Engineering Requirements in Product Lines

Institute of Computer Technology

Mike Mannion, Glasgow Caledonian University

Hermann Kaindl, Vienna University of Technology

Structure

- Introduction
- Product Line Models of Reusable Requirements
- Using a Product Line Model for Building a Single Product
- Verifying a Single Product Model
- Selection Decision Interdependencies
- Tool Support
- Lessons Learned
- Summary

Glasgow Caledonian University

TU WIEN

Part I

Introduction to Software Reuse and Product Line Engineering

Reuse Drivers

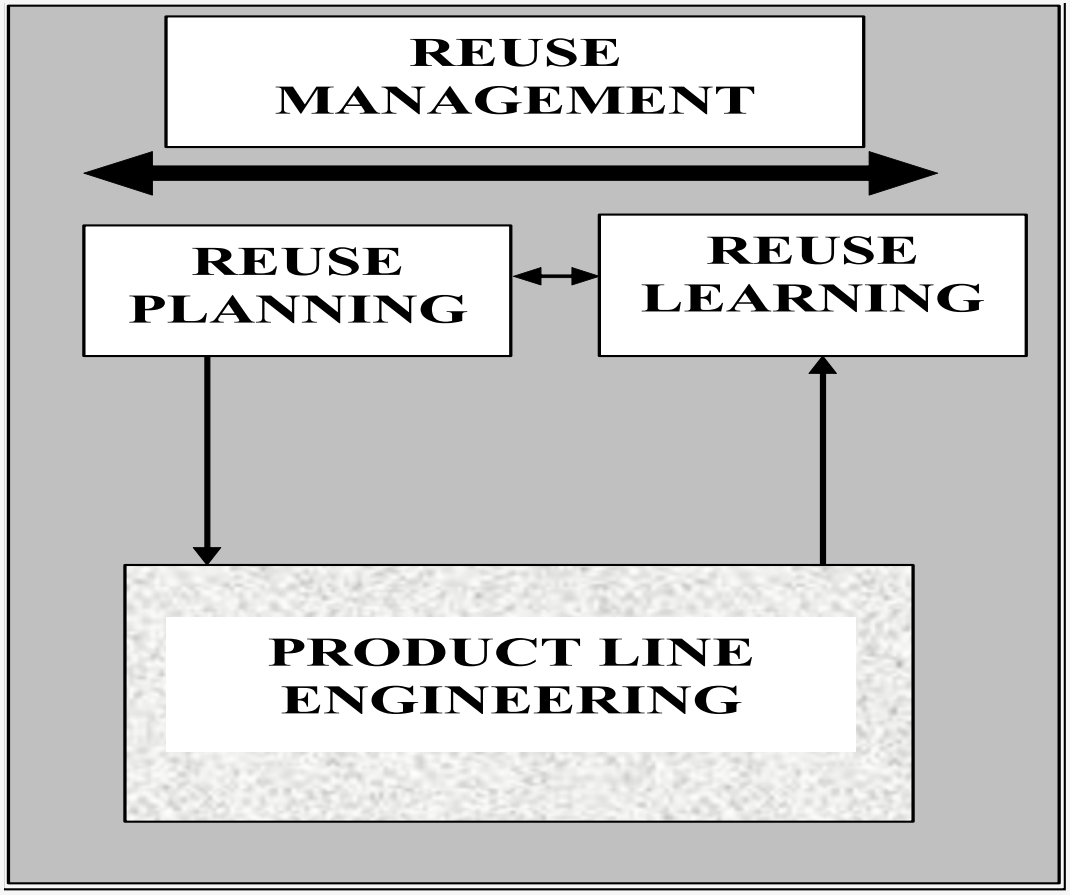
Business

- Mass Customisation
- Time to market
- Response to increase in IT system volume, size, complexity
- Increase quality e.g. reliability, interoperability
- Lower costs
- Greater control over sub-contracted work if insist on use of reusable component
- Staff shortages

Technical

- Object-oriented technologies are a catalyst
- "glue" technologies
- Advances in collaboration technologies: internet, intranet, extranet
- Component development toolkits

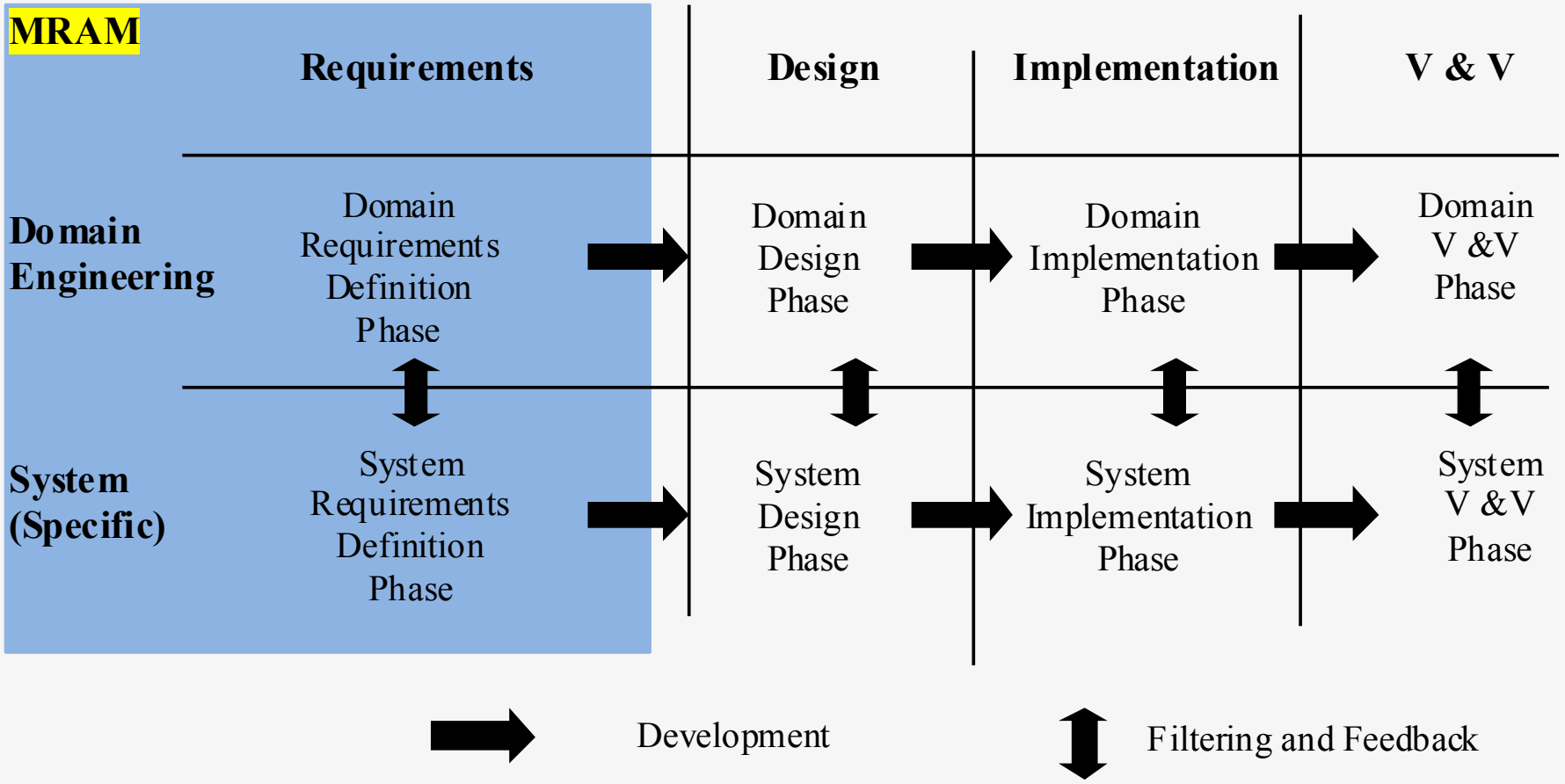
Reuse Framework



Product Line Engineering

- Product Line
  - set of software products sharing a set of common features satisfying the needs of a particular market but containing significant and predictable variability
- Product Line Engineering is a process that delivers software artefacts that can be reused to support the development of new products in the domain
- Product Variation Points, Product Variants
- New product construction grounded in selecting variants at variation points

