A Survey on Software Architecture Analysis Methods

Liliana Bobrica and Eila Niemela
IEEE TOSE July 02

What is SBAR?

• Abbreviation for Scenario-based Architecture Re-engineering

• “SBAR estimates the potential of the designed architecture to reach the software quality requirements.”
  • L. Dobrica: A Survey on Software Architecture Analysis Methods

Importance of SBAR

• A system is never a pure real-time system, or a fault-tolerant system, or a re-usable system.

• Single non-functional requirement (NFR) is not a satisfactory measurement, since NFRs often conflict.

• In a realistic system a balance of NFRs is needed for an accurate assessment of a software architecture.

Assessing Quality Attributes

1. Scenarios
2. Simulation
3. Mathematical Modeling
4. Experience-based Reasoning

An Example.....

• Beer Can Inspection System:
  • To illustrate the architecture reengineering method, a beer can inspection system is used.
  • The inspection system is placed at the beginning of beer can filling process and its goal is to remove dirty beer cans from the input stream. Clean cans should pass the system without any further action.
  • The system consists of a triggering sensor, a camera and an actuator that can remove cans from conveyer belt.
Functions

- When a can is detected, the system receives a trigger from a hardware trigger.
- After a predefined amount of time, the camera samples an image of the can. This sampling is repeated a few times and subsequently the measured images are compared to ideal images and a decision about removing or not removing the can is made.
- If the can should be removed, actuator is invoked at a point in time relative to the point in time when the trigger event took place.

Object Model

Author's Experience

- Generally we handle S/W quality requirements by an informal process.
- If found short-comings, then re-design iteratively over system development, but this proves very costly.
- S/W quality requirements often conflict
  - Real-time Vs Reusability
  - Flexibility Vs Efficiency
  - Reliability Vs Flexibility
- Conventional design methods focus on a single quality attribute and treat all others as having secondary importance.

Architecture Re-engineering Method

- S/W engineers need to balance the various quality attributes for any realistic system.
- The authors propose an architectural re-engineering method that provides an objective approach.

Method Outlined

1. Incorporate new functional requirements in the architecture
2. Software quality assessment
3. Architecture transformation
4. Software quality assessment
### Assessment

#### Assessing Software Quality Requirements

1. **Scenario-based evaluation**: Develop a set of scenarios that concretize the actual meaning of the attribute. Useful for development related S/W qualities like reusability and maintainability.
2. **Simulation**: Complements scenario-based evaluation. Is useful for evaluating operational software qualities like performance or fault-tolerance.
3. **Mathematical Modeling**: Allows for static evaluation of architectural design models.
4. **Experience-based reasoning**: Evaluation process is less explicit and more based on subjective factors as intuition and experience.

### Transformation

#### Iterative Steps:

- Complete architecture design.
- Compare with the requirements.
- Then update architecture.

Note:
- The transformations made are minor.
- The functionality does not change, only the quality attributes change.
- It is not feasible to start bottom-up during design and reengineering.

### Different Approaches

- Impose architectural style. e.g., layered architectural style
- Impose architectural pattern.
- Apply design pattern.
- Convert quality requirements to functionality.
- Distribute requirements.

### S/W Quality Requirements

- Functional requirements generally can be evaluated relatively easy by tracing the requirements in the design.
- On the other hand, S/W quality requirements are much harder to assess.
- Few such quality requirements are:
  - Reusability
  - Maintainability
  - Real-time
  - Robustness
- As mentioned previously, development related S/W qualities are easiest assessed using scenarios.

### Reusability

- This quality attribute should provide a balance between generality and specifics.
- The architecture and its components should be general since they should be applied in other similar situations.
- The architecture should provide concrete functionality that provides considerable benefit when it is reused.
- Five scenarios that are tested in this article:
  - R1: Product packaging quality control
  - R2: Surface finish quality control
  - R3: Quality testing of micro-processors
  - R4: Product sorting and labeling
  - R5: Intelligent quality assurance system

### Maintainability

- The goal here is that the most likely changes in requirements are incorporated in the software system against minimal effort.
- Five scenarios that are tested in this article:
  - M1: The types of input or output devices used in the system is excluded from the suppliers assortment and need to be changed, by the S/W.
  - M2: The S/W needs to be modified to implement new calculation algorithms.
  - M3: The method of calibration is modified.
  - M4: The external systems interface for data exchange change.
  - M5: The hardware platform is updated, with new processor and I/O interface.
**Applying SBAR**

Iterative process until quality requirements are met:
- Evaluate software quality attributes of the application architecture
- Identify the most prominent deficiency
- Transform the architecture to remove the deficiency

