JOHANNES KEPLER
UNIVERSITY LINZ

Requirements and Software Product Lines

Outline

- Introduction and Motivation
- Variability Modeling
- Requirements Engineering and Product Configuration

Product Line Analogy

Product Lines

Building b t

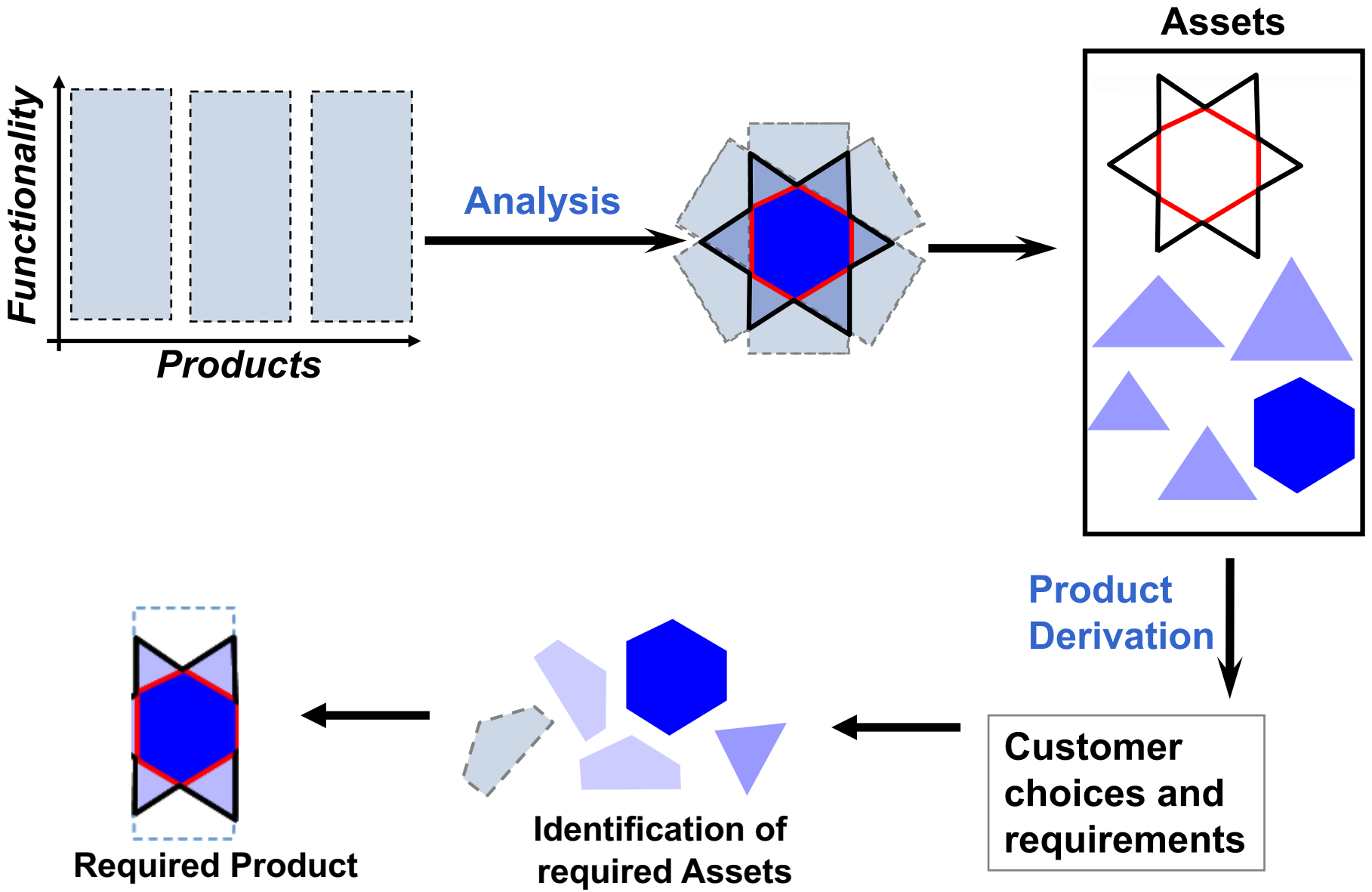
SPL Definition

A software product line (SPL) is a **set of software-intensive systems** that **share** a **common**, managed set of **features** satisfying the specific needs of a particular **market segment** or mission and that are **developed from** a **common set of core assets** in a prescribed way.

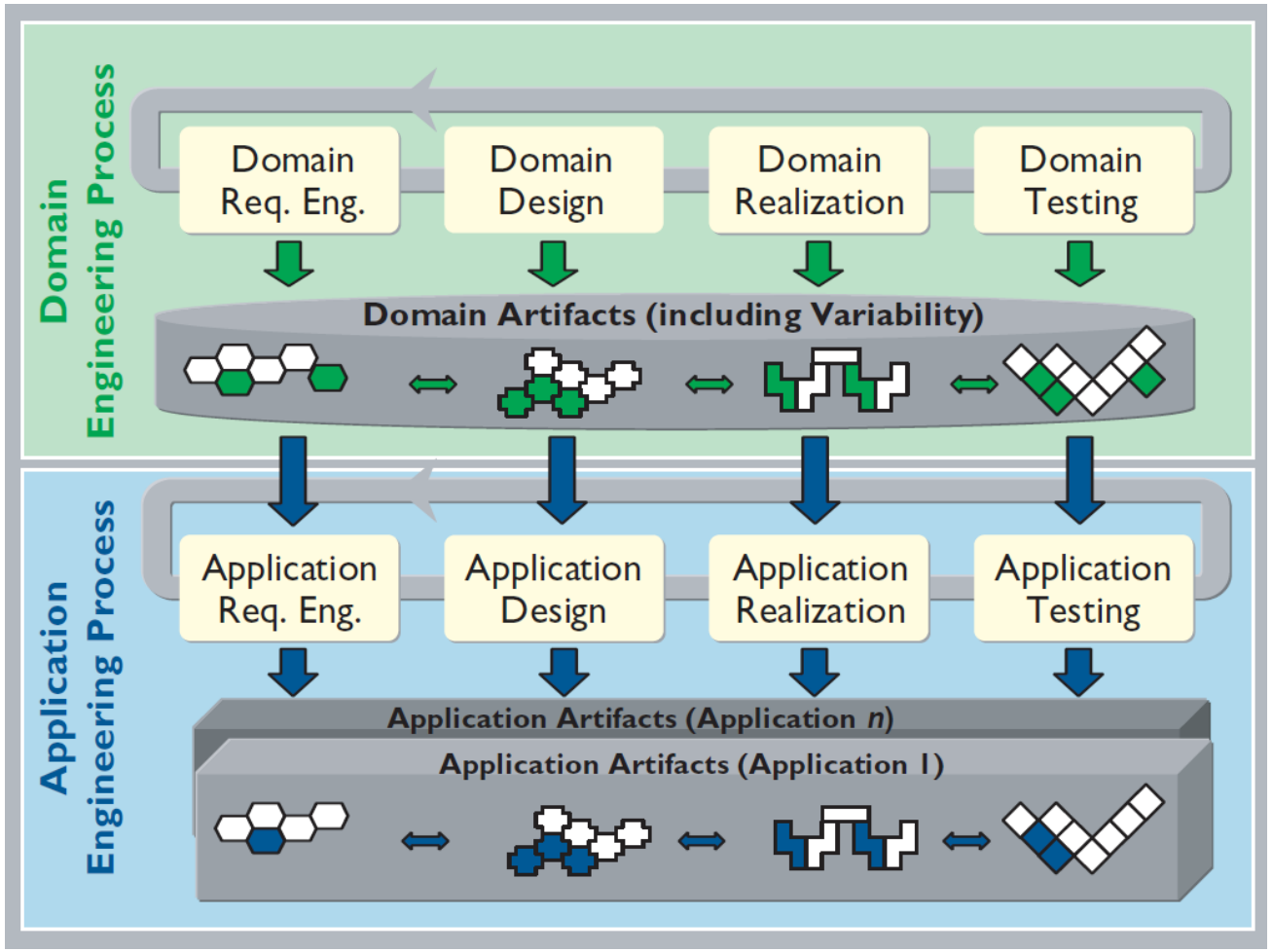
Clements, Northrop: 'Software Product Lines – Practices and Patterns', Addison-Wesley, 2002

