ArchE – An Architecture Design Assistant

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Len Bass
August 2, 2007
Outline

What is ArchE?
What problem are we going to demonstrate?
What is input to ArchE?
What is a reasoning framework?
What is ArchE?

ArchE is a software architecture design assistant, which:

- Takes quality and functional requirements as input
- Elicits key quality attribute information to refine quality requirements
- Elicits key architectural information
- Derives candidate architectures
- Evaluates whether quality requirements are satisfied
- Identifies tradeoffs
- Suggests alternative architectures

ArchE is implemented in Eclipse using Java and the JESS expert system.
What does ArchE “know”?

ArchE “knows”:

• Architecture design process – how to get an architecture from requirements

• Quality knowledge – how to achieve required qualities in an architecture design

• What questions to ask – how to get the architect to think precisely about architectural design.

Key principle: Quality attribute requirements are primary drivers for architecture design and models capture the relations between architecture and desired results.
Sample Problem - Clemson Transit Assistance System (CTAS)

Wireless hand held itinerary planning system

User can plan routes and modes of transportation

Traveler can periodically update information on CTAS and reconsider itinerary.

External information services (hotel, transit systems, parking lot information) assumed.
Use Cases
Initial Input to ArchE

Functions with dependency relations

Quality requirements expressed as quality attribute scenarios
# Initial Functions for CTAS

<table>
<thead>
<tr>
<th>Id</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Show Itinerary</td>
</tr>
<tr>
<td>10</td>
<td>Manage user profile</td>
</tr>
<tr>
<td>10.1</td>
<td>Create user profile</td>
</tr>
<tr>
<td>10.2</td>
<td>Modify user profile</td>
</tr>
<tr>
<td>2</td>
<td>Attach to model</td>
</tr>
<tr>
<td>3</td>
<td>Register views</td>
</tr>
<tr>
<td>4</td>
<td>Handle user interaction</td>
</tr>
<tr>
<td>5</td>
<td>Manage external device</td>
</tr>
<tr>
<td>6</td>
<td>Save data</td>
</tr>
<tr>
<td>7</td>
<td>Query for data</td>
</tr>
<tr>
<td>8</td>
<td>Locate service</td>
</tr>
<tr>
<td>9</td>
<td>Manage itinerary</td>
</tr>
</tbody>
</table>
Responsibilities

“Responsibilities” are fundamental to the design process ArchE supports

Responsibilities are general statements about an architectural element and include: the actions an element performs, the knowledge an element maintains, major decisions an element makes that affect others.

ArchE maps functions into responsibilities

User specifies relationships among responsibilities manually.

Responsibility graph for CTAS

Model
- Create user
- Modify user
- Query for data
- Locate
- Save
- Modify
- Register views
- Attatch to
- Handle user
- Use

View
- Show
- Attach to
- Itinerary

Controller
- Handle user
- Use
- Manage External devices
- Devices, GPS, etc.
- DB

