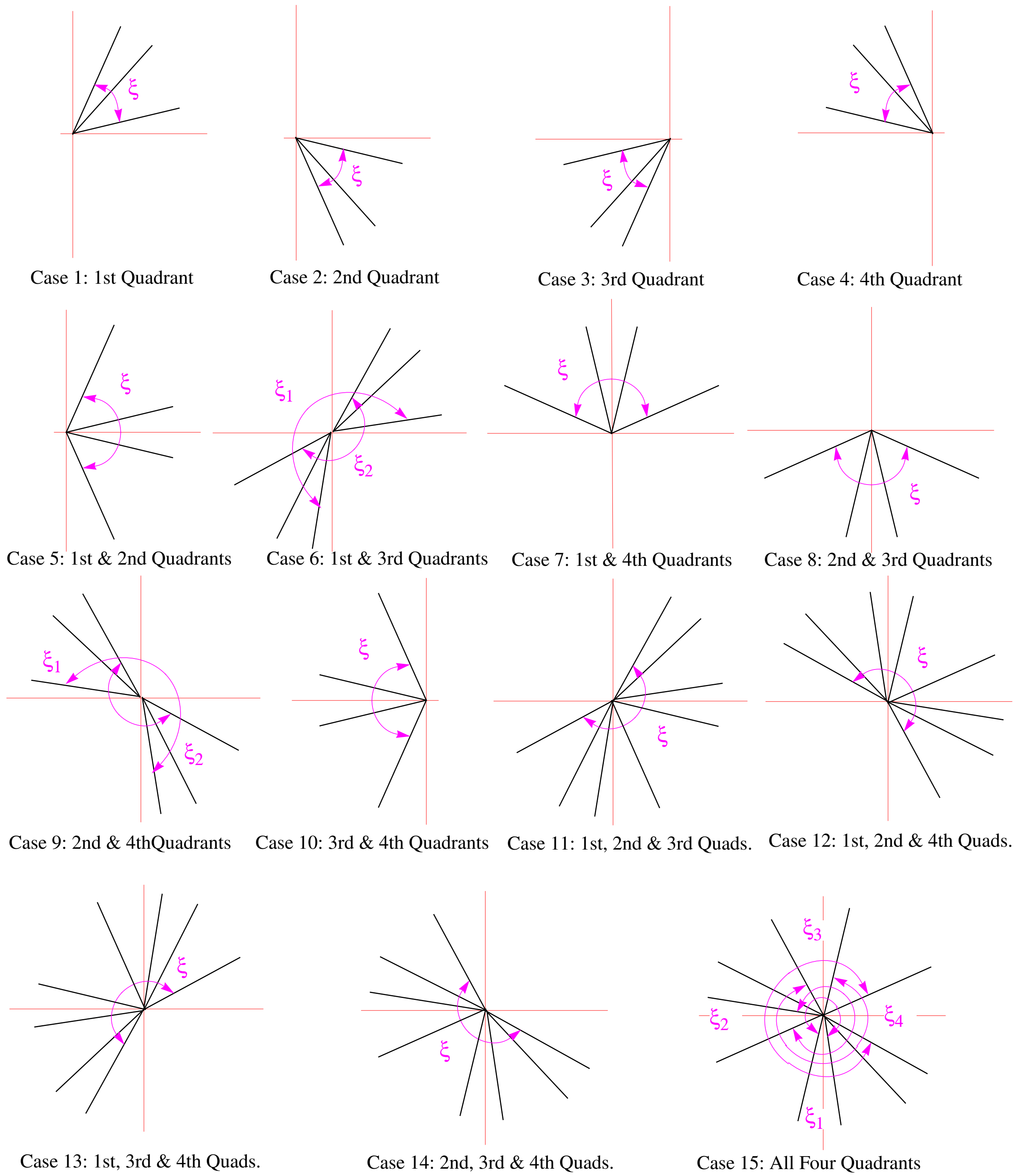


Figure A-5. Different Cases for Calculating Sweep Angle.

In cases 6, 9, and 15, in which there is more than one solution for the sweep angle, we pick the smallest angle assuming that the robot's path planner chooses the shortest path. As it was explained above, to find the sweep angle we first calculate the sum of the sweep angles for each participating quadrants. We calculate the sweep angle for each quadrant by subtracting the minimum angle from the maximum angle in that quadrant. Note that this algorithm produces the right sweep angle (i.e., positive) even for the third and the fourth quadrants with negative angles. Next we calculate the gap angle between the participating quadrants and add their sum to the sum of sweep angles in each quadrant. To calculate the gap angles six different situations may happen as it is shown in Figure A-6:

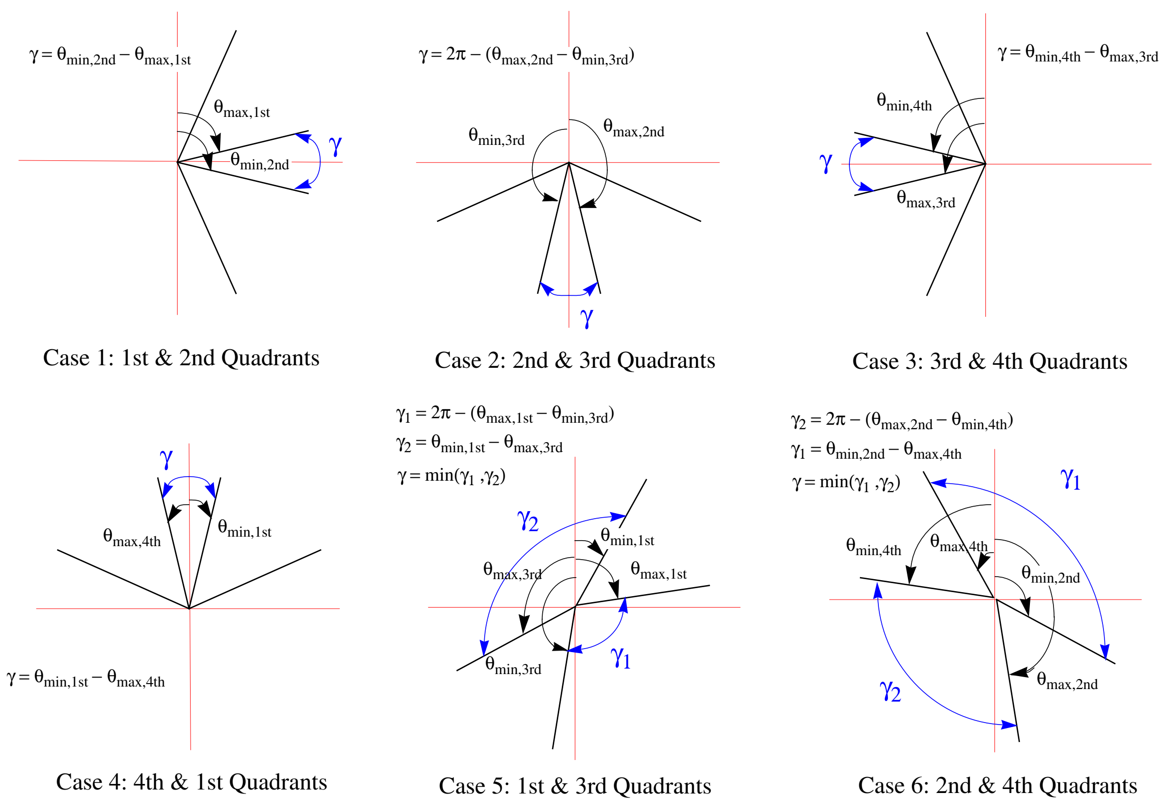


Figure A-6. Different Cases for Calculating Gap Angle.

In [Tsai 81] the last term in the equation for calculating area is expressed as $(\cos\theta_{2,\,min} - \cos\theta_{2,\,max})$ based on the assumption that θ_{2i} angles are always within the zero and π range. This may produce a negative answer when θ_{2i} angles vary between zero and $-\pi$ (for the second solution). For instance, if $\theta_{2,min}$ = -180 and $\theta_{2,max}$ = -60 degrees, we will have $[\cos(-180) - \cos(-60)] = -1 - 0.5 = -1.5$. To solve this problem we first calculate the cosine of θ_{2i} angles and then find the maximum and minimum values among them.

Also, in equation (13) of [Tsai 81] there is a typographical error that causes the area to be calculated larger than its real value (as long as l_2 is less than l_1). This error was discovered when we compared the results for the area of the accessible region to the measured value based on the geometry of the robot (this typo can be guessed by comparing equations 5 and 13 in the paper). The following is the derivation of the equations in order to correct the mistake.

Equations (13) from Tsai's paper: $A' = F'(\theta_{1,\,max} - \theta_{1,\,min})(l_1 + l_2)^2$ $F' = \dfrac{\frac{l_2}{l_1}(\cos\theta_{2,\,min} - \cos\theta_{2,\,max})}{1 + \left(\frac{l_2}{l_1}\right)^2}$

the mistake

Adjusted and corrected equations: $A' = F'\theta_{1,\,sweep}(l_1 + l_2)^2$ $F' = \dfrac{\frac{l_2}{l_1}((\cos\theta_{2i})_{max} - (\cos\theta_{2i})_{min})}{\left(1 + \frac{l_2}{l_1}\right)^2}$

Proving the adjusted equations:

$$F' = \frac{\frac{l_2}{l_1}((\cos\theta_{2i})_{max} - (\cos\theta_{2i})_{min})}{\left(1 + \frac{l_2}{l_1}\right)^2} = \frac{l_1^2\left\{\frac{l_2}{l_1}((\cos\theta_{2i})_{max} - (\cos\theta_{2i})_{min})\right\}}{l_1^2\left\{\left(1 + \frac{l_2}{l_1}\right)^2\right\}} = \frac{l_1 l_2((\cos\theta_{2i})_{max} - (\cos\theta_{2i})_{min})}{(l_1 + l_2)^2}$$

$$A' = F'\theta_{1,\,sweep}(l_1 + l_2)^2 = \frac{l_1 l_2((\cos\theta_{2i})_{max} - (\cos\theta_{2i})_{min})}{(l_1 + l_2)^2}\theta_{1,\,sweep}(l_1 + l_2)^2$$

$A' = l_1 l_2 \theta_{1,\,sweep}((\cos\theta_{2i})_{max} - (\cos\theta_{2i})_{min})$ ✓ as was derived above

Appendix B. Clusters of Traces

This appendix contains the cluster tree of traces. The attributes of each cluster node in the tree is represented as a row in Table B-1. A brief explanation of each attribute is as follows:

1. Cluster: the name of the cluster following the naming convention in "Naming Convention for the Clusters" on page 222;
2. Number of Traces: the number of the traces accumulated in this cluster;
3. Number of Projects: the number of projects that followed the traces accumulated in this cluster;
4. Traces: the indices of the traces accumulated in this cluster;
5. Projects: the indices of the projects that followed the traces accumulated in this cluster;
6. DK1 to DS1: the index of the design approach used by Designer_K_1 to Designer_S_2, respectively;
7. Children: the names of the clusters that are the children of this node in the cluster tree, i.e., the clusters that were re-clustered to build this cluster;
8. Parent: the name of the parent node of this cluster in the cluster tree.

The group of sibling clusters that have the same depth in the cluster tree are separated from each other by a blank and shaded row in Table B-1.

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
0_0	1	52	0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728	0	0	0	0	0	0	0	none	1_0
0_1	1	72	1	13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740	0	0	0	0	0	0	1	none	1_0
0_2	1	64	2	25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376	0	0	0	0	0	0	2	none	1_0
0_3	1	1	1926	45	5	0	0	0	0	1	2	none	1_1
0_4	1	20	770	61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304	2	0	0	0	0	0	2	none	1_0
0_5	1	13	769	62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560	2	0	0	0	0	0	1	none	1_0
0_6	1	11	818	65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328	2	0	1	0	0	0	2	none	1_2
0_7	1	28	1537	69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804	4	0	0	0	0	0	1	none	1_0
0_8	1	2	1105	70, 219	2	1	3	0	0	0	1	none	1_3
0_9	1	2	970	77, 315	2	1	0	0	1	0	2	none	1_4
0_10	1	3	866	83, 108, 109	2	0	2	0	0	0	2	none	1_2
0_11	1	15	1538	89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217	4	0	0	0	0	0	2	none	1_0
0_12	1	3	1010	95, 114, 352	2	1	1	0	0	0	2	none	1_2
0_13	1	2	914	102, 340	2	0	3	0	0	0	2	none	1_2
0_14	1	1	817	110	2	0	1	0	0	0	1	none	1_2
0_15	1	6	1978	117, 221, 227, 337, 440, 459	5	0	1	0	1	0	2	none	1_5
0_16	1	4	962	120, 208, 220, 232	2	1	0	0	0	0	2	none	1_4
0_17	1	6	194	130, 155, 174, 367, 380, 412	0	1	0	0	0	0	2	none	1_4
0_18	1	5	98	137, 162, 290, 375, 400	0	0	2	0	0	0	2	none	1_6
0_19	1	5	50	143, 168, 169, 258, 368	0	0	1	0	0	0	2	none	1_2
0_20	1	2	254	149, 387	0	1	1	0	1	1	2	none	1_7
0_21	1	2	10	150, 388	0	0	0	0	1	0	2	none	1_8
0_22	1	3	2118	161, 392, 399	5	1	0	0	0	1	2	none	1_1
0_23	1	3	2022	167, 263, 362	5	0	2	0	0	1	2	none	1_1
0_24	1	1	2186	179	5	1	1	1	0	0	2	none	1_9
0_25	1	2	146	180, 265	0	0	3	0	0	0	2	none	1_6
0_26	1	5	1536	181, 182, 192, 194, 199	4	0	0	0	0	0	0	none	1_0
0_27	1	3	1009	183, 188, 195	2	1	1	0	0	0	1	none	1_3

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
0_28	1	1	1681	184	4	0	3	0	0	0	1	none	1_10
0_29	1	1	1065	187	2	1	2	0	1	0	1	none	1_11
0_30	1	3	1586	190, 209, 215	4	0	1	0	0	0	2	none	1_2
0_31	1	1	961	196	2	1	0	0	0	0	1	none	1_3
0_32	1	2	1057	200, 207	2	1	2	0	0	0	1	none	1_3
0_33	1	1	1137	212	2	1	3	1	1	0	1	none	1_12
0_34	1	24	1546	222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460	4	0	0	0	1	0	2	none	1_8
0_35	1	54	1545	223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916	4	0	0	0	1	0	1	none	1_8
0_36	1	5	1986	234, 317, 323, 330, 472	5	0	1	0	2	0	2	none	1_5
0_37	1	1	2018	270	5	0	2	0	0	0	2	none	1_6
0_38	1	1	1922	276	5	0	0	0	0	0	2	none	1_0
0_39	1	1	1974	277	5	0	1	0	0	1	2	none	1_1
0_40	1	1	1113	307	2	1	3	0	1	0	1	none	1_3
0_41	1	1	874	308	2	0	2	0	1	0	2	none	1_13
0_42	1	1	994	320	2	1	0	1	1	0	2	none	1_14
0_43	1	1	1018	327	2	1	1	0	1	0	2	none	1_15
0_44	1	1	1594	427	4	0	1	0	1	0	2	none	1_5
0_45	1	1	1554	448	4	0	0	0	2	0	2	none	1_8
0_46	1	33	49	505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764	0	0	1	0	0	0	1	none	1_16
0_47	1	1	205	513	0	1	0	0	1	1	1	none	1_17
0_48	1	1	2021	525	5	0	2	0	0	1	1	none	1_18
0_49	1	14	97	529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776	0	0	2	0	0	0	1	none	1_6
0_50	1	1	153	538	0	0	3	0	1	0	1	none	1_19
0_51	1	12	960	541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784	2	1	0	0	0	0	0	none	1_4
0_52	1	1	1761	553	4	1	0	1	1	0	1	none	1_20
0_53	1	10	816	554, 567, 568, 572, 579, 580, 584, 591, 592, 596	2	0	1	0	0	0	0	none	1_2
0_54	1	3	984	558, 564, 796	2	1	0	1	0	0	0	none	1_21
0_55	1	2	1040	565, 576	2	1	1	1	1	0	0	none	1_22
0_56	1	2	864	566, 571	2	0	2	0	0	0	0	none	1_23
0_57	1	14	1977	569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5	0	1	0	1	0	1	none	1_5
0_58	1	1	1032	577	2	1	1	1	0	0	0	none	1_21
0_59	1	1	1785	578	4	1	1	0	1	0	1	none	1_24
0_60	1	1	872	583	2	0	2	0	1	0	0	none	1_13

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
0_61	1	1	2097	590	5	0	3	1	1	0	1	none	1_25
0_62	1	1	825	595	2	0	1	0	1	0	1	none	1_26
0_63	1	1	48	601	0	0	1	0	0	0	0	none	1_16
0_64	1	12	192	605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856	0	1	0	0	0	0	0	none	1_27
0_65	1	4	2116	617, 623, 848, 855	5	1	0	0	0	1	0	none	1_28
0_66	1	2	2220	629, 867	5	1	2	0	1	1	0	none	1_29
0_67	1	4	240	630, 636, 637, 868	0	1	1	0	0	0	0	none	1_16
0_68	1	1	2164	635	5	1	1	0	0	1	0	none	1_28
0_69	1	1	9	638	0	0	0	0	1	0	1	none	1_19
0_70	1	2	2125	642, 880	5	1	0	0	1	1	1	none	1_17
0_71	1	1	304	648	0	1	2	0	2	0	0	none	1_30
0_72	1	2	209	649, 725	0	1	0	0	2	0	1	none	1_31
0_73	1	1	57	650	0	0	1	0	1	0	1	none	1_19
0_74	1	3	2117	654, 660, 892	5	1	0	0	0	1	1	none	1_17
0_75	1	6	1593	697, 710, 711, 712, 715, 716	4	0	1	0	1	0	1	none	1_19
0_76	1	2	193	738, 744	0	1	0	0	0	0	1	none	1_27
0_77	1	1	1969	745	5	0	1	0	0	0	1	none	1_32
0_78	1	1	2141	750	5	1	0	1	0	1	1	none	1_33
0_79	1	1	2268	756	5	1	3	0	1	1	0	none	1_29
0_80	1	1	2173	757	5	1	1	0	1	1	1	none	1_17
0_81	1	1	249	770	0	1	1	0	1	0	1	none	1_19
0_82	1	6	1585	805, 806, 807, 808, 811, 812	4	0	1	0	0	0	1	none	1_10
0_83	1	1	2192	860	5	1	1	1	1	0	0	none	1_22
1_0	9	270	1, 0, 2, 770, 769, 1537, 1538, 1536, 1922	13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276	0, 2, 4, 5	0	0	0	0	0	1, 0, 2	0_1, 0_0, 0_2, 0_4, 0_5, 0_7, 0_11, 0_26, 0_38	2_1

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
1_1	4	8	1926, 2118, 2022, 1974	45, 161, 392, 399, 167, 263, 362, 277	5	0,1	0,2,1	0	0	1	2	0_3, 0_22, 0_23, 0_39	2_2
1_2	8	38	818, 866, 1010, 914, 817, 50, 1586, 816	65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596	2,0,4	0,1	1,2,3	0	0	0	2,1,0	0_6, 0_10, 0_12, 0_13, 0_14, 0_19, 0_30, 0_53	2_0
1_3	5	9	1105, 1009, 961, 1057, 1113	70, 219, 183, 188, 195, 196, 200, 207, 307	2	1	3,1,0,2	0	0,1	0	1	0_8, 0_27, 0_31, 0_32, 0_40	2_0
1_4	4	24	970, 962, 194, 960	77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784	2,0	1	0	0	1,0	0	2,0	0_9, 0_16, 0_17, 0_51	2_3
1_5	4	26	1978, 1986, 1594, 1977	117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5,4	0	1	0	1,2	0	2,1	0_15, 0_36, 0_44, 0_57	2_4
1_6	4	22	98, 146, 2018, 97	137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776	0,5	0	2,3	0	0	0	2,1	0_18, 0_25, 0_37, 0_49	2_5
1_7	1	2	254	149, 387	0	1	1	0	1	1	2	0_20	2_6
1_8	4	81	10, 1546, 1545, 1554	150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448	0,4	0	0	0	1,2	0	2,1	0_21, 0_34, 0_35, 0_45	2_7
1_9	1	1	2186	179	5	1	1	1	0	0	2	0_24	2_8
1_10	2	7	1681, 1585	184, 805, 806, 807, 808, 811, 812	4	0	3,1	0	0	0	1	0_28, 0_82	2_9
1_11	1	1	1065	187	2	1	2	0	1	0	1	0_29	2_10
1_12	1	1	1137	212	2	1	3	1	1	0	1	0_33	2_11
1_13	2	2	874, 872	308, 583	2	0	2	0	1	0	2,0	0_41, 0_60	2_12
1_14	1	1	994	320	2	1	0	1	1	0	2	0_42	2_13
1_15	1	1	1018	327	2	1	1	0	1	0	2	0_43	2_14
1_16	3	38	49, 48, 240	505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868	0	0,1	1	0	0	0	1,0	0_46, 0_63, 0_67	2_15

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
1_17	4	7	205, 2125, 2117, 2173	513, 642, 880, 654, 660, 892, 757	0,5	1	0,1	0	1,0	1	1	0_47, 0_70, 0_74, 0_80	2_16
1_18	1	1	2021	525	5	0	2	0	0	1	1	0_48	2_17
1_19	5	10	153, 9, 57, 1593, 249	538, 638, 650, 697, 710, 711, 712, 715, 716, 770	0,4	0,1	3,0,1	0	1	0	1	0_50, 0_69, 0_73, 0_75, 0_81	2_7
1_20	1	1	1761	553	4	1	0	1	1	0	1	0_52	2_18
1_21	2	4	984, 1032	558, 564, 796, 577	2	1	0,1	1	0	0	0	0_54, 0_58	2_19
1_22	2	3	1040, 2192	565, 576, 860	2,5	1	1	1	1	0	0	0_55, 0_83	2_20
1_23	1	2	864	566, 571	2	0	2	0	0	0	0	0_56	2_21
1_24	1	1	1785	578	4	1	1	0	1	0	1	0_59	2_22
1_25	1	1	2097	590	5	0	3	1	1	0	1	0_61	2_23
1_26	1	1	825	595	2	0	1	0	1	0	1	0_62	2_24
1_27	2	14	192, 193	605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744	0	1	0	0	0	0	0,1	0_64, 0_76	2_25
1_28	2	5	2116, 2164	617, 623, 848, 855, 635	5	1	0,1	0	0	1	0	0_65, 0_68	2_26
1_29	2	3	2220, 2268	629, 867, 756	5	1	2,3	0	1	1	0	0_66, 0_79	2_27
1_30	1	1	304	648	0	1	2	0	2	0	0	0_71	2_28
1_31	1	2	209	649, 725	0	1	0	0	2	0	1	0_72	2_29
1_32	1	1	1969	745	5	0	1	0	0	0	1	0_77	2_30
1_33	1	1	2141	750	5	1	0	1	0	1	1	0_78	2_31
2_0	13	47	1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816	70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596	2,0,4	1,0	3,1,0,2	0	0,1	0	1,2,0	1_3, 1_2	3_0