How does Requirements Engineering change for software product lines?

Product Line Engineering Process



The Two PLE Lifecycles



Pohl, K., Böckle, G., and van der Linden, F. Software Product Line Engineering: Foundations, Principles, and Techniques. Springer, Berlin, Heidelberg, New York, 2005.

Requirements in the 2 Life Cycles

Domain Requirements Engineering

- Product line scoping
- Commonality and variability modeling

Application Requirements Engineering

- Product configuration based on requirements
- Capturing of customer-specific requirements not covered by the product line

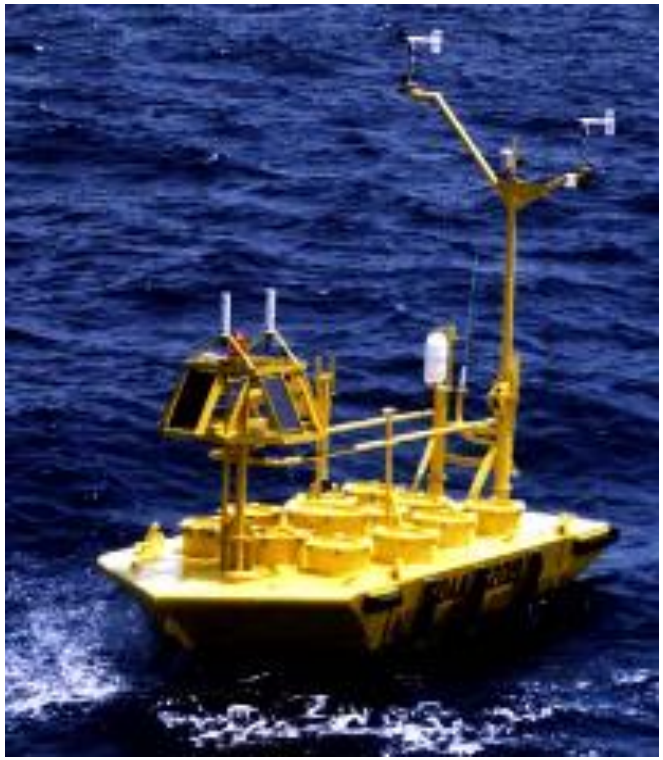
→ 2 Examples from the CDL on ASE

External vs. internal variability

- **External** Variability (problem space)
  - Visible to the customer:
    - manual vs. automatic transmission
    - your cell phone may or may not have a camera and you may have different resolution options
    - …
- **Internal** Variability (solution space)
  - Hidden from customer:
    - battery technology in hybrid electric car
    - communication protocol
    - …

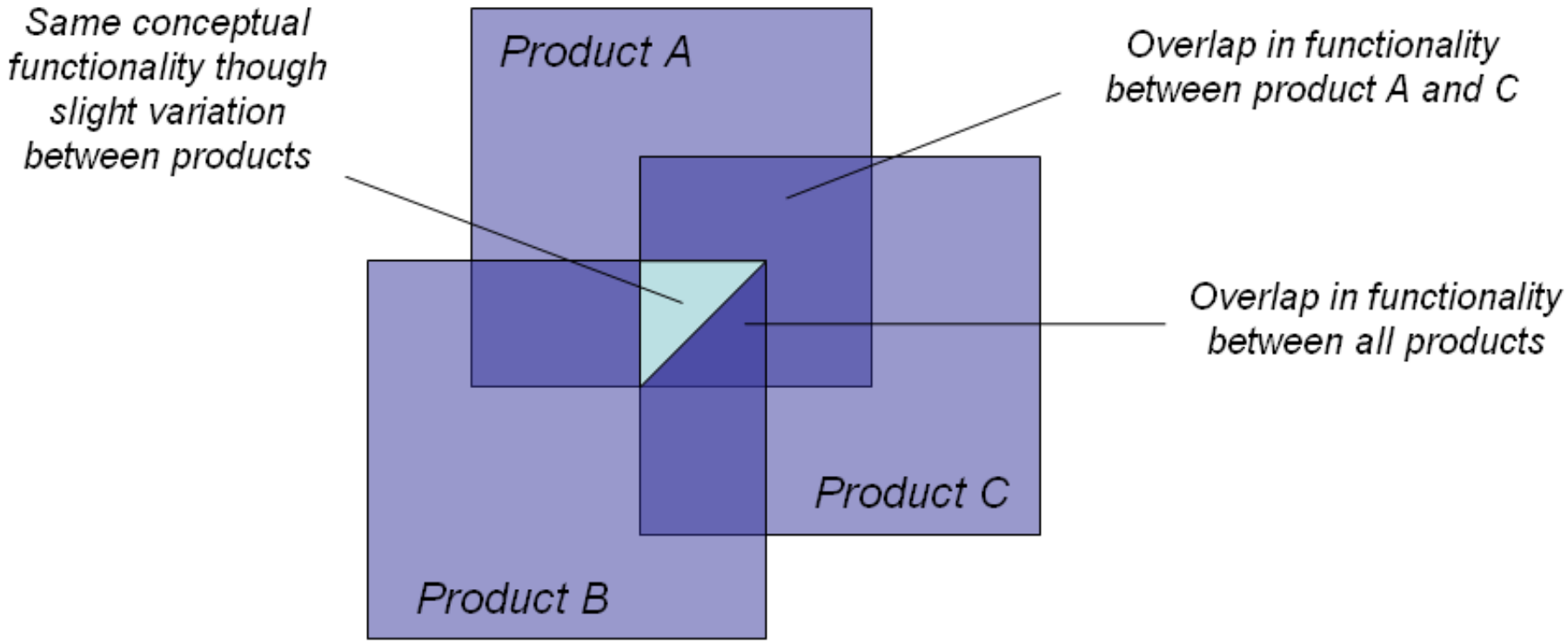
External variability (requirements)

- Commonality Analysis:
  - Commonalities
    - "All FWS shall report the current temperature."
  - Variability
    - "Some FWS may report the wind direction."
  - Constraints
    - "Any FWS that reports the wind direction must also report the wind speed."



