An Analysis of Variability Modeling and Management Tools for Product Line Development

Rafael Capilla, Alejandro Sánchez
Universidad Rey Juan Carlos de Madrid

Juan C. Dueñas
Universidad Politécnica de Madrid

SPAIN

Motivation

- Time to market of complex software systems and Product Family/Product Line approaches demand variability-based solutions.
- Software Variability is key for PL development.
- The increasing number of variation points needs of techniques and tools able to manage the variability of systems.
- Managing variability at runtime becomes a complex problem that has to be addressed.
- Different variability techniques and approaches have been proposed and used but a lack of agreements is missing.
- Several tools for the same goal.

Origins

- FODA models (1990) are used for modeling and representing the common and variable parts of software systems.
- Describe the external and visible properties of systems
- Requirements are modeled as “features”
- Features are used to describe the variability and the relationships among these features
- Complex dependencies are not straightforward

Variability concepts

- Variation point: An area of a software system affected by variability
- Variant: The alternatives defined for each particular VP
- Variability in space (allowed set of product configurations) and variability in time (when the variability is realized)
- Extensibility. Closed VP (variants are known at pre-deployment time) Open VP (VP and variants can be added at runtime)
- Varibility realization: Different techniques proposed for implementing the variability model (i.e.: inheritance, aggregation, parameterization, etc...)

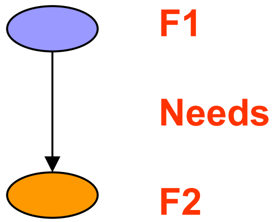
Feature dependency analysis

- Dependencies and constraints limit the number and type of configurable products from the number of possible configurations
- Dependency links introduce a factor of complexity that variability models have to manage.
- Several dimensions and categories are possible in the variability model.
- Complex and simple dependencies are possible.
- Dependencies can be used to establish traceability issues.

Types of dependencies

A feature may depend on other features

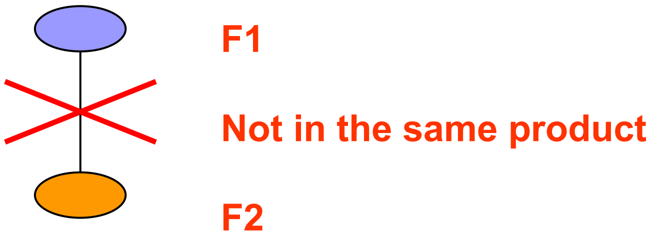
- AND, OR, XOR boolean relationships
- Requires and excludes dependencies



F1

Needs

F2

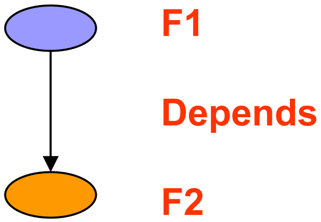


F1

Not in the same product

F2

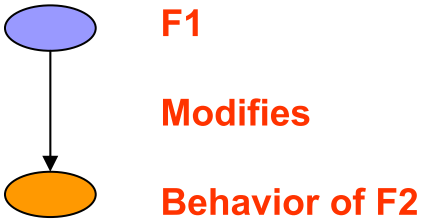
- Usage, Modification, Others [LEE & KANG, ICSR2004]