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
2_1	9	270	1, 0, 2, 770, 769, 1537, 1538, 1536, 1922	13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276	0,2,4,5	0	0	0	0	0	1,0,2	1_0	3_0
2_2	4	8	1926, 2118, 2022, 1974	45, 161, 392, 399, 167, 263, 362, 277	5	0,1	0,2,1	0	0	1	2	1_1	3_1
2_3	4	24	970, 962, 194, 960	77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784	2,0	1	0	0	1,0	0	2,0	1_4	3_2
2_4	4	26	1978, 1986, 1594, 1977	117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5,4	0	1	0	1,2	0	2,1	1_5	3_3
2_5	4	22	98, 146, 2018, 97	137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776	0,5	0	2,3	0	0	0	2,1	1_6	3_4
2_6	1	2	254	149, 387	0	1	1	0	1	1	2	1_7	3_5
2_7	9	91	10, 1546, 1545, 1554, 153, 9, 57, 1593, 249	150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770	0,4	0,1	0,3,1	0	1,2	0	2,1	1_8, 1_19	3_6
2_8	1	1	2186	179	5	1	1	1	0	0	2	1_9	3_7
2_9	2	7	1681, 1585	184, 805, 806, 807, 808, 811, 812	4	0	3,1	0	0	0	1	1_10	3_8
2_10	1	1	1065	187	2	1	2	0	1	0	1	1_11	3_9
2_11	1	1	1137	212	2	1	3	1	1	0	1	1_12	3_10
2_12	2	2	874, 872	308, 583	2	0	2	0	1	0	2,0	1_13	3_11
2_13	1	1	994	320	2	1	0	1	1	0	2	1_14	3_12
2_14	1	1	1018	327	2	1	1	0	1	0	2	1_15	3_13

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
2_15	3	38	49, 48, 240	505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868	0	0,1	1	0	0	0	1,0	1_16	3_14
2_16	4	7	205, 2125, 2117, 2173	513, 642, 880, 654, 660, 892, 757	0,5	1	0,1	0	1,0	1	1	1_17	3_15
2_17	1	1	2021	525	5	0	2	0	0	1	1	1_18	3_16
2_18	1	1	1761	553	4	1	0	1	1	0	1	1_20	3_17
2_19	2	4	984, 1032	558, 564, 796, 577	2	1	0,1	1	0	0	0	1_21	3_18
2_20	2	3	1040, 2192	565, 576, 860	2,5	1	1	1	1	0	0	1_22	3_19
2_21	1	2	864	566, 571	2	0	2	0	0	0	0	1_23	3_20
2_22	1	1	1785	578	4	1	1	0	1	0	1	1_24	3_6
2_23	1	1	2097	590	5	0	3	1	1	0	1	1_25	3_21
2_24	1	1	825	595	2	0	1	0	1	0	1	1_26	3_22
2_25	2	14	192, 193	605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744	0	1	0	0	0	0	0,1	1_27	3_23
2_26	2	5	2116, 2164	617, 623, 848, 855, 635	5	1	0,1	0	0	1	0	1_28	3_24
2_27	2	3	2220, 2268	629, 867, 756	5	1	2,3	0	1	1	0	1_29	3_25
2_28	1	1	304	648	0	1	2	0	2	0	0	1_30	3_26
2_29	1	2	209	649, 725	0	1	0	0	2	0	1	1_31	3_27
2_30	1	1	1969	745	5	0	1	0	0	0	1	1_32	3_28
2_31	1	1	2141	750	5	1	0	1	0	1	1	1_33	3_29

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
3_0	22	317	1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816	13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596	0,2,4,5	0,1	0,3,1,2	0	0,1	0	1,0,2	2_1, 2_0	4_0
3_1	4	8	1926, 2118, 2022, 1974	45, 161, 392, 399, 167, 263, 362, 277	5	0,1	0,2,1	0	0	1	2	2_2	4_1
3_2	4	24	970, 962, 194, 960	77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784	2,0	1	0	0	1,0	0	2,0	2_3	4_0
3_3	4	26	1978, 1986, 1594, 1977	117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5,4	0	1	0	1,2	0	2,1	2_4	4_2
3_4	4	22	98, 146, 2018, 97	137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776	0,5	0	2,3	0	0	0	2,1	2_5	4_3
3_5	1	2	254	149, 387	0	1	1	0	1	1	2	2_6	4_4
3_6	10	92	10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785	150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578	0,4	0,1	0,3,1	0	1,2	0	2,1	2_7, 2_22	4_5
3_7	1	1	2186	179	5	1	1	1	0	0	2	2_8	4_6
3_8	2	7	1681, 1585	184, 805, 806, 807, 808, 811, 812	4	0	3,1	0	0	0	1	2_9	4_7
3_9	1	1	1065	187	2	1	2	0	1	0	1	2_10	4_8
3_10	1	1	1137	212	2	1	3	1	1	0	1	2_11	4_9
3_11	2	2	874, 872	308, 583	2	0	2	0	1	0	2,0	2_12	4_10

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
3_12	1	1	994	320	2	1	0	1	1	0	2	2_13	4_11
3_13	1	1	1018	327	2	1	1	0	1	0	2	2_14	4_12
3_14	3	38	49, 48, 240	505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868	0	0,1	1	0	0	0	1,0	2_15	4_13
3_15	4	7	205, 2125, 2117, 2173	513, 642, 880, 654, 660, 892, 757	0,5	1	0,1	0	1,0	1	1	2_16	4_14
3_16	1	1	2021	525	5	0	2	0	0	1	1	2_17	4_15
3_17	1	1	1761	553	4	1	0	1	1	0	1	2_18	4_16
3_18	2	4	984, 1032	558, 564, 796, 577	2	1	0,1	1	0	0	0	2_19	4_17
3_19	2	3	1040, 2192	565, 576, 860	2,5	1	1	1	1	0	0	2_20	4_18
3_20	1	2	864	566, 571	2	0	2	0	0	0	0	2_21	4_19
3_21	1	1	2097	590	5	0	3	1	1	0	1	2_23	4_20
3_22	1	1	825	595	2	0	1	0	1	0	1	2_24	4_21
3_23	2	14	192, 193	605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744	0	1	0	0	0	0	0,1	2_25	4_22
3_24	2	5	2116, 2164	617, 623, 848, 855, 635	5	1	0,1	0	0	1	0	2_26	4_23
3_25	2	3	2220, 2268	629, 867, 756	5	1	2,3	0	1	1	0	2_27	4_24
3_26	1	1	304	648	0	1	2	0	2	0	0	2_28	4_25
3_27	1	2	209	649, 725	0	1	0	0	2	0	1	2_29	4_5
3_28	1	1	1969	745	5	0	1	0	0	0	1	2_30	4_26
3_29	1	1	2141	750	5	1	0	1	0	1	1	2_31	4_27

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
4_0	26	341	970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816	77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596	2, 0, 4, 5	1, 0	0, 3, 1, 2	0	1, 0	0	2, 0, 1	3_2, 3_0	5_0
4_1	4	8	1926, 2118, 2022, 1974	45, 161, 392, 399, 167, 263, 362, 277	5	0, 1	0, 2, 1	0	0	1	2	3_1	5_1
4_2	4	26	1978, 1986, 1594, 1977	117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5, 4	0	1	0	1, 2	0	2, 1	3_3	5_2
4_3	4	22	98, 146, 2018, 97	137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776	0, 5	0	2, 3	0	0	0	2, 1	3_4	5_0
4_4	1	2	254	149, 387	0	1	1	0	1	1	2	3_5	5_3
4_5	11	94	10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209	150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725	0, 4	0, 1	0, 3, 1	0	1, 2	0	2, 1	3_6, 3_27	5_4
4_6	1	1	2186	179	5	1	1	1	0	0	2	3_7	5_5
4_7	2	7	1681, 1585	184, 805, 806, 807, 808, 811, 812	4	0	3, 1	0	0	0	1	3_8	5_6
4_8	1	1	1065	187	2	1	2	0	1	0	1	3_9	5_7
4_9	1	1	1137	212	2	1	3	1	1	0	1	3_10	5_8
4_10	2	2	874, 872	308, 583	2	0	2	0	1	0	2, 0	3_11	5_9
4_11	1	1	994	320	2	1	0	1	1	0	2	3_12	5_10

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
4_12	1	1	1018	327	2	1	1	0	1	0	2	3_13	5_11
4_13	3	38	49, 48, 240	505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868	0	0,1	1	0	0	0	1,0	3_14	5_12
4_14	4	7	205, 2125, 2117, 2173	513, 642, 880, 654, 660, 892, 757	0,5	1	0,1	0	1,0	1	1	3_15	5_13
4_15	1	1	2021	525	5	0	2	0	0	1	1	3_16	5_14
4_16	1	1	1761	553	4	1	0	1	1	0	1	3_17	5_15
4_17	2	4	984, 1032	558, 564, 796, 577	2	1	0,1	1	0	0	0	3_18	5_16
4_18	2	3	1040, 2192	565, 576, 860	2,5	1	1	1	1	0	0	3_19	5_17
4_19	1	2	864	566, 571	2	0	2	0	0	0	0	3_20	5_18
4_20	1	1	2097	590	5	0	3	1	1	0	1	3_21	5_19
4_21	1	1	825	595	2	0	1	0	1	0	1	3_22	5_20
4_22	2	14	192, 193	605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744	0	1	0	0	0	0	0,1	3_23	5_21
4_23	2	5	2116, 2164	617, 623, 848, 855, 635	5	1	0,1	0	0	1	0	3_24	5_22
4_24	2	3	2220, 2268	629, 867, 756	5	1	2,3	0	1	1	0	3_25	5_23
4_25	1	1	304	648	0	1	2	0	2	0	0	3_26	5_24
4_26	1	1	1969	745	5	0	1	0	0	0	1	3_28	5_25
4_27	1	1	2141	750	5	1	0	1	0	1	1	3_29	5_26

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
5_0	30	363	98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816	137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596	0,5,2,4	0,1	2,3,0,1	0	0,1	0	2,1,0	4_3, 4_0	6_0
5_1	4	8	1926, 2118, 2022, 1974	45, 161, 392, 399, 167, 263, 362, 277	5	0,1	0,2,1	0	0	1	2	4_1	6_1
5_2	4	26	1978, 1986, 1594, 1977	117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5,4	0	1	0	1,2	0	2,1	4_2	6_2
5_3	1	2	254	149, 387	0	1	1	0	1	1	2	4_4	6_3
5_4	11	94	10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209	150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725	0,4	0,1	0,3,1	0	1,2	0	2,1	4_5	6_4
5_5	1	1	2186	179	5	1	1	1	0	0	2	4_6	6_5
5_6	2	7	1681, 1585	184, 805, 806, 807, 808, 811, 812	4	0	3,1	0	0	0	1	4_7	6_0
5_7	1	1	1065	187	2	1	2	0	1	0	1	4_8	6_6
5_8	1	1	1137	212	2	1	3	1	1	0	1	4_9	6_7
5_9	2	2	874, 872	308, 583	2	0	2	0	1	0	2,0	4_10	6_8
5_10	1	1	994	320	2	1	0	1	1	0	2	4_11	6_9
5_11	1	1	1018	327	2	1	1	0	1	0	2	4_12	6_10

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
5_12	3	38	49, 48, 240	505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868	0	0,1	1	0	0	0	1,0	4_13	6_11
5_13	4	7	205, 2125, 2117, 2173	513, 642, 880, 654, 660, 892, 757	0,5	1	0,1	0	1,0	1	1	4_14	6_12
5_14	1	1	2021	525	5	0	2	0	0	1	1	4_15	6_13
5_15	1	1	1761	553	4	1	0	1	1	0	1	4_16	6_14
5_16	2	4	984, 1032	558, 564, 796, 577	2	1	0,1	1	0	0	0	4_17	6_15
5_17	2	3	1040, 2192	565, 576, 860	2,5	1	1	1	1	0	0	4_18	6_16
5_18	1	2	864	566, 571	2	0	2	0	0	0	0	4_19	6_17
5_19	1	1	2097	590	5	0	3	1	1	0	1	4_20	6_18
5_20	1	1	825	595	2	0	1	0	1	0	1	4_21	6_19
5_21	2	14	192, 193	605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744	0	1	0	0	0	0	0,1	4_22	6_20
5_22	2	5	2116, 2164	617, 623, 848, 855, 635	5	1	0,1	0	0	1	0	4_23	6_21
5_23	2	3	2220, 2268	629, 867, 756	5	1	2,3	0	1	1	0	4_24	6_22
5_24	1	1	304	648	0	1	2	0	2	0	0	4_25	6_23
5_25	1	1	1969	745	5	0	1	0	0	0	1	4_26	6_24
5_26	1	1	2141	750	5	1	0	1	0	1	1	4_27	6_25

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
6_0	32	370	1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816	184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596	4,0,5,2	0,1	3,1,2,0	0	0,1	0	1,2,0	5_6, 5_0	7_0
6_1	4	8	1926, 2118, 2022, 1974	45, 161, 392, 399, 167, 263, 362, 277	5	0,1	0,2,1	0	0	1	2	5_1	7_1
6_2	4	26	1978, 1986, 1594, 1977	117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5,4	0	1	0	1,2	0	2,1	5_2	7_2
6_3	1	2	254	149, 387	0	1	1	0	1	1	2	5_3	7_3
6_4	11	94	10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209	150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725	0,4	0,1	0,3,1	0	1,2	0	2,1	5_4	7_4
6_5	1	1	2186	179	5	1	1	1	0	0	2	5_5	7_5
6_6	1	1	1065	187	2	1	2	0	1	0	1	5_7	7_0
6_7	1	1	1137	212	2	1	3	1	1	0	1	5_8	7_6
6_8	2	2	874, 872	308, 583	2	0	2	0	1	0	2,0	5_9	7_7
6_9	1	1	994	320	2	1	0	1	1	0	2	5_10	7_8
6_10	1	1	1018	327	2	1	1	0	1	0	2	5_11	7_9

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
6_11	3	38	49, 48, 240	505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868	0	0,1	1	0	0	0	1,0	5_12	7_10
6_12	4	7	205, 2125, 2117, 2173	513, 642, 880, 654, 660, 892, 757	0,5	1	0,1	0	1,0	1	1	5_13	7_11
6_13	1	1	2021	525	5	0	2	0	0	1	1	5_14	7_12
6_14	1	1	1761	553	4	1	0	1	1	0	1	5_15	7_13
6_15	2	4	984, 1032	558, 564, 796, 577	2	1	0,1	1	0	0	0	5_16	7_14
6_16	2	3	1040, 2192	565, 576, 860	2,5	1	1	1	1	0	0	5_17	7_15
6_17	1	2	864	566, 571	2	0	2	0	0	0	0	5_18	7_16
6_18	1	1	2097	590	5	0	3	1	1	0	1	5_19	7_17
6_19	1	1	825	595	2	0	1	0	1	0	1	5_20	7_18
6_20	2	14	192, 193	605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744	0	1	0	0	0	0	0,1	5_21	7_19
6_21	2	5	2116, 2164	617, 623, 848, 855, 635	5	1	0,1	0	0	1	0	5_22	7_20
6_22	2	3	2220, 2268	629, 867, 756	5	1	2,3	0	1	1	0	5_23	7_21
6_23	1	1	304	648	0	1	2	0	2	0	0	5_24	7_22
6_24	1	1	1969	745	5	0	1	0	0	0	1	5_25	7_23
6_25	1	1	2141	750	5	1	0	1	0	1	1	5_26	7_24

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
7_0	33	371	1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816	187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596	2,4,0,5	1,0	2,3,1,0	0	1,0	0	1,2,0	6_6, 6_0	8_0
7_1	4	8	1926, 2118, 2022, 1974	45, 161, 392, 399, 167, 263, 362, 277	5	0,1	0,2,1	0	0	1	2	6_1	8_1
7_2	4	26	1978, 1986, 1594, 1977	117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5,4	0	1	0	1,2	0	2,1	6_2	8_2
7_3	1	2	254	149, 387	0	1	1	0	1	1	2	6_3	8_3
7_4	11	94	10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209	150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725	0,4	0,1	0,3,1	0	1,2	0	2,1	6_4	8_4
7_5	1	1	2186	179	5	1	1	1	0	0	2	6_5	8_5
7_6	1	1	1137	212	2	1	3	1	1	0	1	6_7	8_6
7_7	2	2	874, 872	308, 583	2	0	2	0	1	0	2,0	6_8	8_0
7_8	1	1	994	320	2	1	0	1	1	0	2	6_9	8_7
7_9	1	1	1018	327	2	1	1	0	1	0	2	6_10	8_8
7_10	3	38	49, 48, 240	505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868	0	0,1	1	0	0	0	1,0	6_11	8_9

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
7_11	4	7	205, 2125, 2117, 2173	513, 642, 880, 654, 660, 892, 757	0,5	1	0,1	0	1,0	1	1	6_12	8_10
7_12	1	1	2021	525	5	0	2	0	0	1	1	6_13	8_11
7_13	1	1	1761	553	4	1	0	1	1	0	1	6_14	8_12
7_14	2	4	984, 1032	558, 564, 796, 577	2	1	0,1	1	0	0	0	6_15	8_13
7_15	2	3	1040, 2192	565, 576, 860	2,5	1	1	1	1	0	0	6_16	8_14
7_16	1	2	864	566, 571	2	0	2	0	0	0	0	6_17	8_15
7_17	1	1	2097	590	5	0	3	1	1	0	1	6_18	8_16
7_18	1	1	825	595	2	0	1	0	1	0	1	6_19	8_17
7_19	2	14	192, 193	605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744	0	1	0	0	0	0	0,1	6_20	8_18
7_20	2	5	2116, 2164	617, 623, 848, 855, 635	5	1	0,1	0	0	1	0	6_21	8_19
7_21	2	3	2220, 2268	629, 867, 756	5	1	2,3	0	1	1	0	6_22	8_20
7_22	1	1	304	648	0	1	2	0	2	0	0	6_23	8_21
7_23	1	1	1969	745	5	0	1	0	0	0	1	6_24	8_22
7_24	1	1	2141	750	5	1	0	1	0	1	1	6_25	8_23

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
8_0	35	373	874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816	308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596	2,4,0,5	0,1	2,3,1,0	0	1,0	0	2,0,1	7_7, 7_0	9_0
8_1	4	8	1926, 2118, 2022, 1974	45, 161, 392, 399, 167, 263, 362, 277	5	0,1	0,2,1	0	0	1	2	7_1	9_1
8_2	4	26	1978, 1986, 1594, 1977	117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5,4	0	1	0	1,2	0	2,1	7_2	9_2
8_3	1	2	254	149, 387	0	1	1	0	1	1	2	7_3	9_3
8_4	11	94	10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209	150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725	0,4	0,1	0,3,1	0	1,2	0	2,1	7_4	9_4
8_5	1	1	2186	179	5	1	1	1	0	0	2	7_5	9_5
8_6	1	1	1137	212	2	1	3	1	1	0	1	7_6	9_6
8_7	1	1	994	320	2	1	0	1	1	0	2	7_8	9_7
8_8	1	1	1018	327	2	1	1	0	1	0	2	7_9	9_0
8_9	3	38	49, 48, 240	505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868	0	0,1	1	0	0	0	1,0	7_10	9_8

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
8_10	4	7	205, 2125, 2117, 2173	513, 642, 880, 654, 660, 892, 757	0,5	1	0,1	0	1,0	1	1	7_11	9_9
8_11	1	1	2021	525	5	0	2	0	0	1	1	7_12	9_10
8_12	1	1	1761	553	4	1	0	1	1	0	1	7_13	9_11
8_13	2	4	984, 1032	558, 564, 796, 577	2	1	0,1	1	0	0	0	7_14	9_12
8_14	2	3	1040, 2192	565, 576, 860	2,5	1	1	1	1	0	0	7_15	9_13
8_15	1	2	864	566, 571	2	0	2	0	0	0	0	7_16	9_14
8_16	1	1	2097	590	5	0	3	1	1	0	1	7_17	9_15
8_17	1	1	825	595	2	0	1	0	1	0	1	7_18	9_16
8_18	2	14	192, 193	605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744	0	1	0	0	0	0	0,1	7_19	9_17
8_19	2	5	2116, 2164	617, 623, 848, 855, 635	5	1	0,1	0	0	1	0	7_20	9_18
8_20	2	3	2220, 2268	629, 867, 756	5	1	2,3	0	1	1	0	7_21	9_19
8_21	1	1	304	648	0	1	2	0	2	0	0	7_22	9_20
8_22	1	1	1969	745	5	0	1	0	0	0	1	7_23	9_21
8_23	1	1	2141	750	5	1	0	1	0	1	1	7_24	9_22

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
9_0	36	374	1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816	327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596	2,4,0,5	1,0	1,2,3,0	0	1,0	0	2,0,1	8_8, 8_0	10_0
9_1	4	8	1926, 2118, 2022, 1974	45, 161, 392, 399, 167, 263, 362, 277	5	0,1	0,2,1	0	0	1	2	8_1	10_1
9_2	4	26	1978, 1986, 1594, 1977	117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5,4	0	1	0	1,2	0	2,1	8_2	10_2
9_3	1	2	254	149, 387	0	1	1	0	1	1	2	8_3	10_3
9_4	11	94	10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209	150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725	0,4	0,1	0,3,1	0	1,2	0	2,1	8_4	10_4
9_5	1	1	2186	179	5	1	1	1	0	0	2	8_5	10_5
9_6	1	1	1137	212	2	1	3	1	1	0	1	8_6	10_6
9_7	1	1	994	320	2	1	0	1	1	0	2	8_7	10_7
9_8	3	38	49, 48, 240	505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868	0	0,1	1	0	0	0	1,0	8_9	10_0
9_9	4	7	205, 2125, 2117, 2173	513, 642, 880, 654, 660, 892, 757	0,5	1	0,1	0	1,0	1	1	8_10	10_8

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
9_10	1	1	2021	525	5	0	2	0	0	1	1	8_11	10_9
9_11	1	1	1761	553	4	1	0	1	1	0	1	8_12	10_10
9_12	2	4	984, 1032	558, 564, 796, 577	2	1	0, 1	1	0	0	0	8_13	10_11
9_13	2	3	1040, 2192	565, 576, 860	2, 5	1	1	1	1	0	0	8_14	10_12
9_14	1	2	864	566, 571	2	0	2	0	0	0	0	8_15	10_13
9_15	1	1	2097	590	5	0	3	1	1	0	1	8_16	10_14
9_16	1	1	825	595	2	0	1	0	1	0	1	8_17	10_15
9_17	2	14	192, 193	605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744	0	1	0	0	0	0	0, 1	8_18	10_16
9_18	2	5	2116, 2164	617, 623, 848, 855, 635	5	1	0, 1	0	0	1	0	8_19	10_17
9_19	2	3	2220, 2268	629, 867, 756	5	1	2, 3	0	1	1	0	8_20	10_18
9_20	1	1	304	648	0	1	2	0	2	0	0	8_21	10_19
9_21	1	1	1969	745	5	0	1	0	0	0	1	8_22	10_20
9_22	1	1	2141	750	5	1	0	1	0	1	1	8_23	10_21