Legend
- Data

© 2007 Carnegie Mellon University
# Relationships among responsibilities

<table>
<thead>
<tr>
<th>Parent responsibility</th>
<th>Relationship</th>
<th>Child responsibility</th>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach to model</td>
<td>dependency</td>
<td>Register views</td>
<td>Probability inco...</td>
<td>0.7</td>
<td>Probability outg...</td>
</tr>
<tr>
<td>Create user profile</td>
<td>dependency</td>
<td>Modify user profile</td>
<td>Probability inco...</td>
<td>0.7</td>
<td>Probability outg...</td>
</tr>
<tr>
<td>Create user profile</td>
<td>dependency</td>
<td>Save data</td>
<td>Probability inco...</td>
<td>0.7</td>
<td>Probability outg...</td>
</tr>
<tr>
<td>Handle user interaction</td>
<td>dependency</td>
<td>Create user profile</td>
<td>Probability inco...</td>
<td>0.7</td>
<td>Probability outg...</td>
</tr>
<tr>
<td>Handle user interaction</td>
<td>dependency</td>
<td>Manage itinerary</td>
<td>Probability inco...</td>
<td>0.7</td>
<td>Probability outg...</td>
</tr>
<tr>
<td>Handle user interaction</td>
<td>dependency</td>
<td>Modify user profile</td>
<td>Probability inco...</td>
<td>0.7</td>
<td>Probability outg...</td>
</tr>
<tr>
<td>Handle user interaction</td>
<td>dependency</td>
<td>Show Itinerary</td>
<td>Probability inco...</td>
<td>0.7</td>
<td>Probability outg...</td>
</tr>
<tr>
<td>Manage external device</td>
<td>dependency</td>
<td>Manage itinerary</td>
<td>Probability inco...</td>
<td>0.7</td>
<td>Probability outg...</td>
</tr>
<tr>
<td>Manage itinerary</td>
<td>dependency</td>
<td>Query for data</td>
<td>Probability inco...</td>
<td>0.7</td>
<td>Probability outg...</td>
</tr>
<tr>
<td>Manage itinerary</td>
<td>dependency</td>
<td>Save data</td>
<td>Probability inco...</td>
<td>0.7</td>
<td>Probability outg...</td>
</tr>
<tr>
<td>Manage itinerary</td>
<td>dependency</td>
<td>Show Itinerary</td>
<td>Probability inco...</td>
<td>0.7</td>
<td>Probability outg...</td>
</tr>
<tr>
<td>Manage user profile</td>
<td>Contains</td>
<td>Create user profile</td>
<td>Probability inco...</td>
<td>0.7</td>
<td>Probability outg...</td>
</tr>
<tr>
<td>Manage user profile</td>
<td>Contains</td>
<td>Modify user profile</td>
<td>Probability inco...</td>
<td>0.7</td>
<td>Probability outg...</td>
</tr>
<tr>
<td>Modify user profile</td>
<td>dependency</td>
<td>Manage itinerary</td>
<td>Probability inco...</td>
<td>0.7</td>
<td>Probability outg...</td>
</tr>
<tr>
<td>Modify user profile</td>
<td>dependency</td>
<td>Save data</td>
<td>Probability inco...</td>
<td>0.7</td>
<td>Probability outg...</td>
</tr>
<tr>
<td>Query for data</td>
<td>dependency</td>
<td>Locate service</td>
<td>Probability inco...</td>
<td>0.7</td>
<td>Probability outg...</td>
</tr>
</tbody>
</table>
Quality Attribute Scenarios

Two modifiability scenarios for now:

1) Add the ability to specify priorities when computing an itinerary. The effort for adding the function should be less than 1 person day.

2) Add a function to notify others of late arrival. The effort for adding the function should be less than .5 person days.
Scenario addition screen

Scenario
A scenario is a quality attribute requirement of a system and is described in six parts.

Scenario Text:
Add the ability to specify priorities when computing an itinerary.

Type: Modifiability
- Insight

Six Parts
- Text | Type | Unit | Value
- Stimulus: change request
- Source of stimulus: super user | End user
- Environment: normal operations
- Artifact: system
- Response: modify the manage itinerary function
- Response measure: effort to modify | Cost Constraint | Days | 1.0

Help | Save | Close | New | Cancel
Scenarios must be related to responsibilities (manually)
ArchE reasoning framework

ArchE uses a modifiability reasoning framework to reason about the scenarios.

What is a reasoning framework?

What is the modifiability reasoning framework?
Reasoning Frameworks

A reasoning framework is a vehicle for encapsulating the quality attribute knowledge and the tools needed to analyze the behavior of a system with respect to some quality attribute.

Can be used:

- To predict behavior before the system is built
- Understand behavior after it is built
- Make design decisions while it is being built

Reason for encapsulating quality attribute knowledge is to enable incorporation of quality attribute knowledge in ArchE without requiring quality attributes to know about each other.
Elements of a Reasoning Framework

1. Problem description
2. Analytic theory
3. Analytic constraints
4. Model representation
5. Interpretation
6. Evaluation procedure
Example: Performance Reasoning Framework

- Constrained interarrival rates for aperiodic streams, no I/O operations, components don’t suspend themselves, fixed execution time

- Observes

- Quality attribute scenarios with latency measures

- Restricted by

- Prediction of average latency

- Based on

- RMA, queueing theory

Key:
- Data representation
- Transformation
- Reasoning framework element
- Packageable implementation
- Dependency
- Data flow
Modifiability Reasoning Framework - 1

Based on coupling and cohesion concepts.

Modules are coupled to each other:

- Tightly (high probability of change propagating)
- Medium (medium probability of change propagating)
- Low (low probability of change propagating)

Responsibilities are assigned to modules.

Cost of change is assigned to each responsibility.

A change to one responsibility in a module is assumed to propagate to other responsibilities in the module.
Modifiability Reasoning Framework - 2

Modifiability scenario is tied to the modification of several responsibilities.

Each responsibility has a cost of change and a probability of propagating to other responsibilities. Each of the propagated to responsibilities, in turn, has a cost of change and a probability of propagating to additional responsibilities.

