Software Product Line Engineering

Pablo Sánchez

Dpto. Matemáticas, Estadística y Computación
Universidad de Cantabria, Santander (Spain)
p.sanchez@unican.es

LSI - Working Sessions
February 16th 2010, Santander (Spain)

About me

- Main research lines:

About me

- Main research lines:
  - Aspect-Oriented Software Development.

About me

- Main research lines:
  - Aspect-Oriented Software Development.
  - Model-Driven (Software) Development.

About me

- Main research lines:
  - Aspect-Oriented Software Development.
  - Model-Driven (Software) Development.
  - PhD Thesis: Almadraba - Model-Driven Development of Aspect-Oriented Executable UML Models. (Supervised by Lidia Fuentes)

About me

- Main research lines:
  - Aspect-Oriented Software Development.
  - Model-Driven (Software) Development.
  - PhD Thesis: Almadraba - Model-Driven Development of Aspect-Oriented Executable UML Models. (Supervised by Lidia Fuentes)
  - Software Product Lines.

About me

- Main research lines:
  - Aspect-Oriented Software Development.
  - Model-Driven (Software) Development.
  - PhD Thesis: Almadraba - Model-Driven Development of Aspect-Oriented Executable UML Models. (Supervised by Lidia Fuentes)
  - Software Product Lines.

About me

- Main research lines:
  - Aspect-Oriented Software Development.
  - Model-Driven (Software) Development.
  - PhD Thesis: Almadraba - Model-Driven Development of Aspect-Oriented Executable UML Models. (Supervised by Lidia Fuentes)
  - Software Product Lines.
- Minor research lines: Component-based software development, software architectures, pervasive computing, executable UML, component coordination and adaptation, software traceability and middleware platforms.

About me

- Main research lines:
  - Aspect-Oriented Software Development.
  - Model-Driven (Software) Development.
  - PhD Thesis: Almadraba - Model-Driven Development of Aspect-Oriented Executable UML Models. (Supervised by Lidia Fuentes)
  - Software Product Lines.
- Minor research lines: Component-based software development, software architectures, pervasive computing, executable UML, component coordination and adaptation, software traceability and middleware platforms.

Software Product Line Engineering

Software Product Line Holly Grail

Create an infrastructure for dealing with the variability of similar software systems [7, 11, 8].

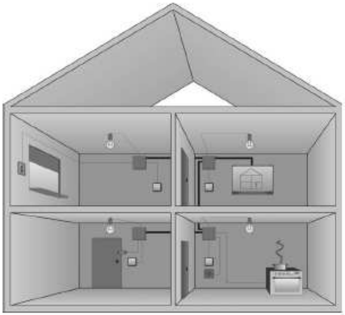
Example: Applications for mobile devices

Software Product Line Engineering

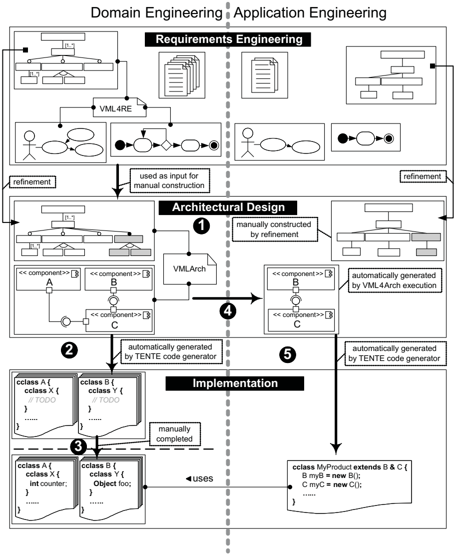
Software Product Line Holly Grail

Create an infrastructure for dealing with the variability of similar software systems [7, 11, 8].

Example: Applications for automated houses



A typical SPL engineering process



Domain Engineering vs Application Engineering

Domain Engineering

Application Engineering

Domain Engineering vs Application Engineering

Domain Engineering **Application Engineering**

Domain Engineering vs Application Engineering

Domain Engineering

Application Engineering

Domain Engineering vs Application Engineering

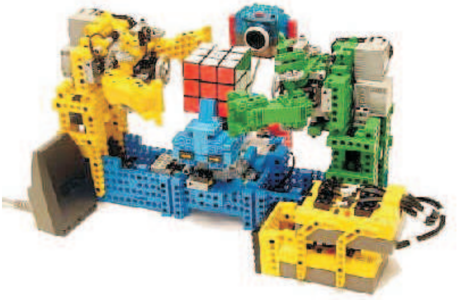
Domain Engineering

Application Engineering

Domain Engineering vs Application Engineering

Domain Engineering

Application Engineering



Software Product Lines Challenges

1. We need languages for analysing and specifying variability in a set of similar software products (e.g. feature models [4, 9]).

