# Using ATAM to Evaluate a Pacemaker Architecture

*Yang Cao, Jinrui Wang*

ATAM allows two different variants of emphasis when carried out. The first is architecture centric and the second is stakeholder-centric. In the first the emphasis is on eliciting the architecture information and analysing the architecture. In the second the emphasis is on eliciting stakeholder points of view. These two approaches are called the two faces of ATAM. In this evaluation we adopt the architecture centric face because the architectural decisions made are the main focus of the evaluation and because of the small number of stakeholders involved.

**Phase 1: Presentation**

There are three steps in this phase. The first step starts by presenting the ATAM process to the stakeholders. The second step defines the business goals of the architecture. Finally in the third step the architect describes the architecture to the evaluation team highlighting the following: the quality attributes pursued, the architectural decisions made, and the overall architecture design. These are illustrated in the following sections.

**Step1: Present the ATAM method**

ATAM (Architecture TradeOff Analysis Method) is an architecture evaluation method developed in the Software Engineering Institute at the end of the 90s. The purpose of ATAM is to assess the consequences of architectural design decisions in the light of quality attributes. ATAM helps in foreseeing how an attribute of interest can be affected by an architectural design decision. The Quality Attributes of interest are clarified by analyzing the stakeholder's scenarios in terms of stimuli and responses. Once the scenarios had been defined, the next step is to define which architectural approaches can affect to those quality attributes. In ATAM, the term architectural approaches are used to refer both to architectural styles or patterns. The ATAM output are:

- A set of architectural approaches identified and or applied: Sometimes we can identify architectural approaches that cannot be applied on our architecture.
- A Utility Tree: a top-down mechanism for directly and efficiently translating the business drivers of a system into concrete quality attribute scenarios.
- A set of scenarios identified and the subset that had been effectively mapped in the architecture.
- A set of questions about the quality attributes in the architecture and the answers to these questions. In our case our questions are a set of metrics and the answers are the values measured.
- The risks identified: risks that the architecture is able to mitigate and the risks that threaten the system and the business goals.

**Step 2: Present Business Drivers**

For this pacemaker architecture, two business goals were specified. The first goal aims to improve the reliability of the entire Pacemaker system which provides dual chamber, rate adaptive bradycardia pacing support, provides historical data on device performance, and provides user diagnostics through brady analysis functions. The second goal is to make the architecture flexible to accommodate different operating modes of Pacemaker and provide high performance.

**Quality Attributes**

The requirements document in a traditional system development cycle usually lacks or weakly articulates the quality attributes of the system. Requirements document tend to be good at describing the functional requirements which are different from the quality attributes. The difference being that the system can have accurate functionality but does not deliver it on time which is a performance quality attribute. A quality attribute is characterized by three categories: external stimuli, responses, and architectural decisions. The stimuli are the events that cause the architecture to respond. The way the architecture addresses these events must be expressed in concrete and measurable terms which form the responses. The last category is the architectural decisions that describe the approaches adopted and how they impact the quality attribute responses. The architecture being evaluated here has three primary quality attributes, and they are in order
of priority: Reliability, Functionality, and Performance.

The following table summarizes the stakeholders in this project and the viewpoints that have been included to address their concerns.

*Table 1: Stakeholders and Relevant Viewpoints*

| **Stakeholder** | **Viewpoint(s) that apply to that class of stakeholder's concerns** |
|---|---|
| Users | the system is reliable and available when needed |
| Customer | the architecture can be implemented on schedule and to budget |
| Project manager | the architecture will allow teams to work largely independently, interacting in disciplined and controlled ways (in addition to cost and schedule) |
| Developers | strategies to achieve all of those goals |
| Security analyst | the system will meet its information assurance requirements |
| Configuration management specialists | maintaining current and past versions of the elements |
| Engineers | producing a running version of the system |
| Maintainers | modifying the software elements |
| Implementers | implementing the elements |
| Software architects | those software elements sufficiently large or complex enough to warrant their own software architectures |

**Step3: Present the architecture**

The system is the Pacemaker System which is mainly used to control pulsing and sensing of heart rate for specific users. The primary purpose of a pacemaker is to maintain an adequate heart rate, either because the heart's natural pacemaker is not fast enough, or because there is a block in the heart's electrical conduction system. The bradycardia analysis functions permit the following pacing measurements and tests to be performed: Lead impedance, Pacing threshold, P and R wave measurement, Battery status, Temporary brady pacing and Motion sensor trending.

The PACEMAKER system consists of three major components:

• Device (also called the pulse generator or PG)

• Device Controller-Monitor (DCM) and associated software

• Leads

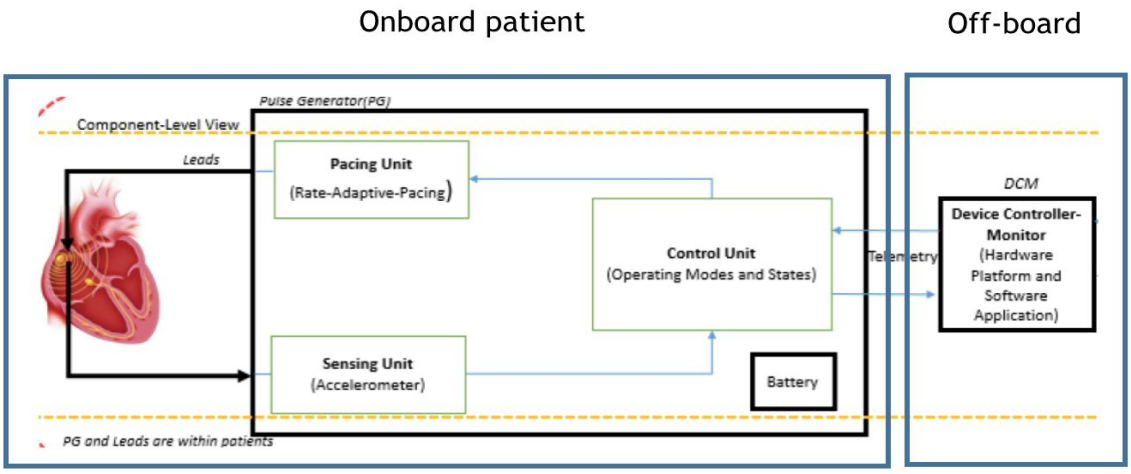



Figure 1: Pacemaker architecture that supports reliability

Figure 1 illustrates a conceptual overview of the architecture that can address this attribute by decomposing into 2 subsystems. The reliability attribute succeeds if the different operating modes of Pacemaker can run successfully to maintain stability and safety. The second attribute is functionality which is the ability to satisfy the fundamental requirements. It is measured by the amount of changes required to the different components of the architecture. The third attribute is performance which is the ability of the system to respond to stimuli within an acceptable timeframe.

