Driving and Managing Architectural Decisions with Aspects

Alessandro Garcia (Lancaster University, UK)
Thaís Batista (UFRN, Brazil)
Awais Rashid (Lancaster University, UK)
Cláudio Sant’Anna (PUC-Rio, Brazil - Lancaster University, UK)
(claudio@les.inf.puc-rio.br)

SHARK – Workshop on SHAring and Reusing architectural Knowledge 2006
Modularity and Architectural Reasoning

• Software architecture must be modular
  – Modular Reasoning
    • Reasoning about individual elements in isolation
  – Compositional Reasoning
    • Reasoning about the architecture as a whole

• Existing architecture design and analysis approaches support some level of both reasoning types
  – E.g.:
    • Views (“4+1” view model)
    • Architectural Styles
Modularity and Architectural Reasoning

• Architectural Decisions
  – Encompass critical architectural choices.
  – It is, therefore, important to document architectural decisions in a systematic fashion.
  – However, documenting architectural decisions alone is not sufficient.
    • Architectural decisions have a broadly-scoped impact on the architecture.
    • Current approaches do not provide modular and compositional reasoning about architectural decisions.
Aspects Driving Architectural Decisions

- Decisions pertaining to broadly-scoped concerns pull the architecture in various directions.
Crosscutting Architectural Decisions
Modularity and Architectural Reasoning

• It is important to
  – Capture the behavioural and structural decisions introduced in elements in the views.
  – Provide a mechanism that can quantify over the various elements to compose such decisions.

• Traditional architectural approaches do not support the separate handling of crosscutting architectural decisions.
Architectural Decisions as Aspects

- We proposed an approach to provide modular and compositional reasoning support based on the use of aspect-oriented software development techniques.
Architectural Decisions as Aspects

• Architectural aspect
  – name of the architectural aspect;
  – structural and behavioral architectural decisions
    • the inclusion of components, interfaces, relationships, processes, and so forth
  – composition rules;
  – reasoning section that captures the rationale behind those decisions.
Architectural Decisions as Aspects

Aspect: Availability

Structural Decisions
- ReplicationManager is in charge of replicating the critical components through the replicate interface in order to increase the availability of their provided services. N-Version programming is the software replication technique chosen due its implementation simplicity. Consistency is achieved through the interface makeConsistent, which synchronizes the replica results with the primary component results; thus the unification of the results also allows for other client components viewing the pairs of primary and backup elements as a single component.
- Each Replica component must provide a syncsec interface to collaborate with the primary component before the result of the component services are delivered to the client.
- The ReplicationManager process is decomposed into two processes, ReplicaController and Consistency Controller, in order to decouple these two tasks.

Composition Rules
- componentSet = InformationRetrieval, LocationManager, TouristInfoManager.
- replicaQuantity = 1
- replicaSet = replicate(componentSet, replicaQuantity)
- makeConsistent(replicaSet, componentSet)

Reasoning
- ReplicationManager is in charge of replicating the critical components through the replicate interface in order to increase the availability of their provided services. N-Version programming is the software replication technique chosen due its implementation simplicity. Consistency is achieved through the interface makeConsistent, which synchronizes the replica results with the primary component results; thus the unification of the results also allows for other client components viewing the pairs of primary and backup elements as a single component.
- Each Replica component must provide a syncsec interface to collaborate with the primary component before the result of the component services are delivered to the client.
- The ReplicationManager process is decomposed into two processes, ReplicaController and Consistency Controller, in order to decouple these two tasks.
Architectural Decisions as Aspects

Aspect: Security

<table>
<thead>
<tr>
<th>Structural Decisions</th>
<th>Behavioral Decisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cryptograph Manager</td>
<td></td>
</tr>
<tr>
<td>crypt</td>
<td></td>
</tr>
<tr>
<td>decrypt</td>
<td></td>
</tr>
</tbody>
</table>

Composition Rules

- crypt(Navigator.get_info, before)
- decrypt(Navigator.get_info, after)
- decrypt(InformationRetrieval.provide_info, before)
- crypt(InformationRetrieval.provide_info, after)

Reasoning

CryptographManager is responsible for encrypting and decrypting information using the crypt and decrypt interfaces. It affects the get_info service of the Navigator component. The parameters of get_info are encrypted. The encrypted solicitation is sent to the InformationRetrieval. This component decrypts the data to identify the solicitation, processes it, and encrypts the desired information before sending them back to the get_info service. Then, the final step is to decrypt the returned information.