Source: National Data Buoy Center

Commonality and Variability Analysis

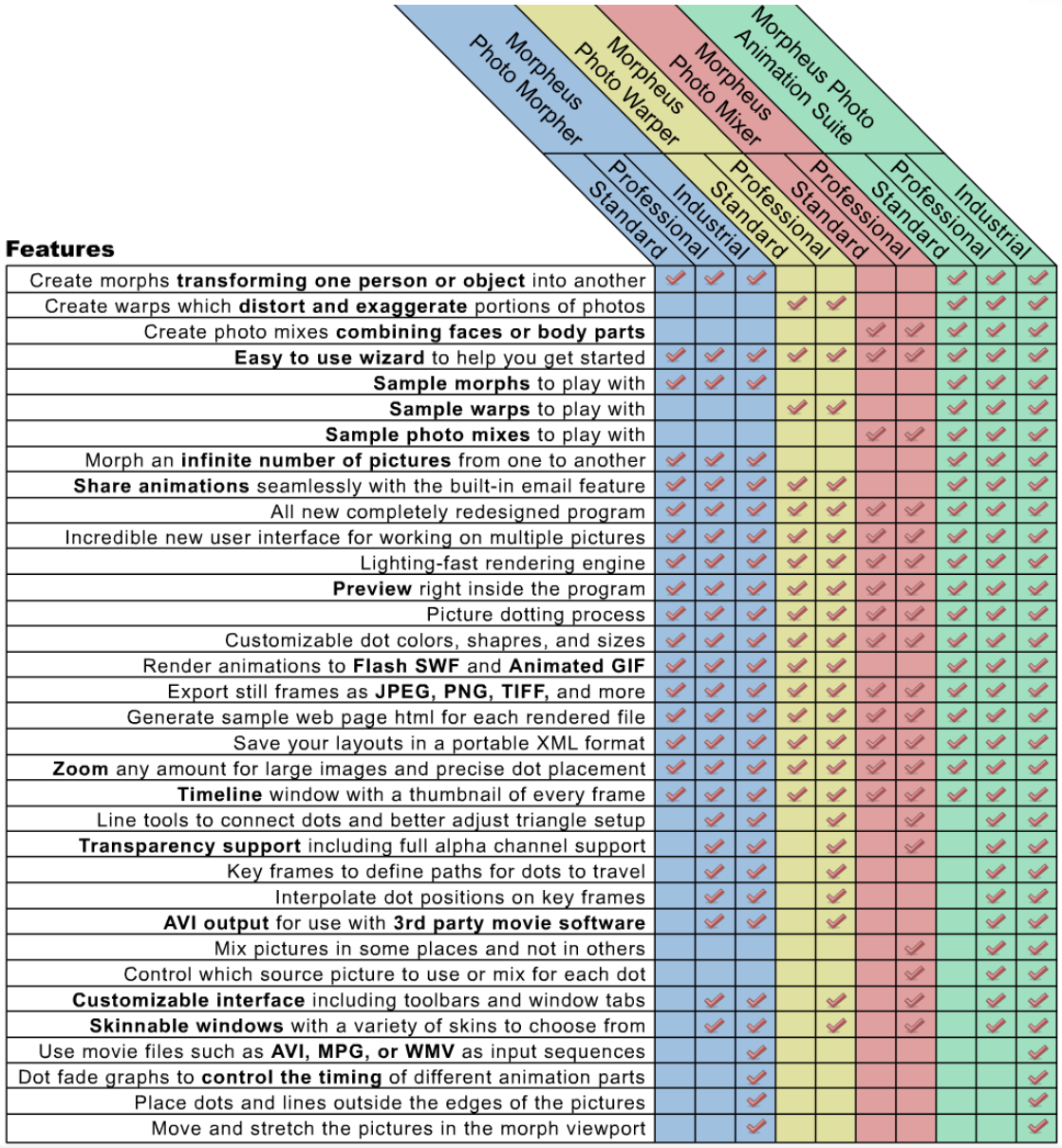


Classroom Exercise 1 – Developing a Product Map

- Create a product map for the Web-based Assessment Tool product line. A product map is a table that maps features to products
- Think of possible features/products based on the given information and arrange them in a matrix:

Feature/ Product	Product 1	Product 2	...	Product N
Feature 1	x	x	x	x
Feature 2		x	x	x
...	x	x	x	
Feature N	x		x	x