**Architectural Decisions**

The success of an architecture is dependent on the achievement of its quality attributes and the forces at work behind these attributes are the architectural decisions. Getting these wrong could lead to a disastrous result. The remainder of this section describes the key architectural decisions made by the architect and organized by the three quality attributes they aim to support: reliability, functionality, and performance.

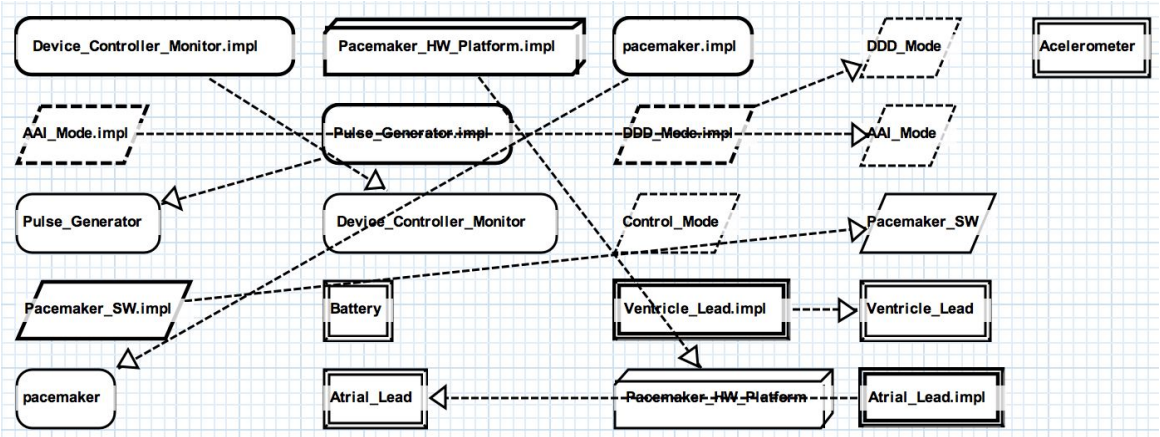



Figure 2: Original pacemaker architecture

In original design of pacemaker-EASY, the system architecture is not complete as Figure 2.

Thus, I used Feedback Control Loop which includes nominal and error models, and three design tactics which include Splitting, Intermediary and Augmenting to refine my pacemaker design. In this way, the performance and reliability of Pacemaker system will be improved efficiently.

**Phase 2: Investigation and Analysis**

This phase comprises of three steps: identifying the architectural decisions made (step 4), generating the quality attribute utility tree (step 5), and analysing the architectural decisions (step 6). Although in perfect documentation all the architectural decisions should be listed that is not always the case as was found when evaluating the Pacemaker projects. Therefore, the evaluation team has the responsibility to elicit architectural decisions from the architecture documentation, and the architect's presentation. This should elicit any architectural decisions not highlighted by the architect.

**Step4: Identify architectural approaches (patterns)**

A functional requirement is a statement of something the system must be able to do to be considered "complete." A module implements a functional requirement. The highest vote getters are the driving requirements. Often these are the only ones we will have time to consider as we make design decisions. Stakeholders and their concerns addressed by this viewpoint include:

- users, who is concerned that the system is reliable and available when needed;
- customer, who is concerned that projected changes to the system over its lifetime can be made economically by confining the effects of each change to a small number of elements;
- project manager, who must define work assignments, form teams, and formulate project plans and budgets and schedules;
- developers, who is worried about strategies to achieve all of those goals;
- security analyst, who is concerned that the system will meet its information assurance requirements;
- testers and integrators who use the modules as their unit of work;
- configuration management specialists who are in charge of maintaining current and past versions of the elements;
- system build engineers who use the elements to produce a running version of the system;
- maintainers, who are tasked with modifying the software elements;
- implementers, who are required to implement the elements;
- software architects for those software elements sufficiently large or complex enough to warrant their own software architectures;

Ten scenarios have been elicited for the architecture. The scenario elicited to prove that the architecture supports Reliability, Functionality, and Performance. The architect scenario is ranked with high for importance because it is the main quality attribute for the success of the architecture. The difficulty rate is set to medium, as the change requires creating a new adapter between the different operating modes of pacemaker.

**Customer Scenarios:**

| Scenario Refinement for Scenario #1 | | |
|---|---|---|
| Scenario(s): | | When a Pulse Generator senses an abnormal decrease in the heart rate, it starts generating pulses in less than one millisecond. |
| Business Goals: | | |
| Relevant Quality Attributes: | | Reliability, Functionality (or Performance) |
| Scenario Components | Stimulus: | The heart rate of a patient decreased abnormally. |
| | Stimulus Source: | The patient |
| | Environment: | The Pulse Generator is in the process of working. |
| | Artifact (If Known): | Leads, Pulse Generator, Device Controller-Monitor |
| | Response: | The Pulse Generator starts generating pulses. |
| | Response Measure: | one millisecond |
| Questions: | | What's the lowest heart rate of a patient before it is detected by the system's sensor? |
| Issues: | | May need to train installers to avoid parameter setting errors and prevent malfunctions. |

| Scenario Refinement for Scenario #2 | | |
|---|---|---|
| Scenario(s): | | When the battery of a Pulse Generator is about to run out, it switches working modes to extend lifespan in less than one second. |
| Business Goals: | | |
| Relevant Quality Attributes: | | Usability, Cost-saving |
| Scenario Components | Stimulus: | The battery of a Pulse Generator is below the critical level. |
| | Stimulus Source: | the Pulse Generator |
| | Environment: | The Device Controller is in the process of monitoring and the Pulse Generator is working. |
| | Artifact (If Known): | Leads, Pulse Generator, Device Controller-Monitor |
| | Response: | The Pulse Generator switches its working mode into a low-power status. |
| | Response Measure: | One second |
| Questions: | | What's the lowest battery could a Pulse Generator be before it switches working modes to maintain essential functional requirements? |
| Issues: | | Patients may need to detect the operating states of battery and relevant components in hospital termly. |

