Outline

- Variability & Variability management defined
- Variability modeling, variability realization
- Tools
- Variability analysis
- Current (own) research
- Want to know more? Pointers & references
Outline

- Variability & Variability management defined
- Variability modeling, variability realization
- Tools
- Variability analysis
- Current (own) research
- Want to know more? Pointers & references
Commonality and Variability

(AL-Msie’Deen, 2014)
Variability

- 'Software variability is 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, p. 706)
- Product line variability: how the applications of a product line can differ
  - product line variability & commonalities define the scope of a product line
  - Part of product line variability is expressed via software variability
  - Variability → flexibility for product differentiation and diversification

Variability Management (VM) (1)

- You need management, when there is enough variability
- "Variability Management (VM) encompasses the activities of
  - explicitly representing variability in software artefacts throughout the lifecycle,
  - managing dependencies among different variabilities, and
  - supporting the instantiations of those variabilities" (Schmid and John, 2004).
- At core of SPLs is the identification and management of commonalities and variations in the systems’ artefacts
- Research and practice: variability management is a central concern in SPLs
Variability management (2)

- Involves complex and challenging tasks, needs to be supported by
  - appropriate approaches,
  - techniques, and
  - tools (Bosch et al., 2001; Schmid and John, 2004)
- Ability to represent variability
  - With large number of variants, representation of them becomes important
    - Adequate concepts for practitioners
    - Proper simplicity, clarity and rigour of concepts
- Management processes
- Tools
- “Systematically identifying and appropriately managing variabilities among different systems of a family are the key characteristics that distinguish SPLE from other reuse-based software development approaches” (Chen et al., 2009).

Outline

- Variability & Variability management defined
- Variability modeling
- Tools
- Variability analysis
- Current (own) research
- Want to know more? Pointers & references
Feature (piirre)

- “a distinguishable characteristic of a concept (system, component, etc.) that is relevant to some stakeholder of the concept” (Czarnecki et al., 2000)
- “a logical unit of behavior specified by a set of functional and non-functional requirements” (Bosch, 2000)
- Many other similar definitions exist (10+) (Berger et al., 2015)
- The usage of term feature and good/bad features, etc. have been characterized in “What is a feature?: a qualitative study of features in industrial software product lines” (Berger et al., 2015)
- SPLC 2015 best paper award

Feature variability

Product feature list

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>f2</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>f3</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>f4</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>f5</td>
<td></td>
<td>+</td>
</tr>
</tbody>
</table>

Product feature - matrix

Feature model

Feature configuration model
The most popular means of variability representation (Chen et al., 2009)

(Myllärniemi et al., 2014)
Adapted from (Benavides et al., 2010)

'Extended feature models': add attributes

Usually: an attribute has at least name, domain, value
Constraints on attributes / their values

(Benavides et al., 2010)
Cardinality based feature model: cardinality instead of mandatory, optional, alternative, or

A variation point: documents a variable item defining “what can vary”
- without saying how it can vary (colour of a car)
- “A variation point is a representation of a variability subject within domain artefacts enriched by contextual information.” (Pohl et al., 2005)
- “place in a design or implementation that identifies a location at which variation occurs”
- facilitate the systematic documentation and traceability of variability, development for reuse and with reuse, assessment, and evolution

A variant: documents a concrete variation of a variation point defining “how something can vary”.
- blue in colour of a car
- “A variant is a representation of a variability object within domain artefacts.” (Pohl et al., 2005)
- “A variant identifies a single option of a variation point and can be associated with other artefacts to indicate that those artefacts correspond to a particular option.” (Pohl et al., 2005)

Variability constraints restrict the variability
- permissible combinations of variants, e.g. selection of one variant requires or excludes the selection of another variant

Orthogonal Variability Modeling (OVM) terminology: Variation point, variability

(Czarnecki et al., 2005)

(Metzger and Pohl, 2014; Pohl et al., 2005)
Other ways to model software variability

- Orthogonal variability modeling: variation points, variants, constraints, no modeling of commonality
  - OVM (Pohl et al., 2005), Covamof (Sinnema et al., 2004), CVL (OMG, 2015)
- Decision modeling: Questions with constraints and a workflow (“wizard”)
  - an overview in (Schmid et al., 2011)
- Clafer (Bąk et al., 2014): Odd but possibly effective mixing of classes and features
- Koala component model: like IC-circuit diagram (van Ommering et al., 2000)
- Kumbang ontology (Asikainen et al., 2007) & Koalish (Asikainen et al., 2004) – extension of Koala: Feature model + component model + types
- Also possible: borrow methods from knowledge-based configuration

Variability realization

- Variability is realized using the capabilities of programming languages, compilers, and linkers
- Approaches
  - use of inheritance
    - e.g., implement alternative sub-classes for an abstract super-class
  - conditional compilation
    - e.g., using preprocessor directives such as #ifdef
  - binary replacement
    - e.g., providing the linker with alternative implementations of libraries
  - aspect-oriented programming
    - e.g., the ‘weaving’ of alternative code
- Conditional compilation has received significant attention, e.g.
  - type-safe feature modularity
  - treatment of feature dependencies.
New approaches