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
10_0	39	412	49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816	505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596	0, 2, 4, 5	0, 1	1, 2, 3, 0	0	0, 1	0	1, 0, 2	9_8, 9_0	11_0
10_1	4	8	1926, 2118, 2022, 1974	45, 161, 392, 399, 167, 263, 362, 277	5	0, 1	0, 2, 1	0	0	1	2	9_1	11_1
10_2	4	26	1978, 1986, 1594, 1977	117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5, 4	0	1	0	1, 2	0	2, 1	9_2	11_2
10_3	1	2	254	149, 387	0	1	1	0	1	1	2	9_3	11_3
10_4	11	94	10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209	150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725	0, 4	0, 1	0, 3, 1	0	1, 2	0	2, 1	9_4	11_4
10_5	1	1	2186	179	5	1	1	1	0	0	2	9_5	11_5
10_6	1	1	1137	212	2	1	3	1	1	0	1	9_6	11_6
10_7	1	1	994	320	2	1	0	1	1	0	2	9_7	11_7
10_8	4	7	205, 2125, 2117, 2173	513, 642, 880, 654, 660, 892, 757	0, 5	1	0, 1	0	1, 0	1	1	9_9	11_8

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
10_9	1	1	2021	525	5	0	2	0	0	1	1	9_10	11_9
10_10	1	1	1761	553	4	1	0	1	1	0	1	9_11	11_10
10_11	2	4	984, 1032	558, 564, 796, 577	2	1	0,1	1	0	0	0	9_12	11_11
10_12	2	3	1040, 2192	565, 576, 860	2,5	1	1	1	1	0	0	9_13	11_12
10_13	1	2	864	566, 571	2	0	2	0	0	0	0	9_14	11_0
10_14	1	1	2097	590	5	0	3	1	1	0	1	9_15	11_13
10_15	1	1	825	595	2	0	1	0	1	0	1	9_16	11_14
10_16	2	14	192, 193	605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744	0	1	0	0	0	0	0,1	9_17	11_15
10_17	2	5	2116, 2164	617, 623, 848, 855, 635	5	1	0,1	0	0	1	0	9_18	11_16
10_18	2	3	2220, 2268	629, 867, 756	5	1	2,3	0	1	1	0	9_19	11_17
10_19	1	1	304	648	0	1	2	0	2	0	0	9_20	11_18
10_20	1	1	1969	745	5	0	1	0	0	0	1	9_21	11_19
10_21	1	1	2141	750	5	1	0	1	0	1	1	9_22	11_20

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
11_0	40	414	864, 49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816	566, 571, 505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596	2,0,4,5	0,1	2,1,3,0	0	0,1	0	0,1,2	10_13, 10_0	12_0
11_1	4	8	1926, 2118, 2022, 1974	45, 161, 392, 399, 167, 263, 362, 277	5	0,1	0,2,1	0	0	1	2	10_1	12_1
11_2	4	26	1978, 1986, 1594, 1977	117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5,4	0	1	0	1,2	0	2,1	10_2	12_2
11_3	1	2	254	149, 387	0	1	1	0	1	1	2	10_3	12_3
11_4	11	94	10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209	150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725	0,4	0,1	0,3,1	0	1,2	0	2,1	10_4	12_4
11_5	1	1	2186	179	5	1	1	1	0	0	2	10_5	12_5
11_6	1	1	1137	212	2	1	3	1	1	0	1	10_6	12_6
11_7	1	1	994	320	2	1	0	1	1	0	2	10_7	12_7
11_8	4	7	205, 2125, 2117, 2173	513, 642, 880, 654, 660, 892, 757	0,5	1	0,1	0	1,0	1	1	10_8	12_8

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
11_9	1	1	2021	525	5	0	2	0	0	1	1	10_9	12_9
11_10	1	1	1761	553	4	1	0	1	1	0	1	10_10	12_10
11_11	2	4	984, 1032	558, 564, 796, 577	2	1	0, 1	1	0	0	0	10_11	12_11
11_12	2	3	1040, 2192	565, 576, 860	2, 5	1	1	1	1	0	0	10_12	12_12
11_13	1	1	2097	590	5	0	3	1	1	0	1	10_14	12_13
11_14	1	1	825	595	2	0	1	0	1	0	1	10_15	12_0
11_15	2	14	192, 193	605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744	0	1	0	0	0	0	0, 1	10_16	12_14
11_16	2	5	2116, 2164	617, 623, 848, 855, 635	5	1	0, 1	0	0	1	0	10_17	12_15
11_17	2	3	2220, 2268	629, 867, 756	5	1	2, 3	0	1	1	0	10_18	12_16
11_18	1	1	304	648	0	1	2	0	2	0	0	10_19	12_17
11_19	1	1	1969	745	5	0	1	0	0	0	1	10_20	12_18
11_20	1	1	2141	750	5	1	0	1	0	1	1	10_21	12_19
12_0	41	415	825, 864, 49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816	595, 566, 571, 505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596	2, 0, 4, 5	0, 1	1, 2, 3, 0	0	1, 0	0	1, 0, 2	11_14, 11_0	13_0

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
12_1	4	8	1926, 2118, 2022, 1974	45, 161, 392, 399, 167, 263, 362, 277	5	0,1	0,2,1	0	0	1	2	11_1	13_1
12_2	4	26	1978, 1986, 1594, 1977	117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5,4	0	1	0	1,2	0	2,1	11_2	13_2
12_3	1	2	254	149, 387	0	1	1	0	1	1	2	11_3	13_3
12_4	11	94	10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209	150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725	0,4	0,1	0,3,1	0	1,2	0	2,1	11_4	13_4
12_5	1	1	2186	179	5	1	1	1	0	0	2	11_5	13_5
12_6	1	1	1137	212	2	1	3	1	1	0	1	11_6	13_6
12_7	1	1	994	320	2	1	0	1	1	0	2	11_7	13_7
12_8	4	7	205, 2125, 2117, 2173	513, 642, 880, 654, 660, 892, 757	0,5	1	0,1	0	1,0	1	1	11_8	13_8
12_9	1	1	2021	525	5	0	2	0	0	1	1	11_9	13_9
12_10	1	1	1761	553	4	1	0	1	1	0	1	11_10	13_10
12_11	2	4	984, 1032	558, 564, 796, 577	2	1	0,1	1	0	0	0	11_11	13_11
12_12	2	3	1040, 2192	565, 576, 860	2,5	1	1	1	1	0	0	11_12	13_12
12_13	1	1	2097	590	5	0	3	1	1	0	1	11_13	13_13
12_14	2	14	192, 193	605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744	0	1	0	0	0	0	0,1	11_15	13_0
12_15	2	5	2116, 2164	617, 623, 848, 855, 635	5	1	0,1	0	0	1	0	11_16	13_14
12_16	2	3	2220, 2268	629, 867, 756	5	1	2,3	0	1	1	0	11_17	13_15
12_17	1	1	304	648	0	1	2	0	2	0	0	11_18	13_16
12_18	1	1	1969	745	5	0	1	0	0	0	1	11_19	13_17
12_19	1	1	2141	750	5	1	0	1	0	1	1	11_20	13_18

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
13_0	43	429	192, 193, 825, 864, 49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816	605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744, 595, 566, 571, 505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596	0, 2, 4, 5	1, 0	0, 1, 2, 3	0	0, 1	0	0, 1, 2	12_14, 12_0	14_0
13_1	4	8	1926, 2118, 2022, 1974	45, 161, 392, 399, 167, 263, 362, 277	5	0, 1	0, 2, 1	0	0	1	2	12_1	14_1
13_2	4	26	1978, 1986, 1594, 1977	117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5, 4	0	1	0	1, 2	0	2, 1	12_2	14_2
13_3	1	2	254	149, 387	0	1	1	0	1	1	2	12_3	14_3
13_4	11	94	10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209	150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725	0, 4	0, 1	0, 3, 1	0	1, 2	0	2, 1	12_4	14_4
13_5	1	1	2186	179	5	1	1	1	0	0	2	12_5	14_5
13_6	1	1	1137	212	2	1	3	1	1	0	1	12_6	14_6
13_7	1	1	994	320	2	1	0	1	1	0	2	12_7	14_7

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
13_8	4	7	205, 2125, 2117, 2173	513, 642, 880, 654, 660, 892, 757	0,5	1	0,1	0	1,0	1	1	12_8	14_8
13_9	1	1	2021	525	5	0	2	0	0	1	1	12_9	14_9
13_10	1	1	1761	553	4	1	0	1	1	0	1	12_10	14_10
13_11	2	4	984, 1032	558, 564, 796, 577	2	1	0,1	1	0	0	0	12_11	14_11
13_12	2	3	1040, 2192	565, 576, 860	2,5	1	1	1	1	0	0	12_12	14_12
13_13	1	1	2097	590	5	0	3	1	1	0	1	12_13	14_13
13_14	2	5	2116, 2164	617, 623, 848, 855, 635	5	1	0,1	0	0	1	0	12_15	14_14
13_15	2	3	2220, 2268	629, 867, 756	5	1	2,3	0	1	1	0	12_16	14_15
13_16	1	1	304	648	0	1	2	0	2	0	0	12_17	14_16
13_17	1	1	1969	745	5	0	1	0	0	0	1	12_18	14_0
13_18	1	1	2141	750	5	1	0	1	0	1	1	12_19	14_17
14_0	44	430	1969, 192, 193, 825, 864, 49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816	745, 605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744, 595, 566, 571, 505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596	5,0,2,4	0,1	1,0,2,3	0	0,1	0	1,0,2	13_17, 13_0	15_0

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
14_1	4	8	1926, 2118, 2022, 1974	45, 161, 392, 399, 167, 263, 362, 277	5	0, 1	0, 2, 1	0	0	1	2	13_1	15_0
14_2	4	26	1978, 1986, 1594, 1977	117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5, 4	0	1	0	1, 2	0	2, 1	13_2	15_0
14_3	1	2	254	149, 387	0	1	1	0	1	1	2	13_3	15_1
14_4	11	94	10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209	150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725	0, 4	0, 1	0, 3, 1	0	1, 2	0	2, 1	13_4	15_1
14_5	1	1	2186	179	5	1	1	1	0	0	2	13_5	15_2
14_6	1	1	1137	212	2	1	3	1	1	0	1	13_6	15_3
14_7	1	1	994	320	2	1	0	1	1	0	2	13_7	15_4
14_8	4	7	205, 2125, 2117, 2173	513, 642, 880, 654, 660, 892, 757	0, 5	1	0, 1	0	1, 0	1	1	13_8	15_5
14_9	1	1	2021	525	5	0	2	0	0	1	1	13_9	15_6
14_10	1	1	1761	553	4	1	0	1	1	0	1	13_10	15_7
14_11	2	4	984, 1032	558, 564, 796, 577	2	1	0, 1	1	0	0	0	13_11	15_8
14_12	2	3	1040, 2192	565, 576, 860	2, 5	1	1	1	1	0	0	13_12	15_8
14_13	1	1	2097	590	5	0	3	1	1	0	1	13_13	15_9
14_14	2	5	2116, 2164	617, 623, 848, 855, 635	5	1	0, 1	0	0	1	0	13_14	15_5
14_15	2	3	2220, 2268	629, 867, 756	5	1	2, 3	0	1	1	0	13_15	15_10
14_16	1	1	304	648	0	1	2	0	2	0	0	13_16	15_11
14_17	1	1	2141	750	5	1	0	1	0	1	1	13_18	15_5

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
15_0	52	464	1926, 2118, 2022, 1974, 1969, 192, 193, 825, 864, 49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816, 1978, 1986, 1594, 1977	45, 161, 392, 399, 167, 263, 362, 277, 745, 605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744, 595, 566, 571, 505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596, 117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5,0,2,4	0,1	0,2,1,3	0	0, 1, 2	1,0	2,1,0	14_1, 14_0, 14_2	16_0
15_1	12	96	254, 10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209	149, 387, 150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725	0,4	1,0	1,0,3	0	1, 2	1,0	2,1	14_3, 14_4	16_0
15_2	1	1	2186	179	5	1	1	1	0	0	2	14_5	16_1
15_3	1	1	1137	212	2	1	3	1	1	0	1	14_6	16_2
15_4	1	1	994	320	2	1	0	1	1	0	2	14_7	16_3
15_5	7	13	205, 2125, 2117, 2173, 2116, 2164, 2141	513, 642, 880, 654, 660, 892, 757, 617, 623, 848, 855, 635, 750	0,5	1	0,1	0,1	1, 0	1	1,0	14_8, 14_14, 14_17	16_4
15_6	1	1	2021	525	5	0	2	0	0	1	1	14_9	16_5
15_7	1	1	1761	553	4	1	0	1	1	0	1	14_10	16_6
15_8	4	7	984, 1032, 1040, 2192	558, 564, 796, 577, 565, 576, 860	2,5	1	0,1	1	0, 1	0	0	14_11, 14_12	16_7

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
15_9	1	1	2097	590	5	0	3	1	1	0	1	14_13	16_8
15_10	2	3	2220, 2268	629, 867, 756	5	1	2, 3	0	1	1	0	14_15	16_9
15_11	1	1	304	648	0	1	2	0	2	0	0	14_16	16_10
16_0	64	560	254, 10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209, 1926, 2118, 2022, 1974, 1969, 192, 193, 825, 864, 49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816, 1978, 1986, 1594, 1977	149, 387, 150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725, 45, 161, 392, 399, 167, 263, 362, 277, 745, 605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744, 595, 566, 571, 505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596, 117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	0, 4, 5, 2	1, 0	1, 0, 3, 2	0	1, 2, 0	1, 0	2, 1, 0	15_1, 15_0	17_0
16_1	1	1	2186	179	5	1	1	1	0	0	2	15_2	17_1
16_2	1	1	1137	212	2	1	3	1	1	0	1	15_3	17_2
16_3	1	1	994	320	2	1	0	1	1	0	2	15_4	17_3
16_4	7	13	205, 2125, 2117, 2173, 2116, 2164, 2141	513, 642, 880, 654, 660, 892, 757, 617, 623, 848, 855, 635, 750	0, 5	1	0, 1	0, 1	1, 0	1	1, 0	15_5	17_0

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
16_5	1	1	2021	525	5	0	2	0	0	1	1	15_6	17_4
16_6	1	1	1761	553	4	1	0	1	1	0	1	15_7	17_5
16_7	4	7	984, 1032, 1040, 2192	558, 564, 796, 577, 565, 576, 860	2, 5	1	0, 1	1	0, 1	0	0	15_8	17_6
16_8	1	1	2097	590	5	0	3	1	1	0	1	15_9	17_7
16_9	2	3	2220, 2268	629, 867, 756	5	1	2, 3	0	1	1	0	15_10	17_8
16_10	1	1	304	648	0	1	2	0	2	0	0	15_11	17_9
17_0	71	573	205, 2125, 2117, 2173, 2116, 2164, 2141, 254, 10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209, 1926, 2118, 2022, 1974, 1969, 192, 193, 825, 864, 49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816, 1978, 1986, 1594, 1977	513, 642, 880, 654, 660, 892, 757, 617, 623, 848, 855, 635, 750, 149, 387, 150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725, 45, 161, 392, 399, 167, 263, 362, 277, 745, 605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744, 595, 566, 571, 505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596, 117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	0, 5, 4, 2	1, 0	0, 1, 3, 2	0, 1	1, 0, 2	1, 0	1, 0, 2	16_4, 16_0	18_0

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
17_1	1	1	2186	179	5	1	1	1	0	0	2	16_1	18_0
17_2	1	1	1137	212	2	1	3	1	1	0	1	16_2	18_1
17_3	1	1	994	320	2	1	0	1	1	0	2	16_3	18_2
17_4	1	1	2021	525	5	0	2	0	0	1	1	16_5	18_3
17_5	1	1	1761	553	4	1	0	1	1	0	1	16_6	18_4
17_6	4	7	984, 1032, 1040, 2192	558, 564, 796, 577, 565, 576, 860	2, 5	1	0, 1	1	0, 1	0	0	16_7	18_5
17_7	1	1	2097	590	5	0	3	1	1	0	1	16_8	18_6
17_8	2	3	2220, 2268	629, 867, 756	5	1	2, 3	0	1	1	0	16_9	18_7
17_9	1	1	304	648	0	1	2	0	2	0	0	16_10	18_8

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
18_0	72	574	2186, 205, 2125, 2117, 2173, 2116, 2164, 2141, 254, 10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209, 1926, 2118, 2022, 1974, 1969, 192, 193, 825, 864, 49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816, 1978, 1986, 1594, 1977	179, 513, 642, 880, 654, 660, 892, 757, 617, 623, 848, 855, 635, 750, 149, 387, 150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725, 45, 161, 392, 399, 167, 263, 362, 277, 745, 605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744, 595, 566, 571, 505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596, 117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5, 0, 4, 2	1, 0	1, 0, 3, 2	1, 0	0, 1, 2	0, 1	2, 1, 0	17_1, 17_0	19_0
18_1	1	1	1137	212	2	1	3	1	1	0	1	17_2	19_0
18_2	1	1	994	320	2	1	0	1	1	0	2	17_3	19_1
18_3	1	1	2021	525	5	0	2	0	0	1	1	17_4	19_2
18_4	1	1	1761	553	4	1	0	1	1	0	1	17_5	19_3
18_5	4	7	984, 1032, 1040, 2192	558, 564, 796, 577, 565, 576, 860	2, 5	1	0, 1	1	0, 1	0	0	17_6	19_4
18_6	1	1	2097	590	5	0	3	1	1	0	1	17_7	19_5
18_7	2	3	2220, 2268	629, 867, 756	5	1	2, 3	0	1	1	0	17_8	19_6

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
18_8	1	1	304	648	0	1	2	0	2	0	0	17_9	19_7
19_0	73	575	1137, 2186, 205, 2125, 2117, 2173, 2116, 2164, 2141, 254, 10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209, 1926, 2118, 2022, 1974, 1969, 192, 193, 825, 864, 49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816, 1978, 1986, 1594, 1977	212, 179, 513, 642, 880, 654, 660, 892, 757, 617, 623, 848, 855, 635, 750, 149, 387, 150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725, 45, 161, 392, 399, 167, 263, 362, 277, 745, 605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744, 595, 566, 571, 505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596, 117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	2,5,0,4	1,0	3,1,0,2	1,0	1,0,2	0,1	1,2,0	18_1, 18_0	20_0
19_1	1	1	994	320	2	1	0	1	1	0	2	18_2	20_0
19_2	1	1	2021	525	5	0	2	0	0	1	1	18_3	20_1
19_3	1	1	1761	553	4	1	0	1	1	0	1	18_4	20_2
19_4	4	7	984, 1032, 1040, 2192	558, 564, 796, 577, 565, 576, 860	2,5	1	0,1	1	0,1	0	0	18_5	20_3
19_5	1	1	2097	590	5	0	3	1	1	0	1	18_6	20_4

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
19_6	2	3	2220, 2268	629, 867, 756	5	1	2,3	0	1	1	0	18_7	20_5
19_7	1	1	304	648	0	1	2	0	2	0	0	18_8	20_6
20_0	74	576	994, 1137, 2186, 205, 2125, 2117, 2173, 2116, 2164, 2141, 254, 10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209, 1926, 2118, 2022, 1974, 1969, 192, 193, 825, 864, 49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816, 1978, 1986, 1594, 1977	320, 212, 179, 513, 642, 880, 654, 660, 892, 757, 617, 623, 848, 855, 635, 750, 149, 387, 150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725, 45, 161, 392, 399, 167, 263, 362, 277, 745, 605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744, 595, 566, 571, 505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596, 117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	2,5,0,4	1,0	0,3,1,2	1,0	1,0,2	0,1	2,1,0	19_1, 19_0	21_0
20_1	1	1	2021	525	5	0	2	0	0	1	1	19_2	21_0
20_2	1	1	1761	553	4	1	0	1	1	0	1	19_3	21_1
20_3	4	7	984, 1032, 1040, 2192	558, 564, 796, 577, 565, 576, 860	2,5	1	0,1	1	0,1	0	0	19_4	21_2
20_4	1	1	2097	590	5	0	3	1	1	0	1	19_5	21_3

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
20_5	2	3	2220, 2268	629, 867, 756	5	1	2,3	0	1	1	0	19_6	21_4
20_6	1	1	304	648	0	1	2	0	2	0	0	19_7	21_5
21_0	75	577	2021, 994, 1137, 2186, 205, 2125, 2117, 2173, 2116, 2164, 2141, 254, 10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209, 1926, 2118, 2022, 1974, 1969, 192, 193, 825, 864, 49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816, 1978, 1986, 1594, 1977	525, 320, 212, 179, 513, 642, 880, 654, 660, 892, 757, 617, 623, 848, 855, 635, 750, 149, 387, 150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725, 45, 161, 392, 399, 167, 263, 362, 277, 745, 605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744, 595, 566, 571, 505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596, 117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5,2,0,4	0,1	2,0,3,1	0,1	0,1,2	1,0	1,2,0	20_1, 20_0	22_0
21_1	1	1	1761	553	4	1	0	1	1	0	1	20_2	22_0
21_2	4	7	984, 1032, 1040, 2192	558, 564, 796, 577, 565, 576, 860	2,5	1	0,1	1	0,1	0	0	20_3	22_1
21_3	1	1	2097	590	5	0	3	1	1	0	1	20_4	22_2

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
21_4	2	3	2220, 2268	629, 867, 756	5	1	2,3	0	1	1	0	20_5	22_3
21_5	1	1	304	648	0	1	2	0	2	0	0	20_6	22_4
22_0	76	578	1761, 2021, 994, 1137, 2186, 205, 2125, 2117, 2173, 2116, 2164, 2141, 254, 10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209, 1926, 2118, 2022, 1974, 1969, 192, 193, 825, 864, 49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816, 1978, 1986, 1594, 1977	553, 525, 320, 212, 179, 513, 642, 880, 654, 660, 892, 757, 617, 623, 848, 855, 635, 750, 149, 387, 150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725, 45, 161, 392, 399, 167, 263, 362, 277, 745, 605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744, 595, 566, 571, 505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596, 117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	4,5,2,0	1,0	0,2,3,1	1,0	1,0,2	0,1	1,2,0	21_1, 21_0	23_0
22_1	4	7	984, 1032, 1040, 2192	558, 564, 796, 577, 565, 576, 860	2,5	1	0,1	1	0,1	0	0	21_2	23_0
22_2	1	1	2097	590	5	0	3	1	1	0	1	21_3	23_1
22_3	2	3	2220, 2268	629, 867, 756	5	1	2,3	0	1	1	0	21_4	23_2