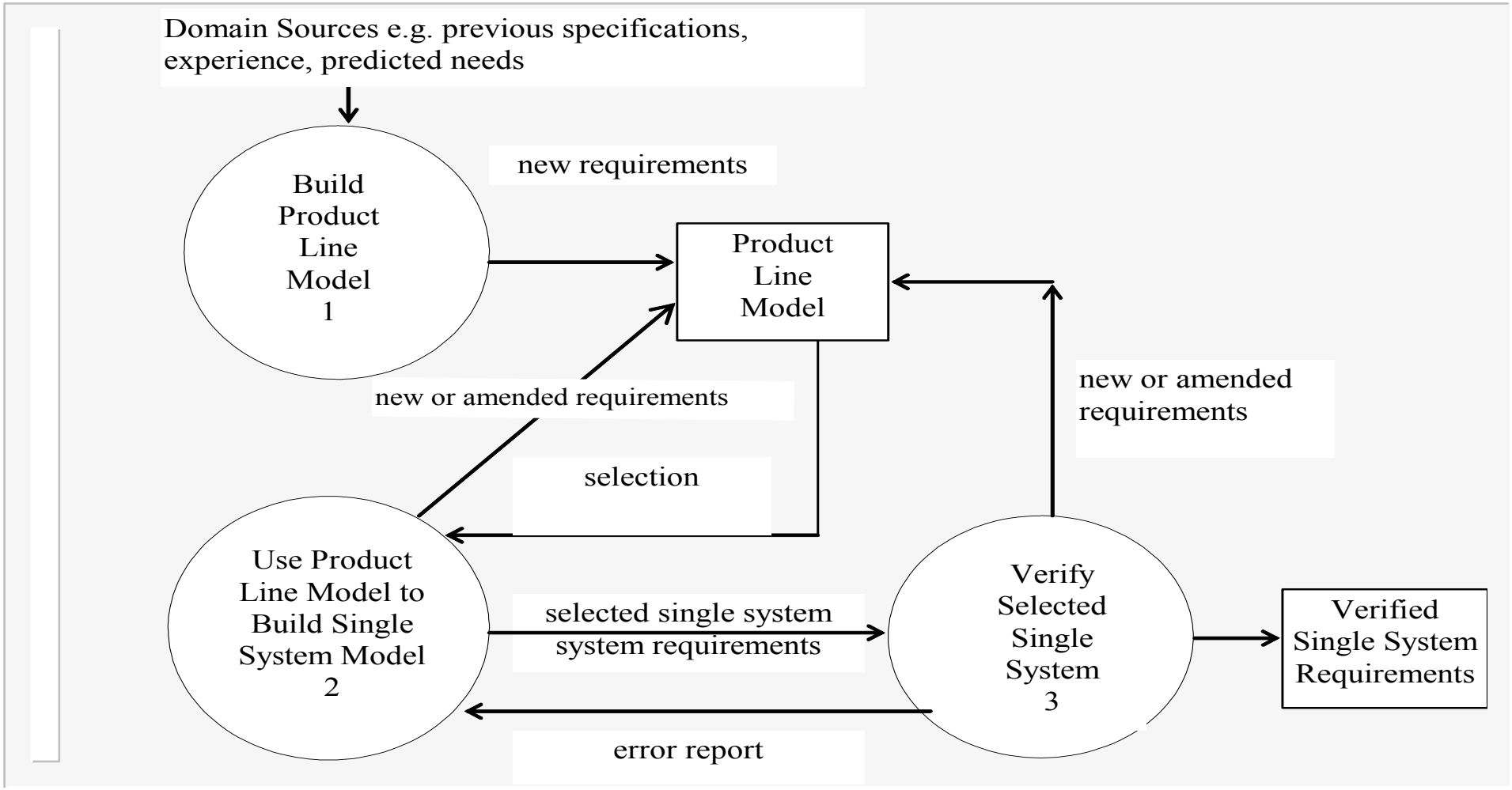
Product Line Engineering



Product Line Engineering Methods

- 3 Tiered SPL [BigLever, ongoing in 2011]
- **F**eature-**O**riented **R**euse **M**ethod (FORM) [Kang 2002]
- **F**eature-**O**riented **D**omain **A**nalysis [Kang 1990]
- **P**roduct **L**ine **P**ractice (**PLP**) Initiative [Clements/Northrop 2001]
  v4.2 on www.sei.cmu.edu/productlines
- **PuLSE** [Fraunhoffer IESE, 2000]
- **R**euse-driven **S**oftware **E**ng **B**usiness [Jacobson et al. 1997]
- **O**rganisation **D**omain **M**odelling [Simos 1996]
- **D**omain-**S**pecific **S**oftware **A**rchitecture [Tracz 1993]
- **Synthesis** [Software Prod Consortium 1993]

MRAM Process Model



Why Requirements Reuse ?

- Well-understood requirements are basis for reusable architecture and components
- There are similarities in existing systems and often similar requirements
- Reuse requirements rationale, terminology, expression, validation information
- Acceptance tests and acceptance test plans and procedures can be reused
- Saves considerable effort

Exercise

Why is Requirements Reuse difficult ?

