

# ソフトウェアプロダクトライン開発セミナー

先人の経験に学ぶバリエーション/バリエアビリティ管理のコツ

Dr. Danilo Beuche



# 予定

ソフトウェアプロダクトライン開発について

ソースコードのバリエーションの扱い

要件管理 (要件の再利用に関する課題)

モデルベース開発 (UML、Simulinkモデルなど)

テストの管理

欠陥 (バグ) ・変更の管理

# 講師



- Dr. Danilo Beuche
  - 2001-: ドイツ、マグデブルグ pure-systems社 代表取締役
    - 組み込みソフトウェア開発とプロダクトライン開発におけるツールとコンサルティングの提供
  - 1997-2003: ドイツ、マグデブルグ大学にて、研究員、研究助手(博士)
    - 組み込みシステムにおけるソフトウェアファミリーに於ける博士
    - 組み込みOSファミリーに着目したOS研究グループに所属
  - 1995-1997: ドイツ国立情報処理研究所 (GMD FIRST 現、フラウンホーファー研究所) 研究助手

Dr.Danilo Product Line Engineering Blog(日本語訳)

[http://www.fuji-setsu.co.jp/products/purevariants/Danilo\\_Blog.html](http://www.fuji-setsu.co.jp/products/purevariants/Danilo_Blog.html)

# pure::variants

## バリエーション管理を 支援するツール

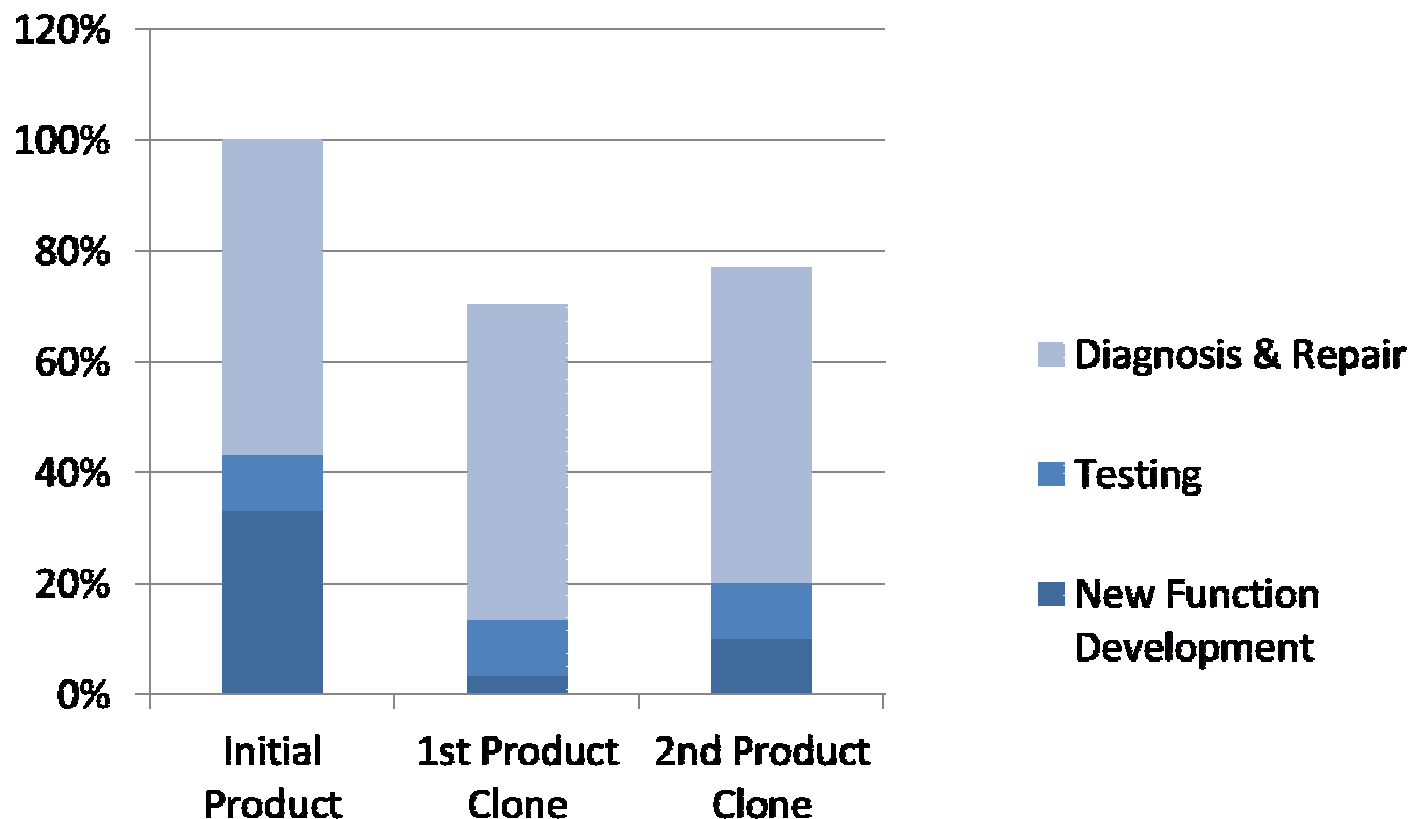




# 組込み機器を中心に広く採用



## 再利用開発の課題: コストが安くない



10% の新機能追加なのに → ~80% のコスト  
テスト・検証作業に多くの時間を要する





**再利用のアプローチ:  
既存ツールの組合わせ・調整  
だけでは、派生製品ファミリーの  
資産を上手く管理できない**





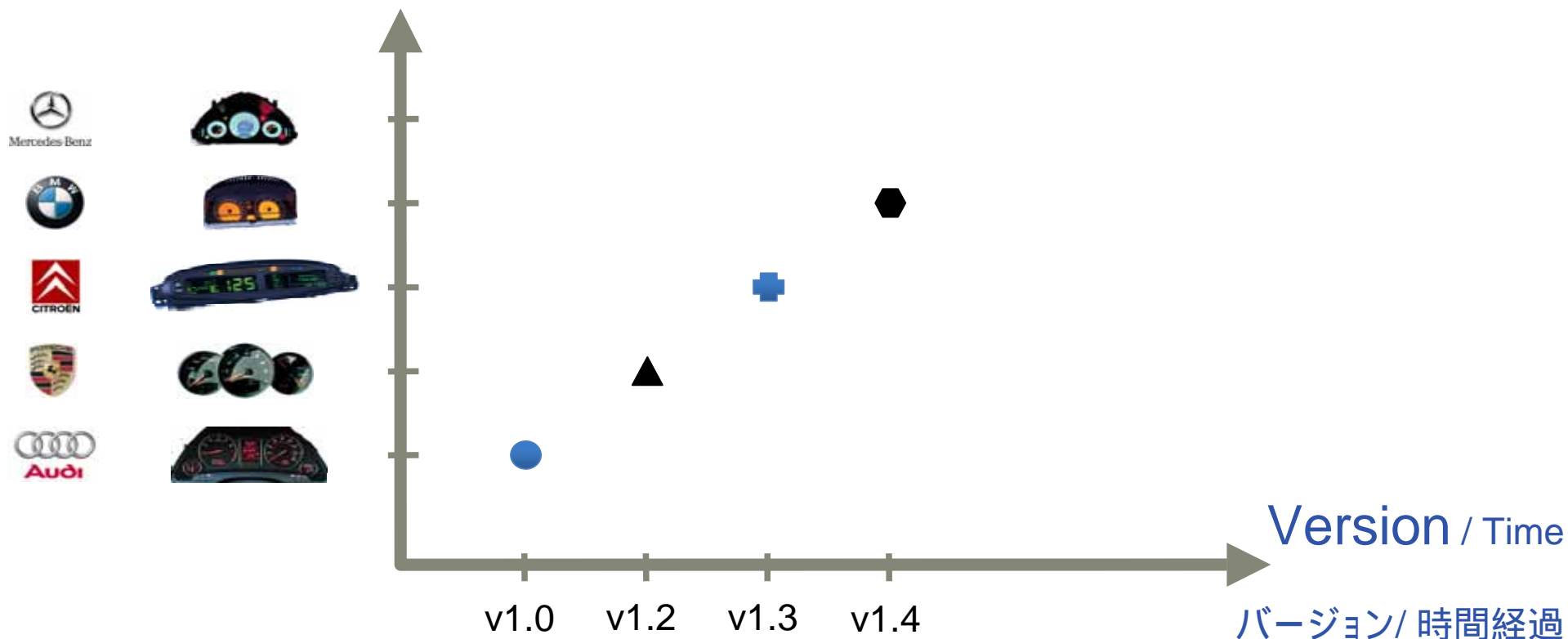




**Failing Reuse Approach:**  
サイロ型要件、設計、コード、テストなど  
ツールごとで資産が別々に管理される

# クローン & オウンの開発では

Variants  
バリエーション

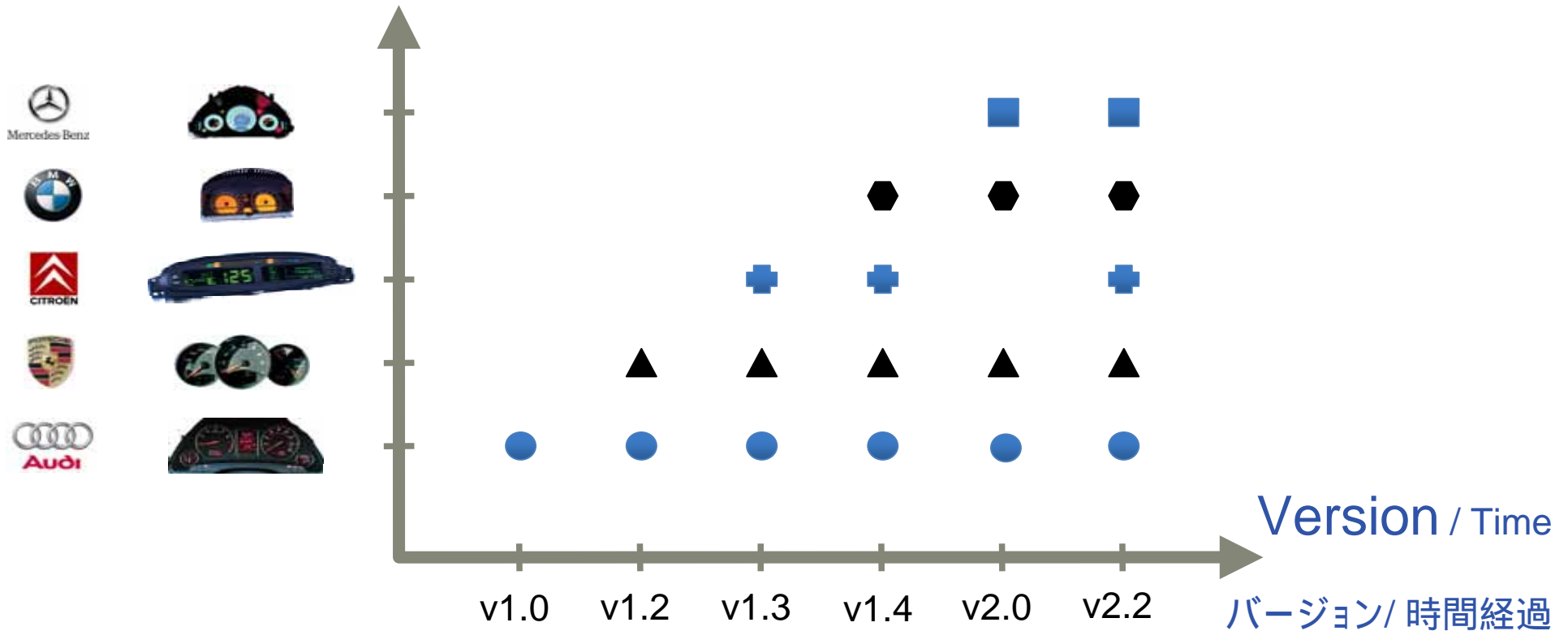


安くつくようでもテストやメンテナンスが軽減できない

# Separation of Variants and Versions

## バリエーションとバージョンの分離

Variants  
バリエーション



# Separation of Concern

関心事の分離 (バージョンとバリエーションの分離)

# Variation Point == First Class Citizen

バリエーションポイントを第一に

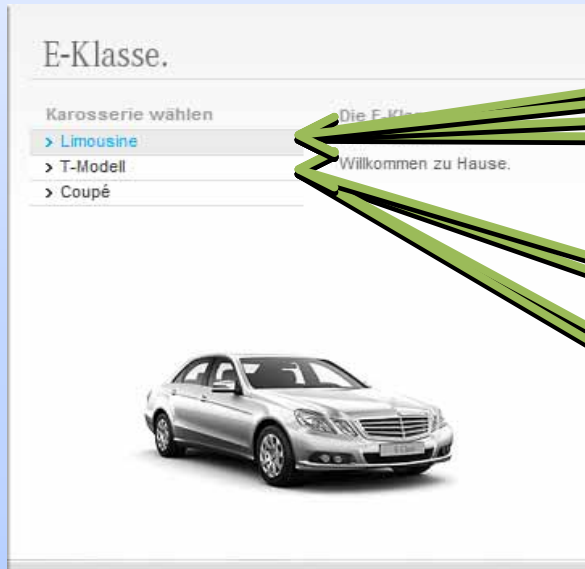
# Explicit Variant Management

系統だったバリエーションの管理

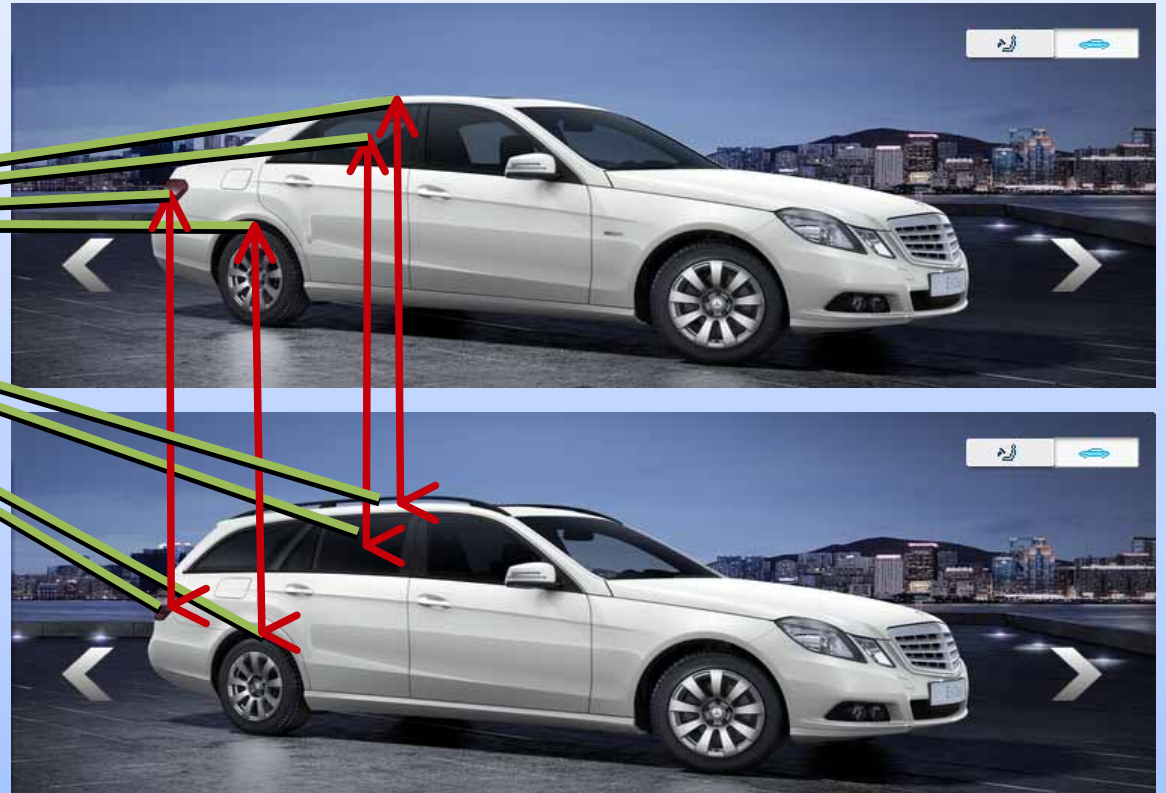


# Variation Points バリエーションポイント

## Problem Space 問題空間

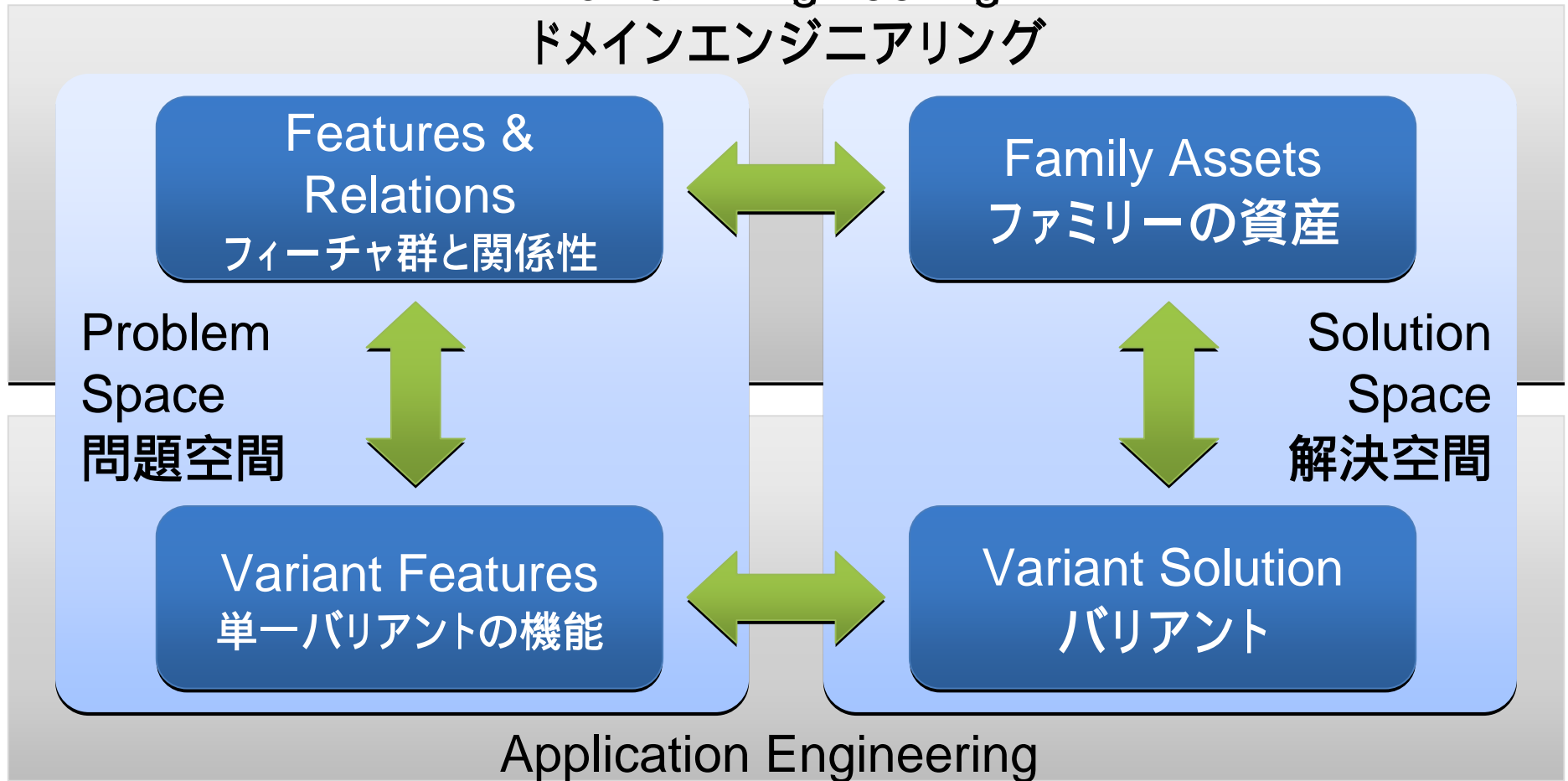


## Solution Space 解決空間



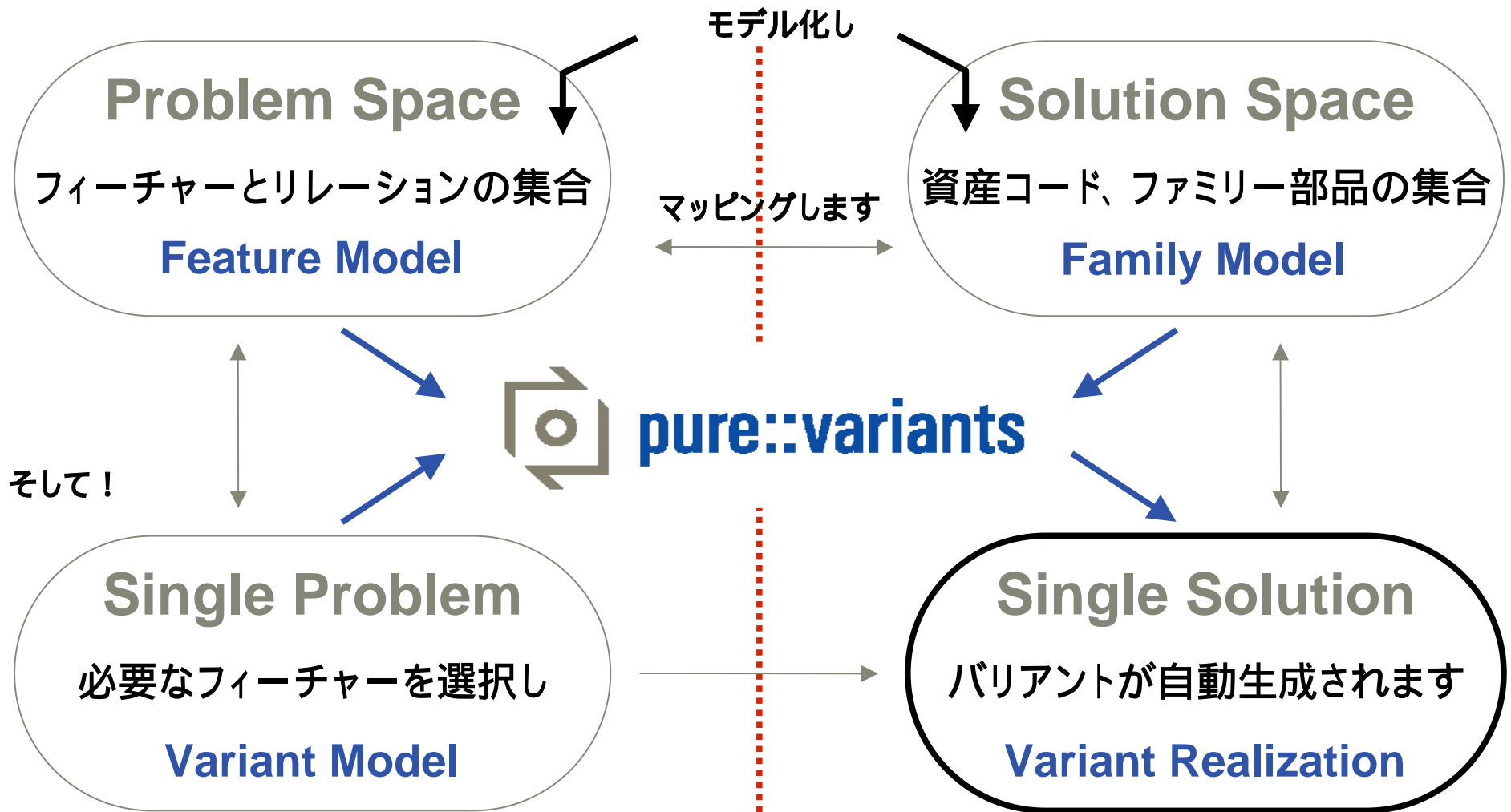
# Separation of Concern 関心事の分離

## Domain Engineering ドメインエンジニアリング



## Application Engineering アプリケーションエンジニアリング

# バリエント管理ツール



# バリエーション管理ツール

pure::variants

## 要件管理

MKS  
Requirements  
Borland Caliber  
RM  
Telelogic DOORS

## デザインツール

Simulink  
openArchitectureWare  
Rational SW. Architect  
*Rhapsody*  
*Artisan Studio*  
*Enterprise Architect.*

## 構成管理

MKS Source  
Rational  
ClearCase  
CVS  
Subversion  
...