**Developer Scenarios:**

| Scenario Refinement for Scenario #3 | | |
|---|---|---|
| Scenario(s): | | When a developer adds the support of DDDR model to the system, it should be accurate and doesn't influence other functions. |
| Business Goals: | | |
| Relevant Quality Attributes: | | Maintainability, Modifiability |
| Scenario Components | Stimulus: | add the support of DDDR model to the system |
| | Stimulus Source: | developer |
| | Environment: | design-time, develop time |
| | Artifact (If Known): | source code |
| | Response: | New model will be accurate, and don't influence other functions. |
| | Response Measure: | finish in one month |
| Questions: | | How developers can familiar with the code quickly? |
| Issues: | | May need perfect and effective mechanism and accurate and complete documents. |

| Scenario Refinement for Scenario #4 | | |
|---|---|---|
| Scenario(s): | | When finishing the code of rate sensing function, The pacemaker experiences a thorough unit test. |
| Business Goals: | | |
| Relevant Quality Attributes: | | |
| Scenario Components | Stimulus: | When finishing the function of rate sensing |
| | Stimulus Source: | developer |
| | Environment: | design-time, develop-time |
| | Artifact (If Known): | source code |
| | Response: | Rate sensing passes the unit test |
| | Response Measure: | every 2 seconds |
| Questions: | | What is the correct value of heart rate? |
| Issues: | | May need a simulator to simulate human's heartbeat |

**Engineer Scenario:**

| Scenario Refinement for Scenario #5 | | |
|---|---|---|
| Scenario(s): | | When the sensors sense arrhythmia, the pacemaker will record relevant data and prepared for doctor to check. |
| Business Goals: | | Practical product; feature-rich product |
| Relevant Quality Attributes: | | performance |
| Scenario Components | Stimulus: | Arrhythmia of the host |
| | Stimulus Source: | Pulses external to the system, irregular heartbeat |
| | Environment: | The pacemaker is in the standby mode. |
| | Artifact (If Known): | System's impulse sensor, info-logout software component |
| | Response: | The pacemaker records the arrhythmia information |
| | Response Measure: | Data in the flash |
| Questions: | | How to distinguish arrhythmia with normal heartbeat rate changing? |
| Issues: | | May need further information to adjust or use machine learning |

| Scenario Refinement for Scenario #6 | | |
|---|---|---|
| Scenario(s): | | When the pacemaker itself meets system error, the pacemaker records and informs doctor about the risk |
| Business Goals: | | Safest system, feature-rich product |
| Relevant Quality Attributes: | | Safety, reliability |
| Scenario Components | Stimulus: | System errors |
| | Stimulus Source: | Internal errors such as software error; external errors such as not recorded pulses. |
| | Environment: | Anytime the pacemaker is at working |
| | Artifact (If Known): | System's impulse sensor, system's component checking software component |
| | Response: | The pacemaker record and inform the doctor about the risk |
| | Response Measure: | Data in the flash & data uploaded to the doctor |
| Questions: | | How to determine whether errors come from external factors or internal factors? |
| Issues: | | May need extra sensors for error detection. |

**Product Manager Scenarios:**

| Scenario Refinement for Scenario #7 | | |
|---|---|---|
| Scenario(s): | | When the system detects that the pacemaker doesn't work properly, it will try to recover or switch to backup system and give patient alerts. |
| Business Goals: | | Safest device, feature-rich |
| Relevant Quality Attributes: | | Safety, availability |
| Scenario Components | Stimulus: | Malfunctions, including low-battery, problems with detecting intra cardiac signals |
| | Stimulus Source: | Main functional parts of the pacemaker |
| | Environment: | The error handler is monitoring the status of other parts |
| | Artifact (If Known): | Error handler |
| | Response: | Try to recover or switch to backup system and give alerts |
| | Response Measure: | Less than one second |
| Questions: | | How long at least should the backup system work properly before doctors fix the problem? |
| Issues: | | Need to tell patients avoid potential dangerous situation and teach them how to recognize the meaning of different notifications. |

| Scenario Refinement for Scenario #8 | | |
|---|---|---|
| Scenario(s): | | When you want to add some functions, or replace some units (like battery) of the device, you can do it easily and keep/reuse most parts from the original device. |
| Business Goals: | | Resource economy, maintainability |
| Relevant Quality Attributes: | | Reusability, portability |
| Scenario Components | Stimulus: | When you want to add some functions, or replace some units. |
| | Stimulus Source: | New feature requirement or equipment maintenance |
| | Environment: | Normal operation |
| | Artifact (If Known): | Technicians, human operator |
| | Response: | Make modification without side effect, deploy it with minimal effort |
| | Response Measure: | Extent of effort/cost |
| Questions: | | What's the specific parameter requirement for different units of the device? |
| Issues: | | The technician must the life cycle of each unit of the device. |

