

Building Systems Using Analysis Patterns

Eduardo B. Fernandez
Florida Atlantic University
Boca Raton, FL
ed@cse.fau.edu

1. ABSTRACT

We consider the use of analysis patterns in the definition of initial object-oriented models. In particular, we consider large patterns, as opposed to the small analysis patterns shown in most of the current literature, and we use analogy in carrying over the patterns between applications. We believe that this approach produces a software architecture that is more flexible and reusable than approaches applying patterns at a later stage.

1.1 Keywords

Analysis patterns, object-oriented analysis, software architecture, software design

2. INTRODUCTION

A pattern is a recurring combination of meaningful units that occurs in some context. Patterns have been used in building construction, enterprise management, and in several other fields. Their use in software is becoming very important because of their value for reusability and quality; they distill the knowledge and experience of many designers.

Software patterns present a generic approach to solving a problem; they have to be tailored to specific cases. Properly used, they can save time and improve quality. Their use has a strong effect on the general architecture of a system.

Depending on what stage or level they are applied, patterns can be classified as:

- Language patterns -- idioms, apply to one (or a few) languages.
- Design patterns -- a few classes that describe a design construct [6].
- Architectural patterns -- sets of perhaps many classes that represent some architectural structure at the system level [1].
- Analysis patterns -- defined in the conceptual model of the application or domain.

Most of the use of patterns until now has been at the design stage. We believe that analysis patterns can contribute more to reusability and software quality than the other varieties. We show here how their use contributes to simplifying the search for the structure of a system and how analogy can help in finding what patterns to apply.

When translating use cases or a similar type of requirements into an initial OOA model the user is confronted with a rather difficult problem where errors are common [4]. The use of analysis patterns can be of considerable help at this stage and it results in an architecture that is easier to extend and reuse than a system where each requirement is mapped in an ad hoc manner and where patterns maybe applied at a later stage.

3. ANALYSIS PATTERNS

An analysis pattern is a set of classes and associations that have some meaning in the context of an application; that is, it is a conceptual model of a part of the application. However, the same structure may be valid for other applications, and this is the aspect that makes them very valuable for reuse and composition.

Another aspect that is important for reuse is that one can find subpatterns within a complex pattern; these subpatterns have use on their own right.

Analysis patterns differ from design patterns in the following ways:

- Design patterns are closer to implementation, they focus on typical design aspects, e.g., user interfaces, creation of objects, basic structural properties.
- Design patterns apply to any application; for example, all applications have user interfaces, need to create objects.
- Analysis patterns are application dependent, their semantics describe specific aspects of some domain or application.

Analysis patterns have been studied in [2] and [5]. In general, they are much less used than design or architectural patterns. In particular, the analysis patterns discussed in these two references are small patterns, of a few classes; our interest is in larger patterns.

4. LOOKING FOR ANALOGIES

A good strategy for patterns is the use of analogies to apply a known pattern to a similar type of situation. In one of my projects I developed a design for a computer repair shop. A computer repair shop fixes broken computers. The shop is part of a chain of similar shops. Computers are brought to the shop by some customer and a reception technician makes an estimate. If the customer agrees, the computer is assigned for repair to some repair technician, who keeps a Repair Event document. All the Repair Event documents for a computer are collected in its repair log. A repair event may be suspended because of a lack of parts or other reasons.

A class diagram for this system is shown in Figure 1, while Figure 2 shows a state diagram for Repair Event. Figure 3 shows a sequence diagram for assigning the repair of some computer to a technician. The class diagram reflects the facts that a computer can be estimated at different shops in the chain and that some of these estimates will become actual repairs. A computer that has been repaired at least once has a repair log that collects all its repair events.

In a later project I needed to design a system to handle patient registration in a hospital. Noticing that a hospital, instead of broken computers, fixes sick people, I arrived at the class diagram of Figure 4. I just needed to reinterpret each class in the repair shop as a corresponding class for a hospital. For example, the computer becomes a patient, the estimate becomes the diagnosis, the repair event a hospital stay, etc. Similarly, sequence diagrams and state diagrams are developed in analogy with those used in computer repair.

This example shows the basic idea for a strategy to reuse patterns. It implies an abstraction of the initial model from where new concrete models can be developed. Using this approach we have developed patterns for inventory/library circulation [3], car rental/videotape rental, flight itineraries/pipe layout, student registration/conference registration, and others.