## 変更・バグ管理

MKS Integrity  
Rational  
ClearQuest  
Atlassian JIRA  
Bugzilla  
Softtest

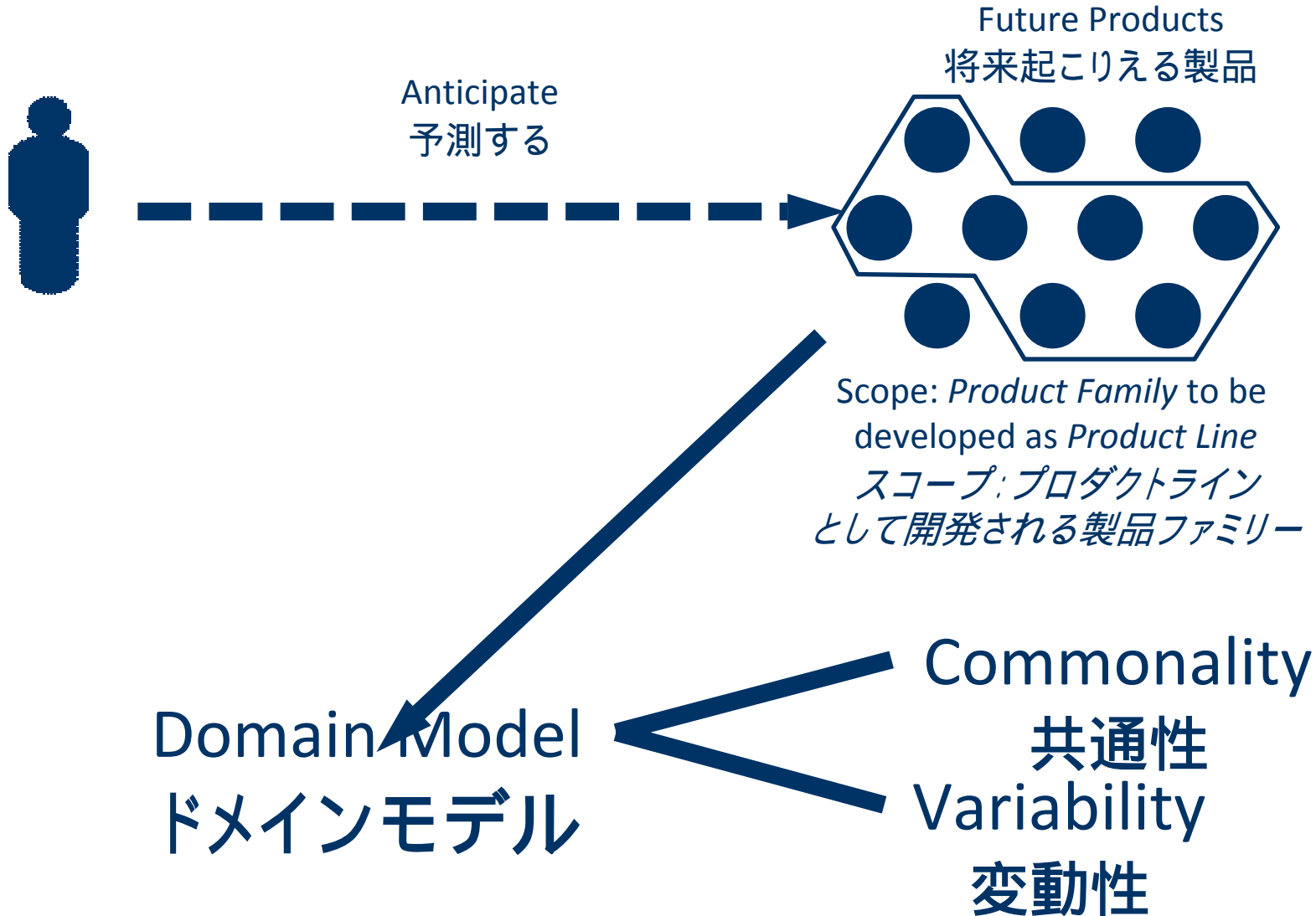
各種ツールを統合

プロダクトライン・ライフサイクル・マネジメントを支援



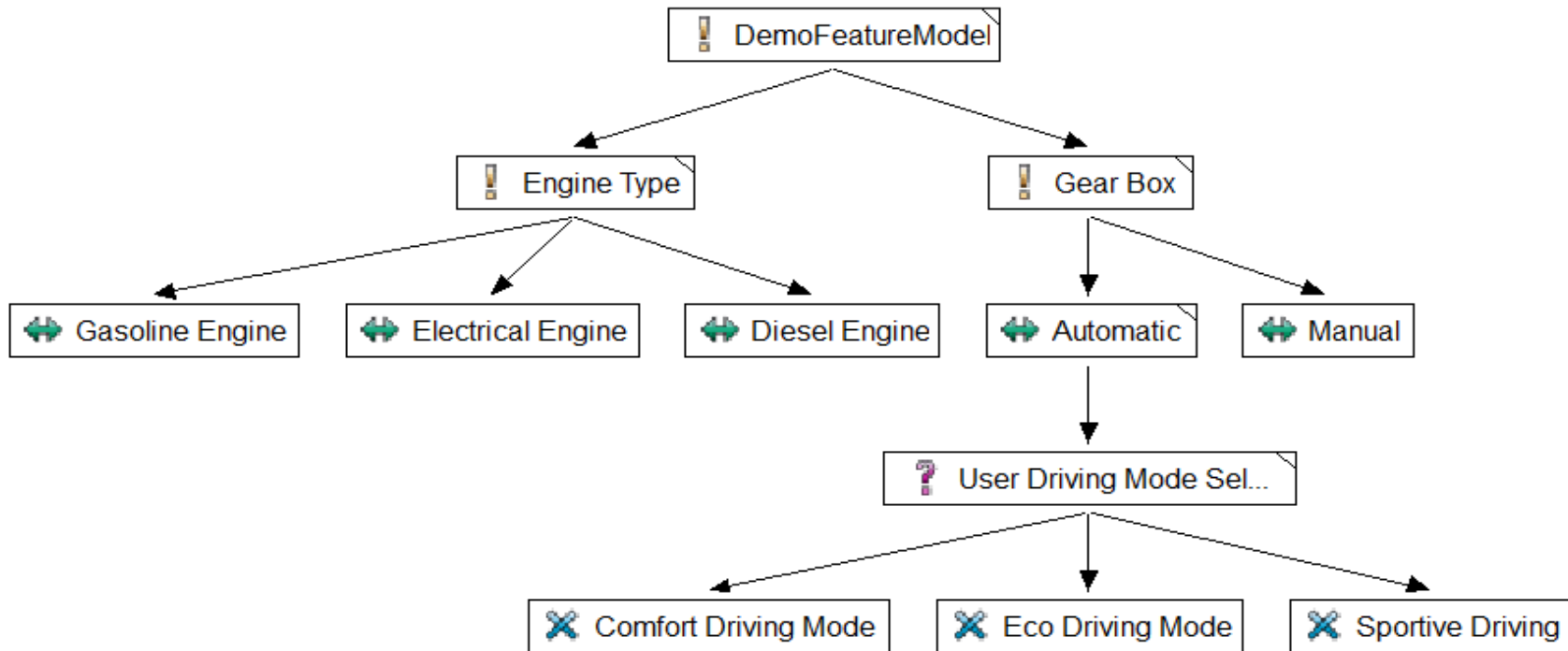
# Problem Space: Define Domain Model

## 問題空間: ドメインのモデルを定義



# Describing Variability Using Feature Models

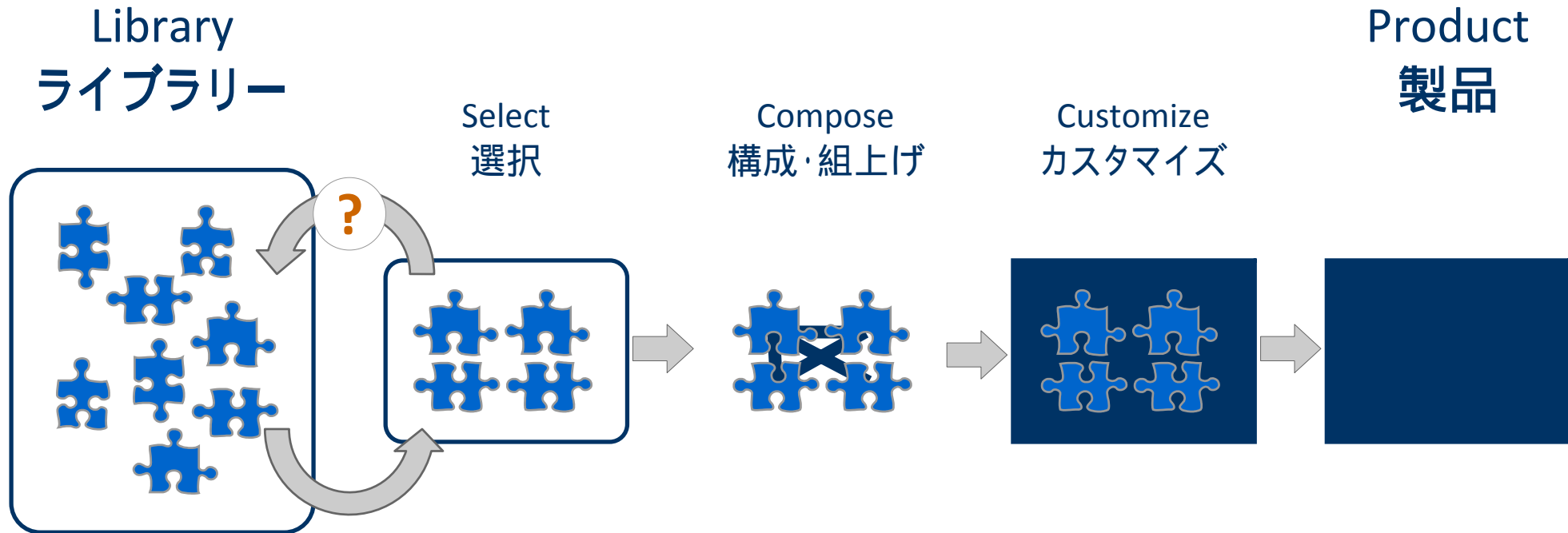
## バリエービリティをフィーチャモデルで記述



**Legend:** ! = Mandatory ? : Optional ↔ Alternative ⊗ Or

# Reuse: The Old Way

## 再利用法: 古いやり方 (ライブラリ)



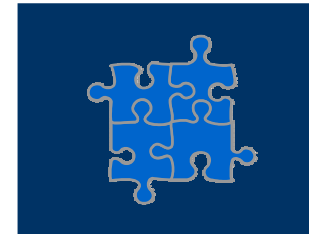
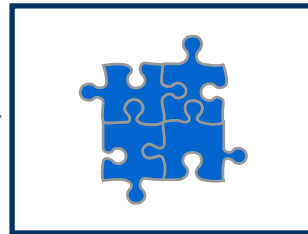
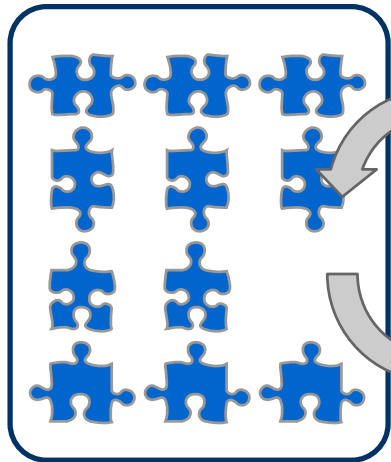
# Reuse: Product Lines 再利用: プロダクトライン

Platform /  
Core Assets  
プラットフォーム/  
コア資産

Select & Compose  
選択して構成、組上げ

Customize  
カスタマイズ

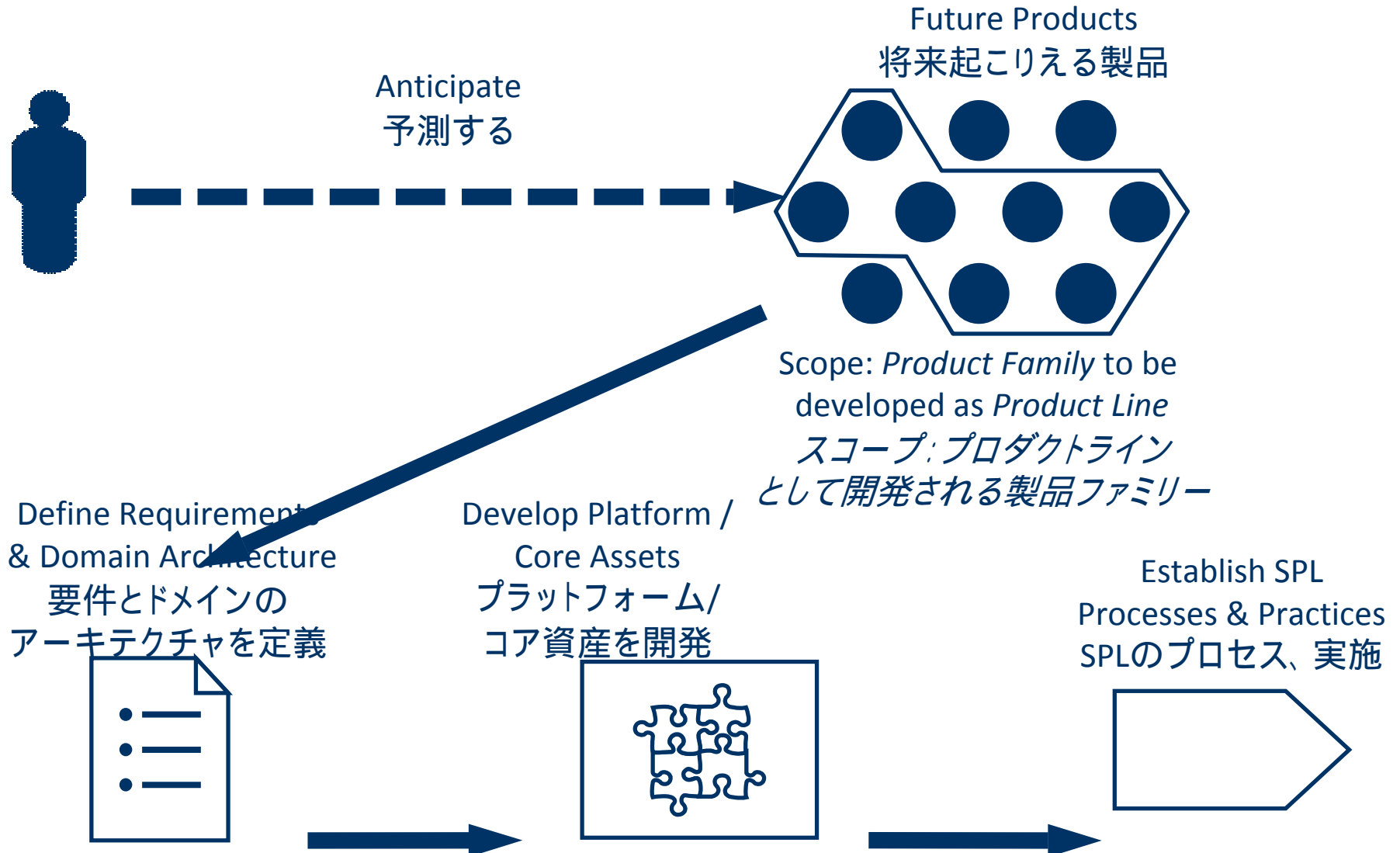
Product  
製品



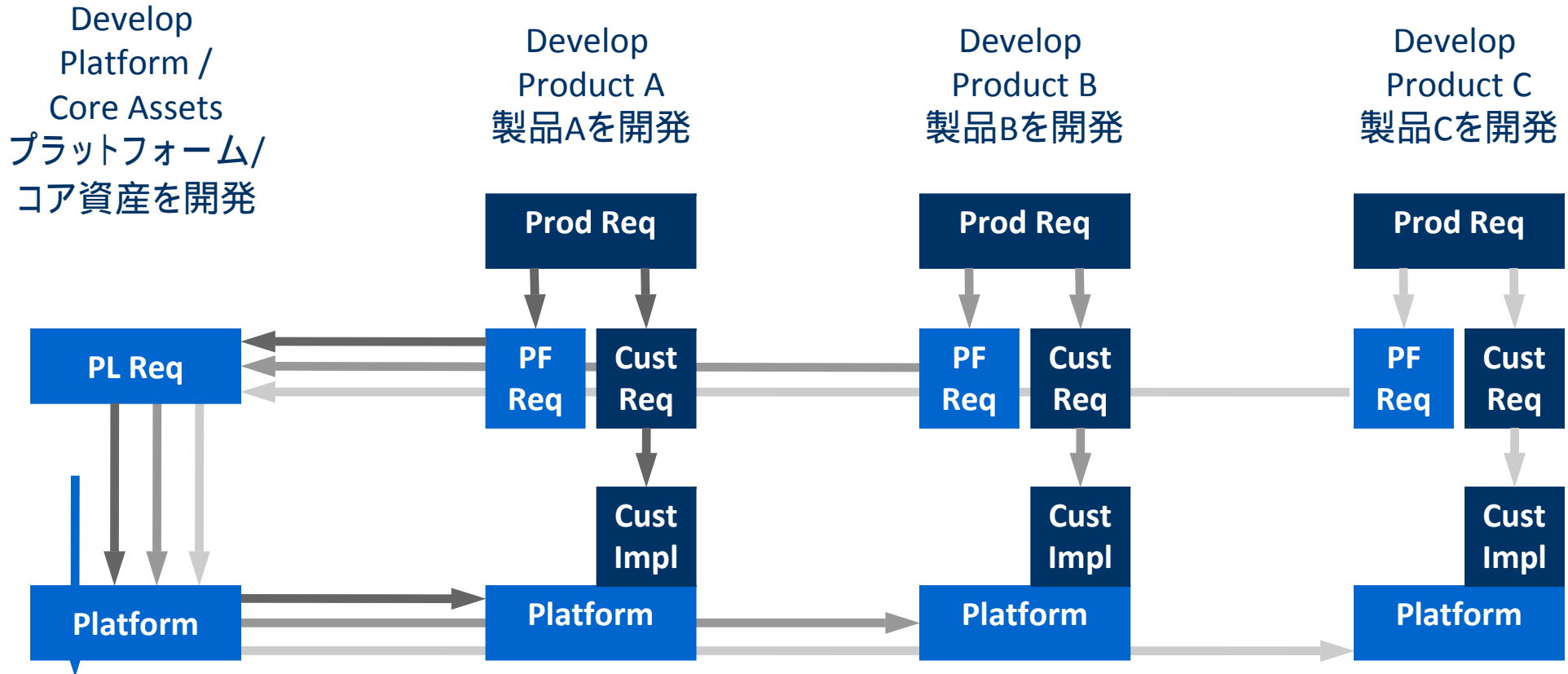


# Setting up a Product Line

## プロダクトラインの構築



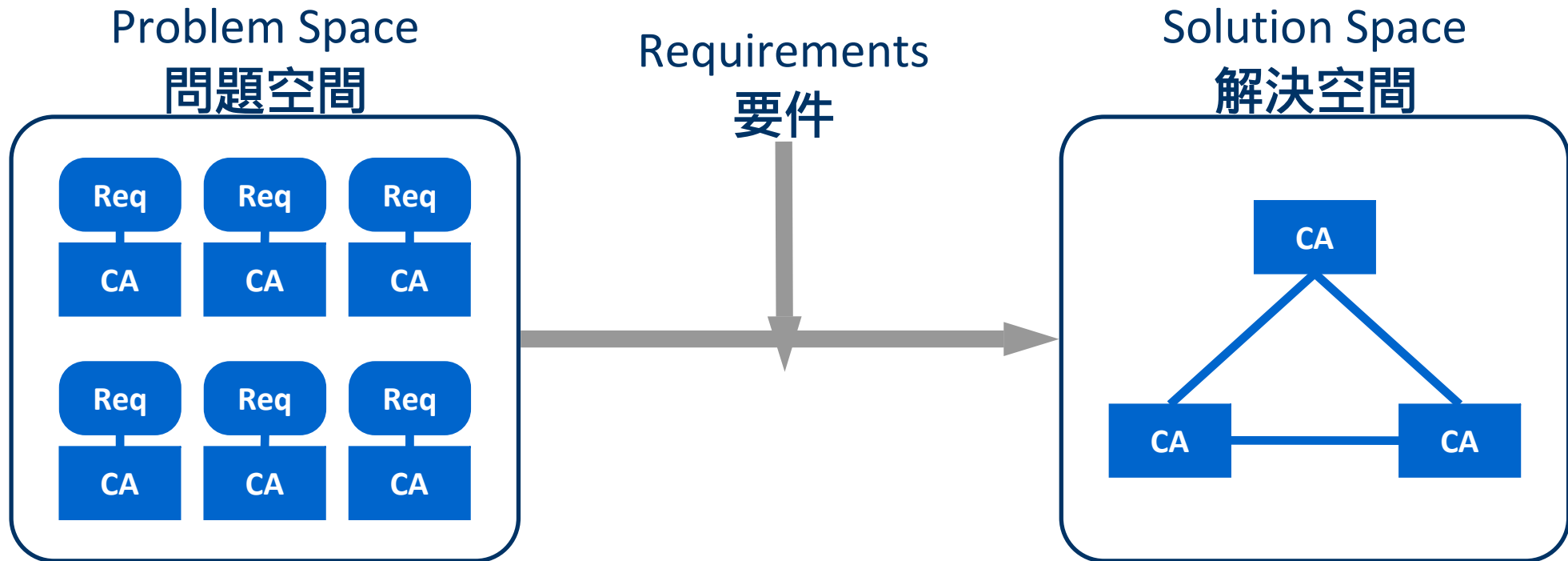
# Operating a Product Line (製品個別で再利用されないものもある)



**Legend:** PL = Product Line; PF = Platform; Prod = Product; Cust = Custom;  
Req = Requirement; Impl = Implementation

# Development with Reuse: Functional Mapping

## 再利用による開発: 機能のマッピング



$f : ( \text{Req}, ( \text{Req} \times \text{Core Asset} ) )$

Reusable Solution  
再利用可能なソリューション

**Legend:** Req = Requirement; CA = Core Asset

Reuse Items  
再利用するアイテム コードだけ？

Test Cases

Architecture

Estimates

Models

**Code**

Requirements

Subsystems

Documentation

Rules



Reuse Items  
再利用するアイテム 全ての成果物！

Test Cases

Architecture

Estimates

Models

Code

Requirements

Subsystems

Documentation

Rules

Reuse Items  
再利用するアイテム 要件について

Test Cases

Architecture

Estimates

Models

Code

**Requirements**

Subsystems

Documentation

Rules

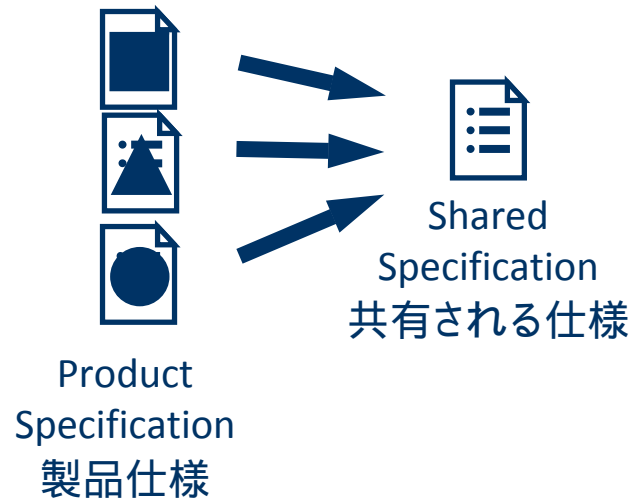
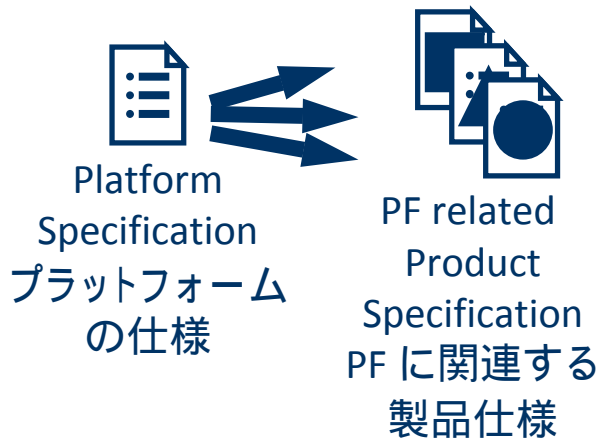
# Scenarios (3)

## Specification Approach 仕様のアプローチ

Derived 派生

Shared 共有

Individual 個別

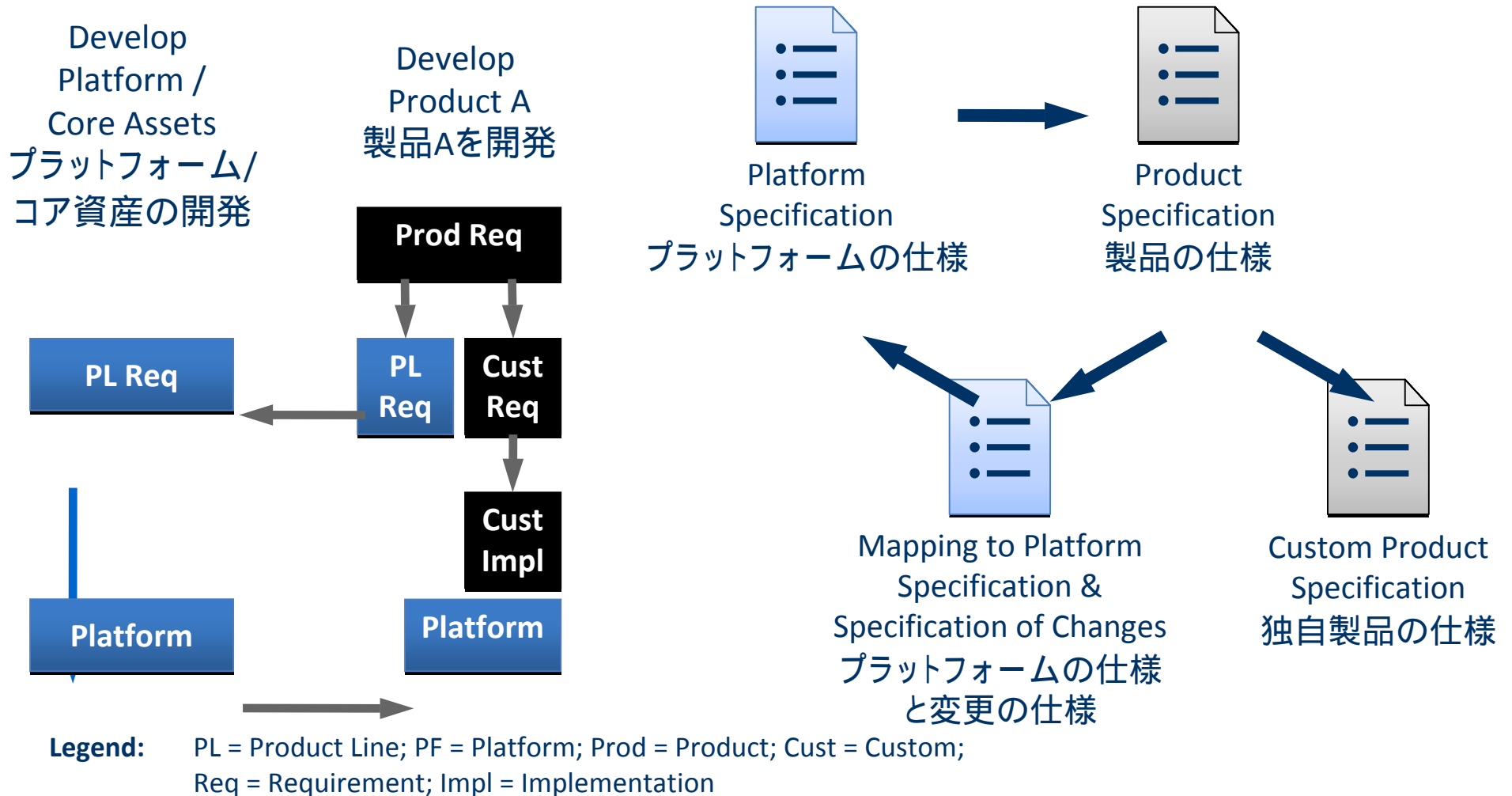


# Exercise

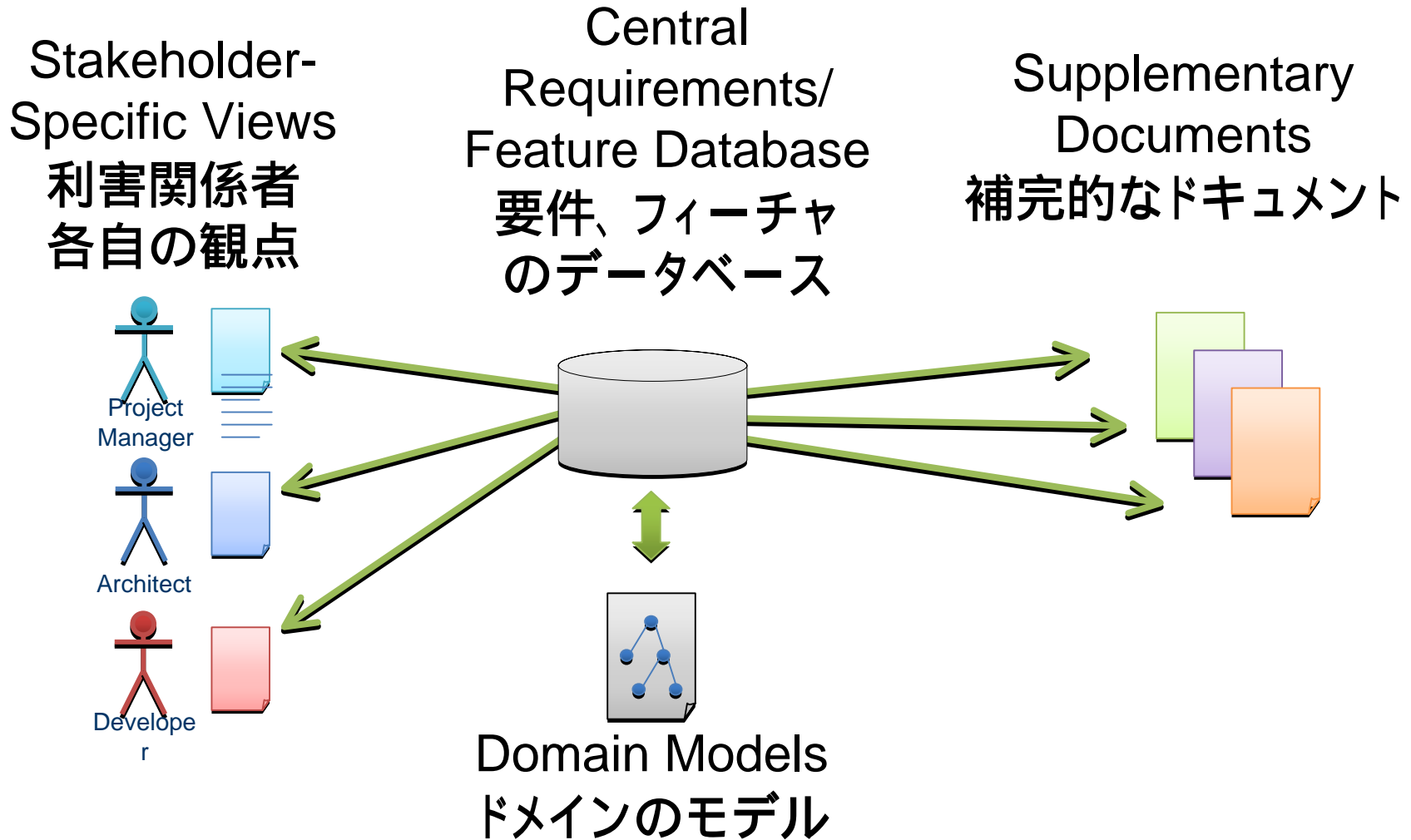
Describe your organization's approach!



