A Mashup of Techniques to Create Reference Architectures

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Rick Kazman, John McGregor

© 2012 Carnegie Mellon University

Copyright 2012 Carnegie Mellon University.

This material is based upon work supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material has been approved for public release and unlimited distribution except as restricted below.

Internal use:* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use:* This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

*These restrictions do not apply to U.S. government entities.

The Context

Our client wanted to create a *reference architecture* to enable large-scale strategic reuse.

A major long-term effort in which assets are to be produced by a central team and used by distributed teams.

The distributed teams are independent of the central team.

This was a cultural as well as a SE challenge.

© 2012 Carnegie Mellon University

The Problem

The client needs an overarching architectural framework if the project is to be successful

- a set of standards, technologies, …
- more importantly a set of *rules* and *architectural approaches*.

Without the rules and architectural approaches the natural tendency—with large numbers of quasi-independent stakeholders, each with their own goals, budgets, and schedules—is towards anarchy.

© 2012 Carnegie Mellon University

The Techniques

We created a mashup of existing SEI architecture methods to begin address their goal of creating a reference architecture:

- QAW
- ADD
- Ecosystem modeling
- Reference Architecture
- Reference Architecture Documentation
- Continuous ATAM

ADD and Architecture Definition had to be adjusted to produce a reference architecture.

This required extensive mentoring.

© 2012 Carnegie Mellon University

Scope

The scope of the development planned by the various teams limits the applicability of the reference architecture.

In our case the scope was broadly stated in an ecosystem model.

Our reference architecture needed to address the requirements of all products in the ecosystem.

© 2012 Carnegie Mellon University

Ecosystem Modeling

For an ultra-large-scale system—a socio-technical ecosystem—to grow and flourish, it needs to enable creativity while minimally restricting developers and users.

The internet accomplishes this by only specifying interconnectivity standards, primarily protocols (e.g. IP). But the internet has no “goal”.

Commercial ecosystems accomplish this by providing a “platform” on which individual applications are built and deployed, e.g. iPhone, Android, Facebook, Eclipse, etc.

Our program is establishing an ecosystem around a centrally provided platform and a set of reusable assets.

© 2012 Carnegie Mellon University

Software Ecosystems

A software ecosystem has a “hub,” which provides a platform. In our case the core asset team will provide an platform consisting of an asset base and runtime environment.

Programs of Interest will obtain resources from, and contribute resources to, the ecosystem.

The organization at the hub can analyze the relations in the ecosystem and develop strategies that enhance ecosystem health.

Software ecosystems such as surrounding the open source Eclipse Foundation and that surrounding the commercial Microsoft use different governance models. Our client’s ecosystem has strict hierarchical control which can make ensuring architectural conformance easier.

© 2012 Carnegie Mellon University

Strategic Ecosystem Modeling for Decision Making

Each development team is the hub of a cluster of consumers and suppliers and will foster competitors and alternatives.

There may be overlaps where organization can be both producers and consumers of each others assets and products.

Strategic modeling is used to understand the ecosystem and to guide decisions to enhance that ecosystem.

© 2012 Carnegie Mellon University

Quality Attributes

For a reference architecture the quality attribute requirements have to be defined

- in a sufficiently broad manner to address the complete range of products to be covered by the architecture or
- in a narrowly focused manner in each of several configurations that address specific markets

We identified a few system categories that affected quality attributes:

- desktop
- mobile
- airborne

For example, the testability and security requirements of each of these will differ.

