Very Large Business Applications Lab

Service-Oriented Product Lines: Towards a Development Process and Feature Management Model for Web Services

SOAPL 2008

http://www.vlba-lab.de/

17. September 2008

Fakultät für Informatik
Institut für betriebliche und technische Informationssysteme
Arbeitsgruppe Wirtschaftsinformatik

Otto-von-Guericke Universität
Magdeburg

Introduction

- **What is software development?**
  - Usage of a software development process
  - Transform requirements into different artifacts (architectural descriptions, interface descriptions, source code…)
- **How to manage artifacts?**
  - Apply changes to existing artifacts
  - Reduce coupling of source code
- **What about reuse?**
  - Commonality and variability
- **Combination of Software Product Lines and Service-Oriented Architectures provides solutions to many common software problems**

Structure

- Introduction
- Definitorial Background
- Development Process for Software Product Lines
- Service-Oriented Product Lines
- Example
- Conclusions

Definitional Background: Software Product Lines

- **Withey: „Product Lines is a group of products sharing a common, managed set of features“ [1]**
- **Specifically, manage variability among features that represent requirements**
- **Goal: Structure and reuse software development artifacts**

Definitional Background: Service-Oriented Architectures

- **Loosely coupled and autonomous services**
- **Properties according to Josuttis: self-containment, coarse-grained interfaces, reusability and composability [2]**
- **Implementation: Web Services or Enterprise Service Bus**

Definitional Background: Web Services

- **„Software applications that can be discovered, described and accessed based on XML and standard Web protocols“ [3]**
- **Described by a WSDL**
  - Abstract definition describes interface, operations and messages
  - Concrete definition describes bindings to operations
- **Distinguish into service broker, provider and consumer**

Structure

- Introduction
- Definitorial Background
- Development Process for Software Product Lines
- Service-Oriented Product Lines
- Example
- Conclusions

Domain Engineering

- **Develop a software family**

- **Analysis**
  - Capture domain specific knowledge
  - Develop a domain model
  - Represent domain concepts and requirements in a central feature model
  - Identify variants with their distinguishing features

- **Design**
  - From architectural description to software entities
  - Decide used frameworks, libraries and programming languages
  - Form technological foundation for implementation of variants

- **Implementation**
  - Make or buy decision for software entities

Application Engineering

- **Develop individual member (of the software family)**

- **Five steps**
  - Problem Analysis (overall problem specification)
  - Product Specification (concrete set of selected features)
  - Collateral Development (Documentation)
  - Product Implementation (Executables and test cases)
  - Deployment

Structure

- Introduction
- Definitorial Background
- Development Process for Software Product Lines
- Service-Oriented Product Lines
- Example
- Conclusions

Service-Oriented Product Lines

- **Implement SPL with an SOA**

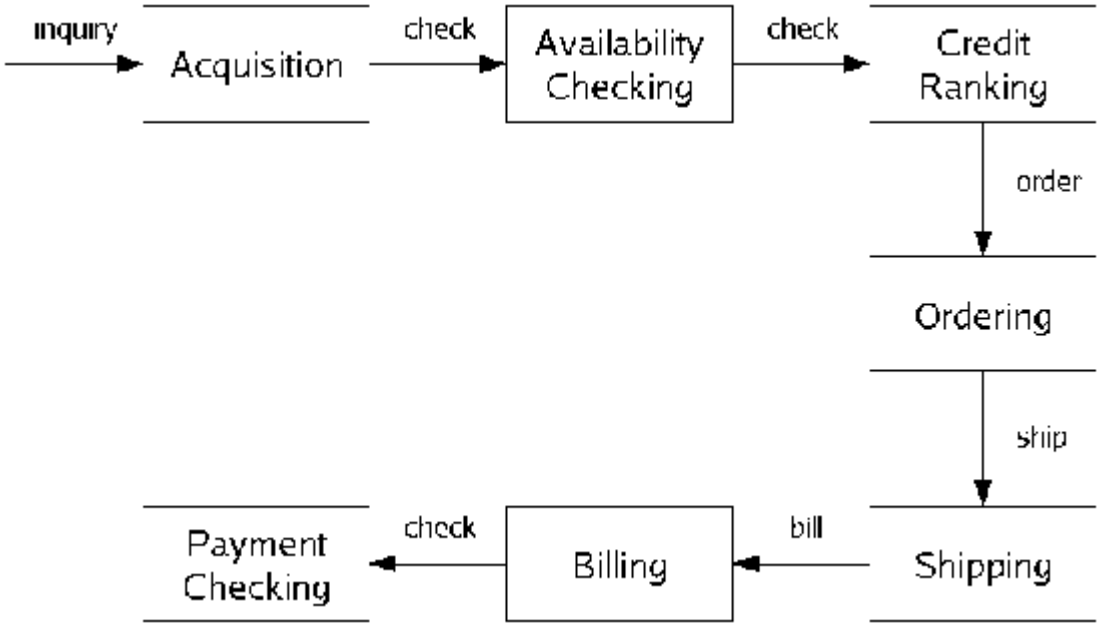
- **Different impacts on development phases**
  - Analysis:
    - Select SOA-specific modeling languages
    - Software requirements can be modeled as features or part of the ESB
  - Design
    - ESB as routing and messaging backbone, and also implements e.g. compliance requirements
    - ESB mostly forms common part of SPL
    - Web Service abstracts whole applications, databases or fine granular software entities
  - Implementation
    - Careful choice of purchased ESB
    - Wrap existing software with Web Services or use web service repositories
    - Full SOPL process (design interface and implementation) vs. light SOPL process (design only interfaces)