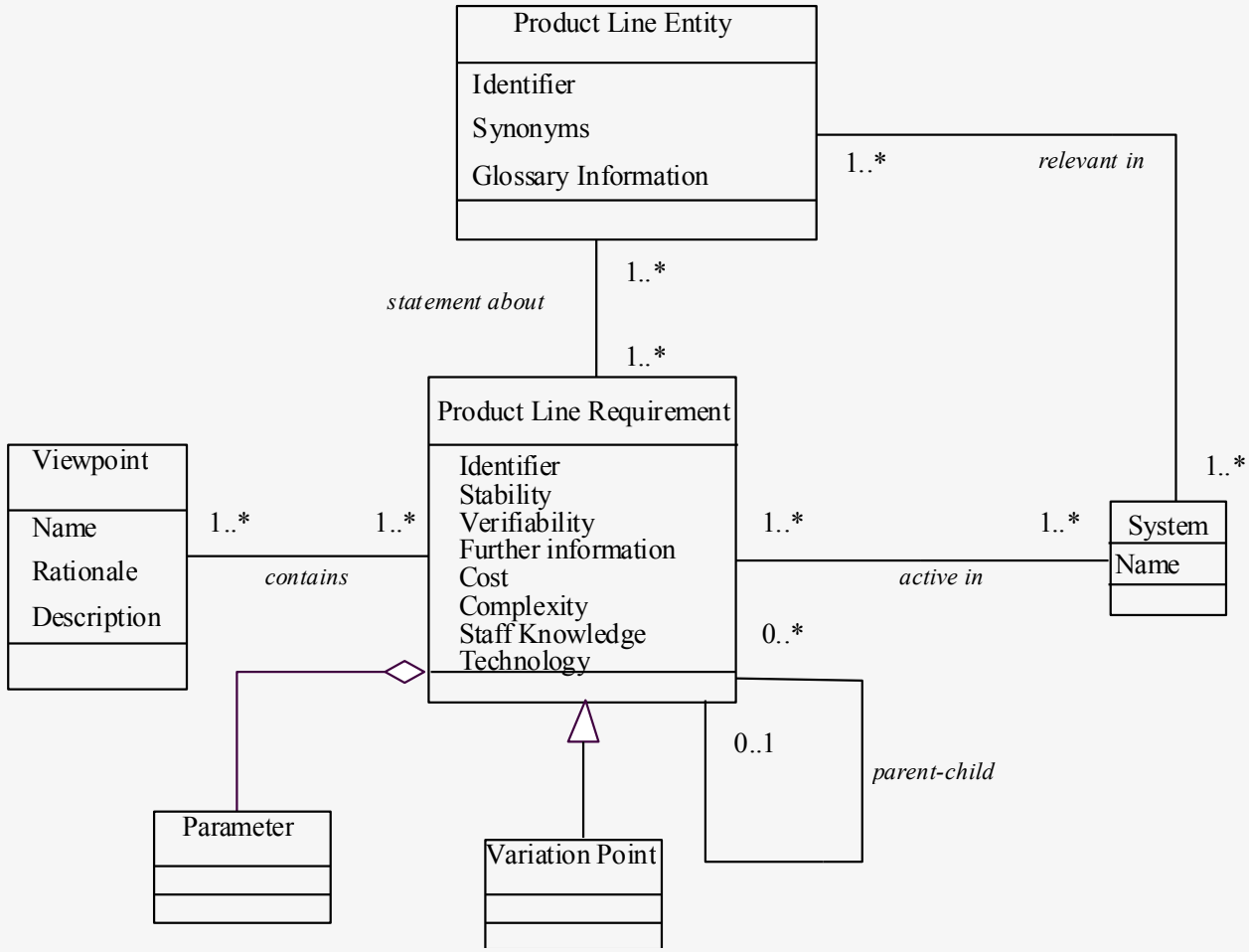
Part II

Product Line Models of Reusable Requirements

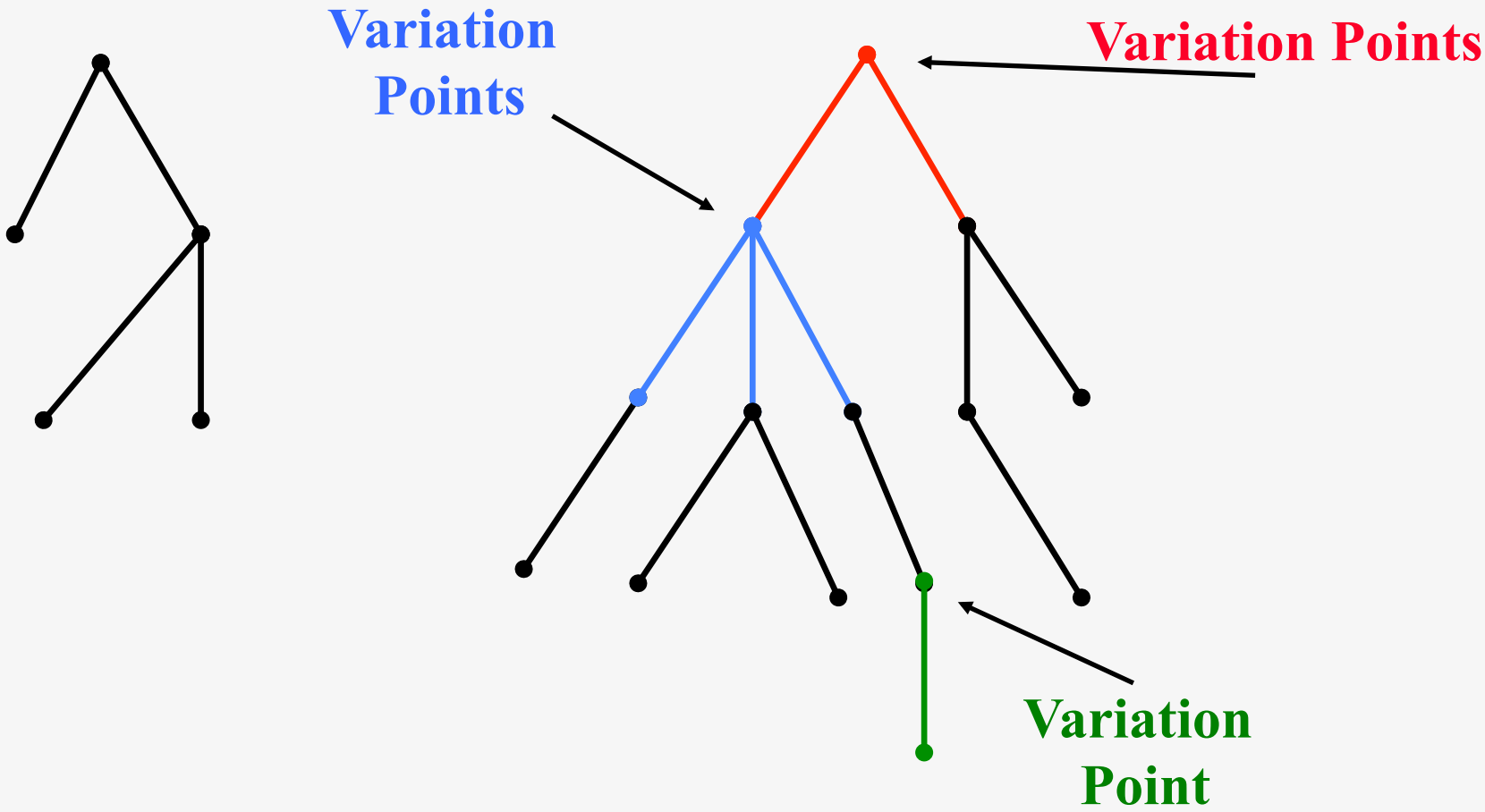
Product Line Model

- Natural language, atomic
- Tree structure
- Classification of reusable requirements
  - common
  - variable (Variation Point)
- Mobile phone example
  - Common: There shall be the capability to make a telephone call.
  - Variable: The mobile phone shall have a TV facility.

