PLE Tool Support

KV Product Line Engineering (343.354)
Dr. Roberto Lopez-Herrejon
Dr. Rick Rabiser

27.4.2016

Why and when to use a Tool?

JOHANNES KEPLER UNIVERSITY LINZ

Fixed project size (*10 features*)–Varying process (*regular process* and *advanced process*)

Requirements planning process activities	Simple process: *regular process*						Complex process: *advanced process*					
	without tool			with tool			without tool			with tool		
	unit effort	execu-tions	total effort	unit effort	execu-tions	total effort	unit effort	execu-tions	total effort	unit effort	execu-tions	total effort
Capture raw requirement	46	12	552	9	12	108	66	12	792	15	12	180
Resolve raw requirements	0.3	89	26.7	10.4	89	925.6	6.8	89	605.2	13.2	89	1174.8
Resolve problem statement	7	32	224	2.9	32	92.8	10.9	32	348.8	3.2	32	102.4
Resolve features	8.3	10	83	3.2	10	32	97.4	10	974	37.7	10	377
Develop release plan	0*	2	0	0	2	0	180	2	360	68	2	136
Total effort			855			1158.4			3080			1970.2
Productivity	28.04			20.71			7.79			12.18		
Productivity Impact	- 26.1%						+ 56.4%					

* this activity does not involve subactivities affected by tool insertion.

[Bruckhaus et al. 1996]

→ SPLE comprises many complex processes!

Motivation

- Many activities/processes of SPLE can (and have to) be automated/supported by tools, e.g.,
  - Variability modeling
    - tools hide the concrete syntax behind a graphical user interface and help users to create models in an efficient way
  - Product derivation
    - tools help to make configuration decisions and support constraint propagation, automated conflict resolution, configuration generation
  - Product line planning/scoping
    - tools can facilitate effective collaboration, e.g., for the development of a feature plan
  - ...

Requirements for SPL tools

- Interoperability and Traceability
  - How to integrate SPL tools with (development) processes, teams, and tools already existing in an organization?
  - How to integrate SPL processes with each other (from PL planning to variability modeling to product derivation)?
- Adaptability
  - How to work with the assets and dependencies specific to a particular domain?
  - How to react to changes e.g., evolving technology or changing customer needs?
- Scaleability
  - How to support multiple teams/large organizations?
  - How to work with large systems?

Requirements for SPL tools

- Extensibility
  - How to add additionally required functionality (e.g., 3rd party capabilities)?
- Usability
  - How „easy“ can different user groups work with the tools?
    - Especially important for „non-software-technicians“
    - Learnability, Efficiency, Memorability, Accuracy, Subjective Satisfaction [Nielsen 1994]
- Automation
  - Which tasks can be automated?
  - Which tasks require user intervention?
- Error handling
  - How to react to problems?

Custom-developed „PL“ Tools

JOHANNES KEPLER UNIVERSITY LINZ

- Typical scenario:
  - Company recognized need to manage variability
    - e.g., too many projects with customers having too many specific requirements, project-specific development no longer accomplishable
  - Company implements their own solution to manage variability
    - Often very close to the code
      - E.g., parameterization, config files, IFDefs
    - Variability is usually not explicitly documented (e.g., in a model)
    - Thus: management of variability becomes a maintenance problem in itself
- However, there are many commercial and academic PL tools existing
  - They are just not well-known and many of them unfortunately are hard to adopt for practical scenarios

SPL Tools – An Overview

- **The DOPLER Tool Suite**
- **Kumbang Tools**
- **pure::variants**
- **Gears**
- **Feature IDE**
- **Other tools**

The DOPLER Tool Suite

CHRISTIAN DOPPLER LABORATORY FOR
AUTOMATED SOFTWARE ENGINEERING

SIEMENS
VAI

JOHANNES KEPLER
UNIVERSITY LINZ | JKU

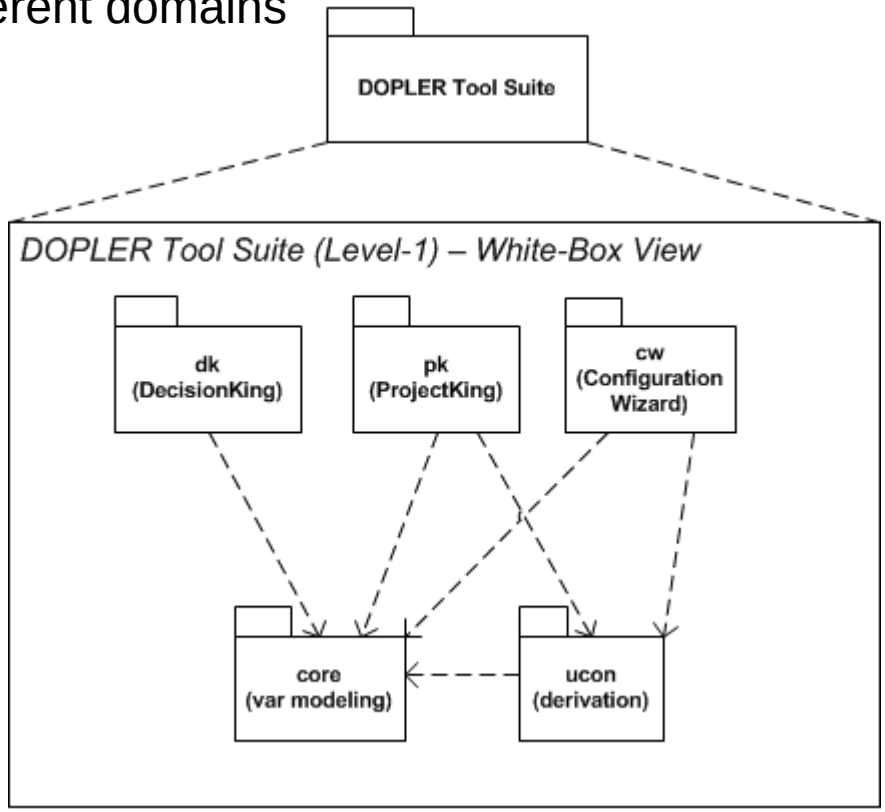
[Dhungana et al. 2006-2009] [Rabiser et al. 2006-2013]

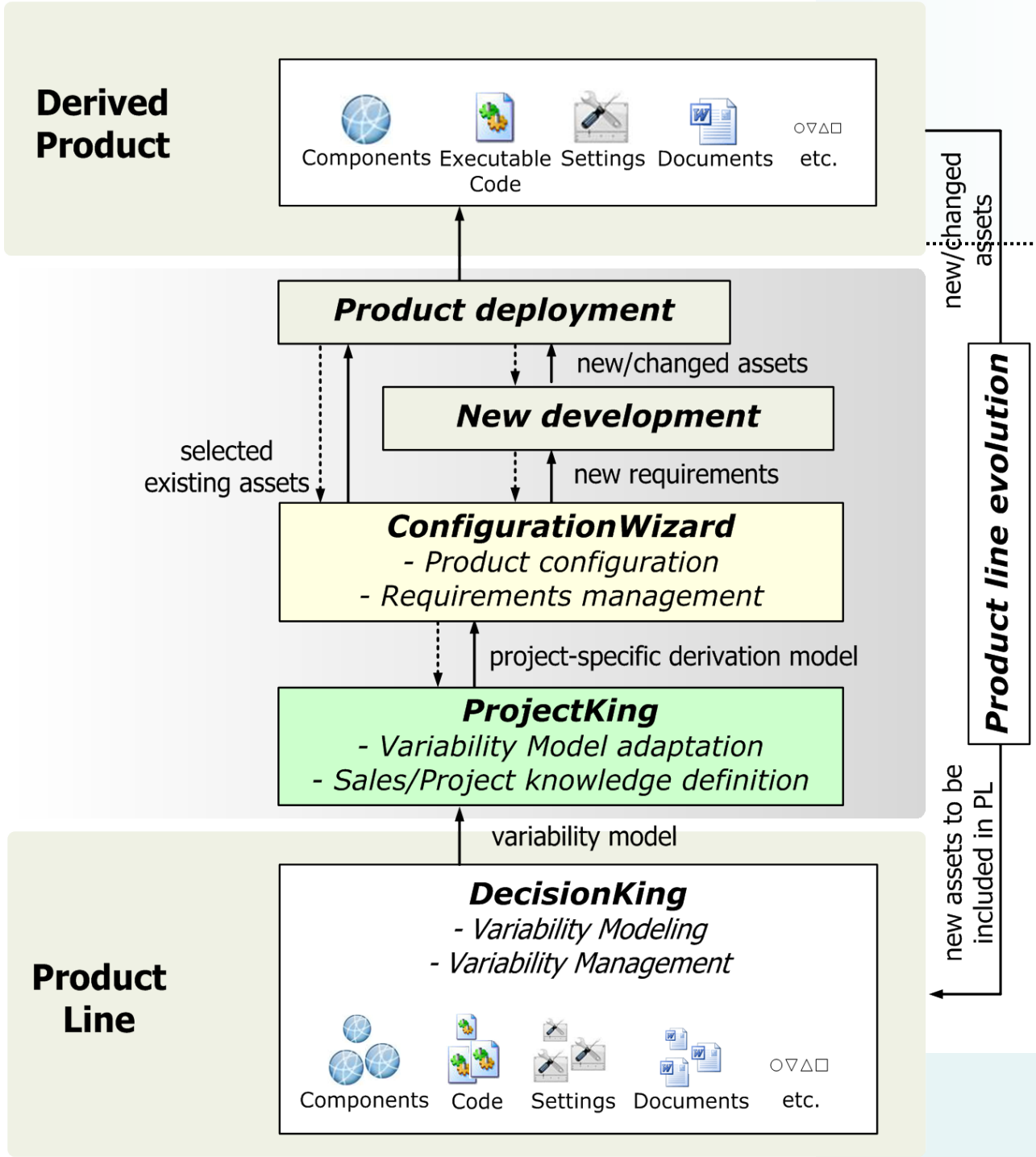
DOPLER Tool Suite

- Supports Decision-oriented modeling and Wizard-oriented usage of variability
- Developed at CD Lab for ASE, JKU in collaboration with Siemens VAI Metals Technologies GmbH and Siemens Corporate Technology
- Flexible and Extensible tool architecture
  - Meta-modeling capabilities for domain-specific adaptations
  - Plug-in Architecture to allow for arbitrary extensions
- Focuses on the needs of end-users for customizing complex products in an intuitive manner

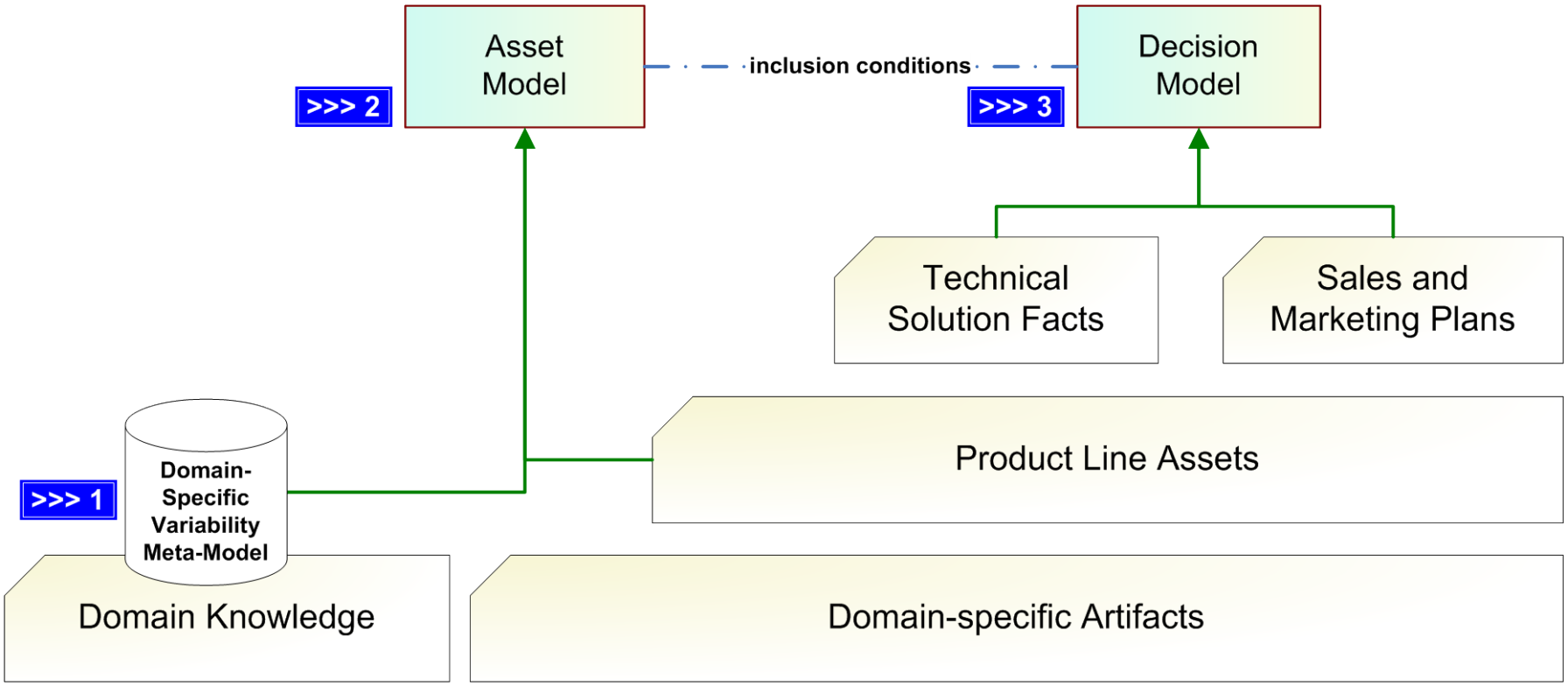
Overview

- 3 tools (composed of multiple plug-ins) and several (optional) extensions
- Customizable- but in general independent of languages, notations and practices in different domains