**Evaluation**

How much re-use is possible?
- How much will I be able to reuse the software
- Ratio of Re-used components 'as-is' to the total number of components
- As close to 1 as possible
- Presence of high coupling limits the possibility of re-use

**Evaluation**

Effort needed to maintain
- How easy is it to fix
- Ratio of Affected components to Total components
- As close to 0 as possible
- Changes usually require many components to be modified

**Transformations**

**Component-level**

Problem: New item type requires the source code of most components to be changed
Transformation: specific type → generic type
Result: Improves reusability and maintainability

**Transformations**

**Abstraction**

Problem: Type dependence at component creation
Transformation: Use Abstract Factory pattern
Results: Improves maintainability

**Transformations**

**Choose Strategy**

Problem: Changes have to be made in every component performing similar task
Transformation: Apply the Strategy pattern
Results: Gained maintainability outweighs loss in reusability
Decrease Dependence on Global State

Problem:
Calibration of the measurement system

Transformation:
Introduce calibration strategy

Results:
Improves maintainability

Reduce Coupling between calibration & measurement

Problem:
Coupling between calibration strategy and the measurement item.

Transformation:
Apply Prototype design pattern.

Results:
Improves maintainability and reusability.

Evaluation

- Overall, the result from the transformations is satisfying and the analysis of the scenarios shows substantial improvement (author’s conclusion).

- Each iteration seems to solve a problem concerning some attribute. The drawback may be that, we do not have a prior idea of how many iterations it is going to take.

- Identifying all possible problems that may lead to difficulties in re-use and maintainability is a challenging task in itself.

References

- *A Survey on Software Architecture Analysis Methods*, L. Dobrica

- *Scenario-based Software Architecture Re-engineering*, PerOlof Bengtsson & Jan Bosch
Architecture Level Prediction of Software Maintenance

PerOlof Bengtsson & Jan Bosch
University of Karlskrona/Ronneby
Sweden

Presented by group 4 & 9
1. Manish Sharma
2. Walamitien Oyenan
3. John Hoestje
4. Nur Shahabudin
5. Lakshmikanth Ganti
6. Daniel Clemons
7. Lori Frank

CIS 740
Instructor: Dr. David A. Gustafson

Introduction

- Software Architecture
  - Sets the boundaries for the software quality
  - Balance between performance, reliability and maintainability
  - Improve adaptability to changes

Goal

- A method for the prediction of software maintainability during software architecture design considering:
  - Requirement Specification
  - Architecture Design
  - Software Expertise
  - Historical Data

Software Maintainability as per IEEE

“The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment”
**Maintenance Prediction Method**

- Identify categories of maintenance tasks
- Synthesize scenarios
- Assign each scenario a weight
- Estimate the size of all elements
- Script the scenarios
- Calculate the predicted maintenance effort

---

**Identify categories of maintenance tasks**

Maintenance tasks focuses on:
- Corrective maintenance
- Adaptive maintenance

Categories are defined based on:
- Application
- Domain description

---

**Synthesize scenarios**

- Set of concrete scenarios is defined for each categories
- Scenarios describes an action, or sequence of actions that might occur related to the system
- Selected by
  - Domain expert
  - Software architect
- Number of scenarios are dependent on the application and domain

---

**Assign Each Scenario A Weight**

- The weight of a scenario is the relative probability of a scenario to occur during the lifetime of the system, which results in a maintenance task
- More recurring scenarios are weighted more
- Weight are based on
  - Historical Data
  - Software Architect or domain expert estimations

---

**Estimate the size of all elements**

- Maintenance Efforts are based on:
  - Estimation techniques of choice (LOC)
  - Metric
  - Historical data from similar applications

---

**Script the scenarios**

- Scenarios are scripted to evaluate
  - Affected components
  -Extent of change
Calculate the predicted maintenance effort

- Prediction value is a weighted average for the effort for each maintenance scenario

\[ M_{pri} = \sum_{i=1}^{k} \left( P_{S_i} \cdot \sum_{n=1}^{l} W_{S_i} C_{n} \right) \]

- \( P_{S_i} \) is the probability weight of scenario \( i \)
- \( W_{S_i} \) is the affected volume of component \( S_i \) in scenario \( i \)
- \( l \) = number of scenarios
- \( k \) = number of components in architecture