Scoping a Product Line: Developing a Product Map

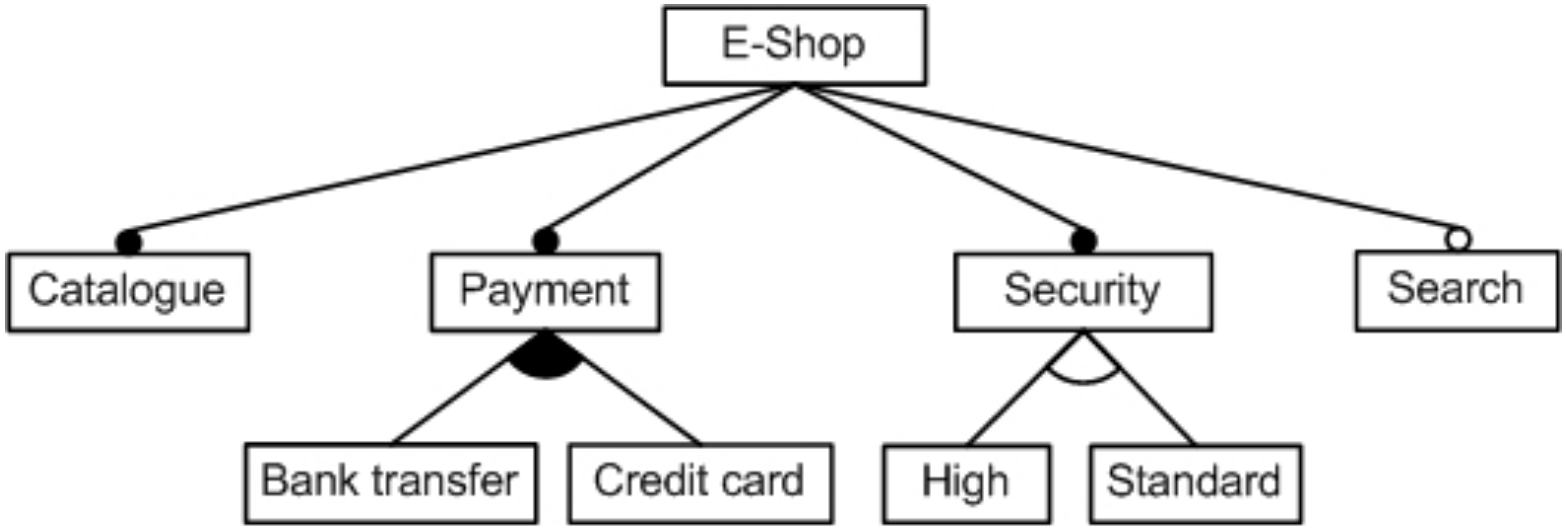


Features	Morpheus Photo Morpher Standard	Morpheus Photo Morpher Professional	Morpheus Photo Morpher Industrial	Morpheus Photo Warper Standard	Morpheus Photo Warper Professional	Morpheus Photo Mixer Standard	Morpheus Photo Mixer Professional	Morpheus Photo Animation Suite Standard	Morpheus Photo Animation Suite Professional	Morpheus Photo Animation Suite Industrial
Create morphs **transforming one person or object** into another	✓	✓	✓					✓	✓	✓
Create warps which **distort and exaggerate** portions of photos				✓	✓			✓	✓	✓
Create photo mixes **combining faces or body parts**						✓	✓	✓	✓	✓
Easy to use wizard to help you get started	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Sample morphs to play with	✓	✓	✓					✓	✓	✓
Sample warps to play with				✓	✓			✓	✓	✓
Sample photo mixes to play with						✓	✓	✓	✓	✓
Morph an **infinite number of pictures** from one to another	✓	✓	✓					✓	✓	✓
Share animations seamlessly with the built-in email feature	✓	✓	✓	✓	✓			✓	✓	✓
All new completely redesigned program	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Incredible new user interface for working on multiple pictures	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Lighting-fast rendering engine	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Preview right inside the program	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Picture dotting process	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Customizable dot colors, shapres, and sizes	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Render animations to **Flash SWF** and **Animated GIF**	✓	✓	✓	✓	✓			✓	✓	✓
Export still frames as **JPEG, PNG, TIFF,** and more	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Generate sample web page html for each rendered file	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Save your layouts in a portable XML format	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Zoom any amount for large images and precise dot placement	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Timeline window with a thumbnail of every frame	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Line tools to connect dots and better adjust triangle setup		✓	✓		✓		✓		✓	✓
Transparency support including full alpha channel support		✓	✓		✓		✓		✓	✓
Key frames to define paths for dots to travel		✓	✓		✓				✓	✓
Interpolate dot positions on key frames		✓	✓		✓				✓	✓
AVI output for use with **3rd party movie software**		✓	✓		✓				✓	✓
Mix pictures in some places and not in others							✓		✓	✓
Control which source picture to use or mix for each dot							✓		✓	✓
Customizable interface including toolbars and window tabs		✓	✓		✓		✓		✓	✓
Skinnable windows with a variety of skins to choose from		✓	✓		✓		✓		✓	✓
Use movie files such as **AVI, MPG, or WMV** as input sequences			✓							✓
Dot fade graphs to **control the timing** of different animation parts			✓							✓
Place dots and lines outside the edges of the pictures			✓							✓
Move and stretch the pictures in the morph viewport			✓							✓

Part I: Variability Modeling

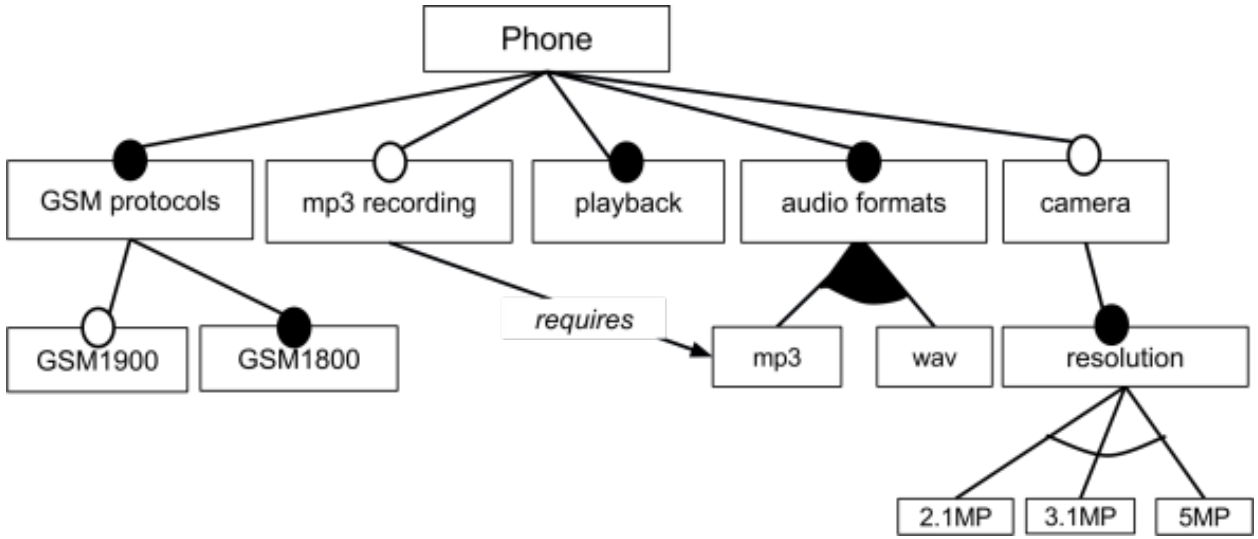
Classroom Exercise 2
Feature Modeling

- Create a feature model based on the features of the Web-based Assessment system product line



Variability Modeling: Feature Modeling vs Decision Modeling

FM



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

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]

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.

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]

Comparing Feature Modeling and Decision Modeling

- Numerous variability modeling approaches exist today
- Most based on feature modeling (FM) or decision modeling (DM)

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

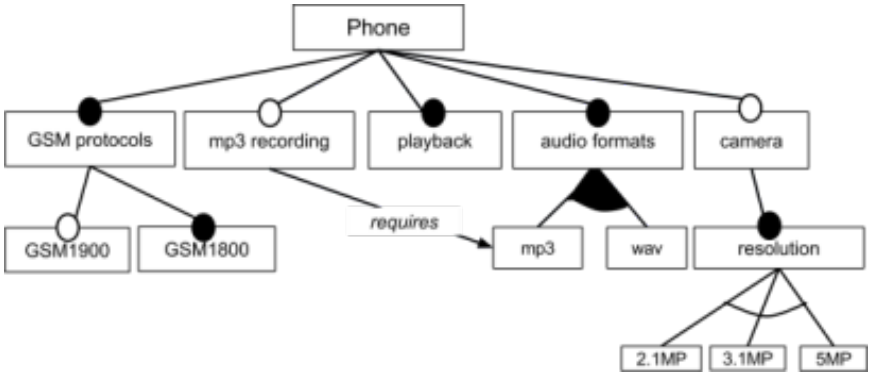
K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid, A. Wąsowski, "Cool Features and Tough Decisions: A Comparison of Variability Modeling Approaches", *In: Proceedings 6th Int'l Workshop on Variability Modelling of Software-Intensive Systems, Leipzig, Germany*, pp. 173-182, 2012.