© 2012 Carnegie Mellon University

Inputs/Constraints

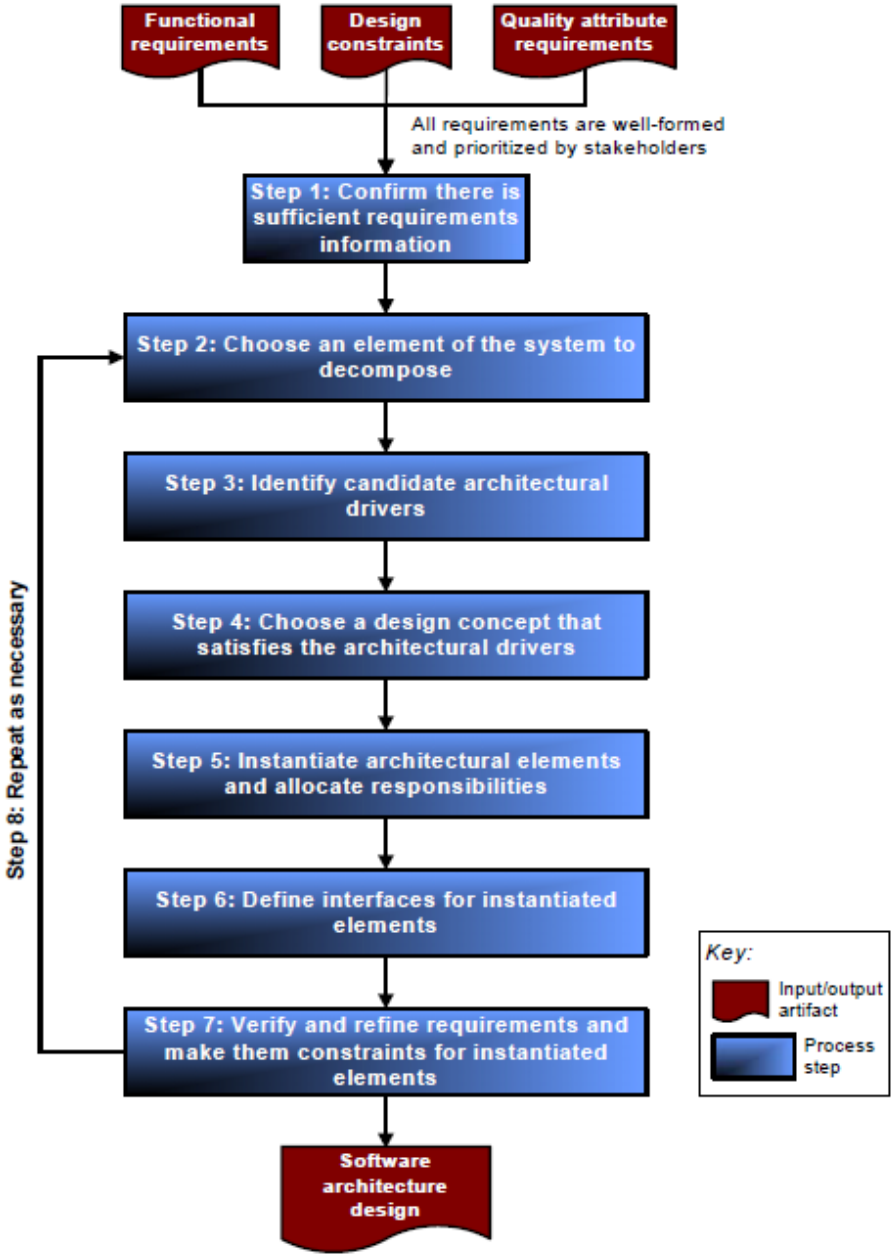
Inputs and constraints came from both general software engineering techniques and problem-specific information

- General
  - ADD technique definition
  - RUP definition
  - Documenting Software Architectures book
  - ISO/IEC/IEEEFDIS 42010

- Specific
  - QAW outputs, primarily quality attribute scenarios
  - Additional scenarios developed through white papers
  - Architectures from similar systems
  - Ecosystem model for the program

© 2012 Carnegie Mellon University

ADD Steps



© 2012 Carnegie Mellon University

Modifications to ADD

The emphasis of ADD becomes more “conceptual”.

The “elements” in the ADD description were described more abstractly. Quality attributes—but not specific quality goals—were identified.

Architecture patterns were described but not concretely instantiated.

Documentation notation (UML) was used: architecture elements are abstractions but crisply defined abstractions rather than vague notions.

© 2012 Carnegie Mellon University

Outputs from ADD

The output of ADD is a system design in terms of the roles, responsibilities, properties, and relationships among software elements.