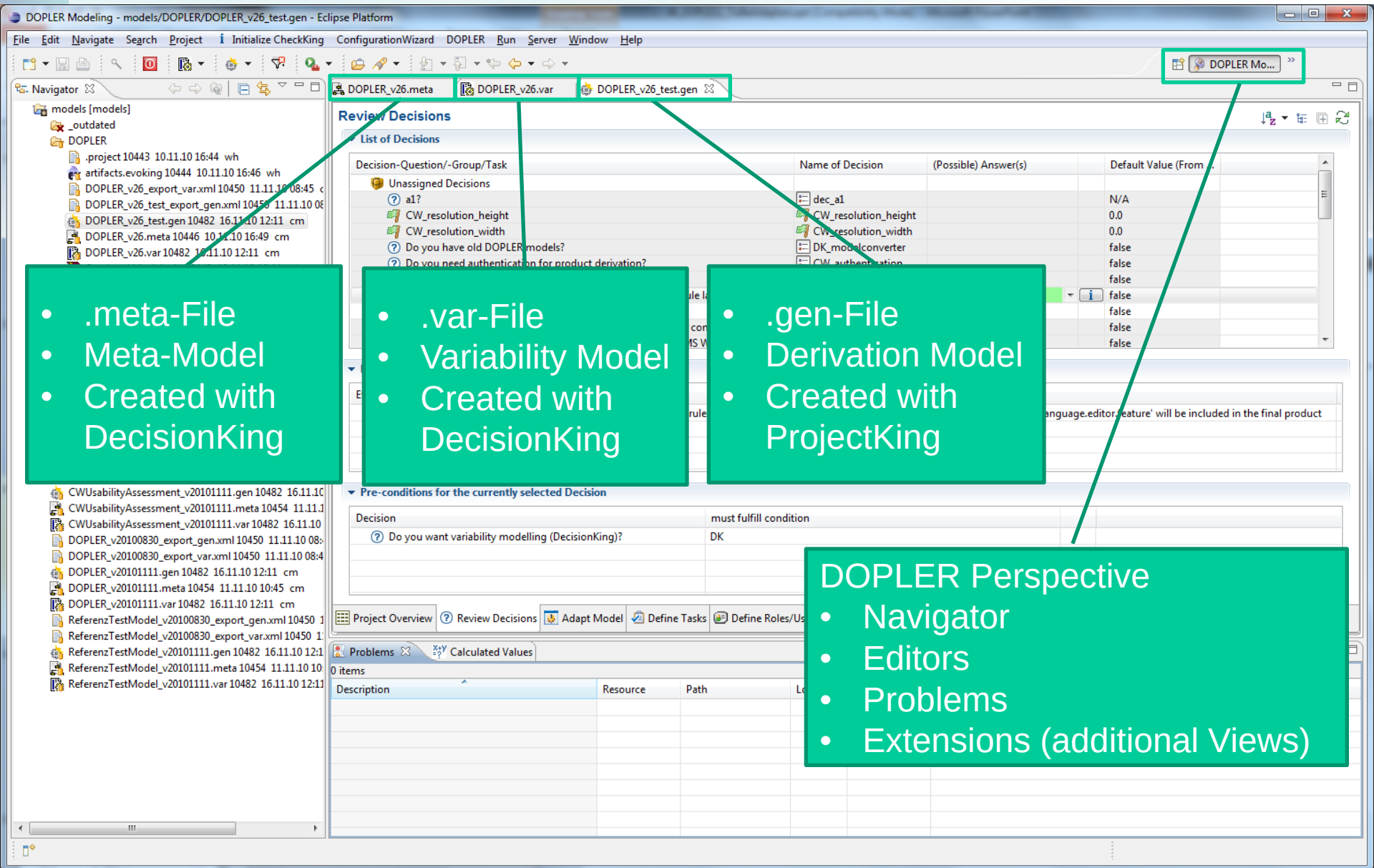



Derived Product
Components
Executable Code
Settings
Documents
etc.
Product deployment
new/changed assets
New development
selected existing assets
new requirements
ConfigurationWizard
- Product configuration
- Requirements management
project-specific derivation model
ProjectKing
- Variability Model adaptation
- Sales/Project knowledge definition
variability model
DecisionKing
- Variability Modeling
- Variability Management
Product Line
Components
Code
Settings
Documents
etc.
new/changed assets
Product line evolution
new assets to be included in PL