- Paper points out commonalities and differences

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

***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**.*

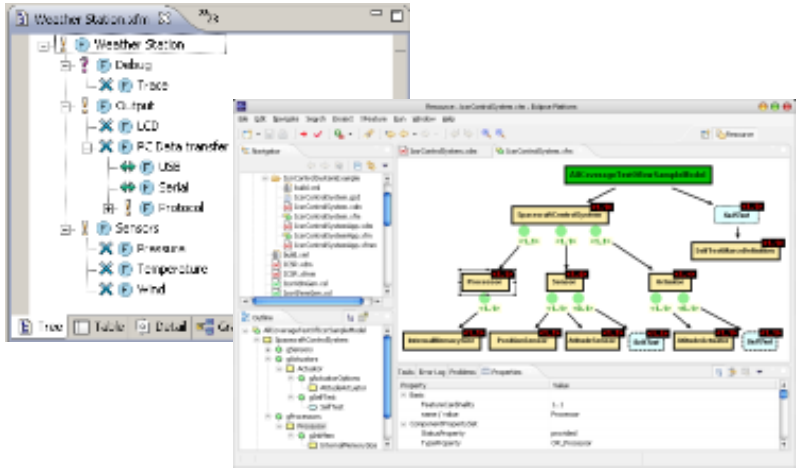
Typically decisions or features (high-level variability abstractions) are related to variation points (locations in artifacts where variability occurs).

Some DM and FM approaches define a separate artifact model.

Tool aspects: modeling as well as derivation support

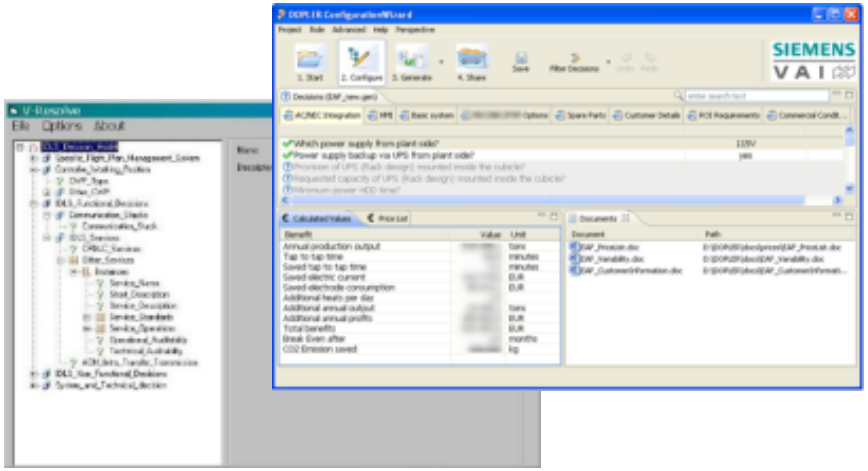
FM

- Configuration UI: usually a tree (unordered)
- Diverse solutions for configuration workflows (secondary concept)

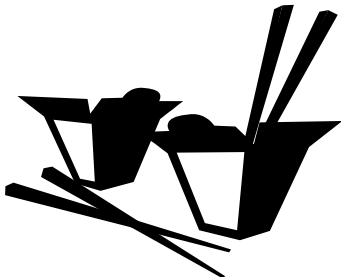


DM

- Configuration UI: typically an (ordered) question list
- Diverse solutions for configuration workflows (essential)



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!

Specific capabilities of approaches are much more important when selecting an approach than classification as DM or FM

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**

Variability Modeling in Practice

- Over the last 20 years VM has been embraced in practice
- In commercial tools like Pure::Variants and Gears
- In specific modeling languages
  - **Linux kernel** project uses the home-grown variability specification language **Kconfig** [Zippel et al. 2011, Sincero and Schröder-Preikschat 2008]
  - **eCos** operating system uses its Component Description Language (**CDL**) [Veer and Dallaway 2010]
  - Similar languages appear in commercial systems, for instance OSEK [Czarnecki et al. 2002]

OSEK/VDX

Variability Modeling in Practice

Dimension	Kconfig	CDL	CVL initial
Applications	Kernel variability; derivation support	eCos variability; derivation support	**var. modeling; derivation support**
Unit of variability	drivers, subsystems, kernel options, build option		**Vspecs: decisions in derivation**
Orthogonality	**orthogonal**		
Data types	**diverse**		
Hierarchy	FM (organization) and DM (visibility)	like in FM	**Vspec tree like in FM**
Dependencies and Constraints	propositional three-valued logics with comparison	propositional Boolean logics with expressions on data	propositional and predicate logic with expressions on data
Mapping to artifacts	to C preprocessor	explicit in var mod.	**mapping model**
Binding time and mode	static or dynamic dec. at compile time	static	depending on application
Modularity	model split into files	loadable packages, reparenting	packages, **configurable units**
Tool aspects	Modeling in textual syntax, conf. UI: tree with controlled visibility		domain of vendors

Part II: Requirements Engineering and Product Configuration

Motivation

- In PLE various stakeholders are involved in creating and adapting documents
  - offers, contracts, technical documents, user manuals, etc.
- Documents often created manually during otherwise automated product derivation
  - “Islands of automation" that are hardly integrated
  - “Traditional” focus: technical assets, e.g., architecture/code

- Goal: *A generic and flexible approach for automating document generation independently of the concrete type of document and regardless of the maturity of the product line*

R. Rabiser, M. Vierhauser, M. Lehofer, P. Grünbacher, T. Männistö, "Configuring and Generating Technical Documents", In: Knowledge-Based Configuration, Elsevier, 2014.

Industrial Example 1:
CC L2 @ Siemens VAI

- Mature SPL for process automation of continuous casting machines in steel plants
- Variability has already been modeled for technical software assets
- Configuration files for concrete solutions can be generated
- Manual adaptation of **technical documentation** for each customer (300+p) is tedious

Industrial Example 2: EAF @ Siemens AG MT

- Variability not yet modeled
- Sales people manually “parse” and “process” **sales documents** (like customer-specific offers, product descriptions, and commercial conditions)
- Erroneous offers may result in actual losses or legal issues
- Thorough review can extend the duration for creating offers to several weeks

Examples of Variability in Documents

SIEMENS

Document Variability	Examples
Placeholders	- customer details (name, company, address, etc.), - total price, - ROI values.
Optional text	- chapters in a user manual or in a bidding document.
Alternative text	- country-specific commercial conditions and policies, - different operating systems, - units (e.g., metric vs. imperial system).
Cross references	- the main index, - figure and table indices, - references to docs like country-specific legal documents.
Simple grammatical variability	- 1 strand vs. 2 or more strands, - multiple drives vs. the drive.
Media objects	- customer's logo, - user interface.
Formatting/Layout	- A4 vs. letter size, - different color schema.