As said earlier, one can also take portions of these patterns and find simpler patterns. For example, the set of personnel-related classes in Figures 1 and 4 can define a Personnel pattern, applicable to other situations.

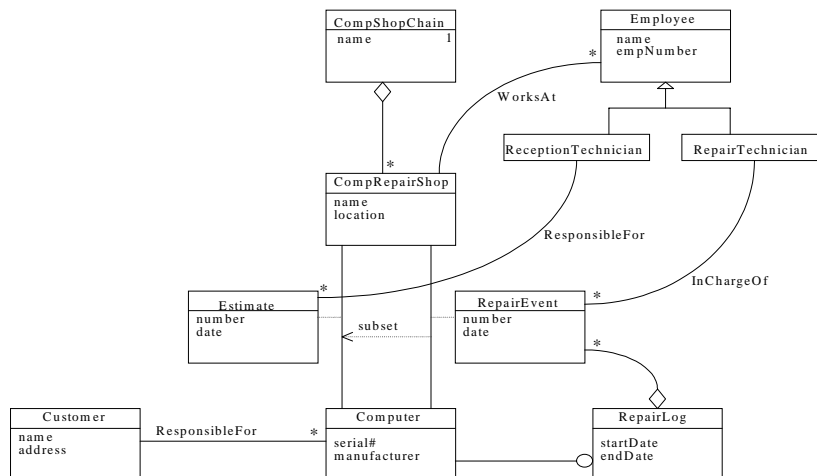


Figure 1. Class diagram for the computer repair shop.