# Requirements Specification 要求仕様



# Continuous Requirements Management 継続的な要件管理



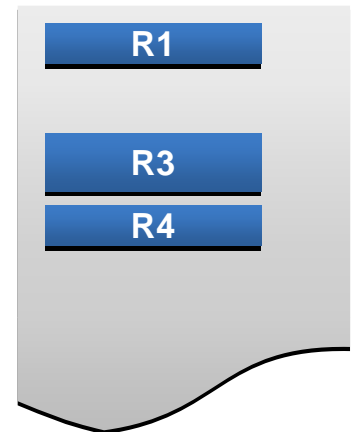
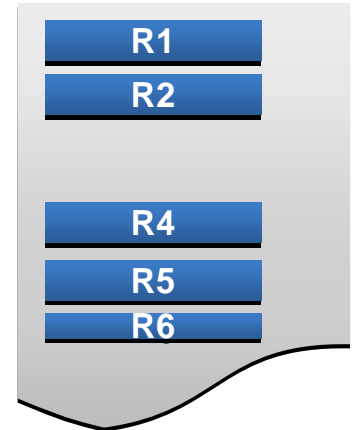
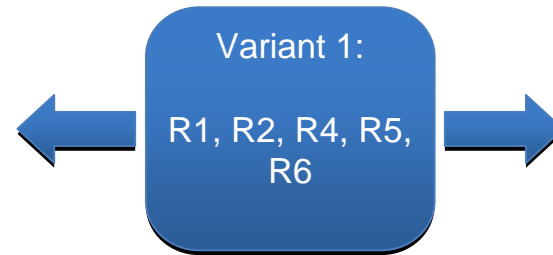
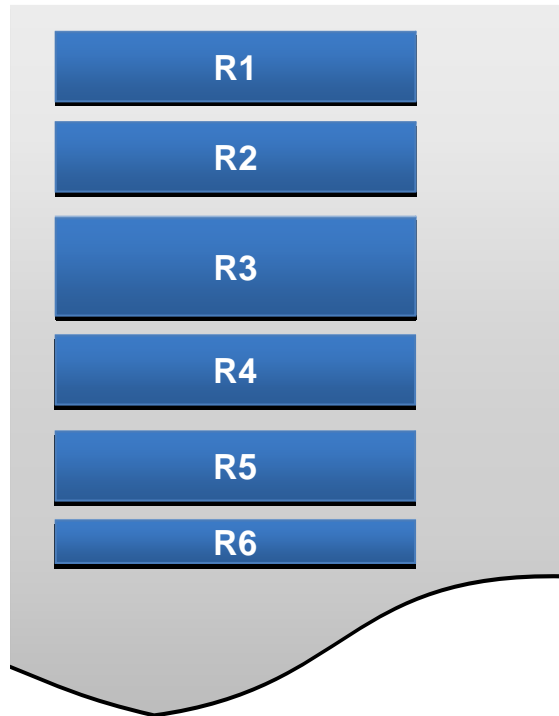


Requirement Variability

Modelling Concepts in RM Tools

要件のバリエービリティのモデリング: 要件管理ツール

# Requirement Selection 要件の選択



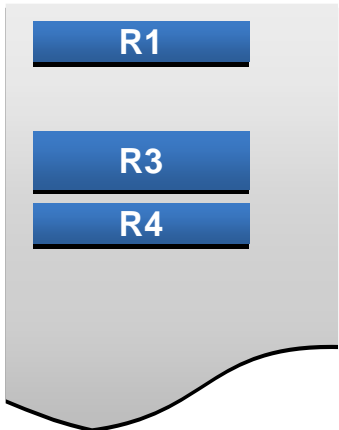
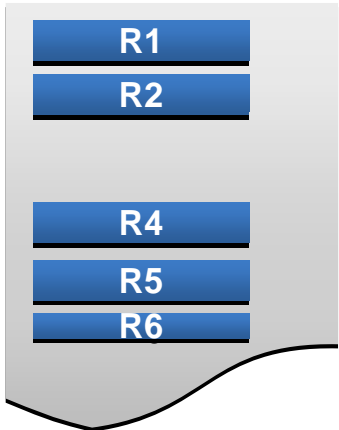
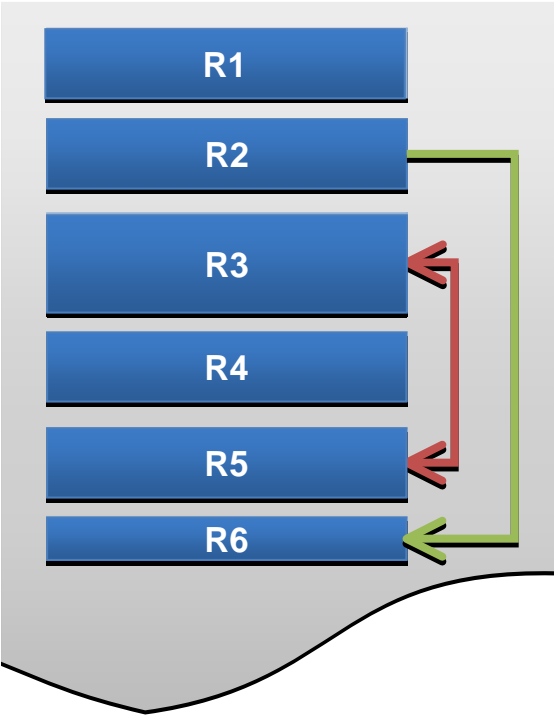
# Requirement Selection

## 要件の選択

The screenshot shows the DOORS software interface for a project named 'DemoModule'. The main window displays a list of requirements (R1 to R6) and their associated variants. The 'Variants' column is highlighted with a red box, indicating the selected variants for each requirement.

ID	Requirement	Variants
R1	1 Control Unit Board	Variant1, Variant2
R2	2 Driving Mode Switch	Variant1
R3	3 Overheat Control	Variant2
R4	> 4 Basic Software Requirements	Variant1, Variant2
R5	5 Overheat Control	Variant1
R6	6 Start-Stop Automatic	Variant1

# Dependency Rules 依存関係のルール



# Dependency Rules 依存関係のルール

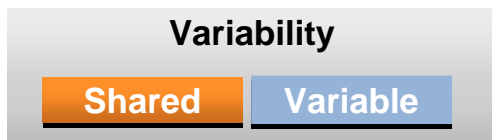
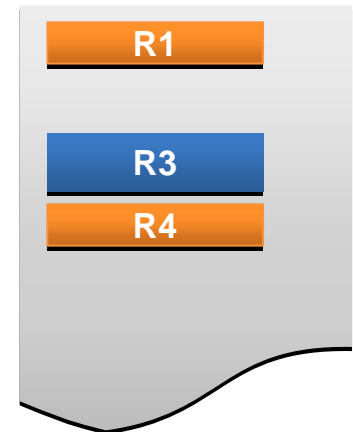
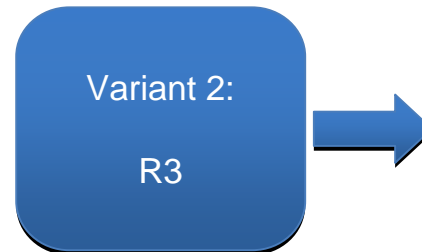
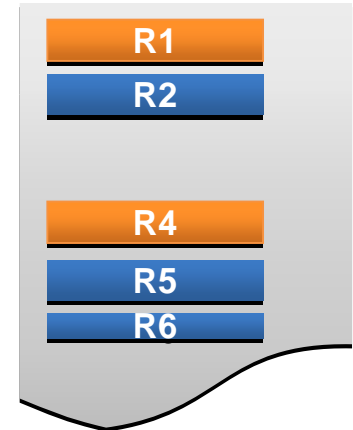
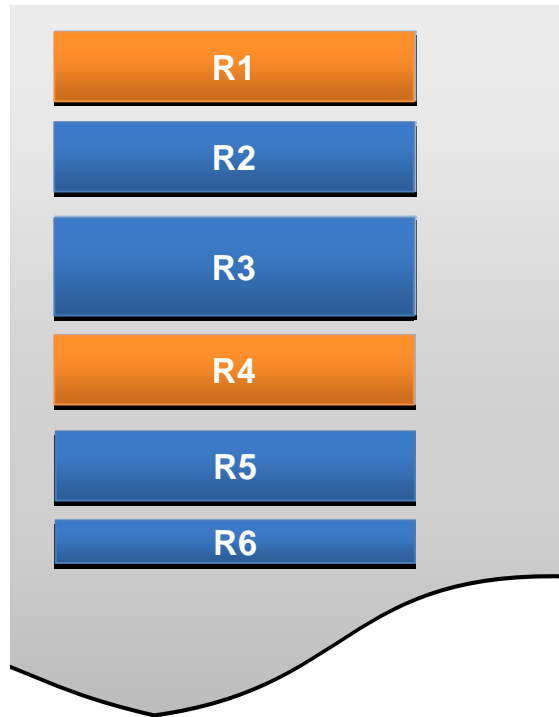
The screenshot shows the DOORS software interface for a module named 'DemoModule'. The main window displays a table of dependency rules. The table has three columns: 'ID', 'Variants', and 'pvConstraint'. The 'pvConstraint' column is highlighted with a red box, showing constraints for rules R2 and R3.

ID	Variants	pvConstraint
R1	1 Control Unit Board Variant1, Variant2	
R2	2 Driving Mode Switch Variant1	SELF requires R6
R3	3 Overheat Control Variant2	SELF conflicts R5
R4	> 4 Basic Software Requirements Variant1, Variant2	
R5	5 Overheat Control Variant1	
R6	6 Start-Stop Automatic Variant1	

Username: pv      Exclusive edit mode

# Shared / Variable Requirement Selection

## 共有/ 変動要件の選択





# Shared / Variable Requirement Selection

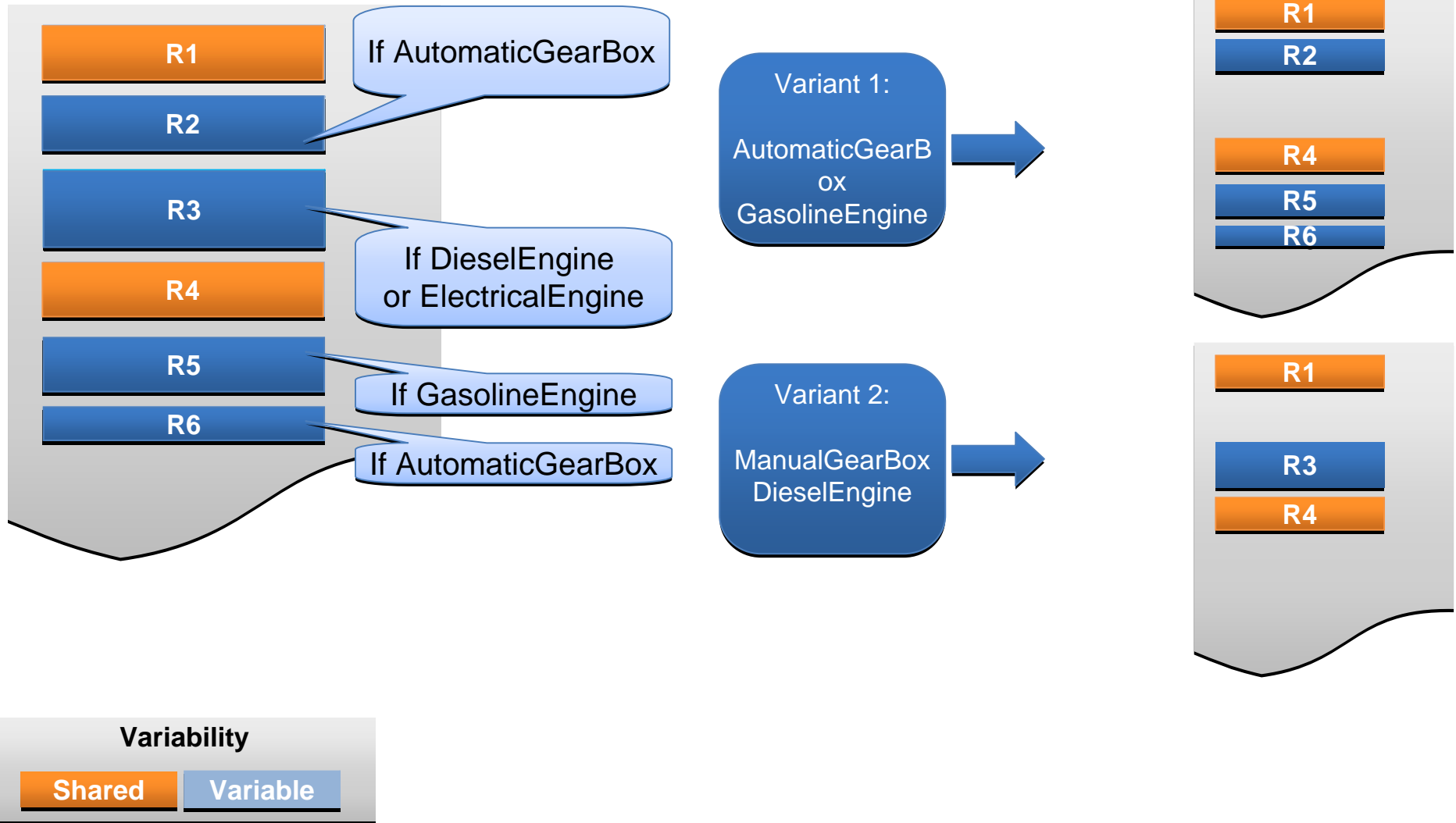
## 共有/ 変動要件の選択

The screenshot shows a software application window titled "'DemoModule' current 0.0 in /DoorsDemo (Formal module) - DOORS". The interface includes a menu bar (File, Edit, View, Insert, Link, Analysis, Table, Tools, Discussions, User, Help), a toolbar, and a main workspace. On the left, a tree view shows the project structure under 'DemoModule', with '4 Basic Software Requirements' selected. The main workspace displays a table of requirements with columns for ID, Name, Variants, and pvElement Type. A red box highlights the 'pvElement Type' column.

ID	Name	Variants	pvElement Type
R1	<b>1 Control Unit Board</b>	Variant1, Variant2	mandatory
R2	<b>2 Driving Mode Switch</b>	Variant1	optional
R3	<b>3 Overheat Control</b>	Variant2	optional
R4	<b>&gt; 4 Basic Software Requirements</b>	Variant1, Variant2	mandatory
R5	<b>5 Overheat Control</b>	Variant1	optional
R6	<b>6 Start-Stop Automatic</b>	Variant1	optional

Username: pv      Exclusive edit mode

# Selection Rules 選択のルール



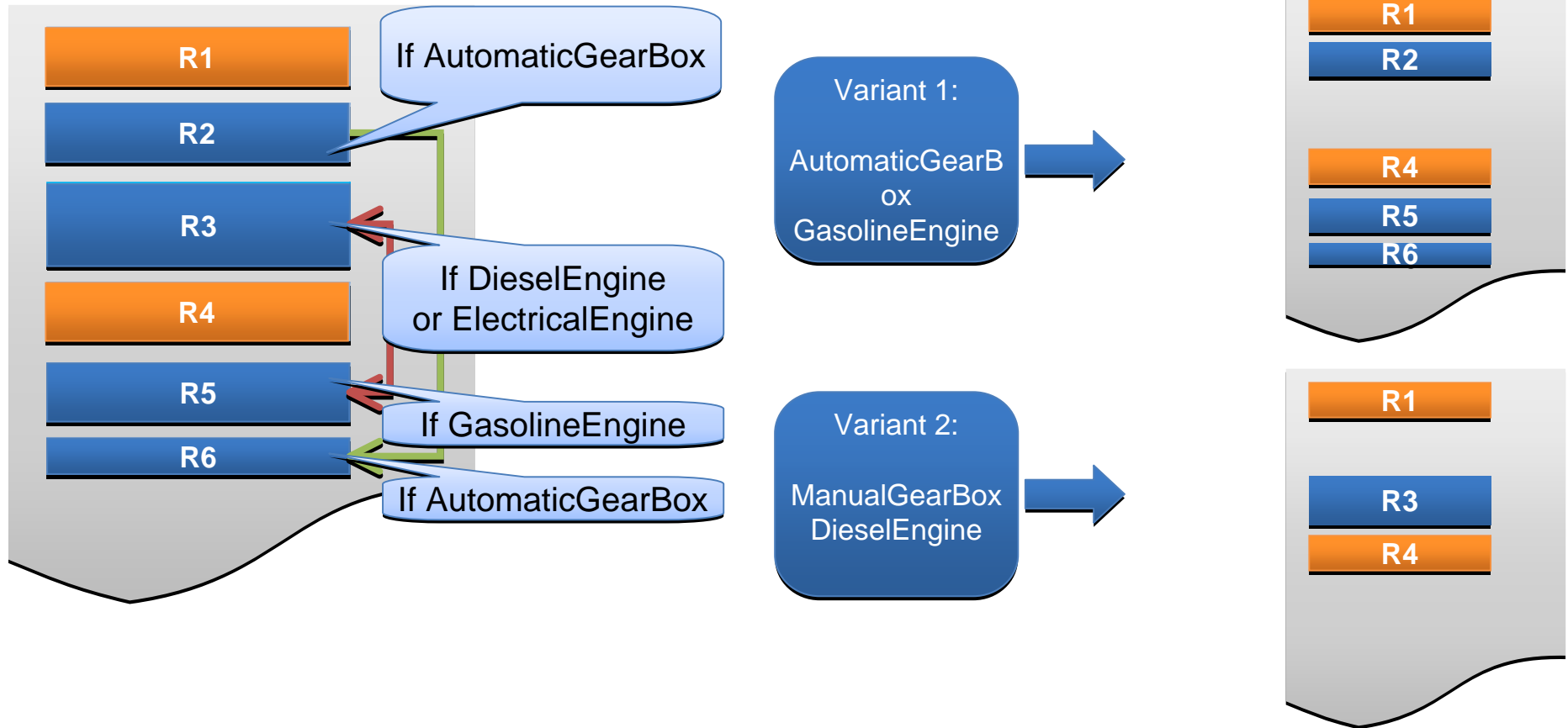
# Selection Rules 選択のルール

The screenshot displays the DOORS software interface for a project named 'DemoModule'. The main window shows a table of selection rules (R1-R6) with columns for ID, Name, Variants, and pvRestriction. A red box highlights the 'pvRestriction' column for rules R2, R3, and R6.

ID	Name	Variants	pvRestriction
R1	<b>1 Control Unit Board</b>	Variant1, Variant2	
R2	<b>2 Driving Mode Switch</b>	Variant1	AutomaticGearBox
R3	<b>3 Overheat Control</b>	Variant2	DieselEngine or ElectricalEngine
R4	<b>&gt; 4 Basic Software Requirements</b>	Variant1, Variant2	
R5	<b>5 Overheat Control</b>	Variant1	GasolineEngine
R6	<b>6 Start-Stop Automatic</b>	Variant1	AutomaticGearBox

Additional interface details include a menu bar (File, Edit, View, Insert, Link, Analysis, Table, Tools, Discussions, User, Help), a toolbar, a left-hand tree view showing the module structure, and a status bar at the bottom indicating 'Username: pv' and 'Exclusive edit mode'.

# All Combined 全てを融合



# All Combined 全てを融合

The screenshot shows the DOORS software interface for a project named 'DemoModule'. The main window displays a table of requirements with columns for ID, Name, Variants, pvRestriction, and pvConstraint. A red box highlights the 'pvRestriction' and 'pvConstraint' columns for requirements R2, R3, and R5.

ID		Variants	pvRestriction	pvConstraint
R1	<b>1 Control Unit Board</b>	Variant1, Variant2		
R2	<b>2 Driving Mode Switch</b>	Variant1	AutomaticGearBox	SELF requires R6
R3	<b>3 Overheat Control</b>	Variant2	DieselEngine or ElectricalEngine	SELF conflicts R5
R4	<b>&gt; 4 Basic Software Requirements</b>	Variant1, Variant2		
R5	<b>5 Overheat Control</b>	Variant1	GasolineEngine	
R6	<b>6 Start-Stop Automatic</b>	Variant1		

Username: pv      Exclusive edit mode

# Requirement Variation Point Modelling

## 要件上のバリエーションポイントのモデリング

Directly in RE tool

要件管理ツールに直接

Pro:

- Variability defined with requirements  
要件にバリエービリティを定義できる

Con:

- Changed RE information model  
REの情報モデルが変更されてしまう
- Missing link to other variability models  
他のバリエービリティモデルへのリンクの欠如

In external tool

外部ツール

Pro:

- Unchanged RE information model  
REの情報モデルが変更されない
- Uniform variability modelling  
バリエービリティモデリングを単一化

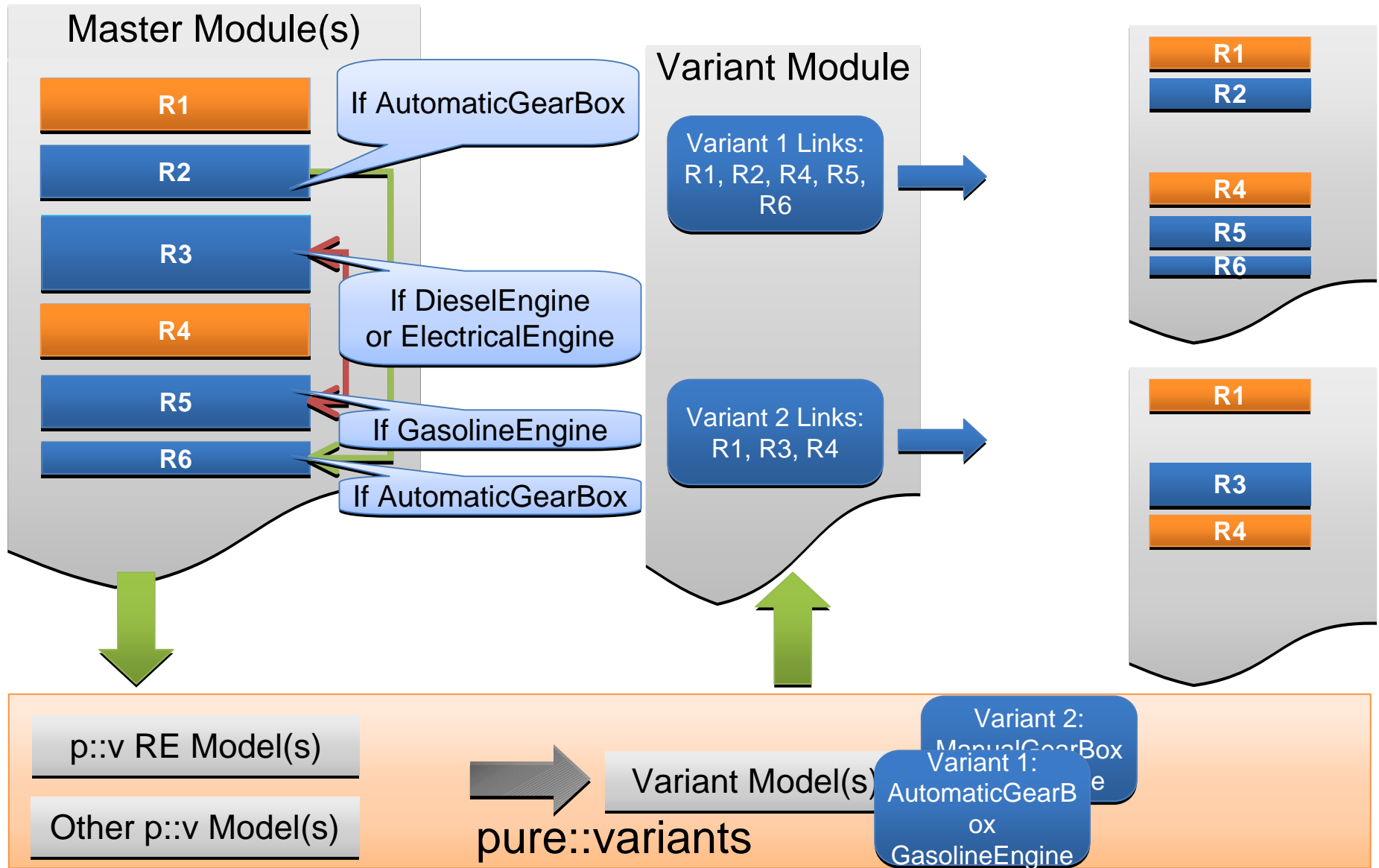
Con:

- Consistency Issues  
一貫性・整合性の課題
- Variability not visible in RE tool  
バリエービリティが要件管理ツールで見えない

# pure::variants for Requirements



# Best of Both – pure::variants for Requirements

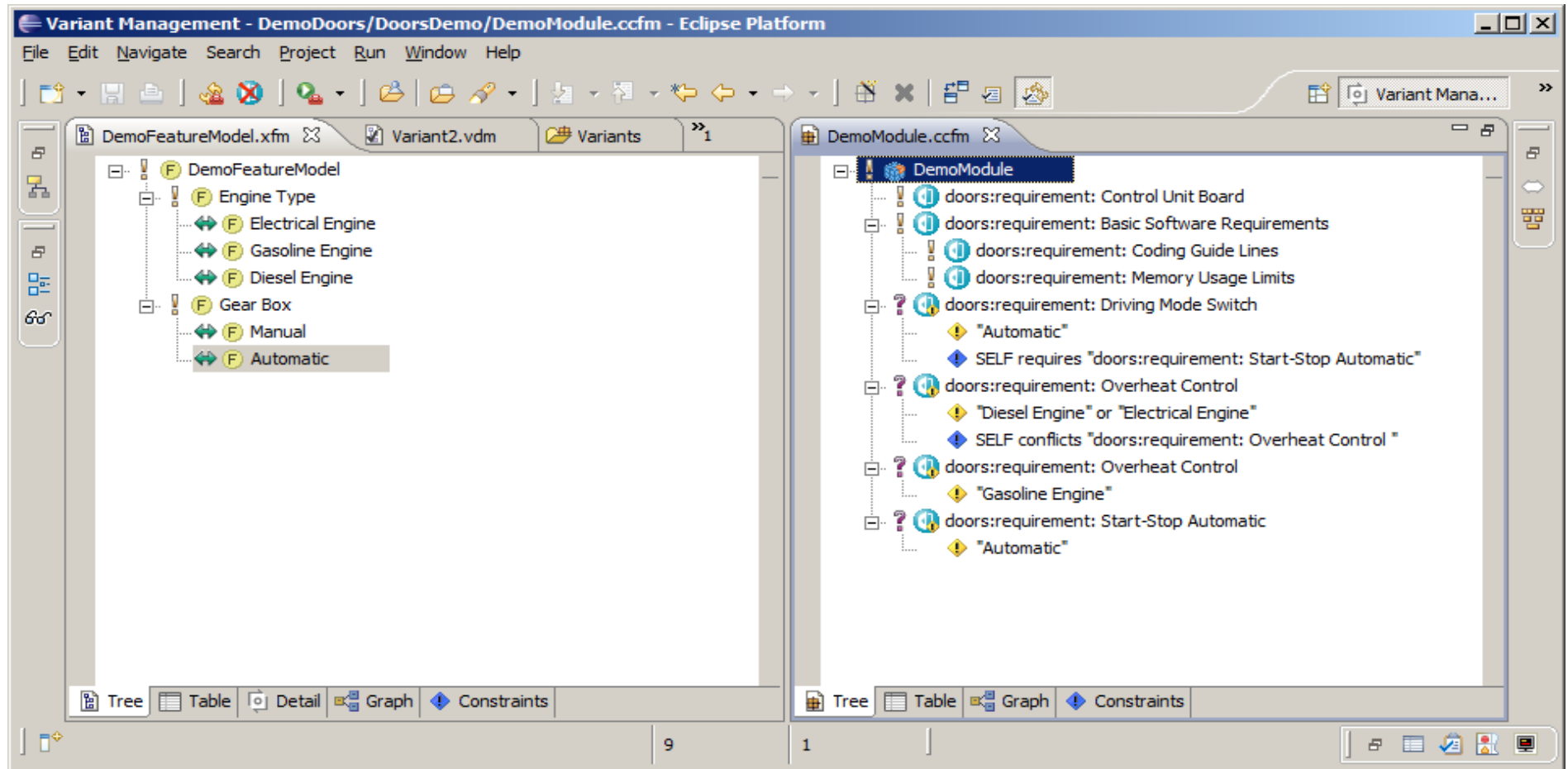


# pure::variants Solution – Variant Matrix

The screenshot shows the Eclipse Platform Variant Management interface. The main window displays a variant matrix for two models: DemoFeatureModel and DemoModule. The matrix columns are Model Elements, Level, Variant1, and Variant2. The rows list various features and requirements with their respective levels and variant availability.