Aspect: Performance

<table>
<thead>
<tr>
<th>Structural Decisions</th>
<th>Behavioral Decisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Manager</td>
<td></td>
</tr>
<tr>
<td>checkRespTime</td>
<td></td>
</tr>
</tbody>
</table>

Composition Rules

- monitoredServices = navigate, ext_service, get_info
- checkRespTime(monitoredServices, during)
- constrain(Availability.replicaQuantity <=2)

Reasoning

PerformanceManager is responsible for encapsulating a timer and monitoring through checkRespTime the response time of critical services of Navigator. Performance also imposes an important upper bound in the number of replicas (replicaQuantity) defined in the Availability aspect.
## Composing Architectural Decisions

<table>
<thead>
<tr>
<th>Mapping Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong> <code>&lt;elem_type&gt;</code> <code>&lt;elem_name1</code> [<code>=value</code>]&gt; to <code>&lt;elem_name2&gt;</code></td>
<td>introduces an architectural element of type <code>&lt;elem_type&gt;</code> and name <code>&lt;elem_name1&gt;</code>, optionally set its value, to other architectural element <code>&lt;elem_name2&gt;</code></td>
</tr>
<tr>
<td><strong>Modify</strong> <code>&lt;elem_name1&gt;</code> to <code>&lt;elem_name2</code></td>
<td>changes the semantics of an architectural element by modifying its name from <code>&lt;elem_name1&gt;</code> to <code>&lt;elem_name2&gt;</code> or setting a new value to <code>&lt;elem_name1&gt;</code></td>
</tr>
<tr>
<td><strong>Remove</strong> <code>&lt;elem_type&gt;</code> <code>&lt;elem_name1&gt;</code> from <code>&lt;elem_name2&gt;</code></td>
<td>removes an architectural element of type <code>&lt;elem_type&gt;</code> and name <code>&lt;elem_name1&gt;</code> from other architectural element <code>&lt;elem_name2&gt;</code></td>
</tr>
<tr>
<td><strong>Split</strong> <code>&lt;elem_type&gt;</code> <code>&lt;elem_name&gt;</code> into <code>&lt;elem_name_list&gt;</code></td>
<td>separates an architectural element of type <code>&lt;elem_type&gt;</code> and name <code>&lt;elem_name&gt;</code> into two or more elements defined in <code>&lt;elem_name_list&gt;</code></td>
</tr>
<tr>
<td><strong>Unify</strong> <code>&lt;elem_type&gt;</code> <code>&lt;elem_name_list&gt;</code> into <code>&lt;elem_name&gt;</code></td>
<td>groups two or more architectural elements defined in <code>&lt;elem_name_list&gt;</code> in the architectural element <code>&lt;elem_name&gt;</code></td>
</tr>
<tr>
<td><strong>Connect</strong> <code>&lt;elem_name1&gt;</code> to <code>&lt;elem_name2&gt;</code></td>
<td>defines a relationship between the elements <code>&lt;elem_name1&gt;</code> and <code>&lt;elem_name2&gt;</code></td>
</tr>
<tr>
<td><strong>Disconnect</strong> <code>&lt;elem_name1&gt;</code> from <code>&lt;elem_name2&gt;</code></td>
<td>removes a relationship between the elements <code>&lt;elem_name1&gt;</code> and <code>&lt;elem_name2&gt;</code></td>
</tr>
</tbody>
</table>
Composing Architectural Decisions

Component-and-Connector View

Performance

monitoredServices = navigate, ext_service, get_info.
Forall S in monitoredServices
   Connect PerformanceManager.CheckRespTime to Navigator.S [during]

Crypt(Navigator.get_info, before)
Decrypt(Navigator.get_info, after)
Decrypt(InformationRetrieval.provide_info, before)
Crypt(InformationRetrieval.provide_info, after)

Security

Connect CryptographManager.Crypt to Navigator.get_info [before]
Connect CryptographManager.Decrypt to Navigator.get_info [after]
Connect CryptographManager.Decrypt to InformationRetrieval.provide_info [before]
Connect CryptographManager.Crypt to InformationRetrieval.provide_info [after]

Availability

ComponentSet = InformationRetrieval, LocationManager, TouristInfoManager
Add constrain replicaQuantity=1 to ReplicationManager
Forall C in ComponentSet
   Create Replica(replicaQuantity)
   ReplicaSet = IR_Replica, LM_Replica, TIM_Replica
Forall C in ComponentSet
   Add interface syncpri to C
   Forall R in ReplicaSet
      Add interface syncsec to R
   Connect Information_Retrieval.syncpri to IR_Replica.syncsec
   Connect Location_Manager.syncpri to LM_Replica.syncsec
   Connect TouristInfoManager.syncpri to TIM_Replica.syncsec

Constrain(Availability.replicaQuantity <= 2)
CheckRespTime(monitoredServices, during )

replicateSet = replicate(componentSet, replicaQuantity)
makeConsistent(replicaSet, componentSet)
Discussion

• Some Benefits
  – Promoting modular and compositional reasoning of architectural decisions
  – Ease of traceability
  – Promoting knowledge management and reuse
  – Enhance evolvability

• Liabilities
  – Extra burden to grasp the notion of aspectual template
  – Need for providing tools that automate the weaving process
Conclusions

- Architectural crosscutting concerns are even more challenging than implementation crosscutting concerns

- Aspectual templates enhance architectural assessment
  - Additional relevant information for architecture evaluation
  - Aspect identification
Driving and Managing Architectural Decisions with Aspects

Alessandro Garcia (Lancaster University, UK)
Thaís Batista (UFRN, Brazil)
Awais Rashid (Lancaster University, UK)
Cláudio Sant’Anna (PUC-Rio, Brazil - Lancaster University, UK)
(claudio@les.inf.puc-rio.br)

SHARK – Workshop on SHAring and Reusing architectural Knowledge 2006