PL Document Management Approach

SIEMENS

(1) Elicit and analyze variability in documents

- Existing documents; workshops with domain experts

(2) *Create or adapt variability models*

- Reuse existing models if possible

(3) *Choose or develop variability mechanism and corresponding generator for domain-specific doc formats*

- Automate creation of documents according to variant selections

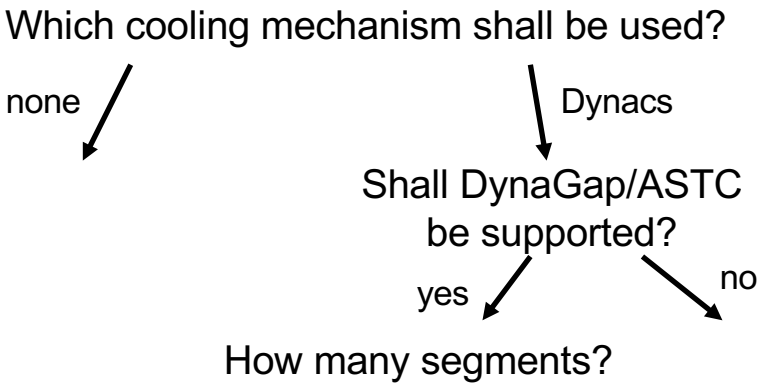
(4) *Augment the documents with variability information*

- Can only be done by domain experts

Background: The DOPLER Approach

SIEMENS

Decisions (represent variability)

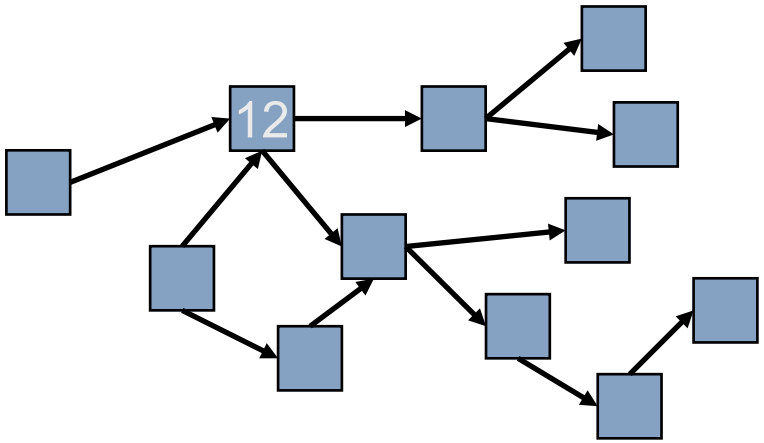


Meta-modeling approach:

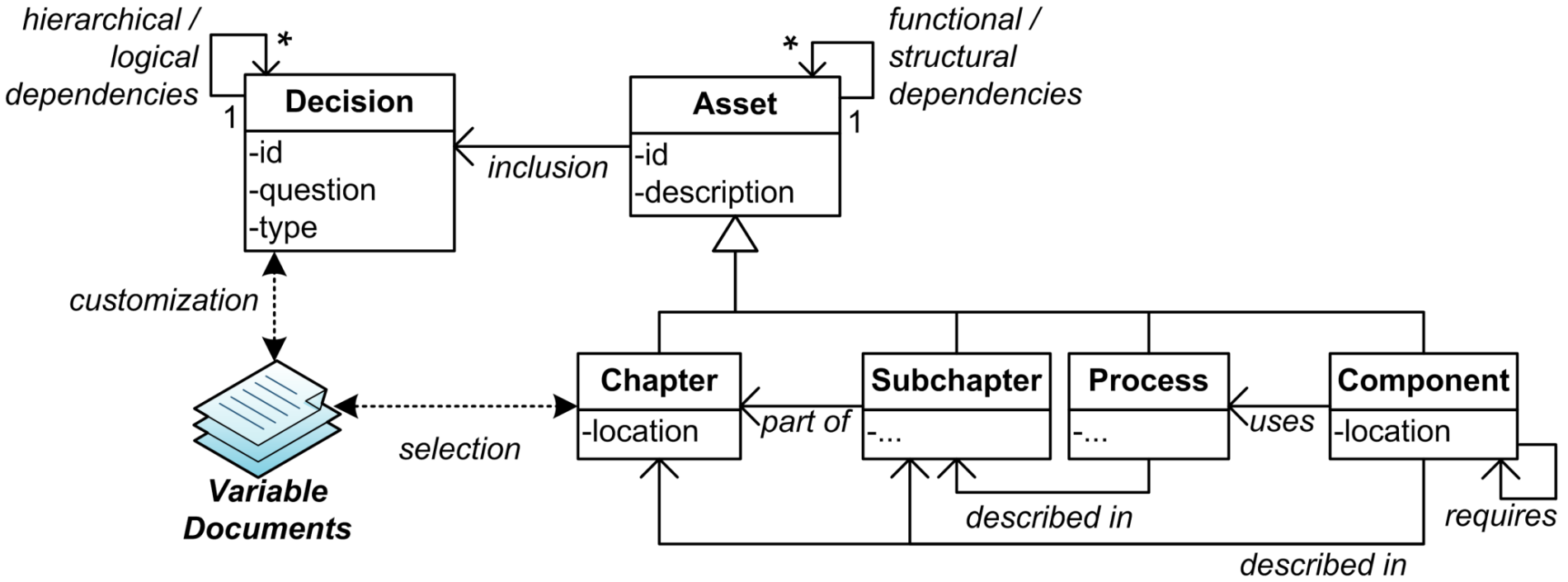
Domain-specific Asset Types, Attributes, and Dependencies can be defined

Supported by the DOPLER Tool Suite

Assets (e.g., Components, Documents, ...)