Example

Haemo Dialysis Machine

Maintenance Profile

<table>
<thead>
<tr>
<th>Category</th>
<th>Scenario Description</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medical Drives</td>
<td>C3: Change measurement units from Fahrenheit to Celsius in a treatment.</td>
<td>0.943</td>
</tr>
<tr>
<td>Hardware</td>
<td>C8: Replace blood pumps using revolutions per minute with pumps using actual flow rate (ml/s).</td>
<td>0.087</td>
</tr>
<tr>
<td>Safety</td>
<td>C9: Add alarm for reverse flow for catheters.</td>
<td>0.957</td>
</tr>
<tr>
<td>Medical Advances</td>
<td>C10: Change alarm from high flow limits to follow treatment.</td>
<td>0.067</td>
</tr>
<tr>
<td>Medical Advances</td>
<td>C11: Add sensor and monitor for patient blood pressure.</td>
<td>0.967</td>
</tr>
</tbody>
</table>

Architecture

Requirements of Haemo Dialysis Machine

Software Quality Requirements
- Maintainable
- Reusable
- Safe
- Real-timeliness
- Demonstratable
Estimated Component Size

<table>
<thead>
<tr>
<th>Component</th>
<th>Size (LOC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDFTreatment</td>
<td>260</td>
</tr>
<tr>
<td>Frame/AnalysisEngine</td>
<td>560</td>
</tr>
<tr>
<td>ConcentrationDevice</td>
<td>160</td>
</tr>
<tr>
<td>TemperatureDevice</td>
<td>160</td>
</tr>
</tbody>
</table>

Impact Analysis per Scenario

Calculations

\[0.043 \times 60 + 0.043 \times 127.5 + 0.087 \times 350 + 0.174 \times 10 + 0.217 \times 100 + 0.087 \times 190 + 0.087 \times 350 + 0.087 \times 120 + 0.043 \times 290 + 0.132 \times 100 = 145 \text{ LOC / change}\]

- 20 Maintenance tasks estimated
- Productivity 1.7 LOC/Day or 0.2 LOC/hour
- 20 Changes per 145 LOC = 2900 LOC
- \(\frac{2900}{0.2} = 14,500\) hours of effort
- This would represent a medium project of about 6-7 people working around 2300 hours per year

Conclusion

Advantages:
- Practical
- Improved project understanding
- Accurate prediction of scenarios using combination of historical data and design expertise
- Efficiently produce artifact with respect to time
- Suitable for iterative design processes

Conclusion

Our opinion:
- Being able to predict early in the software lifecycle helps to produce more quality software
- So architecture level prediction of software maintenance is a very useful method
- However the some estimation will be hard to predict without prior experience and empirical data
A Software Architecture Evaluation Model

SAEM

Presented by:
Jeff Hank, Ping Li, Zhihong Zeng
Gregory Matthews, Yong Peng

Summary

- SAEM is based on ISO/IEC 14598-1 quality evaluation model
- SAEM implements this standard as a means of predicting quality of a product based on its SA.
- SAEM identifies quality attributes
  - External quality expresses the user’s view
  - Internal quality expresses the developer’s view

Presentation Outline

- Context in terms of Dobrica’s Survey of SA Analysis Methods
- Explain The Software Architecture Evaluation Model
- Walkthrough a Quick Example
- Evaluation of the SAEM as an SA Analysis Method

Main elements of Dobrica’s comparison framework

Goal of SAEM

- Establishes the basis for the SA quality evaluation.
- Attempts to predict the quality of the final system.

Quality Model (SAEM)

- The number of attributes can vary with the quality model selected and the attributes deemed important within that model
EXISTENT KNOWLEDGE (SAEM)

- Not applied but assumed to exist in order to define the expected internal attributes and their target values.

- Experts’ knowledge and a company’s accumulated data are used in the mapping of quality requirements to internal attributes (user specified qualities mapped to the internal attributes that must satisfy the user quality requirements).
- The development team specifies all the quality requirements.

STAKEHOLDERS INVOLVEMENT (SAEM)

- Stakeholders’ involvement in the development process.
- The SAEM technique involves the evaluation of software artifacts.

THE EVALUATION TECHNIQUE (SAEM)

- Measuring technique using intrinsic or defined metrics.

SA DESCRIPTION (SAEM)

- Considered from two viewpoints, the developer and the user.
- The SA is considered either an intermediate or final product in the software system process.
- The intrinsic properties can only be detected by measuring techniques applied to the SA representation formalized through an ADL (Architecture Description Language).

ADL (ARCHITECTURAL DESCRIPTION LANGUAGE)

- **What is ADL?**
  - Is a kind of language used to describe a software architecture (an ADL may be a formal or semi-formal descriptive language, a graphics language, or include both).

- **Why use ADL?**
  - Lends the ability to rigorously specify an architecture so that it can be analyzed.
  - To improve software quality and correctness.