Software Product Lines Challenges

1. We need languages for analysing and specifying variability in a set of similar software products (e.g. feature models [4, 9]). Problem Space.

Software Product Lines Challenges

1. We need languages for analysing and specifying variability in a set of similar software products (e.g. feature models [4, 9]). Problem Space.
2. We need mechanism for designing and implementing flexible and reusable software assets.

Software Product Lines Challenges

1. We need languages for analysing and specifying variability in a set of similar software products (e.g. feature models [4, 9]). Problem Space.
2. We need mechanism for designing and implementing flexible and reusable software assets. Main goal: modularize features.

Software Product Lines Challenges

1. We need languages for analysing and specifying variability in a set of similar software products (e.g. feature models [4, 9]). Problem Space.
2. We need mechanism for designing and implementing flexible and reusable software assets. Main goal: modularize features. Solution Space.

Software Product Lines Challenges

1. We need languages for analysing and specifying variability in a set of similar software products (e.g. feature models [4, 9]). Problem Space.
2. We need mechanism for designing and implementing flexible and reusable software assets. Main goal: modularize features. Solution Space.
3. We need languages for specifying how problem and solution space models are related.

Software Product Lines Challenges

1. We need languages for analysing and specifying variability in a set of similar software products (e.g. feature models [4, 9]). Problem Space.
2. We need mechanism for designing and implementing flexible and reusable software assets. Main goal: modularize features. Solution Space.
3. We need languages for specifying how problem and solution space models are related. Mapping between problem and solution space models.

Software Product Lines Challenges

1. We need languages for analysing and specifying variability in a set of similar software products (e.g. feature models [4, 9]). Problem Space.
2. We need mechanism for designing and implementing flexible and reusable software assets. Main goal: modularize features. Solution Space.
3. We need languages for specifying how problem and solution space models are related. Mapping between problem and solution space models.
4. Construction of specific products from domain engineering software assets should be as automatic as possible.

Running Example: Lock Control Framework

As part of a Smart Home SPL, a door lock control framework must be designed. This lock control is placed on doors of rooms whose access must be controlled. Several options are available to end users acquiring a specific Smart Home software installation:

- Different authentication mechanisms can be used: identification cards, fingerprint scanners or a simple numeric keypad.

Running Example: Lock Control Framework

As part of a Smart Home SPL, a door lock control framework must be designed. This lock control is placed on doors of rooms whose access must be controlled. Several options are available to end users acquiring a specific Smart Home software installation:

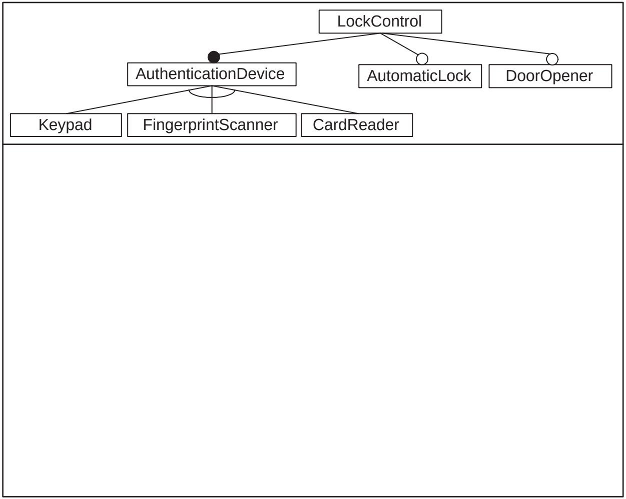
- Different authentication mechanisms can be used: identification cards, fingerprint scanners or a simple numeric keypad.
- Doors are opened manually and users have a time period to authenticate before triggering the alarms. Optionally, it is possible to select a computer-controlled door lock control, which will be released upon successful authentication.

Running Example: Lock Control Framework

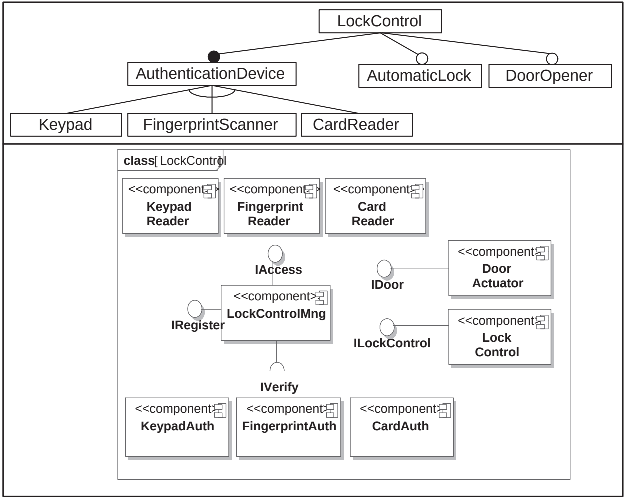
As part of a Smart Home SPL, a door lock control framework must be designed. This lock control is placed on doors of rooms whose access must be controlled. Several options are available to end users acquiring a specific Smart Home software installation:

- Different authentication mechanisms can be used: identification cards, fingerprint scanners or a simple numeric keypad.
- Doors are opened manually and users have a time period to authenticate before triggering the alarms. Optionally, it is possible to select a computer-controlled door lock control, which will be released upon successful authentication.
- Optionally, sliding doors that open automatically can also be used.