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
22_4	1	1	304	648	0	1	2	0	2	0	0	21_5	23_3
23_0	80	585	984, 1032, 1040, 2192, 1761, 2021, 994, 1137, 2186, 205, 2125, 2117, 2173, 2116, 2164, 2141, 254, 10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209, 1926, 2118, 2022, 1974, 1969, 192, 193, 825, 864, 49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816, 1978, 1986, 1594, 1977	558, 564, 796, 577, 565, 576, 860, 553, 525, 320, 212, 179, 513, 642, 880, 654, 660, 892, 757, 617, 623, 848, 855, 635, 750, 149, 387, 150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725, 45, 161, 392, 399, 167, 263, 362, 277, 745, 605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744, 595, 566, 571, 505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596, 117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	2, 5, 4, 0	1, 0	0, 1, 2, 3	1, 0	0, 1, 2	0, 1	0, 1, 2	22_1, 22_0	24_0
23_1	1	1	2097	590	5	0	3	1	1	0	1	22_2	24_0
23_2	2	3	2220, 2268	629, 867, 756	5	1	2, 3	0	1	1	0	22_3	24_1
23_3	1	1	304	648	0	1	2	0	2	0	0	22_4	24_2

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
24_0	81	586	2097, 984, 1032, 1040, 2192, 1761, 2021, 994, 1137, 2186, 205, 2125, 2117, 2173, 2116, 2164, 2141, 254, 10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209, 1926, 2118, 2022, 1974, 1969, 192, 193, 825, 864, 49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816, 1978, 1986, 1594, 1977	590, 558, 564, 796, 577, 565, 576, 860, 553, 525, 320, 212, 179, 513, 642, 880, 654, 660, 892, 757, 617, 623, 848, 855, 635, 750, 149, 387, 150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725, 45, 161, 392, 399, 167, 263, 362, 277, 745, 605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744, 595, 566, 571, 505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596, 117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5,2,4,0	0,1	3,0,1,2	1,0	1,0,2	0,1	1,0,2	23_1, 23_0	25_0
24_1	2	3	2220, 2268	629, 867, 756	5	1	2,3	0	1	1	0	23_2	25_0
24_2	1	1	304	648	0	1	2	0	2	0	0	23_3	25_1

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
25_0	83	589	2220, 2268, 2097, 984, 1032, 1040, 2192, 1761, 2021, 994, 1137, 2186, 205, 2125, 2117, 2173, 2116, 2164, 2141, 254, 10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209, 1926, 2118, 2022, 1974, 1969, 192, 193, 825, 864, 49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816, 1978, 1986, 1594, 1977	629, 867, 756, 590, 558, 564, 796, 577, 565, 576, 860, 553, 525, 320, 212, 179, 513, 642, 880, 654, 660, 892, 757, 617, 623, 848, 855, 635, 750, 149, 387, 150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725, 45, 161, 392, 399, 167, 263, 362, 277, 745, 605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744, 595, 566, 571, 505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596, 117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	5, 2, 4, 0	1, 0	2, 3, 0, 1	0, 1	1, 0, 2	1, 0	0, 1, 2	24_1, 24_0	26_0
25_1	1	1	304	648	0	1	2	0	2	0	0	24_2	26_0

Table B-1. Clusters

Cluster	Number of Traces	Number of Projects	Traces	Projects	DK1	DS2	DS3	DS4	DK2	DK3	DS1	Children	Parent
26_0	84	590	304, 2220, 2268, 2097, 984, 1032, 1040, 2192, 1761, 2021, 994, 1137, 2186, 205, 2125, 2117, 2173, 2116, 2164, 2141, 254, 10, 1546, 1545, 1554, 153, 9, 57, 1593, 249, 1785, 209, 1926, 2118, 2022, 1974, 1969, 192, 193, 825, 864, 49, 48, 240, 1018, 874, 872, 1065, 1681, 1585, 98, 146, 2018, 97, 970, 962, 194, 960, 1, 0, 2, 770, 769, 1537, 1538, 1536, 1922, 1105, 1009, 961, 1057, 1113, 818, 866, 1010, 914, 817, 50, 1586, 816, 1978, 1986, 1594, 1977	648, 629, 867, 756, 590, 558, 564, 796, 577, 565, 576, 860, 553, 525, 320, 212, 179, 513, 642, 880, 654, 660, 892, 757, 617, 623, 848, 855, 635, 750, 149, 387, 150, 388, 222, 228, 229, 230, 231, 235, 236, 240, 325, 326, 331, 332, 336, 338, 339, 343, 344, 350, 351, 355, 356, 428, 435, 460, 223, 224, 301, 302, 305, 306, 311, 312, 313, 314, 318, 319, 324, 423, 424, 436, 549, 550, 557, 561, 562, 563, 661, 662, 663, 664, 665, 666, 667, 668, 671, 672, 673, 674, 675, 676, 678, 679, 680, 684, 685, 686, 687, 688, 691, 692, 698, 699, 700, 703, 704, 903, 904, 916, 448, 538, 638, 650, 697, 710, 711, 712, 715, 716, 770, 578, 649, 725, 45, 161, 392, 399, 167, 263, 362, 277, 745, 605, 606, 611, 612, 613, 618, 624, 625, 731, 843, 844, 856, 738, 744, 595, 566, 571, 505, 506, 507, 508, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 526, 527, 528, 733, 746, 747, 748, 751, 752, 758, 759, 760, 763, 764, 601, 630, 636, 637, 868, 327, 308, 583, 187, 184, 805, 806, 807, 808, 811, 812, 137, 162, 290, 375, 400, 180, 265, 270, 529, 530, 531, 532, 533, 534, 535, 536, 539, 540, 771, 772, 775, 776, 77, 315, 120, 208, 220, 232, 130, 155, 174, 367, 380, 412, 541, 542, 543, 544, 545, 546, 547, 548, 551, 552, 783, 784, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 121, 126, 132, 134, 139, 146, 147, 148, 151, 152, 159, 160, 163, 164, 171, 172, 175, 176, 246, 252, 254, 255, 256, 259, 260, 364, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 614, 619, 626, 627, 628, 631, 632, 639, 640, 643, 644, 651, 652, 655, 656, 726, 732, 734, 735, 736, 739, 740, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 122, 123, 124, 127, 128, 135, 136, 140, 241, 242, 243, 244, 247, 248, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 602, 603, 604, 607, 608, 615, 616, 620, 721, 722, 723, 724, 727, 728, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 125, 131, 133, 138, 144, 145, 156, 157, 158, 170, 245, 251, 253, 264, 266, 267, 268, 271, 272, 278, 279, 280, 283, 284, 291, 292, 295, 296, 363, 376, 61, 66, 72, 73, 74, 86, 87, 88, 91, 92, 98, 99, 100, 103, 104, 111, 112, 115, 116, 304, 62, 63, 64, 67, 68, 75, 76, 79, 80, 555, 556, 559, 560, 69, 81, 82, 185, 186, 191, 193, 198, 204, 205, 206, 211, 218, 781, 782, 785, 786, 787, 788, 791, 792, 793, 794, 795, 798, 799, 800, 804, 89, 93, 94, 101, 105, 106, 107, 113, 118, 119, 197, 203, 210, 216, 217, 181, 182, 192, 194, 199, 276, 70, 219, 183, 188, 195, 196, 200, 207, 307, 65, 71, 78, 84, 85, 90, 96, 97, 303, 316, 328, 83, 108, 109, 95, 114, 352, 102, 340, 110, 143, 168, 169, 258, 368, 190, 209, 215, 554, 567, 568, 572, 579, 580, 584, 591, 592, 596, 117, 221, 227, 337, 440, 459, 234, 317, 323, 330, 472, 427, 569, 570, 573, 574, 575, 690, 696, 702, 708, 709, 720, 816, 928, 940	0, 5, 2, 4	1, 0	2, 3, 0, 1	0, 1	2, 1, 0	0, 1	0, 1, 2	25_1, 25_0	none

Appendix C. Clusters of Problems

This appendix contains the attributes of the problems that followed traces with a high coverage (see Table 10-6 on page 230).

C.1 Trace 0

Table C- 1 shows how the projects that followed Trace 0 differ from each other in their constraints and requirements. We have highlighted the points in the table that show a major change in the constraints or requirements. These points are the candidate points for partitioning the set of projects into different groups that hopefully will have some correlation with the change of approaches in the trace or cluster of traces.

Table C-1. All Projects that followed trace 0. Total 52 projects.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
1	0.01	1000	"small-M"	1.0	0	0
2	0.01	1000	"small-M"	1.0	0	1
3	0.01	1000	"small-M"	1.0	0	2
4	0.01	1000	"small-M"	1.0	0	3
5	0.01	1000	"small-M"	1.0	1	0
6	0.01	1000	"small-M"	1.0	1	1
7	0.01	1000	"small-M"	1.0	1	2
8	0.01	1000	"small-M"	1.0	1	3
9	0.01	1000	"small-M"	1.0	2	0
10	0.01	1000	"small-M"	1.0	2	1
11	0.01	1000	"small-M"	1.0	2	2
12	0.01	1000	"small-M"	1.0	2	3
122	0.01	1000	"big-M"	1.0	0	1
123	0.01	1000	"big-M"	1.0	0	2
124	0.01	1000	"big-M"	1.0	0	3

Table C-1. All Projects that followed trace 0. Total 52 projects.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
127	0.01	1000	"big-M"	1.0	1	2
128	0.01	1000	"big-M"	1.0	1	3
135	0.01	1000	"big-M"	2.0	0	2
136	0.01	1000	"big-M"	2.0	0	3
140	0.01	1000	"big-M"	2.0	1	3
241	0.01	100	"small-M"	1.0	0	0
242	0.01	100	"small-M"	1.0	0	1
243	0.01	100	"small-M"	1.0	0	2
244	0.01	100	"small-M"	1.0	0	3
247	0.01	100	"small-M"	1.0	1	2
248	0.01	100	"small-M"	1.0	1	3
481	0.001	1000	"small-M"	1.0	0	0
482	0.001	1000	"small-M"	1.0	0	1
483	0.001	1000	"small-M"	1.0	0	2
484	0.001	1000	"small-M"	1.0	0	3
485	0.001	1000	"small-M"	1.0	1	0
486	0.001	1000	"small-M"	1.0	1	1
487	0.001	1000	"small-M"	1.0	1	2
488	0.001	1000	"small-M"	1.0	1	3
489	0.001	1000	"small-M"	1.0	2	0
490	0.001	1000	"small-M"	1.0	2	1
491	0.001	1000	"small-M"	1.0	2	2
492	0.001	1000	"small-M"	1.0	2	3
602	0.001	1000	"big-M"	1.0	0	1
603	0.001	1000	"big-M"	1.0	0	2
604	0.001	1000	"big-M"	1.0	0	3
607	0.001	1000	"big-M"	1.0	1	2
608	0.001	1000	"big-M"	1.0	1	3
615	0.001	1000	"big-M"	2.0	0	2
616	0.001	1000	"big-M"	2.0	0	3
620	0.001	1000	"big-M"	2.0	1	3
721	0.001	100	"small-M"	1.0	0	0
722	0.001	100	"small-M"	1.0	0	1

Table C-1. All Projects that followed trace 0. Total 52 projects.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
723	0.001	100	"small-M"	1.0	0	2
724	0.001	100	"small-M"	1.0	0	3
727	0.001	100	"small-M"	1.0	1	2
728	0.001	100	"small-M"	1.0	1	3

An interesting pattern in the projects that followed “Trace 0” is shown in Table C-2. The set of projects is divided into two similar subsets. The similarity is in the number of projects in a continuous block that have followed “Trace 0” as well as in the distance between these blocks.

Table C-2. Patterns in the projects that followed “Trace 0”.

Projects	number of projects in each subset	jump of the project ID to the next subset
1 to 12	12	110
122 to 124	3	3
127 to 128	2	7
135 to 136	2	4
140	1	101
241 to 244	4	3
247 to 248	2	233
481 to 492	12	110
602 to 604	3	3
607 to 608	2	7
615 to 616	2	4
620	1	101
721 to 724	4	3
727 to 728	2	N/A

C.2 Trace 1

Table C-3. All Projects that followed trace 1. Total 72 projects.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
13	0.01	1000	"small-M"	2.0	3.0	50%
14	0.01	1000	"small-M"	2.0	3.0	40%
15	0.01	1000	"small-M"	2.0	3.0	20%
16	0.01	1000	"small-M"	2.0	3.0	10%
17	0.01	1000	"small-M"	2.0	2.0	50%
18	0.01	1000	"small-M"	2.0	2.0	40%
19	0.01	1000	"small-M"	2.0	2.0	20%
20	0.01	1000	"small-M"	2.0	2.0	10%
21	0.01	1000	"small-M"	2.0	1.0	50%
22	0.01	1000	"small-M"	2.0	1.0	40%
23	0.01	1000	"small-M"	2.0	1.0	20%
24	0.01	1000	"small-M"	2.0	1.0	10%
121	0.01	1000	"big-M"	1.0	3.0	50%
126	0.01	1000	"big-M"	1.0	2.0	40%
132	0.01	1000	"big-M"	1.0	1.0	10%
134	0.01	1000	"big-M"	2.0	3.0	40%
139	0.01	1000	"big-M"	2.0	2.0	20%
146	0.01	1000	"big-M"	3.0	3.0	40%
147	0.01	1000	"big-M"	3.0	3.0	20%
148	0.01	1000	"big-M"	3.0	3.0	10%
151	0.01	1000	"big-M"	3.0	2.0	20%
152	0.01	1000	"big-M"	3.0	2.0	10%
159	0.01	1000	"big-M"	4.0	3.0	20%
160	0.01	1000	"big-M"	4.0	3.0	10%
163	0.01	1000	"big-M"	4.0	2.0	20%
164	0.01	1000	"big-M"	4.0	2.0	10%
171	0.01	1000	"big-M"	5.0	3.0	20%
172	0.01	1000	"big-M"	5.0	3.0	10%
175	0.01	1000	"big-M"	5.0	2.0	20%
176	0.01	1000	"big-M"	5.0	2.0	10%
246	0.01	100	"small-M"	1.0	2.0	40%

Table C-3. All Projects that followed trace 1. Total 72 projects.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
252	0.01	100	"small-M"	1.0	1.0	10%
254	0.01	100	"small-M"	2.0	3.0	40%
255	0.01	100	"small-M"	2.0	3.0	20%
256	0.01	100	"small-M"	2.0	3.0	10%
259	0.01	100	"small-M"	2.0	2.0	20%
260	0.01	100	"small-M"	2.0	2.0	10%
364	0.01	100	"big-M"	1.0	3.0	10%
493	0.001	1000	"small-M"	2.0	3.0	50%
494	0.001	1000	"small-M"	2.0	3.0	40%
495	0.001	1000	"small-M"	2.0	3.0	20%
496	0.001	1000	"small-M"	2.0	3.0	10%
497	0.001	1000	"small-M"	2.0	2.0	50%
498	0.001	1000	"small-M"	2.0	2.0	40%
499	0.001	1000	"small-M"	2.0	2.0	20%
500	0.001	1000	"small-M"	2.0	2.0	10%
501	0.001	1000	"small-M"	2.0	1.0	50%
502	0.001	1000	"small-M"	2.0	1.0	40%
503	0.001	1000	"small-M"	2.0	1.0	20%
504	0.001	1000	"small-M"	2.0	1.0	10%
614	0.001	1000	"big-M"	2.0	3.0	40%
619	0.001	1000	"big-M"	2.0	2.0	20%
626	0.001	1000	"big-M"	3.0	3.0	40%
627	0.001	1000	"big-M"	3.0	3.0	20%
628	0.001	1000	"big-M"	3.0	3.0	10%
631	0.001	1000	"big-M"	3.0	2.0	20%
632	0.001	1000	"big-M"	3.0	2.0	10%
639	0.001	1000	"big-M"	4.0	3.0	20%
640	0.001	1000	"big-M"	4.0	3.0	10%
643	0.001	1000	"big-M"	4.0	2.0	20%
644	0.001	1000	"big-M"	4.0	2.0	10%
651	0.001	1000	"big-M"	5.0	3.0	20%
652	0.001	1000	"big-M"	5.0	3.0	10%
655	0.001	1000	"big-M"	5.0	2.0	20%

Table C-3. All Projects that followed trace 1. Total 72 projects.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
656	0.001	1000	"big-M"	5.0	2.0	10%
726	0.001	100	"small-M"	1.0	2.0	40%
732	0.001	100	"small-M"	1.0	1.0	10%
734	0.001	100	"small-M"	2.0	3.0	40%
735	0.001	100	"small-M"	2.0	3.0	20%
736	0.001	100	"small-M"	2.0	3.0	10%
739	0.001	100	"small-M"	2.0	2.0	20%
740	0.001	100	"small-M"	2.0	2.0	10%

There is a pattern in the block of projects and the distance between these blocks for projects that followed “Trace 1”. While this pattern is not as good as the similar one for “Trace 0” it divides the set of projects into two subsets.

Table C-4. Patterns in the projects that followed “Trace 1”.

Projects	number of projects in each subset	jump of the project ID to the next subset
13 to 24	12	97
121	1	5
126	1	6
132	1	2
134	1	5
139	1	7
146 to 148	3	3
151 to 152	2	7
159 to 160	2	3
163 to 164	2	7
171 to 172	2	3
175 to176	2	70
246	1	6
252	1	2
254 to 256	3	3

Table C-4. Patterns in the projects that followed “Trace 1”.

Projects	number of projects in each subset	jump of the project ID to the next subset
259 to 260	2	104
364	1	129
493 to 504	12	110
614	1	5
619	1	7
626 to 628	3	3
631 to 632	2	7
639 to 640	2	3
643 to 644	2	7
651 to 652	2	3
655 to 656	2	70
726	1	6
732	1	2
734 to 736	3	3
739 to 740	2	N/A

C.3 Trace 2

Table C-5. All Projects that followed trace 2. Total 64 projects.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
25	0.01	1000	"small-M"	3.0	3.0	50%
26	0.01	1000	"small-M"	3.0	3.0	40%
27	0.01	1000	"small-M"	3.0	3.0	20%
28	0.01	1000	"small-M"	3.0	3.0	10%
29	0.01	1000	"small-M"	3.0	2.0	50%
30	0.01	1000	"small-M"	3.0	2.0	40%
31	0.01	1000	"small-M"	3.0	2.0	20%
32	0.01	1000	"small-M"	3.0	2.0	10%
33	0.01	1000	"small-M"	3.0	1.0	50%
34	0.01	1000	"small-M"	3.0	1.0	40%
35	0.01	1000	"small-M"	3.0	1.0	20%
36	0.01	1000	"small-M"	3.0	1.0	10%
37	0.01	1000	"small-M"	4.0	3.0	50%
38	0.01	1000	"small-M"	4.0	3.0	40%
39	0.01	1000	"small-M"	4.0	3.0	20%
40	0.01	1000	"small-M"	4.0	3.0	10%
41	0.01	1000	"small-M"	4.0	2.0	50%
42	0.01	1000	"small-M"	4.0	2.0	40%
43	0.01	1000	"small-M"	4.0	2.0	20%
44	0.01	1000	"small-M"	4.0	2.0	10%
46	0.01	1000	"small-M"	4.0	1.0	40%
47	0.01	1000	"small-M"	4.0	1.0	20%
48	0.01	1000	"small-M"	4.0	1.0	10%
49	0.01	1000	"small-M"	5.0	3.0	50%
50	0.01	1000	"small-M"	5.0	3.0	40%
51	0.01	1000	"small-M"	5.0	3.0	20%
52	0.01	1000	"small-M"	5.0	3.0	10%
53	0.01	1000	"small-M"	5.0	2.0	50%
54	0.01	1000	"small-M"	5.0	2.0	40%
55	0.01	1000	"small-M"	5.0	2.0	20%
56	0.01	1000	"small-M"	5.0	2.0	10%

Table C-5. All Projects that followed trace 2. Total 64 projects.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
58	0.01	1000	"small-M"	5.0	1.0	40%
59	0.01	1000	"small-M"	5.0	1.0	20%
60	0.01	1000	"small-M"	5.0	1.0	10%
125	0.01	1000	"big-M"	1.0	2.0	50%
131	0.01	1000	"big-M"	1.0	1.0	20%
133	0.01	1000	"big-M"	2.0	3.0	50%
138	0.01	1000	"big-M"	2.0	2.0	40%
144	0.01	1000	"big-M"	2.0	1.0	10%
145	0.01	1000	"big-M"	3.0	3.0	50%
156	0.01	1000	"big-M"	3.0	1.0	10%
157	0.01	1000	"big-M"	4.0	3.0	50%
158	0.01	1000	"big-M"	4.0	3.0	40%
170	0.01	1000	"big-M"	5.0	3.0	40%
245	0.01	100	"small-M"	1.0	2.0	50%
251	0.01	100	"small-M"	1.0	1.0	20%
253	0.01	100	"small-M"	2.0	3.0	50%
264	0.01	100	"small-M"	2.0	1.0	10%
266	0.01	100	"small-M"	3.0	3.0	40%
267	0.01	100	"small-M"	3.0	3.0	20%
268	0.01	100	"small-M"	3.0	3.0	10%
271	0.01	100	"small-M"	3.0	2.0	20%
272	0.01	100	"small-M"	3.0	2.0	10%
278	0.01	100	"small-M"	4.0	3.0	40%
279	0.01	100	"small-M"	4.0	3.0	20%
280	0.01	100	"small-M"	4.0	3.0	10%
283	0.01	100	"small-M"	4.0	2.0	20%
284	0.01	100	"small-M"	4.0	2.0	10%
291	0.01	100	"small-M"	5.0	3.0	20%
292	0.01	100	"small-M"	5.0	3.0	10%
295	0.01	100	"small-M"	5.0	2.0	20%
296	0.01	100	"small-M"	5.0	2.0	10%
363	0.01	100	"big-M"	1.0	3.0	20%
376	0.01	100	"big-M"	2.0	3.0	10%

C.4 Trace 49

Table C-6. All the projects that followed Trace 49. Total 33 Projects.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
505	0.001	1000	"small-M"	3.0	3.0	50%
506	0.001	1000	"small-M"	3.0	3.0	40%
507	0.001	1000	"small-M"	3.0	3.0	20%
508	0.001	1000	"small-M"	3.0	3.0	10%
509	0.001	1000	"small-M"	3.0	2.0	50%
510	0.001	1000	"small-M"	3.0	2.0	40%
511	0.001	1000	"small-M"	3.0	2.0	20%
512	0.001	1000	"small-M"	3.0	2.0	10%
514	0.001	1000	"small-M"	3.0	1.0	40%
515	0.001	1000	"small-M"	3.0	1.0	20%
516	0.001	1000	"small-M"	3.0	1.0	10%
517	0.001	1000	"small-M"	4.0	3.0	50%
518	0.001	1000	"small-M"	4.0	3.0	40%
519	0.001	1000	"small-M"	4.0	3.0	20%
520	0.001	1000	"small-M"	4.0	3.0	10%
521	0.001	1000	"small-M"	4.0	2.0	50%
522	0.001	1000	"small-M"	4.0	2.0	40%
523	0.001	1000	"small-M"	4.0	2.0	20%
524	0.001	1000	"small-M"	4.0	2.0	10%
526	0.001	1000	"small-M"	4.0	1.0	40%
527	0.001	1000	"small-M"	4.0	1.0	20%
528	0.001	1000	"small-M"	4.0	1.0	10%
733	0.001	100	"small-M"	2.0	3.0	50%
746	0.001	100	"small-M"	3.0	3.0	40%
747	0.001	100	"small-M"	3.0	3.0	20%
748	0.001	100	"small-M"	3.0	3.0	10%
751	0.001	100	"small-M"	3.0	2.0	20%
752	0.001	100	"small-M"	3.0	2.0	10%
758	0.001	100	"small-M"	4.0	3.0	40%
759	0.001	100	"small-M"	4.0	3.0	20%
760	0.001	100	"small-M"	4.0	3.0	10%