MRAM Metamodel



Product Line Model As Tree Structure



Common Requirements

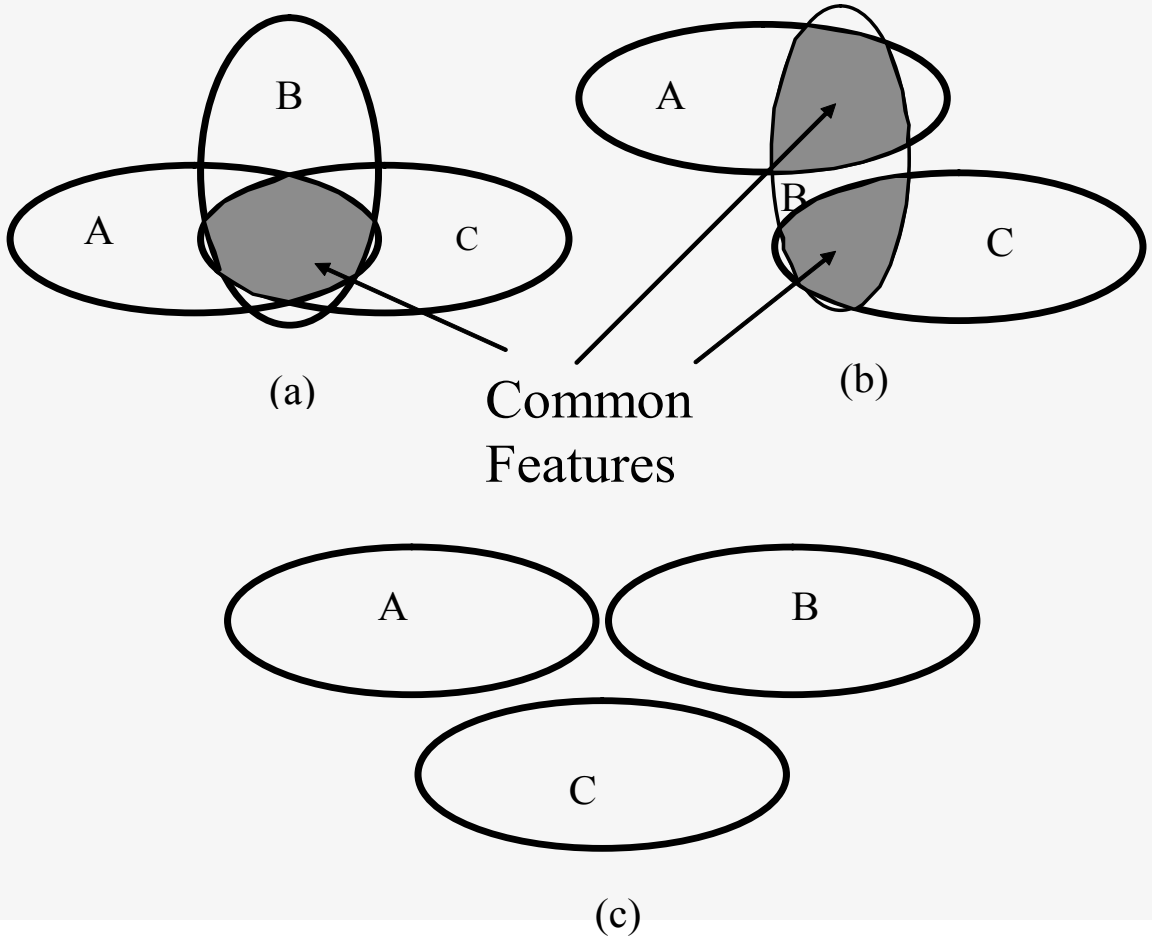
- **REQ 1**
  - There shall be a telephone number address book facility.
- **REQ 1.1**
  - There shall be the facility to add a telephone number.
- **REQ 1.2**
  - There shall be the facility to search for a telephone number.
- **REQ 1.3**
  - There shall be the facility to delete a telephone number.

Parent-Child Relationship

- Often undefined semantics.
- In our experience elaboration on lower level.
- Mutual dependency of parent and child.
- Both “in” or “out”.

Variability Issues

NB: A requirement's variability profile can evolve over time.



Variation Points

- Definition: any requirement which makes a system different from another in the product line.
- Can come from (many) functional or non-functional requirements.
- We model qualitative variation using Variation Points.
- We model quantitative variation using parameters.
- We model qualitative and quantitative variation using parameterised Variation Points.

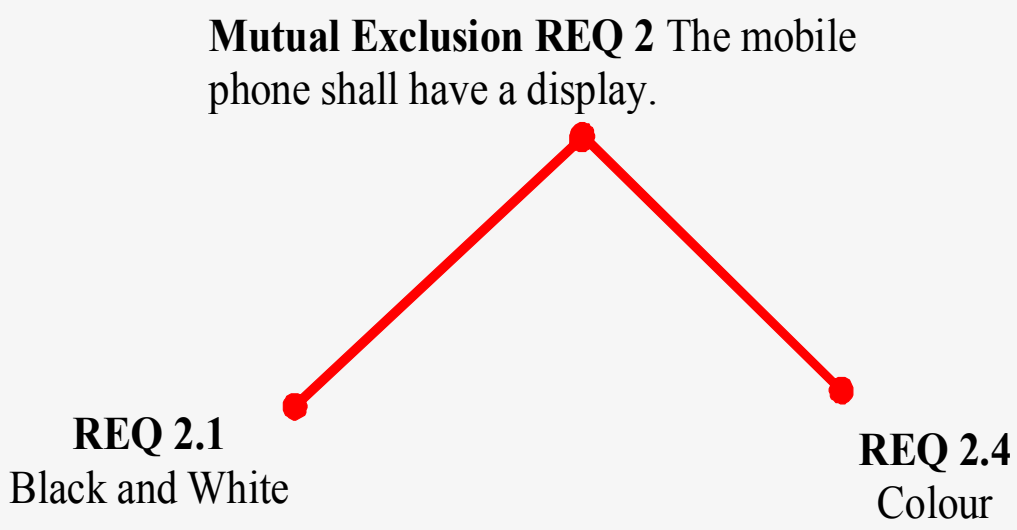
Variation Point Types

1. *Mutual exclusion*: a set of mutually exclusive features from which only one can be used in any system in the domain
2. *List of Alternatives*: a set of features which are optional but not mutually exclusive and at least one will be chose.
3. *Option*: A single optional feature
4. Combination of above.

Mutual Exclusion Example

- **REQ 2**
  - The mobile phone shall have a display.
- **REQ 2.1**
  - The mobile phone shall have a black and white display.
- **REQ 2.2**
  - The mobile phone shall have a colour display.

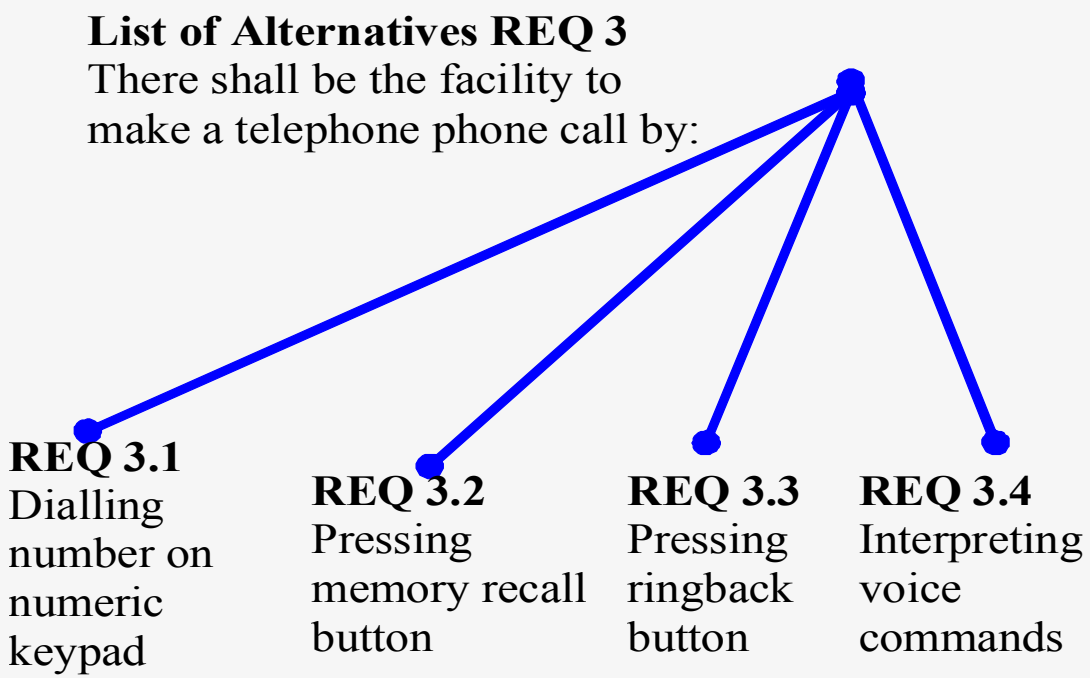
Graphical Representation of a Mutual Exclusion Example