**Maintainer Scenarios:**

| Scenario Refinement for Scenario #9 | | |
|---|---|---|
| Scenario(s): | | When a patient experiences irregular beating from the pacemaker, the doctor/programmer can reprogram the pacemaker by adjusting the threshold or changing the pacing modes. |
| Business Goals: | | feature-rich product, smart system |
| Relevant Quality Attributes: | | Maintainability |
| Scenario Components | Stimulus: | Patients experience irregular beating from the pacemaker. |
| | Stimulus Source: | Problems with sensing including undersensing or oversensing; problems with output including output failure or failure to capture. |
| | Environment: | Pacemaker is implanted in the patient's body |
| | Artifact (If Known): | |
| | Response: | The pacemaker has a reprogramming functionality allows |
| | Response Measure: | As needed |
| Questions: | | How is the pacing modes related to the irregular beating? |
| Issues: | | The doctor meets with patients every 6 months for regular check and responses to patients immediately as requested |

| Scenario Refinement for Scenario #10 | | |
|---|---|---|
| Scenario(s): | | When a pacemaker's battery runs out, it will produce signals asking for battery replacement before it gets fully depleted. |
| Business Goals: | | Safest system, smart system |
| Relevant Quality Attributes: | | Maintainability, Reliability |
| Scenario Components | Stimulus: | A pacemaker's battery becomes depleted |
| | Stimulus Source: | Pacemaker's battery |
| | Environment: | The pacemaker is implanted in the patient's body |
| | Artifact (If Known): | |
| | Response: | The pacemaker sends out signals for replacement |
| | Response Measure: | As soon as the battery life is below the warning threshold |
| Questions: | | How long should the battery keep working after sending out signals? |
| Issues: | | The battery life should be checked during the regular follow-up care. |

We elicited information on the architectural approaches with respect to Reliability, Functionality, and Performance scenarios. As stated above, the system was loosely organized around the notion of PG and

DCM. This dictated both the hardware architecture and the process architecture and affected the system's performance characteristics, as we shall see. In addition to this style,

- for Reliability, a Feedback Control Loop approach will be described
- for Functionality, standard subsystem organizational patterns will be described
- for Performance, Splitting, Intermediary and Augmenting tactics will be described. Each of these approaches was probed for risks, sensitivities, and tradeoffs via our style-specific questions, as we shall show below.

**Step5: Generate quality attribute utility tree**

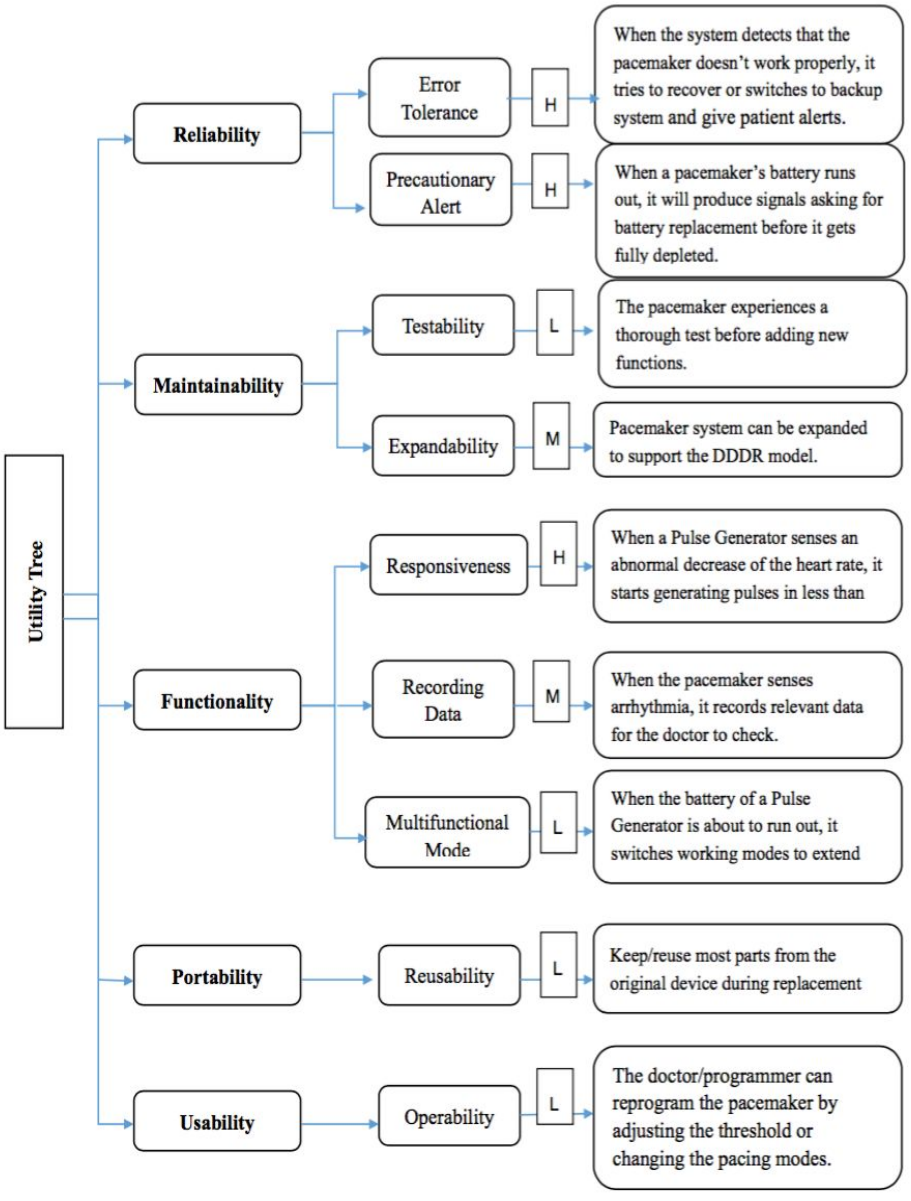



Figure 3: Pacemaker Utility Tree

Four architectural decisions have been listed above. The utility tree elicits the quality attributes down to the scenario level to provide a mechanism for translating business goals into concrete practical scenarios. The utility tree also aids in prioritising the quality attributes. This step is considered a crucial step which guides the rest of the analysis without which the evaluators could spend valuable time analysing the architecture without addressing the important quality attributes as far as the stakeholders are concerned. Figure 3 shows the utility tree for the architecture. There are three levels in the tree: the quality attributes level, the refinements level, and the scenarios level. The aim of the refinement is to decompose the quality attribute further if possible. The last level holds the scenarios in a concrete form. This level also holds the scenarios' rankings which decide their priorities. In ATAM, ranking format can differ based on the participants' preferences. It could be a scale based on 0 to 10 or relative ranking such as High (H), Medium (M), and Low (L). The ranking is done using two variables: importance and difficulty. Importance states how important the scenario is for the success of the architecture and difficulty describes the degree of difficulty in achieving that scenario. We used the relative ranking to prioritise the scenarios.

