Cool Features and Tough Decisions

A Comparison of Variability Modeling Approaches

https://doi.org/10.1145/2110147.2110167
https://doi.org/10.1145/3307630.3342399

Krzysztof Czarnecki
Univ. Waterloo
Canada
czarnecki@acm.org

Paul Grünbacher
JKU Linz
Austria
paul.gruenbacher@jku.at

Rick Rabiser
JKU Linz
Austria
rick.rabiser@jku.at

Klaus Schmid
Univ. Hildesheim
Germany
schmid@sse.uni-hildesheim.de

Andrzej Wąsowski
ITU Copenhagen
Denmark
wasowski@itu.dk

VaMoS – Leipzig, Germany, January 2012

Context – Why a Comparison?

- Numerous variability modeling (VM) approaches exist today
- Most based on feature modeling (FM) or decision modeling (DM)
  - Surveys on FM ***or*** on DM exist -- **so far, no systematic comparison**

- **Many cool features have been added to FM and DM over the years**
- **Its tough to decide which approach to use for what purpose**

- We aim to
  - Systematize the research field and explore potential synergies
  - Improve the understanding of the range of VM approaches
  - Provide insights to users adopting VM in practice
  - Help with the standardization of VM

- Goal is NOT to find out which is better but to point out commonalities and differences – **FM and DM are converging!**

Background and History

FM

features – end user's understanding of the general capabilities of systems in the domain *– and the relationships among them*

- FODA method (1990)
- Many, many extensions, e.g.,
  - Group cardinalities [Riebisch et al. '02]
  - Feature cardinalities [Czarnecki et al. '05]
  - Feature inheritance [Asikainen et al. '06]
- Integral part of FOSD
- Several surveys, e.g., [Hubaux et al. 2010, Schobbens et al. 2006, etc.]

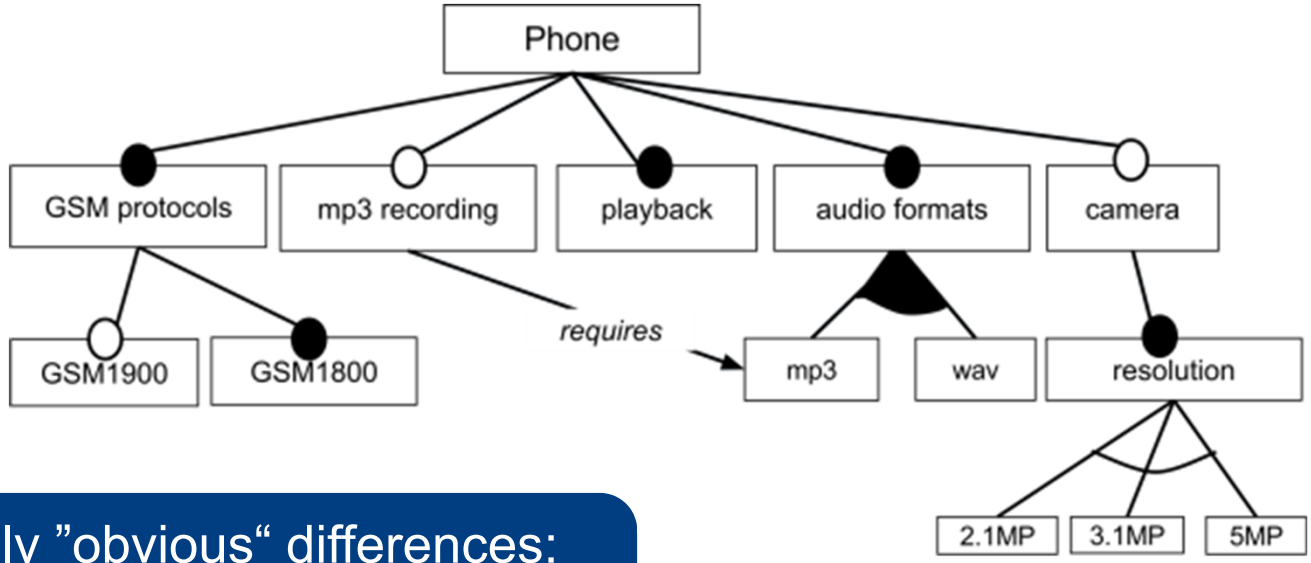
DM

set of ***decisions*** *adequate to* ***distinguish among the members*** *of a product family useful to* ***guide*** *the adaptation of* ***application engineering*** *work products*