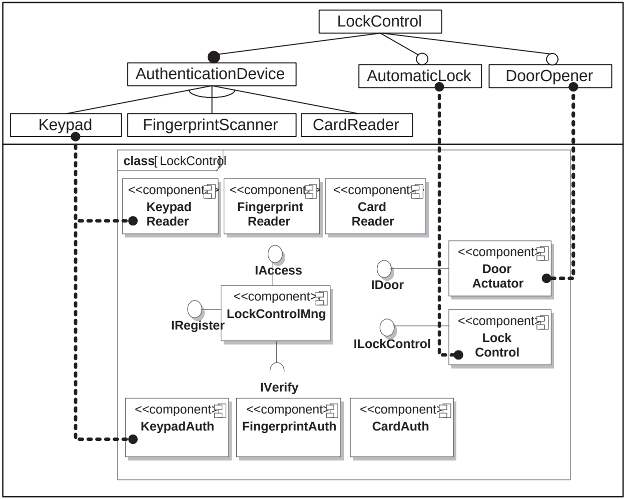
Variability Specification



Variability Realisation/Design



Linking between specification and realisation



Contributions to SPLE where I have participated

1. Hydra, a tool for creating cardinality-based feature models.

Contributions to SPLE where I have participated

1. Hydra, a tool for creating cardinality-based feature models.
2. VML (Variability Management Language).

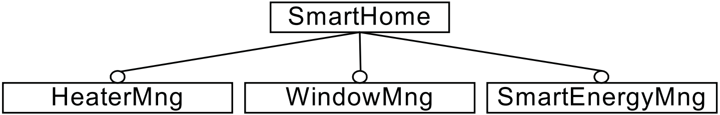
Contributions to SPLE where I have participated

1. Hydra, a tool for creating cardinality-based feature models.
2. VML (Variability Management Language).
3. TENTE, a model-driven feature-oriented process for SPL engineering.

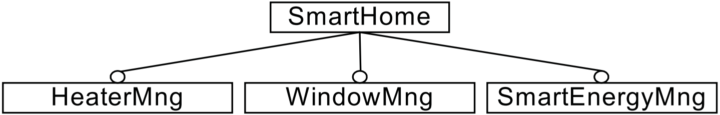
Contributions to SPLE where I have participated

1. Hydra, a tool for creating cardinality-based feature models.
2. VML (Variability Management Language).
3. TENTE, a model-driven feature-oriented process for SPL engineering.

Constraints on feature models

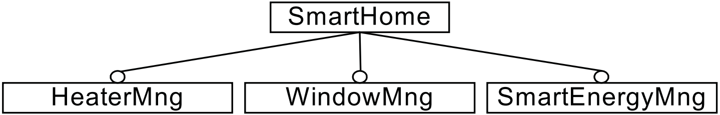


Constraints on feature models



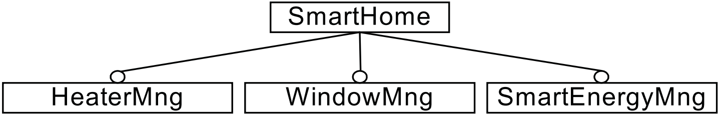
- SmartEnergyMng requires WindowMng and HeaterMng.

Constraints on feature models



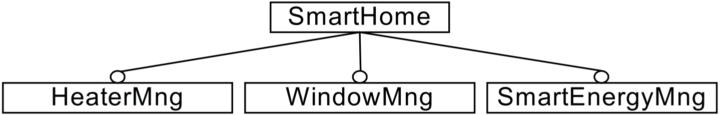
- SmartEnergyMng requires WindowMng and HeaterMng.
- $SmartEnergyMng \rightarrow (WindowMng \wedge HeaterMng)$

Constraints on feature models



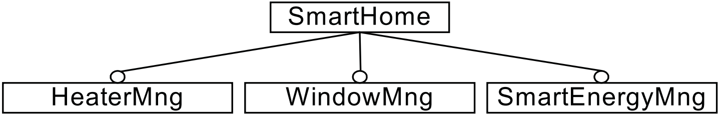
- SmartEnergyMng requires WindowMng and HeaterMng.
- $SmartEnergyMng \rightarrow (WindowMng \wedge HeaterMng)$
- Using BDD, SAT or CSP, we can analize several properties of a feature model [3, 1]:

Constraints on feature models



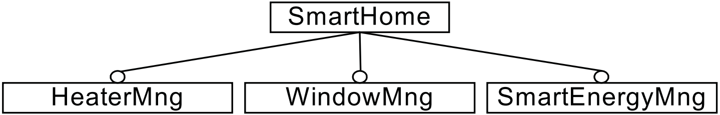
- SmartEnergyMng requires WindowMng and HeaterMng.
- $SmartEnergyMng \rightarrow (WindowMng \wedge HeaterMng)$
- Using BDD, SAT or CSP, we can analize several properties of a feature model [3, 1]:
  1. Autocomplete.