F1

Depends

F2

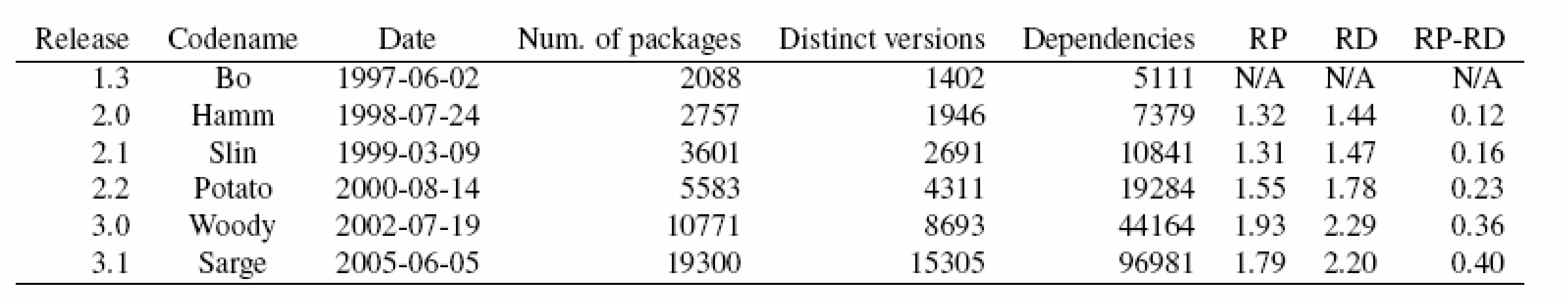


F1

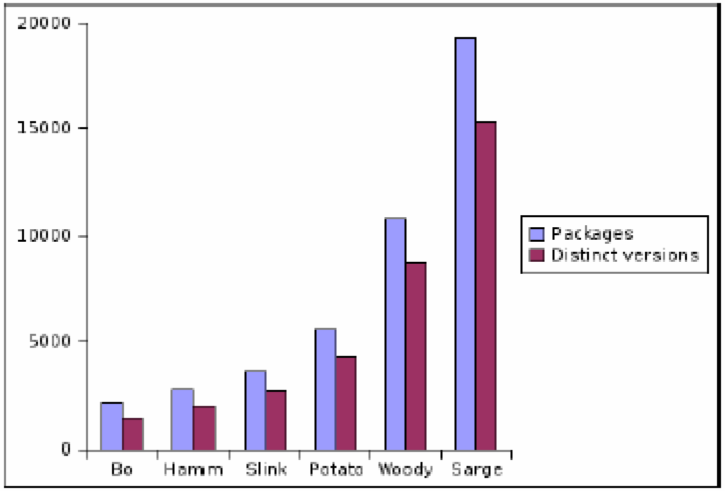
Modifies

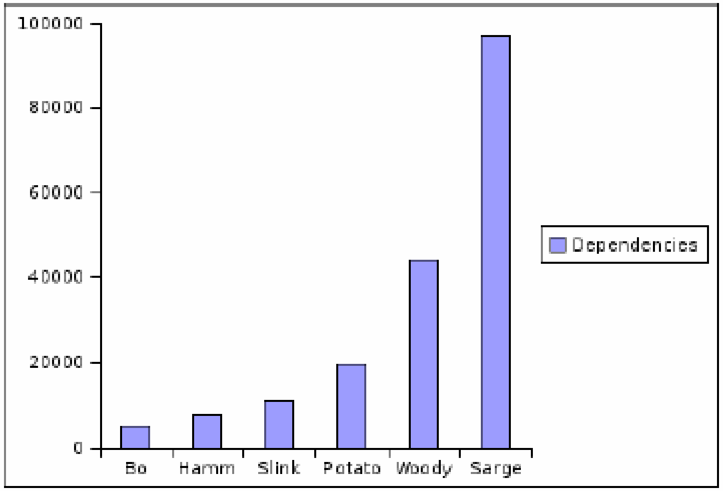
Behavior of F2

Dependencies in Linux systems



Release	Codename	Date	Num. of packages	Distinct versions	Dependencies	RP	RD	RP-RD
1.3	Bo	1997-06-02	2088	1402	5111	N/A	N/A	N/A
2.0	Hamm	1998-07-24	2757	1946	7379	1.32	1.44	0.12
2.1	Slin	1999-03-09	3601	2691	10841	1.31	1.47	0.16
2.2	Potato	2000-08-14	5583	4311	19284	1.55	1.78	0.23
3.0	Woody	2002-07-19	10771	8693	44164	1.93	2.29	0.36
3.1	Sarge	2005-06-05	19300	15305	96981	1.79	2.20	0.40





Variability management

- Deals with the processes to create, use, modify, maintain and document the variability model across different stages of the lifecyle.
- Variation points and their dependencies have to be maintained.
- Some approaches / tools provide adequate visualization support.
- Integration with source code is needed for product derivation.
- Evolution of products and VPs must be supported.
- Feature categorization for PL development facilitates the visualization of feature models.

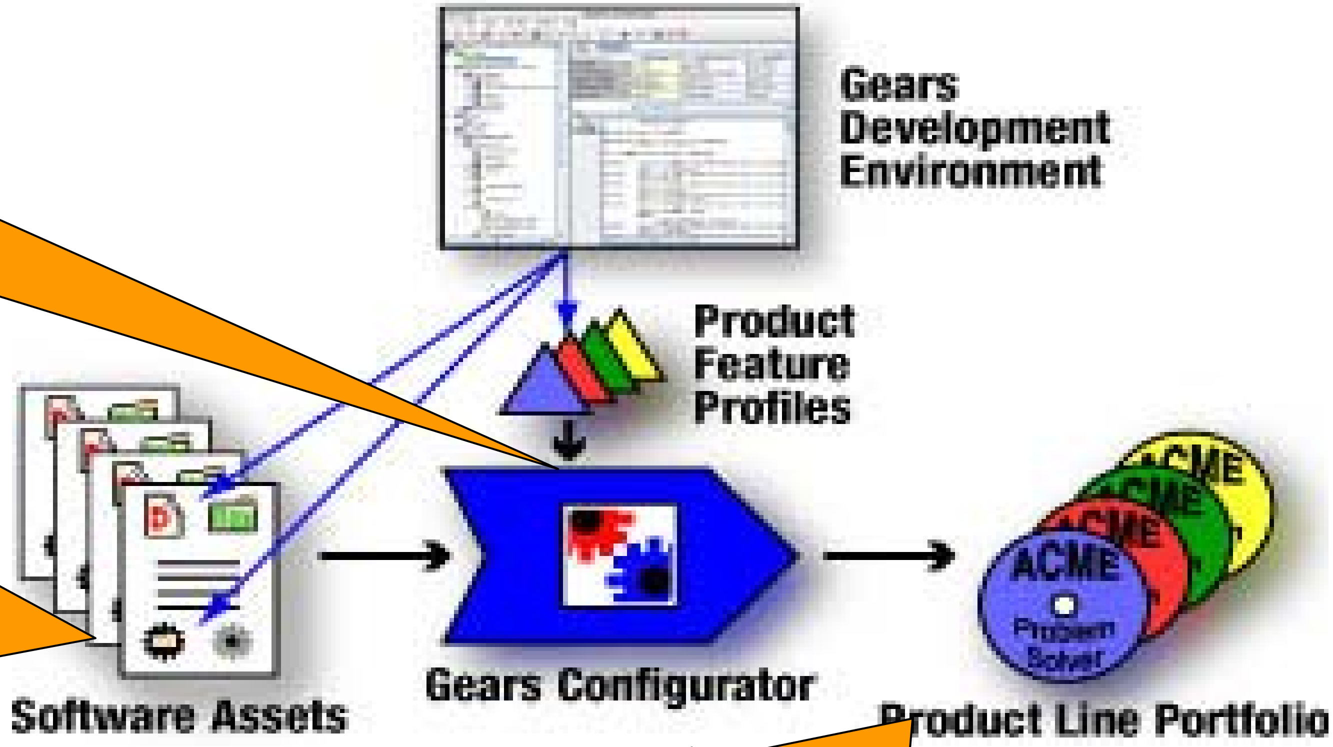
Tools for modeling and managing variabiliy for PL development

- The scope of the study carried out was focused on variability and management and modeling tools.
- The analysis describes the main characteristics of the tools from the information gathered from the authors (interviews, discussions).
- No tool evaluation was done because two of them are not available for the general public.
- No cross-comparison and ranking between tools was made.

Tools for modeling and managing variabiliy for PL development GEARS

- GEARS is a commercial SPL development tool from BigLever Inc.
- GEARS enables modeling optional and varying features for producing different products
- GEARS distinguises between **features at the domain modeling** level and **variations points at the implementation level**
- Components with variation points are turned into reusable assets that are automatically composed and configured into product instances
- Dependencies are expressed as relational assertions which may contain 3 or more features and relationships
- Specific **product profiles** to select the choices for each product in the feature model

Tools for modeling and managing variabiliy in PL development GEARS



Feature models
A product feature profile is created for each product in the portfolio

Configurable assets and variation points
Gears provides a language for programming each VP

A product configurator automatically produces the products in the portfolio composing the assets and configuring the VP

Gears manufactures each product based on specific profiles
and reflect the changes when assets are modified

Tools for modeling and managing variabiliy in PL development V-Manage

- V-Manage tool for small/medium organizations that want to implement a PL and supports SFE in the context of MDA

