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.
About me

Main research lines:

- Aspect-Oriented Software Development.
- Model-Driven (Software) Development.
- Software Product Lines.
About me

Main research lines:

- Aspect-Oriented Software Development.
- Model-Driven (Software) Development.
- Software Product Lines.
About me

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

1. **Domain Engineering**
   - **Requirements Engineering**
     - VML4RE refinement used as input for manual construction
     - VMLArch
     - Manually constructed by refinement
   - Architectural Design
     - Automatically generated by VML4Arch execution
     - TENTE code generator
     - Manually completed

2. **Implementation**
   - `cclass A` (`cclass X` `{ #TODO }`)
   - `cclass B` (`cclass Y` `{ #TODO }`)
   - `cclass A` (`cclass X` `{ int counter; }`)
   - `cclass B` (`cclass Y` `{ Object foo; }`)
   - `cclass MyProduct extends B & C` (`B myB = new B(); C myC = new C();`)

---

**Background:** Software Product Line Engineering

**SPL engineering processes**

**Pablo Sánchez (MATESCO)**

**Software Product Line 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 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.
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.
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.
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

- LockControl
  - AuthenticationDevice
  - CardReader
  - FingerprintScanner
  - Keypad
  - AutomaticLock
  - DoorOpener
Variability Realisation/Design

Background: Software Product Line Engineering
Running Example: Lock Control Framework

Pablo Sánchez (MATESCO)
Linking between specification and realisation

Background: Software Product Line Engineering

Running Example: Lock Control Framework

Pablo Sánchez (MATESCO)
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.
Hydra

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

- SmartHome
  - HeaterMng
  - WindowMng
  - SmartEnergyMng
SmartEnergyMng requires WindowMng and HeaterMng.
Constraints on feature models

- SmartEnergyMng requires WindowMng and HeaterMng.
- \( \text{SmartEnergyMng} \rightarrow (\text{WindowMng} \land \text{HeaterMng}) \)
Constraints on feature models

- SmartEnergyMng requires WindowMng and HeaterMng.
- \( \text{SmartEnergyMng} \rightarrow (\text{WindowMng} \land \text{HeaterMng}) \)
- Using BDD, SAT or CSP, we can analyze several properties of a feature model [3, 1]:

[Diagram showing the hierarchy of feature models: SmartHome with submodels HeaterMng, WindowMng, and SmartEnergyMng.]
Constraints on feature models

SmartEnergyMng requires WindowMng and HeaterMng.

\[ \text{SmartEnergyMng} \rightarrow (\text{WindowMng} \land \text{HeaterMng}) \]

Using BDD, SAT or CSP, we can analyze several properties of a feature model [3, 1]:

1. Autocomplete.
Constraints on feature models

- SmartEnergyMng requires WindowMng and HeaterMng.

- SmartEnergyMng → (WindowMng ∧ HeaterMng)

- Using BDD, SAT or CSP, we can analyze several properties of a feature model [3, 1]:
  1. Autocomplete.
  2. Dead features.
Constraints on feature models

SmartEnergyMng requires WindowMng and HeaterMng.

\[ \text{SmartEnergyMng} \rightarrow (\text{WindowMng} \land \text{HeaterMng}) \]

Using BDD, SAT or CSP, we can analyze 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

Challenges of constraints involving clonable features

- \textit{LightMng} → \textit{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

- \( LightMng \rightarrow Light \). 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

Hydra

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

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

- 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

- 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
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

class LockControl

<<component Keypad Reader
<<component Fingerprint Reader
<<component Card Reader

<<component LockControlMng
<<component KeypadAuth
<<component FingerprintAuth
<<component CardAuth

IDoor
IRegister
IVerify
IAccess
ILockControl
AutomaticLock
DoorOpener
AuthenticationDevice
Keypad
FingerprintScanner
CardReader

Pablo Sánchez (MATESCO) Software Product Line Engineering
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

```
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

class LockControl

<<component>> KeypadReader
<<component>> FingerprintReader
<<component>> CardReader

ILockControl
IRegister
IAccess
IVerify

<<component>> KeypadAuth
<<component>> FingerprintAuth
<<component>> CardAuth

<<component>> LockControlMng

<<component>> DoorActuator

<<component>> LockControl

AutomaticLock
DoorOpener
Variability Management Language
VML compilation process

1. VML Syntax (EBNF) - conformsTo
2. Library of VML operators as M2M transformations (xTend)
3. xPand templates
4. Derivation process as a M2M transformation (xTend)
5. VML Specification (text in VML syntax) parsed by VML editor
6. VML Configuration (text in VML syntax) parsed by VML editor

VML Editor
VML Metamodel (Ecore metamodel)
VML Model (Ecore model)
VML Configuration Model (Ecore model)
Script file (workflow file)

xTend trans
xPand templates

Pablo Sánchez (MATESCO)
Software Product Line Engineering
LSI-WorSe 19 / 30
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

Architectural Design

1. manually constructed by refinement

2. automatically generated by TENTE code generator

3. manually completed

4. automatically generated by VML4Arch execution

5. automatically generated by TENTE code generator

Implementation

cclass A

class X {
  // TODO
  // ...
}

cclass B

class Y {
  // TODO
  // ...
}

cclass MyProduct extends B & C {
  B myB = new B();
  C myC = new C();
  // ...
}

uses

cclass A

class X {
  int counter;
  // ...
}

cclass B

class Y {
  Object foo;
  // ...
}

Pablo Sánchez (MATESCO)  Software Product Line Engineering  LSI -WorSe 22 / 30
Is SPL cost-effective?

![Graph showing engineering effort vs. initial extra cost and benefits](image)

**Key**
- No MDD
- MDD

**Benefits**
- Initial Extra Cost
- Engineering effort (time)
- ReqArch Mappings executed

**Legend**
- Solid line: No MDD
- Dashed line: MDD
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

David Benavides, Pablo Trinidad Martín-Arroyo, and Antonio Ruiz Cortés.  
Automated Reasoning on Feature Models.  

Krzysztof Czarnecki, Simon Helsen, and Ulrich W. Eisenecker.  
Staged Configuration through Specialization and Multilevel Configuration of Feature Models.  
Krzysztof Czarnecki and Chang Hwan Peter Kim. 
Cardinality-Based Feature Modeling and Constraints: A Progress Report. 

Lidia Fuentes, Carlos Nebrera, and Pablo Sánchez. 
Feature-oriented model-driven software product lines: The TENTE approach. 
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.

Feature-Oriented Domain Analysis (FODA) Feasibility Study.  
References V


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

Questions ?