**Step6: Analyze architectural approaches**

At this stage in the analysis we had in our possession a set of architectural documentation including some architectural approaches that had been identified, but we had little insight into the way the architectural approaches actually performed to accomplish the Pacemaker's mission. So we used a set of screening questions and qualitative analysis heuristics to flesh out our understanding of the approaches and to probe them for risks, sensitivity points, and tradeoffs. For example, for the Feedback Control Loop approach (for Reliability) we generated a set of questions. The following is a subset of the questions applied to the style:

- How is the failure of a component detected?
- How is the failure of a communication channel detected?
- What is the latency for a spare component to be turned into a working replacement?
- By what means is a failed component marked for replacement?
- How are the system's working components notified that they should use the services of the spare? These questions allowed us to probe the availability architectural approach and formed the basis for our analysis of this approach, as we shall show below.

Recall that a sensitivity point for a quality attribute is defined as a parameter in the architecture to which some measurable attribute is highly correlated; small changes in such parameters are likely to have significant effects on the measurable behavior of the system. For this reason we need to focus our attention on these points as they pose the highest risks to the system's success, particularly as the system evolves. We find sensitivity points by first using architectural approaches and their associated questions to guide us in our architectural elicitation. Based upon the elicited information we can begin to build models of some aspect of a quality attribute. For example, we might build a collection of formal analytic models such as rate monotonic analysis or queuing models to understand a performance goal such as predicting the worst-case inter-node latency.

For the Pacemaker system we realized, via the utility tree construction process, that three quality attributes were the major architectural drivers for overall system quality: Reliability, Functionality, and Performance. Hence we can say that system quality (QS), is a function f of the quality of the Reliability (QR), the Functionality (QF), and the Performance (QP):

QS = f(QR, QF, QP)

Each of these analyses was created by first eliciting an architectural approach and then applying quality attribute-specific analysis questions to elicit information about the attribute in question. Using this information we then built simple but adequate analytic models.

**Tactic 1: Splitting**

The pacemaker system is decomposed into 2 subsystems: Pulse Generator system and Device Controller Monitor system. Distributed architecture techniques require synchronization. This is a closed system so no need to translate between systems.

Besides, the Accelerometer module in previous Pulse Generator system is divided into four modules, including Controller, Accelerometer, Error Handler, and Pacing Unit. Each module is responsible for respective functions after modification. In this way, the error tolerance□ of reliability, software independence of portability and performance of system have been improved dramatically. There is also a feedback control loop forming inside the Pulse Generator.

**Tactic 2: Intermediary**

The process Pacemaker_TR is added as an intermediary between process Pacemaker_SW and general operating modes for better expansibility and performance. Control Mode is separated from general operating modes, because it determines switching of modes and judging of errors.

I considered tactic Encapsulation as an alternative of Intermediary, but I didn't select it. A major reason is that an efficient interface of process Pacemaker_SW is needed in tactic Encapsulation, but I didn't know how to create an interface between process and threads which include DDD Mode, AAI Mode and any other operating modes. So I chose tactic Intermediary to create another process Pacemaker_TR for transition.

**Tactic 3: Augmenting**

Two additional threads and three additional processes are added to the system in order to implement a feedback control loop including nominal and error flows. Also, a pacemaker_HR _Platform is added to the Device Controller Monitor system because of the new Pacemaker_TR process.

**Feedback Control Loop**

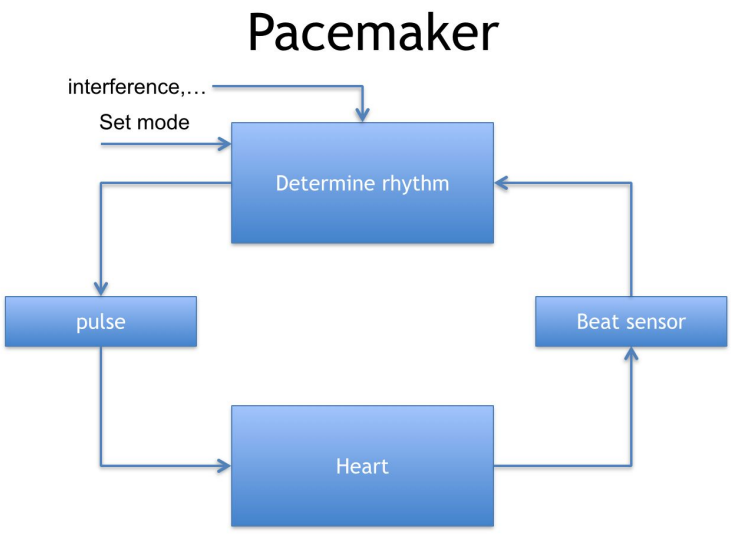



Figure 4: Pacemaker Feedback Control Loop

Instead of the previous design, a feedback control loop is added into the Pulse Generator system, including nominal and error model and identifying the controller, sensors, actuators, controlled process. Control unit is a finite state machine that controls timing and strength of pulse. Actuators/Pacing Unit uses leads to actuate the heart when needed. In Sensing Unit, the same leads are used to administer shock. In this way, no matter how quick the heart rate changes, we can sense these current values and determine if it is necessary to generate a pulse or change the operating modes of pacemaker. The frequency and magnitude of Pulse are chosen by controller. If sampling misses a beat, may pulse too soon. If sampling reads in between beats, the software may deduce the wrong magnitude.

**Phase 3: Testing**

**Step7: Brainstorm and prioritize scenarios**

Scenarios represent uses of—stimuli to—the Pacemaker architecture. These are applied not only to determine if the architecture meets a functional requirement, but also for further understanding of the system's architectural approaches and the ways in which these approaches meet the quality requirements such as performance, availability, modifiability, and so forth. The scenarios for the Pacemaker ATAM were collected by a round-robin brainstorming activity in which no criticism and little or no clarification was provided. Table 2 shows a few of the 10 growth-and-exploratory scenarios that were elicited in the course of the ATAM evaluation.