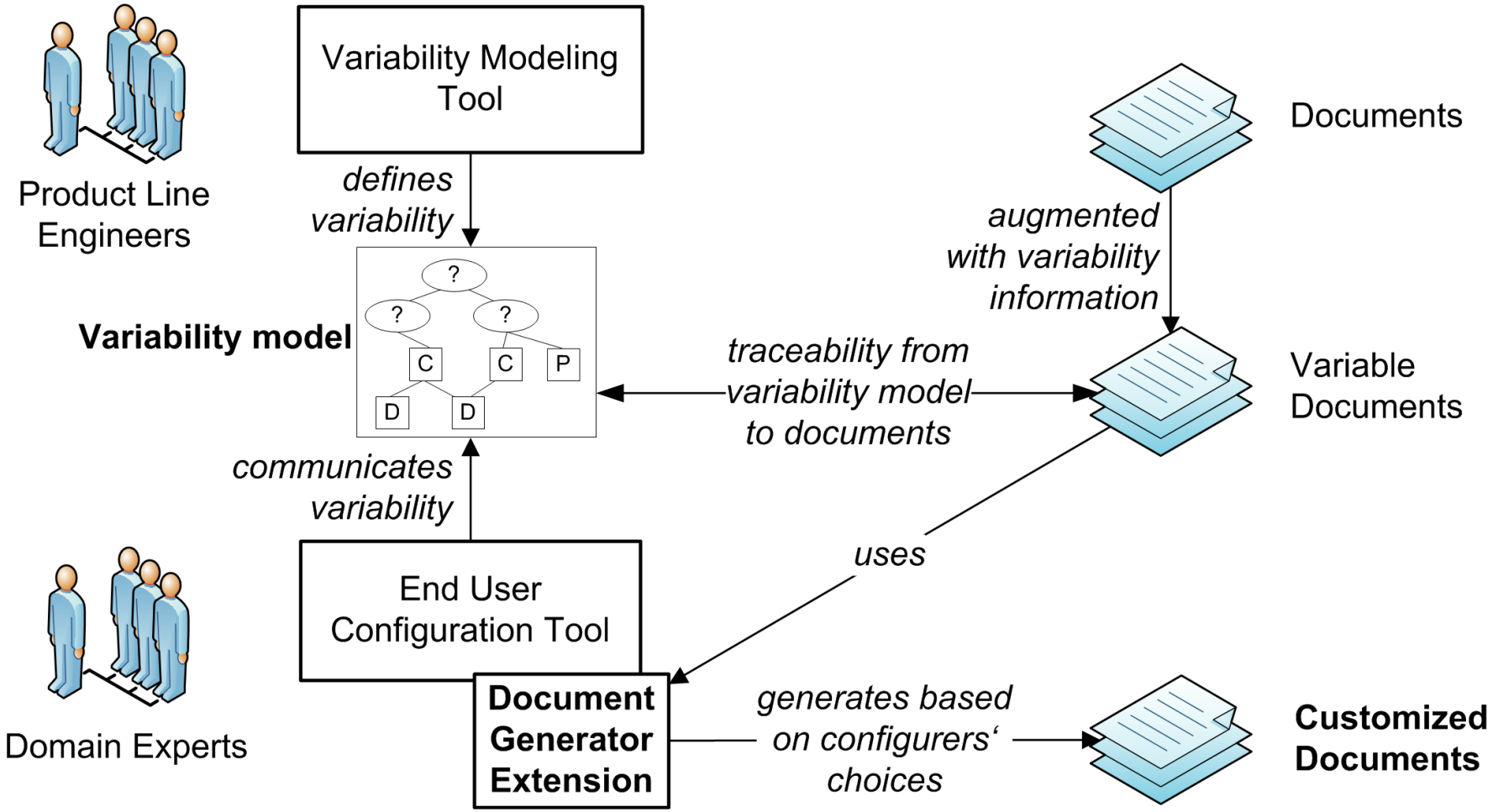


Modeling Documents and their Variability with DOPLER



- Decisions represent document variability as questions that a user is expected to answer during product derivation
- Document fragment assets represent arbitrary parts of documents and are used to model coarse-grained variability

Generating Documents with DOPLER



Implementation with DocBook

Document Variability	Examples of Implementation with DocBook

Placeholders	`<section id="caster">`
	`The caster has <doplerdocplaceholder`
	`doplerdoc="numStrands"/>strands. </section>`

Optional text	`<chapter id="hmi" doplerdoc="hmichapt">` `...</chapter>`

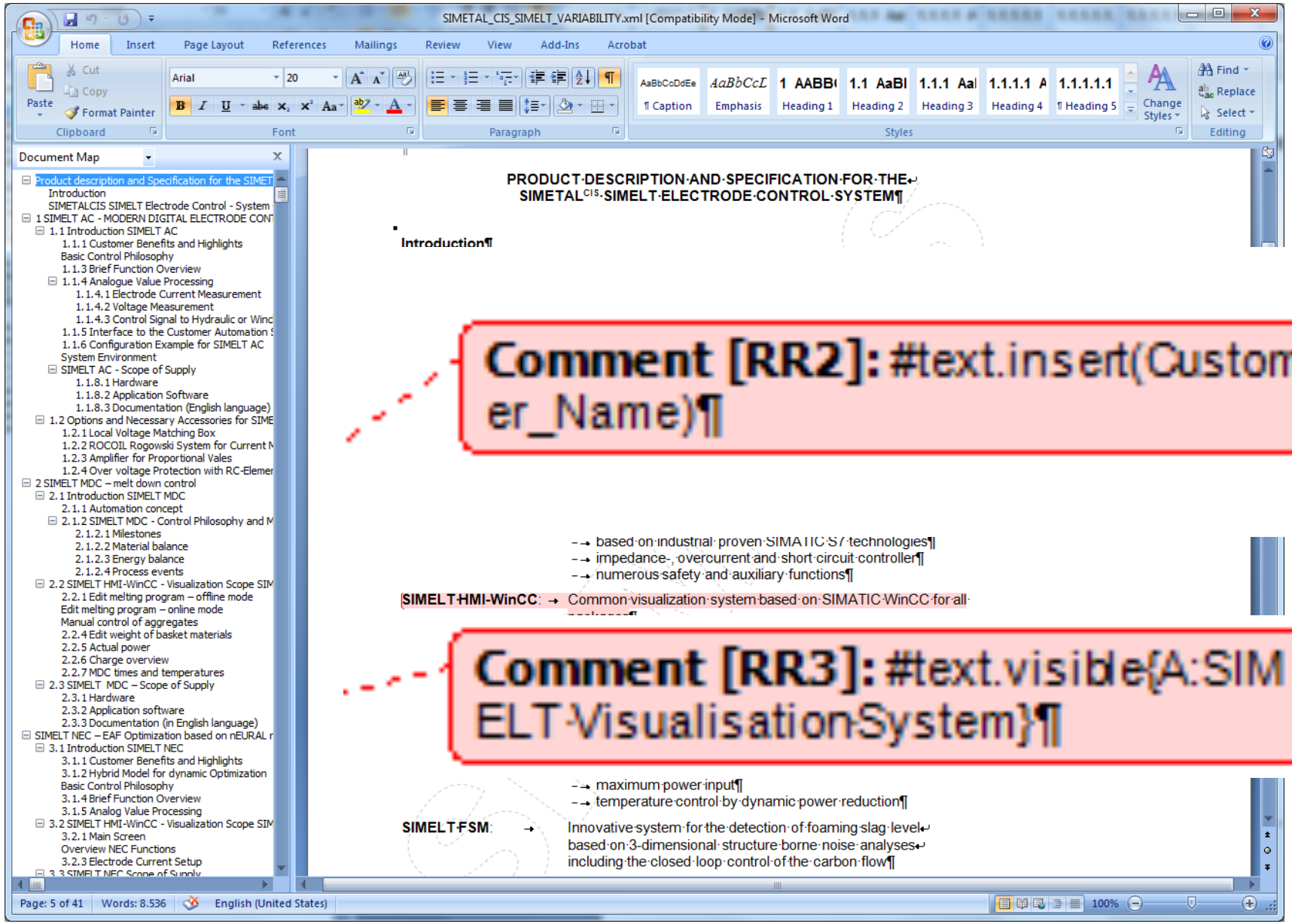
Alternative text	`<section id="dex" doplerdoc="dexchapter">`
	`Data Exchange is supported via`
	`<phrase doplerdoc="asciiphrase">ASCII</phrase>`
	`<phrase doplerdoc="dbphrase">DataBase</phrase>`
	`in your system. ...</section>`