Constraints on feature models



- SmartEnergyMng requires WindowMng and HeaterMng.
- $SmartEnergyMng \rightarrow (WindowMng \wedge HeaterMng)$
- Using BDD, SAT or CSP, we can analize several properties of a feature model [3, 1]:
  1. Autocomplete.
  2. Dead features.

Constraints on feature models



- SmartEnergyMng requires WindowMng and HeaterMng.
- $SmartEnergyMng \rightarrow (WindowMng \wedge HeaterMng)$
- Using BDD, SAT or CSP, we can analize several properties of a feature model [3, 1]:
  1. Autocomplete.
  2. Dead features.
  3. Number of remaining configurations.

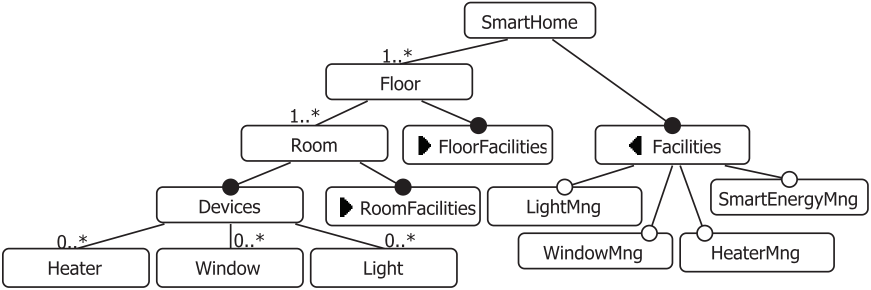
Shortcomings of traditional feature models

Shortcomings of traditional feature models

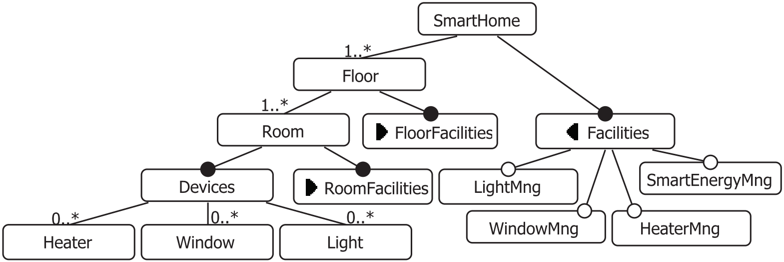
- I want to construct software for automated houses with one or more floors, and one or more rooms per floor [4].

Shortcomings of traditional feature models

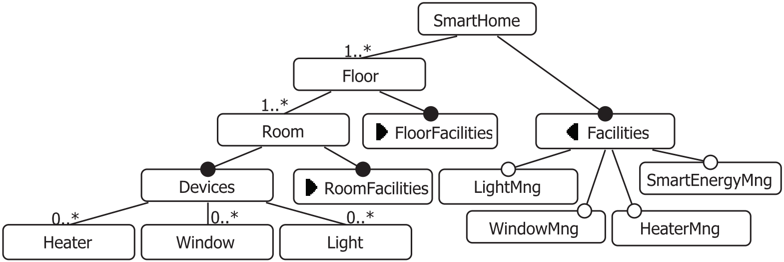
- I want to construct software for automated houses with one or more floors, and one or more rooms per floor [4].



Challenges of constraints involving clonable features

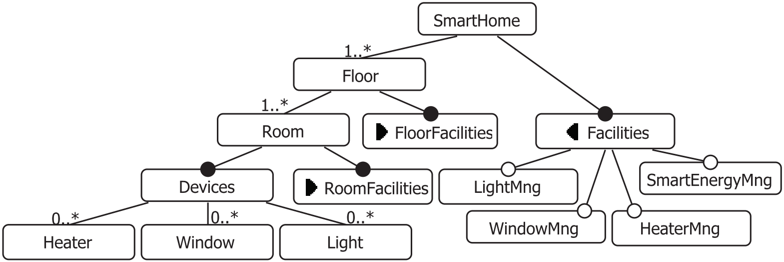


Challenges of constraints involving clonable features



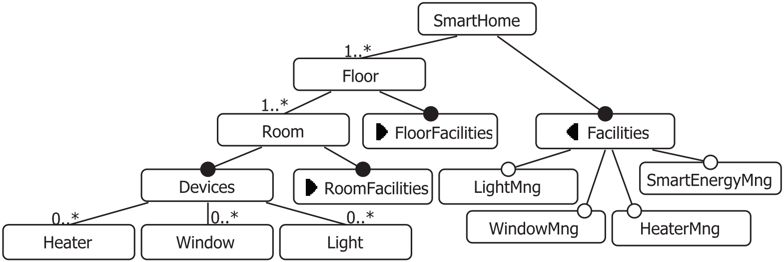
- *LighMng* $\rightarrow$ *Light*. What does it mean? How many Lights? [5]