- **How many different ADLs?**
  - A lot
  - The list of different ADLs.

SAEM’S VALIDATION

- SAEM has not been validated on any software system.
- Duenas’ paper on SAEM suggests the method was to be validated on a telecommunication package, but no results have ever been published. Nor have any other examples of the application of SAEM been published.
The Software Architecture Evaluation Model

- From ISO/IEC 14598-1
  - The quality model
  - The method for evaluation
  - The metrics
  - The supporting tools
- SAEM implements this standard as a means of predicting quality of a product based on its SA.

Elements Required for SA Quality Evaluation

- From ISO/IEC 14598-1
  - The quality model
  - The method for evaluation
  - The metrics
  - The supporting tools
- SAEM implements this standard as a means of predicting quality of a product based on its SA.

Elements of the ISO 9126-1 Quality Reference Model

- The Quality Model: ISO/IEC 9126-1
- The Quality Specification: ISO 8402 and/or Quality Function Deployment (QFD)
- The Metrics Specification: from intrinsic metrics and/or defined by Goal-Question-Metric (GQM)
- The Quality Evaluation: data collection, measurement, analysis of results

1: The Quality Model: ISO 9126-1

- What is it?
  - ISO 9126-1 is the software product evaluation standard from the International Organization for Standardization. This international standard defines six characteristics that describe, with minimal overlap, software quality.
- What is the benefit?
  - ISO 9126-1 intends to eliminate any misunderstanding between purchaser and supplier.

Functionality

**Functionality** is the set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.
Reliability

Reliability is the set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.

Usability

Usability is the set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.

Efficiency

Efficiency is the set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.

Maintainability

Maintainability is the set of attributes that bear on the effort needed to make specified modifications.

Portability

Portability is the set of attributes that bear on the ability of software to be transferred from one environment.

2. Quality Specification

ISO 8402

- The quality specification contains the totality of the characteristics bearing on the product’s ability to satisfy the stated and implied needs.
- The user view is expressed in the external quality specification.
- The developer is express in the internal quality specification.
Quality Function Deployment (QFD) is a structured approach to defining customer needs or requirements and translating them into specific plans to produce products to meet those needs. The "voice of the customer" is the term to describe these stated and unstated customer needs or requirements. The voice of the customer is captured in a variety of ways: direct discussion or interviews, surveys, focus groups, customer specifications, observation, warranty data, field reports, etc. This understanding of the customer needs is then summarized in a product planning matrix or "house of quality". These matrices are used to translate higher level "whats" or needs into lower level "hows" - product requirements or technical characteristics to satisfy these needs.

**QFD (Quality Function Deployment)**

**What is QFD**
- Is a systematic process for motivating a business to focus on its customers

**Why use QFD**
- To improve the communication of customer wants throughout the organization
- To improve the completeness of specifications and to make them traceable directly to customer wants and needs

**What are the basic tools of QFD**
- Project Roadmaps, Documents, Lists and Matrices

**What are the approaches to QFD**
- Four-Phase approach: uses a QFD matrix to translate customer wants into Product Characteristics
- Matrix of Matrices: used to address a wide variety of development issues
- etc.

### 3: Metrics Specification

- **The Metrics Specification must contain:**
  - The selected measure for each quality characteristic
  - A measurement scale
  - The set of methods for measurement
- Divided into external and internal metrics specification
- Metrics can be determined with a tool such as the Goal-Quality-Metric method

#### External Metrics Specification

- **Purpose:**
  - Provide data for the evaluation of SA as a product itself, focusing on its usage during development.
- **Metrics:**
  - Have to be evaluated as part of the working environment for the intended usage.
  - Eg. Mean time for analysis = total analysis times / number of analyses

#### Internal Metrics Specification

- **Purpose:**
  - Provide data for the SW product evaluation regardless of its environment and is used to evaluate product quality before its usage.
- **Metrics:**
  - Special elements: Eg. Data encryption ratio = number of encrypted components / specified number of components requiring encryption
  - Intrinsic properties: Eg. Size, complexity, coupling, modularity, cohesion.

**GQM (Goal-Question-Metric)**

- **GQM:**
  - Used to define measurement on the software project, process, and product in such a way that 1) resulting metrics are tailored to the organization and its goal, 2) resulting measurement data play a constructive and instructive role in the organization and 3) metrics and their interpretation reflect the values and the viewpoints of the different groups affected (e.g., developers, users, operators).