Model Elements	Level	Variant1	Variant2
<b>Model Elements</b>			
<b>[-] DemoFeatureModel</b>			
DemoFeatureModel		✓	✓
Engine Type	1	✓	✓
Electrical Engine	1.1	✗	✗
Gasoline Engine	1.2	✓	✗
Diesel Engine	1.3	✗	✓
Gear Box	2	✓	✓
Manual	2.1	✗	✓
Automatic	2.2	✓	✗
<b>[-] DemoModule</b>			
DemoModule		✓	✓
doors:requirement: Control Unit Board	1	✓	✓
doors:requirement: Basic Software Requirements	2	✓	✓
doors:requirement: Coding Guide Lines	2.1	✓	✓
doors:requirement: Memory Usage Limits	2.2	✓	✓
doors:requirement: Driving Mode Switch	3	✓	✗
doors:requirement: Overheat Control	4	✗	✓
doors:requirement: Overheat Control	5	✓	✗
doors:requirement: Start-Stop Automatic	6	✓	✗

# pure::variants Solution – Models



# pure::variants Solution – Configuration

The screenshot displays the Eclipse IDE interface for Variant Management. The main editor shows a feature model tree for 'DemoFeatureModel' with the following structure:

- ✓ DemoFeatureModel
  - ✓ Engine Type
    - ✗ Electrical Engine
    - ✓ Gasoline Engine (selected)
    - ✗ Diesel Engine
  - ✓ Gear Box
    - ✓ Manual
    - ✗ Automatic

The 'Relations' view on the right shows a dependency graph:

- Parent (1)
  - Engine Type
    - Simple Constraint Language (1)
      - Is Referenced By (1)
        - doors:requirement: Overheat Control

The bottom console displays an error message:

```
1 error, 0 warnings, 0 others
Description
Errors (1 item)
  ✗ open alternatives are 'Electrical Engine', 'Gasoline Engine', 'Diesel Engine'
```

Description	Resource	Path	Location
✗ open alternatives are 'Electrical Engine', 'Gasoline Engine', 'Diesel Engine'	Variant2.vdm	DemoDoors/Varia...	Electric

Reuse Items

Test Cases

Estimates  
**Architecture**

**Models**

Code

Requirements

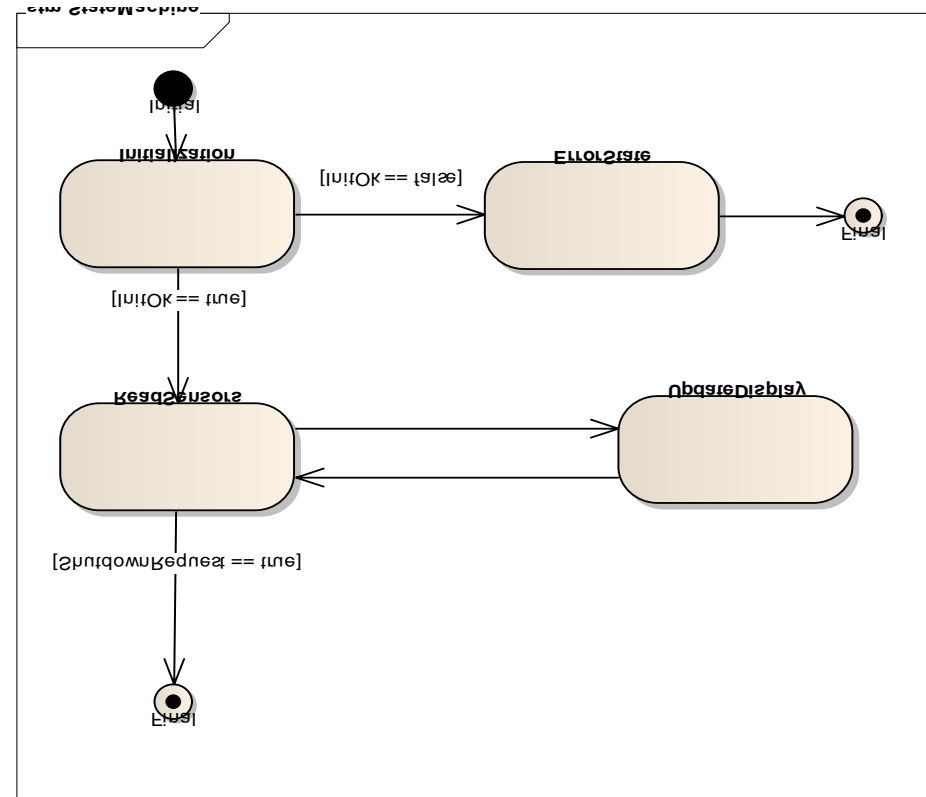
Subsystems

Documentation

Rules

# Complexity Reduction

```
1  typedef enum states { INIT, FINAL, Initialization,  
2                          ReadSensor, UpdateDisplay, ErrorState  
3                          } state_t;  
4  
5  state_t state;  
6  int main(int argc, char[]* argv)  
7  {  
8      state = INIT;  
9      while (state != FINAL)  
10     {  
11         case INIT:  
12             app_init();  
13             state = Initialization;  
14             break;  
15         case Initialization:  
16             sensor_init();  
17             display_init();  
18             if (InitOK == 1) {  
19                 state = ReadSensors;  
20             } else {  
21                 state = ErrorState;  
22             }  
23             break;  
24         case ReadSensors:  
25             read_sensors();  
26             state = UpdateDisplay;  
27             if (ShutdownRequest == 1) {  
28                 state = FINAL;  
29             }  
30             break;  
31         case UpdateDisplay:  
32             update_display();  
33             state = ReadSensors;  
34             break;  
35         case ErrorState:  
36             log_errorstate();  
37             state = FINAL;  
38             break;  
39     }  
40     app_shutdown();  
41 }  
42
```

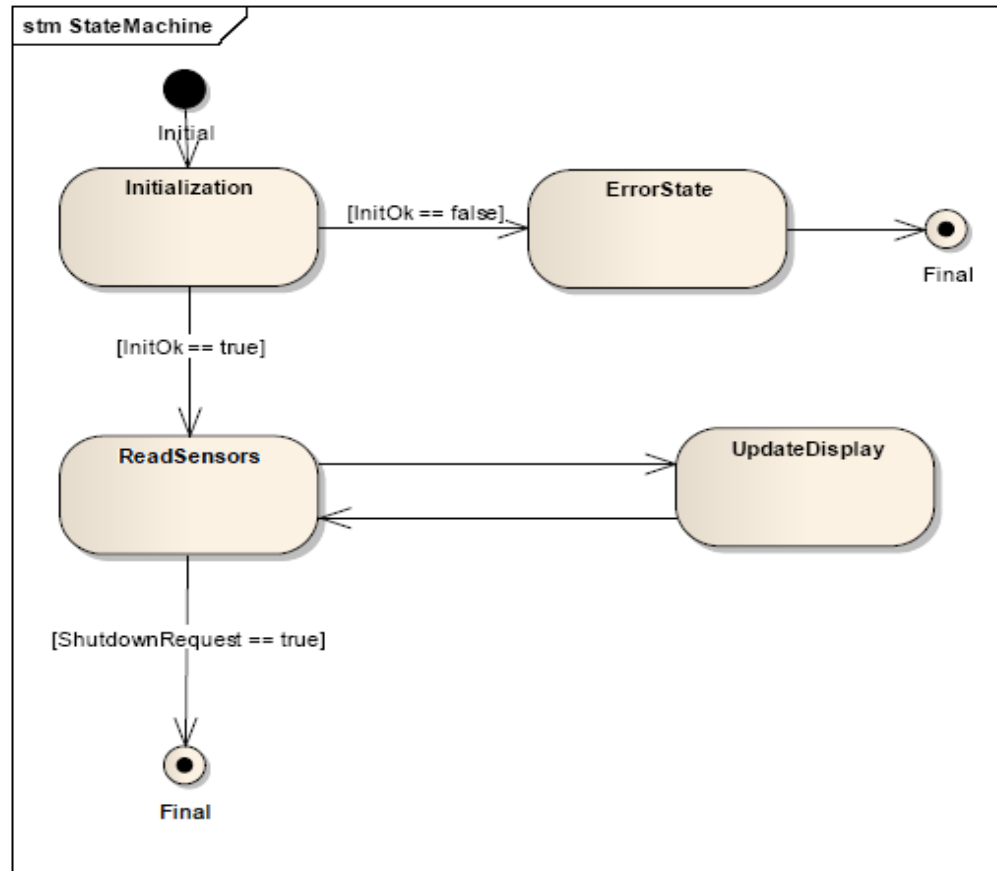


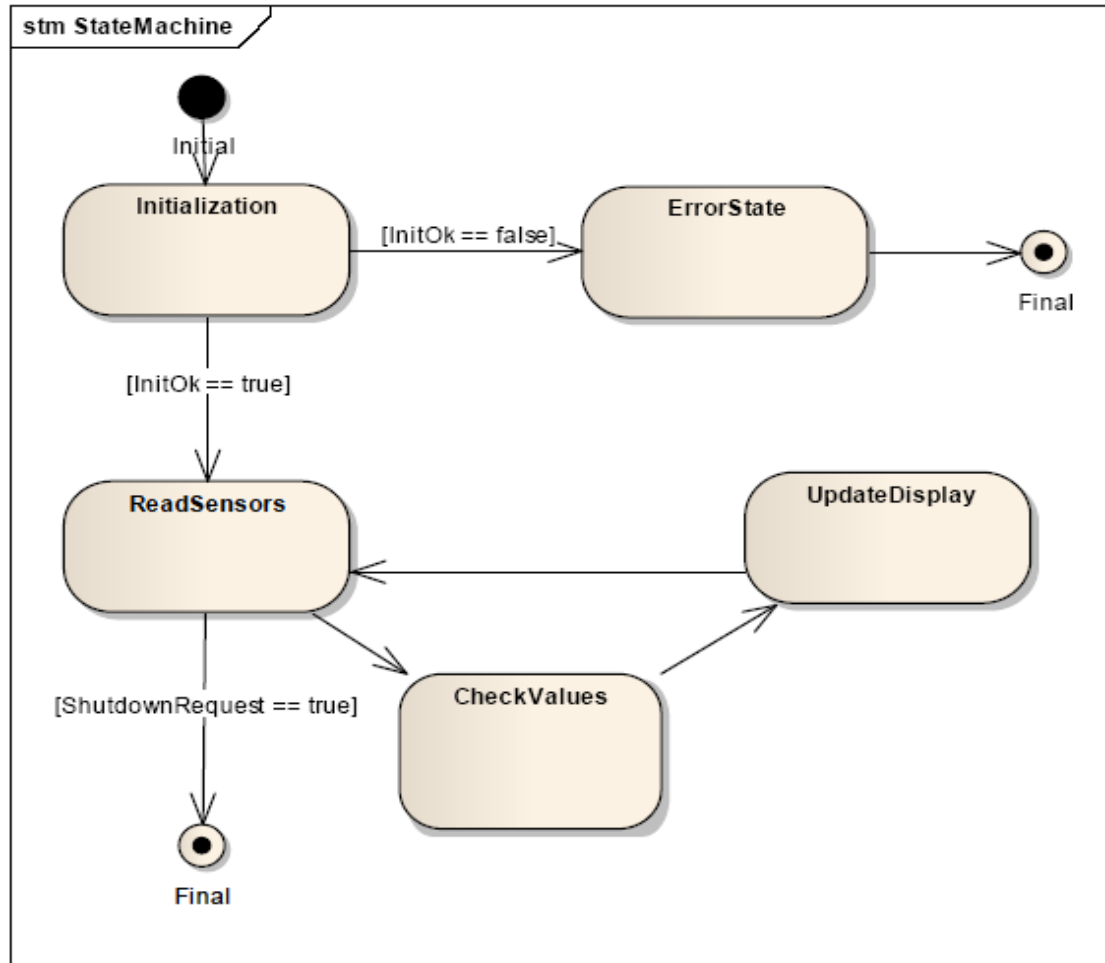




# Variability in UML

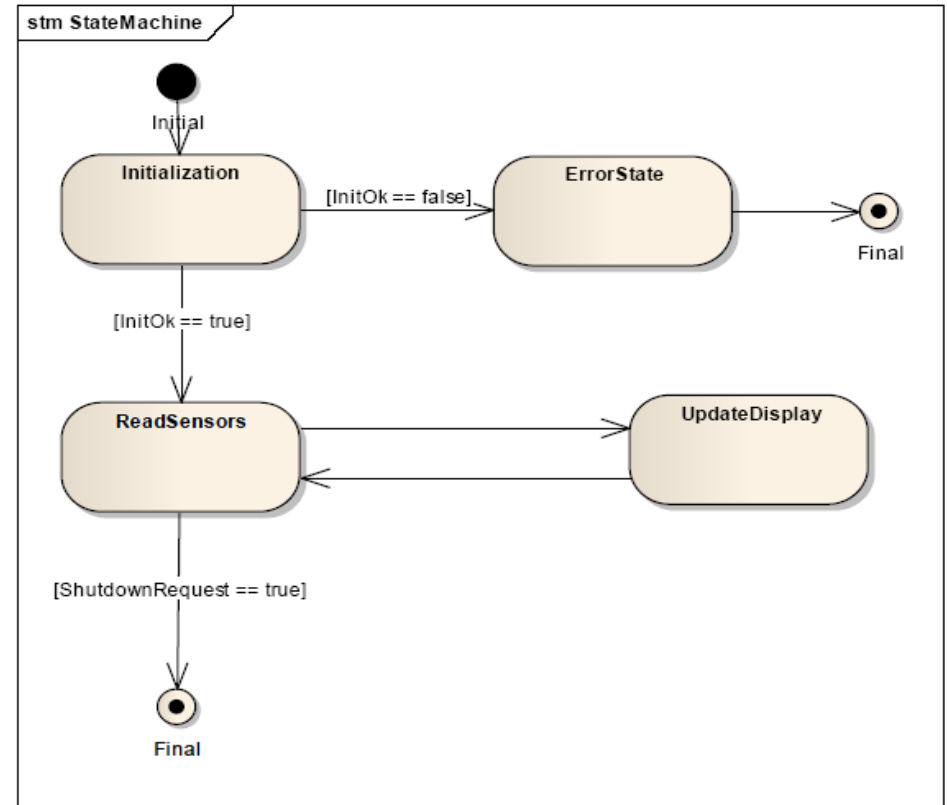
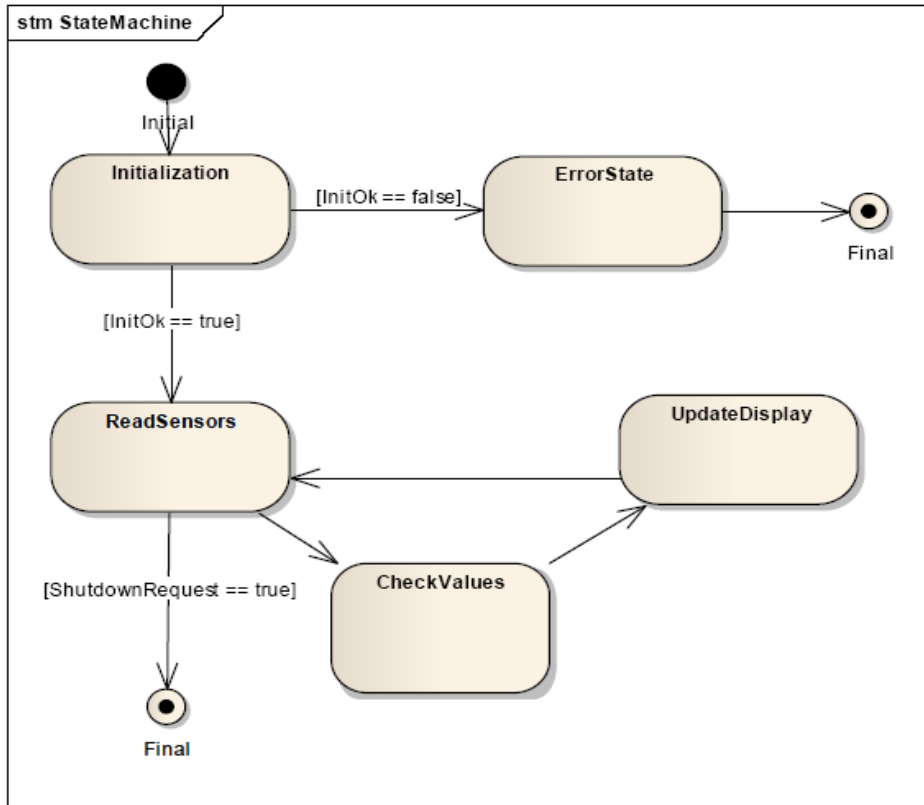
# Initial UML Model