Table C-6. All the projects that followed Trace 49. Total 33 Projects.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
763	0.001	100	"small-M"	4.0	2.0	20%
764	0.001	100	"small-M"	4.0	2.0	10%

C.5 Trace 770

Table C-7. All the projects that followed Trace 770. Total 20.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
61	0.01	1000	"small-L"	1.0	3.0	50%
66	0.01	1000	"small-L"	1.0	2.0	40%
72	0.01	1000	"small-L"	1.0	1.0	10%
73	0.01	1000	"small-L"	2.0	3.0	50%
74	0.01	1000	"small-L"	2.0	3.0	40%
86	0.01	1000	"small-L"	3.0	3.0	40%
87	0.01	1000	"small-L"	3.0	3.0	20%
88	0.01	1000	"small-L"	3.0	3.0	10%
91	0.01	1000	"small-L"	3.0	2.0	20%
92	0.01	1000	"small-L"	3.0	2.0	10%
98	0.01	1000	"small-L"	4.0	3.0	40%
99	0.01	1000	"small-L"	4.0	3.0	20%
100	0.01	1000	"small-L"	4.0	3.0	10%
103	0.01	1000	"small-L"	4.0	2.0	20%
104	0.01	1000	"small-L"	4.0	2.0	10%
111	0.01	1000	"small-L"	5.0	3.0	20%
112	0.01	1000	"small-L"	5.0	3.0	10%
115	0.01	1000	"small-L"	5.0	2.0	20%
116	0.01	1000	"small-L"	5.0	2.0	10%
304	0.01	100	"small-L"	1.0	3.0	10%

C.6 Trace 1537

Table C-8. All projects that followed Trace 1537. Total 28.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
69	0.01	1000	"small-L"	1.0	1.0	50%
81	0.01	1000	"small-L"	2.0	1.0	50%
82	0.01	1000	"small-L"	2.0	1.0	40%
185	0.01	1000	"big-L"	1.0	2.0	50%
186	0.01	1000	"big-L"	1.0	2.0	40%

Table C-8. All projects that followed Trace 1537. Total 28.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
191	0.01	1000	"big-L"	1.0	1.0	20%
193	0.01	1000	"big-L"	2.0	3.0	50%
198	0.01	1000	"big-L"	2.0	2.0	40%
204	0.01	1000	"big-L"	2.0	1.0	10%
205	0.01	1000	"big-L"	3.0	3.0	50%
206	0.01	1000	"big-L"	3.0	3.0	40%
211	0.01	1000	"big-L"	3.0	2.0	20%
218	0.01	1000	"big-L"	4.0	3.0	40%
781	0.001	100	"small-L"	1.0	3.0	50%
782	0.001	100	"small-L"	1.0	3.0	40%
785	0.001	100	"small-L"	1.0	2.0	50%
786	0.001	100	"small-L"	1.0	2.0	40%
787	0.001	100	"small-L"	1.0	2.0	20%
788	0.001	100	"small-L"	1.0	2.0	10%
791	0.001	100	"small-L"	1.0	1.0	20%
792	0.001	100	"small-L"	1.0	1.0	10%
793	0.001	100	"small-L"	2.0	3.0	50%
794	0.001	100	"small-L"	2.0	3.0	40%
795	0.001	100	"small-L"	2.0	3.0	20%
798	0.001	100	"small-L"	2.0	2.0	40%
799	0.001	100	"small-L"	2.0	2.0	20%
800	0.001	100	"small-L"	2.0	2.0	10%
804	0.001	100	"small-L"	2.0	1.0	10%

C.7 Trace 1545

Table C-9. All projects that followed Trace 1545. Total 54.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
223	0.01	1000	“big-L”	4.0	2.0	20%
224	0.01	1000	“big-L”	4.0	2.0	10%
301	0.01	100	“small_L”	1.0	3.0	50%
302	0.01	100	"small-L"	1.0	3.0	40%
305	0.01	100	"small-L"	1.0	2.0	50%
306	0.01	100	"small-L"	1.0	2.0	40%
311	0.01	100	"small-L"	1.0	1.0	20%
312	0.01	100	"small-L"	1.0	1.0	10%
313	0.01	100	"small-L"	2.0	3.0	50%
314	0.01	100	"small-L"	2.0	3.0	40%
318	0.01	100	"small-L"	2.0	2.0	40%
319	0.01	100	"small-L"	2.0	2.0	20%
324	0.01	100	"small-L"	2.0	1.0	10%
423	0.01	100	"big-L"	1.0	3.0	20%
424	0.01	100	"big-L"	1.0	3.0	10%
436	0.01	100	"big-L"	2.0	3.0	10%
549	0.001	1000	"small-L"	1.0	1.0	50%
550	0.001	1000	"small-L"	1.0	1.0	40%
557	0.001	1000	"small-L"	2.0	2.0	50%
561	0.001	1000	"small-L"	2.0	1.0	50%
562	0.001	1000	"small-L"	2.0	1.0	40%
563	0.001	1000	"small-L"	2.0	1.0	20%
661	0.001	1000	"big-L"	1.0	3.0	50%
662	0.001	1000	"big-L"	1.0	3.0	40%
663	0.001	1000	"big-L"	1.0	3.0	20%
664	0.001	1000	"big-L"	1.0	3.0	10%
665	0.001	1000	"big-L"	1.0	2.0	50%
666	0.001	1000	"big-L"	1.0	2.0	40%
667	0.001	1000	"big-L"	1.0	2.0	20%
668	0.001	1000	"big-L"	1.0	2.0	10%
671	0.001	1000	"big-L"	1.0	1.0	20%

Table C-9. All projects that followed Trace 1545. Total 54.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
672	0.001	1000	"big-L"	1.0	1.0	10%
673	0.001	1000	"big-L"	2.0	3.0	50%
674	0.001	1000	"big-L"	2.0	3.0	40%
675	0.001	1000	"big-L"	2.0	3.0	20%
676	0.001	1000	"big-L"	2.0	3.0	10%
678	0.001	1000	"big-L"	2.0	2.0	40%
679	0.001	1000	"big-L"	2.0	2.0	20%
680	0.001	1000	"big-L"	2.0	2.0	10%
684	0.001	1000	"big-L"	2.0	1.0	10%
685	0.001	1000	"big-L"	3.0	3.0	50%
686	0.001	1000	"big-L"	3.0	3.0	40%
687	0.001	1000	"big-L"	3.0	3.0	20%
688	0.001	1000	"big-L"	3.0	3.0	10%
691	0.001	1000	"big-L"	3.0	2.0	20%
692	0.001	1000	"big-L"	3.0	2.0	10%
698	0.001	1000	"big-L"	4.0	3.0	40%
699	0.001	1000	"big-L"	4.0	3.0	20%
700	0.001	1000	"big-L"	4.0	3.0	10%
703	0.001	1000	"big-L"	4.0	2.0	20%
704	0.001	1000	"big-L"	4.0	2.0	10%
903	0.001	100	"big-L"	1.0	3.0	20%
904	0.001	100	"big-L"	1.0	3.0	10%
916	0.001	100	"big-L"	2.0	3.0	10%

C.8 Trace 1546

Table C-10. All projects that followed Trace 1546. Total 24.

Project ID	Constraint on Deflection	Constraint on Gain 1	Workspace	Workload (kg)	Settling Time (sec)	Maximum Overshoot
222	0.01	1000	"big-L"	4.0	1	1
228	0.01	1000	"big-L"	4.0	2	3
229	0.01	1000	"big-L"	5.0	0	0
230	0.01	1000	"big-L"	5.0	0	1
231	0.01	1000	"big-L"	5.0	0	2
235	0.01	1000	"big-L"	5.0	1	2
236	0.01	1000	"big-L"	5.0	1	3
240	0.01	1000	"big-L"	5.0	2	3
325	0.01	100	"small-L"	3.0	0	0
326	0.01	100	"small-L"	3.0	0	1
331	0.01	100	"small-L"	3.0	1	2
332	0.01	100	"small-L"	3.0	1	3
336	0.01	100	"small-L"	3.0	2	3
338	0.01	100	"small-L"	4.0	0	1
339	0.01	100	"small-L"	4.0	0	2
343	0.01	100	"small-L"	4.0	1	2
344	0.01	100	"small-L"	4.0	1	3
350	0.01	100	"small-L"	5.0	0	1
351	0.01	100	"small-L"	5.0	0	2
355	0.01	100	"small-L"	5.0	1	2
356	0.01	100	"small-L"	5.0	1	3
428	0.01	100	"big-L"	1.0	1	3
435	0.01	100	"big-L"	2.0	0	2
460	0.01	100	"big-L"	4.0	0	3

Appendix D. RD: User's Guide

D.1 Data Files Format

A data file is organized in the form of blocks. Lines of comments can be inserted in any line in the file by preceding the line by a number sign (#). Each block has a name and the name of the block should follow the tag `Block:`. The name of the block should be one word but combining words using underscore character '_' is permitted. Each block starts with a `{` and ends with a `}`. Blank lines, extra spaces, and tabs are ignored in reading a data file.

A block is consisted of a set of parameters. Each parameter has a name a type and a value. The name of the parameter should be only one word but combining words using underscore character '_' is permitted. The following types are recognized in a data file:

1. numeric scalar identified by =
2. string scalar identified by `:`
3. numeric vector started with `=[` and ended with `]` . Vector's elements are separtaed from each other by a space, and there should be at least one space around the start and end identifiers.
4. numeric matrix started with `=[[` , and ended with `]` . Except the first row each row starts with a `[` and all the rows should end with a `]` . There should be at least one space between matrix and row identifiers.

5. string vector started with `:[` and ended with `]`. Vector's elements are separtaed from each other by a space, and there should be at least one space around the start and end identifiers. boolean scalar identified by `::`

The contents of a data file is returned as a hashtable with block names as hash keys and the content of the blocks as values. The contents of each block is itself a hashtable with parameter names as hash keys and the contents of the parameters as values. The contents of each parameter is itself a hashtable with only one element that its key is the type of the parameter (= for numeric scalar, `:` for string scalar, `=[` for numeric vector, `=[[` for numeric matrix, and `:[` for string vector) and its vaue is the value of the parameter.

D.1.1 An example of a data file:

```
//This is a sample of the data file format
Block: DESIGN_REQUIREMENTS
{
      operational_plane :  horizontal
      workspace =[[ 1.0  1.1  1.0939  1.05  1.05  1.0939  1.3  1.4]
[ 0.15  0.15  0.1061  0.15  0.3  0.5561  0.6  0.6 ] ]
      workload = 1.0
      settling_time = 3.0
      maximum_overshoot = 50.0
}
Block: DESIGNER_TRACES
{
      Designer_1_1_depth = 0
      Designer_1_1_approaches = 2
      Designer_1_1_current_approach:minimize_link_lengths_summatin
      Designer_1_1_design_cases = 40
      Designer_1_1_suppliers :[ DesignRequirements ]
      Designer_1_1_consumers :[ Designer_1_2 Designer_1_3 ]
}
```

In the `DESIGNER TRACES` block the first row shows the name, the depth of the designer in a dependency tree, and the total number of design approaches. The second row shows what design method was used by the designer at the time of recording the trace. The number of design cases that the designer agent has done so far is given in the third row. Design cases differ in the design approach that is used or in the values of the input parameters. Finally, the fourth and the fifth rows show what designers provided the inputs and what designers used the outputs of this designer.

D.2 Project Data File Reading

D.2.1 CurrentProject

RD (**R**obot **D**esigner) reads the name of the current project and the path to the directory that contains the data files of this project from a file named `CurrentProject` (in `~/Java/Thesis/RobotDesignerProjects/`). The name of different data files are constructed by taking the name of the project and adding extensions such as `pref` which is explained in the following section. Having the name of the project and the data files path separated makes it possible to use the same data files for different projects.

```
Block: CURRENT_PROJECT
{
        PROJECT_NAME : default
        project_data_file_directory : /users/cirrus/Java/Thesis/
                                      RobotDesignerProjects/de-
                                      fault/ProjectData/
}
```

D.2.2 default.pref

This file containts the parameters that determine whether the file or GUI reporting is needed or what should be the size of report buffers..

```
# This file contains the preferences for the project

Block: PROJECT_REPORTING
{
        MESSAGE_REPORTING_NEEDED :: false
```

```
        MESSAGE_LOG_FILES_NEEDED :: false
        GUI_MESSAGE_REPORTING_NEEDED :: false
        GUI_SHOWING_NEEDED :: false
}

Block: DESIGN_REQUIREMENTS_SOURCE
{
        READ_DESIGN_REQUIREMENTS_FROM_FILE :: true
}

Block: BUFFERS_SIZE_LIMIT
{
        TEXT_AREA_BYTE_SIZE_LIMIT = 10000
        LOG_FILES_BUFFER_SIZE = 10000
}
```

D.3 Log Files

Each agent in RD generates the following log files, if `MESSAGE_LOG_FILES_NEEDED` attribute in `default.pref` file is set to `true`:

- `ignoredMessagesLogFile`
- `processingReportLogFile`
- `pendingMessagesLogFile`
- `receivedMessagesLogFile`
- `processedMessagesLogFile`
- `sentMessagesLogFile`
- `processingMessagesLogFile`

Also, the following log files are generated by **RD** in order to record the traces of the system:

- `console`: contains the report of the system as sent to the standard output;
- `trace`: contains the summary of the design project trace;

- `DesignersApproach`: contains the list of design approaches taken ineach design cycle;
- `DesignConstraints`: contains the boundaries of the acceptable values for the constraints in each design cycle;
- `DesignParameters`: contains the values of every design parameter in each design cycle;
- `RDSpecific`: contains specific information about, date and time of the run, memory usage, total messages exchanged, total number of design cycles, etc.

D.4 A Sample Project: Project 61

D.4.1 Input Files

D.4.1.1 CurrentProject

```
Block: CURRENT_PROJECT
{
        PROJECT_NAME : default
        project_data_file_directory : /users/cirrus/Java/Thesis/
RobotDesignerProjects/default/ProjectData/
}
```

D.4.1.2 default.pref

```
# This file contains the preferences for the project

Block: PROJECT_REPORTING
{
        MESSAGE_REPORTING_NEEDED :: true
        MESSAGE_LOG_FILES_NEEDED :: true
        GUI_MESSAGE_REPORTING_NEEDED :: false
        GUI_SHOWING_NEEDED :: false
}

Block: DESIGN_REQUIREMENTS_SOURCE
{
        READ_DESIGN_REQUIREMENTS_FROM_FILE :: true
```

```
}

Block: BUFFERS_SIZE_LIMIT
{
        TEXT_AREA_BYTE_SIZE_LIMIT = 10000
        LOG_FILES_BUFFER_SIZE = 10000
}
```

D.4.1.3 default.requirements

```
# This file contains the requirements for the project

Block: DESIGN_REQUIREMENTS
{
       workspace =[[ 0.5 0.75 0.75 0.75 1.0 1.0 1.25 1.25 1.25 1.5
1.5 1.75 1.75 2.0 ] [ 0.5 0.25 0.75
 1.75 1.0 1.5 0.5 1.25 2.0 0.75 1.5 1.0 1.75 1.5 ] ]
        workload = 1.0
        settling_time = 3.0
        maximum_overshoot = 50
}

Block: PROJECT_ID
{
        project_id = 61
}
```

D.4.1.4 default.constraints

```
# This file contains the constraints boundaries or parametrs

Block: DESIGN_CONSTRAINTS_DATA
{
        minLink1Length = 0.0
        maxLink1LengthToWorkspaceLengthRatio = 1.0
        minLink2Length = 0.0
        maxLink2LengthToLink1LengthRatio = 1.0

        minTheta1Min = -3.141592653589793
        maxTheta1Max = 3.141592653589793
        minTheta2Min = -3.141592653589793
        maxTheta2Max = 3.141592653589793

        minLink1Dimension = 0.0
        maxLink1DimensionToLink1LengthRatio = 0.1
        minLink2Dimension = 0.0
        maxLink2DimensionToLink2LengthRatio = 0.1

        minLink1ThicknessToLink1DimensionRatio = 0.05
```

```
        maxLink1ThicknessToLink1DimensionRatio = 0.25
        minLink2ThicknessToLink2DimensionRatio = 0.05
        maxLink2ThicknessToLink2DimensionRatio = 0.25

        minAccessibleRegionArea = 0.0
        maxAccessibleRegionAreaToWorkspaceAreaRatio = 1.0

        minTipDeflection = 0.0
        maxTipDeflectionToLinkLengthsSumRatio = 0.01

        minProportionalGain1 = 0.0
        maxProportionalGain1 = 1000
        minDerivativeGain1 = 0.0
        maxDerivativeGain1 = 1000
minProportionalGain2 = 0.0
        maxProportionalGain2 = 1000
        minDerivativeGain2 = 0.0
        maxDerivativeGain2 = 1000
}
```

D.4.2 Output Files

D.4.2.1 console

```
Block: FILE_ATTRIBUTES
      path_name : /export/home/shakeri/Java/Thesis/RobotDesigner-
Projects/default/ProjectData/
      file_name : default.constraints
Block: DESIGN_CONSTRAINTS_DATA
      maxAccessibleRegionAreaToWorkspaceAreaRatio = 1.0
      minProportionalGain2 = 0.0
      minProportionalGain1 = 0.0
      maxLink2LengthToLink1LengthRatio = 1.0
      maxTheta1Max = 3.141592653589793
      minAccessibleRegionArea = 0.0
      maxLink2DimensionToLink2LengthRatio = 0.1
      maxLink1DimensionToLink1LengthRatio = 0.1
      minLink1Length = 0.0
      minDerivativeGain2 = 0.0
      minDerivativeGain1 = 0.0
      maxDerivativeGain2 = 1000.0
      maxDerivativeGain1 = 1000.0
      minTipDeflection = 0.0
      minTheta2Min = -3.141592653589793
      maxLink2ThicknessToLink2DimensionRatio = 0.25
      minLink2ThicknessToLink2DimensionRatio = 0.05
      maxLink1ThicknessToLink1DimensionRatio = 0.25
      minLink1ThicknessToLink1DimensionRatio = 0.05
      maxProportionalGain2 = 1000.0
```

```
      maxProportionalGain1 = 1000.0
      minLink2Dimension = 0.0
      minLink1Dimension = 0.0
      maxTheta2Max = 3.141592653589793
      minLink2Length = 0.0
      minTheta1Min = -3.141592653589793
      maxLink1LengthToWorkspaceLengthRatio = 1.0
      maxTipDeflectionToLinkLengthsSumRatio = 0.01

Block: DESIGN_REQUIREMENTS
      maximum_overshoot = 50.0
      workspace = [ [ 0.5 0.75 0.75 0.75 1.0 1.0 1.25 1.25 1.25 1.5
1.5 1.75 1.75 2.0 ] [ 0.5 0.25 0.75 1.75 1.0 1.5 0.5 1.25 2.0 0.75
1.5 1.0 1.75 1.5 ] ]
      settling_time = 3.0
      workload = 1.0
Block: FILE_ATTRIBUTES
      path_name : /export/home/shakeri/Java/Thesis/RobotDesigner-
Projects/default/ProjectData/
      file_name : default.requirements
Block: PROJECT_ID
      project_id = 61.0

new design state group was added at: 0
>> design state: 0 with parent ID: -1 at depth: 0 was created
new design state group was added at: 1
>> design state: 1 with parent ID: 0 at depth: 1 was created
new design state group was added at: 2
>> design state: 2 with parent ID: 1 at depth: 2 was created
new design state group was added at: 3
>> design state: 3 with parent ID: 2 at depth: 3 was created

  - index of rejected path: 0

  - unresolved constraints are:
   constraint constraint_1_4_1 of type numeric_continuous_b<x<=c:
0.0 < accessible_region_area (2.820110874804171) <= 2.625
   constraint constraint_3_1_1 of type numeric_continuous_b<x<=c:
0.0 < proportional_gain1 (1375.245122855074) <= 1000.0
>>>>creating new backtracking session because the collection is
empty
>> design state: 4 with parent ID: 1 at depth: 2 was created
>> design state: 5 with parent ID: 4 at depth: 3 was created

  - index of rejected path: 1

  - unresolved constraints are:
   constraint constraint_1_4_1 of type numeric_continuous_b<x<=c:
0.0 < accessible_region_area (2.820110874804171) <= 2.625
```

```
>>>>even though current session is not null, creating new back-
tracking session because the collection does not match the violated
constraints
>> design state: 6 with parent ID: 1 at depth: 2 was created
>> design state: 7 with parent ID: 6 at depth: 3 was created

  - index of rejected path: 5

  - unresolved constraints are:
   constraint constraint_1_4_1 of type numeric_continuous_b<x<=c:
0.0 < accessible_region_area (2.9909553613956126) <= 2.625
>> design state: 8 with parent ID: 0 at depth: 1 was created
>> design state: 9 with parent ID: 8 at depth: 2 was created
>> design state: 10 with parent ID: 9 at depth: 3 was created

  - index of rejected path: 9

  - unresolved constraints are:
   constraint constraint_1_4_1 of type numeric_continuous_b<x<=c:
0.0 < accessible_region_area (2.805439246655777) <= 2.625
>> design state: 11 with parent ID: 8 at depth: 2 was created
>> design state: 12 with parent ID: 11 at depth: 3 was created

  - index of rejected path: 13

  - unresolved constraints are:
   constraint constraint_1_4_1 of type numeric_continuous_b<x<=c:
0.0 < accessible_region_area (3.614846500459787) <= 2.625
>> design state: 13 with parent ID: 0 at depth: 1 was created
>> design state: 14 with parent ID: 13 at depth: 2 was created
>> design state: 15 with parent ID: 14 at depth: 3 was created

  - index of rejected path: 17

  - unresolved constraints are:
   constraint constraint_1_4_1 of type numeric_continuous_b<x<=c:
0.0 < accessible_region_area (3.0194741798453535) <= 2.625
>> design state: 16 with parent ID: 13 at depth: 2 was created
>> design state: 17 with parent ID: 16 at depth: 3 was created

  - index of rejected path: 21

  - unresolved constraints are:
   constraint constraint_1_4_1 of type numeric_continuous_b<x<=c:
0.0 < accessible_region_area (4.225362012380889) <= 2.625
>> design state: 18 with parent ID: -1 at depth: 0 was created
>> design state: 19 with parent ID: 18 at depth: 1 was created
>> design state: 20 with parent ID: 19 at depth: 2 was created
>> design state: 21 with parent ID: 20 at depth: 3 was created
```

```
  - index of rejected path: 385

  - unresolved constraints are:
   constraint constraint_1_4_1 of type numeric_continuous_b<x<=c:
0.0 < accessible_region_area (3.2318251863236065) <= 2.625
   constraint constraint_3_1_1 of type numeric_continuous_b<x<=c:
0.0 < proportional_gain1 (1301.9890850434604) <= 1000.0
>> design state: 22 with parent ID: -1 at depth: 0 was created
>> design state: 23 with parent ID: 22 at depth: 1 was created
>> design state: 24 with parent ID: 23 at depth: 2 was created
>> design state: 25 with parent ID: 24 at depth: 3 was created

  - index of rejected path: 2

  - unresolved constraints are:
   constraint constraint_1_4_1 of type numeric_continuous_b<x<=c:
0.0 < accessible_region_area (2.820110874804171) <= 2.625
>> design state: 26 with parent ID: -1 at depth: 0 was created
>> design state: 27 with parent ID: 26 at depth: 1 was created
>> design state: 28 with parent ID: 27 at depth: 2 was created
>> design state: 29 with parent ID: 28 at depth: 3 was created

  - index of rejected path: 390

  - unresolved constraints are:
   constraint constraint_1_4_1 of type numeric_continuous_b<x<=c:
0.0 < accessible_region_area (2.8756172456197544) <= 2.625
>> design state: 30 with parent ID: 26 at depth: 1 was created
>> design state: 31 with parent ID: 30 at depth: 2 was created
>> design state: 32 with parent ID: 31 at depth: 3 was created

  - index of rejected path: 394

  - unresolved constraints are:
   constraint constraint_1_4_1 of type numeric_continuous_b<x<=c:
0.0 < accessible_region_area (4.090869074405497) <= 2.625
>> design state: 33 with parent ID: 30 at depth: 2 was created
>> design state: 34 with parent ID: 33 at depth: 3 was created

  - index of rejected path: 398

  - unresolved constraints are:
   constraint constraint_1_4_1 of type numeric_continuous_b<x<=c:
0.0 < accessible_region_area (2.9563297381314446) <= 2.625
>> design state: 35 with parent ID: 26 at depth: 1 was created
>> design state: 36 with parent ID: 35 at depth: 2 was created
>> design state: 37 with parent ID: 36 at depth: 3 was created

  - index of rejected path: 402
```

```
  - unresolved constraints are:
   constraint constraint_1_4_1 of type numeric_continuous_b<x<=c:
0.0 < accessible_region_area (4.981729006637212) <= 2.625
>> design state: 38 with parent ID: 35 at depth: 2 was created
>> design state: 39 with parent ID: 38 at depth: 3 was created

  - index of rejected path: 406

  - unresolved constraints are:
   constraint constraint_1_4_1 of type numeric_continuous_b<x<=c:
0.0 < accessible_region_area (3.4933335588362864) <= 2.625
>> design state: 40 with parent ID: -1 at depth: 0 was created
>> design state: 41 with parent ID: 40 at depth: 1 was created
>> design state: 42 with parent ID: 41 at depth: 2 was created
>> design state: 43 with parent ID: 42 at depth: 3 was created
> terminating the program--the reason is success in the design pro-
cess
clean up at: DatabaseCoordinator completed
clean up at: DesignRequirements completed
clean up at: DesignState completed
clean up at: DesignProduct completed
clean up at: DesignConstraints completed
clean up at: DesignersCoordinator completed
clean up at: Designer_1_1 completed
clean up at: Designer_1_2 completed
clean up at: Designer_1_3 completed
clean up at: Designer_1_4 completed
clean up at: Designer_2_1 completed
clean up at: Designer_2_2 completed
clean up at: Designer_2_3 completed
clean up at: Designer_2_4 completed
clean up at: Designer_2_5 completed
clean up at: Designer_3_1 completed
clean up at: Evaluator completed
clean up at: DependencyProvider completed
total number of active threads: 4
waiting for thread: Processor_2896 to joine
thread: Processor_2896 joined
waiting for thread: Processor_2897 to joine
clean up at: ExceptionHandler completed
thread: Processor_2897 joined
waiting for thread: Processor_2898 to joine
clean up at: Tracer completed
thread: Processor_2898 joined
clean up at: Coordinator completed
current thread: Processor_2837 is about to stop too
> wrap up completed--the program will stop
```