Modeling Approach

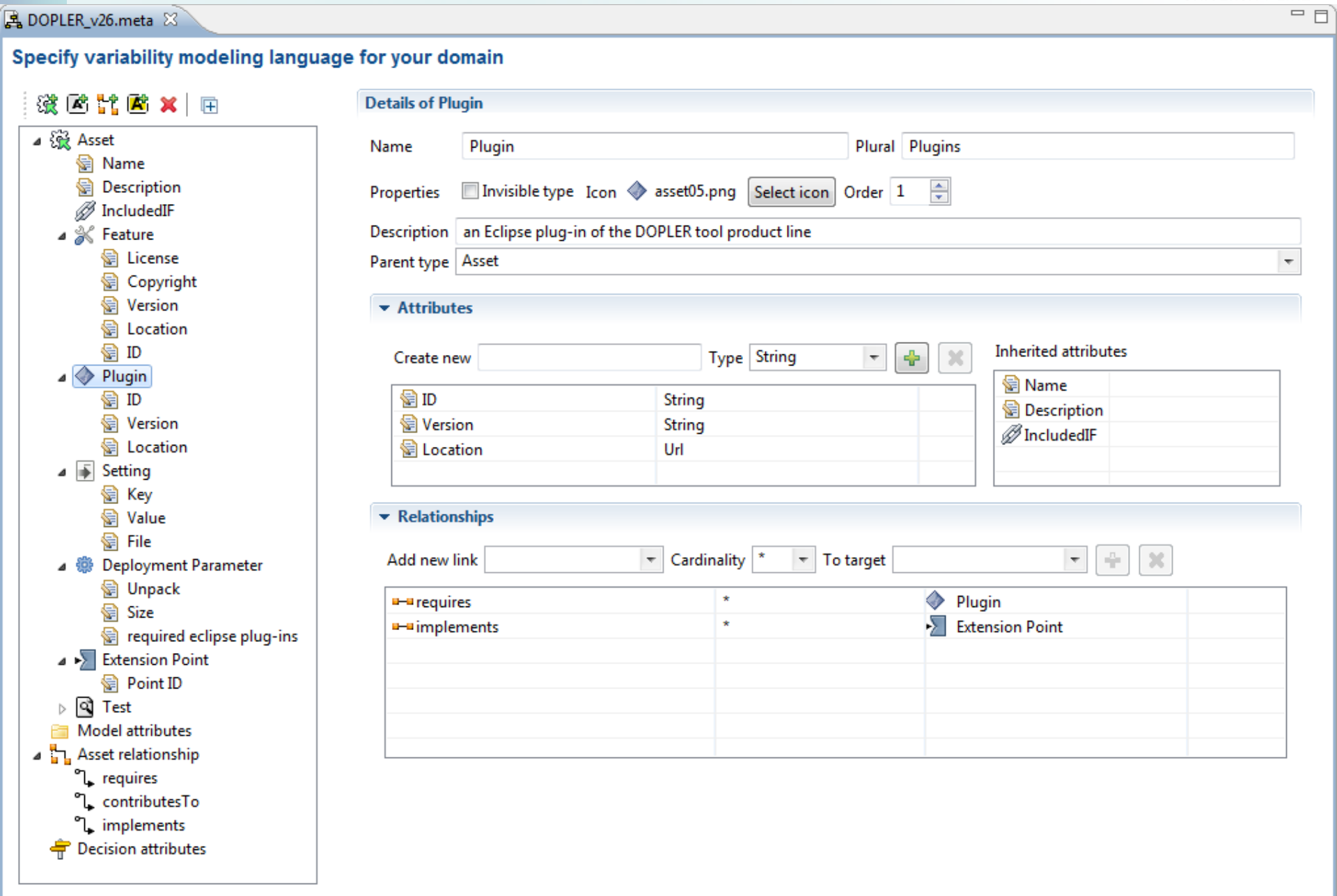


DOPLER Modeling Perspective



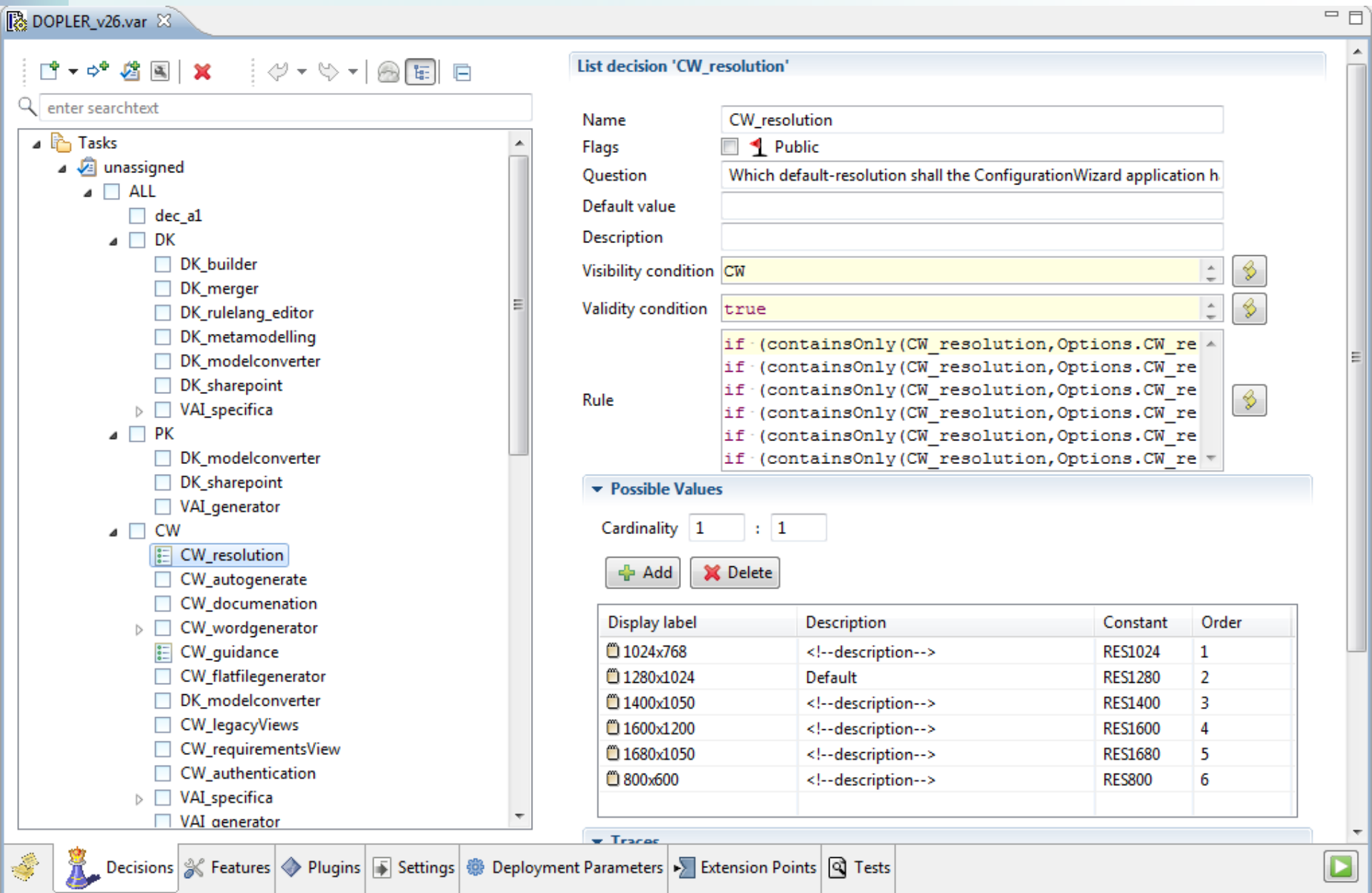
DecisionKing: Meta-model Editor

JOHANNES KEPLER UNIVERSITY LINZ



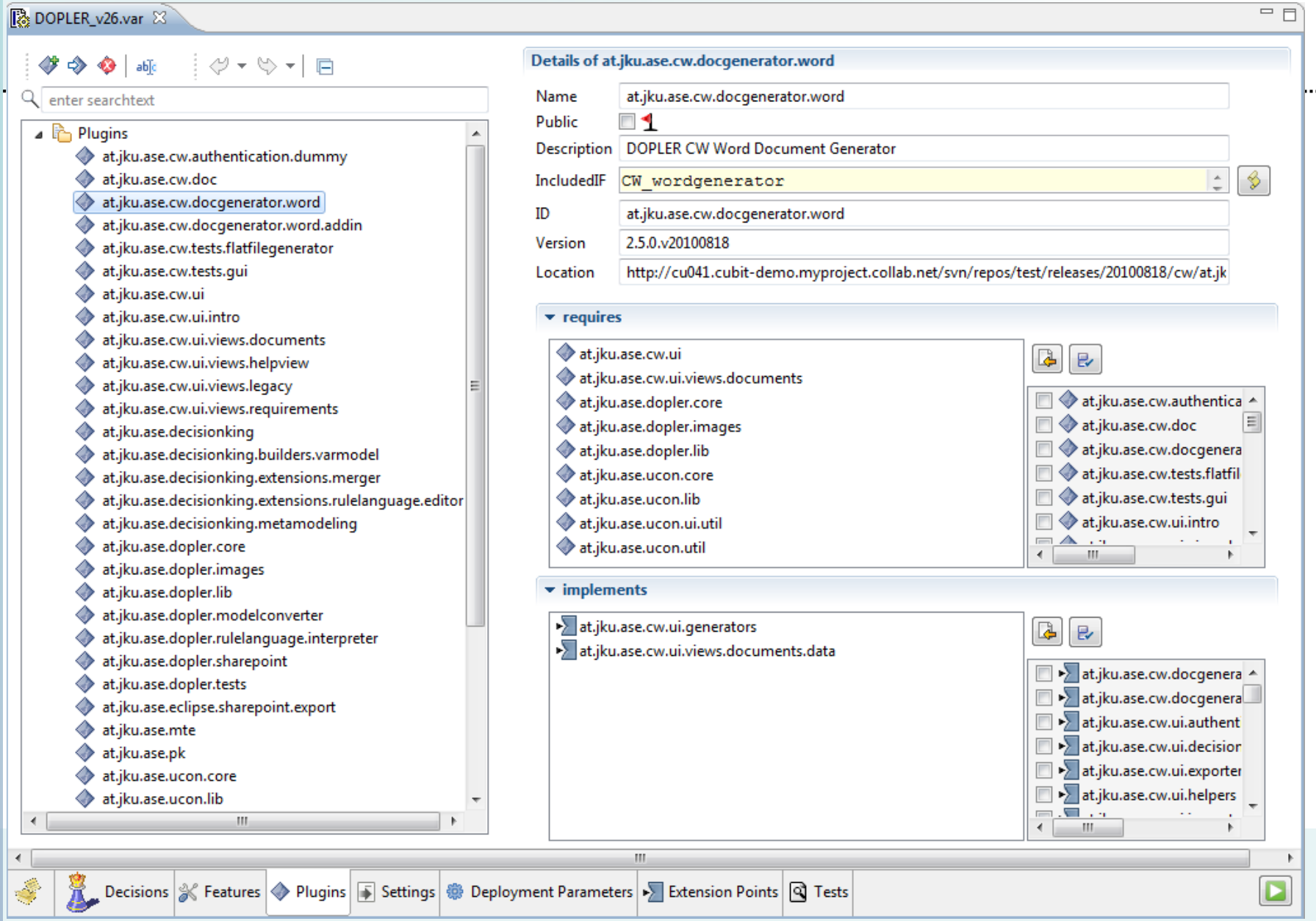
Variability Model Editor - Decisions

JOHANNES KEPLER UNIVERSITY LINZ



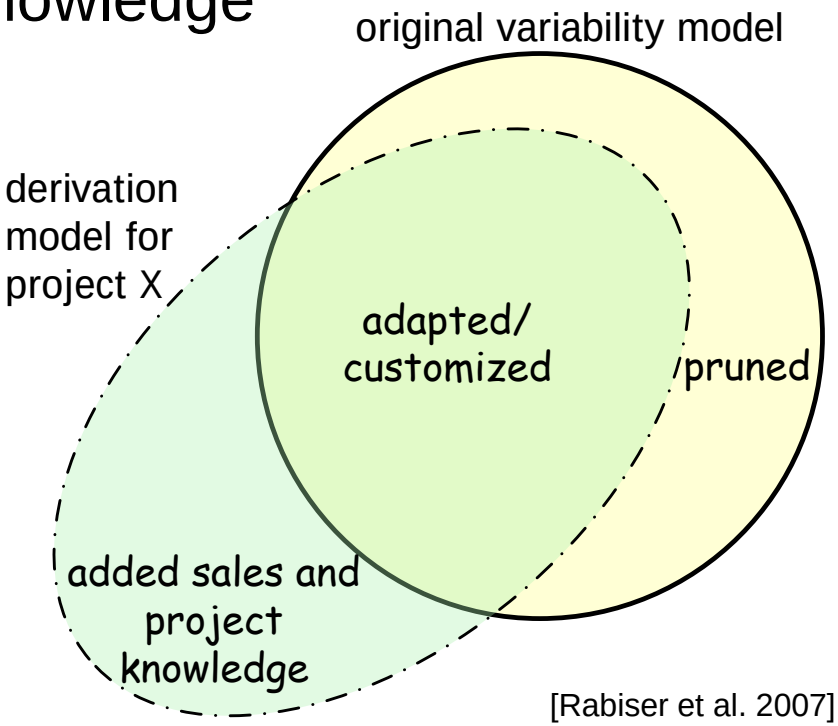
Variability Model Editor - Assets

JOHANNES KEPLER UNIVERSITY LINZ



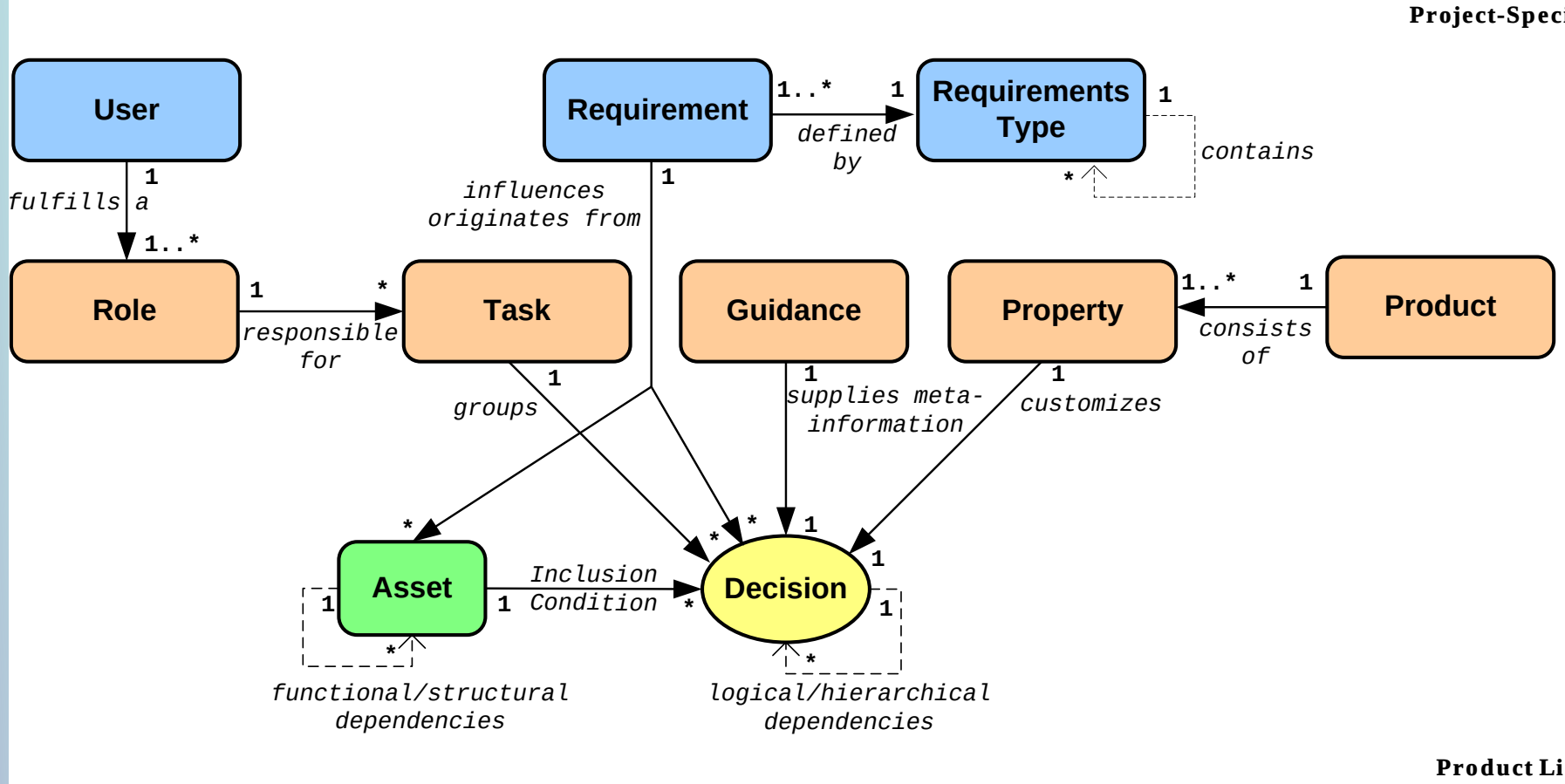
Product Derivation: The Concept of Derivation Models

- Prepare variability models for concrete projects
  - Identify irrelevant variability
  - Add project-specific/sales knowledge
- **Project-specific Derivation Model**
  - Adapted and augmented version of the original variability model
  - Tool-Support: ProjectKing

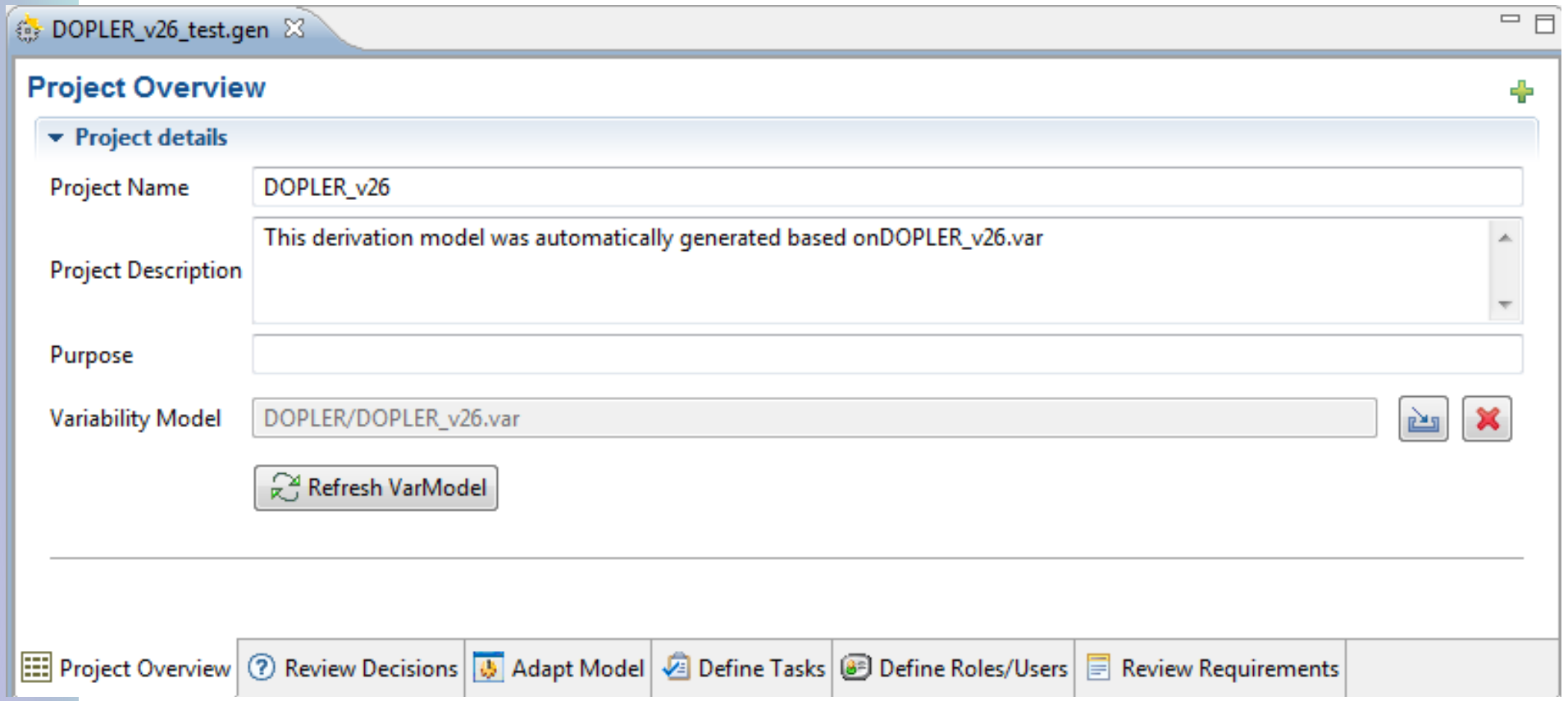


[Rabiser et al. 2007]