Cross references	`<xref linkend="hmi" doplerdoc="hmichapt"/>`

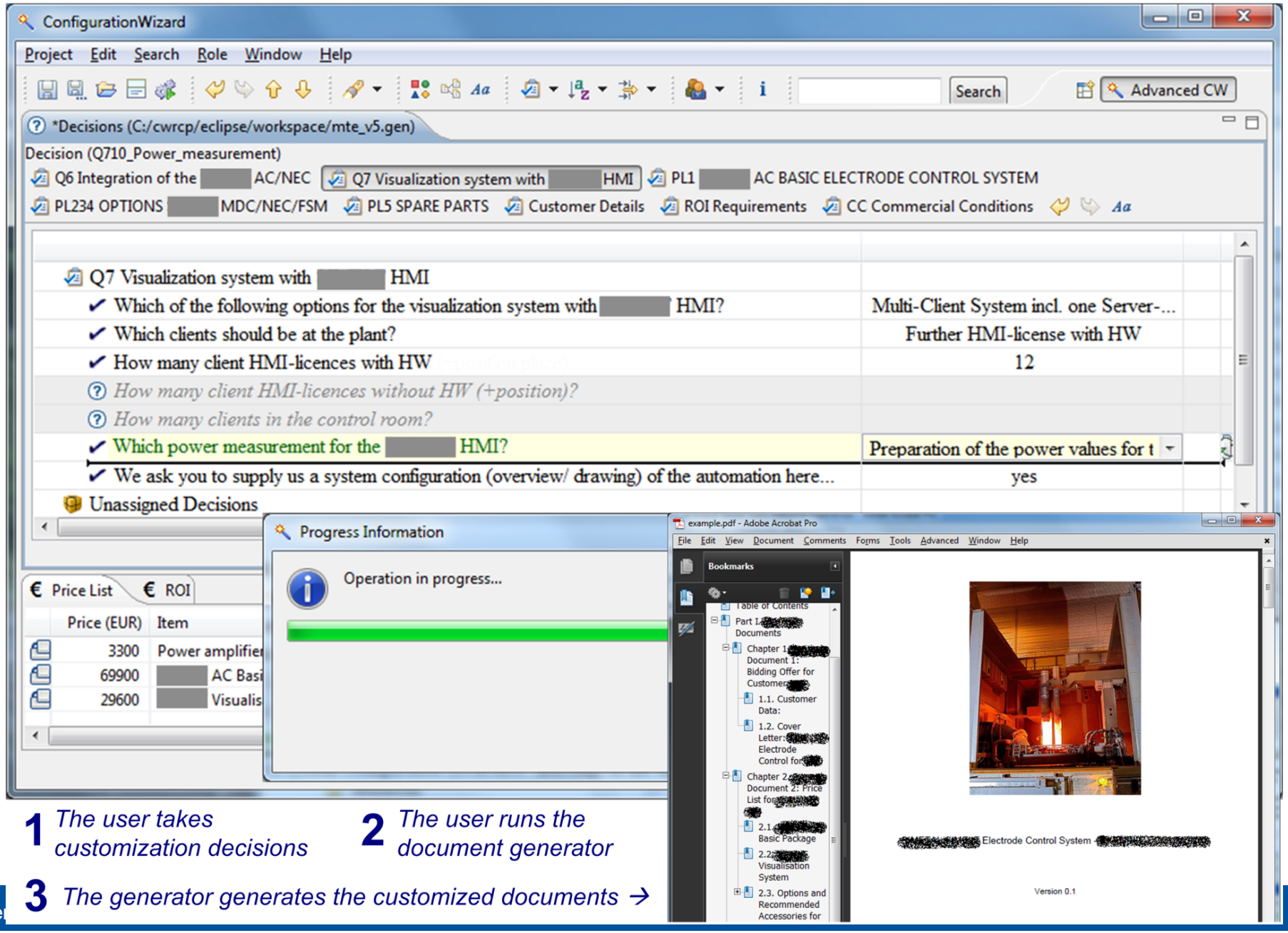
Simple grammatical variability	`The caster has <doplerdocplaceholder`
	`doplerdoc="numStrands"/> strand`
	`<phrase doplerdoc="numStrands#2+">s</phrase>`

Media objects	`<mediaobject doplerdoc="hmichapt">...</mediaobject>`
Formatting/Layout	`XSL transformation and CSS style sheets related to a property file which can be generated based on decisions`

Screenshot of Parameterized MS Word



Tool Screenshot : CW & Generator



1 *The user takes customization decisions*

2 *The user runs the document generator*

3 *The generator generates the customized documents →*

Industrial Example 1:
CC L2 Technical User Documentation

- Variability
  - Parts of the system to be delivered
  - Grammatical changes (e.g., strand vs. strands)
  - Specific documents if customers intend to develop extensions
- Models
  - We could reuse about 70% of software configuration decisions
- Generator
  - Extension for the DOPLER configuration wizard generates MS Word documents

Industrial Example 2: EAF Sales Documents

- Variability
  - Component descriptions, customer details, price lists, total prices, ROI values, etc. depend on the customer and selected features
- Models
  - *Feature* asset with attribute *price* for calculations
  - 101 decisions for EAF
- Generator
  - We first reused/extended the CC L2 doc generator
  - We then implemented a generator for MS Word
    - Word commenting feature to define variation points
    - VB Add-In and Generator

Discussion

SIEMENS

- First case: mature software product line
  - Possible to largely reuse the existing variability models
- Second case: no automated product derivation techniques
  - Document generation provided a convincing showcase for the benefits of variability analysis, modeling, and automation
- Both cases: decision models considered very helpful for describing variability, especially by non-technical people

- Effort mainly depends on number and type of documents
  - Example 1: Less than one work week (docs were already in DocBook, variability was modeled)
  - Example 2: About a work month (variability was not yet modeled)

- Many existing model-driven approaches and tools could also be used for implementing the approach