Challenges of constraints involving clonable features



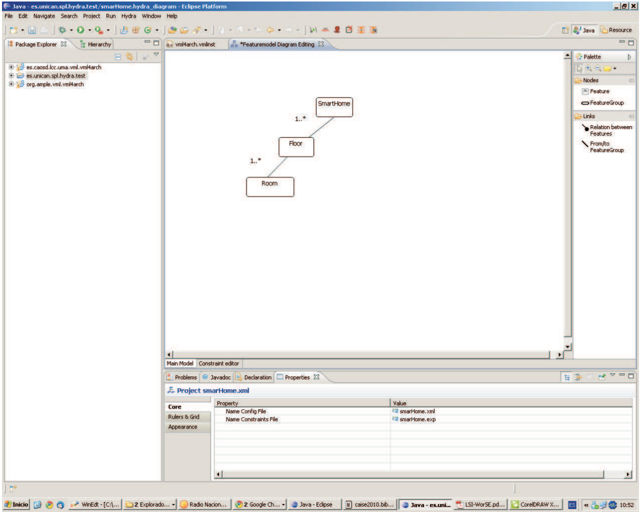
- *LighMng* $\rightarrow$ *Light*. What does it mean? How many Lights? [5]
- If LightMng is selected globally, it must also be selected per floor and room.

Challenges of constraints involving clonable features



- *LighMng* $\rightarrow$ *Light*. What does it mean? How many Lights? [5]
- If LightMng is selected globally, it must also be selected per floor and room.
- If LightMng is selected in a Room, a Light, at least, must also be selected in that a room.

Hydra


Java - es.unican.spl.hydra.test/smarHome.hydra_diagram - Eclipse Platform
Package Explorer
Hierarchy
es.caosd.lcc.uma.vml.vml4arch
es.unican.spl.hydra.test
org.ample.vml.vml4arch
vml4arch.vmlinst
*Featuremodel Diagram Editing
SmartHome
1..*
Floor
1..*
Room
Palette
Nodes
Feature
FeatureGroup
Links
Relation between Features
From/to FeatureGroup
Main Model
Constraint editor
Problems
Javadoc
Declaration
Properties
Project smarHome.xml
Core
Rulers & Grid
Appearance
Property
Value
Name Config File
smarHome.xml
Name Constraints File
smarHome.exp

Hydra

- Visual edition of cardinality-based feature models.

Hydra

- Visual edition of cardinality-based feature models.
- Assisted and visual edition of (multiple) configurations of cardinality-based feature models.

Hydra

- Visual edition of cardinality-based feature models.
- Assisted and visual edition of (multiple) configurations of cardinality-based feature models.
- A textual editor for constraints including clonable features (without contexts).

Hydra

- Visual edition of cardinality-based feature models.
- Assisted and visual edition of (multiple) configurations of cardinality-based feature models.
- A textual editor for constraints including clonable features (without contexts).
- A reasoner for these constraints.

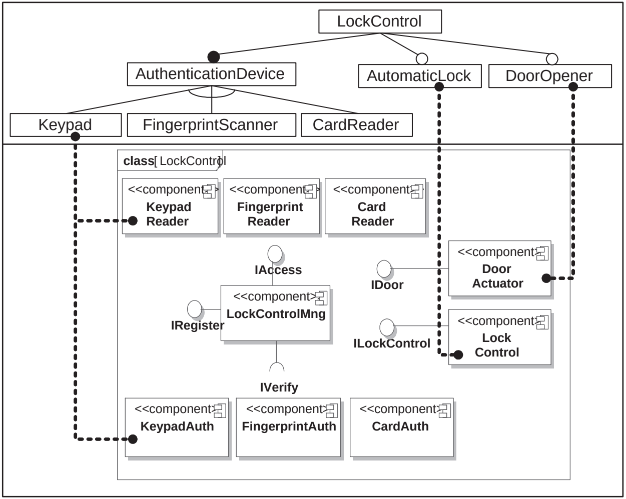
Hydra

- Visual edition of cardinality-based feature models.
- Assisted and visual edition of (multiple) configurations of cardinality-based feature models.
- A textual editor for constraints including clonable features (without contexts).
- A reasoner for these constraints.
- Constructed following model-driven engineering principles with Ecore, TEF and GMF.