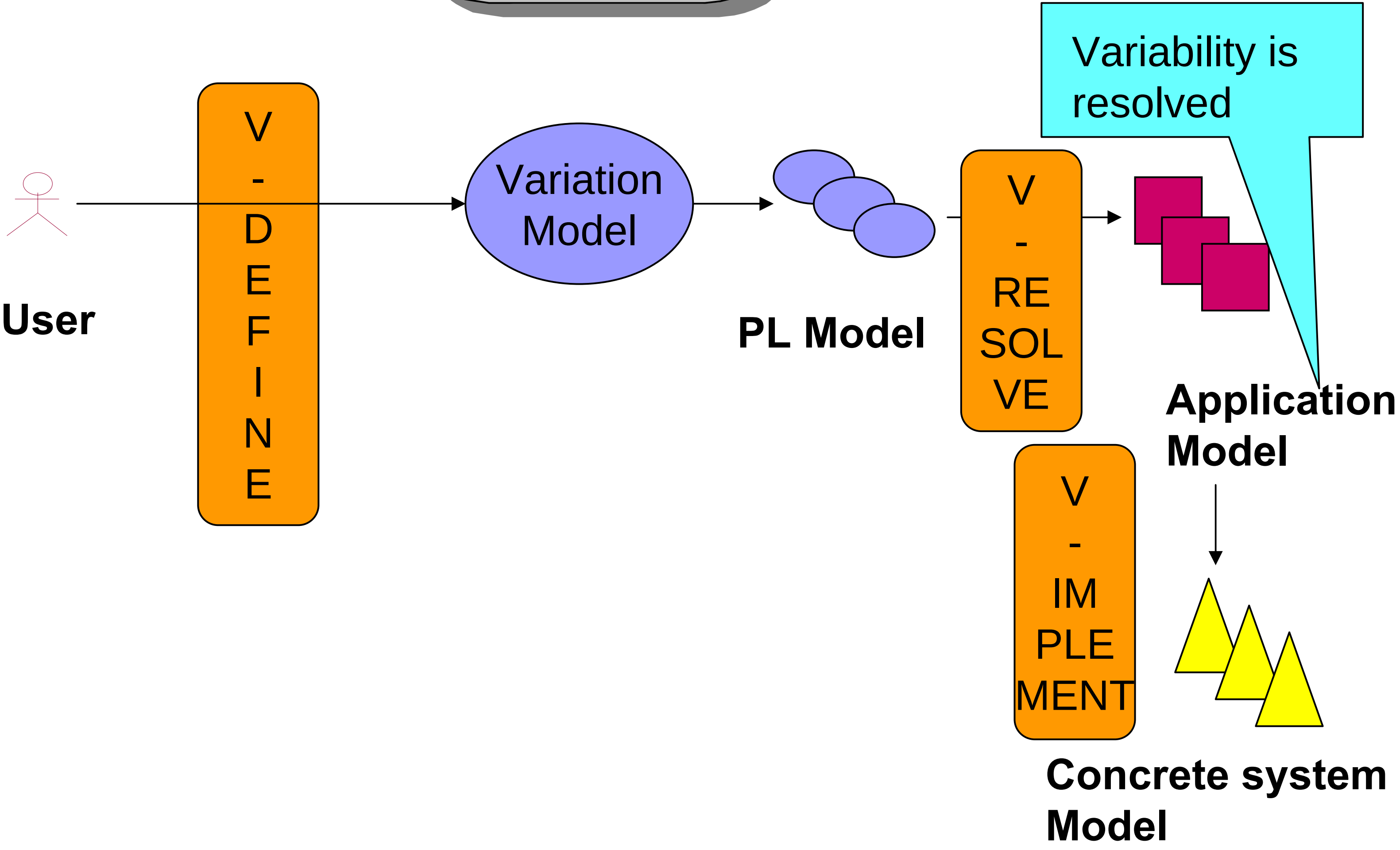
- V-manage consists of 3 modules:

  - V-define: Represents the variation model (i.e.: decision model) and its relationships

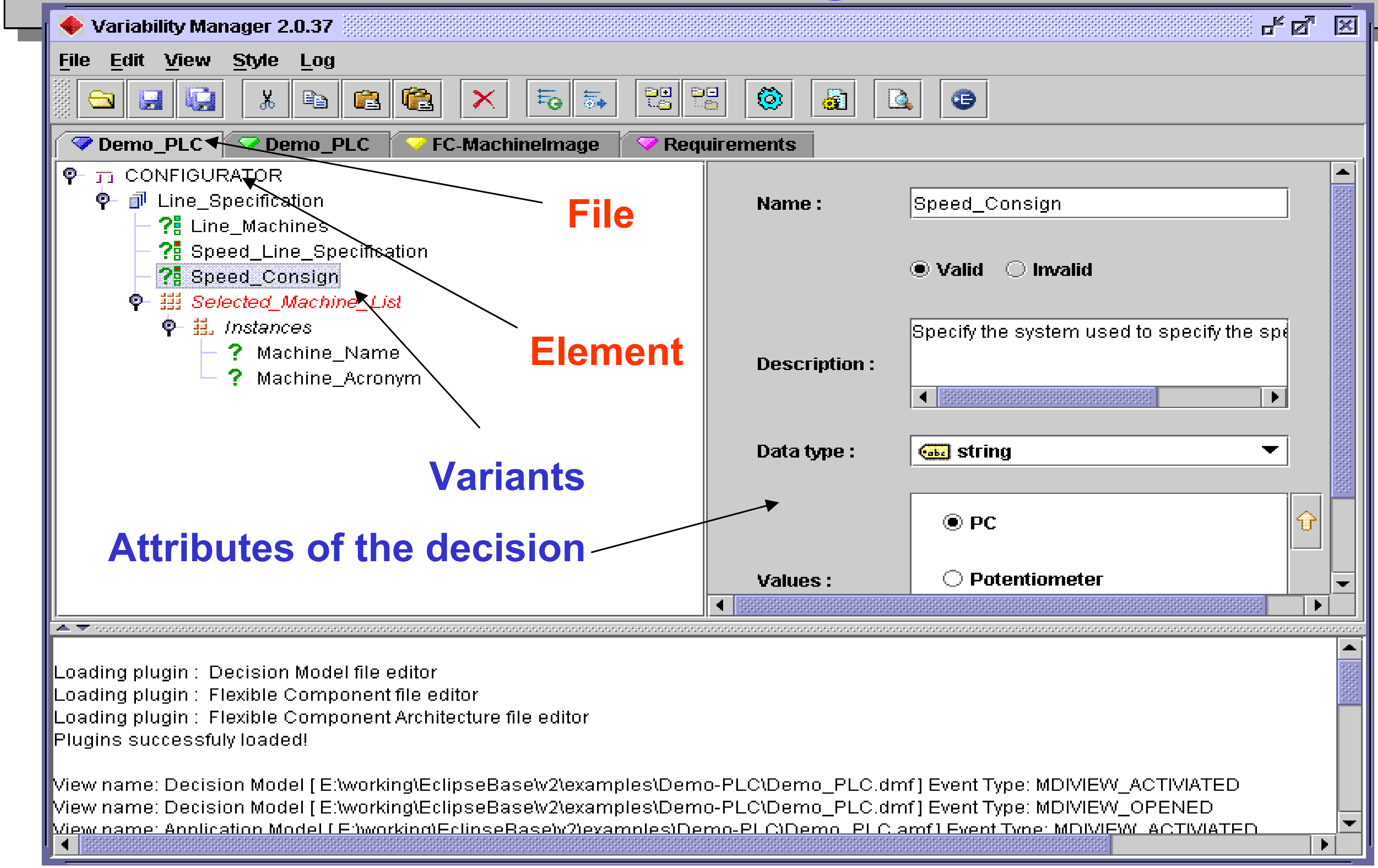
  - V-Resolve: Builds application models and sets the values of the decision model to produce a suitable configuration