Structure

- Introduction
- Definitorial Background
- Development Process for Software Product Lines
- Service-Oriented Product Lines
- Example
- Conclusions

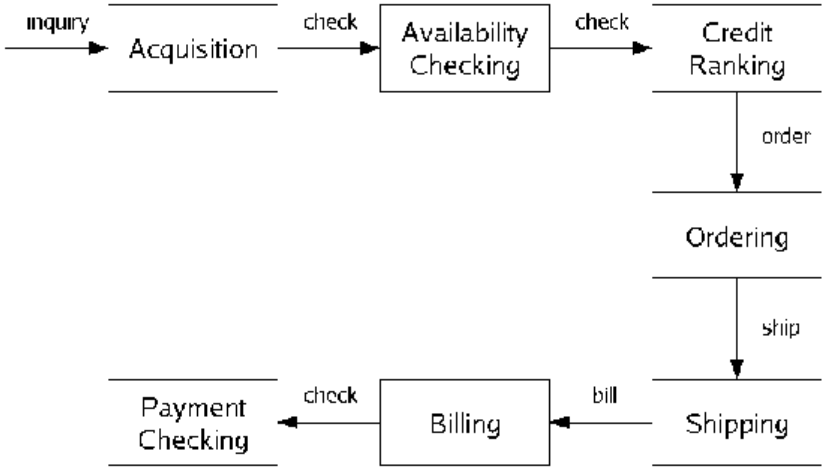
The Web Store

- **Domain Engineering for a Web Store (base taken from [4])**
- **Web store contains 7 modules:**



Web Store: A Feature Model

- **Abstract representation of the overall product line**
- **Uses set-like notations for features**
  - *Base = {Acq,Chk,Crd,Ord,Shp,Bil,Pay}*



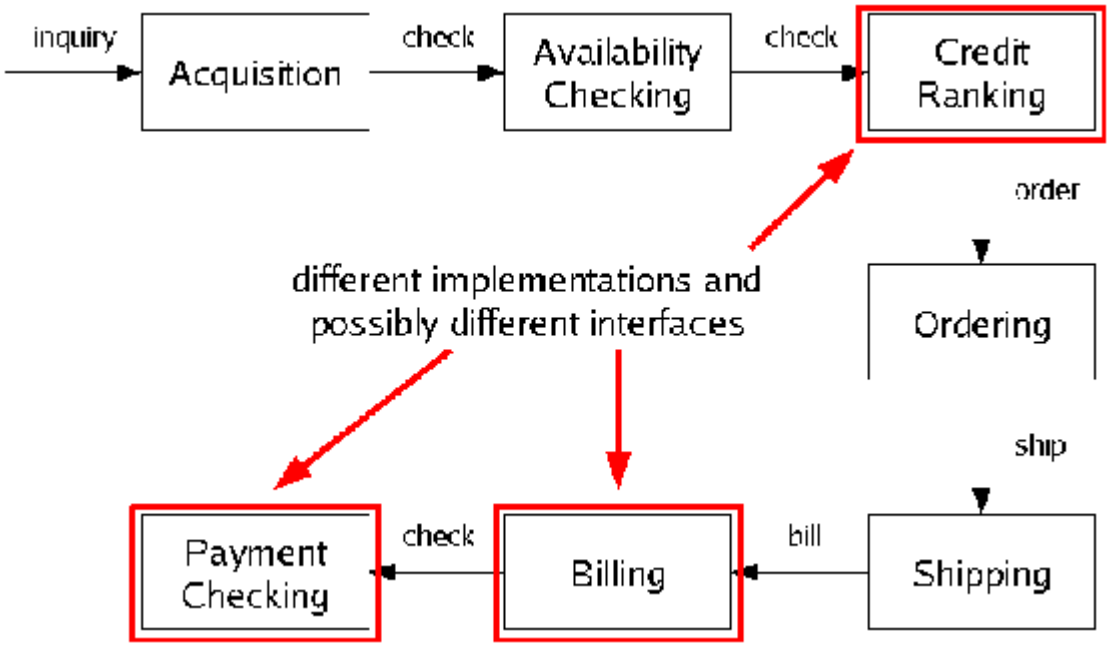
- **Detail out features**
  - Credit Ranking: Use an independent agency (Agc) or explanation of the bank (Bak)
    - *Crd = {Agc, Bak}*
  - Shipment via surface (Sur) or airmail (Air)
    - *Shp = {Sur, Air}*
  - Surface shipment with standard (Std) or Express (Exp) Mail
    - *Sur = {Std, Exp}*

