Software Product Line Engineering and Variability Management
Achievements and Challenges

Andreas Metzger & Klaus Pohl
paluno
The Ruhr Institute for Software Technology
University of Duisburg-Essen
Germany

Paper accessible (open access) from:
http://dx.doi.org/10.1145/2593882.2593888
(1) Two complementary development processes

- **Domain Engineering**: Development for reuse
- **Application Engineering**: Derivation of product line applications from domain artifacts

→ Systematic, proactive reuse

(2) Clear differentiation between commonality & variability

- **Commonality** = Features/requirements/artifacts are part of ALL product line applications
  - *E.g.*, *all mobile phones support the same three communication protocols*
- **PL variability** = Variation among product line applications
  - *E.g.*, *some mobile phones include a high-resolution camera*

→ Product line variability (i.e. what varies) in a product line requires an explicit decision, e.g., by product management
Software Product Line Framework

Definition/Realization of Variability and Commonality
- Domain Req. Eng.
- Domain Design
- Domain Realization
- Domain QA

Domain Artifacts
- Domain VM

Binding of Variability
- Application Req. Eng.
- Application Design
- Application Realization
- Application QA

Application 1 – Artifacts
- Application VM

Framework adopted as part of ISO/IEC standard #26550
Product Line Engineering Successes

Reported Benefits: http://www.sei.cmu.edu/productlines

- Decreased cost up to 60%
- Improved productivity up to 10x
- Decreased labour needs up to 87%
- Decreased time to market up to 98%
- Ability to move into new markets months not years
- Increased quality up to 10x

Product Line Hall of Fame: http://splc.net/fame.html

- 20 success stories from companies including Boeing, Bosch, HP, Nokia, Philips, Siemens, Toshiba
Software Variability refers to the ability of a software system or artefact to be efficiently extended, changed, customized or configured for use in a particular context. Svahnberg et al. 2005 (SP&E)

Examples

- Super-class ⇔ different specializations
- Interfaces ⇔ different implementations
- Conditional compilation (#ifdefs)
- ...

Details see Svahnberg et al. 2005 (Software Practice and Experience) and recent survey by Galster et a. 2013 (IEEE TSE)
Software vs. Product Line Variability

**Product Line Variability** describes the variation (differences) between the applications of a product line in terms of properties and qualities (like features that are provided or requirements that are fulfilled).

Realized through

Software variability
Software vs. Product Line Variability

Concrete examples for software variability

- **Commonality**: Is WiFi and MobileBroadband **part of ALL** applications?
- **OR**: **Variability**: WiFi and MobileBroadband are **NOT part of all** applications

  If so, which selection of WiFi / MobileBroadband allowed?
  (e.g., WiFi xor MobileBroadband)

---

You can’t identify product line variability from software artefacts cf. ICSE 2014 FOSE paper, Metzger et al. 2007 (RE)

→ **Product line variability must be explicitly defined**
Two Types of Variability Models

**Domain Variability Model**
- Documents what can vary and how in a product line

**Application Variability Model**
- Specific variability model for each product line application
- Documents variability bindings for application
Variation points

• What differs? (without how)

Variants

• How can it differ?

Constraints

Variation Point

Color

1 out of 3

Variant
red
white
black
Research Achievements - Variability Modelling

Variability Modelling Languages

**Feature Models**

- Focus on *user-observable features*
- Variation points *not explicitly* defined
- Defines *commonality & variability*
- Interrelation with development artefacts required

**Orthogonal Variability Models**

- Focus on *product line variability*
- Variation points *explicitly* defined
- *Only* defines *variability*
- Interrelation with development artefacts required
Achievements
Software Product Line Research in Past 7 Years

Over 600 papers in established international journals & conferences!
Software Product Line Research in Past 7 Years

+ product line evolution (~40), dynamic product lines (~40), multi-product lines (~20), ...
Verification in the presence of variability: The problem

- E.g., take two variable requirements
  - \( r \) related to variant \( v_1 \)
  - \( \neg r \) related to variant \( v_2 \)
- Check \( R = \{ r, \neg r \} \) with single systems techniques → inconsistent
- Assume \( v_1 \) and \( v_2 \) cannot be selected together → consistent

Strategies to consider variability, comparison in Thüm et al. 2014 (ACM CSUR)