Derivation Meta-Model

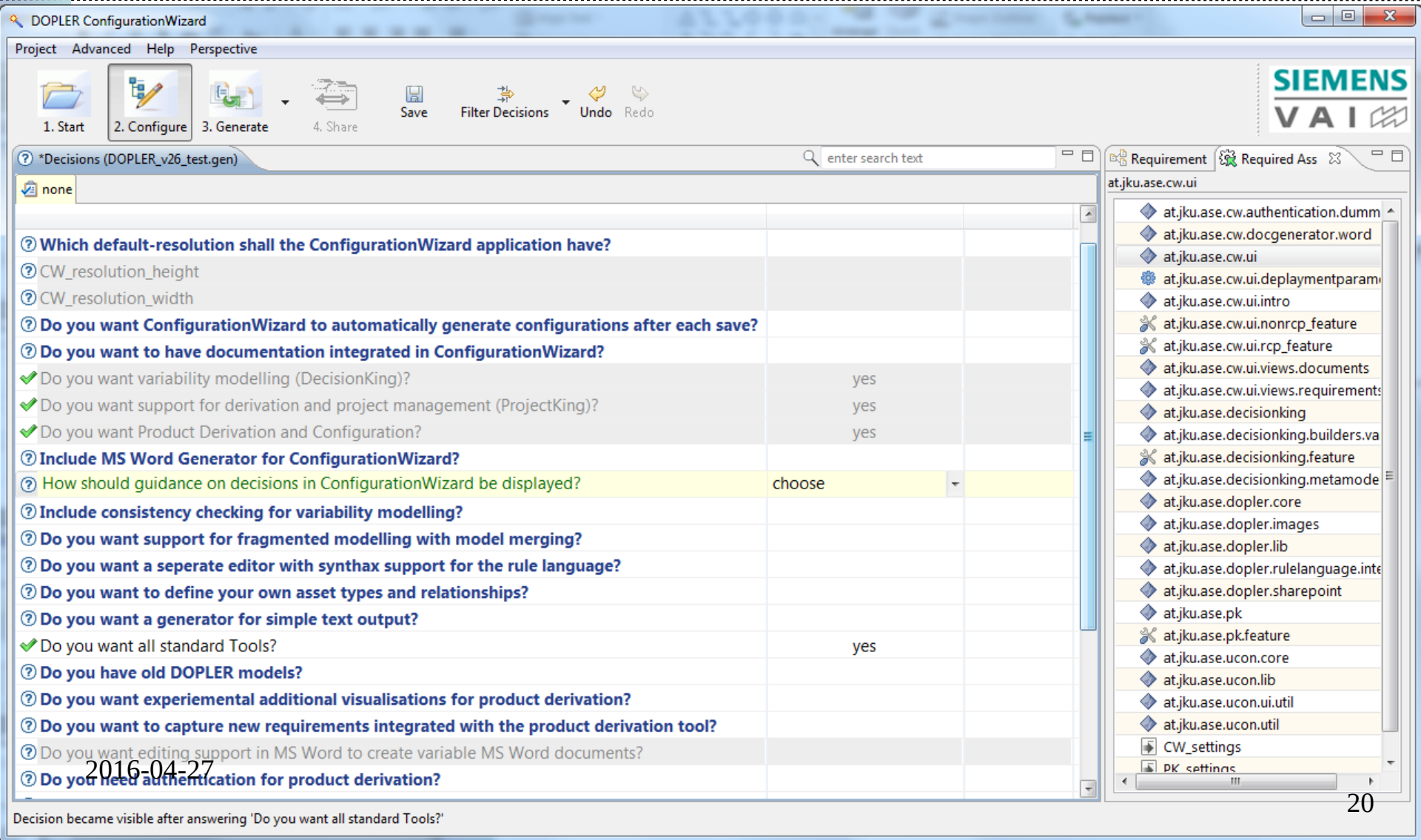


ProjectKing: Derivation Model Editor



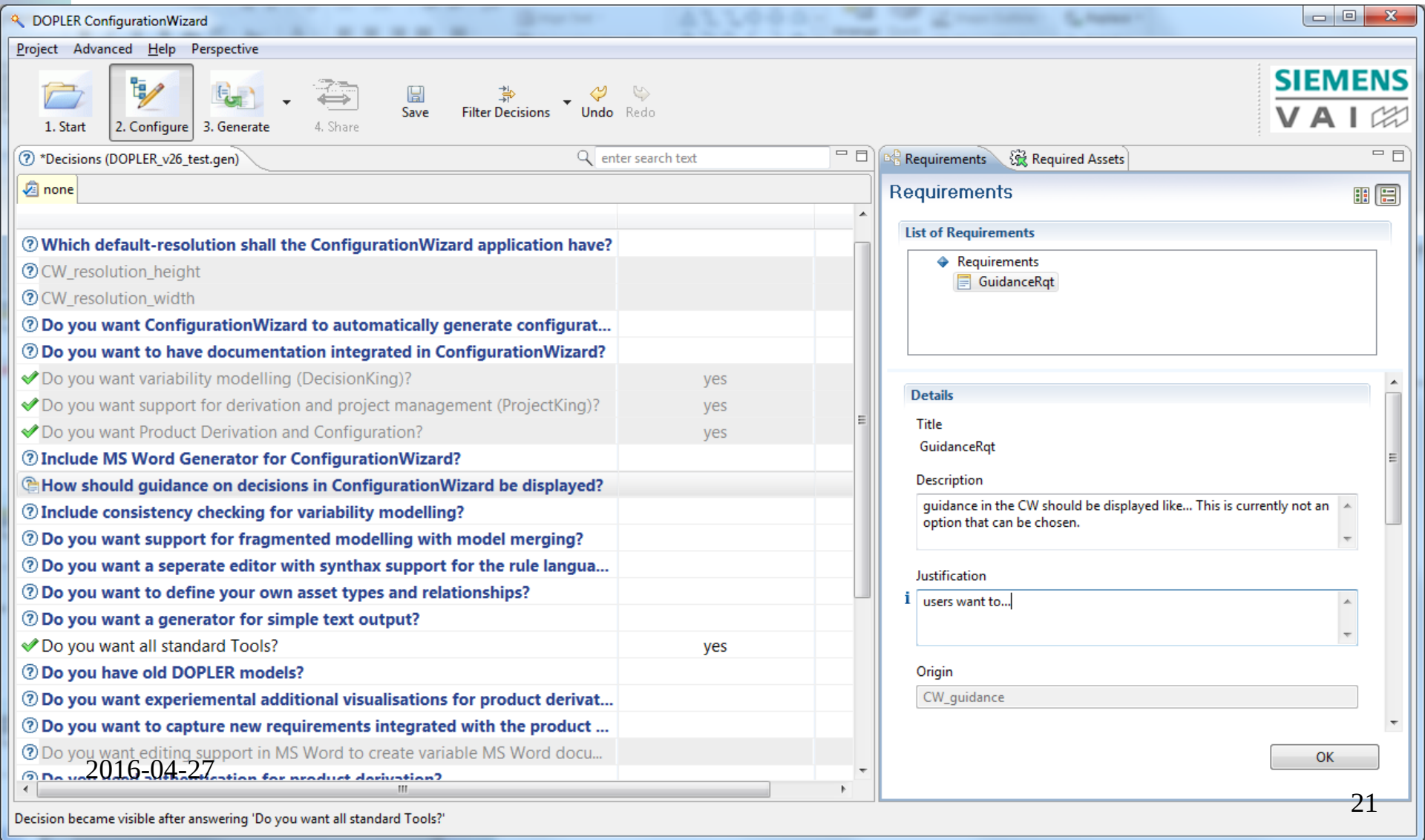
Product Derivation with ConfigurationWizard

JOHANNES KEPLER UNIVERSITY LINZ



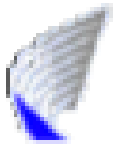
Application Reqs. Mgmt in ConfigurationWizard

JOHANNES KEPLER UNIVERSITY LINZ



Kumbang Tools

SoberIT
Software Business and Engineering Institute

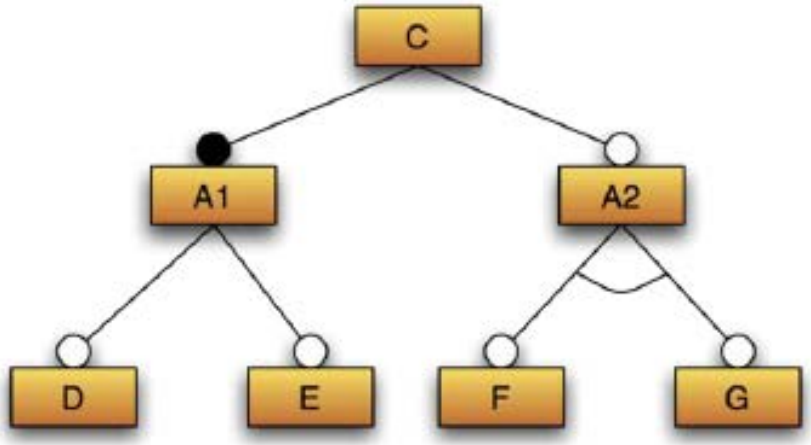


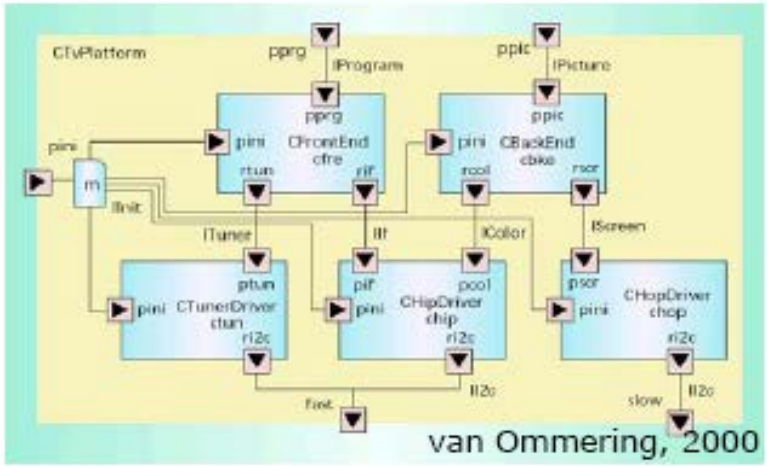
HELSINKI UNIVERSITY OF TECHNOLOGY

[Myllärniemi 2007]

Kumbang Tools

- Developed at Helsinki University of Technology, SoberIT
- Eclipse plug-ins, AI inference engine (smodels), GPL
- Basic Idea:
  - Combining feature modeling and structural modeling
  - Meta-Model called Kumbang





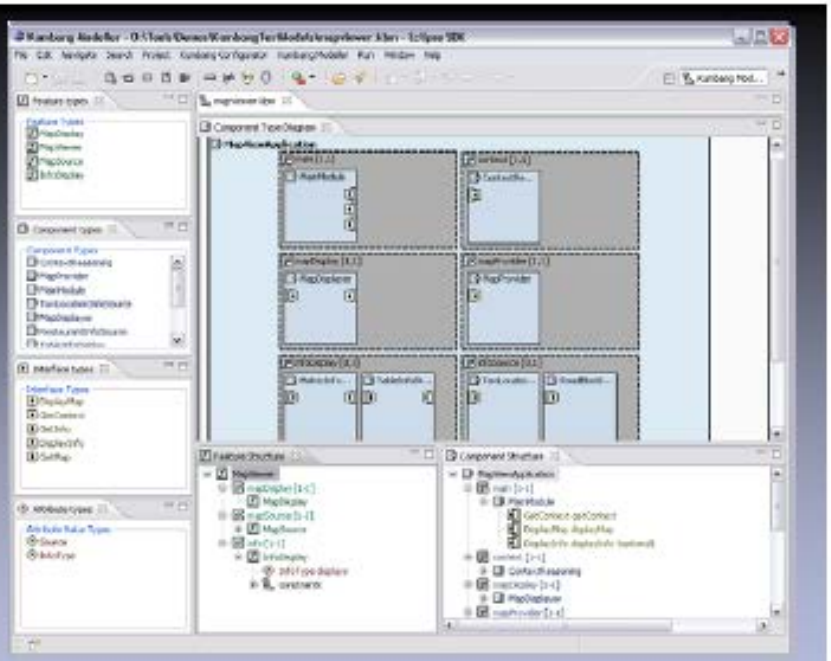
Kumbang Meta-Model

- Meta-model and language for configurable applications
- Modeling variability from two viewpoints adhering to IEEE 1471-2000 standard (Recommended Practice for Architecture Description of Software-Intensive Systems)
  - User-visible aspects: Feature Model
  - Structure of products in terms of components, interfaces, ports, etc.: Koala
  - Interrelations between FM/Koala
- Formal semantics by defining a mapping from the meta-model to Weight Constraint Rule Language

