



Quality Attribute Driven Software Architecture Reconstruction

SATURN Workshop April 7, 2005

Liam O'Brien

Sponsored by the U.S. Department of Defense
© 2005 by Carnegie Mellon University



Motivation - 1

- Software architectures are critical to implement an organization's business goals and critical for intellectual property.
- Industry demands to:
 - Evolve existing products into Product Lines
 - Evaluate existing systems to improve response to quality attributes
 - Check conformance of the implementation to design
 - System Modernization
- Disconnect: Few organizations are willing to pay for an architecture reconstruction effort.

Motivation - 2

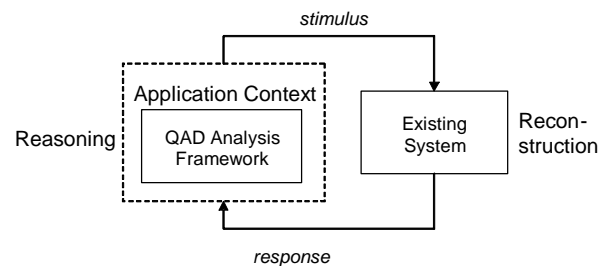
- Architecture Reconstruction is an embedded activity in a larger effort in an organization.
- Business goals are primarily incorporated as quality goals that shape the software architecture of a product.
- The analysis of software architectures is quality attribute driven.

=> QADSAR
Quality Attribute Driven Software Architecture
Reconstruction

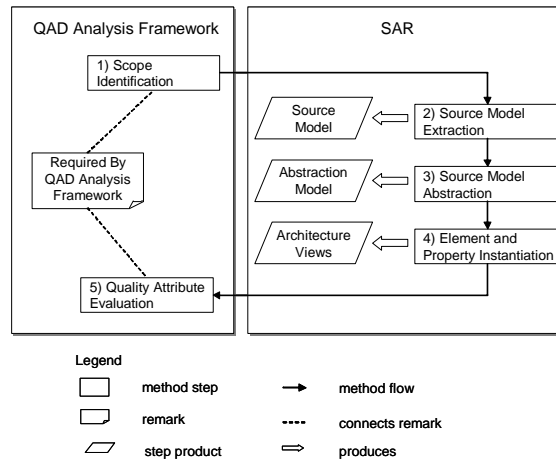
QADSAR Approach

Characteristics

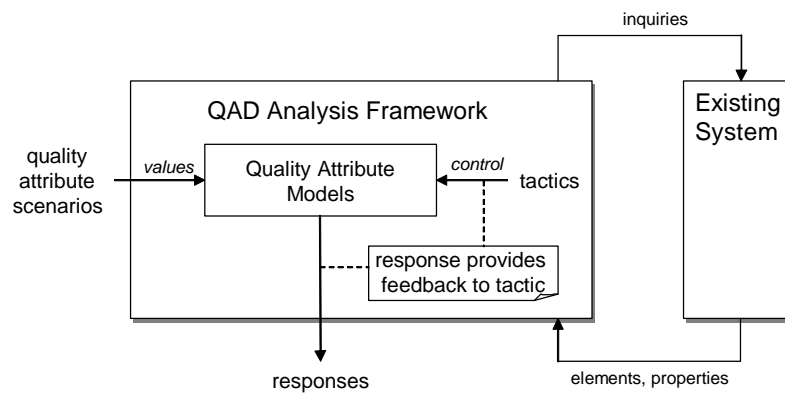
1. Goal-Driven Reconstruction (what to reconstruct?)
2. Stimulus/Response Approach
3. Requires Quality Attribute Reasoning Frameworks



QADSAR – SAR Part



QADSAR – QAD Part



Case Study – Overview - 1

A case study in an automotive embedded system

64KLOC system, written in “C”

Case study was part of a larger effort for a Product Line Migration

The reconstruction was performed in the software reconstruction environment ARMIN. ARMIN jointly developed by the SEI and Robert Bosch Corporation.

Case Study – Overview - 2

The software consists of three packages from different vendors:

1. Boot Loader
2. Communication
3. Application

The communication package is typically predetermined by the Original Equipment Manufacturer (OEM) and has to be deployed by all suppliers.

One task is to investigate the modifiability of the application software with regards to incorporating other communication software packages.

Quality Attribute Analysis

A Quality Attribute of Interest:

Modifiability

Scenario:

The organization has to replace the communication package from vendor 1 by a package from vendor 2 in one day.

Modifiability Model - 1

Modifiability is strongly influenced by different types of dependencies.

A dependency among modules exists, if a modification to some aspects of module A requires a modification in module B to accommodate the modification to module A.

We say:

Module A depends in some way on Module B.

Modifiability Model - 2

Types of Dependencies between modules A and B:

- Syntax Dependencies
 - *can be either data (type/format of data is consistent) or service (signatures of services are consistent) related*
- Semantics Dependencies
 - *can be either data or service related*
- Sequence-of-use dependencies
 - *can be either data or control related*
- Interface identity dependencies
 - *interfaces between modules must be consistent (name)*
- Runtime location dependencies
 - *must be consistent (same of different processor or located within same process)*
- Quality-of-service or quality of data dependencies
 - *involves the service or data provided by the modules*
- Existence-of-module dependencies
 - *either must be present for the other to function properly*
- Resource behavior dependencies
 - *relates to such issues as memory usage, resource ownership between the modules*

Modifiability Model - 3

Types of Dependencies:

- **Syntax Dependencies**
 - Module view (both data and functions with parameters)
- **Semantics Dependencies**
 - Difficult to extract (analysis of denoted interfaces with semantic descriptions)
- **Sequence-of-use dependencies**
 - Dataflow views (for data) – Interaction diagrams or state machine views (for service)
- Interface identity dependencies
 - Not of relevance in this context
- **Runtime location dependencies**
 - Deployment view
- Quality-of-service or quality of data dependencies
- Existence-of-module dependencies
- Resource behavior dependencies

Modifiability Tactics

Modifiability might be achieved in a software system by different architecture tactics. Tactics for modifiability include:

- Maintain semantic coherence
- Isolate expected changes
- Hide information

Other Modifiability tactics include:

- Use a virtual machine
- Limit communications paths
- Abstract common services

Common to the tactics are strategies to reduce and manage the dependencies between modules.