Table2: Scenarios for the Pacemaker Evaluation

| Scenario# | Description | Votes |
|---|---|---|
| #1 | When a Pulse Generator senses an abnormal decrease of the heart rate, it starts generating pulses in less than one millisecond. | 4 |
| #2 | When the battery of a Pulse Generator is about to run out, it switches working modes to extend lifespan in less than one second. | 1 |
| #3 | When a developer wants to add the support of DDDR model to the system, it could be accurate and don't influence other functions. | 2 |
| #4 | When finish the code of rate sensing function, we should also write a unit test. | 0 |
| #5 | When the sensors sense arrhythmia, the pacemaker will record relevant data and prepared for doctor to check. | 2 |
| #6 | When the pacemaker itself meets system error, the pacemaker records and informs doctor about the risk | 1 |
| #7 | When the system detects that the pacemaker doesn't work properly, it will try to recover or switch to backup system and give patient alerts. | 5 |
| #8 | When you want to add some functions, or replace some units (like battery) of the device, you can do it easily and keep/reuse most parts from the original device. | 1 |
| #9 | When a patient experiences irregular beating from the pacemaker, the doctor/programmer can reprogram the pacemaker by adjusting the threshold or changing the pacing modes. | 1 |
| #10 | When a pacemaker's battery runs out, it will produce signals asking for battery replacement before it gets fully depleted. | 3 |

Prioritization of the scenarios allows the most important scenarios to be addressed within the limited amount of time (and energy) available for the evaluation. Here, “important” is defined entirely by the stakeholders. The prioritization was accomplished by giving each stakeholder a fixed number of votes; 30% of the total number of scenarios has been determined to be a useful heuristic. Thus, for the Pacemaker system, each stakeholder was given 4 votes that they use to vote for scenarios in which they were most interested. Typically, the resulting totals will provide an obvious cutoff point; 10-15 scenarios are the most that can be considered in a normal one-day session; for the Pacemaker system a natural cutoff occurred at 3 scenarios. Some negotiation is appropriate in choosing which scenarios to consider; a stakeholder with a strong interest in a particular scenario can argue for its inclusion, even if it did not receive a large number of votes in the initial prioritization.

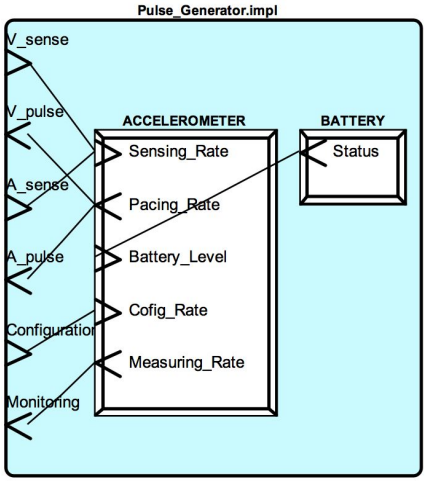
**Step8: Analyze architectural approaches**

In the ATAM process, once a set of scenarios have been chosen for consideration, these are “mapped” onto the architecture as test cases of the architectural approaches that have been documented. In the case of a scenario that implies a change to the architecture, the architect demonstrates how the scenario would affect the architecture in terms of the changed, added, or deleted components, connectors, and interfaces. For the use case scenarios, the architect traces the execution path through the relevant architectural views. Stakeholder discussion is important here to elaborate the intended meaning of a scenario description and to discuss how the mapping is or is not suitable from their perspective. The mapping process also illustrates weaknesses in the architecture and its documentation, or even missing architectural approaches. For the Pacemaker, each of the high-priority scenarios were mapped onto the appropriate architectural approach. For example, when a scenario implied a modification to the architecture, the ramifications of the change were mapped onto the source view, and scenario interactions were identified as sensitivity points. For reliability and performance, use case scenarios describing the execution and failure modes of the system were mapped onto runtime and system views of the architecture. During this mapping the models of latency and availability that we had built in the style-based analysis phase were further probed and refined.

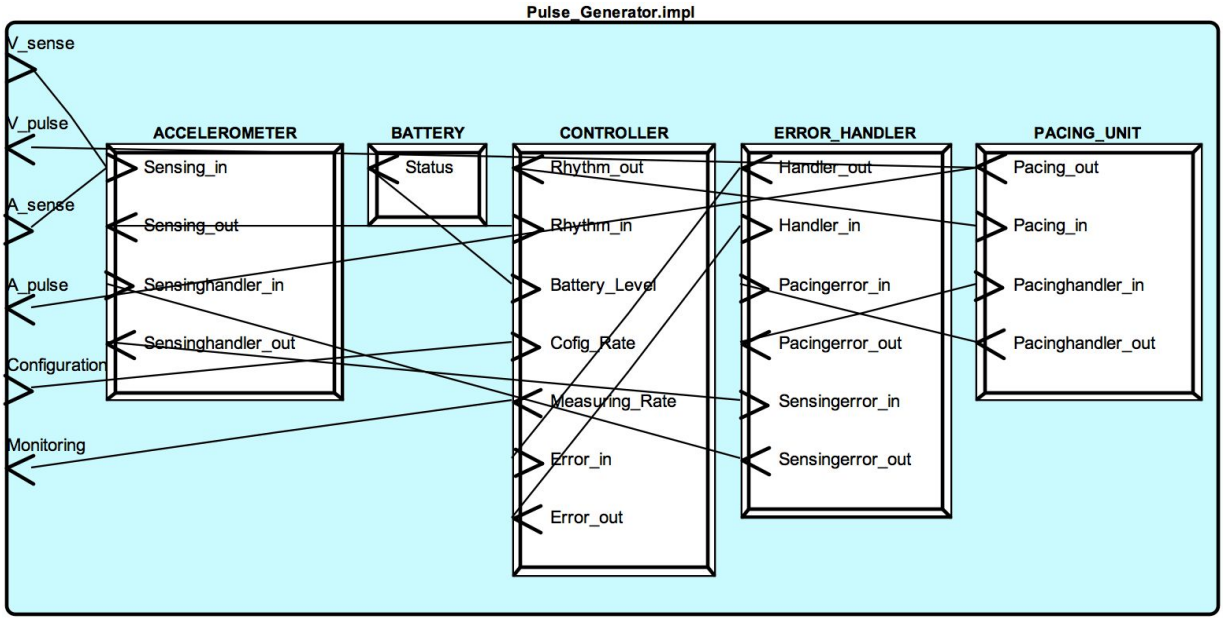
**Tactic 1: Splitting**

The Accelerometer module in Pulse Generator system is divided into four modules, including Accelerometer, Controller, Error Handler, and Pacing Unit. Each module is responsible for respective functions after modification.