- New types of programming languages consider features and variability as first-class concepts
- explicitly handle feature modularity and feature dependencies/interactions at the language
- Feature-oriented programming (FOP)
  - supports the flexible and modular composition of systems from individual features
  - “a feature module encapsulates changes that are made to a program in order to add a new capability or functionality”
- Delta-oriented programming
  - a compositional programming language
  - a product line is realized by a core module and a set of delta modules
  - The core module implements a valid application developed with single system development techniques
  - Delta modules specify changes to be applied to the core module to implement additional applications
  - Changes to the core model include the adding of additional code (as in FOP), but also removing and even the modification of code
  - Maintainability?
-Variability often cross-cuts the decomposition structure “Cross-cutting variability”
- Introduce additional composition operations on top of sequential composition
- treat features as aspects

Outline

- Variability & Variability management defined
- Variability modeling, variability realization
- Tools
- Variability analysis
- Current (own) research
- Want to know more? Pointers & references
Kumbang tools ≈ Features + Koala Structure

Feature modeling

Structural modeling (Koala)

Domain Engineering: Kumbang modeler

Kumbang ontology

Application Engineering: Kumbang configurator

Kumbang ontology has a simple mapping to Weight Constrain rules (a form of logic programming) providing clear formal semantics.

- Answer set programming solvers (Smodels, or clasp) provide inference

Clafer

- Clafer unifies class, association, and property (attribute, reference, role) into a single construct called *clafer* (CLAss FEature Relationship)

- A clafer declaration includes multiplicities and may optionally contain a superclafer or a reference to a clafer or both.

```plaintext
1 abstract options
2    xor size
3      small ?
4      large ?
5      cache ?
6      size -> integer
7      fixed ?
8      [ small & cache -> fixed ]
```

Fig. 7 Feature model of component options in Clafer

(Bąk et al., 2014)
### Clafer meaning

An abstract clafer called **Person**.

A concrete child reference clafer of **Person** with type **string**.

**UnitNo** is a child of **Street**, which in turn is a child of **Address**.

A concrete clafer called **JohnDoe**, which is followed by a list of constraints restricting the set of persons to persons with the given **Name**, **Street**, etc.

Some constraints do not set values but simply assert claferes, such as, **Male** and **Married**, that is, **JohnDoe** can be characterized as a **Married Male**.

(Antkiewicz, 2015)

### Clafer tools

![Figure 1: Architecture and Capabilities of Clafer and Tools](image)

(Antkiewicz et al., 2013)
BigLever Software: Gears

Gears exercises the variation points according to the feature profile of the product you want to build.

(Krueger and Clements, 2015)

Gears & configurable SPLs (1)

Modern PLE is like a factory that produces the products' engineering artifacts.

(Krueger and Clements, 2015)
Gears & configurable SPLs (2)

Shared assets are like the factory's supply chain.

(Pure:Systems, 2015)
Outline

- Variability & Variability management defined
- Variability modeling, variability realization
- Tools
- Variability analysis
- Current (own) research
- Want to know more? Pointers & references
(Automated) Analysis of Feature models: big picture

Fig. 4. Process for the automated analysis of feature models.

Benavides et al. (2010) see product derivation / configuration as one form of analysis (requirements are additional inputs)

Examples

Valid product:
- Input: feature model + a product (i.e. set of features)
- Output: a value that indicates if the products is valid according to the feature model

Dead features i.e. features that cannot appear in any of the products of the software product line

Fig. 6. Common cases of dead features. Grey features are dead.
Analyses of feature models

(Benavides et al., 2010)

Table 3

Summary of operations and support: Benavides et al. [51], Czarnecki et al. [52], Chey et al. [53], Moorsin et al. [54], Mendonca et al. [55], Mendonca et al. [56], Sun et al. [57], Theis et al. [58], van der Storz [59], Zhang et al. [60], Zhang et al. [61], Zhang et al. [62], Zhang et al. [63], Zhang et al. [64], Zhang et al. [65].

Variability & Variability management defined
- Variability modeling, variability realization
- Tools
- Variability analysis
- Current (own) research
- Want to know more? Pointers & references

Outline
Evidence-based software engineering (EBSE)

• “EBSE aims to improve decision making related to software development and maintenance by integrating current best evidence from research with practical experience and human values.”
• Idea borrowed from evidence-based medicine

1. Relevant problem or information need \(\rightarrow\) an answerable question
2. Search the literature for the best available evidence
3. Critically appraise the evidence for its validity, impact, and applicability
4. Integrate the appraised evidence & practical experience and the current (customer’s) context to make decisions
5. Evaluate performance and seek ways to improve it

(Dybå et al., 2005)

Systematic reviews (SRs)

• Summarize studies and synthesize evidence about a specific topic following a predefined, systematic and reliable research method
• Systematic literature reviews (SLR)
• Systematic mapping studies (Map)
• Tertiary studies summarize SRs
• SPLs and variability increasingly summarized in SRs

Evidence based SPL and Variability management?