Kumbang Architecture Overview

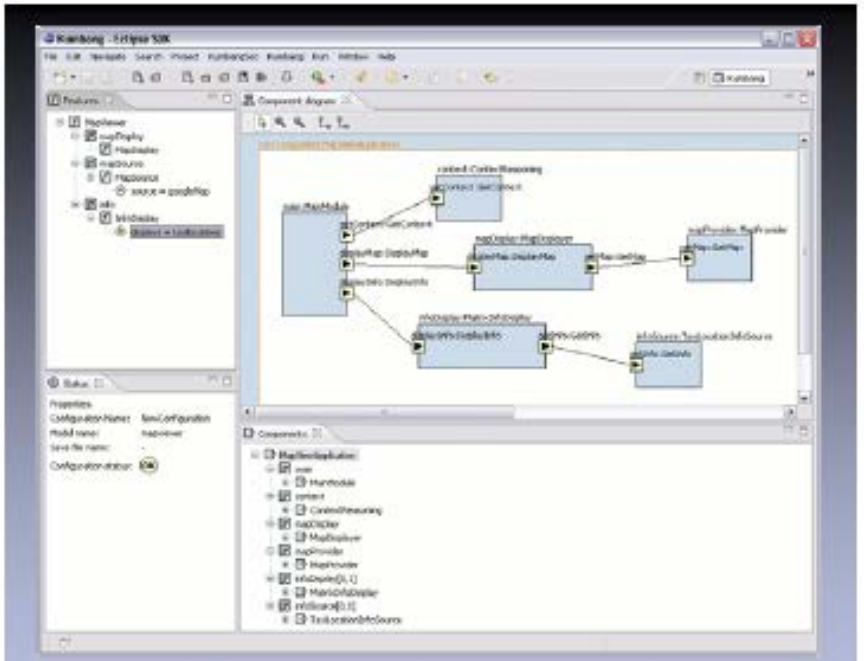
JOHANNES KEPLER UNIVERSITY LINZ

Types



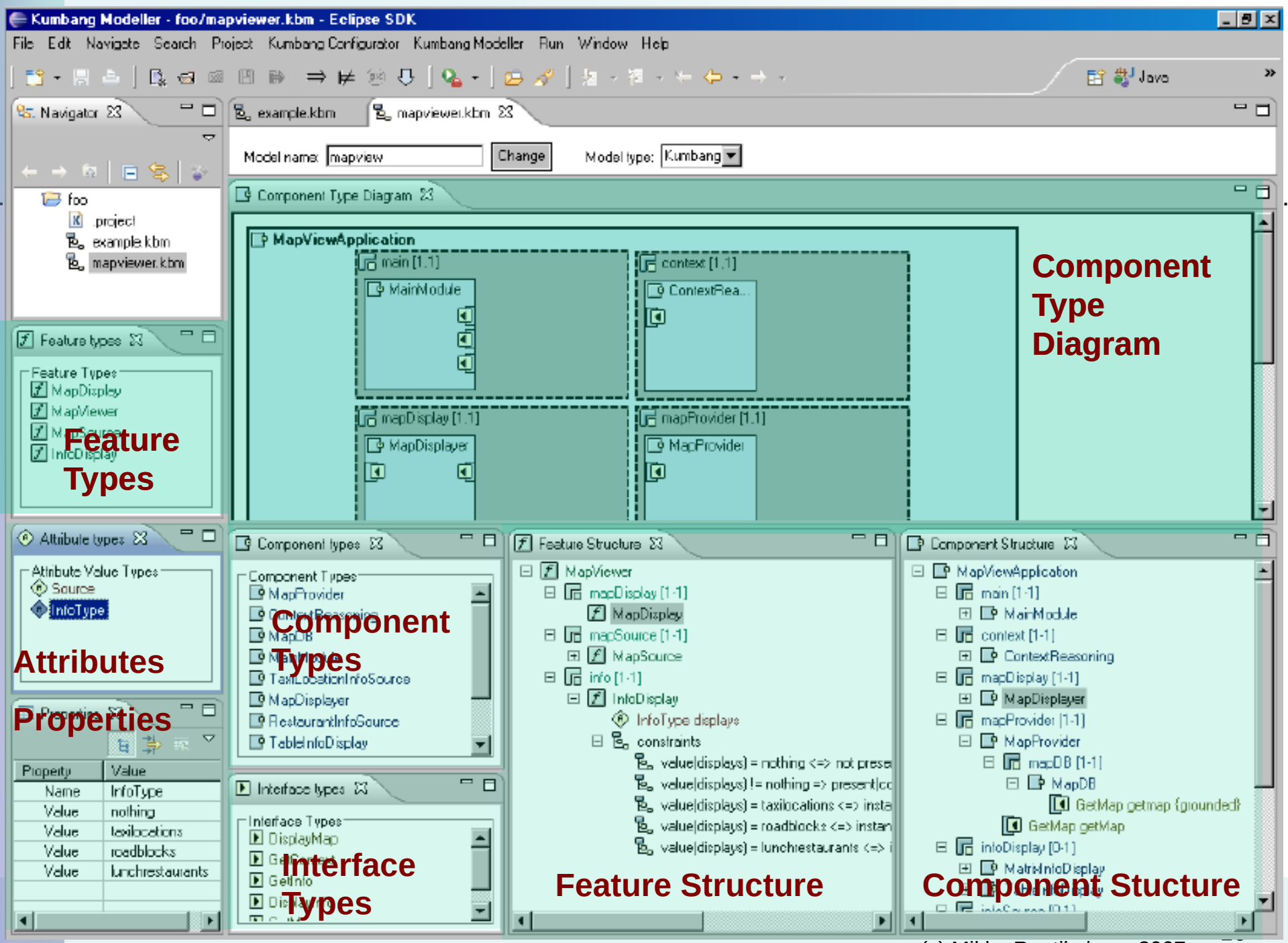
Domain engineering by Kumbang Modeller

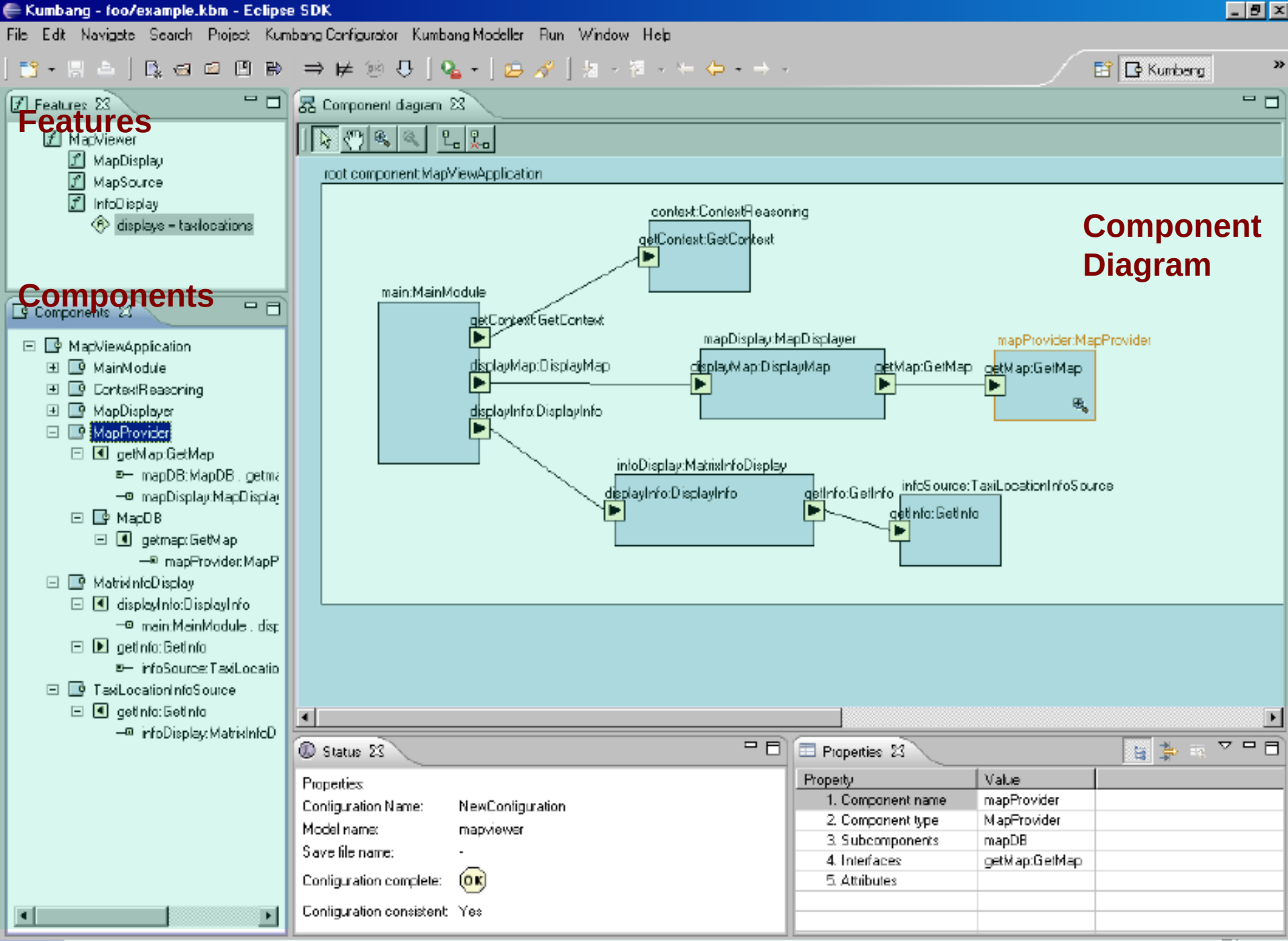
Instances



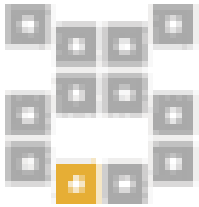
Application engineering by Kumbang Configurator

Kumbang Core
uses AI inference engine smodels





Pure::Variants



pure-systems

[pure systems GmbH]

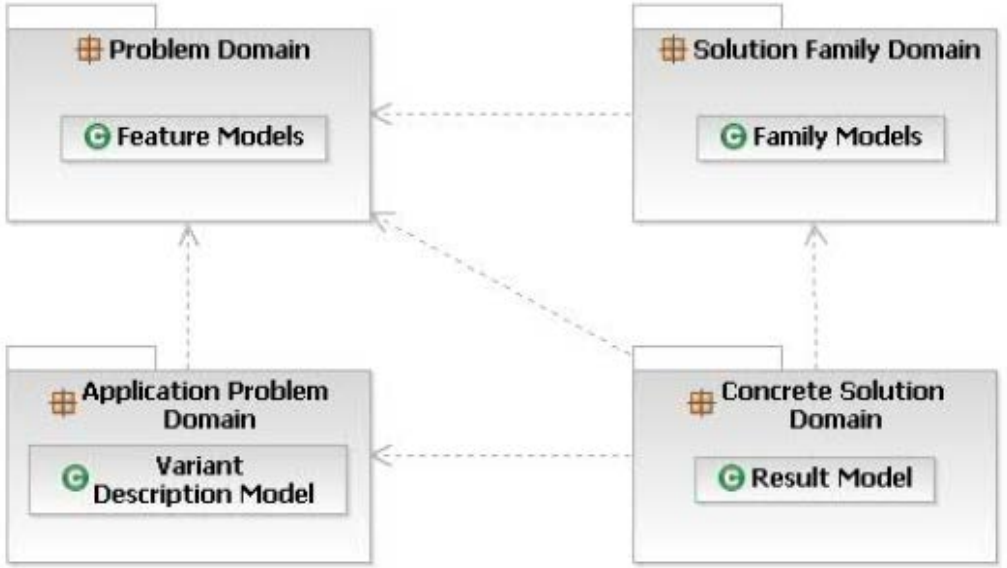
pure::variants

[pure systems GmbH]

- Developed by pure-systems GmbH/Magdeburg
- Commercial tool available in an evaluation, community, developer, integration, and server edition
- Variant management and product configuration based on feature models
- Uses a Prolog-based constraint-solver
- Strong focus on integration and extensibility
  - E.g., pv can be integrated in the Eclipse IDE, used with a webbrowser, in a custom application or as a command line client
  - Several extensions to other tools exist, e.g., to the requirements management tool DOORS or to the enterprise business tool SAP

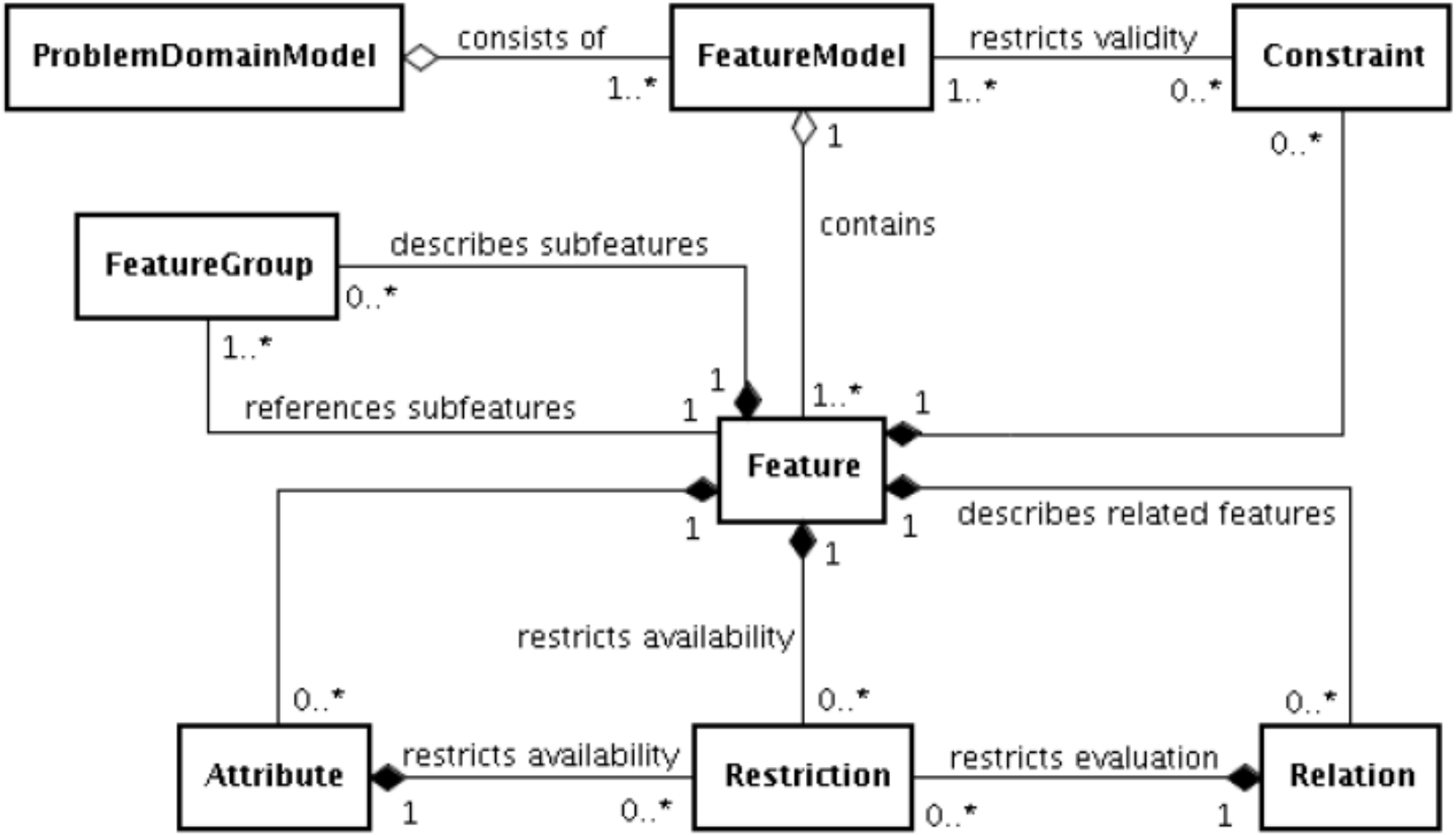
Family-based software development with pure::variants [pure systems GmbH]