- **Three steps:**
  1. to define the goal in terms of purpose, perspective and environment
  2. to establish the questions that indicate the attributes related with the goal
  3. to answer each question and propose new existing metrics.
**GQM (Goal-Question-Metric) 2**

What are the GQM phases:

1. The Planning phase, during which the project for measurement application is selected, defined, characterized, and planned, resulting in a project plan.
2. The Definition phase, during which the measurement program is defined (goal, questions, metrics and hypotheses are defined and documented).
3. The Data collection phase, during which the actual data collection takes place, resulting in collected data.
4. The Interpretation phase, during which the collected data is processed with respect to the defined metrics into measurement results, that provide answers to the defined questions, after which goal attainment can be evaluated.

---

**4: Quality Evaluation**

- Data collection, measurement, analysis of results
- End result: the measurement result can be used as feedback for the improvement of the SA development process.

---

**Example**

1. Select the quality model and the characteristics of interest
2. Define the sub-characteristics in terms of the external and internal views
3. Define the attributes to measure the internal and external characteristics
4. Define the metrics for the attributes
5. Select the target values for the attributes
6. Perform the quality evaluation on the measured values

---

**1. Select the quality model and the characteristics of interest:** ISO 9126-1

- Reliability
- Usability
- Maintainability

---

**in terms of the external and internal views: ISO8402**

<table>
<thead>
<tr>
<th>External Characteristic</th>
<th>External Sub-Characteristic</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>Robustness</td>
<td></td>
</tr>
<tr>
<td>Usability</td>
<td>Understandability</td>
<td></td>
</tr>
<tr>
<td>Maintainability</td>
<td>Modifiability</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>External Characteristic</th>
<th>Internal Sub-Characteristic</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintainability</td>
<td>Modifiability</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--Complexity modifiability</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--Coupling</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Complexity modifiability</td>
<td></td>
</tr>
</tbody>
</table>

---

**measure the internal and external characteristics**

<table>
<thead>
<tr>
<th>External Characteristic</th>
<th>Internal Sub-Characteristic</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>Robustness</td>
<td></td>
</tr>
<tr>
<td>Usability</td>
<td>Understandability</td>
<td></td>
</tr>
<tr>
<td>Maintainability</td>
<td>Modifiability</td>
<td></td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>External Characteristic</th>
<th>Internal Sub-Characteristic</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintainability</td>
<td>Modifiability</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--Complexity modifiability</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--Modifiability</td>
<td></td>
</tr>
</tbody>
</table>

---

**Examples**

Time spent to modify an existing GUI functionality

Time spent to understand the functionality offered by the GUI
4. Define the metrics for the attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Metric</th>
<th>Time spent to modify an existing GUI functionality</th>
<th>Modifiability</th>
<th>Maintainability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>Robustness</td>
<td>Time spent to understand the functionality offered by the GUI</td>
<td>Understandability</td>
<td>Usability</td>
</tr>
<tr>
<td>Variability</td>
<td>Modifiability</td>
<td>Time spent to modify an existing GUI functionality</td>
<td>Maintainability</td>
<td>Robustness</td>
</tr>
</tbody>
</table>

5. Select the target values for the metrics

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Metric</th>
<th>Time spent to modify an existing GUI functionality</th>
<th>Modifiability</th>
<th>Maintainability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>Robustness</td>
<td>Time spent to understand the functionality offered by the GUI</td>
<td>Understandability</td>
<td>Usability</td>
</tr>
<tr>
<td>Variability</td>
<td>Modifiability</td>
<td>Time spent to modify an existing GUI functionality</td>
<td>Maintainability</td>
<td>Robustness</td>
</tr>
</tbody>
</table>

6. Perform the quality evaluation on the measured values

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Metric</th>
<th>Measured Value</th>
<th>Target Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>Robustness</td>
<td>Percentage</td>
<td>100%</td>
</tr>
<tr>
<td>Variability</td>
<td>Modifiability</td>
<td>Level</td>
<td>100</td>
</tr>
<tr>
<td>Maintainability</td>
<td>Modifiability</td>
<td>Level</td>
<td>100</td>
</tr>
</tbody>
</table>

Analysis of SAEM

✿ Is SAEM useful?

On paper, SAEM would appear useful in demonstrating a quantifiable level of SA quality.

However, in practice, it’s never been used.

✿ What’s good about SAEM?

• It’s adaptable
• Uses established standards and proven analysis tools such as ISO documents, QFD, GQM

✿ What’s bad about SAEM?

• SAEM has never been validated and therefore not used
In Conclusion

SAEM does not establish optimal values for the measured quality attributes, but just formalizes a framework providing the means to measure these values and predict the quality of the final product from the SA evaluation.

REFERENCES

  - http://www.qfd.org/