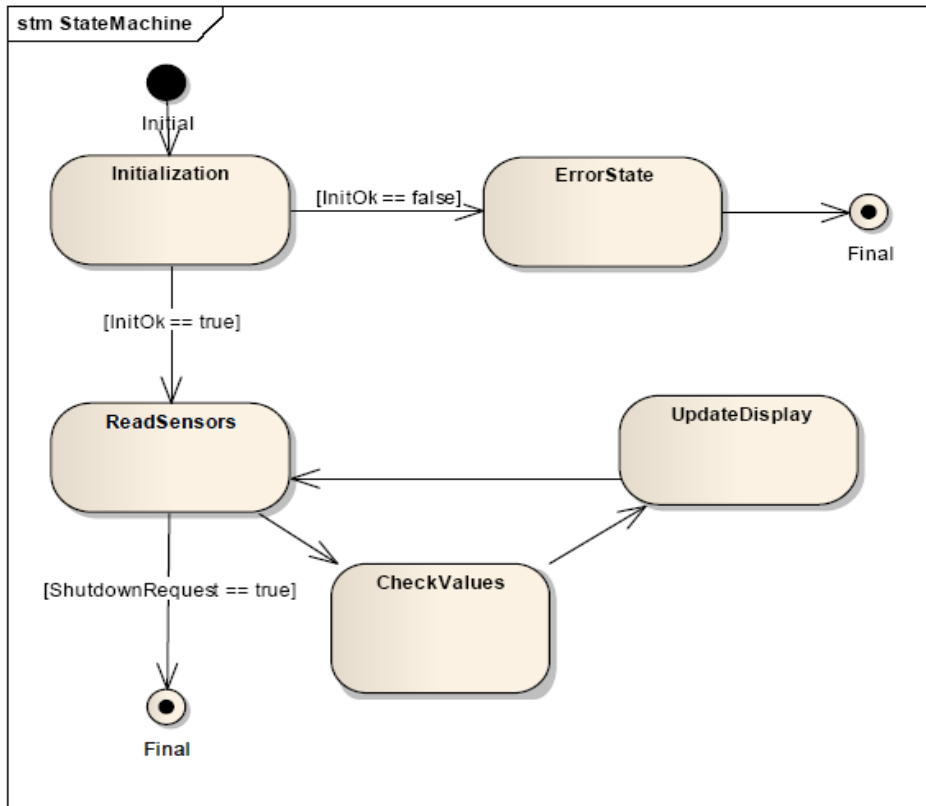


# Changed Model

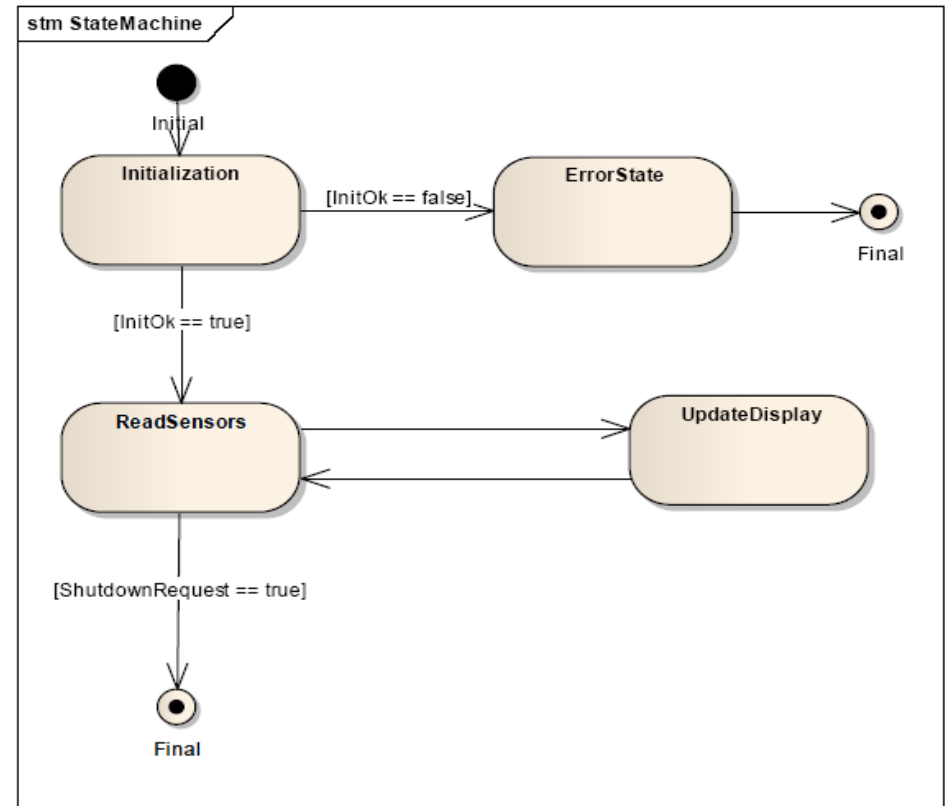
# The Result – 2 Variants



# Unified UML Model      The Result – 2 Variants

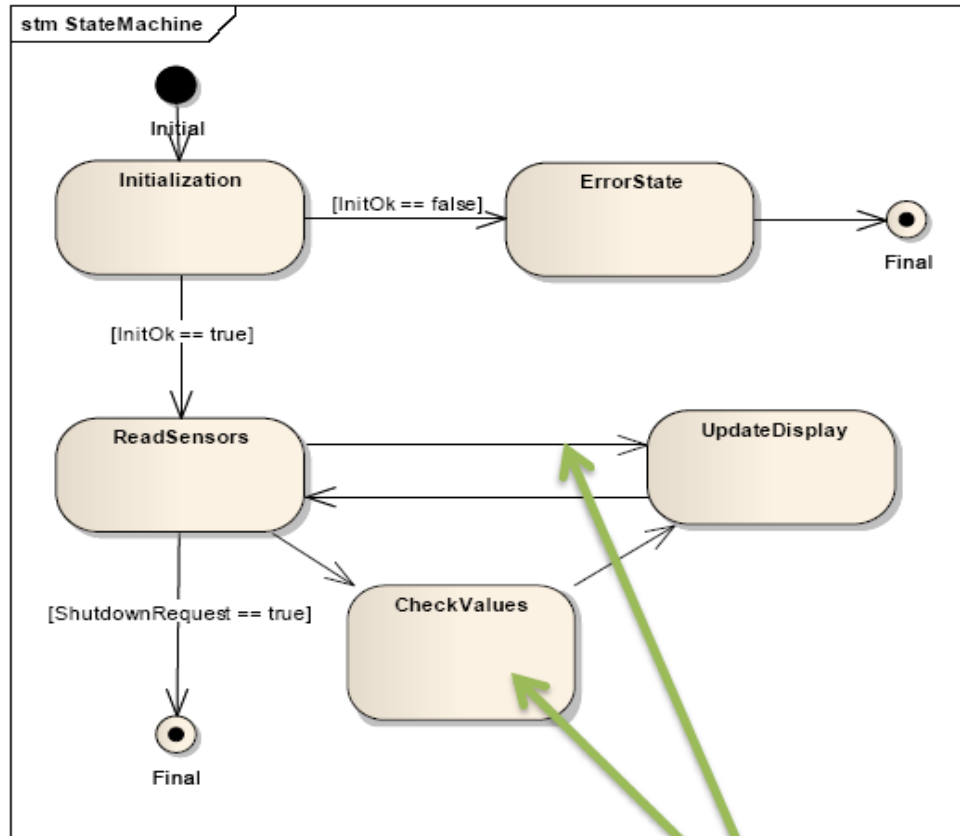


## Variant 1



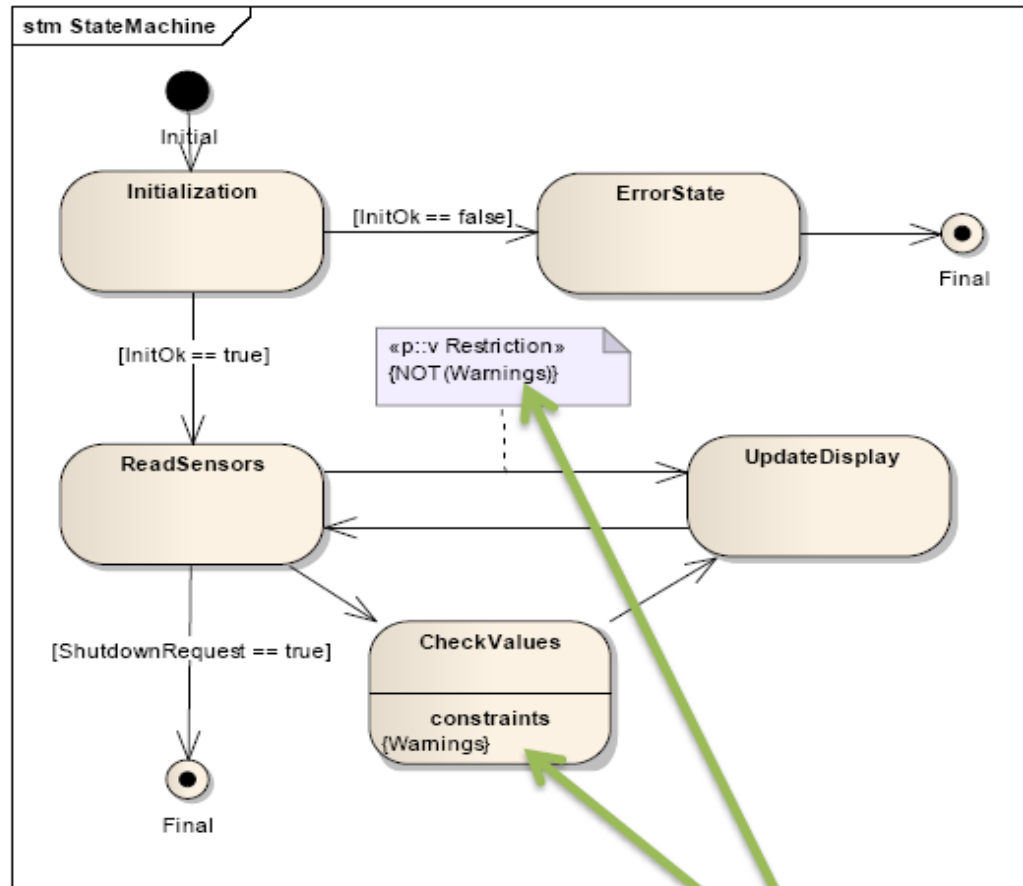
## Variant 2

# Unified UML Model



2 Variation Points

# Linked to Variability Model



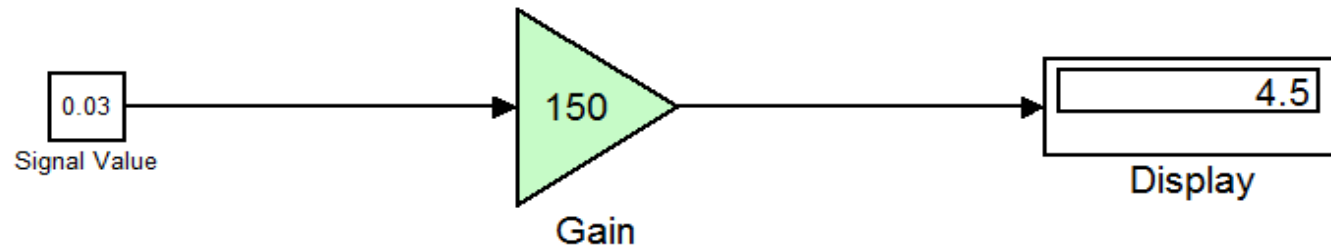
2 Rules

# pure::variants for UML



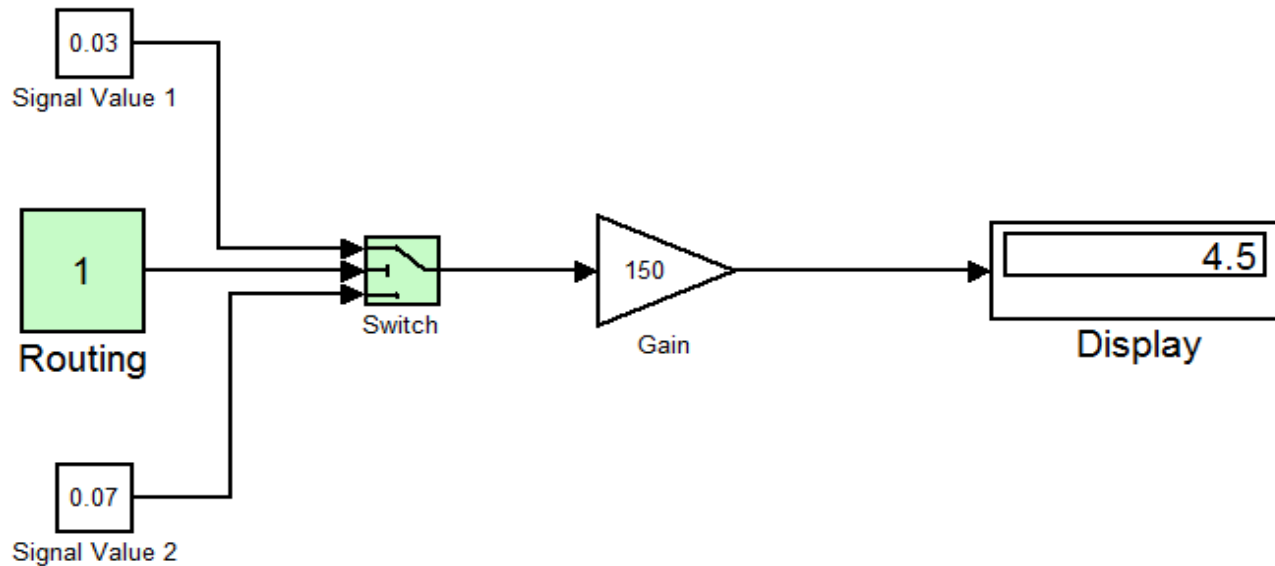
# Variability in Simulink

# Expressing Variability in Simulink



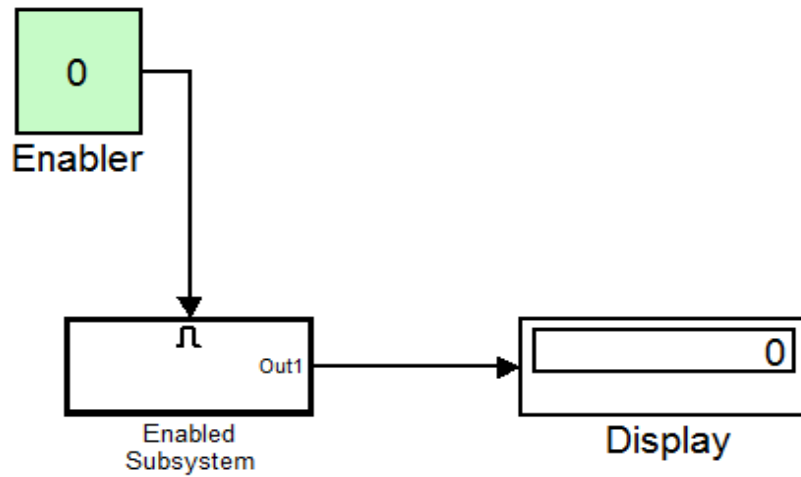
## Parametrization

# Expressing Variability in Simulink



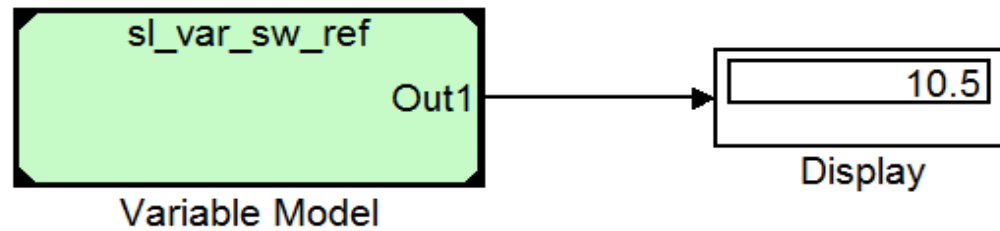
## Signal Routing

# Expressing Variability in Simulink



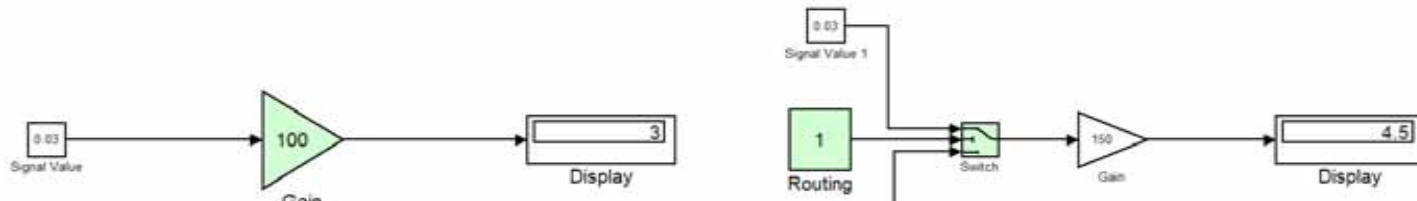
## Subsystems

# Expressing Variability in Simulink

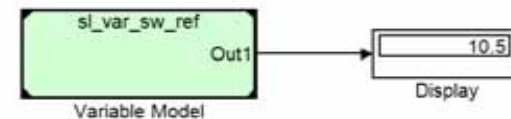
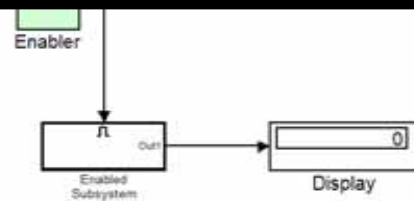


## Model Reference

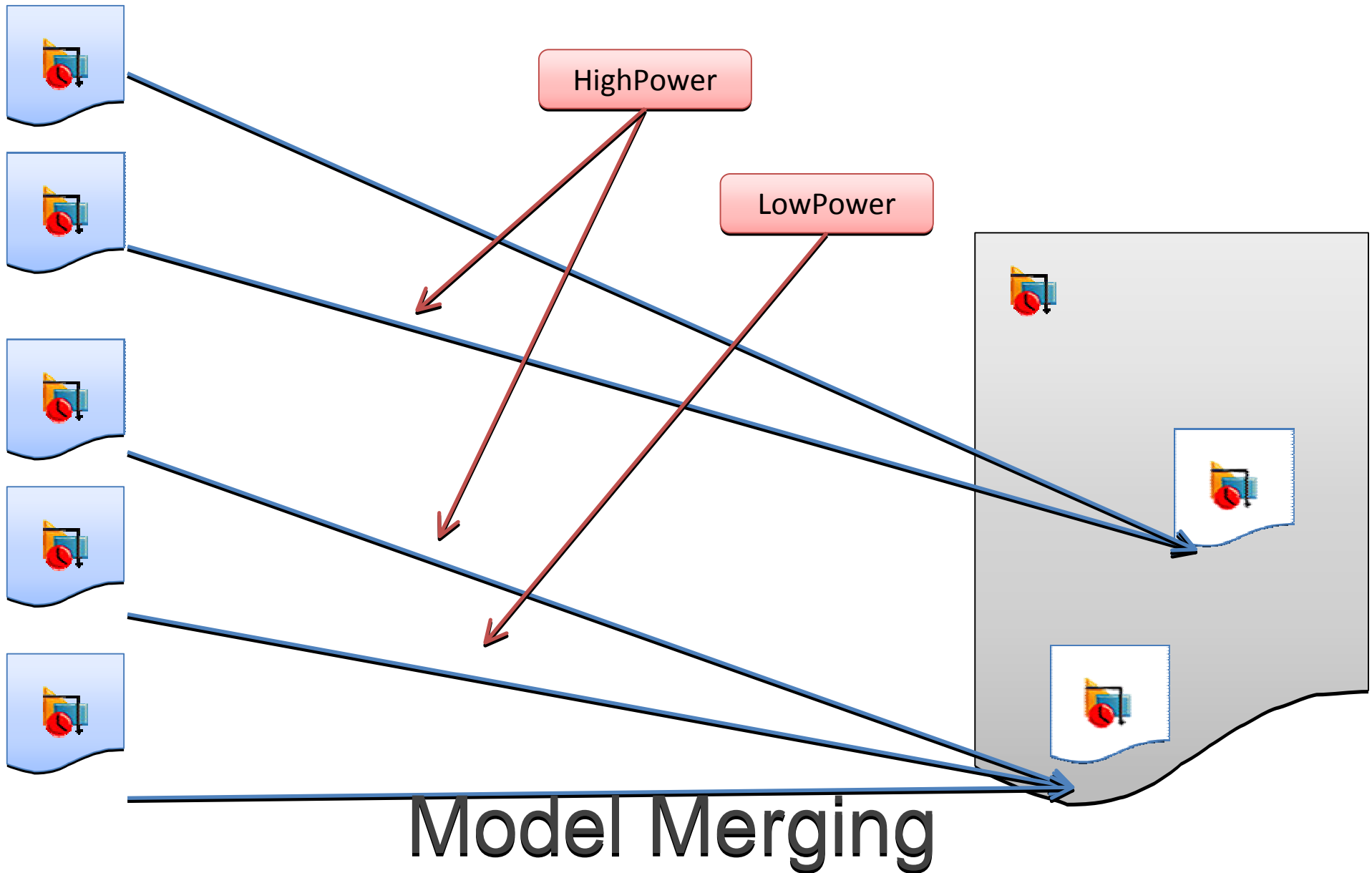
# Expressing Variability in Simulink



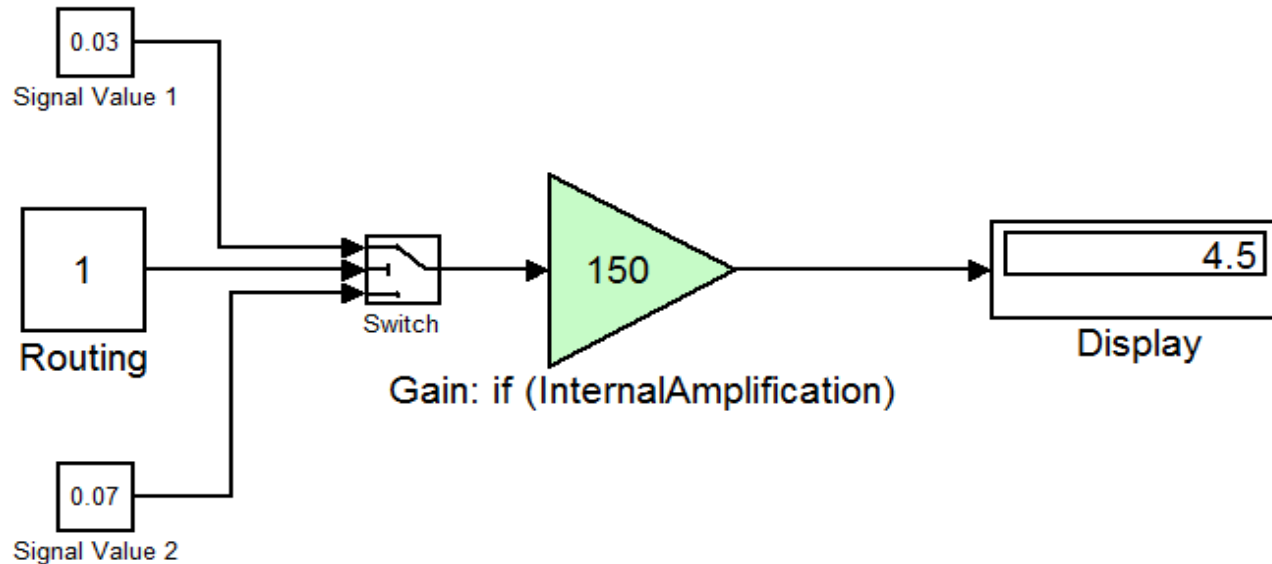
All Controlled by Parameters



# Expressing Variability in Simulink



# Expressing Variability in Simulink



## Structural Manipulation



# Expressing Variability in Simulink



# Expressing Variability in Simulink

## Using Code Generation (TargetLink):

Variable Classes

Code or Data Variants in Data Dictionary

Generated preprocessor statements

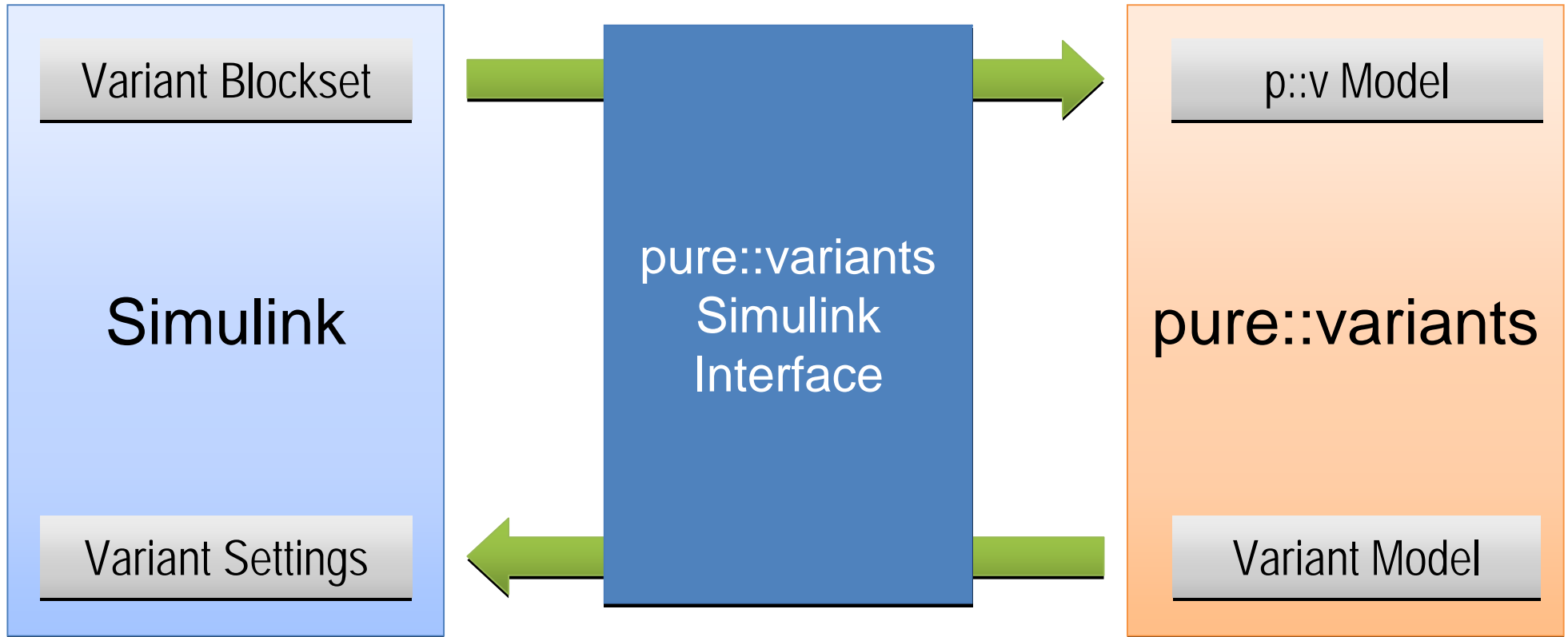
## Variability in Stateflow:

Use of “variation point” inputs

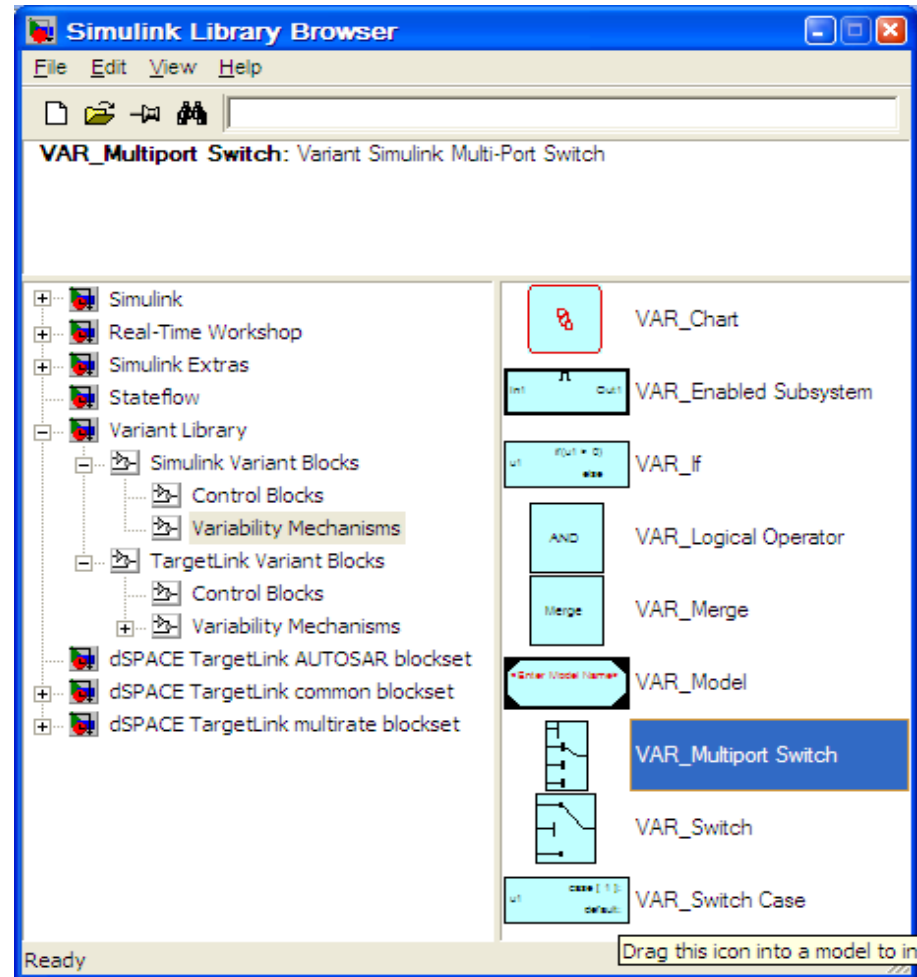
Model manipulation ( pure::variants Connector for Simulink)