- *software element*: a computational or developmental concept that fulfills roles and responsibilities, has defined properties, and relates to other software elements to compose the system architecture
- *role*: a set of related responsibilities
- *responsibility*: the functionality, data, or information that a software element provides
- *property*: additional information about a software element such as name, type, quality attribute characteristic, protocol, and so on
- *relationship*: a definition of how two software elements are associated with or interact with one another

© 2012 Carnegie Mellon University

“n” Week Review Cycle

We used ADD in an iterative, incremental manner

For each iteration:

- Assign a set of requirements
- Architecture sub-team:
  - designs a solution
  - documents it
  - presents it in a review following the peer review process
- Risks, sensitivities, tradeoffs and issues are analyzed and collected
- Revisions are planned
- Here we go again

Based on F. Bachmann’s “Give the Stakeholders What They Want: Design Peer Reviews the ATAM Style”, *Crosstalk*

© 2012 Carnegie Mellon University

A Starting Point for a Reference Architecture

The client has adopted the Eclipse Platform as the basis for a set of development environment products:

- It might release all of the SDK under the Eclipse Public License (EPL) setting up an environment where the client could freely provide the SDK to all teams.
- A new license could be established that restricts certain activities and gives client greater control. Paths through the ecosystem can be examined for license compatibility.

To ensure that the identified product qualities are actually achieved, information needs to be propagated through the ecosystem. Suppliers, whose products negatively impact quality, need to be notified of required quality levels.

The ecosystem can be the basis for organizational qualities such as performance or productivity.

© 2012 Carnegie Mellon University

Accomplishments

Two initial releases of the reference architecture have been made.

Two primary foci

- the basic environment – Eclipse Platform plus
- communication buses – 2 buses plus gateways to link instances of both

Overarching risks have been identified

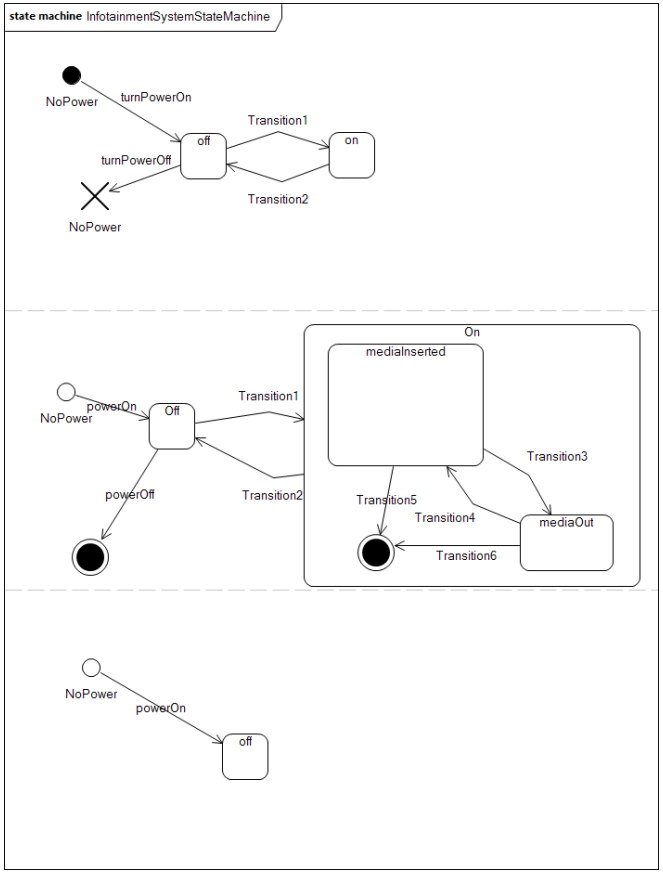
- if the scope is too narrowly defined there is a risk the architecture will be inadequate
- if the concepts in the reference architecture are too concrete the architecture may not be sufficiently flexible

© 2012 Carnegie Mellon University

Using UML

Even though the definitions in the reference architecture are conceptual they still need to be precise.