- Synthesis method (1991)
- Diverse approaches, e.g.,
  - FAST [Weiss and Lai 1999]
  - DOPLER [Dhungana et al. 2011]
  - Schmid and John [Schmid and John 2004]
- Most inspired by industrial applications
- Survey [Schmid et al. 2011]

Examples

FM



tree notation, slighty adapted from FODA [Kang et al. 1990]

Seemingly ”obvious“ differences:

- commonalities
- hierarchy

DM

decision name	description	type	Range	cardinality/constraint	visible/relevant if
GSM_Protocol_1900	Support GSM 1900 protocol?	Boolean	true | false		
Audio_Formats	Which audio formats shall be supported?	Enum	WAV | MP3	1:2	
Camera	Support for taking photos?	Boolean	true | false		
Camera_Resolution	Required camera resolution?	Enum	2.1MP | 3.1MP | 5MP	1:1	Camera == true
MP3_Recording	Support for recording MP3 audio?	Boolean	true | false	ifSelected Audio_Formats.MP3 = true	

tabular notation, combining concepts from [Schmid and John 2004] and [Dhungana et al. 2011]

Development of our Comparison

- Started at Dagstuhl Seminar on FOSD in Jan 2011
- Extraction of 10 dimensions from existing surveys, i.e., Berger et al. ASE 2010 and Schmid et al. VaMoS 2011
- Several meetings and telephone conferences
- Our results are based on:
  - our experiences as experts in DM/FM
  - our knowledge of the literature in these fields
  - other comparison frameworks
  - discussion with other people in the community
  - reviewers' detailed comments

Dimension	Feature Modeling	Decision Modeling
Applications	**div. applications**: concept modeling, variability and comm. modeling; derivation support	variability modeling; **derivation** support
Unit of variability	**features**	**decisions**
Orthogonality	**mostly** used in **orthogonal** fashion	**orthogonal**
Data types	**comprehensive set of basic types**	
Hierarchy	**essential** concept, single appr.	**secondary** concept, div. appr.
Dependencies and Constraints	**no standard** constraint language but similar range of approaches (Boolean, numeric, sets)	
Mapping to artifacts	**optional** aspect (no standard mechanism)	**essential** aspect (no standard mechanism)
Binding time and mode	not standardized, **occasionally supported**	
Modularity	**no standard** mechanism; feature hierarchy plays partly this role	**no standard** mechanism; decision groups play partly this role
Tool aspects	mainly **trees**	div. vis. incl. tree, **workflow**

Unit of variability: key concepts that are used to model variability

FM

- Features
- Highly overloaded term
- Characteristic of a concept (e.g., system, component, etc.) that is relevant to some stakeholder of the concept

DM

- Decisions
- Differences among systems
- Anything that an application engineer needs to decide during derivation

Mobile Phone example
GSM 1800 is mandatory → is a feature, but no decision needed.

Engineer "only" needs to decide whether a particular phone will support the GSM 1900 protocol or not.

Data types: available primitive values and composite structures for configuration

FM

- Boolean implicit in optional features
- composite types by relying on hierarchy, group constraints, and feature cardinalities
- Some support reference types – values are references to instances of other features

DM

- Boolean either explicit or encoded as an enumeration
- All DM notations offer enumerations as primitive data types and some offer records or sets or both

Comparable range in FM and DM

Many FM and DM notations support additional primitive types, including strings, integers, and reals. Synthesis includes even date and time.