# pure::variants for Simulink

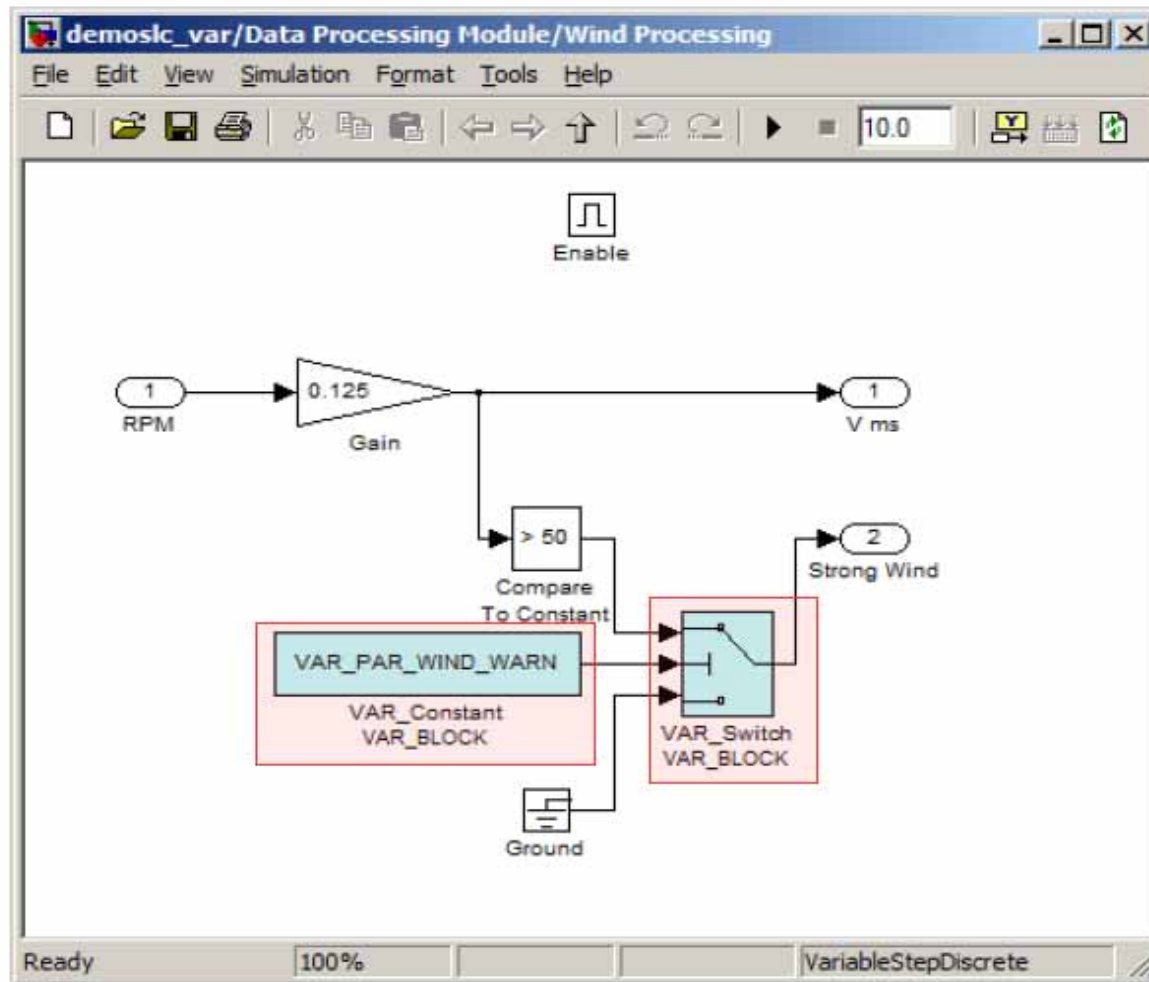
# Overview



# Variant Blockset



# Variant Blocks in Model



# Variant Block Properties and Variation Point Explorer

**Variant Block Properties**

Variation Point Selection

Name: VAR\_PRESS

Parameter: VAR\_PAR\_PRESS

Parameter Value: <<no variant set>>

Variation Label & Value

On	1
Off	0

Comment

Refresh VP List      Open VP Explorer

Block      OK      Cancel

**Variation Point Explorer**

Variation Point List

- VAR\_PRESS
- VAR\_TEMP
- VAR\_TEMP\_PT\_TYPE
- VAR\_TEMP\_WARN
- VAR\_WIND
- VAR\_WIND\_WARN

Refresh      Delete

Static Text

Variation Point Definition for VAR\_PRESS

Name: VAR\_PRESS

Parameter: VAR\_PAR\_PRESS

Variations

Variation Label	Variation Value		
		Add	Edit
On	1		
Off	0		
		Del	

Comment

Clear      Create      Apply      Revert

Variation Point Configuration

Value: <<no variant set>>

Apply      Revert

# pure::variants for Simulink: Model Import

Import MATLAB/Simulink Model - Import Preferences

Specify the specific settings for the model Import

!	Element name	Element type
X	SubSystem	BlockType

!	Parameter name

Base Workspace regular expression:  
VAR\_!w+

Model Workspace regular expression:  
VAR\_!w+

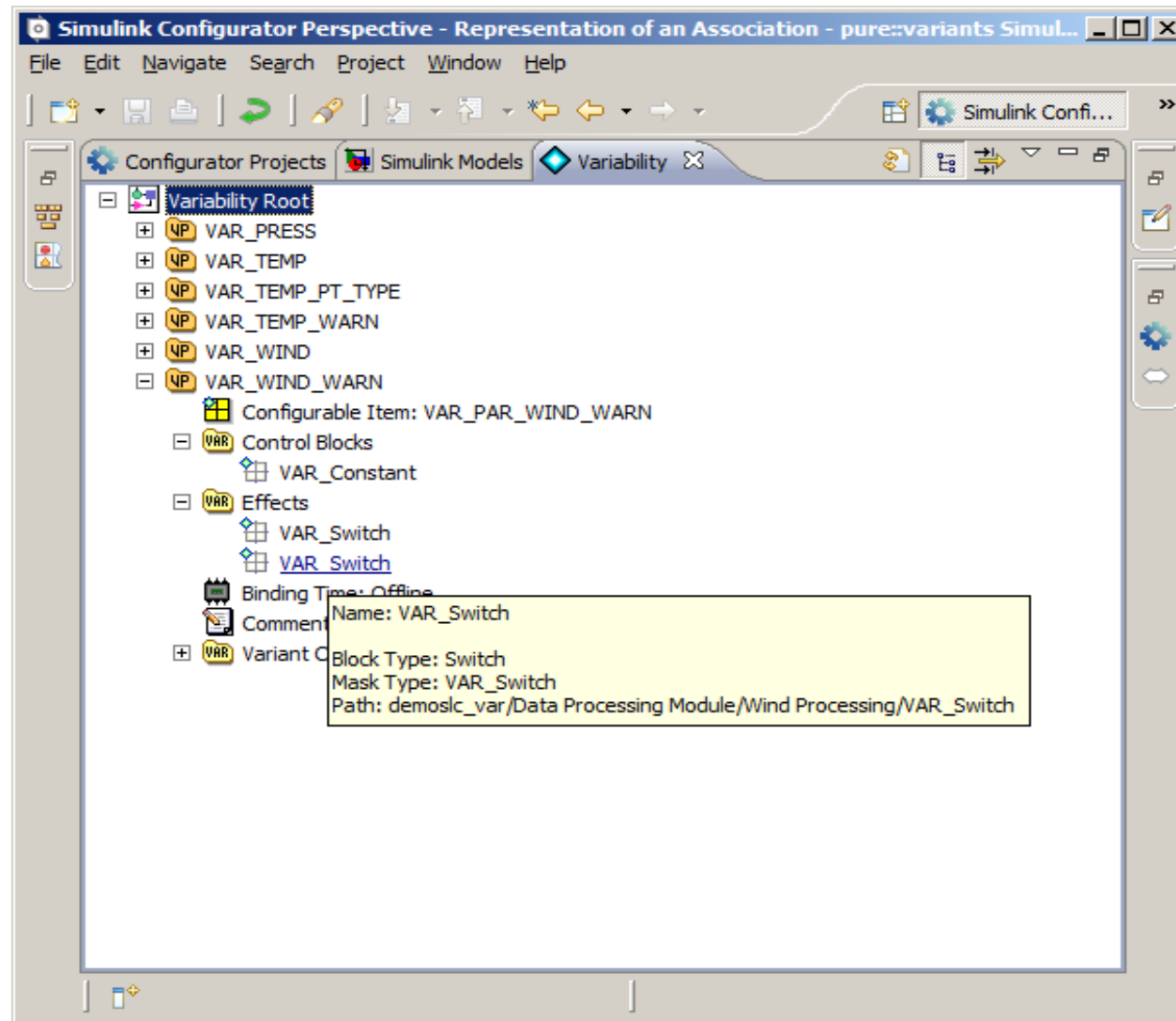
Element regular expression:

Buttons: Add, Edit, Remove, Clear, XML Import ..., XML Export ...

Navigation: < Back, Next >, Finish, Cancel



# Variability View



# Configuration and Association Views

The screenshot displays the Simulink Configurator interface for a model named 'Thermometer.vdm'. The main window is divided into several panes:

- Left Pane:** A tree view showing the project structure, including 'DemoSLC', 'Transformation Scripts', 'Simulink Data', 'demoslc\_var', 'WeatherAssoc', and 'Thermometer'.
- Center Pane:** A table titled 'Simulink Variability Table' showing the configuration for various variation points. The table has columns for 'Variation Point', 'Configurable Item', 'Value', and 'Binding Time'.
- Right Pane:** An 'Association Model' view showing a hierarchy of assignments. It includes 'Default Assignments' and specific assignments for 'VAR\_TEMP' and 'VAR\_WIND'.
- Bottom Pane:** A 'Results' table that provides a detailed view of the configuration, including an 'Icon' column for each row.

Variation Point	Configurable Item	Value	Binding Time
VAR_WIND	VAR_PAR_WIND	[1] On	Offline
VAR_PRESS	VAR_PAR_PRESS	[0] Off	Offline
VAR_TEMP	VAR_PAR_TEMP	[1] On	Offline
VAR_TEMP_WARN	VAR_PAR_TEMP_...	[0] Off	Offline
VAR_TEMP_PT_...	VAR_PAR_PT_TYPE	[100] PT...	Offline
VAR_WIND_WARN	VAR_PAR_WIND_...	[1] On	Offline

Icon	Variation Point	Configurable Item	Value	Binding Time
ⓘ	VAR_WIND_WARN	VAR_PAR_WIND_...	[1] On	Offline
ⓘ	VAR_TEMP	VAR_PAR_TEMP	[1] On	Offline
ⓘ	VAR_WIND	VAR_PAR_WIND	[1] On	Offline
ⓘ	VAR_TEMP_PT_TYPE	VAR_PAR_PT_TYPE	[100] PT100	Offline
ⓘ	VAR_PRESS	VAR_PAR_PRESS	[0] Off	Offline
ⓘ	VAR_TEMP_WARN	VAR_PAR_TEMP_...	[0] Off	Offline

The right pane shows the following association model structure:

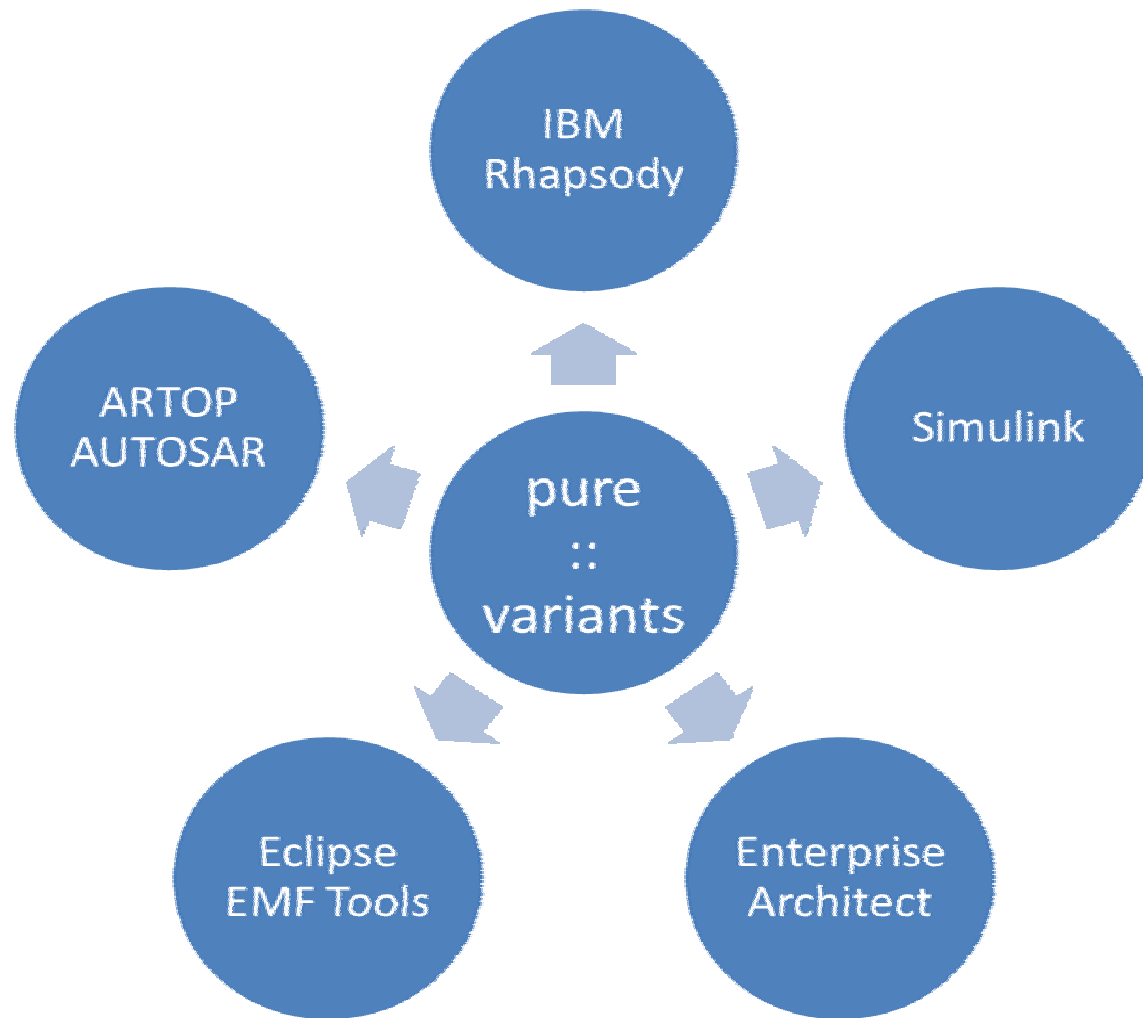
- Association Model
  - Default Assignments
    - VAR\_PAR\_PRESS = [0] Off
    - VAR\_PAR\_TEMP\_WARN = [0]
    - VAR\_PAR\_TEMP = [1] On
    - VAR\_PAR\_WIND\_WARN = [0]
    - VAR\_PAR\_WIND = [0] Off
  - VAR\_TEMP
    - VAR\_PAR\_TEMP\_WARN = [1] On
    - VAR\_PAR\_TEMP = [1] On
  - VAR\_WIND
    - VAR\_PAR\_WIND\_WARN = [1] On
    - VAR\_PAR\_WIND = [1] On

# Variant Model

The screenshot displays the Simulink Configurator Perspective interface. The main window shows a table of configuration parameters for a variant model. The table has five columns: Variation Point, Configurable Item, Value, and Binding Time. The data is as follows:

Variation Point	Configurable Item	Value	Binding Time
VAR_WIND	VAR_PAR_WIND	[1] On	Offline
VAR_PRESS	VAR_PAR_PRESS	[0] Off	Offline
VAR_TEMP	VAR_PAR_TEMP	[1] On	Offline
VAR_TEMP_WARN	VAR_PAR_TEMP_...	[0] Off	Offline
VAR_TEMP_PT_...	VAR_PAR_PT_TYPE		Offline
VAR_WIND_WARN	VAR_PAR_WIND_...	[1] On	Offline

The interface also shows a menu bar (File, Edit, Navigate, Search, Variant, Project, Window, Help), a toolbar with various icons, and a status bar at the bottom with the text "33 : 54" and "80 (1)".



# Test Cases

Reuse Items

Estimates

Architecture

Models

Code

Requirements

Subsystems

Documentation

Rules

# SPL Testing vs. Single Product Testing

More Asset Changes



Test Runs Increase



Test Affected  
Products Only



Combinatorial  
Testing



Test  
Automation

# SPL Testing vs. Single Product Testing

Testing Whole Product Line



Potential Variant Number  
Explodes



Test Only  
Relevant  
Products

# SPL Testing vs. Single Product Testing

More Test Reuse



Test Creation Effort  
Decrease



Variation Points



Test Selection



# What's The Difference?

Does it have...?

PL  
Asset

Test  
Asset

Identity

YES

YES

Relations

YES

Variations

YES

Selection Rules

YES

YES

**No difference!**

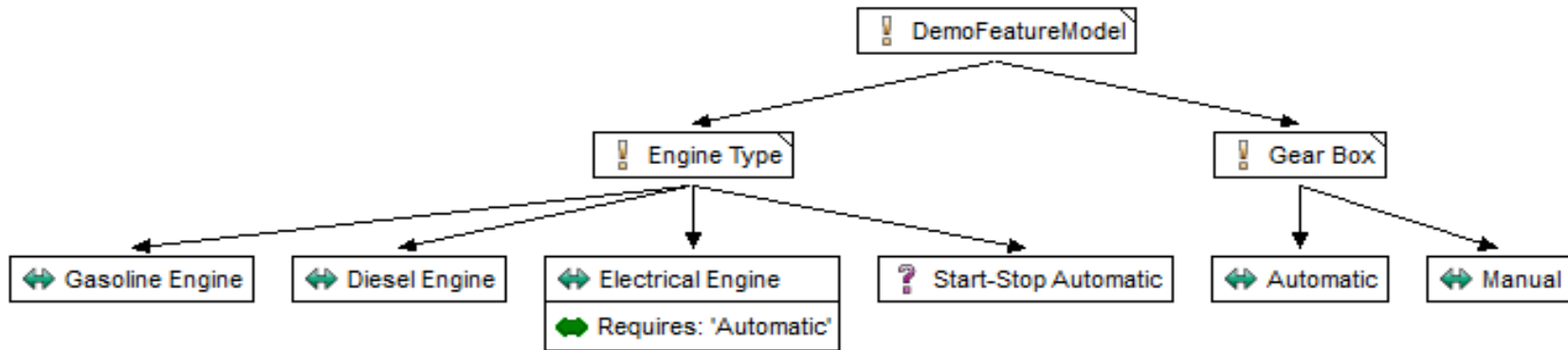
# Test Affected Products Only

Model Elements			Level			pure_var2	pure_var3
[-] Climate Control	F					✓	✓
+ Compressors	F		1			✓	✓
+ Control Panel	F		2	✓	✓	✓	✓
+ Climate Zones	F		3	✓	✓	✓	✓
+ Control Board	F		4	✓	✓	✓	✓
[-] Root				✓	✓	✓	✓
[-] Root			1	✓	✓	✓	✓
[-] puredemo			1.1	✓	✓	✓	✓
+ bus			1.1.1	✓	✓	✓	✓
+ unittests			1.1.2	✓	✓	✓	✓
+ ps:class: AbstractCar	G		1.1.3	✓	✓	✓	✓
+ ps:class	G		1.1.4	✓	✓	✓	✓
+ ps:class	G		1.1.5	✓	✓	<input type="checkbox"/>	✓
+ ps:class	G		1.1.6	✓	✓	<input type="checkbox"/>	✓
[-] ps:class: Controller	G		1.1.7	<input type="checkbox"/>	<input type="checkbox"/>	✓	<input type="checkbox"/>
ps:file: Car2ClimaControl.java			1.1.7.1	<input type="checkbox"/>	<input type="checkbox"/>	✓	<input type="checkbox"/>
+ ps:class: Car2ClimaControlTest	G		1.1.8	<input type="checkbox"/>	<input type="checkbox"/>	✓	<input type="checkbox"/>
+ ps:class: CarClimaControlFactory	G		1.1.9	✓	✓	✓	✓
+ ps:class: Compressor	G		1.1.10	✓	✓	✓	✓
+ ps:class: Controller	G		1.1.11	✓	✓	✓	✓

Affected Variant

Changed Asset

# Combinatorial Testing

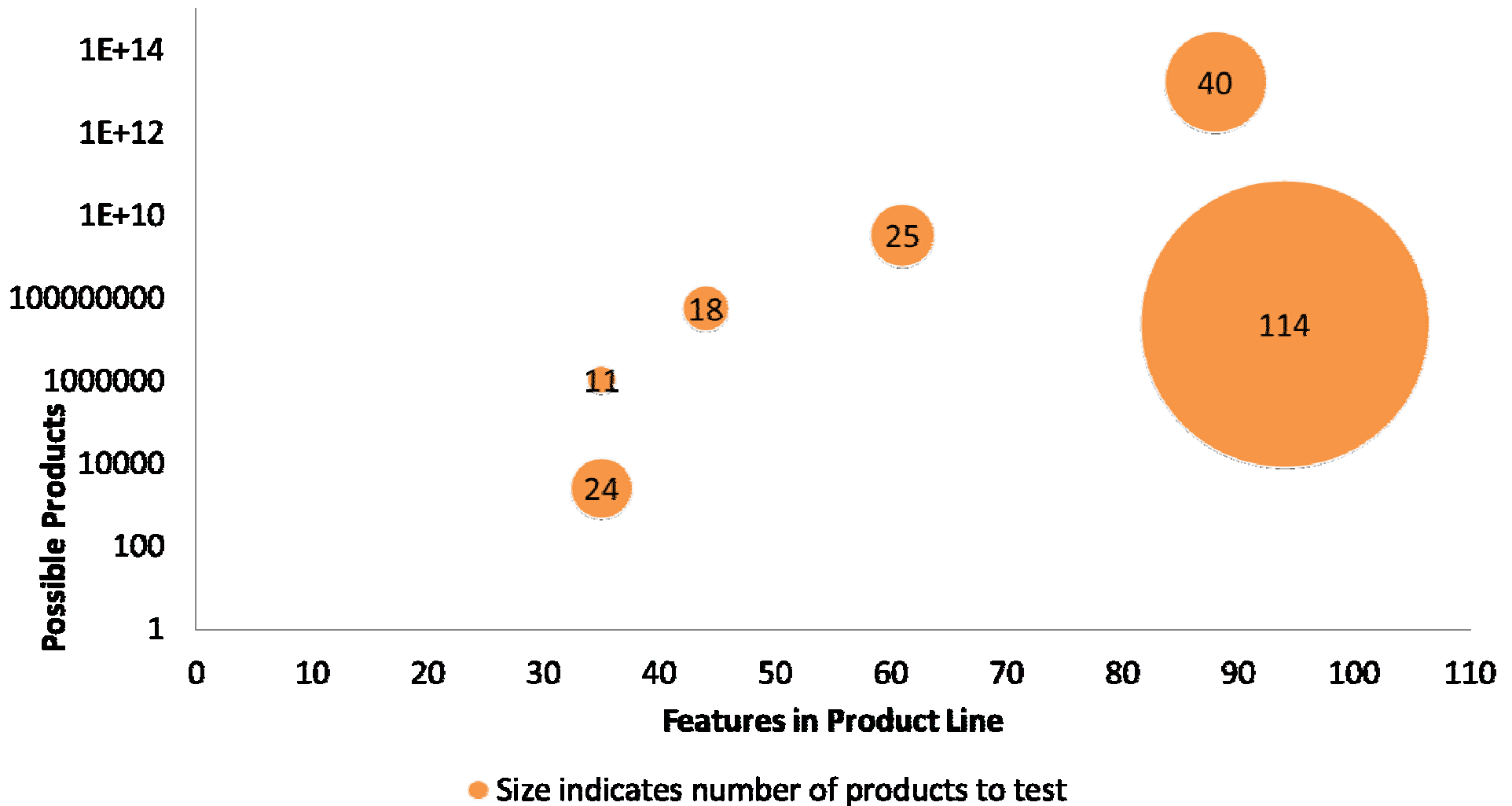


All feature pairs covered by other variants

Model Elements			L...	F...	F...	F...	F...	F...	F...	F...	F...	F...	F...
[-] DemoFeatureModel													
Gasoline Engine			1.1										
Diesel Engine			1.2										
Electrical Engine			1.3										
Start-Stop Automatic			1.4										
Automatic			2.1										
Manual			2.2										

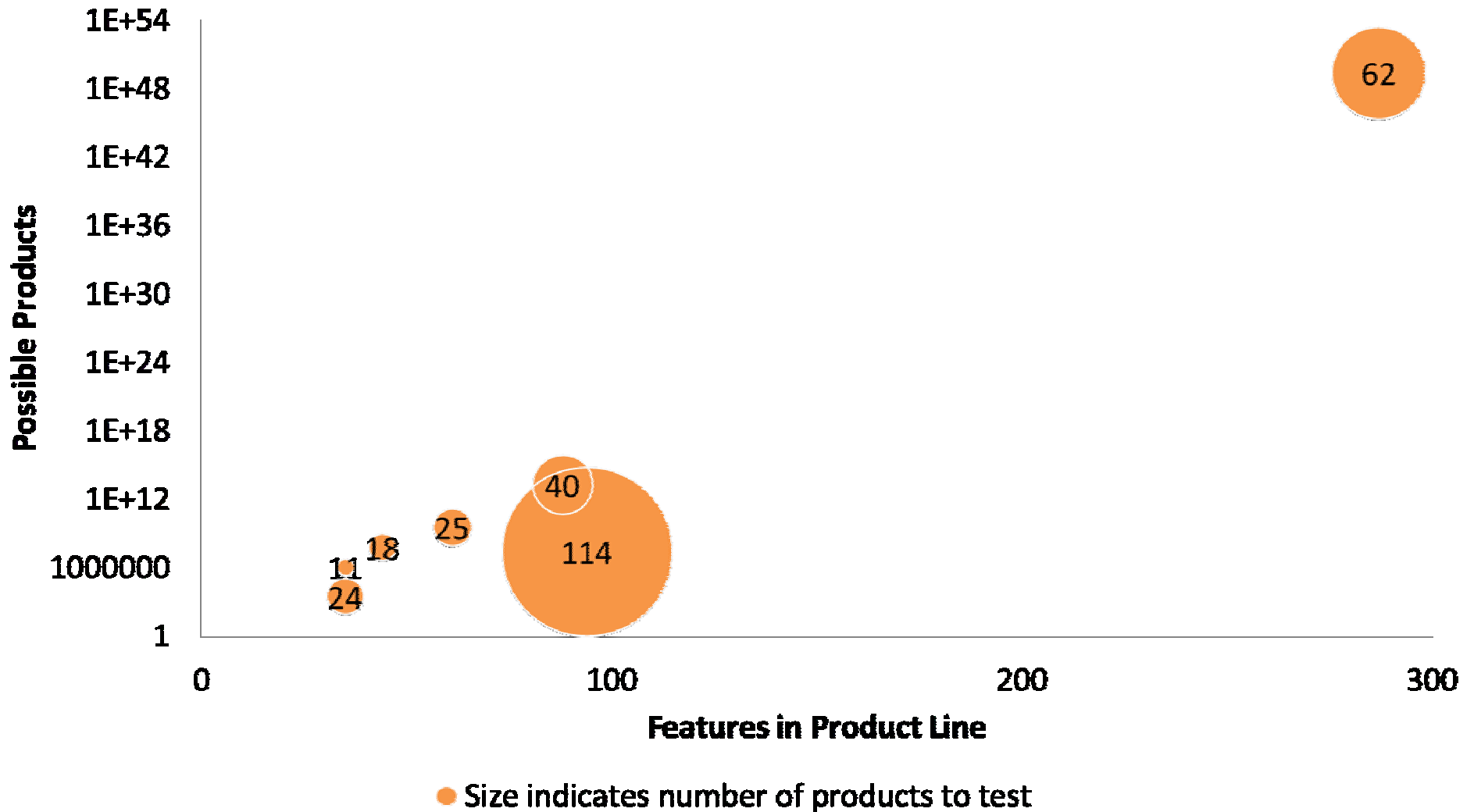
# Combinatorial Testing

## Pair-wise Feature Tests



# Combinatorial Testing

## Pair-wise Feature Tests



# Test Selection

	A	B	C	E	F
1	Test	Type	Title	Description	Execution Steps
2	T1	Automatic	Start-Stop Test for Combustion Engines with manual gear box	Test to make sure combustion engine is stopped under proper conditions automatically	<ol style="list-style-type: none"><li>1. Accelerate to 50 km/h</li><li>2. Reduce speed to zero by braking</li><li>3. Press clutch</li><li>4. Move to neutral gear</li></ol>
3	T2	Automatic	Start-Stop Test for Combustion Engines with automatic gear box	Test to make sure combustion engine is stopped under proper conditions automatically	<ol style="list-style-type: none"><li>1. Accelerate to 50 km/h</li><li>2. Reduce speed to zero by braking</li></ol>
4	T3	Automatic	Energy Efficiency Test	Test to measure degree of efficiency in the whole system by measuring full charge energy vs. energy provide by electrical motors to move car	<ol style="list-style-type: none"><li>1. Run Drive Cycle 10 until battery charge is minimal</li><li>2. Charge battery and measure consumed energy</li></ol>
5	T4	Automatic	Fuel Efficiency Test	Test to measure degree of efficiency in the whole system by measuring how much fuel is consumed when running the Test Drive Cycle	<ol style="list-style-type: none"><li>1. Fill tank to maximum</li><li>2. Run Drive Cycle 20</li><li>3. Measure used fuel by filling up to maximum again.</li></ol>