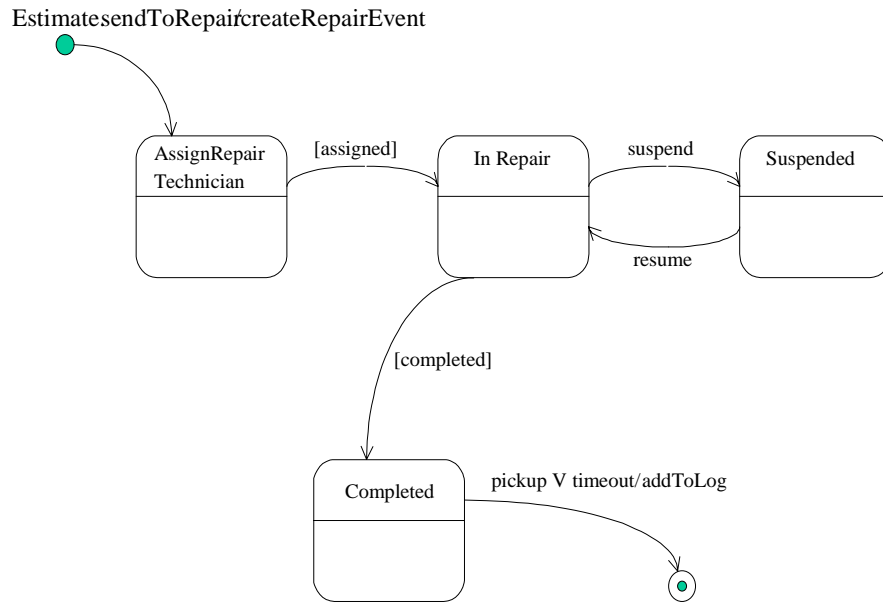


Figure 2. State diagram for Repair Event

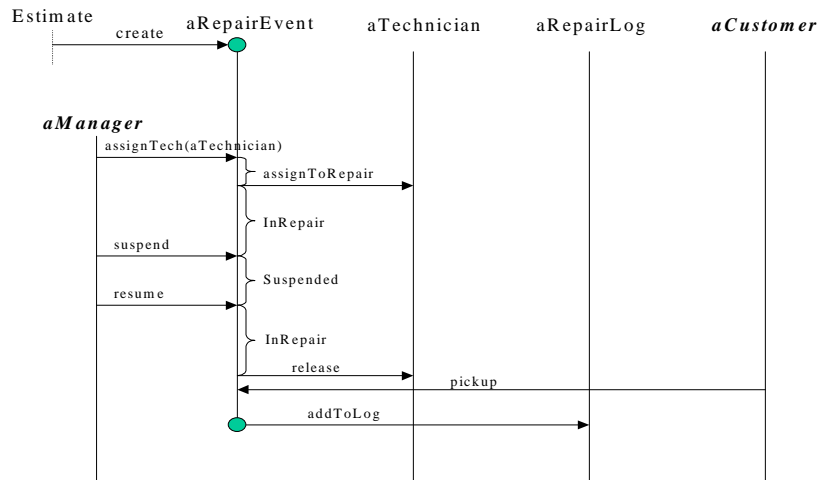


Figure 3. Sequence diagram for assigning repair jobs to technicians

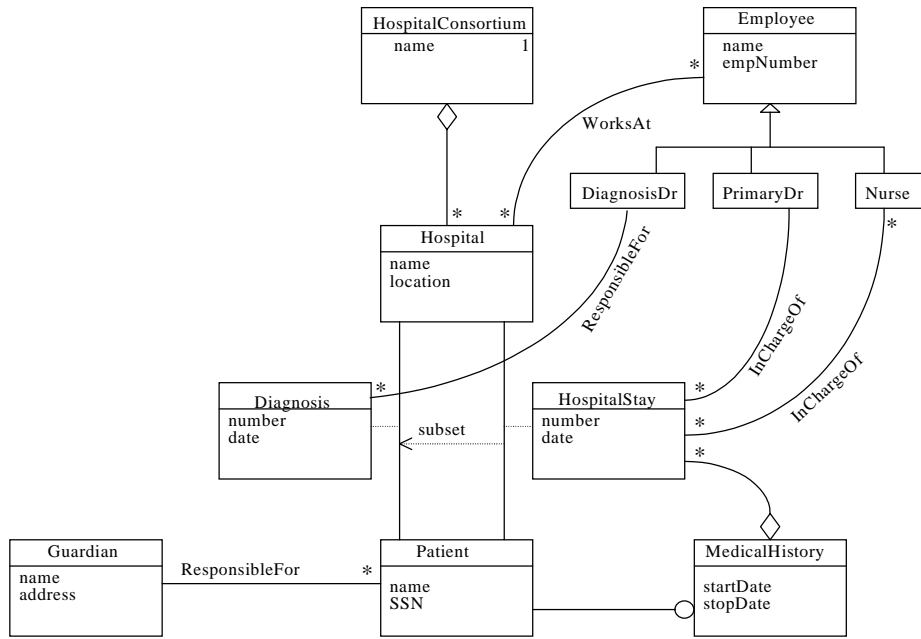


Figure 5. Class diagram for hospital registration.

5. CONCLUSIONS

A good analysis model for a subsystem of a complex system can be abstracted and become an analysis pattern that can be used in other applications. Analogy plays an important role in reusing an analysis pattern. Subsets of a pattern may also have their own application in other situations. All this can save time and improve the quality of a system. One can start a new system by trying to apply a large pattern (8-12 classes), or several small patterns (2-5 classes). One of the most difficult steps in practice is to get an initial model; this approach makes it easier to start. In subsequent steps the initial model can be modified to suit better the needs of the application. An analysis model using patterns is easier to understand and to extend. It should also result in a higher quality design. A software architecture constructed this way is more reusable and extensible than an architecture defined directly from the requirements or where patterns are applied later. While we haven't shown this in this paper, it is clear that combinations of patterns are extensible because of the possibility of replacing a pattern with another concrete realization of the same pattern. They are reusable because of the possibility of replacing several of the used patterns to fit the requirements of a new application. All this needs

detailed study, one can use the comparison of [7] as a framework of reference to analyze the proposed approach.

6. REFERENCES

- [1] F. Buschmann et al., *Pattern-oriented software architecture*, Wiley 1996.
- [2] P.Coad, *Object models – Strategies, patterns, and applications* (2nd. Edition), Prentice-Hall 1997.
- [3] E. B. Fernandez and Z. W. Peng, "An object-oriented model for manufacturing inventory control systems", Tech. Report TR-CSE-97-30, Dept. of Computer Science and Eng., Florida Atlantic University, April 1997.
- [4] E. B. Fernandez, "Good analysis as the basis for good design and implementation", Report TR-CSE-97-45, Dept. of Computer Science and Eng., Florida Atlantic University, September 1997. Presented at OOPSLA'97.
- [5] M. Fowler, *Analysis patterns -- Reusable object models*, Addison- Wesley, 1997.
- [6] E. Gamma et al., *Design patterns –Elements of reusable object-oriented software*, Addison-Wesley 1995.
- [7] M. Shaw, "Comparing architectural design styles", *IEEE Software*, November 1995, 27-41.