List of Alternatives Example

- **REQ 3**
  - There shall be the facility to make a telephone call.
- **REQ 3.1**
  - A telephone call shall be made by dialling a number on the numeric keypad.
- **REQ 3.2**
  - A telephone call shall be made by pressing a memory recall button to recall a stored number.
- **REQ 3.3**
  - A telephone call shall be made by pressing a ring-back facility to dial the number of the last incoming call.
- **REQ 3.4**
  - A telephone call shall be made by using speech recognition technology to interpret voice commands.

Graphical Representation of a List of Alternatives



Option Example

- **REQ 4**

  The mobile phone shall have an email facility.

Graphical Representation of an Option

Option REQ 4 The mobile phone shall have an email facility.

Variation Point Combination Example

- **REQ 4 (Option)**

  The mobile phone shall have an email facility.

- **REQ 5 (Parent of List of Alternatives)**

  The email facility shall use one of the following protocols.

- **REQ 5.1**

  There shall be the facility to use the Post Office Protocol.

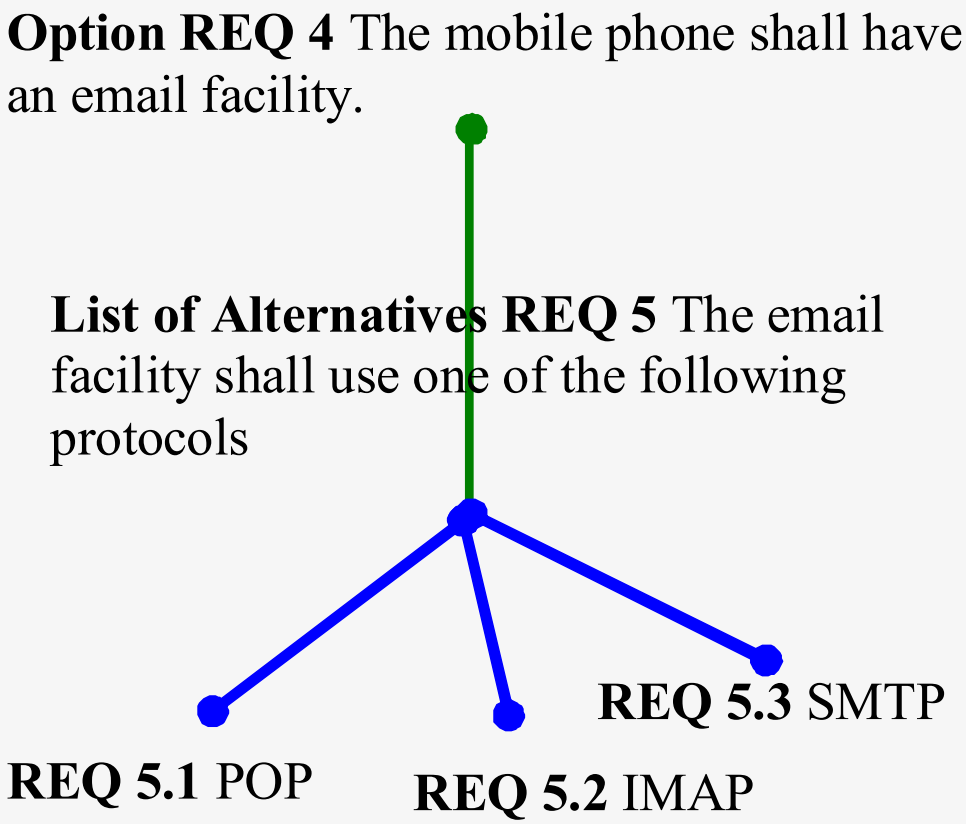
- **REQ 5.2**

  There shall be the facility to use the Internet Message Access Protocol.