  - V-Implement: Produces the reusable components

Tools for modeling and managing variabiliy in PL development V-Manage



Tools for modeling and managing variabiliy in PL development **V-Manage**



File

Element

Variants

Attributes of the decision

Name : Speed_Consign

● Valid ○ Invalid

Description : Specify the system used to specify the spe

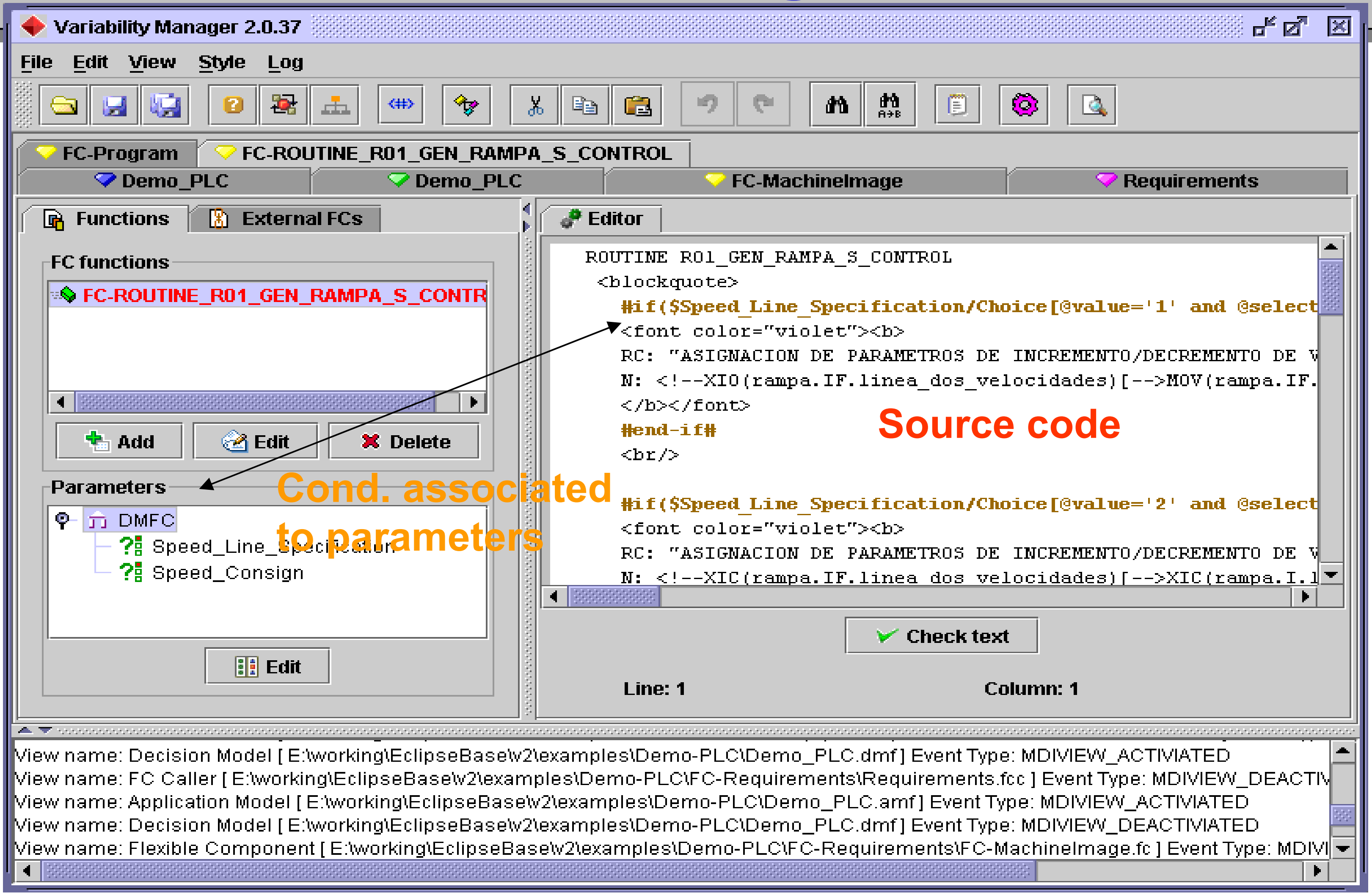
Data type : string

Values : ● PC ○ Potentiometer

```
Loading plugin :  Decision Model file editor
Loading plugin :  Flexible Component file editor
Loading plugin :  Flexible Component Architecture file editor
Plugins successfuly loaded!

View name: Decision Model [ E:\working\EclipseBase\v2\examples\Demo-PLC\Demo_PLC.dmf ] Event Type: MDIVIEW_ACTIVATED
View name: Decision Model [ E:\working\EclipseBase\v2\examples\Demo-PLC\Demo_PLC.dmf ] Event Type: MDIVIEW_OPENED
View name: Application Model [ E:\working\EclipseBase\v2\examples\Demo-PLC\Demo_PLC.amf ] Event Type: MDIVIEW_ACTIVATED
```

Tools for modeling and managing variabiliy in PL development V-Manage



```
ROUTINE R01_GEN_RAMPA_S_CONTROL
 <blockquote>
   #if($Speed_Line_Specification/Choice[@value='1' and @select
   <font color="violet"><b>
   RC: "ASIGNACION DE PARAMETROS DE INCREMENTO/DECREMENTO DE V
   N: <!--XIO(rampa.IF.linea_dos_velocidades)[-->MOV(rampa.IF.
   </b></font>
   #end-if#
   <br/>

   #if($Speed_Line_Specification/Choice[@value='2' and @select
   <font color="violet"><b>
   RC: "ASIGNACION DE PARAMETROS DE INCREMENTO/DECREMENTO DE V
   N: <!--XIC(rampa.IF.linea_dos_velocidades)[-->XIC(rampa.I.1
```

```
View name: Decision Model [ E:\working\EclipseBase\v2\examples\Demo-PLC\Demo_PLC.dmf] Event Type: MDIVIEW_ACTIVIATED
View name: FC Caller [ E:\working\EclipseBase\v2\examples\Demo-PLC\FC-Requirements\Requirements.fcc ] Event Type: MDIVIEW_DEACTIV
View name: Application Model [ E:\working\EclipseBase\v2\examples\Demo-PLC\Demo_PLC.amf] Event Type: MDIVIEW_ACTIVIATED
View name: Decision Model [ E:\working\EclipseBase\v2\examples\Demo-PLC\Demo_PLC.dmf] Event Type: MDIVIEW_DEACTIVIATED
View name: Flexible Component [ E:\working\EclipseBase\v2\examples\Demo-PLC\FC-Requirements\FC-MachineImage.fc ] Event Type: MDIVI
```