- (Hierarchical) feature models to represent the problem domain
- Concrete design and implementation of the PL in family models
- Variant description model to store selected features and values
- Result model describes a concrete instance from the PL



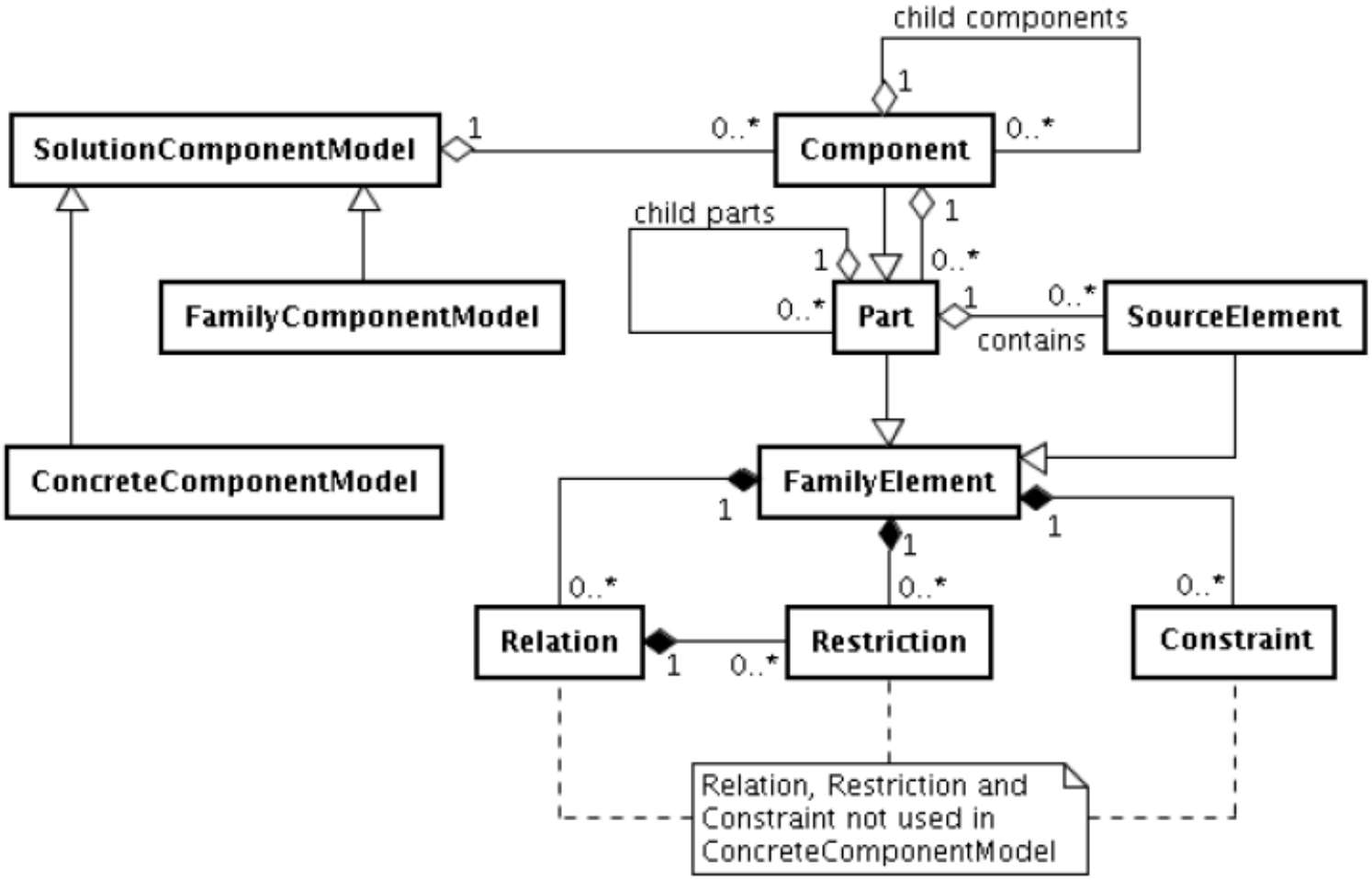
pure::variants feature models

[pure systems GmbH]



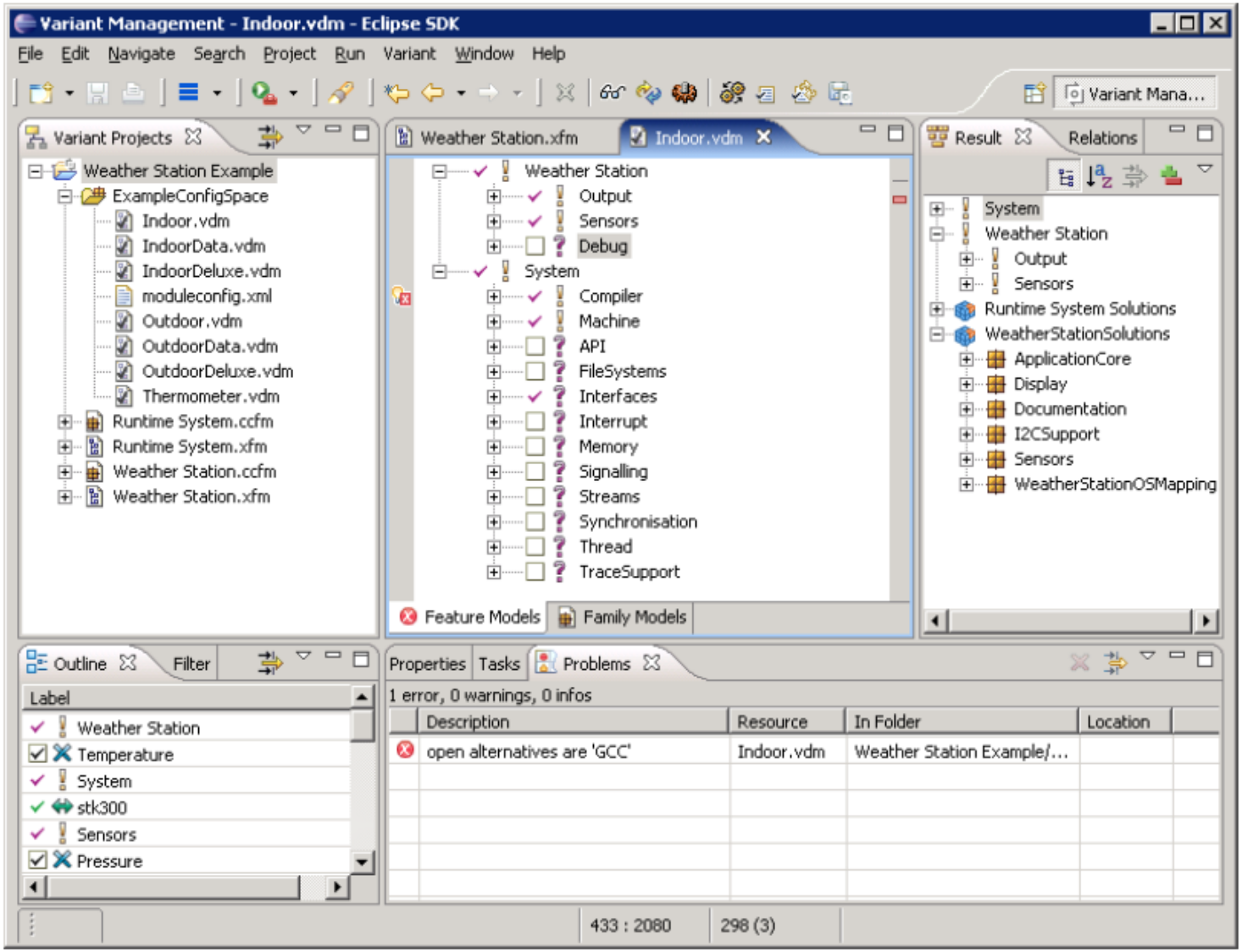
pure::variants family models

[pure systems GmbH]



pure::variants tool screenshot

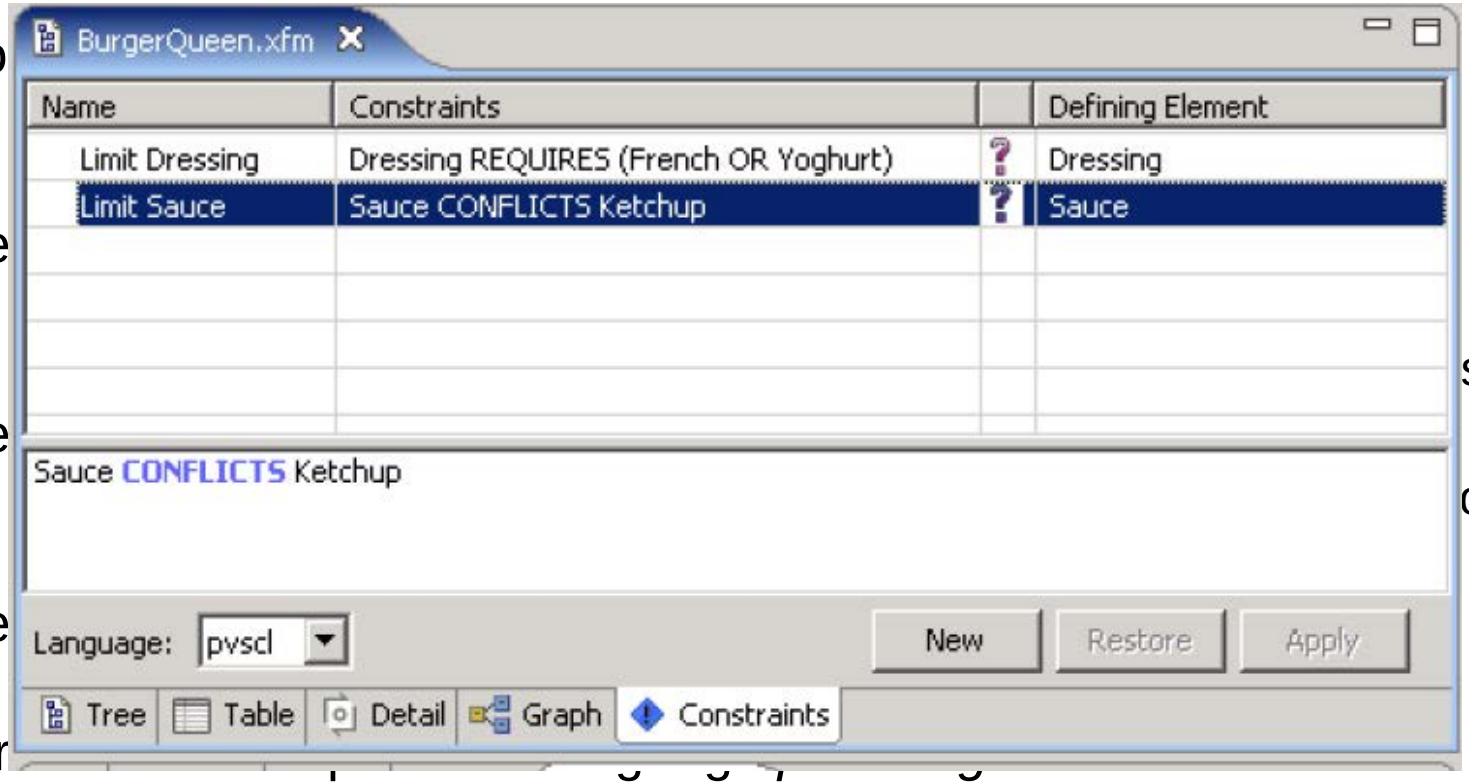
[pure systems GmbH]



pure::variants and constraints

[pure systems GmbH]

- Mo
  -
  -
- Ele
  -
  -
- Ele
  -
  -
- Ele
  -
- pur
  - Dialect of Prolog
  - Grammar and available rules, see: http://www.pure-systems.com/fileadmin/downloads/pure-variants/doc/pv-user-manual.pdf