Sum the costs weighted by the probability of a responsibility being changed.
ArchE calculations for CTAS

ArchE calculates whether cost of change for particular scenario is within bounds.

Out of bounds is indicated by red light.
Scenario – Notify others of late arrival

ArchE suggests several tactics – encapsulate and localize:

<table>
<thead>
<tr>
<th>Question type</th>
<th>Question category</th>
<th>Question text</th>
</tr>
</thead>
<tbody>
<tr>
<td>confirmCost</td>
<td>Applying tactics</td>
<td>Please verify that the given cost are correct or specify the new cost when preparing the</td>
</tr>
<tr>
<td>confirmCost</td>
<td>Applying tactics</td>
<td>Please verify that the given cost are correct or specify the new cost when preparing the</td>
</tr>
<tr>
<td>confirmCost</td>
<td>Applying tactics</td>
<td>Please verify that the given cost are correct or specify the new cost when preparing the</td>
</tr>
<tr>
<td>confirmCost</td>
<td>Applying tactics</td>
<td>Please verify that the given cost are correct or specify the new cost when preparing the</td>
</tr>
<tr>
<td>confirmCost</td>
<td>Applying tactics</td>
<td>Please verify that the given cost are correct or specify the new cost when preparing the</td>
</tr>
<tr>
<td>encapsulate</td>
<td>Applying modifiability tactics</td>
<td>Please specify what level of encapsulation you want (0 = none, 10 = completely)?</td>
</tr>
<tr>
<td>encapsulate</td>
<td>Applying modifiability tactics</td>
<td>Please specify what level of encapsulation you want (0 = none, 10 = completely)?</td>
</tr>
<tr>
<td>localize</td>
<td>Applying modifiability tactics</td>
<td>Do you want me to apply the localization tactic for scenario &quot;Add a function to notify other</td>
</tr>
<tr>
<td>wrapper</td>
<td>Applying modifiability tactics</td>
<td>Do you want to create a wrapper around the responsibility Modify user profile? If so, ple</td>
</tr>
</tbody>
</table>
Localize changes – before

Diagram showing processes and data flow related to user and service management, including modes like Create user, Modify user, and Query for user, along with related actions such as registering, saving, and attaching to data. The diagram also includes external devices, GPS, and services, with legend indicating data flow.
Localize changes - after

View
- Show
- Attach to

Model
- Create user
- Modify user
- Save
- Dispatch
- Query for data
- Locate
- Itinerary

Controller
- Handle user
- Manage External devices

Legend
- Data

Services
- Devices, GPS, etc.
- DB
ArchE creates new responsibility

ArchE does not know semantics of application – architect must label new responsibility. In CTAS it is called “dispatch”

Cost of change must be entered for “dispatch”

New probabilities of propagation must be entered for “dispatch”
## New responsibility in ArchE

<table>
<thead>
<tr>
<th>Name</th>
<th>Cost of change ($)</th>
<th>Exec.time (ms)</th>
<th>Level of encapsulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach to model</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create user profile</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Handle user interaction</td>
<td>2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Locate service</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manage external device</td>
<td>2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manage Itinerary</td>
<td>5.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manage user profiles</td>
<td>2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modify user profile</td>
<td>1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>New responsibility because of localization of scenario gen...</strong></td>
<td><strong>0.0</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Query for data</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Register views</td>
<td>0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Save data</td>
<td>1.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Continuing with ArchE

Architect continues choosing one tactic at a time.

ArchE has reasoning frameworks for modifiability and real time performance.

Architect interacts, choosing tactics until all of the scenarios have been satisfied.

The resulting design is then exported.
Use of ArchE

ArchE has been used to support a graduate class in software architecture at Clemson University

Student feedback:

• The overall concept is very convincing… with a little refining the software should be great.
• The good thing about ArchE during the architecture design process is that it automatically computes the effort of changing one quality attribute on the whole architecture
• The scenario based approach makes it easier to think about how architectural decisions will impact the required quality attributes of a system.
ArchE now and in the future

ArchE and the ArchE Users’ Guide can be downloaded from http://www.sei.cmu.edu/architecture/arche.html

The available version of ArchE has reasoning frameworks for modifiability and real time performance

Toward the end of this year, we will distribute a version of ArchE that is extensible in reasoning frameworks.

- A researcher in quality attributes generates a reasoning framework embodying their theory
- ArchE will manage trade offs with other quality attributes
- ArchE will enable a comparison of a particular theoretical approach to other approaches for the same quality attribute.
DEMO