Source code

Cond. associated to parameters

Tools for modeling and managing variabiliy in PL development COVAMOF

- COVAMOF (ConIPF Variability Modeling Framework) is a tool for representing VP and variants at all levels of abstraction

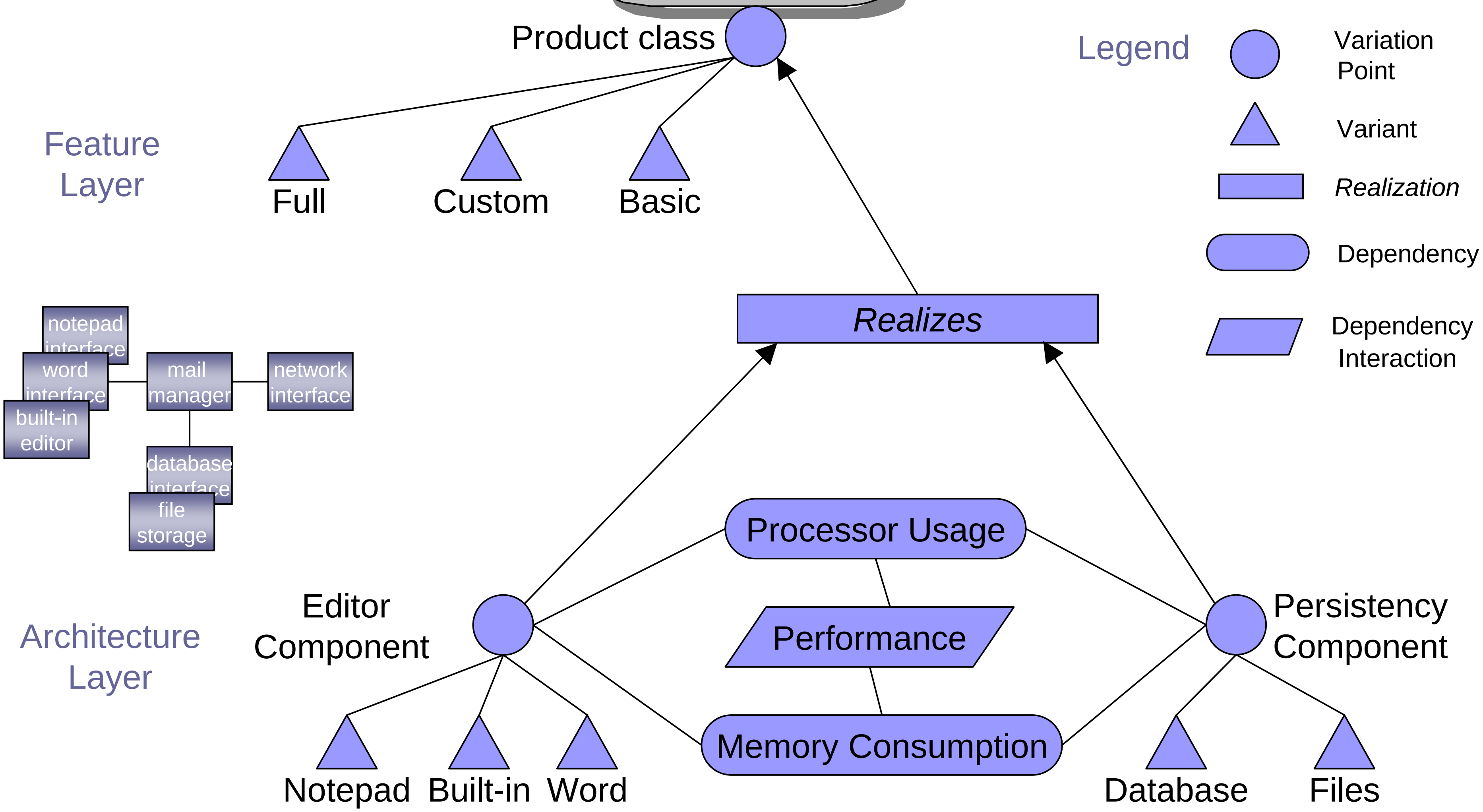
- Supports dependencies and a hierarchical variability model. Complex dependencies are defined as dynamically analizable dependencies