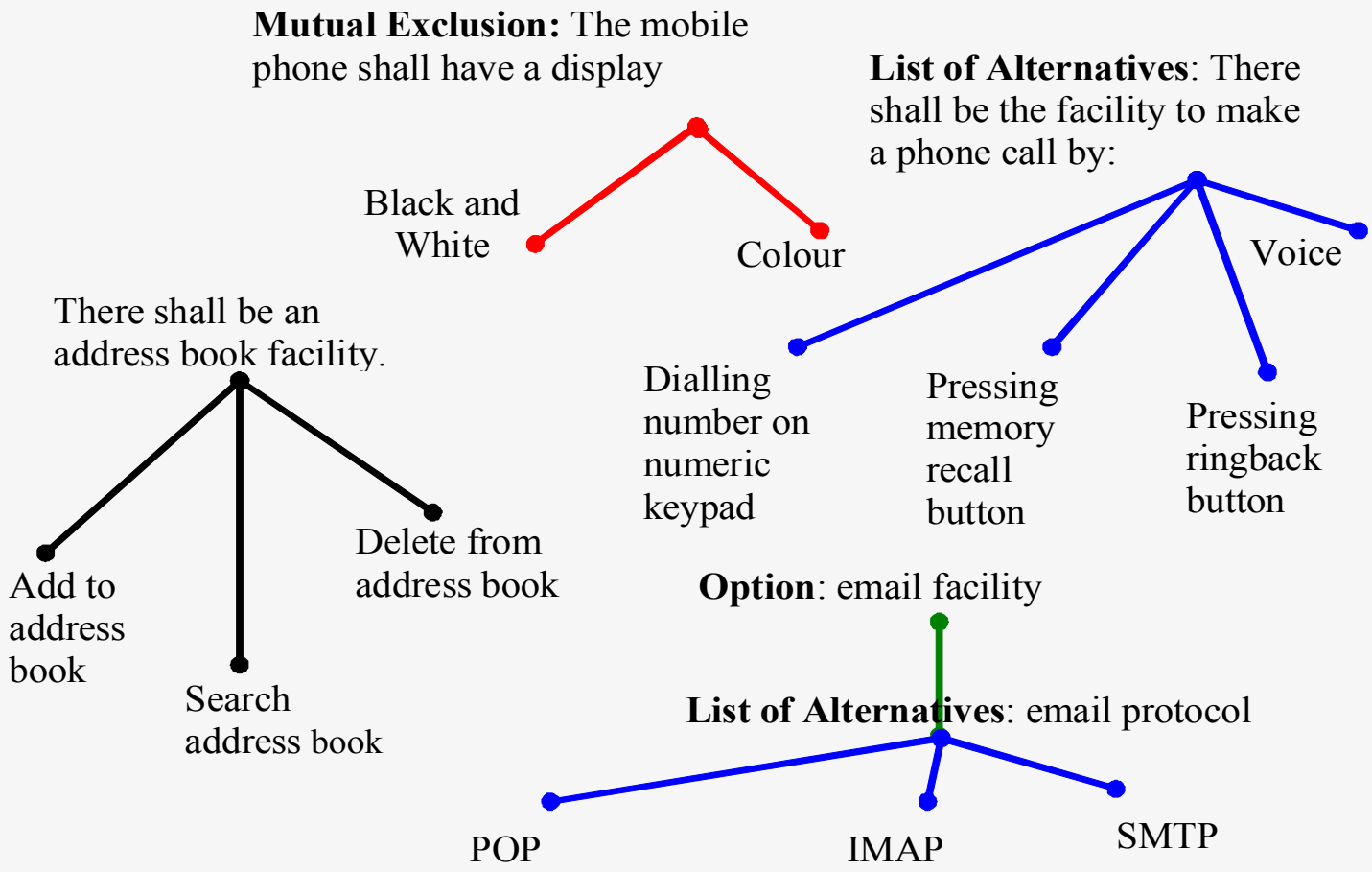
- **REQ 5.3**

  There shall be the facility to use the Simple Mail Transfer Protocol.

Graphical Representation of a Variation Point Combination



Mobile Phone Example



Question ?

What type of Variation Point is the availability of Office applications on mobile phones e.g. Word, Excel, Powerpoint?

Parameterised Requirements

- Example: The mobile phone shall respond to @X commands simultaneously within $Y seconds.
- Global parameters i.e. across many requirements (denoted by @).
- Local parameters i.e. local to this requirement only (denoted by $).
- A parameterised variation point is a variation point that also happens to contain parameters.
- If parameters removed, requirement remains variation point.

Introducing New Requirements Into A Product Line Model

Step 1 Common requirement ?
If yes, go to Step 4.

Step 2 Variable requirement - qualitative ?
If yes, determine whether the requirement is a Mutual Exclusion, List of Alternatives or option and go to Step 4.

Step 3 Variable requirement - quantitative
If yes, determine how many parameters there should be and whether they are local or global.

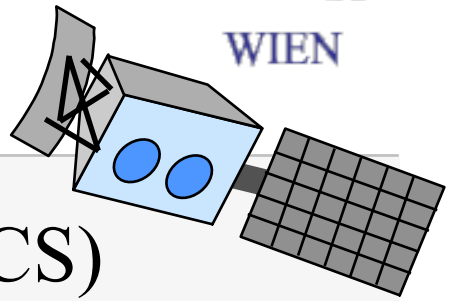
Step 4 Determine if the requirement can be a leaf child node of an existing Variation Point requirement in the hierarchy or whether a new parent requirement must be formed.

End.

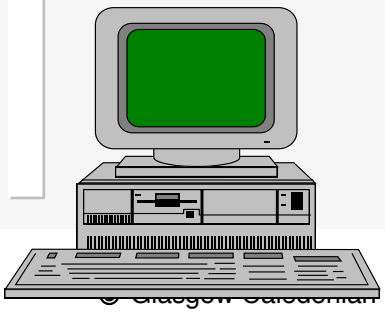
European Space Missions

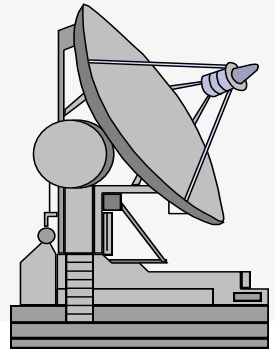
- Cluster II: to investigate solar wind interaction with the Earth's magnetic field in three dimensions.
- European Remote Sensing (ERS) ERS-1 ERS-2: earth observation satellites measuring data and taking images regardless of cloud and lighting conditions.
- Infrared Space Observatory (ISO): infrared waves to investigate molecular structure of space.

Spacecraft Control Systems

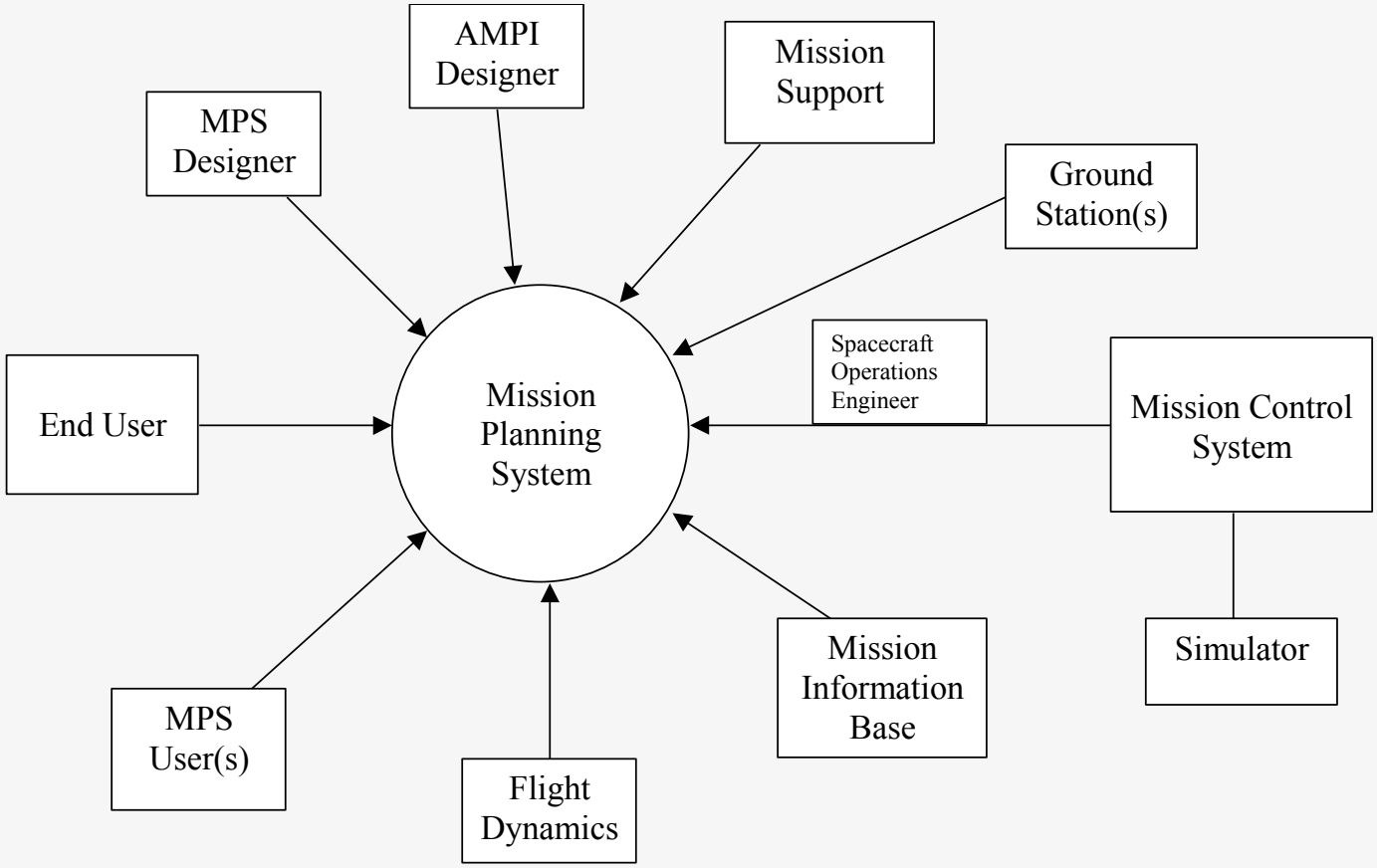


- A Spacecraft Mission Control System (MCS) monitors and controls a satellite.
- Communicates to satellite using spacecraft command sequences.
- MCS has a Mission Planning System (MPS).
- MPS generates commands but can be overridden.