Pure::Variants Screencast

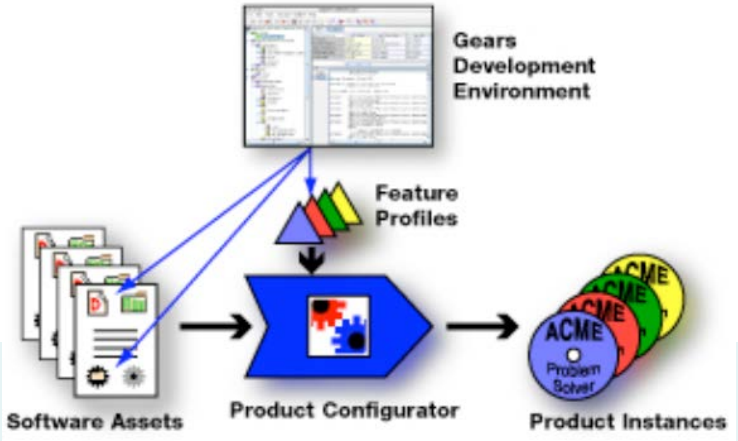
- 10min tutorial
  - http://www.pure-systems.com/flash/pv-spl-10min/flash.html

Gears

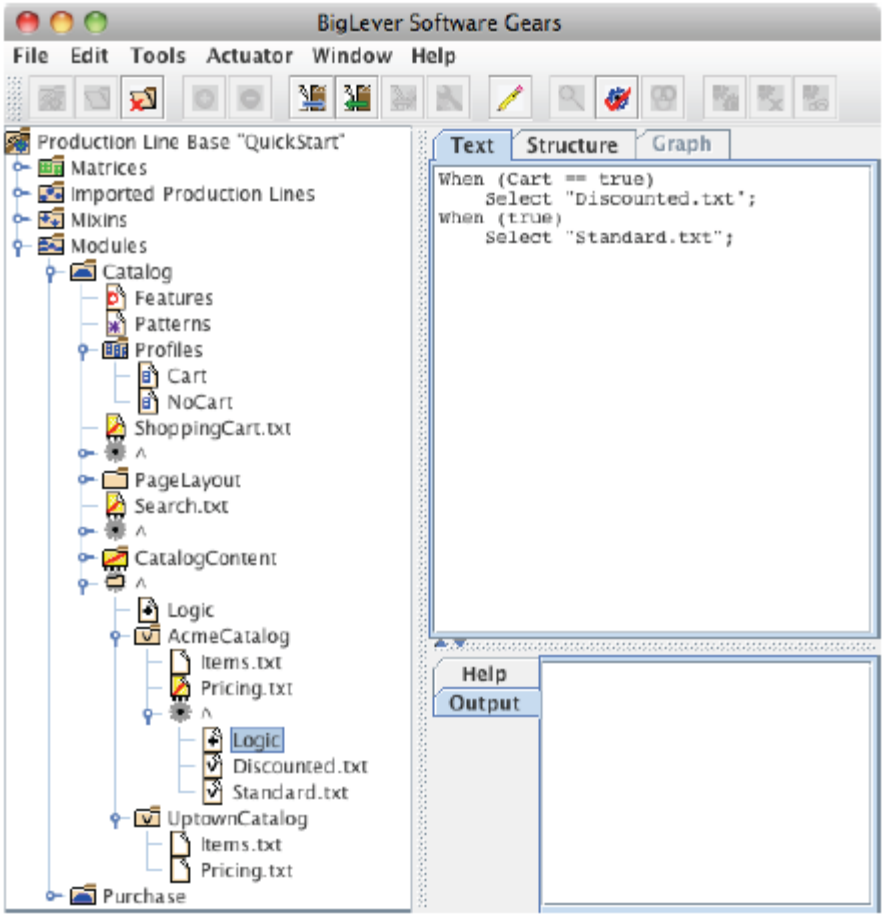
(c) BigLever, Inc.

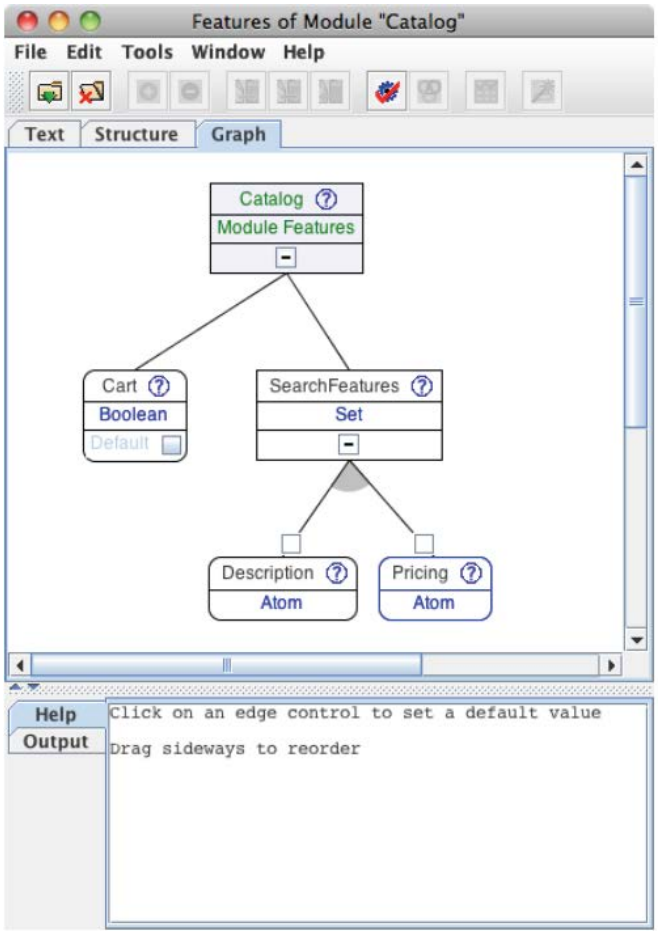
Basic Concepts of Gears

- 3 key elements:
  - Software Assets: the configurable software artifacts
  - Feature Profiles: model each product in terms of optional and varying feature choices for the PL
  - Product Configurator: assembles and configures the software assets, guided by the product feature profiles, to produce each of the product instances



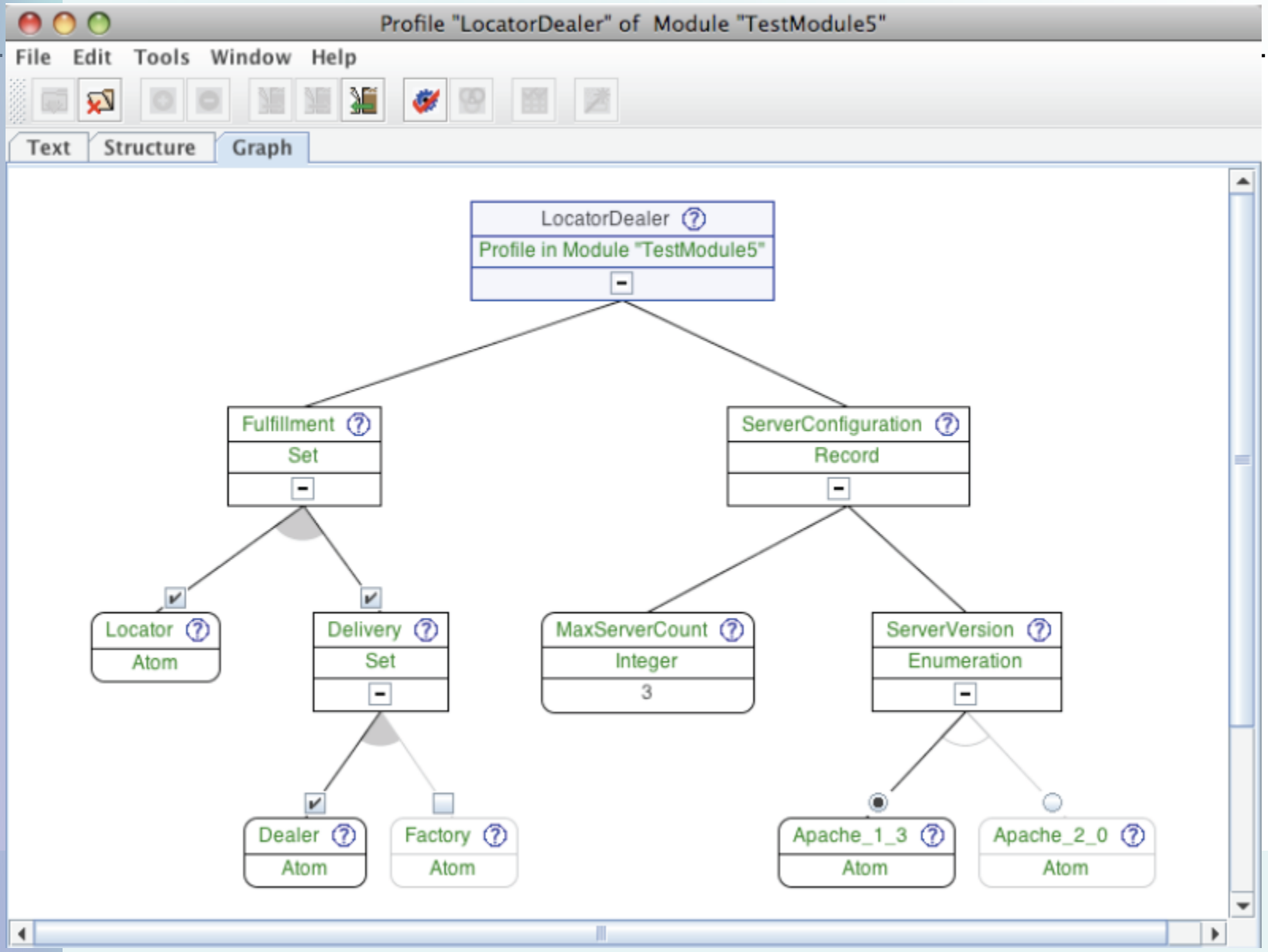
Gears Development Environment





Feature Profile Graph Editor

JOHANNES KEPLER UNIVERSITY LINZ



FeatureIDE

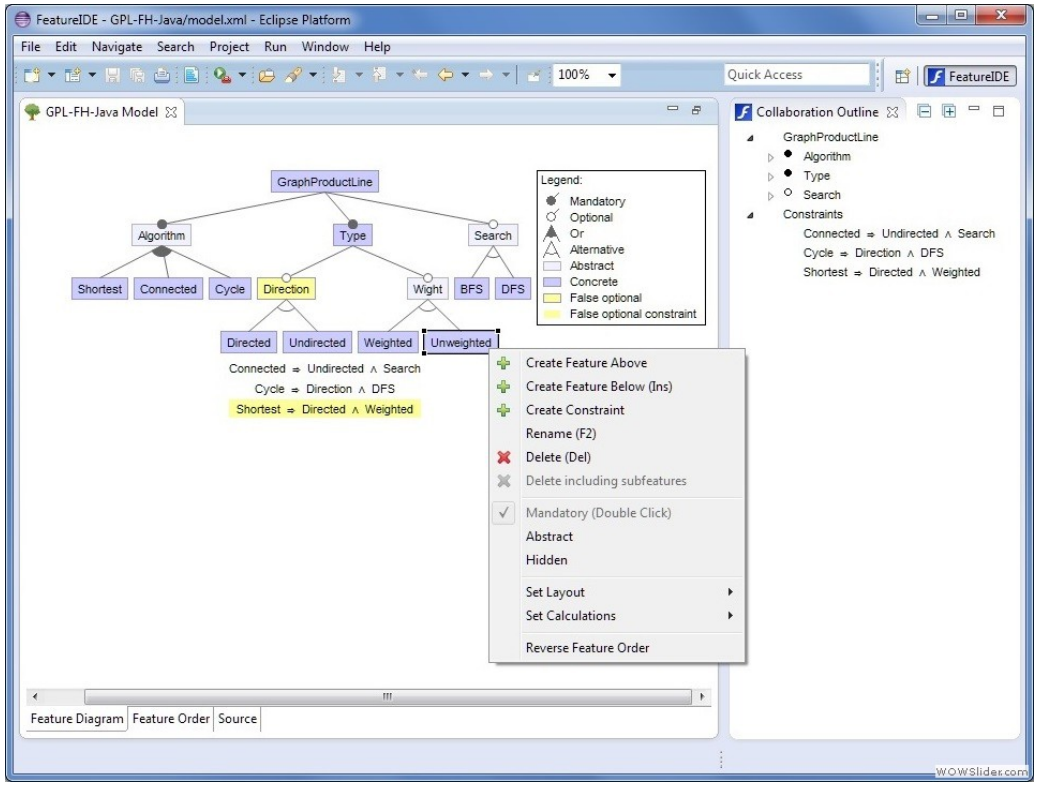
(c) BigLever, Inc.

Feature IDE

- Eclipse plug-in for Feature-Oriented Software Development
- supports all phases of FOSD
  - domain analysis, domain implementation, requirements analysis, and software generation
- implementation techniques integrated
  - feature-oriented programming (FOP), aspect-oriented programming (AOP), delta-oriented programming (DOP), and preprocessors
- provides tool support for AHEAD, FeatureC++, FeatureHouse, AspectJ, DeltaJ, Munge, and Antenna

Tool Demo of the Feature IDE

- You will get a demo during the invited talk by Dipl.-Ing. Daniela Rabiser



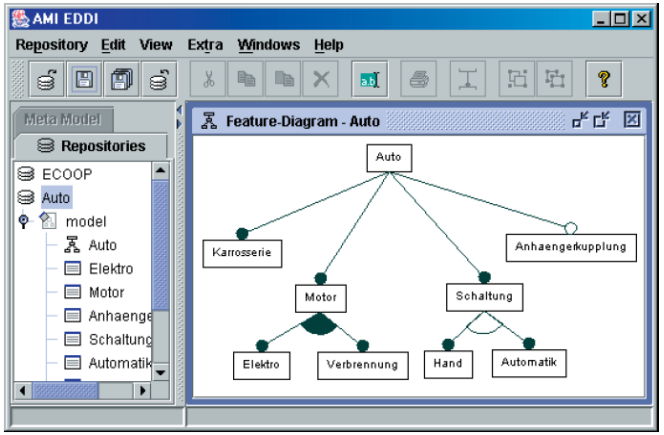
Other SPL Tools

(sorted alphabetically; only a selection, there are more!)