Contributions to SPLE where I have participated

1. Hydra, a tool for creating cardinality-based feature models.
2. VML (Variability Management Language).
3. TENTE, a model-driven feature-oriented process for SPL engineering.

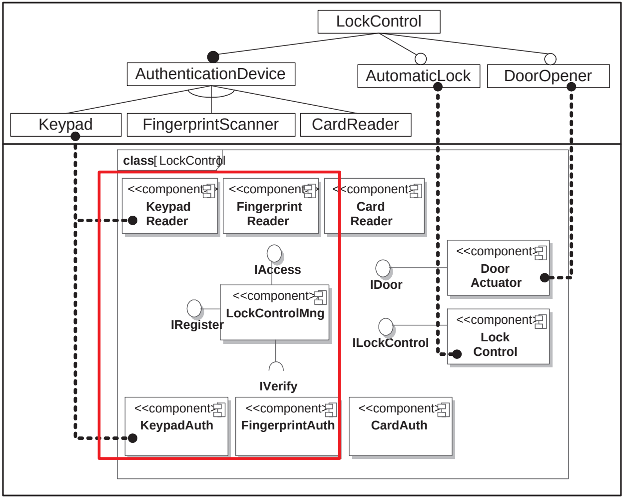
Variability Management Language



Variability Management Language

```
import features <"/lockControl.fmp">;
import core <"/lockControl.uml">;

Concern LockControl{

  variant for Keypad {
     connect(KeypadReader,LockControlMng) using interface(IAccess);
     connect(KeypadReader,LockControlMng) using interface(IRegister);
     connect(LockControlMng,KeypadAuth) using interface(IVerify);
     connect(KeypadAuth,LockControlMng) using interface(IRegister);
  }

  variant for not(Keypad) {
       remove(KeypadReader);
       remove(KeypadAuth);

} // CrisisManagementSystem
```

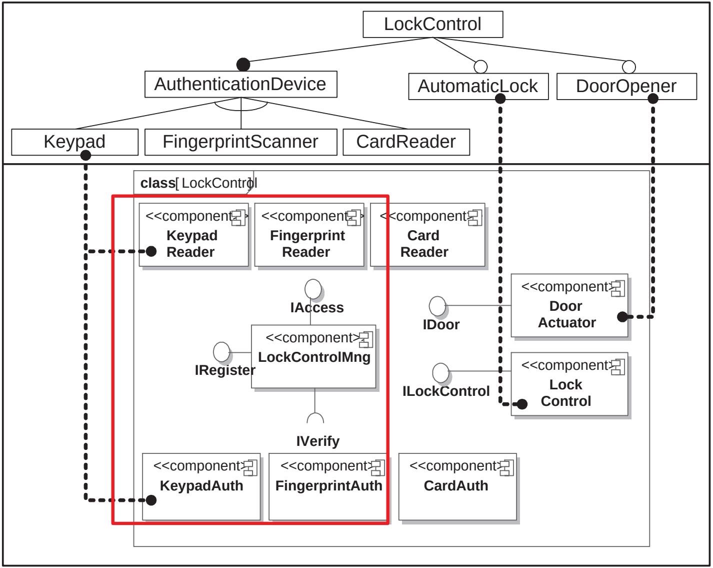
Variability Management Language



Variability Management Language

```
import features <"/lockControl.fmp">;
import core <"/lockControl.uml">;

Concern LockControl{

  variant for Keypad {
     connect(KeypadReader,LockControlMng) using interface(IAccess);
     connect(KeypadReader,LockControlMng) using interface(IRegister);
     connect(LockControlMng,KeypadAuth) using interface(IVerify);
     connect(KeypadAuth,LockControlMng) using interface(IRegister);
  }

  variant for not(Keypad) {
       remove(KeypadReader);
       remove(KeypadAuth);

} // CrisisManagementSystem
```