Advanced Mission Planning Infrastructure Viewpoints



Case Study
Mission Planning Systems

- 4 MPS requirement spec:100-200 requirements each.
- Product Line Requirement Spec: 539 requirements; 13 viewpoints.
- 66% requirements emanated from existing requirement specifications
  - of these, 49% were common to more than 1 existing requirements specification.
- 12% requirements brand new; 23% from other sources.
- 263 person-hours reviewing previous documentation and talking to staff.
- 450 person-hours creating, reviewing, writing product line requirement specification.

Part I and II Summary

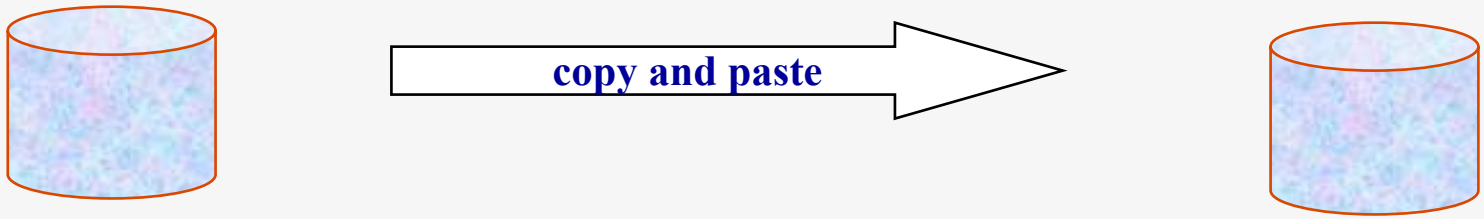
- Plan, commit and organise for Product Line Software Engineering.
- Product line and systems engineering lifecycles.
- Managing variability.
- Structure and examples of product line models of requirements.

Part III

Using a Product Line Model for Building a Single Product

Free Selection

- Free selection means allowing a single system requirements engineer (user) to browse a product line model and simply copy and paste a single requirement from anywhere in the model to the single system model.



Product Line Model

Single System Model

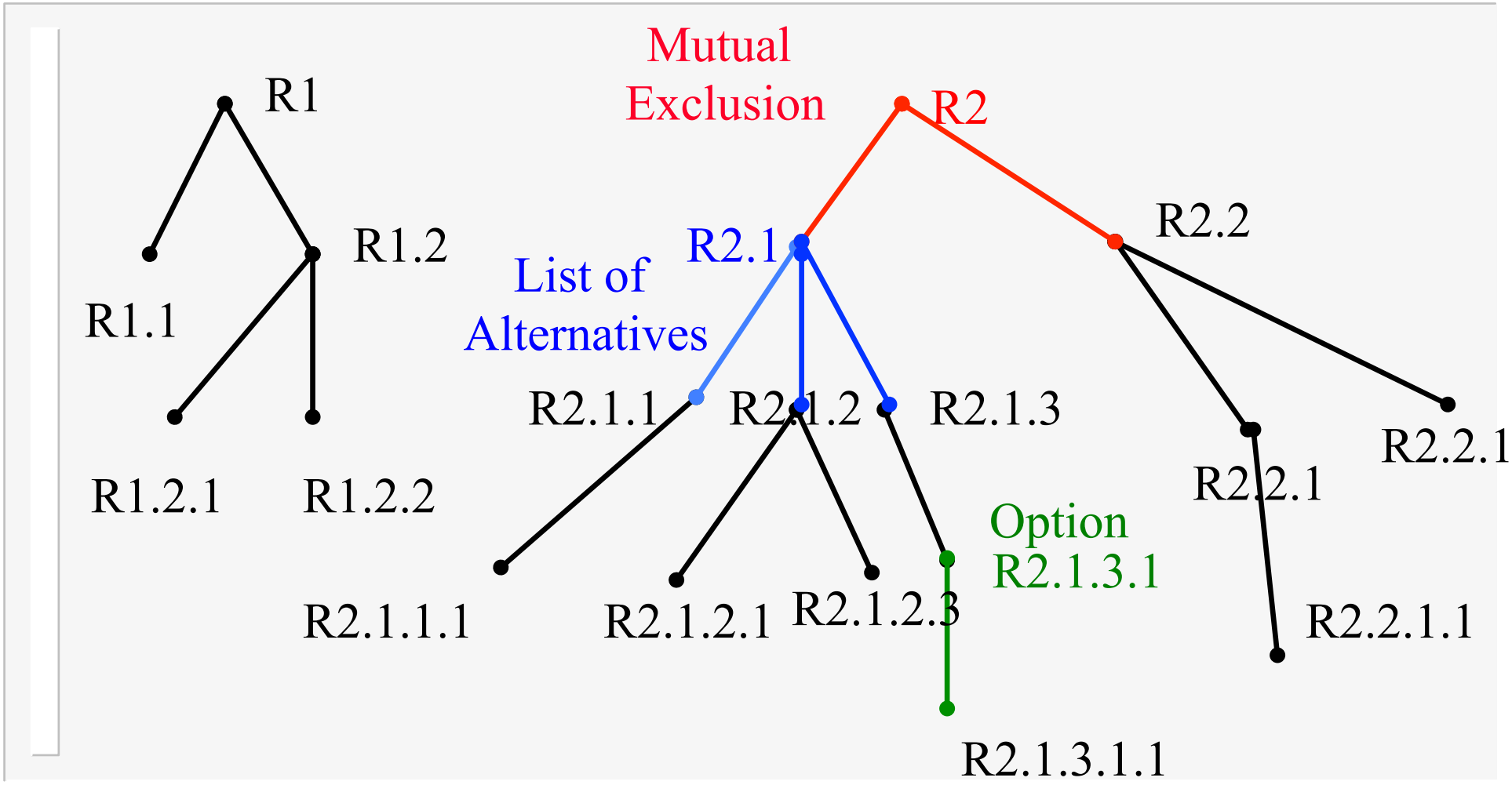
Problems of Free Selection

- Selecting a single requirement is often not sufficient.
- Random choice can mean illegal choice
  - —e.g. 2 mutually exclusive requirements
  - —e.g. not choosing generic requirement.
- Untenable number of choices.
- BUT engineers like freedom of choice!!!