- 5 types of VP are supported

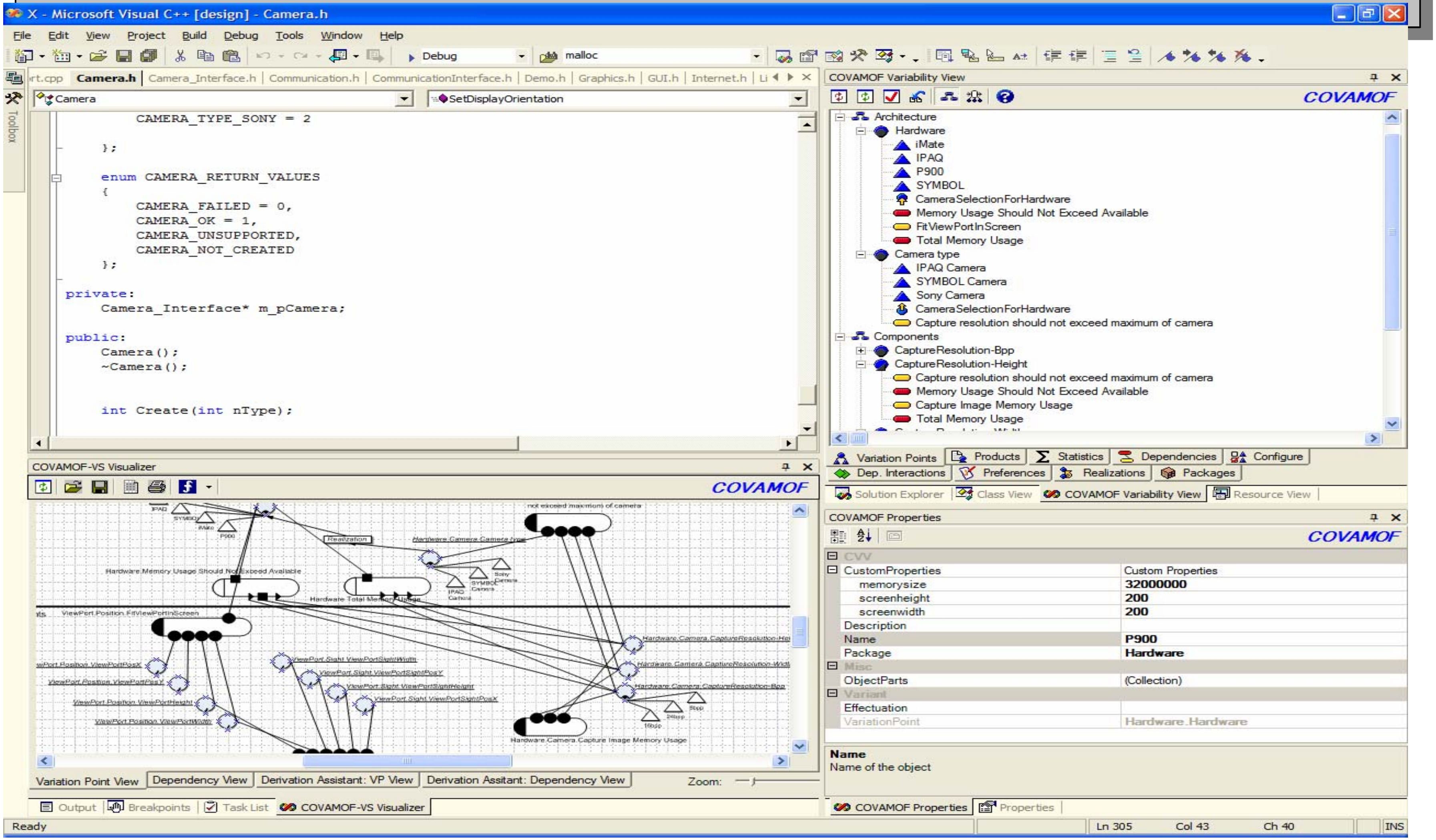
- COVAMOF Variability View (CVV) represents the view of the variability for PF artifacts and unifies this variability on all layers. CVV also models the dependencies to restrict the binding of VPs

- The Mocca tool supports multiple variability views in COVAMOF

Tools for modeling and managing variabiliy in PL development COVAMOF



Tools for modeling and managing variabiliy in PL development COVAMOF



```
        CAMERA_TYPE_SONY = 2

    };

    enum CAMERA_RETURN_VALUES
    {
        CAMERA_FAILED = 0,
        CAMERA_OK = 1,
        CAMERA_UNSUPPORTED,
        CAMERA_NOT_CREATED
    };

private:
    Camera_Interface* m_pCamera;

public:
    Camera();
    ~Camera();


    int Create(int nType);
```

COVAMOF Variability View

- Architecture
  - Hardware
    - iMate
    - IPAQ
    - P900
    - SYMBOL
    - CameraSelectionForHardware
    - Memory Usage Should Not Exceed Available
    - FitViewPortInScreen
    - Total Memory Usage
  - Camera type
    - IPAQ Camera
    - SYMBOL Camera
    - Sony Camera
    - CameraSelectionForHardware
    - Capture resolution should not exceed maximum of camera
- Components
  - CaptureResolution-Bpp
  - CaptureResolution-Height
    - Capture resolution should not exceed maximum of camera
    - Memory Usage Should Not Exceed Available
    - Capture Image Memory Usage
    - Total Memory Usage

COVAMOF
Variation Points
Products
Statistics
Dependencies
Configure
Dep. Interactions
Preferences
Realizations
Packages
Solution Explorer
Class View
COVAMOF Variability View
Resource View
COVAMOF-VS Visualizer
COVAMOF
IPAQ
SYMBOL
iMate
P900
Realization
Hardware.Camera.Camera type
not exceed maximum of camera
Hardware.Memory Usage Should Not Exceed Available
Hardware.Total Memory Usage
IPAQ Camera
SYMBOL Camera
Sony Camera
ViewPort.Position.FitViewPortInScreen
wPort.Position.ViewPortPosX
ViewPort.Position.ViewPortPosY
ViewPort.Position.ViewPortHeight
ViewPort.Position.ViewPortWidth
ViewPort.Sight.ViewPortSightWidth
ViewPort.Sight.ViewPortSightPosY
ViewPort.Sight.ViewPortSightHeight
ViewPort.Sight.ViewPortSightPosX
Hardware.Camera.CaptureResolution-Hei
Hardware.Camera.CaptureResolution-Widt
Hardware.Camera.CaptureResolution-Bpp
16bpp
24bpp
8bpp
Hardware.Camera.Capture Image Memory Usage
Variation Point View
Dependency View
Derivation Assistant: VP View
Derivation Assitant: Dependency View
Zoom:
Output
Breakpoints
Task List
COVAMOF-VS Visualizer
COVAMOF Properties
COVAMOF
COVAMOF Properties
Properties
Ready
Ln 305
Col 43
Ch 40
INS

COVAMOF Properties

⊟ CVV	
⊟ CustomProperties	Custom Properties
memorysize	32000000
screenheight	200
screenwidth	200
Description	
Name	P900
Package	Hardware
⊟ Misc	
ObjectParts	(Collection)
⊟ Variant	
Effectuation	
VariationPoint	Hardware.Hardware