- A tertiary study: 59 systematic reviews on SPLs and variability
  - SRs included ~2500 primary studies (duplicates included)
  - Domain engineering addressed more often than application engineering
  - Scarcely explicit separation
  - Few practitioner guidelines
    - Except listings and taxonomies of existing research
  - Focus mostly on researchers’ interests
    - Identify gaps in the research
    - Justify future research
  - Lack of empirical primary studies
    ⇒ missing basis for building an evidence-based foundation for SPLs & Variability management

Systematic reviews

<table>
<thead>
<tr>
<th>Topic category</th>
<th>#</th>
<th>Studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements engineering</td>
<td>7</td>
<td>S1 S3 S25 S26 S32 S40 S50</td>
</tr>
<tr>
<td>Design</td>
<td>8</td>
<td>S6 S19 S27 S47 S56 S59</td>
</tr>
<tr>
<td>Testing</td>
<td>10</td>
<td>S7 S8 S15 S24 S29 S30 S31 S35 S43 S46</td>
</tr>
<tr>
<td>Variability management</td>
<td>13</td>
<td>S4 S10 S11 S12 S13 S18 S20 S22 S23 S34 S46 S49 S50</td>
</tr>
<tr>
<td>Quality attributes</td>
<td>7</td>
<td>S36 S38 S39 S41 S45 S55</td>
</tr>
<tr>
<td>Process model</td>
<td>2</td>
<td>S14 S53</td>
</tr>
<tr>
<td>Maintenance</td>
<td>4</td>
<td>S3 S27 S28 S51</td>
</tr>
<tr>
<td>Management</td>
<td>6</td>
<td>S2 S16 S25 S33 S48 S57</td>
</tr>
<tr>
<td>Specific SPLs</td>
<td>11</td>
<td>S5 S9 S17 S21 S36 S37 S42 S44 S52 S54 S58</td>
</tr>
<tr>
<td>Empirics</td>
<td>6</td>
<td>S11 S13 S22 S24 S25 S26</td>
</tr>
</tbody>
</table>

Realisation: none
~2500 articles included in SR:s (inclusive duplicates)
Note: any life-cycle or process model (e.g., V-model, spiral model, agile models) can be used

(Metzger and Pohl, 2014; [1]=Pohl et al., 2005)

Potential to learn from knowledge-based configuration?

- Product configuration has long history
- Variability management in product configuration shares with software product lines, including similarities in conceptual foundation
  - Potential for knowledge sharing
- SPL modeling has been researched a lot.
- There is potential to transfer principles from product configuration to SPL community
Some principles and potential effects to aim for

- Separation between types and instances
  - Conceptual separation of domain and application models
  - Types modularize models
  - Reuse via instantiation of types
- Conceptual clarity
  - Distinct relationships such as has-part and is-a
  - Cardinality as a basis for compositional relationships
  - Balance between representational gap and simplicity
- Concepts before representation
  - Domain phenomena as concepts with semantics
  - Multiple representations of concepts such as textual and graphical
  - Equivalence and synchronization of different representations.
  - Representations need a conceptual basis.
- Support different viewpoints with corresponding concepts.

Three instantiation levels of FM according to the KBC approach

Adapted from (Soininen et al., 1998)
Ambiguous FM concepts: alternative was originally meant to imply specialization: is-a instead of consists-of! (Kang et al., 1990)

Summary

- SPLs can be effective -- significant potential gains
- Business based on SPLs is not easy but it is doable
  - E.g. important management (human) aspects not discussed today
- There are numerous methods and some (quite mature) tools
  - Many research proposals have not been validated
- Active research
  - But limited evidence-based advice for practitioners
- Many challenges and research opportunities exist
  - Linda Northrop: Major challenges include Accelerating SPL development, Software assurance, Scaling
Outline

- Variability & Variability management defined
- Variability modeling, variability realization
- Tools
- Variability analysis
- Current (own) research
- Want to know more? Pointers & references

Systematic reviews

S18 M. Galster, D. Weys, D. Tofan, B. Michalik, and P. Avergou, “Variability in software systems — a systematic literature review,” IEEE Transactions on
Systematic reviews


S50 I. S. Santos, R. M. C. C. Andrade, and P. A. Santos Neto, “How to describe spf variabilities in textual use cases: A systematic mapping study,” in *Brazilian Symposium on Software Components, Architectures and Reuse (SRBars)*, 2014, pp. 64–73.


Systematic reviews


Other major sources of information

- A Framework for Software Product Line Practice, Version 5.0


- A bibliography of 600 articles categorized according to 'the big picture’ of Pohl & Linden

- Software Product Line Conference (SPLC), 2000-

- SPLC Hall of Fame: [http://splc.net/fame.html](http://splc.net/fame.html)

- Accepted through the SPLC conference series

- Concrete cases described, many embedded SW
References


References

References


References

Thank you for your attention!

Questions?