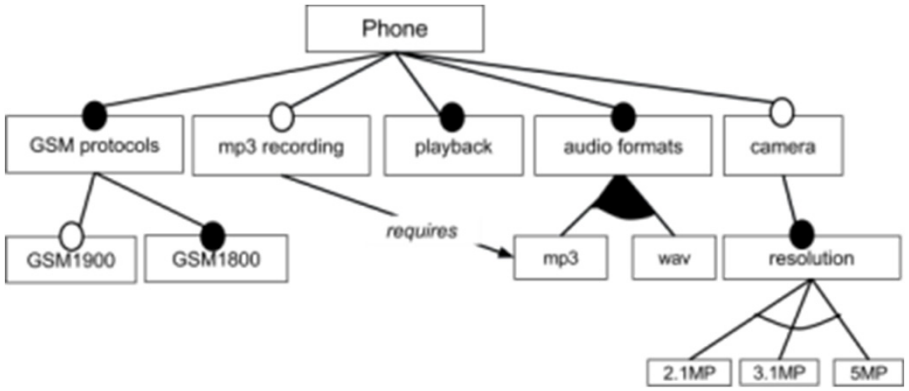
Hierarchy: organization of units of variability

FM

- Supported in all approaches as an **essential** concept
- Feature hierarchy imposes configuration constraints
  - selecting a feature implies selecting its parent

DM

- **Secondary** concept
- Supported differently by approaches, e.g., decision groups or visibility conditions
- To guide configuration process



decision name	visible/relevant if
Camera	
Camera_Resolution	Camera == true

Hierarchy: organization of units of variability

FM

- Supported in all approaches as an **essential** concept
- Feature hierarchy imposes configuration constraints
  - selecting a feature implies selecting its parent

DM

- **Secondary** concept
- Supported differently by approaches, e.g., decision groups or visibility conditions
- To guide configuration process

***Both** FM and DM **support hierarchy**.*

The main difference is that FM follows a single approach while in DM all approaches differ.

Mapping to artifacts: features or decisions just abstract variabilities in dev. artifacts

FM

- **Optional** aspect
- Supported by several approaches

DM

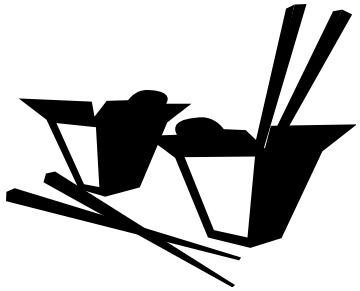
- **Essential** aspect
- Supported by all approaches

***Wide range** of mapping techniques **in both DM and FM**.*

Typically decisions or features (high-level variability abstractions) are related to variation points (locations in artifacts where variability occurs).

Some DM and FM approaches define a separate artifact model.

„Take-away“ Message
4 key differences of FM and DM



FM

- Focus on modeling commonalities and differences
- Hierarchy essential with uniform semantics
- Mapping to artifacts optional
- Focus on analysis and modeling

DM

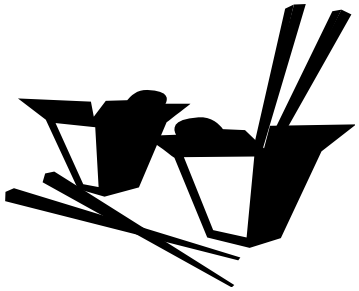
- Focus on modeling differences
- Hierarchy secondary with varied semantics
- Mapping to artifacts essential
- Focus on application engineering

More commonalities than differences; differences are mainly historical!

Conclusions

- **Significant convergence** between FM and DM
  - practical VM approaches combine concepts from FM and DM
- **Specific capabilities** of a VM approach are much **more important** when selecting an approach **than classification as DM or FM**
  - data types offered, expressiveness of the constraint language, support for modularity, available tool support

Added for MODEVAR: towards a simple, standard variability modeling language



- support the **typical basic data types** known from programming languages and **some type of composite**
- be **orthogonal** and independent of specific artifacts
- provide a simple and clear concept to realize **hierarchy** and **modularity**
- offer a simple and expressive way to define **constraints** and **dependencies** including **mapping to** concrete **artifacts**
- support different **use cases** such as domain analysis or product configuration, but have a clear focus on the core use case: variability modeling
- consider **binding time and mode**
- be as **tool-independent** as possible, i.e., allowing to define models with standard text editors as well as fully-fledged IDEs