D.4.2.2 trace

```
DESIGN REQUIREMENTS
{
      Parameter: operational_plane, Value: horizontal, Owner:
Agent: DesignRequirements, ID: 2
      Parameter: workspace, Value: [ 0.5, 0.75, 0.75, 0.75, 1.0,
1.0, 1.25, 1.25, 1.25, 1.5, 1.5, 1.75, 1.75, 2.0,  ] [ 0.5, 0.25,
0.75, 1.75, 1.0, 1.5, 0.5, 1.25, 2.0, 0.75, 1.5, 1.0, 1.75, 1.5,
] , Owner: Agent: DesignRequirements, ID: 2
      Parameter: workload, Value: 1.0, Owner: Agent: DesignRequire-
ments, ID: 2
      Parameter: settling_time, Value: 3.0, Owner: Agent: DesignRe-
quirements, ID: 2
      Parameter: maximum_overshoot, Value: 50.0, Owner: Agent:
DesignRequirements, ID: 2
}


DESIGN CONSTRAINTS
{
      constraint constraint_1_2_1 of type
numeric_continuous_b<x<=c: 0.0 < link1_length (1.5567951410224505)
<= 1.75
      constraint constraint_1_2_2 of type
numeric_continuous_b<x<=c: 0.0 < link2_length (0.7783975705112253)
<= 1.5567951410224505
      constraint constraint_1_3_1 of type
numeric_continuous_b<=x<c: -3.141592653589793 <= theta1_min (-
2.4580611917887825) < -1.370467760770914
      constraint constraint_1_3_2 of type
numeric_continuous_b<x<=c: -2.4580611917887825 < theta1_max (-
1.370467760770914) <= 3.141592653589793
      constraint constraint_1_3_3 of type
numeric_continuous_b<=x<c: -3.141592653589793 <= theta2_min (0.0)
< 2.8573894984143893
      constraint constraint_1_3_4 of type
numeric_continuous_b<x<=c: 0.0 < theta2_max (2.8573894984143893)
<= 3.141592653589793
      constraint constraint_2_1_1 of type
numeric_continuous_b<x<=c: 0.0 < link1_cross_section_dimension
(0.05071617460646335) <= 0.15567951410224506
      constraint constraint_2_1_2 of type
numeric_continuous_b<x<=c: 0.0 < link2_cross_section_dimension
(0.024905129862261795) <= 0.07783975705112253
      constraint constraint_2_1_3 of type
numeric_continuous_b<=x<=c: 0.0025358087303231675 <=
link1_cross_section_thickness (0.005071617460646335) <=
0.012679043651615837
```

constraint constraint_2_1_4 of type numeric_continuous_b<=x<=c: 0.0012452564931130898 <= link2_cross_section_thickness (0.0024905129862261796) <= 0.006226282465565449
constraint constraint_1_4_1 of type numeric_continuous_b<x<=c: 0.0 < accessible_region_area (2.5830343986674387) <= 2.625
constraint constraint_2_5_1 of type numeric_continuous_b<=x<=c: 0.0 <= tip_deflection (0.009120385595540107) <= 0.02335192711533676
constraint constraint_3_1_1 of type numeric_continuous_b<x<=c: 0.0 < proportional_gain1 (362.90360496525614) <= 1000.0
constraint constraint_3_1_2 of type numeric_continuous_b<x<=c: 0.0 < derivative_gain1 (33.69220551656433) <= 1000.0
constraint constraint_3_1_3 of type numeric_continuous_b<x<=c: 0.0 < proportional_gain2 (17.756563493544025) <= 1000.0
constraint constraint_3_1_4 of type numeric_continuous_b<x<=c: 0.0 < derivative_gain2 (1.648530844849792) <= 1000.0
}

DESIGNER TRACES
{
Designer_1_1: depth: 0, # of approaches: 6, current approach: 2 (base_at_right_above_midway_workspace_length), # of design cases: 5, # of suppliers: 1 (DesignRequirements,) , # of consumers: 2 (Designer_1_2, Designer_1_3,)
Designer_1_2: depth: 1, # of approaches: 3, current approach: 0 (link_lengths_ratio_0.5), # of design cases: 9, # of suppliers: 2 (DesignRequirements, Designer_1_1,) , # of consumers: 5 (Designer_1_3, Designer_2_1, Designer_1_4, Designer_2_5, Designer_3_1,)
Designer_1_3: depth: 2, # of approaches: 2, current approach: 0 (theta1_is_alpha1_minus_alpha2), # of design cases: 15, # of suppliers: 3 (DesignRequirements, Designer_1_1, Designer_1_2,) , # of consumers: 1 (Designer_1_4,)
Designer_1_4: depth: 3, # of approaches: 1, current approach: 0 (default), # of design cases: 15, # of suppliers: 2 (Designer_1_2, Designer_1_3,) , # of consumers: 0
Designer_2_1: depth: 2, # of approaches: 4, current approach: 2 (dimention_min_ratio_2), # of design cases: 15, # of suppliers: 5 (Designer_1_2, Designer_2_4, Designer_2_2, DesignRequirements, Designer_2_3,) , # of consumers: 2 (Designer_2_5, Designer_3_1,)
Designer_2_2: depth: 0, # of approaches: 2, current approach: 0 (steel_stainless_AISI_302_annealed), # of design cases: 5, # of

suppliers: 0, # of consumers: 3 (Designer_2_1, Designer_2_5, Designer_3_1,)

Designer_2_3: depth: 0, # of approaches: 4, current approach: 0 (safety_factor_3), # of design cases: 5, # of suppliers: 0, # of consumers: 1 (Designer_2_1,)

Designer_2_4: depth: 0, # of approaches: 2, current approach: 0 (hollow_round), # of design cases: 5, # of suppliers: 0, # of consumers: 3 (Designer_2_1, Designer_2_5, Designer_3_1,)

Designer_2_5: depth: 3, # of approaches: 1, current approach: 0 (default), # of design cases: 15, # of suppliers: 5 (Designer_1_2, Designer_2_4, Designer_2_2, DesignRequirements, Designer_2_1,) , # of consumers: 0

Designer_3_1: depth: 3, # of approaches: 1, current approach: 0 (default), # of design cases: 15, # of suppliers: 5 (Designer_1_2, Designer_2_4, Designer_2_2, DesignRequirements, Designer_2_1,) , # of consumers: 0

}

DESIGN PRODUCT

{

Parameter: link1_length, Value: 1.5567951410224505, Owner: Agent: Designer_1_2, ID: 8

Parameter: link2_length, Value: 0.7783975705112253, Owner: Agent: Designer_1_2, ID: 8

Parameter: link1_cross_section_dimension, Value: 0.05071617460646335, Owner: Agent: Designer_2_1, ID: 11

Parameter: link2_cross_section_dimension, Value: 0.024905129862261795, Owner: Agent: Designer_2_1, ID: 11

Parameter: link1_cross_section_thickness, Value: 0.005071617460646335, Owner: Agent: Designer_2_1, ID: 11

Parameter: link2_cross_section_thickness, Value: 0.0024905129862261796, Owner: Agent: Designer_2_1, ID: 11

Parameter: proportional_gain1, Value: 362.90360496525614, Owner: Agent: Designer_3_1, ID: 16

Parameter: derivative_gain1, Value: 33.69220551656433, Owner: Agent: Designer_3_1, ID: 16

Parameter: proportional_gain2, Value: 17.756563493544025, Owner: Agent: Designer_3_1, ID: 16

Parameter: derivative_gain2, Value: 1.648530844849792, Owner: Agent: Designer_3_1, ID: 16

Parameter: workspace, Value: [0.5, 0.75, 0.75, 0.75, 1.0, 1.0, 1.25, 1.25, 1.25, 1.5, 1.5, 1.75, 1.75, 2.0,] [0.5, 0.25, 0.75, 1.75, 1.0, 1.5, 0.5, 1.25, 2.0, 0.75, 1.5, 1.0, 1.75, 1.5,] , Owner: Agent: DesignRequirements, ID: 2

Parameter: accessible_region_area, Value: 2.5830343986674387, Owner: Agent: Designer_1_4, ID: 10

Parameter: base_location, Value: [2.75, 1.125,] , Owner: Agent: Designer_1_1, ID: 7

Parameter: theta1_array, Value: [-1.841743177133317, -2.234645941276905, -2.1030228587442052, -1.5803360074509354, -2.101691900266075, -1.808417716248621, -2.4580611917887825, -2.000640459862265, -1.5075116724748197, -2.3849331973654104, -1.8020196084096765, -2.1393342274091958, -1.5200267188657943, -1.370467760770914,] , Owner: Agent: Designer_1_3, ID: 9

Parameter: theta2_array, Value: [0.0, 0.7722645770738099, 1.094542932510318, 0.9744725044399294, 1.5507376751335993, 1.4991016848920589, 1.7319514332910713, 1.8914492237267633, 1.576527017187388, 2.1499047363494395, 2.1499047363494395, 2.5516014242815754, 2.313412773511665, 2.8573894984143893,] , Owner: Agent: Designer_1_3, ID: 9

Parameter: theta1_min, Value: -2.4580611917887825, Owner: Agent: Designer_1_3, ID: 9

Parameter: theta1_max, Value: -1.370467760770914, Owner: Agent: Designer_1_3, ID: 9

Parameter: theta2_min, Value: 0.0, Owner: Agent: Designer_1_3, ID: 9

Parameter: theta2_max, Value: 2.8573894984143893, Owner: Agent: Designer_1_3, ID: 9

Parameter: material_name, Value: steel_stainless_AISI_302_annealed, Owner: Agent: Designer_2_2, ID: 12

Parameter: material_mass_density, Value: 7920.0, Owner: Agent: Designer_2_2, ID: 12

Parameter: material_yield_stress, Value: 2.6E8, Owner: Agent: Designer_2_2, ID: 12

Parameter: material_elasticity_modulus, Value: 1.9E11, Owner: Agent: Designer_2_2, ID: 12

Parameter: workload, Value: 1.0, Owner: Agent: DesignRequirements, ID: 2

Parameter: tip_deflection, Value: 0.009120385595540107, Owner: Agent: Designer_2_5, ID: 15

Parameter: structural_safety_factor, Value: 3.0, Owner: Agent: Designer_2_3, ID: 13

Parameter: settling_time, Value: 3.0, Owner: Agent: DesignRequirements, ID: 2

Parameter: maximum_overshoot, Value: 50.0, Owner: Agent: DesignRequirements, ID: 2

}

D.4.2.3 DesignersApproach

3 3 Designer_1_1: 0 Designer_1_2: 0 Designer_1_3: 0 Designer_1_4: 0 Designer_2_1: 0 Designer_2_2: 0 Designer_2_3: 0 Designer_2_4: 0 Designer_2_5: 0 Designer_3_1: 0 unsuccessful

```
5 3 Designer_1_1: 0 Designer_1_2: 0 Designer_1_3: 0 Designer_1_4:
0 Designer_2_1: 1 Designer_2_2: 0 Designer_2_3: 0 Designer_2_4: 0
Designer_2_5: 0 Designer_3_1: 0 unsuccessful
7 3 Designer_1_1: 0 Designer_1_2: 0 Designer_1_3: 1 Designer_1_4:
0 Designer_2_1: 1 Designer_2_2: 0 Designer_2_3: 0 Designer_2_4: 0
Designer_2_5: 0 Designer_3_1: 0 unsuccessful
10 3 Designer_1_1: 0 Designer_1_2: 1 Designer_1_3: 0 Designer_1_4:
0 Designer_2_1: 1 Designer_2_2: 0 Designer_2_3: 0 Designer_2_4: 0
Designer_2_5: 0 Designer_3_1: 0 unsuccessful
12 3 Designer_1_1: 0 Designer_1_2: 1 Designer_1_3: 1 Designer_1_4:
0 Designer_2_1: 1 Designer_2_2: 0 Designer_2_3: 0 Designer_2_4: 0
Designer_2_5: 0 Designer_3_1: 0 unsuccessful
15 3 Designer_1_1: 0 Designer_1_2: 2 Designer_1_3: 0 Designer_1_4:
0 Designer_2_1: 1 Designer_2_2: 0 Designer_2_3: 0 Designer_2_4: 0
Designer_2_5: 0 Designer_3_1: 0 unsuccessful
17 3 Designer_1_1: 0 Designer_1_2: 2 Designer_1_3: 1 Designer_1_4:
0 Designer_2_1: 1 Designer_2_2: 0 Designer_2_3: 0 Designer_2_4: 0
Designer_2_5: 0 Designer_3_1: 0 unsuccessful
21 3 Designer_1_1: 1 Designer_1_2: 0 Designer_1_3: 0 Designer_1_4:
0 Designer_2_1: 1 Designer_2_2: 0 Designer_2_3: 0 Designer_2_4: 0
Designer_2_5: 0 Designer_3_1: 0 unsuccessful
25 3 Designer_1_1: 0 Designer_1_2: 0 Designer_1_3: 0 Designer_1_4:
0 Designer_2_1: 2 Designer_2_2: 0 Designer_2_3: 0 Designer_2_4: 0
Designer_2_5: 0 Designer_3_1: 0 unsuccessful
29 3 Designer_1_1: 1 Designer_1_2: 0 Designer_1_3: 1 Designer_1_4:
0 Designer_2_1: 2 Designer_2_2: 0 Designer_2_3: 0 Designer_2_4: 0
Designer_2_5: 0 Designer_3_1: 0 unsuccessful
32 3 Designer_1_1: 1 Designer_1_2: 1 Designer_1_3: 0 Designer_1_4:
0 Designer_2_1: 2 Designer_2_2: 0 Designer_2_3: 0 Designer_2_4: 0
Designer_2_5: 0 Designer_3_1: 0 unsuccessful
34 3 Designer_1_1: 1 Designer_1_2: 1 Designer_1_3: 1 Designer_1_4:
0 Designer_2_1: 2 Designer_2_2: 0 Designer_2_3: 0 Designer_2_4: 0
Designer_2_5: 0 Designer_3_1: 0 unsuccessful
37 3 Designer_1_1: 1 Designer_1_2: 2 Designer_1_3: 0 Designer_1_4:
0 Designer_2_1: 2 Designer_2_2: 0 Designer_2_3: 0 Designer_2_4: 0
Designer_2_5: 0 Designer_3_1: 0 unsuccessful
39 3 Designer_1_1: 1 Designer_1_2: 2 Designer_1_3: 1 Designer_1_4:
0 Designer_2_1: 2 Designer_2_2: 0 Designer_2_3: 0 Designer_2_4: 0
Designer_2_5: 0 Designer_3_1: 0 unsuccessful
43 3 Designer_1_1: 2 Designer_1_2: 0 Designer_1_3: 0 Designer_1_4:
0 Designer_2_1: 2 Designer_2_2: 0 Designer_2_3: 0 Designer_2_4: 0
Designer_2_5: 0 Designer_3_1: 0 successful
```

D.4.2.4 DesignConstraints

`DesignConstraints` file contains the boundaries and the value for the argument of each active constraints as they change during the design process. The report includes any design cycle in which the design has not been successful—that is, a constraint violation has happened. If the design process succeeds in finding a satisfactory design, the resulted values for the argument and the boundaries of the constraints are reported in the `Design-Constraints` file as the last line.The two fields at the beginning of each report for a design cycle are the design cycle ID and the depth of the corresponding design state in the dependency graph. The rest of the fields include the name of the design parameter and the value assigned. Tow adjacent fields are separeted from each other by a semicolon ';'.

```
3 ; 3 ; 0.0 < link1_length (1.5206906325745548) <= 1.75 ; 0.0 < link2_length (0.7603453162872774) <= 1.5206906325745548 ; -3.141592653589793 <= theta1_min (0.4992720772603446) < 1.8263830771681904 ; 0.4992720772603446 < theta1_max (1.8263830771681904) <= 3.141592653589793 ; -3.141592653589793 <= theta2_min (0.0) < 2.564106837681789 ; 0.0 < theta2_max (2.564106837681789) <= 3.141592653589793 ; 0.0 < link1_cross_section_dimension (0.12873882479062435) <= 0.1520690632574555 ; 0.0 < link2_cross_section_dimension (0.049265974918586156) <= 0.07603453162872775 ; 0.006436941239531218 <= link1_cross_section_thickness (0.012873882479062435) <= 0.032184706197656086 ; 0.002463298745929308 <= link2_cross_section_thickness (0.004926597491858616) <= 0.012316493729646539 ; 0.0 < accessible_region_area (2.820110874804171) <= 2.625 ; 0.0 <= tip_deflection (0.0011051471277170148) <= 0.022810359488618325 ; 0.0 < proportional_gain1 (1375.245122855074) <= 1000.0 ; 0.0 < derivative_gain1 (127.67864711435413) <= 1000.0 ; 0.0 < proportional_gain2 (29.61009191218797) <= 1000.0 ; 0.0 < derivative_gain2 (2.749020093546075) <= 1000.0 ; unsuccessful
5 ; 3 ; 0 .0 < link1_length (1.5206906325745548) <= 1.75 ; 0.0 < link2_length (0.7603453162872774) <= 1.5206906325745548 ; -3.141592653589793 <= theta1_min (0.4992720772603446) < 1.8263830771681904 ; 0.4992720772603446 < theta1_max (1.8263830771681904) <= 3.141592653589793 ; -3.141592653589793 <= theta2_min (0.0) < 2.564106837681789 ; 0.0 < theta2_max (2.564106837681789) <= 3.141592653589793 ; 0.0 < link1_cross_section_dimension (0.08529699024103615) <= 0.1520690632574555 ; 0.0 < link2_cross_section_dimension (0.03694948118893962) <= 0.07603453162872775 ; 0.0042648495120518074 <= link1_cross_section_thickness (0.008529699024103615) <= 0.021324247560259038 ; 0.0018474740594469812 <= link2_cross_section_thickness (0.0036949481188939624) <= 0.009237370297234905 ; 0.0 < accessible_region_area (2.820110874804171) <= 2.625 ; 0.0 <= tip_deflection (0.002602398618321901) <= 0.022810359488618325 ; 0.0 < proportional_gain1 (706.8546566810312) <= 1000.0 ; 0.0 < derivative_gain1 (65.62484372542369) <= 1000.0 ; 0.0 < proportional_gain2 (22.104360987620353) <= 1000.0 ; 0.0 < derivative_gain2 (2.0521831776196673) <= 1000.0 ; unsuccessful
7 ; 3 ; 0.0 < link1_length (1.5206906325745548) <= 1.75 ; 0.0 < link2_length (0.7603453162872774) <= 1.5206906325745548 ; -3.141592653589793 <= theta1_min (1.4056476493802699) < 2.8131560547429113 ; 1.4056476493802699 < theta1_max (2.8131560547429113) <= 3.141592653589793 ; -3.141592653589793 <= theta2_min (-2.564106837681789) < 0.0 ; -2.564106837681789 < theta2_max (0.0) <= 3.141592653589793 ; 0.0 < link1_cross_section_dimension (0.08529699024103615) <= 0.1520690632574555 ; 0.0 < link2_cross_section_dimension (0.03694948118893962) <= 0.07603453162872775 ; 0.0042648495120518074 <= link1_cross_section_thickness (0.008529699024103615) <= 0.021324247560259038 ; 0.0018474740594469812 <= link2_cross_section_thickness (0.0036949481188939624) <= 0.009237370297234905 ;
```