- **Individual member is a composition of specific features**
  - *Store1 = Base*
  - *Acq • Chk • Agc • Bak • Ord • Std • Exp • Air • Bil • Pay*

Web Store: Variability Management

- **Customers demand new features**
  - Discounting for bigger quantities of ordered goods
  - Traceability of features

- **Impacts existing services of the Web Store**



Web Store: Variability Management 2

- **Discounting concepts refines four basic features**
  - Disc = {Crd = {ΔAgc, ΔBak}, ΔBil, ΔPay}

- **Build a new member**
  - Include discounting feature
  - Limit shipment to standard surface mail
  - Store2 = {Base – {Exp, Air}} • Disc
  - Store2 = Acq • Chk • Agc • Bak • Ord • Std • Bil • Pay • Disc
  - Store2 = Acq • Chk • Agc • Bak • Ord • Std • Bil • Pay • ΔAgc • ΔBak • ΔBil • ΔPay

- **Combination of a basic and refined feature leads to the final representation**
  - Store2 = Acq • Chk • Agc´ • Bak´ • Ord • Std • Bil´ • Pay´

Web Store: Variability Management with WSDL

- **Description of Web Services with WSDL gives a high level view**
- **Feature granularity must manage WSDL descriptions**
- **Example: WSDL for Billing**

```
<element name="CalcBillOutput">
 <!– Other definitions ommitted -->
   <xsd:sequence>
     <xsd:element name="customerName“ type="xsd:string"/>
     <xsd:element name="customerAddress“ type="xsd:string"/>
     <xsd:element name="items" type="ItemOrder“ minOcurs="1" maxOccurs="unbound"/>
     <xsd:element name="totalPrice“ type="xsd:integer"/>
   </xsd:sequence>
  <!– Other definitions ommitted -->
</element>
```

- **Variability Management with XAK [5]**

```
<element name="CalcBillOutput“ xak:artifact="STOREbillOutput">
  <!-- Other definitions omitted --!>
    <xsd:sequence xak:module="billOutput">
      <xsd:element name="customerName" type="xsd:string"/>
<!-- Other definitions omitted --!>
```

Web Store: Variability Management with WSDL 2

WSDL Refinement

```
<xak:refines xak:artifact="STOREbillOutput">
  <xak:extends xak:module="billOutput">
    <xak:super xak:module="billOutput"/>
    <xsd:element name="discount" type="xsd:integer"/>
    <xsd:element name="discountedPrice“ type="xsd:integer"/>
</xak:extends>
</xak:refines>
```

Combined WSDL

```
<element name="CalcBillOutput">
 <!– Other definitions ommitted -->
    <xsd:sequence>
      <xsd:element name="customerName“ type="xsd:string"/>
      <xsd:element name="customerAddress“ type="xsd:string"/>
      <xsd:element name="items" type="ItemOrder“ minOcurs="1" maxOccurs="unbound"/>
      <xsd:element name="totalPrice“ type="xsd:integer"/>
      <xsd:element name=“discount“ type="xsd:integer"/>
      <xsd:element name=“discountedPrice“ type="xsd:integer"/>
    </xsd:sequence>
  <!– Other definitions ommitted -->
</element>
```

Conclusions

- Introduction
- Definitorial Background
- Development Process for Software Product Lines
- Service-Oriented Product Lines
- Example
- Conclusions

Conclusions

- **Feature models and variability management models can be used for Service-Oriented Product Lines as well**
- **XML refinements allow practical solution to feature management**
- **Focus on models leads to a high-level view**
- **Promising**
  - If existing code base can be reused efficiently: focus on light SOPL process (only define interfaces)
  - Introduce Domain Specific Languages for domain modeling and SPL configuration, allowing participation of end-users

Fin

http://www.vlba-lab.de/

Thanks for your attention!

References

- [1] J. Withey, “Investment analysis of software assets for product lines,” Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-96-TR-10, 1996.
- [2] N. M. Josuttis, SOA in Practice: The Art of Distributed System Design. Sebastopol, California, USA: O’Reilly Media, Inc., 2007.
- [3] M. C. Daconta, L. J. Obrst, and K. T. Smith, The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management. Indianapolis, Indiana, USA: Wiley Publishing, Inc., 2003.
- [4] S. Apel, C. Kästner, and C. Lengauer, “Research challenges in the tension between features and services,” Proceedings ICSE Workshop on Systems Development in SOA Environments (SDSOA). New York, NY, USA: ACM, 2008, pp. 53–58.
- [5] S. Trujillo, D. Batory, and O. Diaz, “Feature refactoring a multirepresentation program into a product line,” Proceedings of the 5th international conference on Generative programming and component engineering (GPCE). Portland, Oregon, USA: ACM, 2006, pp. 191– 200.