Before modification：



After modification：

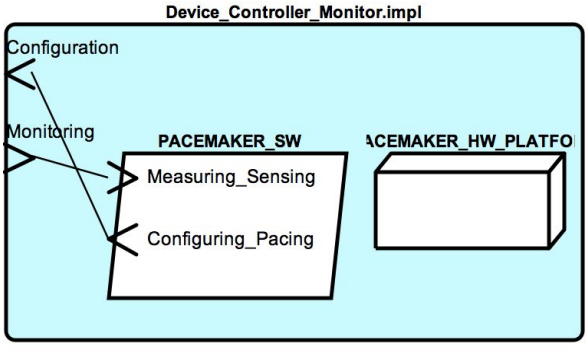


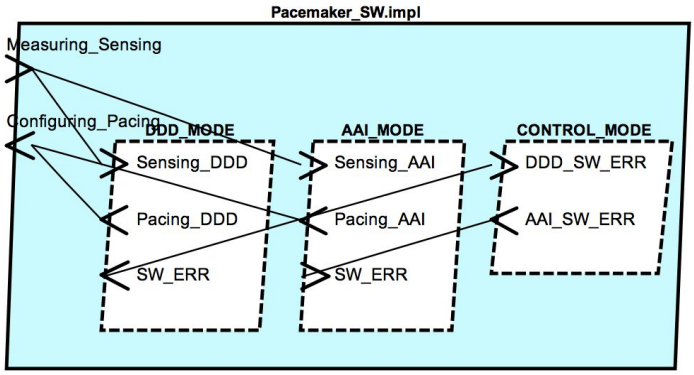

In this way, the testability of maintainability, hardware independence□ of portability and efficiency have been improved in the pacemaker system.

**Tactic 2: Intermediary**

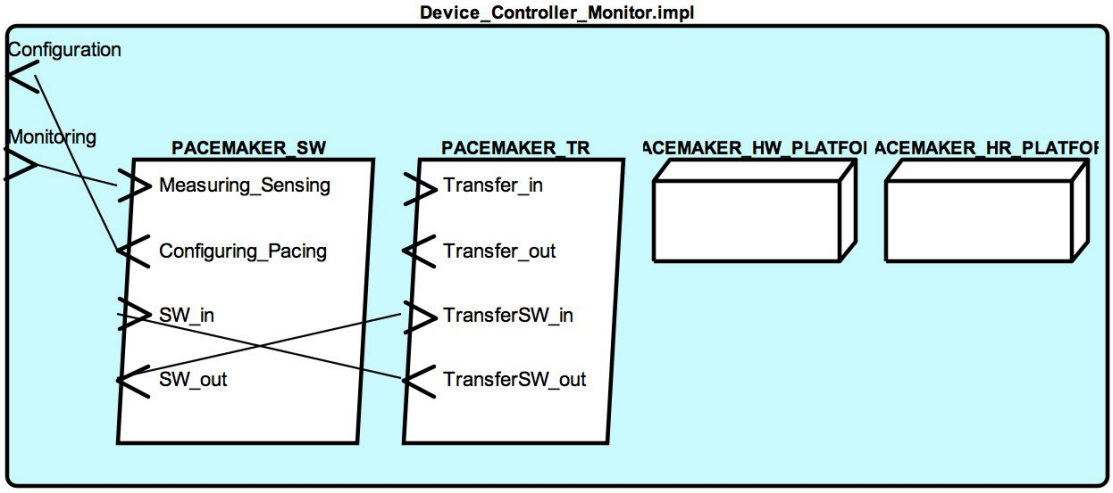
The process Pacemaker_TR is added as an intermediary between process Pacemaker_SW and general operating modes for better expansibility and performance.