0.0 < accessible_region_area (2.9909553613956126) <= 2.625 ; 0.0 <= tip_deflection (0.002602398618321901) <= 0.022810359488618325 ; 0.0 < proportional_gain1 (706.8546566810312) <= 1000.0 ; 0.0 < derivative_gain1 (65.62484372542369) <= 1000.0 ; 0.0 < proportional_gain2 (22.104360987620353) <= 1000.0 ; 0.0 < derivative_gain2 (2.0521831776196673) <= 1000.0 ; unsuccessful

10 ; 3 ; 0.0 < link1_length (1.3034491136353328) <= 1.75 ; 0.0 < link2_length (0.9775868352264996) <= 1.3034491136353328 ; -3.141592653589793 <= theta1_min (0.20863343229332243) < 1.5288401366019233 ; 0.20863343229332243 < theta1_max (1.5288401366019233) <= 3.141592653589793 ; -3.141592653589793 <= theta2_min (0.0) < 2.3018677747641307 ; 0.0 < theta2_max (2.3018677747641307) <= 3.141592653589793 ; 0.0 < link1_cross_section_dimension (0.09164457471547738) <= 0.1303449113635333 ; 0.0 < link2_cross_section_dimension (0.04189922992805236) <= 0.09775868352264996 ; 0.004582228735773869 <= link1_cross_section_thickness (0.009164457471547737) <= 0.022911143678869345 ; 0.002094961496402618 <= link2_cross_section_thickness (0.004189922992805236) <= 0.01047480748201309 ; 0.0 < accessible_region_area (2.805439246655777) <= 2.625 ; 0.0 <= tip_deflection (0.002288904138117323) <= 0.022810359488618325 ; 0.0 < proportional_gain1 (683.6900561763787) <= 1000.0 ; 0.0 < derivative_gain1 (63.47422722497149) <= 1000.0 ; 0.0 < proportional_gain2 (46.96085698845915) <= 1000.0 ; 0.0 < derivative_gain2 (4.359876350747827) <= 1000.0 ; unsuccessful

12 ; 3 ; 0.0 < link1_length (1.3034491136353328) <= 1.75 ; 0.0 < link2_length (0.9775868352264996) <= 1.3034491136353328 ; -3.141592653589793 <= theta1_min (1.4056476493802699) < 3.106751884890758 ; 1.4056476493802699 < theta1_max (3.106751884890758) <= 3.141592653589793 ; -3.141592653589793 <= theta2_min (-2.3018677747641307) < 0.0 ; -2.3018677747641307 < theta2_max (0.0) <= 3.141592653589793 ; 0.0 < link1_cross_section_dimension (0.09164457471547738) <= 0.1303449113635333 ; 0.0 < link2_cross_section_dimension (0.04189922992805236) <= 0.09775868352264996 ; 0.004582228735773869 <= link1_cross_section_thickness (0.009164457471547737) <= 0.022911143678869345 ; 0.002094961496402618 <= link2_cross_section_thickness (0.004189922992805236) <= 0.01047480748201309 ; 0.0 < accessible_region_area (3.614846500459787) <= 2.625 ; 0.0 <= tip_deflection (0.002288904138117323) <= 0.022810359488618325 ; 0.0 < proportional_gain1 (683.6900561763787) <= 1000.0 ; 0.0 < derivative_gain1 (63.47422722497149) <= 1000.0 ; 0.0 < proportional_gain2 (46.96085698845915) <= 1000.0 ; 0.0 < derivative_gain2 (4.359876350747827) <= 1000.0 ; unsuccessful

15 ; 3 ; 0.0 < link1_length (1.1405179744309162) <= 1.75 ; 0.0 < link2_length (1.1405179744309162) <= 1.1405179744309162 ; -3.141592653589793 <= theta1_min (-0.015281376429308269) < 1.4056476493802699 ; -0.015281376429308269 < theta1_max (1.4056476493802699) <= 3.141592653589793 ; -3.141592653589793 <= theta2_min (0.0) < 2.257037398547587 ; 0.0 < theta2_max (2.257037398547587) <= 3.141592653589793 ; 0.0 < link1_cross_section_dimension (0.09746653452047076) <= 0.11405179744309163 ; 0.0 < link2_cross_section_dimension (0.04577063421833817) <= 0.11405179744309163 ; 0.004873326726023538 <= link1_cross_section_thickness (0.009746653452047076) <= 0.02436663363011769 ; 0.0022885317109169086 <= link2_cross_section_thickness (0.004577063421833817) <= 0.011442658554584543 ; 0.0 < accessible_region_area (3.0194741798453535) <= 2.625 ; 0.0 <= tip_deflection (0.0023009346396080143) <= 0.022810359488618325 ; 0.0 < proportional_gain1 (688.5486322528478) <= 1000.0 ; 0.0 < derivative_gain1 (63.925300571850855) <= 1000.0 ; 0.0 < proportional_gain2 (77.9986007314333) <= 1000.0 ; 0.0 < derivative_gain2 (7.2414405640844794) <= 1000.0 ; unsuccessful

17 ; 3 0; .0 < link1_length (1.1405179744309162) <= 1.75 ; 0.0 < link2_length (1.1405179744309162) <= 1.1405179744309162 ; -3.141592653589793 <= theta1_min (-3.0445985812993093) < 3.0131010733886336 ; -3.0445985812993093 < theta1_max (3.0131010733886336) <= 3.141592653589793 ; -3.141592653589793 <= theta2_min (-2.257037398547587) < 0.0 ; -2.257037398547587 < theta2_max (0.0) <= 3.141592653589793 ; 0.0 < link1_cross_section_dimension (0.09746653452047076) <= 0.11405179744309163 ; 0.0 < link2_cross_section_dimension (0.04577063421833817) <= 0.11405179744309163 ; 0.004873326726023538 <= link1_cross_section_thickness

(0.009746653452047076) <= 0.02436663363011769 ; 0.0022885317109169086 <= link2_cross_section_thickness (0.004577063421833817) <= 0.011442658554584543 ; 0.0 < accessible_region_area (4.225362012380889) <= 2.625 ; 0.0 <= tip_deflection (0.0023009346396080143) <= 0.022810359488618325 ; 0.0 < proportional_gain1 (688.5486322528478) <= 1000.0 ; 0.0 < derivative_gain1 (63.925300571850855) <= 1000.0 ; 0.0 < proportional_gain2 (77.9986007314333) <= 1000.0 ; 0.0 < derivative_gain2 (7.2414405640844794) <= 1000.0 ; unsuccessful

21 ; 3 ; 0.0 < link1_length (1.75) <= 1.75 ; 0.0 < link2_length (0.875) <= 1.75 ; -3.141592653589793 <= theta1_min (-1.100195808110164) < 0.0 ; -1.100195808110164 < theta1_max (0.0) <= 3.141592653589793 ; -3.141592653589793 <= theta2_min (0.0) < 2.734731173047627 ; 0.0 < theta2_max (2.734731173047627) <= 3.141592653589793 ; 0.0 < link1_cross_section_dimension (0.09834270010059333) <= 0.17500000000000002 ; 0.0 < link2_cross_section_dimension (0.039545791119743025) <= 0.08750000000000001 ; 0.004917135005029667 <= link1_cross_section_thickness (0.009834270010059334) <= 0.024585675025148333 ; 0.0019772895559871514 <= link2_cross_section_thickness (0.003954579111974303) <= 0.009886447779935756 ; 0.0 < accessible_region_area (3.2318251863236065) <= 2.625 ; 0.0 <= tip_deflection (0.003367425157030643) <= 0.02625 ; 0.0 < proportional_gain1 (1301.9890850434604) <= 1000.0 ; 0.0 < derivative_gain1 (120.87750916061464) <= 1000.0 ; 0.0 < proportional_gain2 (33.339936976502194) <= 1000.0 ; 0.0 < derivative_gain2 (3.0953013228654953) <= 1000.0 ; unsuccessful

25 ; 3 ; 0.0 < link1_length (1.5206906325745548) <= 1.75 ; 0.0 < link2_length (0.7603453162872774) <= 1.5206906325745548 ; -3.141592653589793 <= theta1_min (0.4992720772603446) < 1.8263830771681904 ; 0.4992720772603446 < theta1_max (1.8263830771681904) <= 3.141592653589793 ; -3.141592653589793 <= theta2_min (0.0) < 2.564106837681789 ; 0.0 < theta2_max (2.564106837681789) <= 3.141592653589793 ; 0.0 < link1_cross_section_dimension (0.04963495688385445) <= 0.1520690632574555 ; 0.0 < link2_cross_section_dimension (0.024632987459293078) <= 0.07603453162872775 ; 0.0024817478441927225 <= link1_cross_section_thickness (0.004963495688385445) <= 0.012408739220963612 ; 0.001231649372964654 <= link2_cross_section_thickness (0.002463298745929308) <= 0.0061582468648232695 ; 0.0 < accessible_region_area (2.820110874804171) <= 2.625 ; 0.0 <= tip_deflection (0.008766542738009385) <= 0.022810359488618325 ; 0.0 < proportional_gain1 (332.8892198101605) <= 1000.0 ; 0.0 < derivative_gain1 (30.905650576732263) <= 1000.0 ; 0.0 < proportional_gain2 (16.743124612929197) <= 1000.0 ; 0.0 < derivative_gain2 (1.5544425233865187) <= 1000.0 ; unsuccessful

29 ; 3 ; 0.0 < link1_length (1.75) <= 1.75 ; 0.0 < link2_length (0.875) <= 1.75 ; -3.141592653589793 <= theta1_min (-0.1684102529634819) < 0.8105232774602644 ; -0.1684102529634819 < theta1_max (0.8105232774602644) <= 3.141592653589793 ; -3.141592653589793 <= theta2_min (-2.734731173047627) < 0.0 ; -2.734731173047627 < theta2_max (0.0) <= 3.141592653589793 ; 0.0 < link1_cross_section_dimension (0.05680396263726892) <= 0.17500000000000002 ; 0.0 < link2_cross_section_dimension (0.02636386074649535) <= 0.08750000000000001 ; 0.002840198131863446 <= link1_cross_section_thickness (0.005680396263726892) <= 0.01420099065931723 ; 0.0013181930373247677 <= link2_cross_section_thickness (0.0026363860746495354) <= 0.006590965186623838 ; 0.0 < accessible_region_area (2.8756172456197544) <= 2.625 ; 0.0 <= tip_deflection (0.01105592332430511) <= 0.02625 ; 0.0 < proportional_gain1 (568.7736692745627) <= 1000.0 ; 0.0 < derivative_gain1 (52.80531550366831) <= 1000.0 ; 0.0 < proportional_gain2 (23.980702322073615) <= 1000.0 ; 0.0 < derivative_gain2 (2.2263839212735537) <= 1000.0 ; unsuccessful

32 ; 3 ; 0.0 < link1_length (1.5) <= 1.75 ; 0.0 < link2_length (1.125) <= 1.5 ; -3.141592653589793 <= theta1_min (-1.3926362806486803) < 0.0 ; -1.3926362806486803 < theta1_max (0.0) <= 3.141592653589793 ; -3.141592653589793 <= theta2_min (0.0) < 2.4049686515706643 ; 0.0 < theta2_max (2.4049686515706643) <= 3.141592653589793 ; 0.0 < link1_cross_section_dimension (0.058845025683956616) <= 0.15000000000000002 ; 0.0 < link2_cross_section_dimension (0.030261681468808716) <= 0.1125 ; 0.002942251284197831 <=

link1_cross_section_thickness (0.005884502568395662) <= 0.014711256420989154 ; 0.0015130840734404359 <= link2_cross_section_thickness (0.0030261681468808717) <= 0.007565420367202179 ; 0.0 < accessible_region_area (4.090869074405497) <= 2.625 ; 0.0 <= tip_deflection (0.010195070649215895) <= 0.02625 ; 0.0 < proportional_gain1 (553.0202499656973) <= 1000.0 ; 0.0 < derivative_gain1 (51.342757861140285) <= 1000.0 ; 0.0 < proportional_gain2 (48.23124298809619) <= 1000.0 ; 0.0 < derivative_gain2 (4.477819809009244) <= 1000.0 ; unsuccessful

34 ; 3 ; 0.0 < link1_length (1.5) <= 1.75 ; 0.0 < link2_length (1.125) <= 1.5 ; -3.141592653589793 <= theta1_min (0.0) < 1.0064101236192153 ; 0.0 < theta1_max (1.0064101236192153) <= 3.141592653589793 ; -3.141592653589793 <= theta2_min (-2.4049686515706643) < 0.0 ; -2.4049686515706643 < theta2_max (0.0) <= 3.141592653589793 ; 0.0 < link1_cross_section_dimension (0.058845025683956616) <= 0.15000000000000002 ; 0.0 < link2_cross_section_dimension (0.030261681468808716) <= 0.1125 ; 0.002942251284197831 <= link1_cross_section_thickness (0.005884502568395662) <= 0.014711256420989154 ; 0.0015130840734404359 <= link2_cross_section_thickness (0.0030261681468808717) <= 0.007565420367202179 ; 0.0 < accessible_region_area (2.9563297381314446) <= 2.625 ; 0.0 <= tip_deflection (0.010195070649215895) <= 0.02625 ; 0.0 < proportional_gain1 (553.0202499656973) <= 1000.0 ; 0.0 < derivative_gain1 (51.342757861140285) <= 1000.0 ; 0.0 < proportional_gain2 (48.23124298809619) <= 1000.0 ; 0.0 < derivative_gain2 (4.477819809009244) <= 1000.0 ; unsuccessful

37 ; 3 ; 0.0 < link1_length (1.3125) <= 1.75 ; 0.0 < link2_length (1.3125) <= 1.3125 ; -3.141592653589793 <= theta1_min (-1.6959077469403279) < 0.0 ; -1.6959077469403279 < theta1_max (0.0) <= 3.141592653589793 ; -3.141592653589793 <= theta2_min (0.0) < 2.3535232653876097 ; 0.0 < theta2_max (2.3535232653876097) <= 3.141592653589793 ; 0.0 < link1_cross_section_dimension (0.06150337501251079) <= 0.13125 ; 0.0 < link2_cross_section_dimension (0.03342265575891957) <= 0.13125 ; 0.0030751687506255397 <= link1_cross_section_thickness (0.006150337501251079) <= 0.015375843753127698 ; 0.0016711327879459785 <= link2_cross_section_thickness (0.003342265575891957) <= 0.008355663939729892 ; 0.0 < accessible_region_area (4.981729006637212) <= 2.625 ; 0.0 <= tip_deflection (0.010030399962673136) <= 0.02625 ; 0.0 < proportional_gain1 (570.2584810235578) <= 1000.0 ; 0.0 < derivative_gain1 (52.94316638725306) <= 1000.0 ; 0.0 < proportional_gain2 (77.7232311714316) <= 1000.0 ; 0.0 < derivative_gain2 (7.215875075944809) <= 1000.0 ; unsuccessful

39 ; 3 ; 0.0 < link1_length (1.3125) <= 1.75 ; 0.0 < link2_length (1.3125) <= 1.3125 ; -3.141592653589793 <= theta1_min (0.0) < 1.1892199349229913 ; 0.0 < theta1_max (1.1892199349229913) <= 3.141592653589793 ; -3.141592653589793 <= theta2_min (-2.3535232653876097) < 0.0 ; -2.3535232653876097 < theta2_max (0.0) <= 3.141592653589793 ; 0.0 < link1_cross_section_dimension (0.06150337501251079) <= 0.13125 ; 0.0 < link2_cross_section_dimension (0.03342265575891957) <= 0.13125 ; 0.0030751687506255397 <= link1_cross_section_thickness (0.006150337501251079) <= 0.015375843753127698 ; 0.0016711327879459785 <= link2_cross_section_thickness (0.003342265575891957) <= 0.008355663939729892 ; 0.0 < accessible_region_area (3.4933335588362864) <= 2.625 ; 0.0 <= tip_deflection (0.010030399962673136) <= 0.02625 ; 0.0 < proportional_gain1 (570.2584810235578) <= 1000.0 ; 0.0 < derivative_gain1 (52.94316638725306) <= 1000.0 ; 0.0 < proportional_gain2 (77.7232311714316) <= 1000.0 ; 0.0 < derivative_gain2 (7.215875075944809) <= 1000.0 ; unsuccessful

43 ; 3 ; 0.0 < link1_length (1.5567951410224505) <= 1.75 ; 0.0 < link2_length (0.7783975705112253) <= 1.5567951410224505 ; -3.141592653589793 <= theta1_min (-2.4580611917887825) < -1.370467760770914 ; -2.4580611917887825 < theta1_max (-1.370467760770914) <= 3.141592653589793 ; -3.141592653589793 <= theta2_min (0.0) < 2.8573894984143893 ; 0.0 < theta2_max (2.8573894984143893) <= 3.141592653589793 ; 0.0 < link1_cross_section_dimension (0.05071617460646335) <= 0.15567951410224506 ; 0.0 < link2_cross_section_dimension (0.024905129862261795) <= 0.07783975705112253 ; 0.002535808730323167 5 <= link1_cross_section_thickness (0.005071617460646335) <= 0.012679043651615837 ; 0.0012452564931130898 <=

link2_cross_section_thickness (0.0024905129862261796) <= 0.006226282465565449 ; 0.0 < accessible_region_area (2.5830343986674387) <= 2.625 ; 0.0 <= tip_deflection (0.009120385595540107) <= 0.02335192711533676 ; 0.0 < proportional_gain1 (362.90360496525614) <= 1000.0 ; 0.0 < derivative_gain1 (33.69220551656433) <= 1000.0 ; 0.0 < proportional_gain2 (17.756563493544025) <= 1000.0 ; 0.0 < derivative_gain2 (1.648530844849792) <= 1000.0 ; successful

D.4.2.5 DesignParameters

The report on the values of each design parameter during the design process is given in `DesignParameters` file. The report includes any design cycle in which the design has not been successful—that is, a constraint violation has happened. If the design process succeeds in finding a satisfactory design, the resulted values for the design parameters are included in the `DesignParameters` file as the last line. The two fields at the beginning of each report for a design cycle are the design cycle ID and the depth of the corresponding design state in the dependency graph. The rest of the fields include the name of the design parameter and the value assigned. Tow adjacent fields are separeted from each other by a semicolon ';'.

3 ; 3 ; base_location: [-0.25, 1.125,] ; link1_length: 1.5206906325745548 ; link2_length: 0.7603453162872774 ; theta1_array: [1.8263830771681904, 1.7660965980901309, 1.445493514101444, 0.4992720772603446, 1.148799562275311, 0.7558123379402226, 1.481769289557572, 0.9797513603926523, 0.590231429395585, 1.3474065484141704, 0.9252198819686772, 1.295316288057006, 0.9881006072902687, 1.4056476493802699,] ; theta2_array: [2.564106837681789, 2.0788600827515014, 2.429133701286048, 2.276603865067832, 2.1743447682918706, 2.1100707391131763, 1.679116133310014, 1.8444700645133778, 1.5167159151807206, 1.4352464794149695, 1.4352464794149695, 1.0627325708382909, 0.8649887885219596, 0.0,] ; theta1_min: 0.4992720772603446 ; theta1_max: 1.8263830771681904 ; theta2_min: 0.0 ; theta2_max: 2.564106837681789 ; accessible_region_area: 2.820110874804171 ; link1_cross_section_dimension: 0.12873882479062435 ; link2_cross_section_dimension: 0.049265974918586156 ; link1_cross_section_thickness: 0.012873882479062435 ; link2_cross_section_thickness: 0.004926597491858616 ; material_name: steel_stainless_AISI_302_annealed ; material_mass_density: 7920.0 ; material_yield_stress: 2.6E8 ; material_elasticity_modulus: 1.9E11 ; structural_safety_factor: 3.0 ; link_cross_sectional_shape: hollow_round ; tip_deflection: 0.0011051471277170148 ; proportional_gain1: 1375.245122855074 ; proportional_gain2: 29.61009191218797 ; derivative_gain1: 127.67864711435413 ; derivative_gain2: 2.749020093546075 ; unsuccessful

5 ; 3 ; base_location: [-0.25, 1.125,] ; link1_length: 1.5206906325745548 ; link2_length: 0.7603453162872774 ; theta1_array: [1.8263830771681904, 1.7660965980901309, 1.445493514101444, 0.4992720772603446,

1.148799562275311, 0.7558123379402226, 1.481769289557572, 0.9797513603926523, 0.590231429395585, 1.3474065484141704, 0.9252198819686772, 1.295316288057006, 0.9881006072902687, 1.4056476493802699,] ; theta2_array: [2.564106837681789, 2.0788600827515014, 2.429133701286048, 2.276603865067832, 2.1743447682918706, 2.1100707391131763, 1.679116133310014, 1.8444700645133778, 1.5167159151807206, 1.4352464794149695, 1.4352464794149695, 1.0627325708382909, 0.8649887885219596, 0.0,] ; theta1_min: 0.4992720772603446 ; theta1_max: 1.8263830771681904 ; theta2_min: 0.0 ; theta2_max: 2.564106837681789 ; accessible_region_area: 2.820110874804171 ; link1_cross_section_dimension: 0.08529699024103615 ; link2_cross_section_dimension: 0.03694948118893962 ; link1_cross_section_thickness: 0.008529699024103615 ; link2_cross_section_thickness: 0.0036949481188939624 ; material_name: steel_stainless_AISI_302_annealed ; material_mass_density: 7920.0 ; material_yield_stress: 2.6E8 ; material_elasticity_modulus: 1.9E11 ; structural_safety_factor: 3.0 ; link_cross_sectional_shape: hollow_round ; tip_deflection: 0.002602398618321901 ; proportional_gain1: 706.8546566810312 ; proportional_gain2: 22.104360987620353 ; derivative_gain1: 65.62484372542369 ; derivative_gain2: 2.0521831776196673 ; unsuccessful

