Analysis of software variability concerns in the deployment aspect of component-based models

Introduction, concepts, and solutions

09/02/2012 Anthony Lee
Presentation content

- Research Background
  - Domains and terminologies

- The particular problems
  - Component deployment and variability

- Solution concepts
  - Architecture and development process

- Significants

- Summary
  - Targets and perspectives
Backgrounds

- Distributed application developments are difficult and complicated
- Evolving distributed applications are costly and time consuming
- Adaptive, autonomic and context-aware applications are the possible solutions because of their variability and flexibility
- However, building such applications are far more complex, and existing methodologies, approaches and tools are still primitive
- Systematic approaches, generic architectures and effective tools are needed in order to facilitate the developments of such applications for software engineers
Terminologies

Component-based software engineering (CBSE):
- ideas: separate services into individual components (e.g. CORBA, COM+, EJB, SCA)
- aims: reusability, emphasis software architecture

Model-Driven Engineering (MDE):
- ideas: Implement applications by using Models - Graphical tools and well established development process (e.g. CASE, MDA)
- aims: readability, adaptively, efficiency

Software Product Line (SPL):
- ideas: to plan for reuse by defining variation points, to build reusable parts and common parts (e.g. Variability specification with Feature Model)
- aims: variability, reusability

High level of abstraction:
- ideas: analyzing/implementing software at conceptual level, considering different levels of detail, hiding/representing information (e.g. MDE, UML)
- aims: readability, complexity, adaptively
Problems of distributed deployment

- Executing distributed deployment require lots of manual efforts
- Technically, no explicit relation between application features/services/functions and deployment strategies
- Lack of high-level abstraction supports to relate application features and deployment requirements directly.
- Making decision of deployment could be tedious and rarely automated.
Software development in 2 levels

Conceptual level
Application models, configurations …

Realization level
Implementations, components …

Connection between 2 levels

Refinements
Model Transformation, Decision selection …

Non-functional requirements, Variability, Heterogeneity

Architecture, Functionalities, Services, Interoperations, Dependencies

Realization
Architecture, Deployment plan, Configuration generation
### Application variability support in existing component models

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Models</th>
<th>Component models</th>
<th>Virtualization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EJB</td>
<td>OSGi</td>
<td>SCA</td>
</tr>
<tr>
<td>Application framework</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modeling level</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System level</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cardinality</td>
<td>+</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>Dependency</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Versioning</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node description</td>
<td></td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>Network description</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extensibility</td>
<td></td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>Interpreted</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deployment mechanism</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run-time changeable</td>
<td>+</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>Executable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generatable</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automated</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* + Direct support*
Application Variability supports in existing component models

<table>
<thead>
<tr>
<th>Criteria</th>
<th>EJB</th>
<th>CSGi</th>
<th>SCA</th>
<th>Fractal</th>
<th>Virtualization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual framework</td>
<td></td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variation point and variant definition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conceptual process</td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Implementation generation</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td>+</td>
</tr>
</tbody>
</table>

+ Direct support
Solution concepts: scenarios

Use case 1: service offering

1. A smart phone (SP) user arrived home
2. Home WiFi connection is detected and offered
3. Mobile phone connected to the home WiFi LAN
4. The home network discovered an appearance of a registered smart phone
5. A Video On Demand (VoD) list is pushed on the SP according to a current time schedule and a predefined device configuration
6. User is interested in a new movie and select it on the mobile
7. STB and TV are then woken up
8. VoD service starts

Use case 2: service transfer

1. A SP user is enjoying the VoD service on a train
2. It is paused by the user when the train arrived at its destination
3. The user arrived home
4. STB is detected in the home WiFi LAN
5. The application suggests a new streaming location
6. The streaming status is saved and stopped on the mobile
7. Video streaming is transferred on the STB
The architecture

Application

Controllers

Repositories

Components with sensors and managers are connected together in different areas

Cloud

Mobile

Home

Monitoring

Analyzing

Planning

Execution

Deployment plans, Configurations, Constraints

Models, Binaries, Resources

Adaptation engine with different processors

Models, Binaries, Resources

Components
The development process
An example of a deployment plan:

```
                    xmlns:ns1="http://francetelecom.com/hase/honeteForHas">
    <SOAP-ENV:Body>
        <ns1:prepareDeployment>
            <plan>
                <applicationId>3</applicationId>
                <applicationName>Home Automation</applicationName>
                <description>Gardez un oeil sur vos dépenses énergétiques !</description>
                <lanId>25</lanId>
                <steps>
                    <actions>
                        <basicActionName>INSTALL</basicActionName>
                        <devId>95</devId>
                        <parameters>
                            <name>package</name>
                            <value>homeautomation</value>
                        </parameters>
                    </actions>
                    <timeout>60</timeout>
                </steps>
                <version>1.5</version>
            </plan>
        </ns1:prepareDeployment>
    </SOAP-ENV:Body></SOAP-ENV:Envelope>
```
Significants

- Take advantage of existing environment descriptions and configurations model
- Enabling variability description to execution environment
- Relating services to execution environments
- Automated deployment plans according to service and execution environment variability (conceptual level and realization level engagement)
Summary and perspectives

- **Main objectives**
  - A methodology for involving deployment properties and variability in development process (To relate application services/components and platforms/devices)
  - A realization mechanism for the mapping between 2 levels (Generating deployment plans, mapping rules)
  - A prototype engine for demonstrating the deployment/adaptation process (Application architecture and execution)

- **Perspectives**
  - Replacing user manual indications by computable constraints
  - An optimizer for dynamic service allocation and re-allocation

- **Technical tools**
  - SCA, OSGi, D&C model (OMG), OCL
Thank you.

Any questions?