others are relatively harmonious. These different styles are important because the success of the ATAM rests squarely on eliciting the right scenarios and prioritizing them correctly.

Also worthy of note is that almost invariably the step 0: information exchange / pre-meeting results in a request for additional architectural documentation. The initial documentation is often far too vague to support any analysis, sometimes consisting only of high level data flows and divisions of functionality with no clear realization in software. In one evaluation additional information was requested, in the form of the following questions, to address the gaps in the original documentation:

- what is the structure of the message handling software (i.e. how the functionality is broken down in terms of modules, functions, APIs, layers, etc.)?
- what facilities exist in the software architecture (if any) for self-testing and monitoring of software components?
- what facilities exist in the software architecture (if any) for redundancy, liveness monitoring, failover, and how data consistency is maintained (so that one component can take over from another and be sure that it is in a consistent state with the failed component)?
- what is the process and/or task view of the system, including mapping of these processes/tasks to hardware and the communication mechanisms between them?
- what functional dependencies exist among the software components (often called a “uses” view)?
- what data is kept in the database (which was mentioned by one of your stakeholders), how big it is, how much it changes, and who reads/writes it?
- what is the anticipated frequency and volume of data being transmitted among the system components?

The preparation process for an ATAM, which results in creating and/or collecting suitable architectural documentation has by itself been of value to our customers.

Our final observation culled from our experience with the ATAM is that we have noticed that the iteration of design and analysis really works. Each time that we go through this process—collect scenarios, ask questions, ask to have the architecture presented to us, build models, and critique the architecture—the architecture and the way that it is documented changes (and, in our opinion and the opinions of the project teams, changes for the better). The ATAM is truly working as a spiral model of analysis and design.

6. REFERENCES
view one cannot be done without the other. A design is what you analyze and an analysis tells you how to go about refining a design. Thus, this process is not simply a front-end gate that a system passes through on its journey from requirements to fielded system. Analyses must live with a design throughout a system’s lifetime, so that the system is built appropriately and maintained correctly. In addition, the ATAM helps drive design: it identifies areas of risk and helps to plan for risk mitigation. It also drives the documentation of the architecture.

The ATAM draws its inspiration and techniques from three areas: the Software Architecture Analysis Method (SAAM) [4], quality attribute communities, and the notion of architectural styles. The ATAM is intended to analyze an architecture with respect to its quality attributes, not its functional correctness. Although this is the ATAM’s focus, there is a problem in operationalizing this focus: we (and the software engineering community in general) do not understand quality attributes well: what it means to be “open” or “interoperable” or “secure” or “high performance” changes from system to system and from community to community. So, we turn to scenarios as a means of operationalizing the analysis of quality attributes. The focus of the SAAM is the use of scenarios for architectural modifiability evaluation. Scenarios provided a vehicle for concretizing modifiability; they represent specific examples of current and future uses of a system. The future uses typically imply modifications against which the architecture can be assessed thereby transforming the abstract notion of modifiability into concrete modification scenarios. Performing the ATAM has taught us that scenarios are also the driving force in understanding runtime qualities (such as performance or availability) because they specify the kinds of operations over which performance needs to be measured or the kinds of failures the system will have to withstand.

The ATAM also builds on the knowledge bases associated with quality attributes. We organize this knowledge into what we call attribute models. We augment scenarios with attribute-centric questions such as those shown in Section 2, based upon an analytic model of each attribute. An attribute model answers the following three questions: 1) What are the measurable or observable manifestations of the attribute? 2) What are the attribute-relevant stimuli or events to which the architecture must respond? 3) What are the characteristics of the architecture that contribute to the observable manifestation?