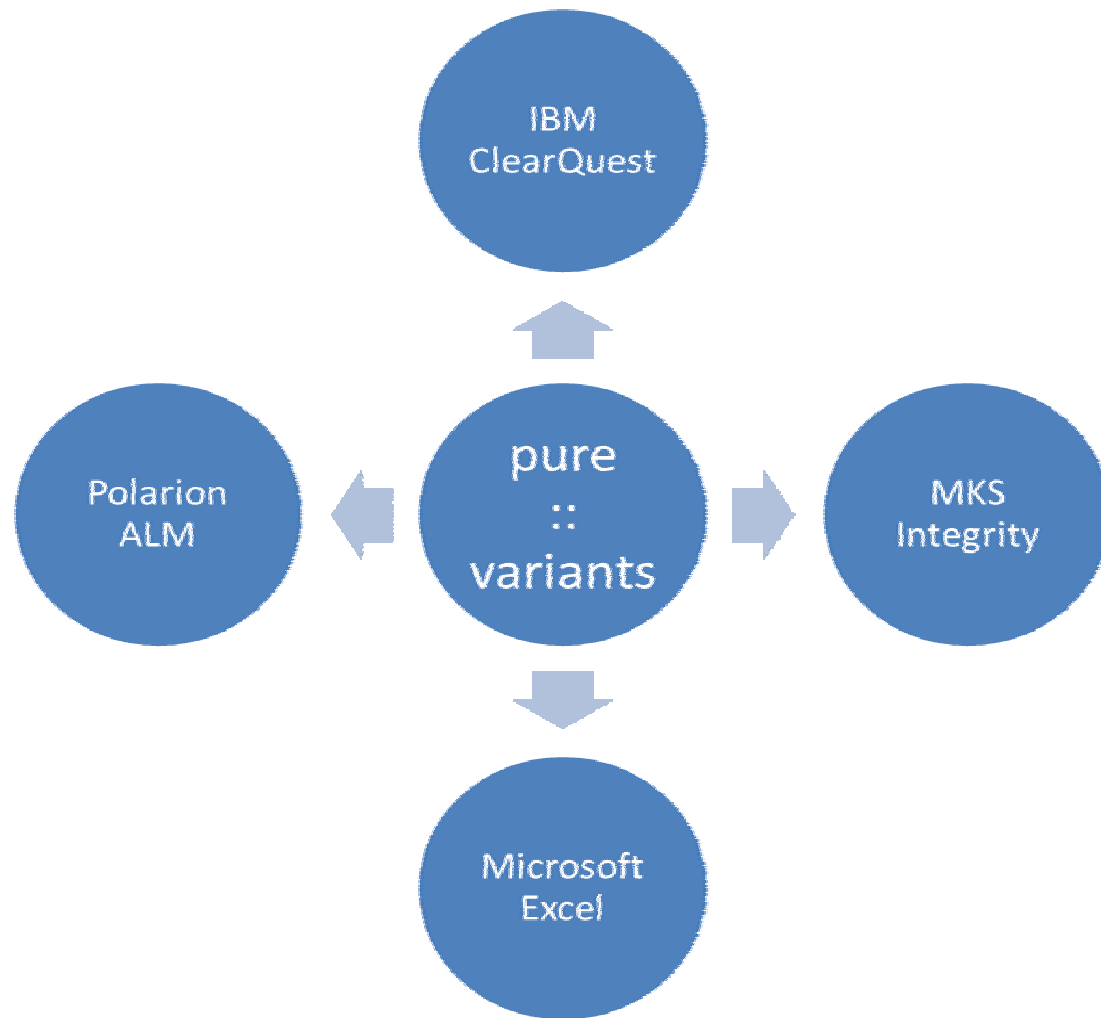
# Test Selection

	A	B	C	D	E
1	Test	Type	Title	Condition	Description
2	T1	Automatic	Start-Stop Test for Combustion Engines with manual gear box	StartStop AND ManualGearbox AND (GasolineEngine OR DieselEngine)	Test to make sure combustion engine stopped under proper conditions automatically
3	T2	Automatic	Start-Stop Test for Combustion Engines with automatic gear box	StartStop AND AutomaticGearbox AND (GasolineEngine OR DieselEngine)	Test to make sure combustion engine stopped under proper conditions automatically
4	T3	Automatic	Energy Efficiency Test	ElectricalEngine	Test to measure degree of efficiency of whole system by measuring full power vs. energy provided by electrical engine to move car
5	T4	Automatic	Fuel Efficiency Test	NOT(ElectricalEngine)	Test to measure degree of efficiency of whole system by measuring how much fuel consumed when running the Test

# Test Selection

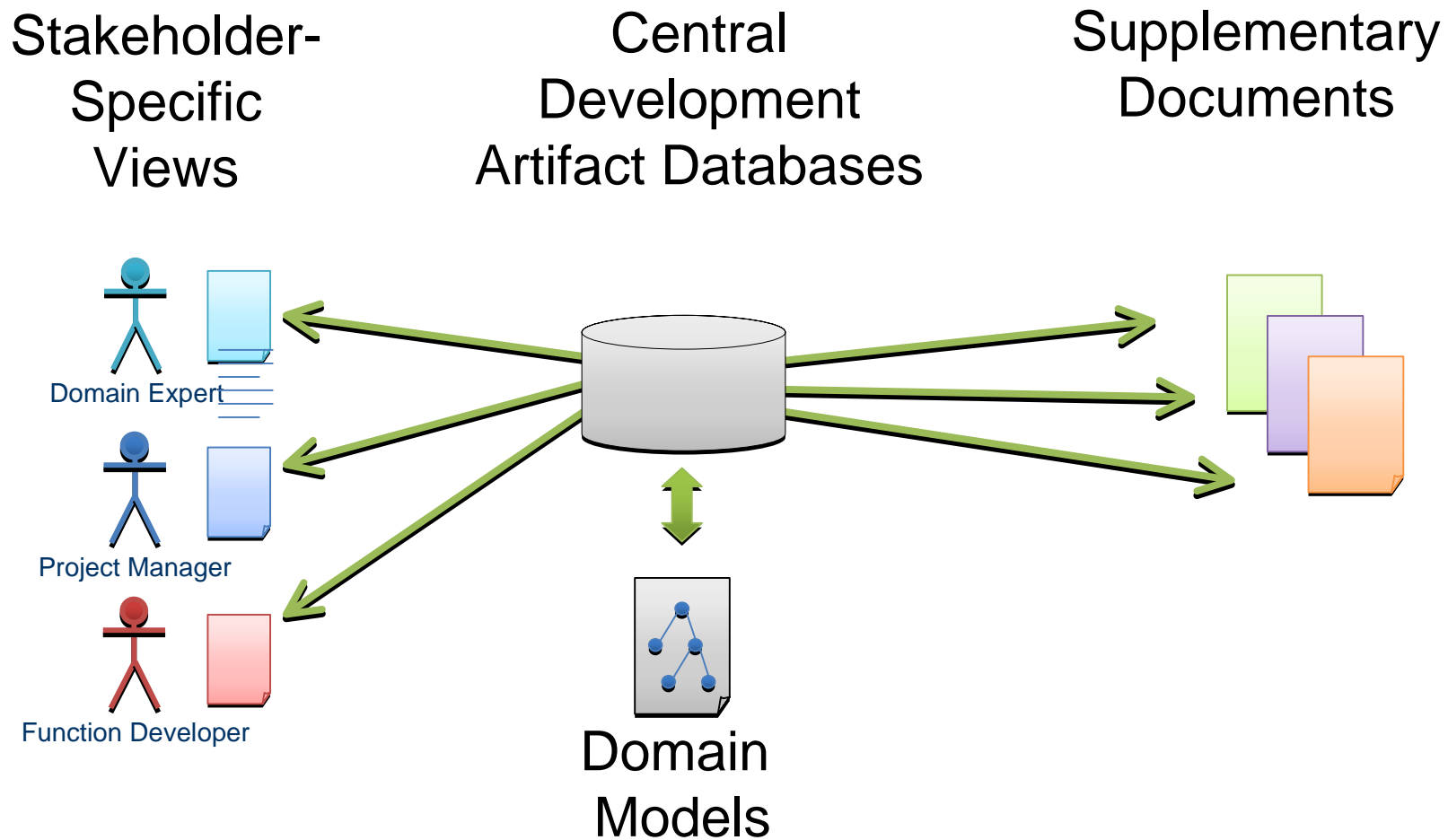
Model Elements			L...	F...	F...	F...	F...	F...	F...	F...	F...	F...	F...	
[-] DemoFeatureModel														
Gasoline Engine			1.1											
Diesel Engine			1.2											
Electrical Engine			1.3											
Start-Stop Automatic			1.4											
Automatic			2.1											
Manual			2.2											
[-] TestPlan														
TestCase_T1			1											
TestCase_T2			2											
TestCase_T3			3											
TestCase_T4			4											





# Running a Product Line – Change and Configuration Management

# Variant Management and Development Artifacts



# Challenges of PL: Change Management

Change Impact Increase

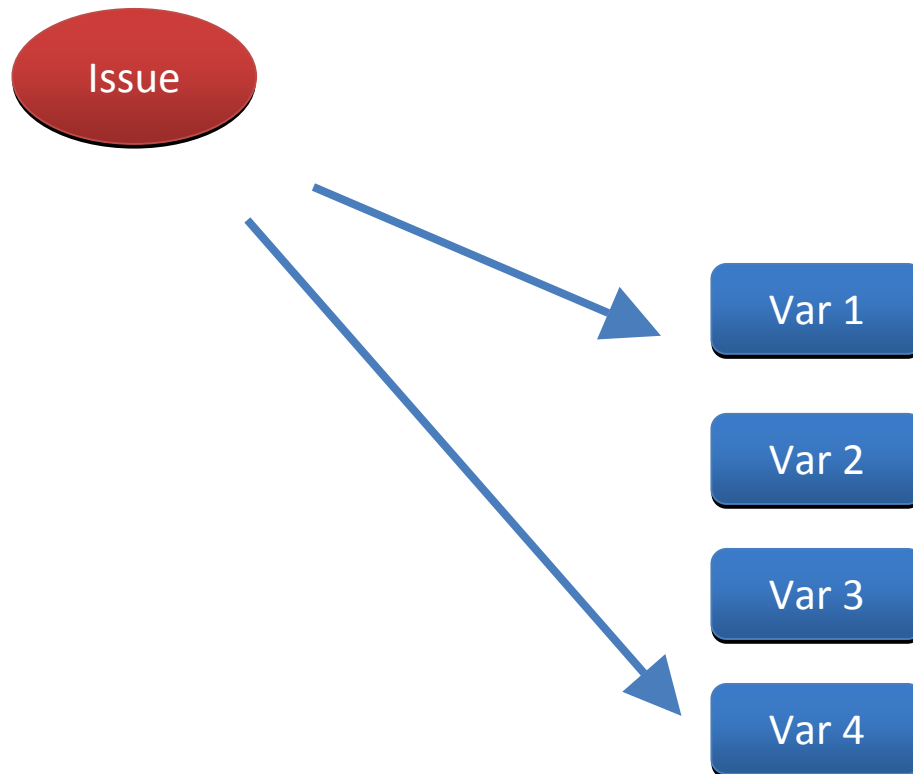
Number of Changes in Core Assets

Changes in Products vs. Changes in Core Assets

Controlling Speed of Change

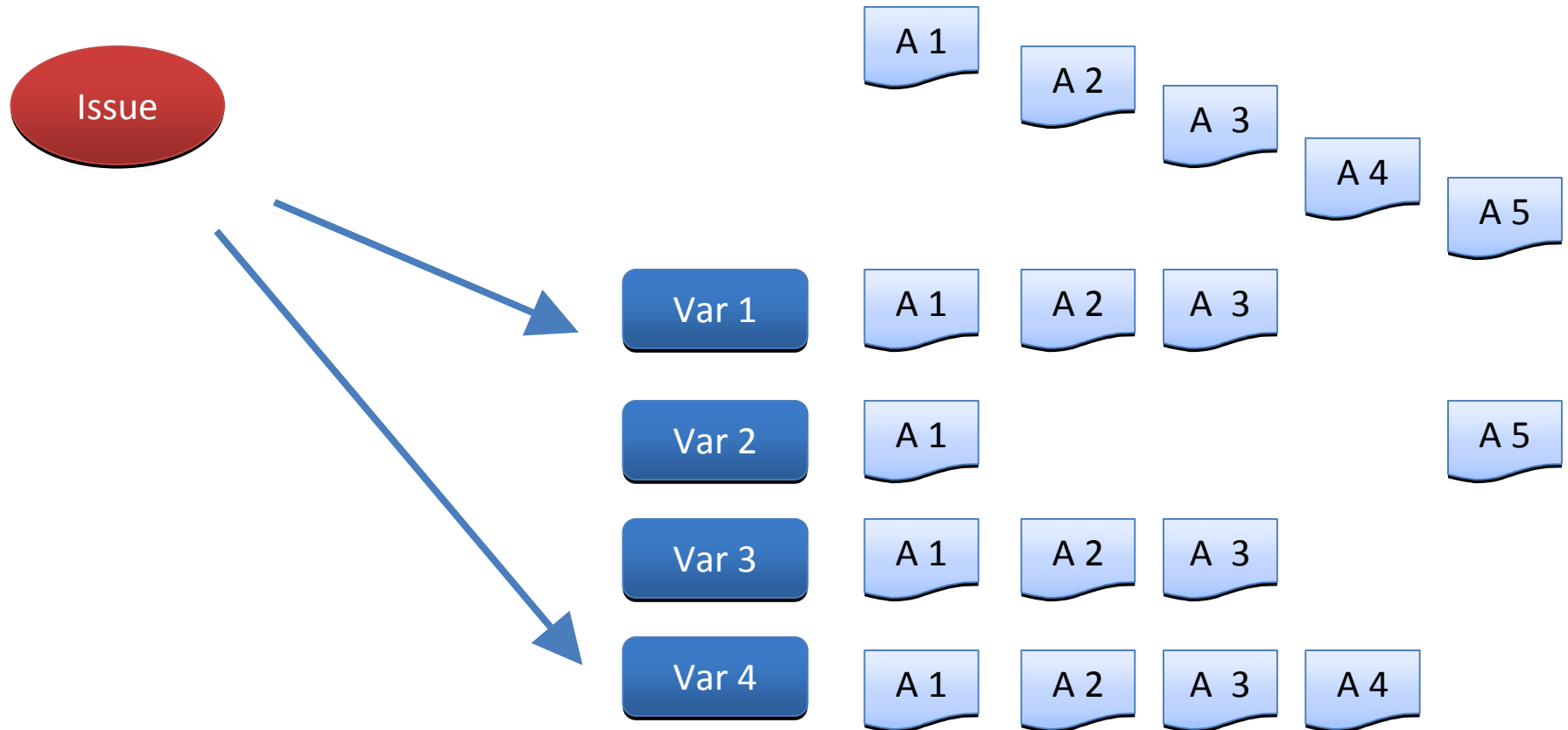
Coordination between many more stakeholders