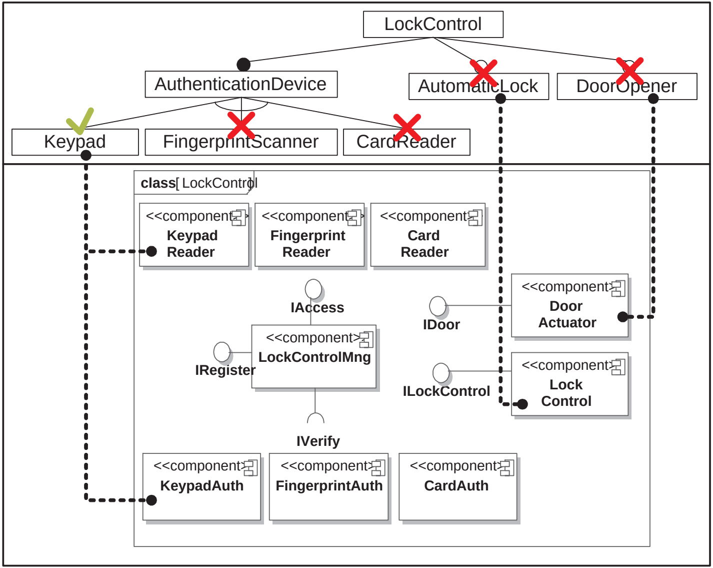
Variability Management Language



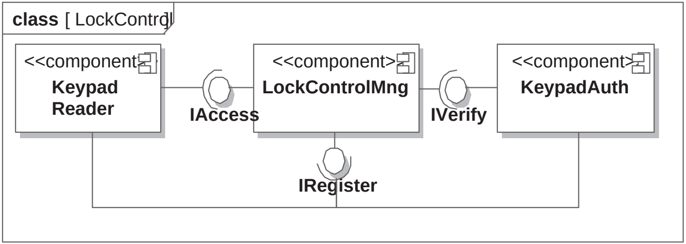
Variability Management Language

```
import features <"/lockControl.fmp">;
import core <"/lockControl.uml">;

Concern LockControl{

  variant for Keypad {
     connect(KeypadReader,LockControlMng) using interface(IAccess);
     connect(KeypadReader,LockControlMng) using interface(IRegister);
     connect(LockControlMng,KeypadAuth) using interface(IVerify);
     connect(KeypadAuth,LockControlMng) using interface(IRegister);
  }

  variant for not(Keypad) {
       remove(KeypadReader);
       remove(KeypadAuth);

} // CrisisManagementSystem
```

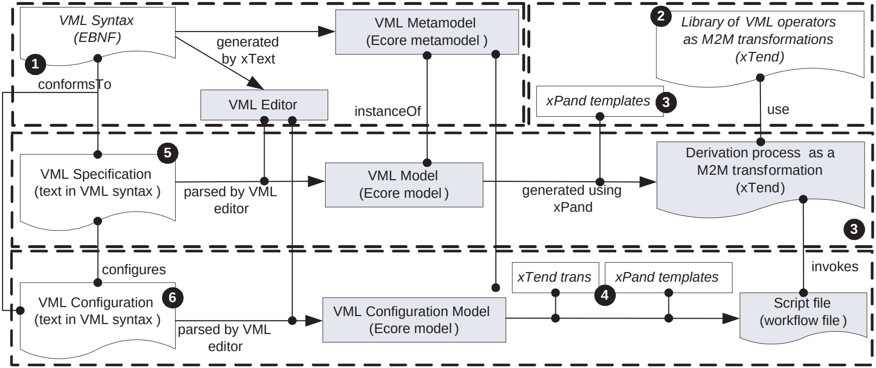
Variability Management Language



Variability Management Language



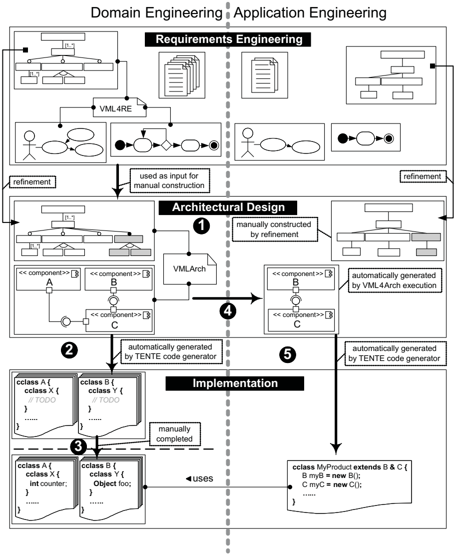
VML compilation process



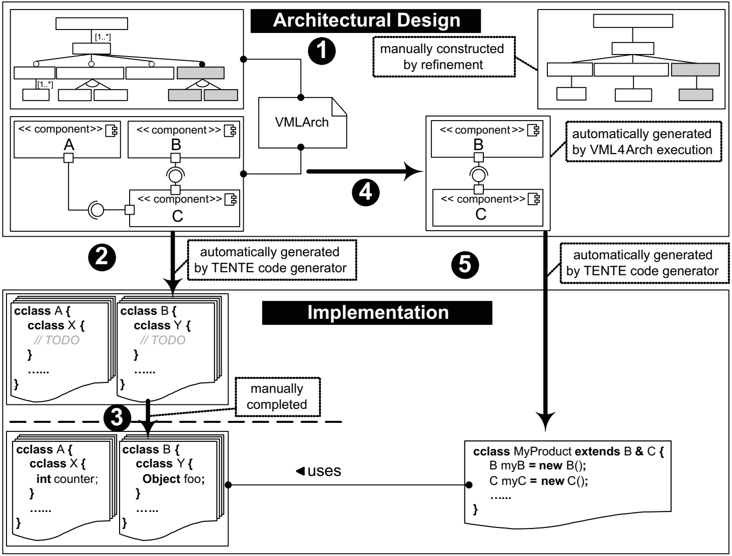
Contributions to SPLE where I have participated

1. Hydra, a tool for creating cardinality-based feature models.
2. VML (Variability Management Language).
3. TENTE, a model-driven feature-oriented process for SPL engineering.