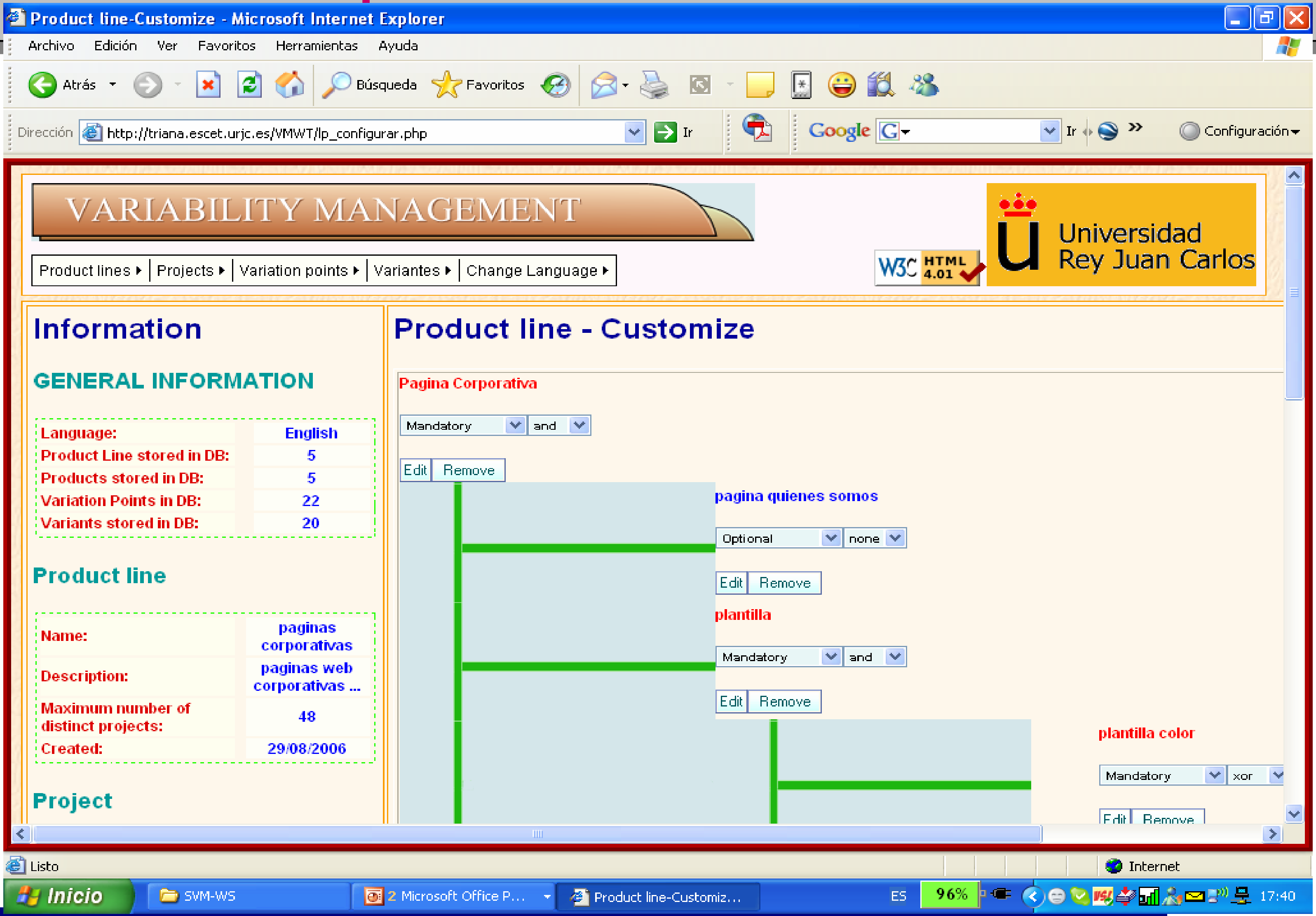
Name
Name of the object

R. Capilla, A. Sánchez, J.C. Dueñas, SVM-WS'07, Helsinki, Finland

Tools for modeling and managing variabiliy in PL development VMWT

- The variability Modeling Web Tool is a Web-based tool with PHP+AJAX for modeling variability for product line development
- FODA trees for visualizing the variability model
- Dependencies supported: Boolean connectors and requires, excludes
- Dependency checking before producing the product configuration
- Computes the number of allowed products
- Automatic documentation as PDF documents
- VP and variants are included in the code assets

Tools for modeling and managing variabiliy in PL development VMWT



Product line-Customize - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Búsqueda Favoritos
Dirección http://triana.escet.urjc.es/VMWT/lp_configurar.php Ir
Google Ir Configuración
VARIABILITY MANAGEMENT
Product lines Projects Variation points Variantes Change Language
W3C HTML 4.01
Universidad Rey Juan Carlos
Listo Internet
Inicio SVM-WS 2 Microsoft Office P... Product line-Customiz... ES 96% 17:40

Information

GENERAL INFORMATION

Language:	English
Product Line stored in DB:	5
Products stored in DB:	5
Variation Points in DB:	22
Variants stored in DB:	20

Product line

Name:	paginas corporativas
Description:	paginas web corporativas ...
Maximum number of distinct projects:	48
Created:	29/08/2006

Project

Product line - Customize

Pagina Corporativa

Mandatory and

Edit Remove

pagina quienes somos

Optional none

Edit Remove

plantilla

Mandatory and

Edit Remove

plantilla color

Mandatory xor

Edit Remove

Tools for modeling and managing variabiliy in PL development AHEAD

- The AHEAD (Algebraic Hierarchical Equations for Application Development) tool suite supports the development of PL by means of compositional programming and based on the GenVoca methodology for incrementally add features to product family members