Reconstruction for Modifiability

The tactics help us in the reconstruction process to identify what to look for in the system and to create particular views

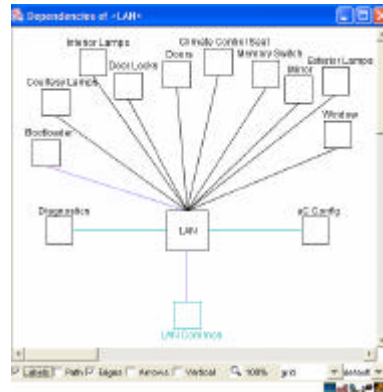
Applied in this context:

- Maintain semantic coherence – did not evaluate because the communication package is already determined to be separated by the Original Equipment Manufacturer
- Isolate expected changes – is there something actively done in the application package to mitigate changes
- Hide information – e.g. is there an explicit model to separate interfaces from their realization

Module View – showing LAN

Response measure for the scenario – ‘number of modules affected’.

Generate a dependency layout for the LAN part of the communications package and analyze the application modules that access the LAN module.



Case Study Analysis

The two tactics ‘isolate expected changes’ and ‘hide information’ reduce the dependencies and the number of modules affected by changes.

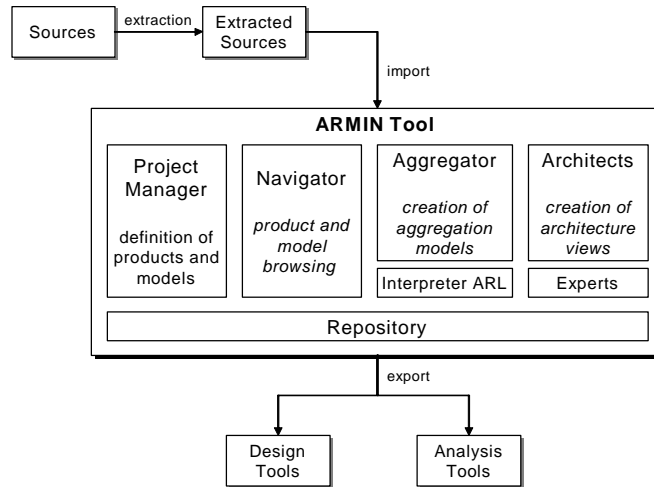
Both tactics were not identifiable from our analysis

No explicit notion of interface identifiable – no separation of interface and implementation. Several attempts to capture an interface by different aggregations failed.

Analysis with the developers identified that changes are likely to be expected in more than a dozen modules.

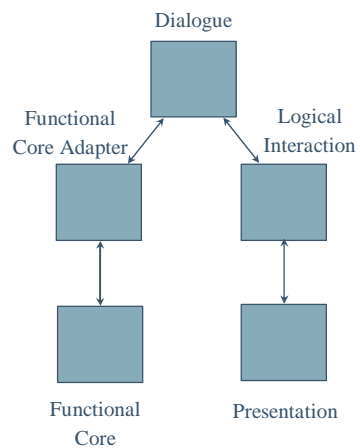
Resulted in an improvement effort of the existing system to support the quality attribute scenario sufficiently.

ARMIN Demo



Example: VANISH

VANISH was designed with specific modifiability goals.



Conclusions

- QADSAR establishes the link between quality attribute driven analysis and architecture reconstruction
- The business goal and quality attribute driven approach of system understanding provides a means to steer the reconstruction process by providing a set of required views that are needed for a particular system
- Architecture tactics can infuse the reconstruction and analysis process to measure the response for a particular quality attribute scenario. The response measure is linked back to the business goal of the system.
- Further Quality Attribute Models and their related tactics can help to drive further connections between reconstruction and quality attribute analysis.

Future Work

Some future work:

- Investigate the relationships between the quality attribute analysis and the architecture views that would need to be reconstructed to support their analysis
- Further develop ARMIN for QADSAR with export mechanisms to use specialized analysis tools for particular quality attributes, such as, performance tools.

Quality-Attribute-Driven Software Architecture Reconstruction
Liam O'Brien

Abstract:

Architecture reconstruction is the process by which architectural views of an implemented system are obtained from existing artifacts. This presentation outlines research on architecture reconstruction that is driven by quality attribute analysis. The analysis typically occurs when existing systems hit their architectural boundaries caused by product growth or expansion scenarios. The information gathered during architecture reconstruction has to satisfy the information needs for these scenarios in order to provide reasoning during decision-making processes. The work is crucial for organizations that have to make architectural decisions about existing systems, or want to lower the adoption barriers for product lines by investigating the reuse of existing assets.

Bio:

Dr Liam O'Brien is a Senior Member of the Technical Staff at the Software Engineering Institute. His primary research interests are architecture reconstruction, reengineering, service-oriented architectures, asset reuse, migration planning and enterprise integration. Before joining the SEI, Liam was a Research Engineer at CSIRO (Commonwealth Scientific and Industrial Research Organisation) and prior to that he worked as a software engineer at IMRglobal, both in Australia. He has a B.Sc. (Computer Systems, 1989) and a Ph.D. (Computer Science, 1996) both from the University of Limerick, Ireland.