<table>
<thead>
<tr>
<th>Application-based</th>
<th>Product-line-based</th>
<th>Feature-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Derive concrete applications(s)</td>
<td>Checks domain artefacts (only)</td>
<td>Checks domain artefacts (only)</td>
</tr>
<tr>
<td>Check each application like in single systems development</td>
<td>Variability model used to consider allowed feature combinations</td>
<td>Considers artefacts realizing a single feature</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Feature combinations not considered</td>
</tr>
</tbody>
</table>
Product-line specific properties to be checked in variability models

- **Membership**: Given a selection of product line features, is this selection valid with regard to the variability definition?
- **Satisfiability**: Is there at least one valid selection of product line features? (non-empty set of derivable application)
- **“Dead” features**: Which features cannot be selected in any application?

Comparison of techniques for variability model analysis see Benavides et al. 2013 (Information Systems)
Generative approaches for product derivation

- Step 1: Feature selection
- Step 2: Generation of SPL application from domain artefacts

Decision support for product derivation based on decision models

e.g., see Rabiser et al. 2012 (ASE)
- Possible decisions (to resolve variability)
- Impacts
- Ordering / pre-conditions

Automated feature selection

- E.g., using search-based techniques
  see Sayyad et al. 2013 (ICSE)

But: In practice product line applications often cannot be fully realized by reusing domain artifacts

→ Application-specific adaptations are required
→ Very little research contribution so far
Challenges
Open Research Challenges (Examples)

- Scope optimization
- Artifact-interrelations in multi-level SPLs

- Interrelation between scoping and RE
- Impact of requirements changes
- +++

- Delayed design decision and variability
- Variability in quality attributes
- +++

- Mapping of product line variability and software variability

- Understanding tradeoffs between expressiveness and analyzability
- Handling large-scale variability models
- +++

- Inter-model verification
- Validation with large-scale, realistic variability models
- +++

- Handling application-specific deviations
- Support for eliciting application-specific requirements

- Documenting application design alternatives
- Impact analysis for application-specific adaptations

- Extended configuration mechanisms
- SPL development environments
- +++

- Minimizing test redundancy
- Correct variability bindings
- +++

See ICSE 2014 FOSE paper for details 😊
Focus on Real Engineering Problems
Don’t Mind the Gap ;-)
Emerging Challenges
Apply Product Line Principles to non-PL Settings, such as

**Framework-based development**
- How to manage invariants across applications developed using a common framework?
- How to handle application specific extensions in conformance with the framework?
- How to determine the effect of changes to quality assurance activities like testing?
- +++

**Clone-and-own** (aka. ”copy-and-modify”)
- How to support invariants in such setting?
- How to manage variations in such settings?
- How to estimate the effect of modifications, e.g. on quality and existing test results?
- +++
Emerging Challenges
Future Internet Applications (IoT, IoS, IoD), e.g., Smart City
Emerging Challenges
Future Internet Applications (IoT, IoS, IoD), e.g., Smart City

**Future software-intensive applications**
- Are IoT-, IoS- and IoD-based
- Are highly complex and distributed
- Have intrinsic, adaptive behavior
- Interact with systems unknown during design
- Face inconsistent & conflicting requirements
- Cannot be designed top-down
- Open-world assumption (instead of closed-world)
- +++

*Can product line principles (e.g. variability management) be applied in such settings?*
Summary

• SPLE has matured as discipline over the last 7 years
  → Significant research contributions
    (over 600 papers in well established journals and conferences)
  → Intl. Software Product Line Conference (SPLC) no. 15 @ google scholar ranking of software engineering (systems) conferences & journals

• Mainly “generic” research contributions
  → Rarely application domain specific

• Validation with “toy examples” or artificial, larger examples
  → Lack of validation based on real industrial cases

• Future challenges are
  → “Incremental” as usual 😊
  → But also radical due to significant changes in technology and economy

Details: ICSE 2014 FOSE paper
Acknowledgments

Thanks for valuable feedback on earlier versions of the paper:

- David Garlan, Carnegie Mellon University
- Linda Northrop, SEI
- Klaus Schmid, University of Hildesheim
- Frank van der Linden, PHILIPS
- Our PhD-students: André Heuer, Richard Pohl, Vanessa Stricker

Supported by DFG (German Research Foundation) under grant PO 607/4-1 (KOPI)