The ATAM also builds on the concept of architectural styles. “An architectural style is a description of component types and a pattern of their runtime control and/or data transfer. A style can be thought of as a set of constraints on an architecture—constraints on component types and their interactions—and these constraints define a set or family or architectures that satisfy them”. [1] The ATAM uses a particular specialization of architectural styles, ABASs. An ABAS is an architecture style in which the constraints focus on component types and patterns of interaction that are particularly relevant to quality attributes such as performance, modifiability, security or reliability. ABASs aid architecture evaluation by focusing attention on patterns that dominate the architecture, by highlighting the attribute-specific questions associated with the pattern, and by placing the answers to these questions into an analytic framework. For example, if an architecture used a collection of interacting processes, this could be recognized as a performance ABAS. The questions associated with this performance ABAS would probe important architectural parameters such as the priority of the processes, estimates of their execution time, places where they synchronize, queuing disciplines, etc.; information that relevant to understanding the performance of this style. The answers to these questions then feed into an explicit analytic framework such as rate monotonic analysis for performance [6].

The ATAM caters to both the sociological and scientific sides of software engineering and to both quantitative and qualitative attributes. Scenario elicitation fosters stakeholder interaction and helps to elicit and refine quality attribute requirements. Scenarios also help to concretize the more qualitative attributes such as modifiability. Quality attribute models such as schedulability, queuing and Markov models provide some of the scientific underpinnings for evaluating architectures. We are finding that the interaction between these two points of view brings many issues to the fore that might remain undiscovered until much later in development.

5. EXPERIENCE SO FAR
We have now performed, or are performing, 5 ATAM-based evaluations (2 internal; 3 external). Although our ATAM experience base is still small, it builds upon a much larger experience base garnered in doing SAAM evaluations. From this experience, we have observed a number of issues worth noting.

First, since this is a scenario-based method and scenarios come from the system’s stakeholders, dealing with stakeholders successfully is crucial. However, it is not always easy. The goals of the architectural evaluation must be made clear, because we, as external evaluators, are always regarded with some suspicion and we are taking time out of the stakeholder’s schedules. Getting the stakeholders to “buy-in” to the process is essential and this means making them understand the steps of the ATAM, why these steps are important, and why they need to occur in this order. For example, the stakeholders need to be focused on the scenarios that represent critical uses of and anticipated changes to the system.

Another issue that we have noted from performing ATAM (and SAAM) evaluations is that getting consensus on the right set of scenarios works differently in different organizations. So, we must be sensitive to the different styles of organizations. Some organizations are democratic, others are strictly top-down hierarchies, some have centralized decision making while others are distributed, some organizations have openly antagonistic sub-groups,
architecture (for modifiability) and how the architecture reacts to it (for quality attributes such as performance, security and reliability).

- Step 6 - Perform quality attribute-specific analysis: the architect guides the analysis showing why the architecture meets the attribute-specific requirements, as illuminated by the scenarios of interest. The architectural parameters or elements to which this scenario is sensitive (i.e. statistically correlated) are identified. For example, if end-to-end latency is sensitive to the size of a queue, the queue’s size is noted as a sensitivity point.

- Step 7 - Identify trade-off points: to find trade-offs we locate all important architectural elements in which multiple sensitivities exist. For example, the number of hot spare databases might be a sensitivity point for availability and performance.

- Step 8 - Consolidate findings and develop action plan: this plan is a set of recommendations for improving the architecture in the light of the analysis findings. Additionally, we might ask for more supporting documentation such as: more architectural information, scenarios, environmental information, platform information, details about constraints, or justification for requirements.

These steps and their relationships are illustrated in Figure 1. In addition to these steps, we are developing a handbook: a set of materials that accompanies the evaluation that describe many of the evaluation artifacts. We have a handbook section that describe attribute based architectural styles (ABASs)—styles that address performance, modifiability, security, and availability along with their accompanying analytical frameworks. We also have a section with a set of quality attribute-specific questions that aid us in probing the architecture. For example, when building a performance model of some portion of the system, we ask questions about the scenarios such as the following:

Identify scenarios or create additional scenarios which highlight performance issues. The following questions might be helpful:

- What event starts the scenario. For example, is it a message arrival, keystroke, mouse click, state change, the passage of time....?
- How often does this initiating event occur (e.g., periodically with a fixed rate, stochastically with a known mean)?
- What is the performance requirement (e.g., hard deadline, soft deadline, throughput, average-case response time)? Quantify if possible (e.g., a hard deadline of 100 ms).
- What is the consequence of not meeting the performance requirement (e.g., catastrophe, inconvenience, annoyance, system failure and reboot).