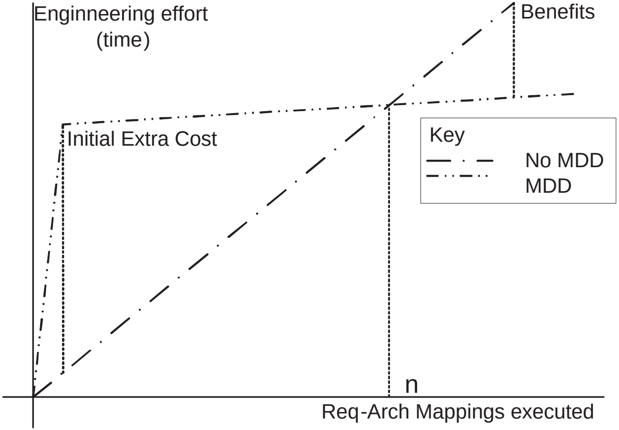
The TENTE SPL engineering process



The TENTE SPL engineering process



Is SPL cost-effective?



Websites

1. AMPLE project & VML: http://www.ample-project.net
2. TENTE: http://caosd.lcc.uma.es/spl/TENTE
3. Hydra: http://caosd.lcc.uma.es/spl/hydra

References I

Don S. Batory.
Feature Models, Grammars, and Propositional Formulas.
In J. Henk Obbink and Klaus Pohl, editors, *Proc. of the 9th Int. Conference on Software Product Lines (SPLC)*, volume 3714 of *LNCS*, pages 7–20, Rennes (France), September 2005.

Don S. Batory, Maider Azanza, and João Saraiva.
The Objects and Arrows of Computational Design.
In Krzysztof Czarnecki, Ileana Ober, Jean-Michel Bruel, Axel Uhl, and Markus Völter, editors, *Proc. of the 11th Int. Conference Model Driven Engineering Languages and Systems (MoDELS)*, volume 5301 of *LNCS*, pages 1–20, Toulouse (France), September - October 2008.

References II

David Benavides, Pablo Trinidad Martín-Arroyo, and Antonio Ruiz Cortés.
Automated Reasoning on Feature Models.
In Oscar Pastor and João Falcão e Cunha, editors, *Proc. of the 17th Int. Conference on Advanced Information Systems Engineering (CAiSE)*, pages 491–503, Porto (Portugal), June 2005.

Krzysztof Czarnecki, Simon Helsen, and Ulrich W. Eisenecker.
Staged Configuration through Specialization and Multilevel Configuration of Feature Models.
Software Process: Improvement and Practice, 10(2):143–169, January-March 2005.

References III

Krzysztof Czarnecki and Chang Hwan Peter Kim.
Cardinality-Based Feature Modeling and Constraints: A Progress Report.
In *Proc. of the 1st. Int. Workshop on Software Factories (SF), 24th Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*. San Diego (California, USA), October 2005.

Lidia Fuentes, Carlos Nebrera, and Pablo Sánchez.
Feature-oriented model-driven software product lines: The TENTE approach.
In Eric Yu, Johann Eder, and Colette Rolland, editors, *Proceedings of the Forum at the CAiSE 2009 Conference*, pages 67–72, 2009.

References IV

Hassan Gomaa.
Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures.
Addison-Wesley, July 2004.

Timo Käkölä and Juan Carlos Dueñas.
Software Product Lines: Research Issues in Engineering and Management.
Springer, October 2006.

Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, and A. Spencer Peterson.
Feature-Oriented Domain Analysis (FODA) Feasibility Study.
Technical Report CMU/SEI-90-TR-021, Software Engineering Institute, Carnegie Mellon University, November 1990.

References V

Neil Loughran, Pablo Sánchez, Alessandro Garcia, and Lidia Fuentes.
Language Support for Managing Variability in Architectural Models.
In Cesare Pautasso and Éric Tanter, editors, *Proc. of the 7th Int. Symposium on Software Composition (SC)*, volume 4954 of *LNCS*, pages 36–51, Budapest (Hungary), March 2008.

Klaus Pohl, Gunther Bockle, and Frank van der Linden.
August.

Pablo Sánchez, Neil Loughran, Lidia Fuentes, and Alessandro Garcia.
Engineering languages for specifying product-derivation processes in software product lines.
In Dragan Gasevic, Ralf Lämmel, and Eric Van Wyk, editors, *Proceedings of the 1st International Conference on Software Language Engineering (SLE)*, volume 5452 of *LNCS*, pages 188–207, Toulouse (France), September 2008.

Questions ?