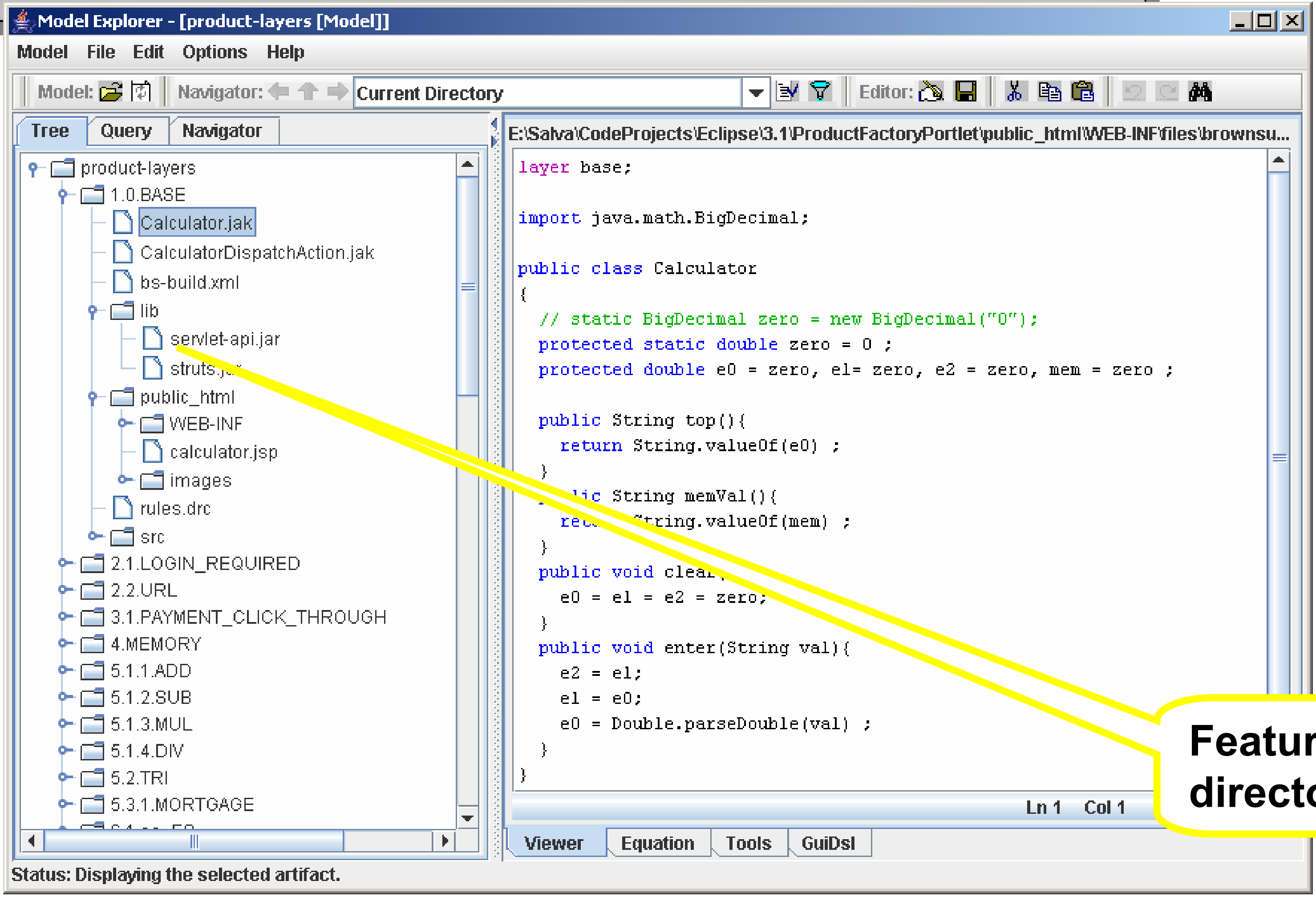
- The key tool in AHEAD tool suit is the composer, which expands AHEAD equations to yield the target system

- AHEAD distinguishes between “product features” and “built-in features”

- AHEAD uses a step-wise refinement process. Refinements are packaged in layers. The base layer contains base artifacts which are enhanced with specific features

- Automatic derivation process

Tools for modeling and managing variabiliy in PL development AHEAD



E:\Salva\CodeProjects\Eclipse\3.1\ProductFactoryPortlet\public_html\WEB-INF\files\brownsu...

```
layer base;

import java.math.BigDecimal;

public class Calculator
{
  // static BigDecimal zero = new BigDecimal("0");
  protected static double zero = 0 ;
  protected double e0 = zero, e1= zero, e2 = zero, mem = zero ;

  public String top(){
    return String.valueOf(e0) ;
  }
  public String memVal(){
    return String.valueOf(mem) ;
  }
  public void clear(){
    e0 = e1 = e2 = zero;
  }
  public void enter(String val){
    e2 = e1;
    e1 = e0;
    e0 = Double.parseDouble(val) ;
  }
}
```

Status: Displaying the selected artifact.

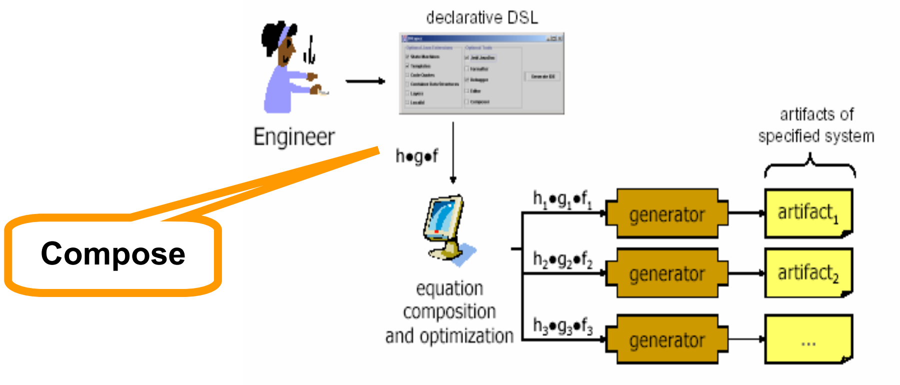
model explorer

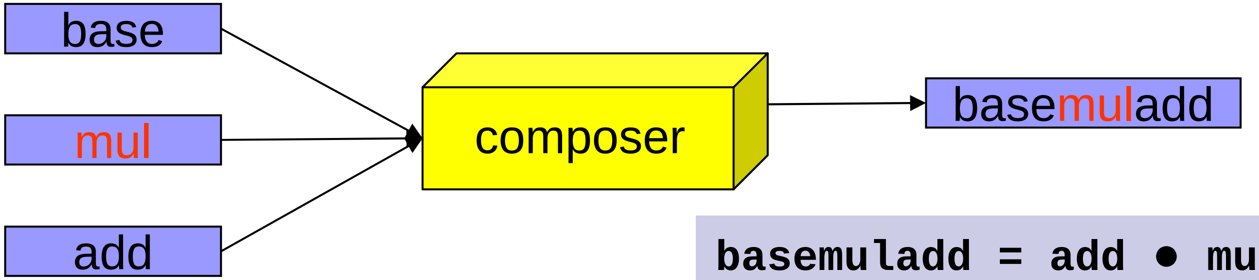
Features are directories!!

Tools for modeling and managing variabiliy in PL development AHEAD

Composing Features

```
> composer --equation=baseRef
#baseref.equation
base
ref
```





```
basemuladd = add • mul • base
```

```
> composer –target=basemuladd add mul base
```

Impact on SPL development

- Several successful examples have demonstrated how Product Lines are useful for industry to meet market demands

- Examples of savings and reuse producing multiple products in parallel are provided in the literature

  e.g.: Around the 80% of the code produced by Engenio (a firm to high performance disk storage systems) is common to 82 products of the firm

- Tools are becoming key pieces for managing the increasing amount of variability in PFE and for automating the derivation process

Conlusions

- Most of the tools examines share many similarities
- Lack of a unified variability approach that leads to several tools / approaches, and notations
- Visualizing hundreds of VP is a limitation to overcome
- Need to handle complex dependencies
- Incompatible versions of products must be checked before product derivation
- Managing variability at runtime is hard

Future needs – Discussion topics

- Integration with other SE development tools and SCM tools
- More visualization capabilities
- Support for runtime variability
- Agreement on standard notation?
- Estimation of the cost of products from the product portfolio
- Better integration from modeling and configuration to product synthesis
- Identify and extract features from code (reverse engineering, feature location) to integrate them with an existing feature model