- **AmiEddi/Captain Feature**
- **COVAMOF-VS**
- **Feature Modeling Plug-in (FMP)**
- **FeatureIDE**
- **V-Manage**
- **VARMOD**

AmiEddi/Captain Feature

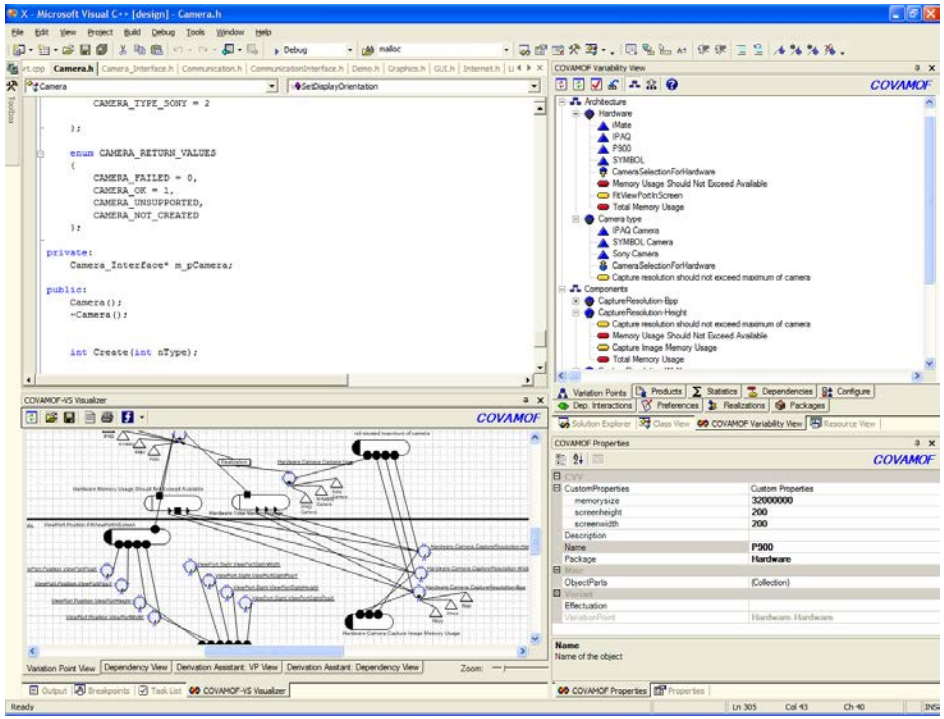
- Developed at Fachhochschule Kaiserslautern
- Implemented in Java/Swing
- Supporting the feature modeling notation by Czarnecki and Eisenecker
- Successor of AmiEddi and implements a cardinality-based notation



- https://sourceforge.net/projects/captainfeature/
- *Last Update: 2013-03-22*

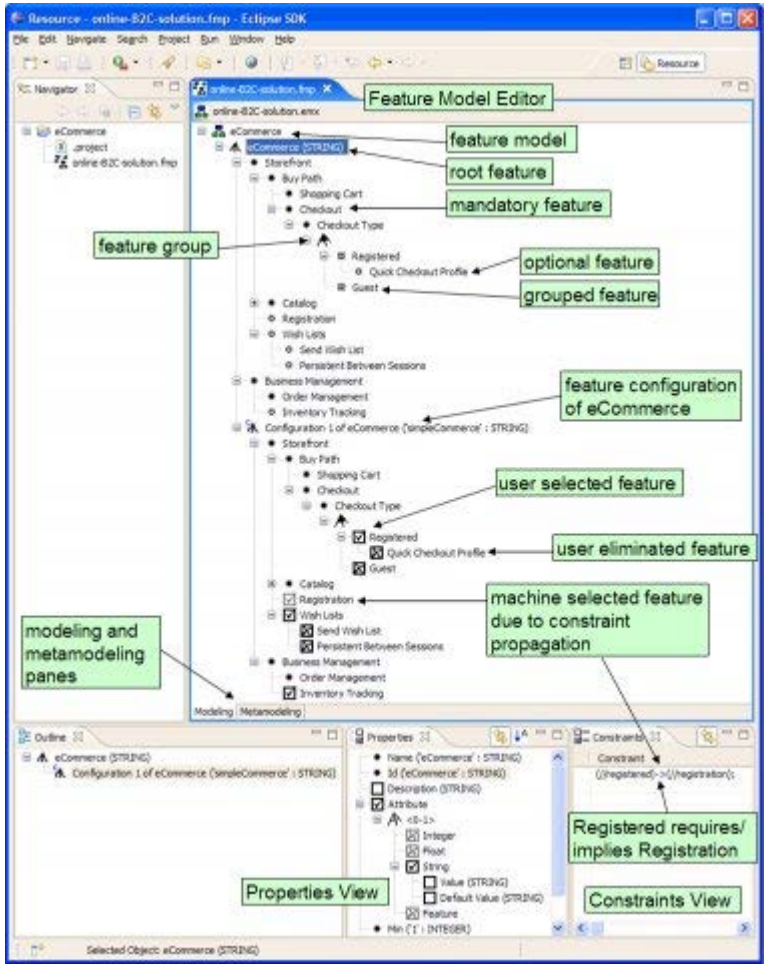
COVAMOF-VS

- Developed at University of Groningen
- Implemented in Microsoft Visual Studio .NET 2003/.NET Framework 1.1/C#
- Supporting the COVAMOF Methodology/Variability Modeling framework
- Variation points and dependencies are modeled over different abstraction levels
- e.g., XML-based feature models, parameter settings, C++/C# source files
- No longer available
- Part of a book: http://dx.doi.org/10.1007/978-3-642-36583-6



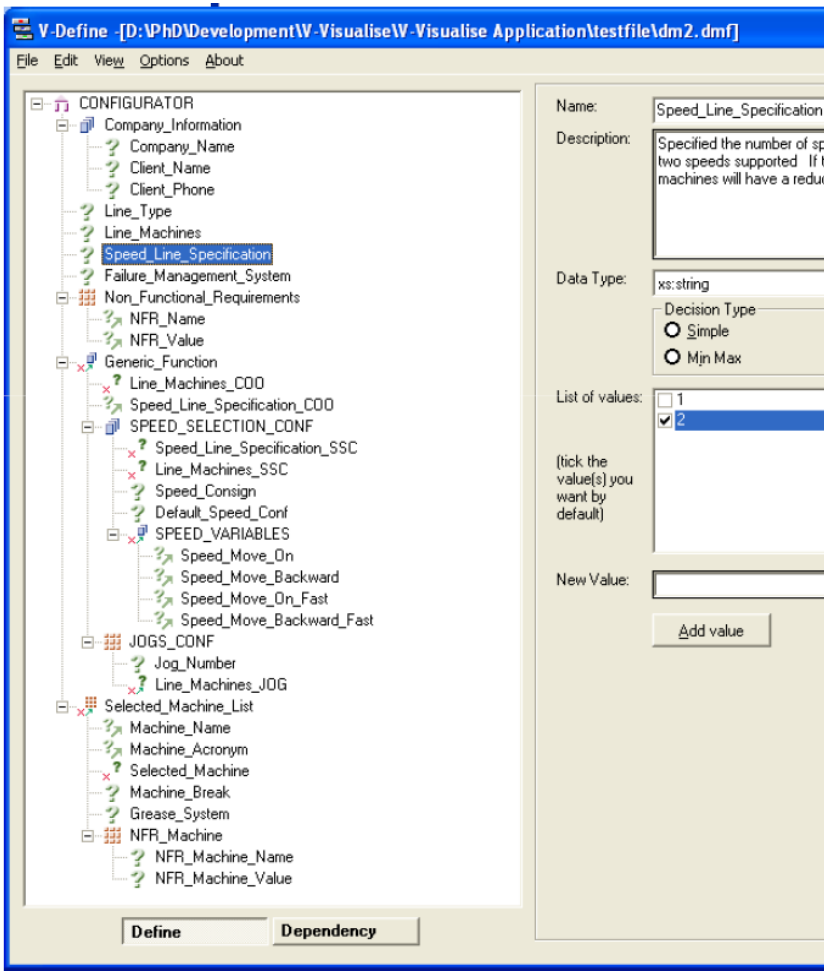
Feature Modeling Plug-in (FMP)

- Developed at University of Waterloo
- Implemented in Java/Eclipse/SWT
- Supports cardinality-based feature modeling as proposed by Czarnecki et al. and configuration based on feature diagrams
- https://sourceforge.net/projects/fmp
- *Last Update: 2013-04-03*



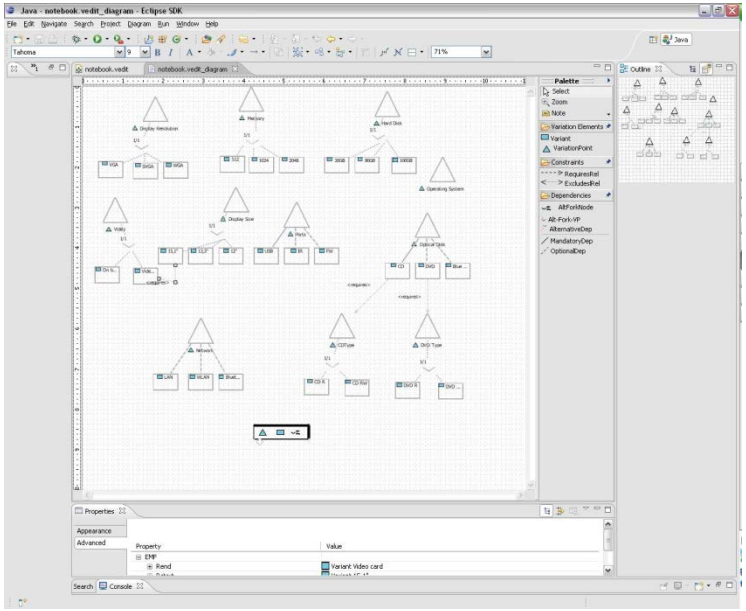
V-Manage Toolset

- Developed at the European Software Insitute (ESI)
- Implemented in Visual Basic(.NET)
- Decision-oriented modeling approach
- V-Define to create decision models
- V-Implement to define variation points in components/component descriptions
- V-Resolve to take decisions
- *No longer active/available*



VARMOD

- Developed at University of Duisburg Essen (Software Systems Engineering Institute)
- Implemented in Java/Eclipse
- Supports Orthogonal Variability Modeling as proposed by Pohl et al.
- Replaced by https://sse.uni-due.de/remidemmi in 2014
- *Last update: 2014-06-18*



Further tools

- Also check:
  - http://www.splot-research.org/ -- live demo now
  - http://fosd.de/

Conclusion

- SPL tool support is an absolute necessity
- Several SPL tools already exist
  - Few commercial and many research tools
  - All have their benefits and drawbacks
- Most tools are not stable/adaptable/interoparable/ scalable enough to be usable in industry/practice
- Most tools focus on product line modeling and do not provide (or provide only little) support for the product derivation/application engineering process
- Empirical data on the usefulness of existing tools is rare
  - You have to try out every tool to find out if it is really useful for your purpose/domain

Next Week (4.5.)

- No Lecture (Landespatron)
- Next lecture: 11.5. PL Testing (RL)

Used/Useful References

- [Bruckhaus et al. 1996]

  T. Bruckhaus, N. H. Madhavji , I. Janssen, and J. Henshaw, "The Impact of Tools on Software Productivity," *IEEE Software*, vol. 13, pp. 29-38, 1996.

- [Nielsen 1994]

  J. Nielsen, *Usability Engineering*: Morgan Kaufmann, 1994.

- [Dhungana et al. 2011]

  D. Dhungana, P. Grünbacher, and R. Rabiser, "The DOPLER Meta-Tool for Decision-Oriented Variability Modeling: A Multiple Case Study," Automated Software Engineering, vol. 18(1), pp. 77-114, 2011.

- [Rabiser et al. 2007]

  Rabiser, R., Grünbacher, P., Dhungana, D.: Supporting Product Derivation by Adapting and Augmenting Variability Models, Proc. of the 11th International Software Product Line Conference (SPLC 2007), Kyoto, Japan, 2007.

- [Myllärniemi 2007]

  V. Myllärniemi, M. Raatikainen, and T. Männistö, "Kumbang Tools," 11th International Software Product Line Conference (SPLC 2007), Tool Demonstration, Kyoto, Japan, 2007.

- [pure systems GmbH]

  pure systems GmbH, "Variant Management with pure::variants, Technical Whitepaper," http://www.pure-systems.com/fileadmin/downloads/pv-whitepaper-en-04.pdf, 2008.

  http://www.pure-systems.com/fileadmin/downloads/pure-variants/doc/pv-user-manual.pdf

- [Krueger 2007]

  C. Krueger, "BigLever software gears and the 3-tiered SPL methodology, "Proc. of the Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA'07), Montreal, Quebec, Canada, ACM, 2007, pp. 844-845.