# Variants and Change Management



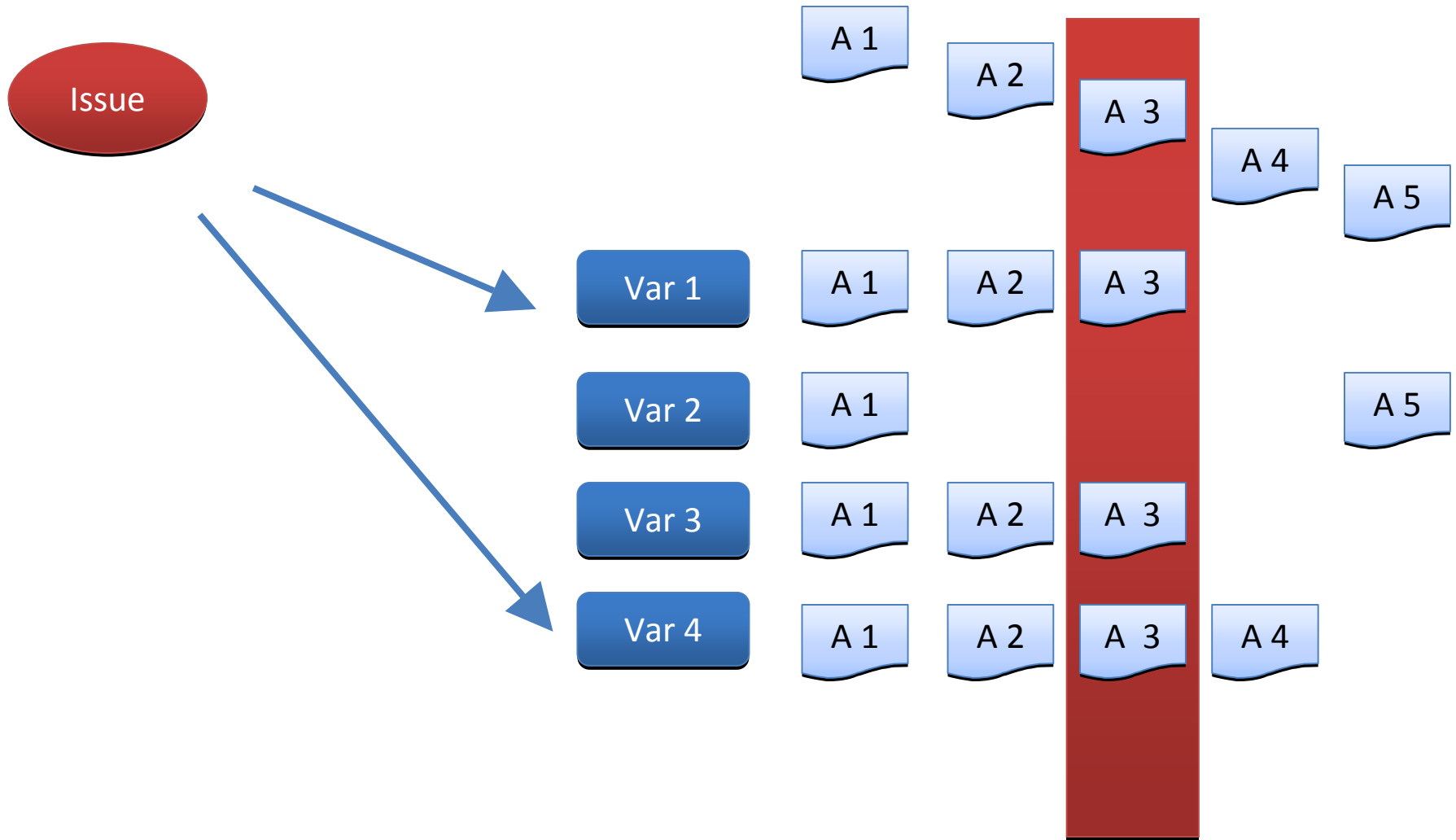
**Issue to Products**

# Variants and Change Management

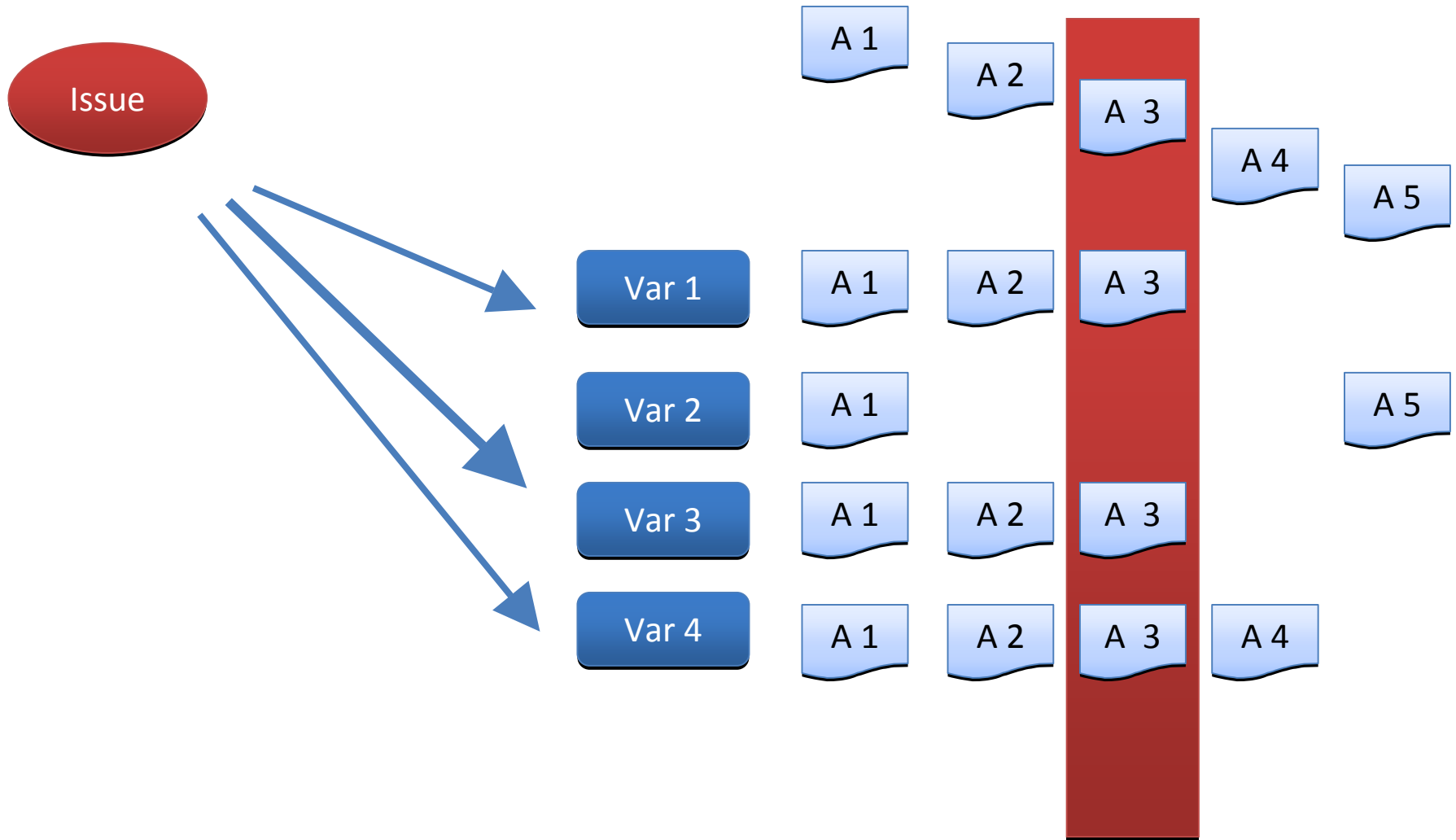


**Products to Assets**

# Variants and Change Management

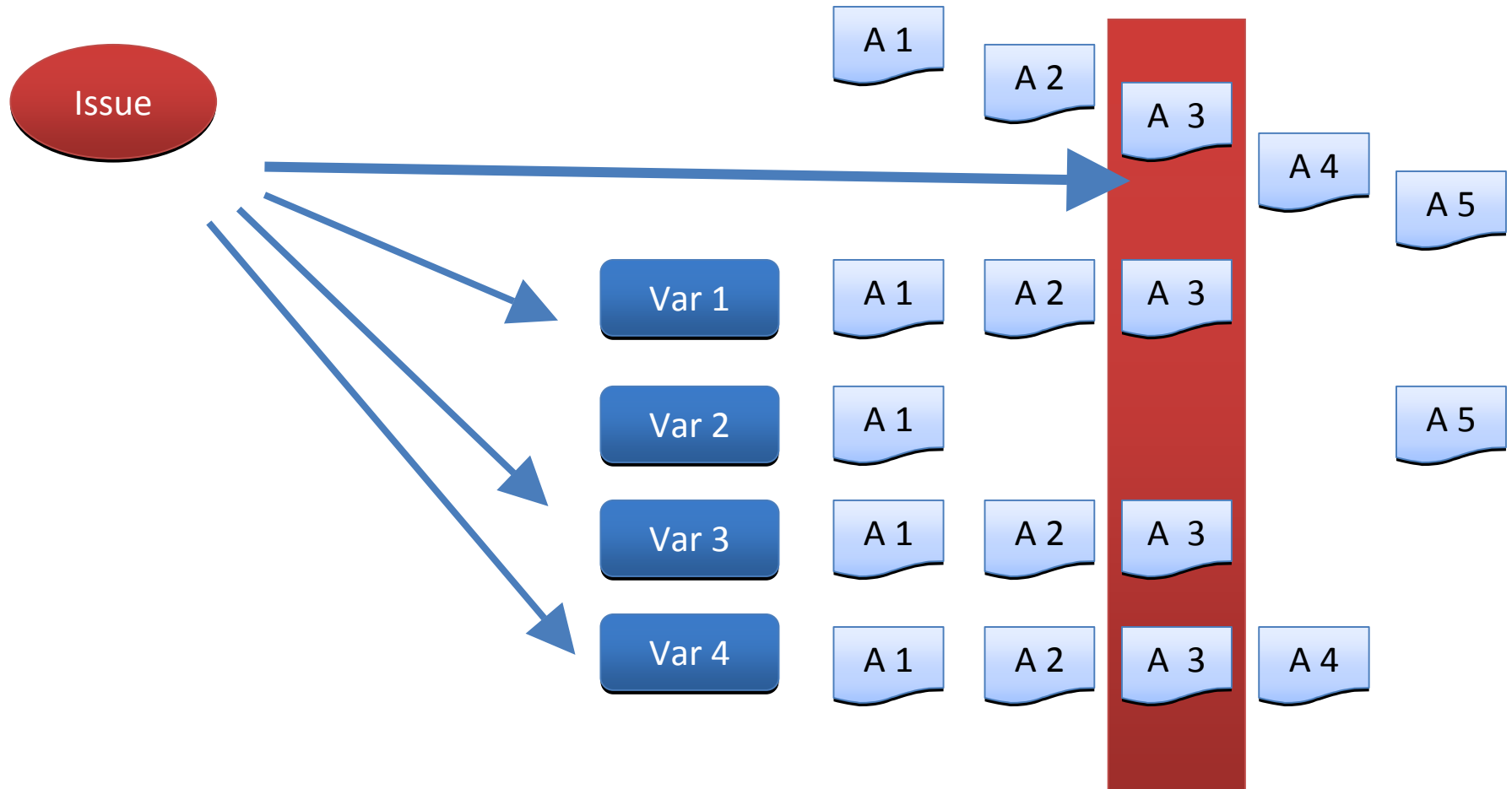


# Variants and Change Management



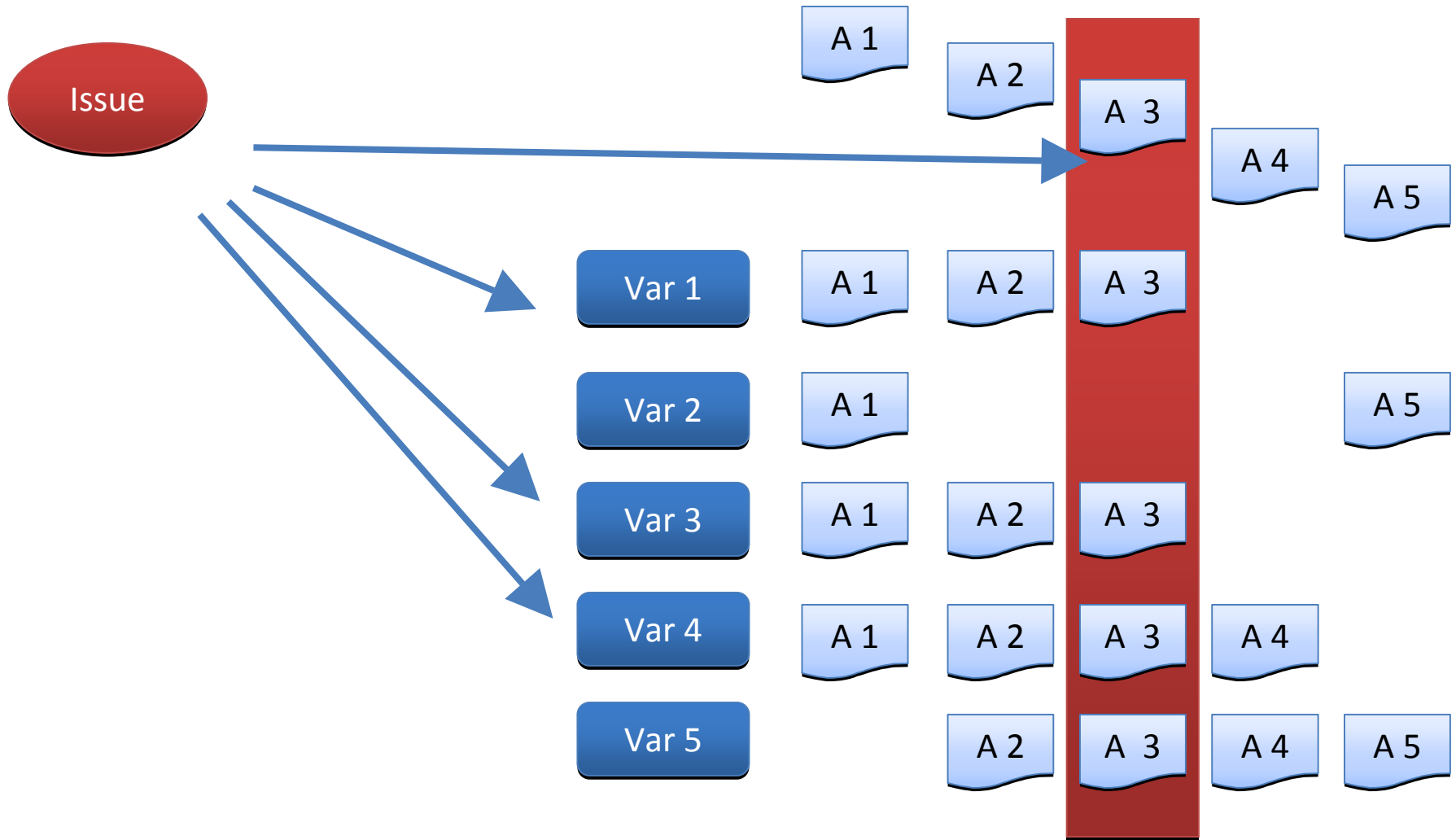


# Variants and Change Management

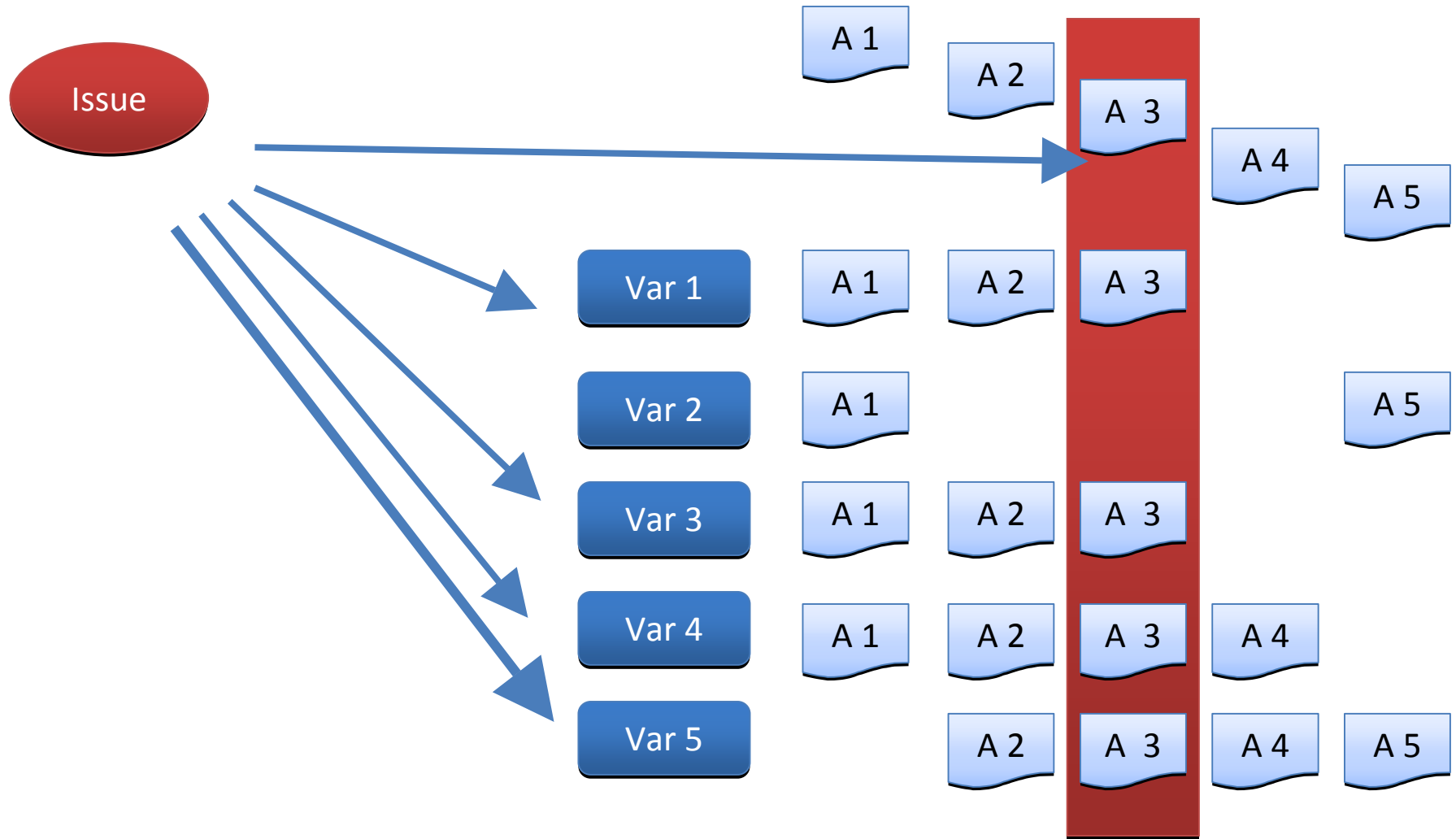


**Issues to Assets**

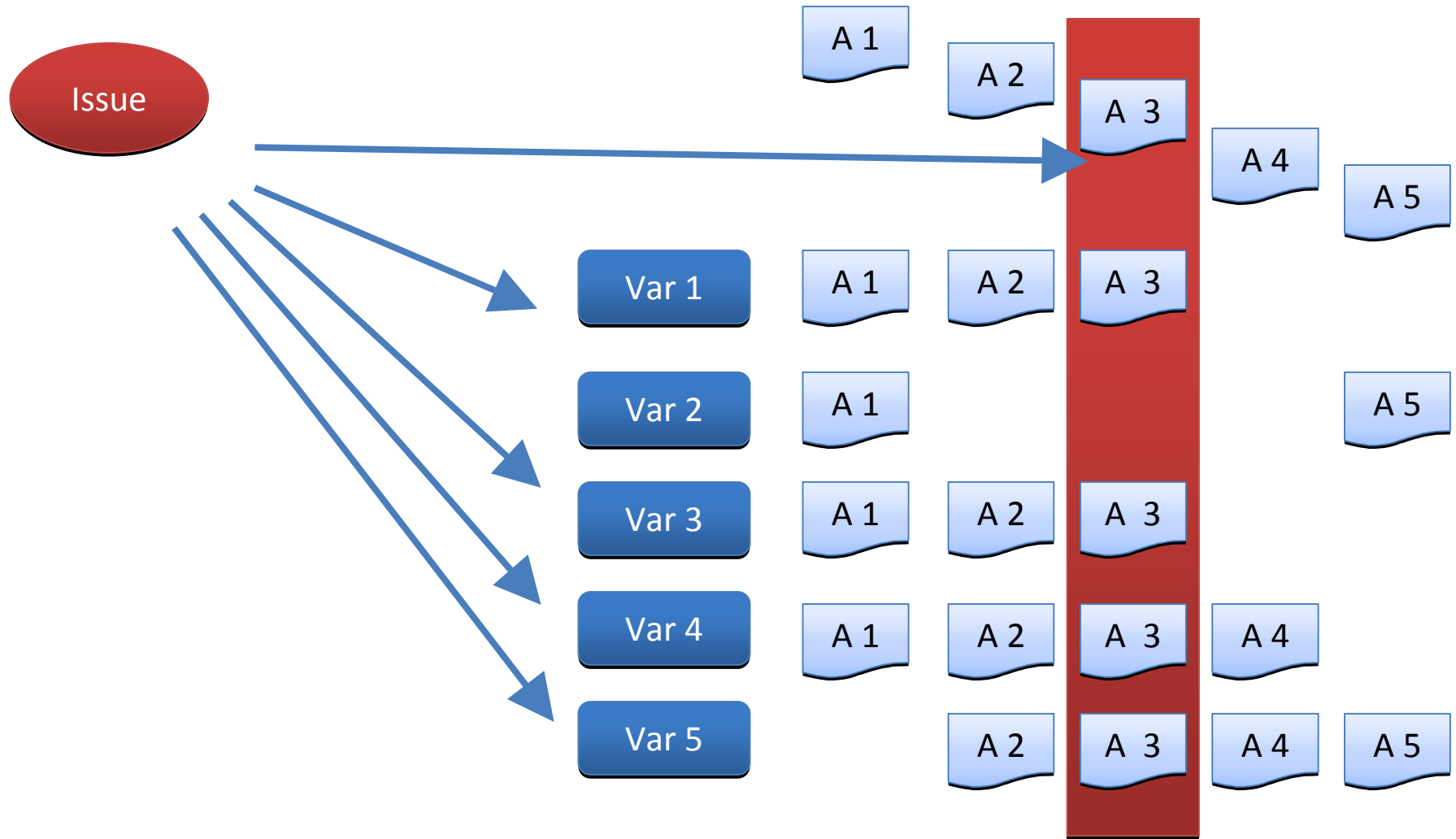
# Variants and Change Management



# Variants and Change Management

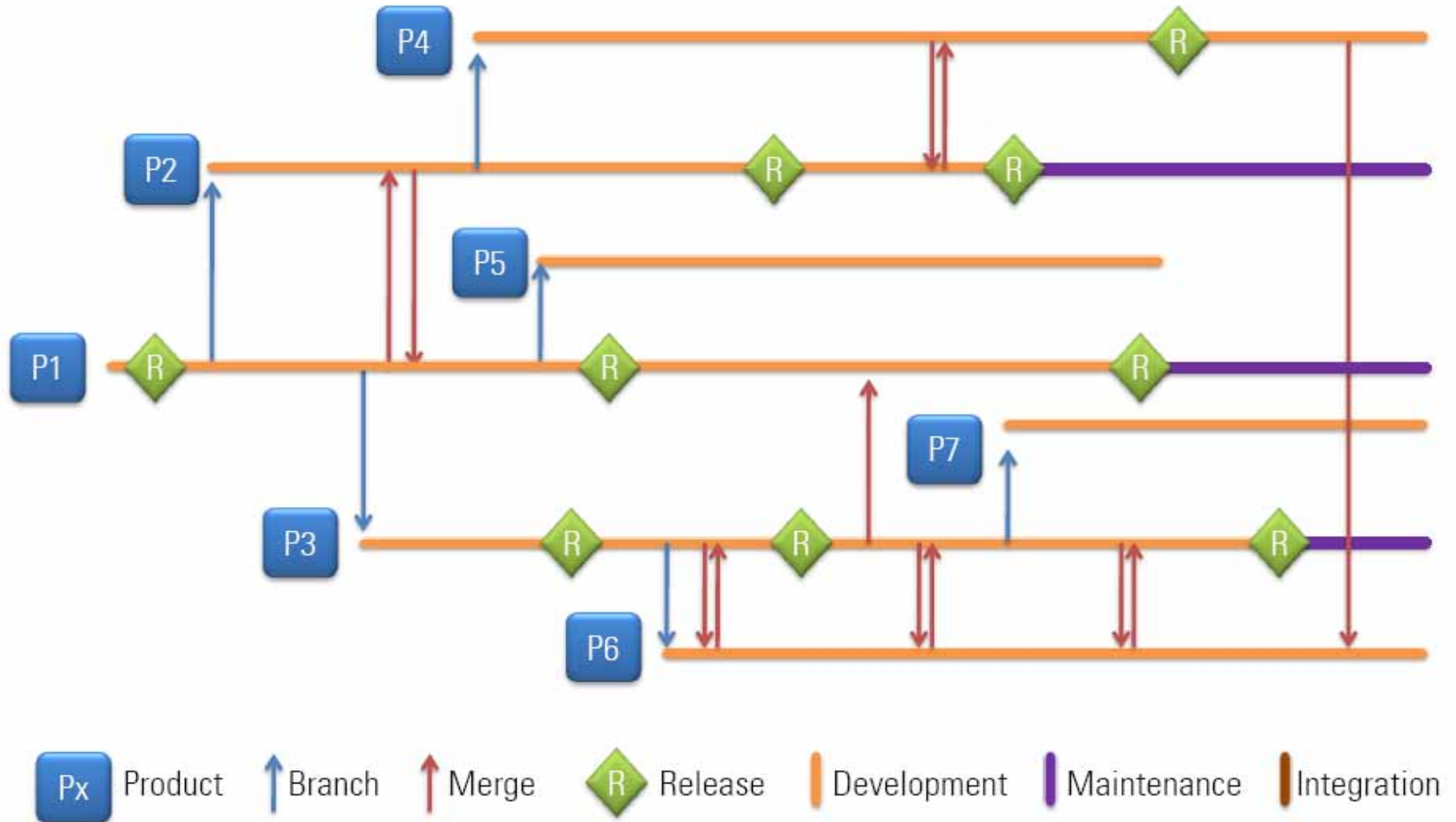


# Variants and Change Management

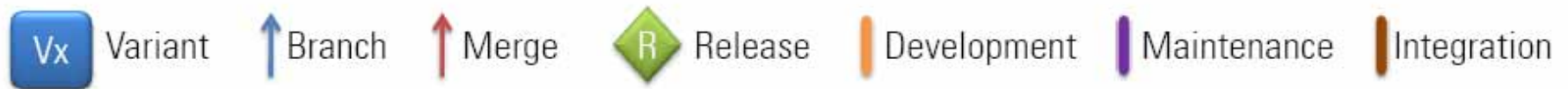
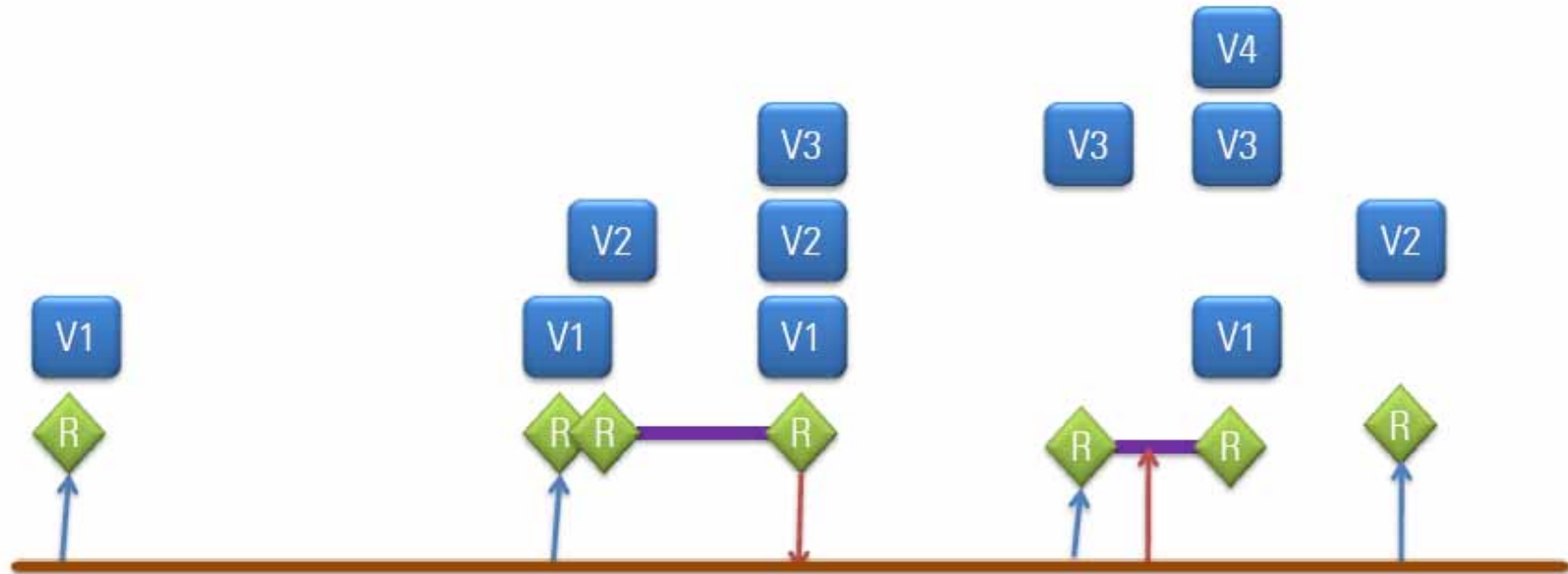


# Variants and Configuration Management

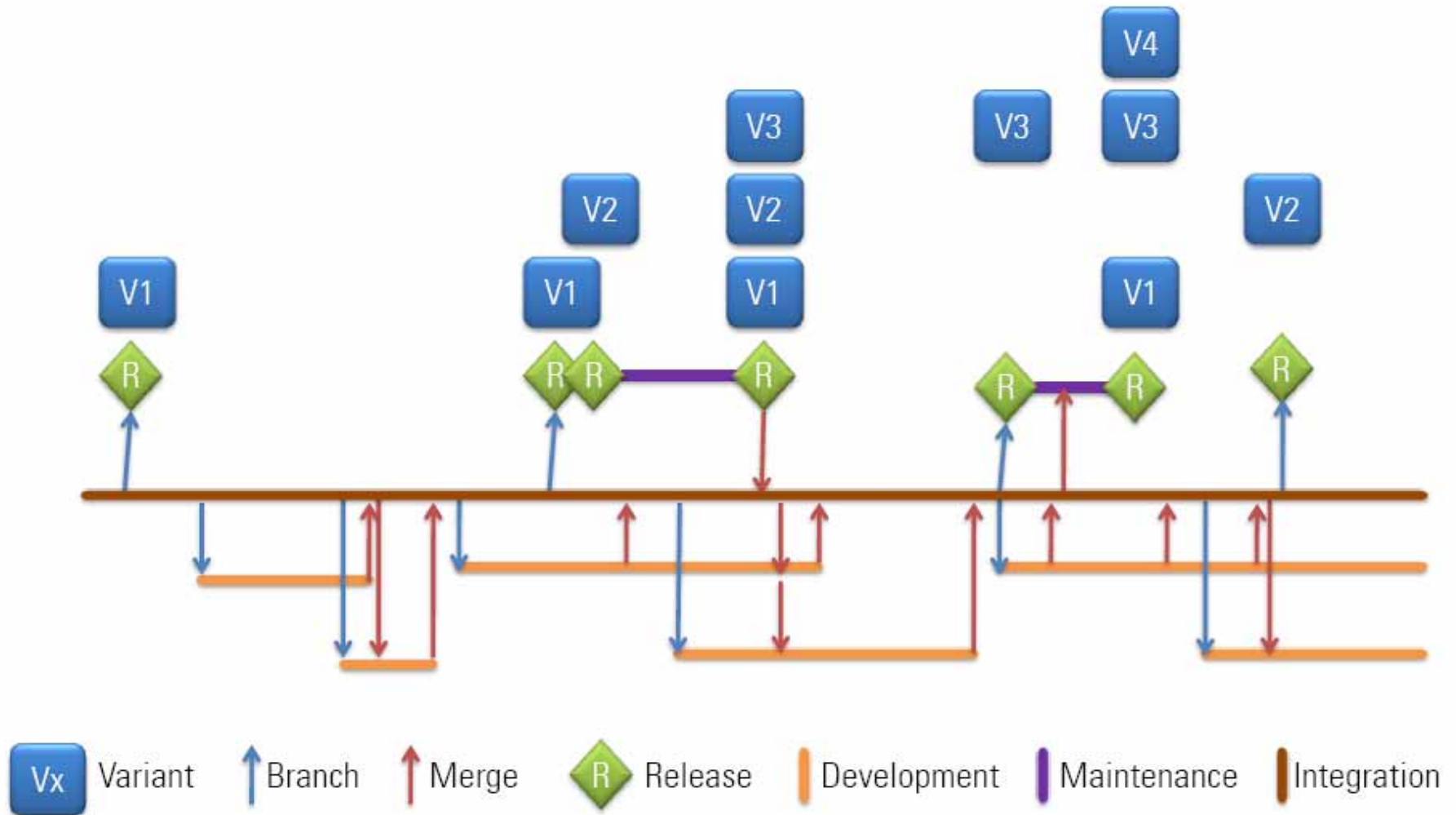
## Clone and Own with Branches

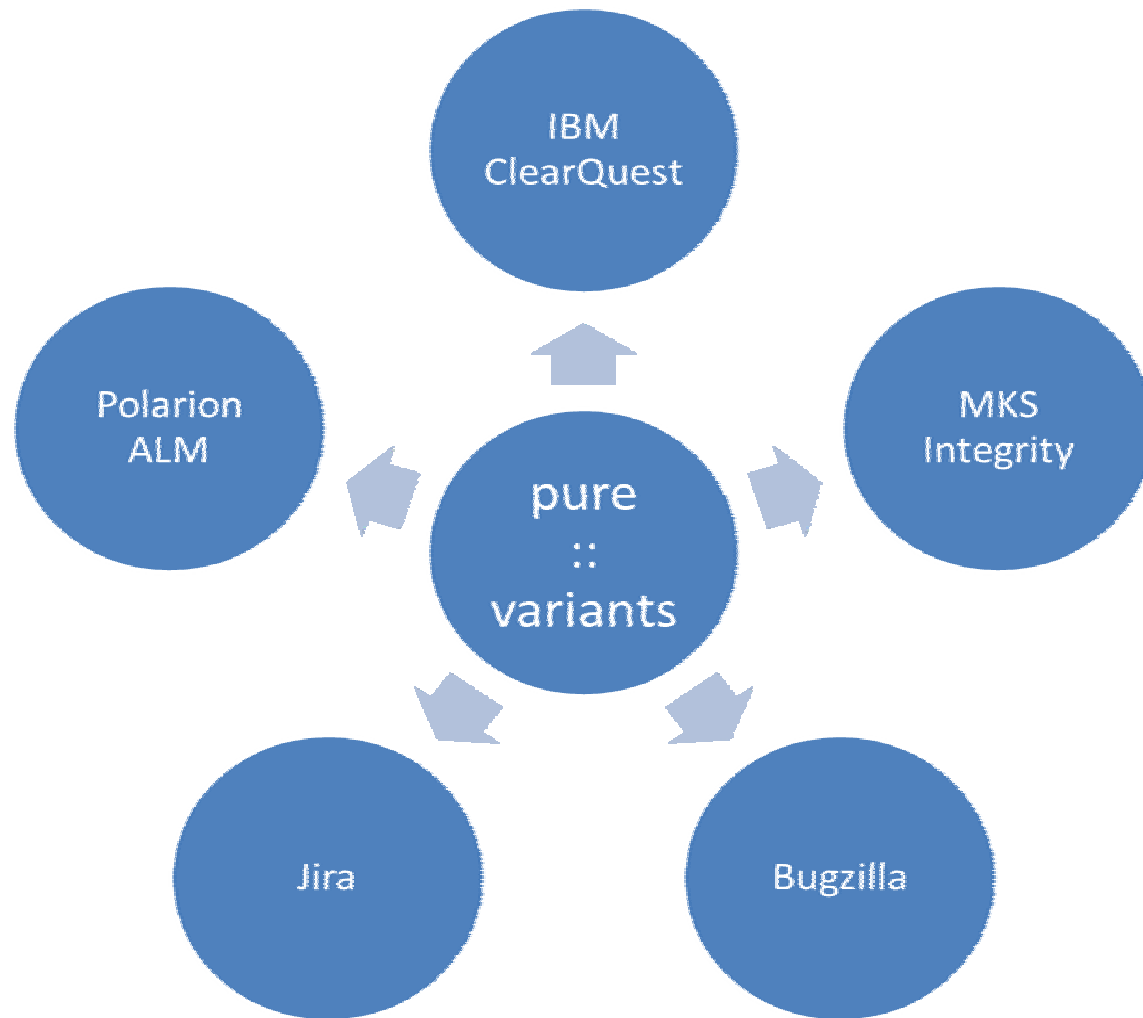


# Variants and Configuration Management



# Variants and Configuration Management







# バリエーション管理ツールに必要なこと

## ・ツールに合わせる必要の無いこと

既存のプロジェクトの構成、成果物をベースに、それらをツールやその手法に合わせてことなく、インクリメンタルにバリエーション管理を、安全に取り入れることが出来ること。

既存のビルドやその他のプロセスの変更は最小限に

## ・大規模・複雑なシステムへの対応

追加のフィーチャにより、取り得るバリエーションは簡単に倍増することを考慮に入れて、複雑で規模の大きいバリエーションのモデリングにも対応できること

## ・派生開発(差分開発)への対応

資産・プラットフォームなどの変更・修正・進化に順応できること

## \* Eclipse のような共通のツールプラットフォーム、ユーザインターフェイス

バグ管理・追跡ツールのようなツールとは、リアルタイムな協調が取れること

## \* APIを介して、あらゆるツールとの統合が誰にでも出来ること

# pure::variants バリエーション管理ツール

## 既存の開発プロセスへ統合が容易

- 既存のコード、ツールチェーンを使い続けられる
- バージョン管理をサポート
- テクノロジーに依存しない(C、Javaなど、あらゆる言語や、HWの管理にも)

## 効果的なバリエーションモデリング

- フィーチャーの組み合わせに関するノウハウを蓄積し、共有できる
- バリエーションの構成を検証

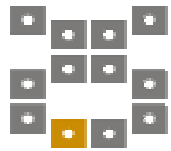
## 自動的にリソースを活用して、バリエーションを生成

- ソースコードをパッケージに(例:バージョン管理ツールから)
- コード、ドキュメント、部品表の生成

# pure::variants バリエーション管理ツール

- プロブレムとソリューションの両モデルを完全に分離。再利用、構成を促進。
- 自動的に機能間の矛盾、不一致を検出。  
バリエーションモデリング時に実施し、早期段階に問題を排除できる
  - 最新鋭の技術で論理的ルールを解釈
- 低い導入障壁：
  - 既存資産、ファイルシステム構造を変えずに導入できる
  - 製品バリエーションは 制約無しに自在に生成できる。複数のバリエーションを同時に生成できる。(ファイル構造などの仕組みに依存することなく、完全に独立した部品としてコア資産を管理し、利用できる)
- エクリプス、クライアント/サーバアーキテクチャをベースに、柔軟に拡張できる
  - ローカル、あるいはサーバ管理してマルチユーザでモデルを管理できるので、大規模システムにもフィットする
  - 多くのエクステンションにより、既存開発環境への統合や改善が行える

# ありがとうございました！



**pure-systems**

- Telephone: +49 391 5445 69 -0

the fast path for all questions – we look forward to your call

- Internet: <http://www.pure-systems.com>

e-mail: [info@pure-systems.com](mailto:info@pure-systems.com)



**FUJI SETSUBI**

販売代理店: 富士設備工業株式会社 電子機器事業部

TEL: 072-252-2128 <http://www.fuji-setsu.co.jp>

担当: 浅野 E-Mail: [yoshio@fuji-setsu.co.jp](mailto:yoshio@fuji-setsu.co.jp)