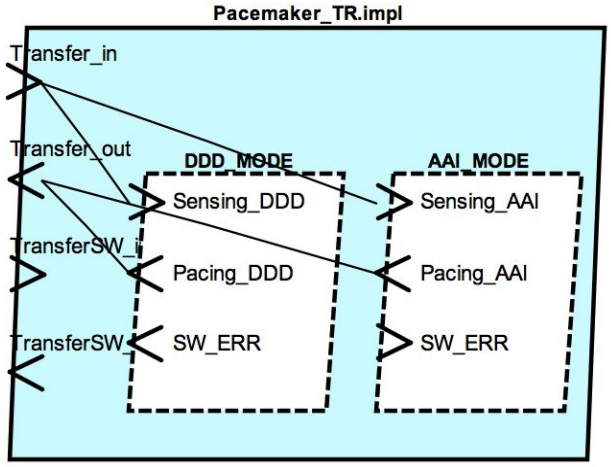
Before modification：

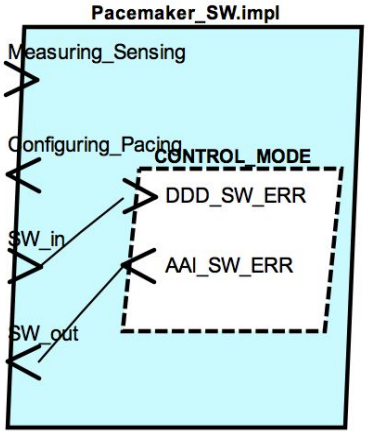


After modification：



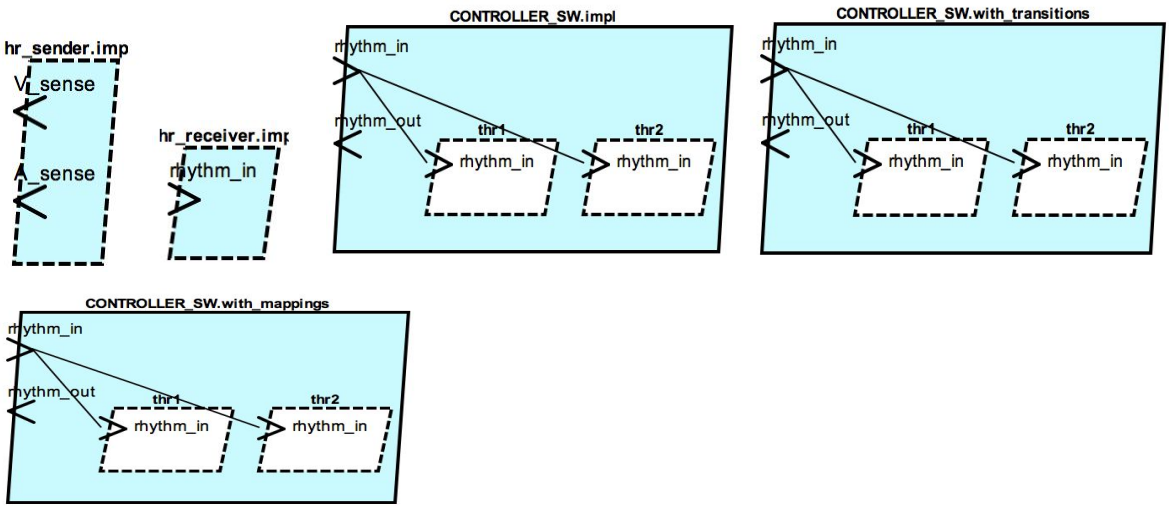




In this way, not only the expandability and testability of maintainability, but also the interoperability of functionality have been improved in the Device Controller Monitor system.

**Tactic 3: Augmenting**

Two additional threads and three additional processes are added to the system in order to implement a feedback control loop including nominal and error flows.




In this way, the completeness, correctness and compatibility of functionality in the entire pacemaker system have been improved.

Table 3: Example of Scenario

| | |
|---|---|
| Scenario: | When a Pulse Generator senses an abnormal decrease in the heart rate, it starts generating pulses in less than one millisecond. |
| Attribute: | Reliability |
| Stimulus: | The heart rate of a patient decreased abnormally. |
| Response: | The Pulse Generator starts generating pulses. |

| **Architectural decision** | **Sensitivity** | **Trade-off** | **Risk** | **Non-risk** |
|---|---|---|---|---|
| Feedback Control Loop | Using ontologies allows for better pacemaker model scalability but it makes the architecture very sensitive to change as the change propagates to behaviour and adapters. | Reliability (+) vs. Performance (-) | | |
| Augmenting | Concern over network latency | Modifiability (-) vs. Performance (+) | Data integrity | Should stay compatible |

**Phase 4: Reporting**

**Step9: Present results**

The ATAM that we performed on the architecture for the Pacemaker System revealed some potentially serious problems in the documentation of the architecture, the clarity of its requirements, its performance, its availability, and a potential architectural tradeoff. We will briefly summarize each of these problems in turn.

**Documentation**

The documentation provided at the inception of this project was minimal: two pages of diagrams that did not correspond to software artifacts in any rigorous way. This is, in our experience, typical, and is the single greatest impediment to having a productive architecture evaluation. Having a distinct Phase 1 and having the opportunity to request additional documentation from the contractor made the evaluation successful. As a result of our interaction with the Pacemaker System contractor team, substantially augmented and higher quality architectural documentation was produced. This improved documentation became the basis for the evaluation. And the improvement in the documentation was identified by management as a major success of the ATAM process, even before we presented any findings.

**Requirements**

One benefit of doing any architectural evaluation is increased stakeholder communication, resulting in better understanding of requirements. Frequently, new requirements surface as a result of the evaluation. The Pacemaker System experience was typical, even though the requirements for this system were "frozen" and had been made public for over two years. For example, in the Pacemaker System the only performance timing requirements were that the system be ready to operate in five minutes from power-on. In particular, there were no timing requirements for other specific operations of the system, such as responding to a particular order, or updating the environment database. These were identified as lacking by the questions we asked in building the performance model.

**Sensitivities and Tradeoffs**

The most important trade off identified for the Pacemaker System was the communications load on the system, as it was affected by various information exchange requirements and availability schemes. The overall performance and availability of the system is highly sensitive to the latency of the (limited and shared) communications channel, as controlled by the parameters n and m. Not only should the current performance characteristics be modeled, but also the anticipated performance changes in the future as the system scales in its size and scope.

**Architectural Risks**

In addition to the sensitivities and tradeoffs, we discovered, in building the models of the Pacemaker System's availability and performance, a serious architectural weakness that had not been previously identified: there exists the possibility of an opposing force identifying the distinctive communication pattern between the PG and the DCM and thus targeting those nodes specifically. The PG and DCM exchange far more data than any other nodes in the system. This identification can be easily done by an attacker who could discern the distinctive pattern of communication between the PG and DCM, even without being able to decrypt the actual contents of the messages. Thus, it must be assumed that the probability of failure for the PG and DCM increases over the duration of a mission under the existing Pacemaker System architecture.

**Conclusions**

An architecture analysis method, any architecture analysis method, is a garbage-in-garbage out process. The ATAM is no different. It crucially relies on the active and willing participation of the stakeholders (particularly the architecture team); some advance preparation by the key stakeholders; an understanding of architectural design issues and analytic models; and a clearly articulated set of quality attribute requirements and a set of business goals from which they are derived. Our purpose in creating a method (rather than, say, just putting some intelligent and experienced people together in a room and having them

chat about the architecture or inspect it in an arbitrary way) is to increase the effectiveness and repeatability of the analysis.

The architectures of substantial software-intensive systems are large and complex. They involve many stakeholders, each of whom has their own agenda. These architectures are frequently incompletely thought out or only partially documented. A method for architecture evaluation has to consider and overcome all of these daunting challenges. Our techniques are aimed at taming this considerable complexity and ensuring the achievement of the main goals for an ATAM: to obtain a precise statement of the quality attribute requirements; to understand the architectural decisions that have been made; and to determine if the architectural decisions made adequately address the quality attribute requirements.