By following through a standard set of quality-specific questions, we elicit the information needed to analyze that quality. In addition to scenario-specific elicitation questions we have a set of questions that aid us in gathering the information needed to build an analytic model of the quality. For example, in gathering performance information we elicit information about resource usage and contention: the raw data needed to build an analytic performance model [6].

4. THE RATIONALE FOR ATAM
The ATAM is a spiral model of design and analysis; in our
1. ABSTRACT
Software architectures, like complex designs in any field, embody trade-offs made by the designers. However, these trade-offs are not always made explicitly by the designers and they may not understand the impacts of their decisions. This paper introduces a scenario-based analysis technique for software architectures—called ATAM—that not only analyzes a software architecture with respect to multiple quality attributes, but explicitly considers the trade-offs inherent in the design.

1.1 Keywords
Architecture analysis, analytic models, architectural styles

2. WHY ARCHITECTURE TRADEOFF ANALYSIS?
In the past the software engineering community has paid lip-service to quality attributes in designing software architectures, but we have done little to ensure that these quality attributes are satisfied by the design. Recent efforts on cataloguing the implications of using design patterns and architectural styles contribute, in an informal way, to ensuring the quality of a design [2]. More formal efforts also exist to ensure that quality attributes are addressed. These consist of analyses in areas such as performance evaluation [6], Markov modeling for availability [3], and inspection and review methods for modifiability [4].

But these techniques are typically performed in isolation and their implications are considered in isolation. This is dangerous. It is dangerous because all design involves trade-offs and if we simply optimize for a single quality attribute, we stand the chance of ignoring other attributes of importance. Even more significantly, if we do not analyze for multiple attributes, we have no way of understanding the trade-offs made in the architecture—places where improving one attribute causes another one to be compromised.

We have developed a scenario-based method, called the architecture trade-off analysis method (ATAM) that is specifically aimed at locating and analyzing trade-offs in a software architecture. Having a structured method helps ensure that the right questions regarding an architecture will be asked early, during the requirements and design stages when discovered problems can be solved cheaply. It guides users of the method—the stakeholders—to look for conflicts and for resolutions to these conflicts in the software architecture. This position paper will briefly describe the method (more details can be found in [5]) and will discuss the implications of the ATAM and our experience with it thus far.

3. ATAM STEPS
The steps of the method are defined as follows:

- **Step 0 - Information exchange/pre-meeting:** in this step we describe the method to the stakeholders, set expectations, and see the architect’s initial presentation of the architecture.
- **Step 1 - Scenario brainstorming:** scenarios of uses of the system, faults of the system, and anticipated changes to the system are elicited from stakeholders. During this phase the analysts add or augment the scenarios based upon the quality attributes under review, their experience, and their need for additional insight into the architecture.
- **Step 2 - Architecture presentation:** during this phase the architecture is presented in detail and the most important and illustrative normal usage scenarios are mapped onto the architecture, to aid in understanding the system and, in particular, how data and control flow through it. The analysts will attempt to identify and probe architectural styles here.
- **Step 3 - Scenario coverage checking:** we use a set of standard quality attribute-specific questions to ensure proper coverage of an attribute by the scenarios. In particular, we look to see if boundary conditions have been covered.
- **Step 4 - Group and prioritize scenarios:** the stakeholders vote on the scenarios that are of highest concern for them. During this phase they can suggest grouping scenarios. After the voting is complete, we determine a cut-off point at 10-12 scenarios.
- **Step 5 - Map high priority scenarios onto architecture:** in this step the architect walks through each high-priority attribute specific scenario, showing how it affects the