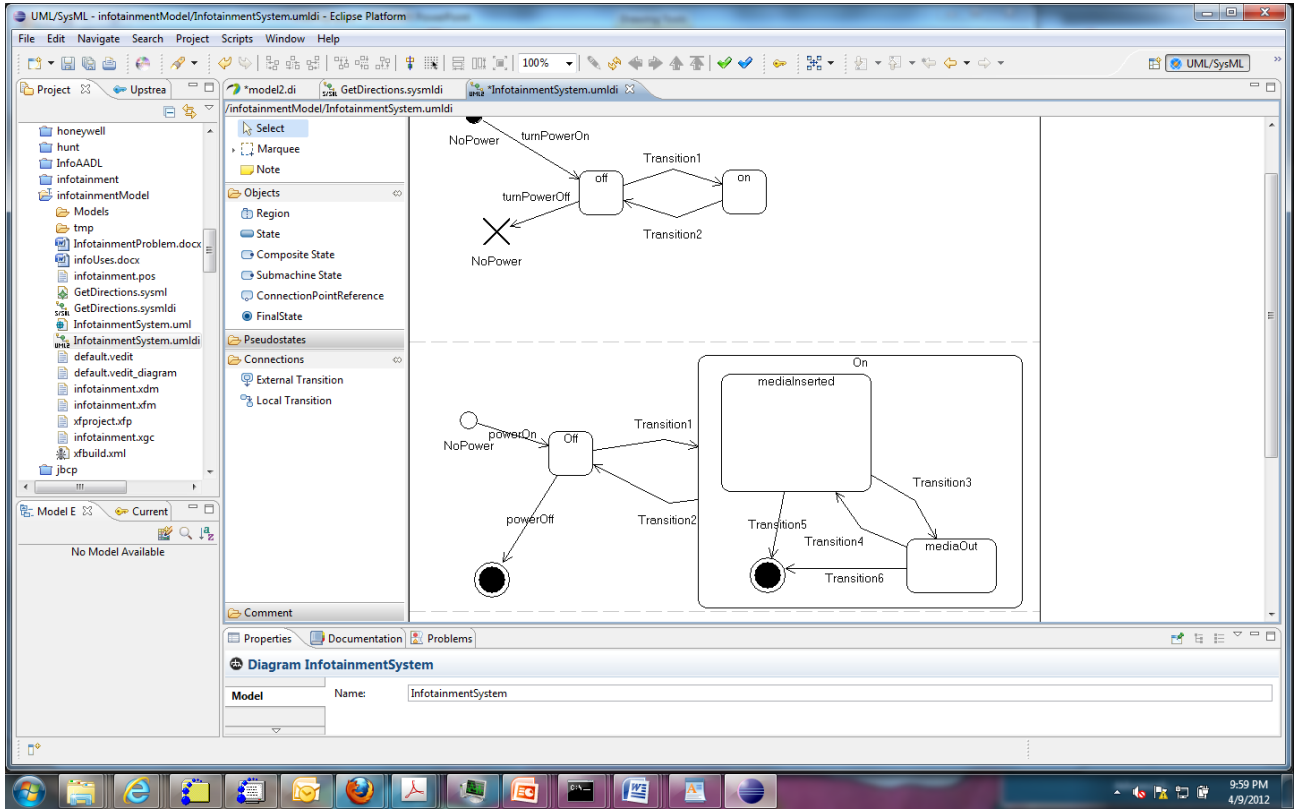
The team began UML training to provide clear, concise abstractions.



© 2012 Carnegie Mellon University

Dual Purpose

Using Eclipse to develop the architecture description had the additional benefit of getting the architects familiar with the Eclipse IDE.



© 2012 Carnegie Mellon University

Results and Observations - Challenges

Architects gave too much detail – implementation versus architecture.

Constantly got wrapped up in functionality and forgot QAs until reminded.

They wanted strategic reuse but they were operating tactically; they failed to see the ecosystem as charting strategic directions.

Reference architecture is a long-term strategy and the team continued with a short-term view.

Explaining the value of strategy to tactical engineers is difficult.

© 2012 Carnegie Mellon University

Result and Observations - Wins

They have existing architectures and deep experience. But they were very focused on their own legacy and details.

Now teams are beginning to focus on architectural concepts; amount of “useless” (i.e. implementation, short-term focused) material generated has decreased.

They have begun to use architecture concepts as vocabulary, e.g. using a “gateway pattern” to generalize different bus protocols.

Starting to realize what should go into the architecture documentation.

Moving toward the goal of a single reference architecture.

© 2012 Carnegie Mellon University

Conclusions

Creating a reference architecture is hard.

This required experienced architects to think differently! None of them had ever described or even used a reference architecture before.

To meet this challenge we had to tailor and combine a number of existing architecture creation, analysis, and description methods.

Early results from this mashup are promising.

© 2012 Carnegie Mellon University