7 ; 3 ; base_location: [-0.25, 1.125,] ; link1_length: 1.5206906325745548 ; link2_length: 0.7603453162872774 ; theta1_array: [2.7046861288150095, 2.8131560547429113, 2.4136404800294935, 1.5251219456423235, 2.192130396296806, 1.8028667266938365, 2.449405603431744, 1.9955588294202582, 1.4952123273414886, 2.216372771621116, 1.794186105175623, 1.9711139855247022, 1.5477223095495816, 1.4056476493802699,] ; theta2_array: [-2.564106837681789, -2.0788600827515014, -2.429133701286048, -2.276603865067832, -2.1743447682918706, -2.1100707391131763, -1.679116133310014, -1.8444700645133778, -1.5167159151807206, -1.4352464794149695, -1.4352464794149695, -1.0627325708382909, -0.8649887885219596, 0.0,] ; theta1_min: 1.4056476493802699 ; theta1_max: 2.8131560547429113 ; theta2_min: -2.564106837681789 ; theta2_max: 0.0 ; accessible_region_area: 2.9909553613956126 ; link1_cross_section_dimension: 0.08529699024103615 ; link2_cross_section_dimension: 0.03694948118893962 ; link1_cross_section_thickness: 0.008529699024103615 ; link2_cross_section_thickness: 0.0036949481188939624 ; material_name: steel_stainless_AISI_302_annealed ; material_mass_density: 7920.0 ; material_yield_stress: 2.6E8 ; material_elasticity_modulus: 1.9E11 ; structural_safety_factor: 3.0 ; link_cross_sectional_shape: hollow_round ; tip_deflection: 0.002602398618321901 ; proportional_gain1: 706.8546566810312 ; proportional_gain2: 22.104360987620353 ; derivative_gain1: 65.62484372542369 ; derivative_gain2: 2.0521831776196673 ; unsuccessful

10 ; 3 ; base_location: [-0.25, 1.125,] ; link1_length: 1.3034491136353328 ; link2_length: 0.9775868352264996 ; theta1_array: [1.424317321092442, 1.5288401366019233, 1.1019842048851936, 0.20863343229332243, 0.8875495067908978, 0.5110397779965163, 1.3201063897994902, 0.7907317942585098, 0.4516272023620346, 1.2192495168197963, 0.7970628503743032, 1.2080711506977486, 0.9193315561019147, 1.4056476493802699,] ; theta2_array: [2.3018677747641307, 1.9271388340635636, 2.206935311932865, 2.089777814761733, 2.0069125658978155, 1.9534390483677553, 1.576301860113302, 1.7240481573006552, 1.4286767467902144, 1.353884122857439, 1.353884122857439, 1.0074315214662204, 0.8214215182898781, 0.0,] ; theta1_min: 0.20863343229332243 ; theta1_max: 1.5288401366019233 ; theta2_min: 0.0 ; theta2_max: 2.3018677747641307 ; accessible_region_area: 2.805439246655777 ; link1_cross_section_dimension: 0.09164457471547738 ; link2_cross_section_dimension: 0.04189922992805236 ; link1_cross_section_thickness: 0.009164457471547737 ; link2_cross_section_thickness: 0.004189922992805236 ; material_name: steel_stainless_AISI_302_annealed ; material_mass_density: 7920.0 ; material_yield_stress: 2.6E8 ; material_elasticity_modulus: 1.9E11 ; structural_safety_factor: 3.0 ; link_cross_sectional_shape: hollow_round ;

tip_deflection: 0.002288904138117323 ; proportional_gain1: 683.6900561763787 ; proportional_gain2: 46.96085698845915 ; derivative_gain1: 63.47422722497149 ; derivative_gain2: 4.359876350747827 ; unsuccessful

12 ; 3 ; base_location: [-0.25, 1.125,] ; link1_length: 1.3034491136353328 ; link2_length: 0.9775868352264996 ; theta1_array: [3.106751884890758, 3.050412516231119, 2.757149789245744, 1.815760590609346, 2.4533804517812197, 2.047639286637543, 2.611068503189826, 2.184578395554401, 1.6338165543750391, 2.34452980321549, 1.922431367699968, 2.0583591228839597, 1.6164913607379354, 1.4056476493802699,] ; theta2_array: [-2.3018677747641307, -1.9271388340635636, -2.206935311932865, -2.089777814761733, -2.0069125658978155, -1.9534390483677553, -1.576301860113302, -1.7240481573006552, -1.4286767467902144, -1.353884122857439, -1.353884122857439, -1.0074315214662204, -0.8214215182898781, 0.0,] ; theta1_min: 1.4056476493802699 ; theta1_max: 3.106751884890758 ; theta2_min: -2.3018677747641307 ; theta2_max: 0.0 ; accessible_region_area: 3.614846500459787 ; link1_cross_section_dimension: 0.09164457471547738 ; link2_cross_section_dimension: 0.04189922992805236 ; link1_cross_section_thickness: 0.009164457471547737 ; link2_cross_section_thickness: 0.004189922992805236 ; material_name: steel_stainless_AISI_302_annealed ; material_mass_density: 7920.0 ; material_yield_stress: 2.6E8 ; material_elasticity_modulus: 1.9E11 ; structural_safety_factor: 3.0 ; link_cross_sectional_shape: hollow_round ; tip_deflection: 0.002288904138117323 ; proportional_gain1: 683.6900561763787 ; proportional_gain2: 46.96085698845915 ; derivative_gain1: 63.47422722497149 ; derivative_gain2: 4.359876350747827 ; unsuccessful

15 ; 3 ; base_location: [-0.25, 1.125,] ; link1_length: 1.1405179744309162 ; link2_length: 1.1405179744309162 ; theta1_array: [1.1370159037178065, 1.3406659269527652, 0.8460329207423041, -0.015281376429308269, 0.6829069715921305, 0.3176379961234216, 1.1876970727287808, 0.6375115785982658, 0.3372431634147429, 1.1131630073599905, 0.6909763409144974, 1.1351434646514271, 0.8616629222248777, 1.4056476493802699,] ; theta2_array: [2.257037398547587, 1.8979207989275113, 2.16706815264633, 2.054956775761285, 1.9751160153878562, 1.923403072387216, 1.5557807475317544, 1.7002870326163788, 1.4109574299075878, 1.337453305315305, 1.337453305315305, 0.9961433442788543, 0.8124970723900948, 0.0,] ; theta1_min: -0.015281376429308269 ; theta1_max: 1.4056476493802699 ; theta2_min: 0.0 ; theta2_max: 2.257037398547587 ; accessible_region_area: 3.0194741798453535 ; link1_cross_section_dimension: 0.09746653452047076 ; link2_cross_section_dimension: 0.04577063421833817 ; link1_cross_section_thickness: 0.009746653452047076 ; link2_cross_section_thickness: 0.004577063421833817 ; material_name: steel_stainless_AISI_302_annealed ; material_mass_density: 7920.0 ; material_yield_stress: 2.6E8 ; material_elasticity_modulus: 1.9E11 ; structural_safety_factor: 3.0 ; link_cross_sectional_shape: hollow_round ; tip_deflection: 0.0023009346396080143 ; proportional_gain1: 688.5486322528478 ; proportional_gain2: 77.9986007314333 ; derivative_gain1: 63.925300571850855 ; derivative_gain2: 7.2414405640844794 ; unsuccessful

17 ; 3 ; base_location: [-0.25, 1.125,] ; link1_length: 1.1405179744309162 ; link2_length: 1.1405179744309162 ; theta1_array: [-2.8891320049141926, -3.0445985812993093, 3.0131010733886336, 2.0396753993319763, 2.6580229869799865, 2.2410410685106377, 2.7434778202605354, 2.337798611214645, 1.7482005933223306, 2.450616312675296, 2.0284296462298026, 2.131286808930281, 1.6741599946149726, 1.4056476493802699,] ; theta2_array: [-2.257037398547587, -1.8979207989275113, -2.16706815264633, -2.054956775761285, -1.9751160153878562, -1.923403072387216, -1.5557807475317544, -1.7002870326163788, -1.4109574299075878, -1.337453305315305, -1.337453305315305, -0.9961433442788543, -0.8124970723900948, 0.0,] ; theta1_min: -3.0445985812993093 ; theta1_max: 3.0131010733886336 ; theta2_min: -2.257037398547587 ; theta2_max: 0.0 ;

accessible_region_area: 4.225362012380889 ; link1_cross_section_dimension: 0.09746653452047076 ; link2_cross_section_dimension: 0.04577063421833817 ; link1_cross_section_thickness: 0.009746653452047076 ; link2_cross_section_thickness: 0.004577063421833817 ; material_name: steel_stainless_AISI_302_annealed ; material_mass_density: 7920.0 ; material_yield_stress: 2.6E8 ; material_elasticity_modulus: 1.9E11 ; structural_safety_factor: 3.0 ; link_cross_sectional_shape: hollow_round ; tip_deflection: 0.0023009346396080143 ; proportional_gain1: 688.5486322528478 ; proportional_gain2: 77.9986007314333 ; derivative_gain1: 63.925300571850855 ; derivative_gain2: 7.2414405640844794 ; unsuccessful

21 ; 3 ; base_location: [1.25, -0.625,] ; link1_length: 1.75 ; link2_length: 0.875 ; theta1_array: [-1.100195808110164, -0.8698819755295635, -0.8712940501598057, -0.4772567274755424, -0.6704736106568384, -0.5248007342188962, -0.4399759547909189, -0.48276592332573415, 0.0, -0.33802344229706194, -0.29058324508516703, -0.21352541428790528, -0.06226427460513628, -0.023058374852110985,] ; theta2_array: [2.2824160139785437, 2.734731173047627, 2.1543828616858436, 0.8319041739002149, 1.9469512385592467, 1.323381881327229, 2.5620892507176984, 1.673015058430811, 0.0, 2.229693048804815, 1.323381881327229, 1.881913585692951, 0.8319041739002149, 1.1503551299650279,] ; theta1_min: -1.100195808110164 ; theta1_max: 0.0 ; theta2_min: 0.0 ; theta2_max: 2.734731173047627 ; accessible_region_area: 3.2318251863236065 ; link1_cross_section_dimension: 0.09834270010059333 ; link2_cross_section_dimension: 0.039545791119743025 ; link1_cross_section_thickness: 0.009834270010059334 ; link2_cross_section_thickness: 0.003954579111974303 ; material_name: steel_stainless_AISI_302_annealed ; material_mass_density: 7920.0 ; material_yield_stress: 2.6E8 ; material_elasticity_modulus: 1.9E11 ; structural_safety_factor: 3.0 ; link_cross_sectional_shape: hollow_round ; tip_deflection: 0.003367425157030643 ; proportional_gain1: 1301.9890850434604 ; proportional_gain2: 33.339936976502194 ; derivative_gain1: 120.87750916061464 ; derivative_gain2: 3.0953013228654953 ; unsuccessful

25 ; 3 ; base_location: [-0.25, 1.125,] ; link1_length: 1.5206906325745548 ; link2_length: 0.7603453162872774 ; theta1_array: [1.8263830771681904, 1.7660965980901309, 1.445493514101444, 0.4992720772603446, 1.148799562275311, 0.7558123379402226, 1.481769289557572, 0.9797513603926523, 0.590231429395585, 1.3474065484141704, 0.9252198819686772, 1.295316288057006, 0.9881006072902687, 1.4056476493802699,] ; theta2_array: [2.564106837681789, 2.0788600827515014, 2.429133701286048, 2.276603865067832, 2.1743447682918706, 2.1100707391131763, 1.679116133310014, 1.8444700645133778, 1.5167159151807206, 1.4352464794149695, 1.4352464794149695, 1.0627325708382909, 0.8649887885219596, 0.0,] ; theta1_min: 0.4992720772603446 ; theta1_max: 1.8263830771681904 ; theta2_min: 0.0 ; theta2_max: 2.564106837681789 ; accessible_region_area: 2.820110874804171 ; link1_cross_section_dimension: 0.04963495688385445 ; link2_cross_section_dimension: 0.024632987459293078 ; link1_cross_section_thickness: 0.004963495688385445 ; link2_cross_section_thickness: 0.002463298745929308 ; material_name: steel_stainless_AISI_302_annealed ; material_mass_density: 7920.0 ; material_yield_stress: 2.6E8 ; material_elasticity_modulus: 1.9E11 ; structural_safety_factor: 3.0 ; link_cross_sectional_shape: hollow_round ; tip_deflection: 0.008766542738009385 ; proportional_gain1: 332.8892198101605 ; proportional_gain2: 16.743124612929197 ; derivative_gain1: 30.905650576732263 ; derivative_gain2: 1.5544425233865187 ; unsuccessful

29 ; 3 ; base_location: [1.25, -0.625,] ; link1_length: 1.75 ; link2_length: 0.875 ; theta1_array: [-0.07580939898497163, -0.1684102529634819, 0.1737520429919912, 0.06226427460513628, 0.36517495386630794, 0.29058324508516703, 0.4399759547909189, 0.48276592332573415, 0.0, 0.6977304418820188, 0.5248007342188962, 0.8105232774602644, 0.4772567274755424, 0.7016436037602003,] ; theta2_array: [-2.2824160139785437, -2.734731173047627,

-2.1543828616858436, -0.8319041739002149, -1.9469512385592467, -1.323381881327229, -2.5620892507176984, -1.673015058430811, 0.0, -2.229693048804815, -1.323381881327229, -1.881913585692951, -0.8319041739002149, -1.1503551299650279,] ; theta1_min: -0.1684102529634819 ; theta1_max: 0.8105232774602644 ; theta2_min: -2.734731173047627 ; theta2_max: 0.0 ; accessible_region_area: 2.8756172456197544 ; link1_cross_section_dimension: 0.05680396263726892 ; link2_cross_section_dimension: 0.02636386074649535 ; link1_cross_section_thickness: 0.005680396263726892 ; link2_cross_section_thickness: 0.0026363860746495354 ; material_name: steel_stainless_AISI_302_annealed ; material_mass_density: 7920.0 ; material_yield_stress: 2.6E8 ; material_elasticity_modulus: 1.9E11 ; structural_safety_factor: 3.0 ; link_cross_sectional_shape: hollow_round ; tip_deflection: 0.01105592332430511 ; proportional_gain1: 568.7736692745627 ; proportional_gain2: 23.980702322073615 ; derivative_gain1: 52.80531550366831 ; derivative_gain2: 2.2263839212735537 ; unsuccessful

32 ; 3 ; base_location: [1.25, -0.625,] ; link1_length: 1.5 ; link2_length: 1.125 ; theta1_array: [-1.3926362806486803, -1.3671005125019788, -1.1272751934818368, -0.5430906798841296, -0.8789827928117926, -0.63961211611278, -0.8410686705679303, -0.6435011087932843, 0.0, -0.6146562790145749, -0.4053946269790508, -0.4094122604468562, -0.12809822701372353, -0.1190991371835603,] ; theta2_array: [2.0943951023931953, 2.4049686515706643, 1.9904097103647405, 0.7901993345727791, 1.8139252921554911, 1.2505347847626034, 2.3005239830218627, 1.5707963267948966, 0.0, 2.052131395668969, 1.2505347847626034, 1.7570566303714525, 0.7901993345727791, 1.0894612579208243,] ; theta1_min: -1.3926362806486803 ; theta1_max: 0.0 ; theta2_min: 0.0 ; theta2_max: 2.4049686515706643 ; accessible_region_area: 4.090869074405497 ; link1_cross_section_dimension: 0.058845025683956616 ; link2_cross_section_dimension: 0.030261681468808716 ; link1_cross_section_thickness: 0.005884502568395662 ; link2_cross_section_thickness: 0.0030261681468808717 ; material_name: steel_stainless_AISI_302_annealed ; material_mass_density: 7920.0 ; material_yield_stress: 2.6E8 ; material_elasticity_modulus: 1.9E11 ; structural_safety_factor: 3.0 ; link_cross_sectional_shape: hollow_round ; tip_deflection: 0.010195070649215895 ; proportional_gain1: 553.0202499656973 ; proportional_gain2: 48.23124298809619 ; derivative_gain1: 51.342757861140285 ; derivative_gain2: 4.477819809009244 ; unsuccessful

34 ; 3 ; base_location: [1.25, -0.625,] ; link1_length: 1.5 ; link2_length: 1.125 ; theta1_array: [0.21663107355354483, 0.32880828400893347, 0.42973318631402224, 0.12809822701372353, 0.5736841360212621, 0.4053946269790508, 0.8410686705679303, 0.6435011087932843, 0.0, 0.9743632785995318, 0.63961211611278, 1.0064101236192153, 0.5430906798841296, 0.7976843660916495,] ; theta2_array: [-2.0943951023931953, -2.4049686515706643, -1.9904097103647405, -0.7901993345727791, -1.8139252921554911, -1.2505347847626034, -2.3005239830218627, -1.5707963267948966, 0.0, -2.052131395668969, -1.2505347847626034, -1.7570566303714525, -0.7901993345727791, -1.0894612579208243,] ; theta1_min: 0.0 ; theta1_max: 1.0064101236192153 ; theta2_min: -2.4049686515706643 ; theta2_max: 0.0 ; accessible_region_area: 2.9563297381314446 ; link1_cross_section_dimension: 0.058845025683956616 ; link2_cross_section_dimension: 0.030261681468808716 ; link1_cross_section_thickness: 0.005884502568395662 ; link2_cross_section_thickness: 0.0030261681468808717 ; material_name: steel_stainless_AISI_302_annealed ; material_mass_density: 7920.0 ; material_yield_stress: 2.6E8 ; material_elasticity_modulus: 1.9E11 ; structural_safety_factor: 3.0 ; link_cross_sectional_shape: hollow_round ; tip_deflection: 0.010195070649215895 ; proportional_gain1: 553.0202499656973 ; proportional_gain2: 48.23124298809619 ; derivative_gain1: 51.342757861140285 ; derivative_gain2: 4.477819809009244 ; unsuccessful

37 ; 3 ; base_location: [1.25, -0.625,] ; link1_length: 1.3125 ; link2_length: 1.3125 ; theta1_array: [-1.6176994043853186, -1.6959077469403279, -1.3283565971082436, -0.5983223570106451, -1.0466081901196598, -0.7349915458796219, -1.1278852827212578, -0.7751933733103613, 0.0, -0.8295129353380343, -0.5007740567458927, -0.5677499767558721, -0.18332990414023892, -0.19923532829879925,] ; theta2_array: [2.0593936016755015, 2.3535232653876097, 1.9591711870486728, 0.7816522611508834, 1.787917723448789, 1.2357656026255144, 2.255770565442515, 1.5503867466207224, 0.0, 2.018732870261025, 1.2357656026255144, 1.7324978166841027, 0.7816522611508834, 1.0770558855056875,] ; theta1_min: -1.6959077469403279 ; theta1_max: 0.0 ; theta2_min: 0.0 ; theta2_max: 2.3535232653876097 ; accessible_region_area: 4.981729006637212 ; link1_cross_section_dimension: 0.06150337501251079 ; link2_cross_section_dimension: 0.03342265575891957 ; link1_cross_section_thickness: 0.006150337501251079 ; link2_cross_section_thickness: 0.003342265575891957 ; material_name: steel_stainless_AISI_302_annealed ; material_mass_density: 7920.0 ; material_yield_stress: 2.6E8 ; material_elasticity_modulus: 1.9E11 ; structural_safety_factor: 3.0 ; link_cross_sectional_shape: hollow_round ; tip_deflection: 0.010030399962673136 ; proportional_gain1: 570.2584810235578 ; proportional_gain2: 77.7232311714316 ; derivative_gain1: 52.94316638725306 ; derivative_gain2: 7.215875075944809 ; unsuccessful

39 ; 3 ; base_location: [1.25, -0.625,] ; link1_length: 1.3125 ; link2_length: 1.3125 ; theta1_array: [0.4416941972901832, 0.6576155184472824, 0.6308145899404292, 0.18332990414023892, 0.7413095333291294, 0.5007740567458927, 1.1278852827212578, 0.7751933733103613, 0.0, 1.1892199349229913, 0.7349915458796219, 1.1647478399282312, 0.5983223570106451, 0.8778205572068885,] ; theta2_array: [-2.0593936016755015, -2.3535232653876097, -1.9591711870486728, -0.7816522611508834, -1.787917723448789, -1.2357656026255144, -2.255770565442515, -1.5503867466207224, 0.0, -2.018732870261025, -1.2357656026255144, -1.7324978166841027, -0.7816522611508834, -1.0770558855056875,] ; theta1_min: 0.0 ; theta1_max: 1.1892199349229913 ; theta2_min: -2.3535232653876097 ; theta2_max: 0.0 ; accessible_region_area: 3.4933335588362864 ; link1_cross_section_dimension: 0.06150337501251079 ; link2_cross_section_dimension: 0.03342265575891957 ; link1_cross_section_thickness: 0.006150337501251079 ; link2_cross_section_thickness: 0.003342265575891957 ; material_name: steel_stainless_AISI_302_annealed ; material_mass_density: 7920.0 ; material_yield_stress: 2.6E8 ; material_elasticity_modulus: 1.9E11 ; structural_safety_factor: 3.0 ; link_cross_sectional_shape: hollow_round ; tip_deflection: 0.010030399962673136 ; proportional_gain1: 570.2584810235578 ; proportional_gain2: 77.7232311714316 ; derivative_gain1: 52.94316638725306 ; derivative_gain2: 7.215875075944809 ; unsuccessful

43 ; 3 ; base_location: [2.75, 1.125,] ; link1_length: 1.5567951410224505 ; link2_length: 0.7783975705112253 ; theta1_array: [-1.841743177133317, -2.234645941276905, -2.1030228587442052, -1.5803360074509354, -2.101691900266075, -1.808417716248621, -2.4580611917887825, -2.000640459862265, -1.5075116724748197, -2.3849331973654104, -1.8020196084096765, -2.1393342274091958, -1.5200267188657943, -1.370467760770914,] ; theta2_array: [0.0, 0.7722645770738099, 1.094542932510318, 0.9744725044399294, 1.5507376751335993, 1.4991016848920589, 1.7319514332910713, 1.8914492237267633, 1.576527017187388, 2.1499047363494395, 2.1499047363494395, 2.5516014242815754, 2.313412773511665, 2.8573894984143893,] ; theta1_min: -2.4580611917887825 ; theta1_max: -1.370467760770914 ; theta2_min: 0.0 ; theta2_max: 2.8573894984143893 ; accessible_region_area: 2.5830343986674387 ; link1_cross_section_dimension: 0.05071617460646335 ; link2_cross_section_dimension: 0.024905129862261795 ; link1_cross_section_thickness: 0.005071617460646335 ; link2_cross_section_thickness: 0.0024905129862261796 ; material_name: steel_stainless_AISI_302_annealed ; material_mass_density: 7920.0 ;

```
material_yield_stress: 2.6E8 ; material_elasticity_modulus: 1.9E11 ;
structural_safety_factor: 3.0 ; link_cross_sectional_shape: hollow_round ;
tip_deflection: 0.009120385595540107 ; proportional_gain1: 362.90360496525614 ;
proportional_gain2: 17.756563493544025 ; derivative_gain1: 33.69220551656433 ;
derivative_gain2: 1.648530844849792 ; successful
```

D.4.2.6 RDSpecific

The fields are separeted by commas ',' and are ordered as follow:

- date and time of starting the project;
- date and time of finishing the project;
- the time spent during the run time;
- the memory available to Java interpreter at the beginning of the project;
- the memory available to Java interpreter at the end of the project;
- the amount of memory spent for the project in kilo bytes;
- the total number of design cycles generated during the project;
- the total number of ticks of the counter of the message events;
- the total number of messages exchanged during the project;
- the report of any exception that might have occured;
- the total number of paths (design approach combinations) that were tried in the project.

```
28-Oct-98 5:25:07 AM , 28-Oct-98 5:25:53 AM , 0:0:46 , 1048568 ,
2424824 , 1376.0 K , 43 , 18087 , 2899, no exception , 15
```