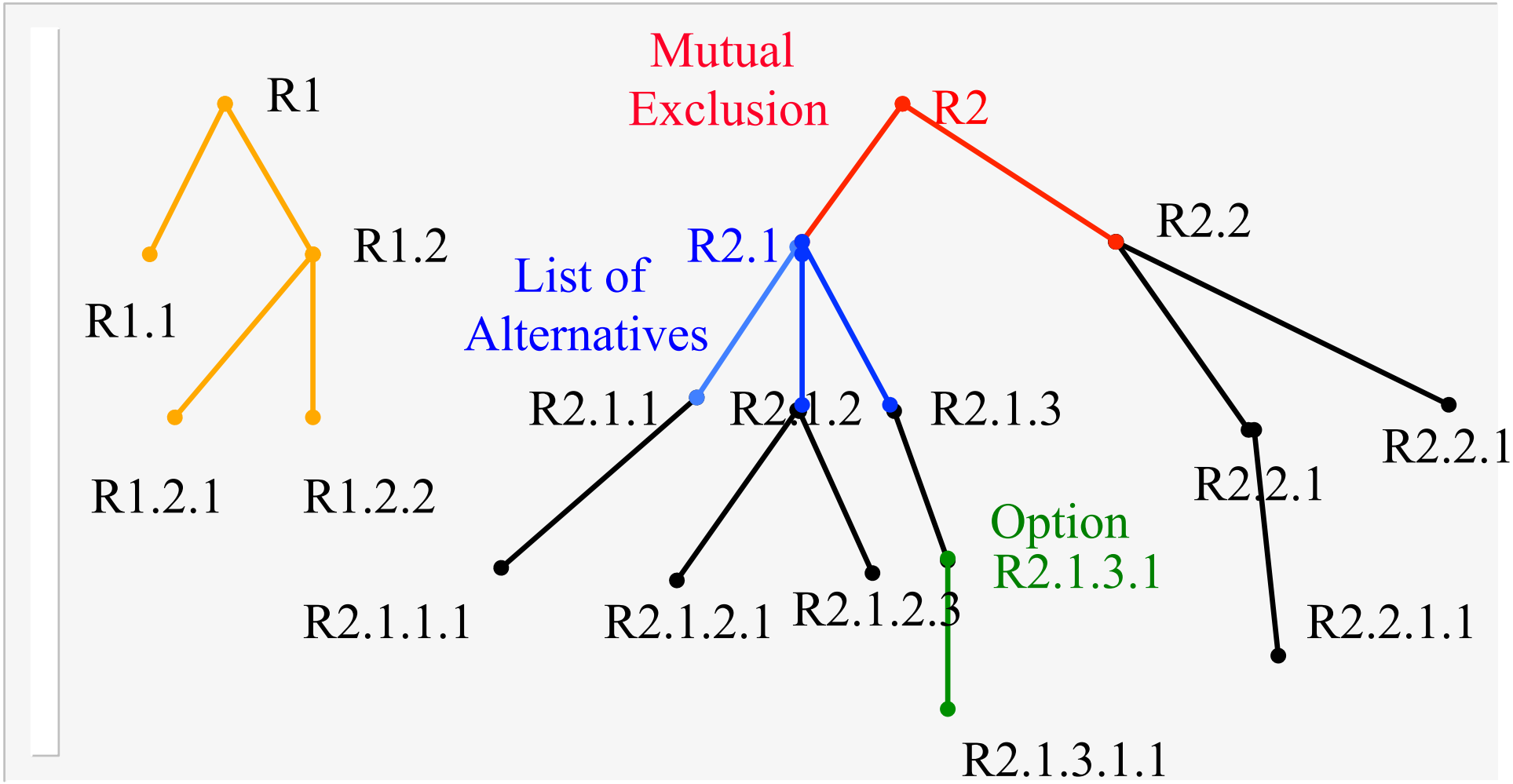
Variation Point-Based Selection

- Use tree structure and Variation Points to direct requirements selection.
- Start at one of the roots.
- Traverse depth first.
- Ask user to make a choice at each Variation Point.
- Common requirements are automatically selected if their parents are already selected or if they are a root node.

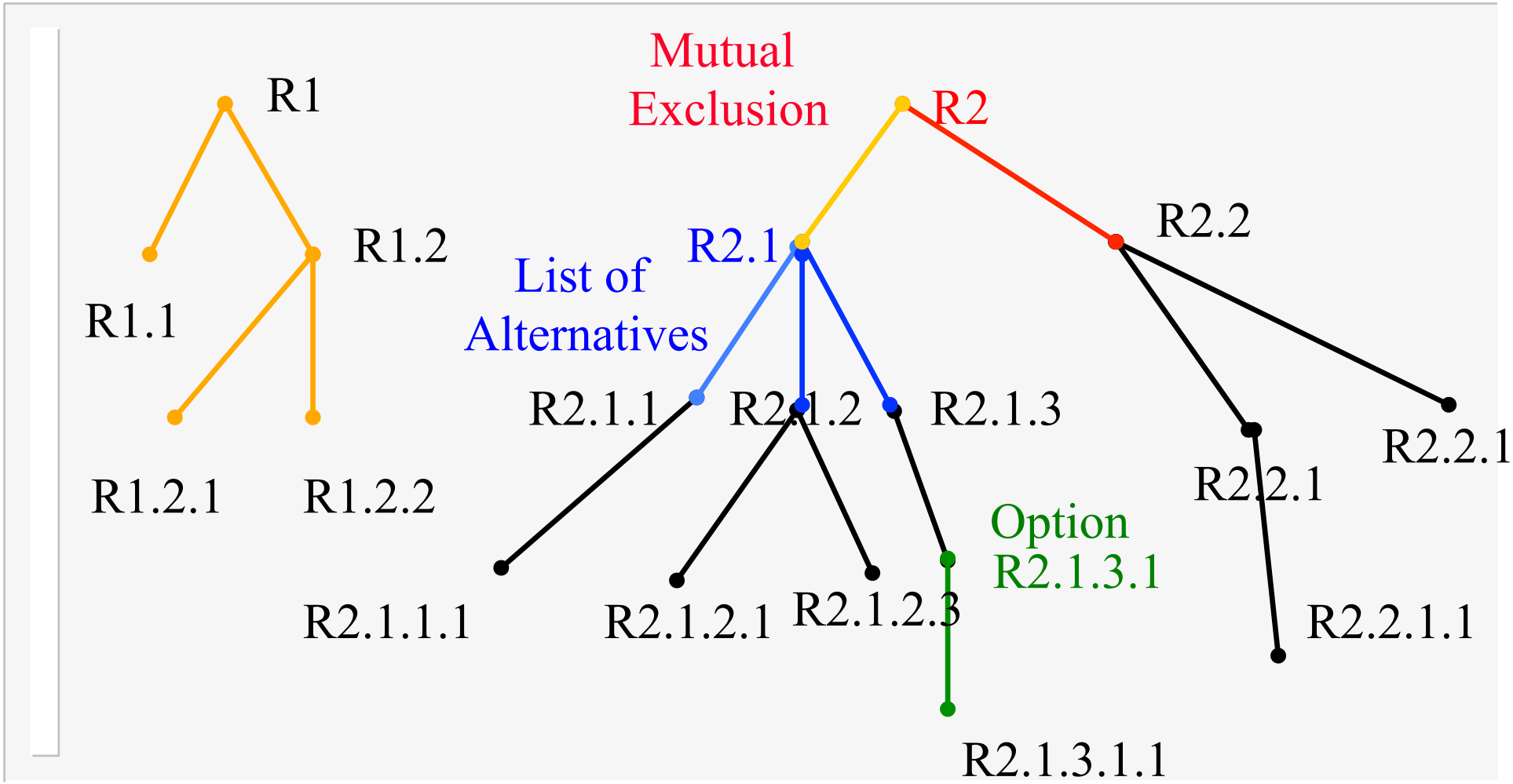
Example



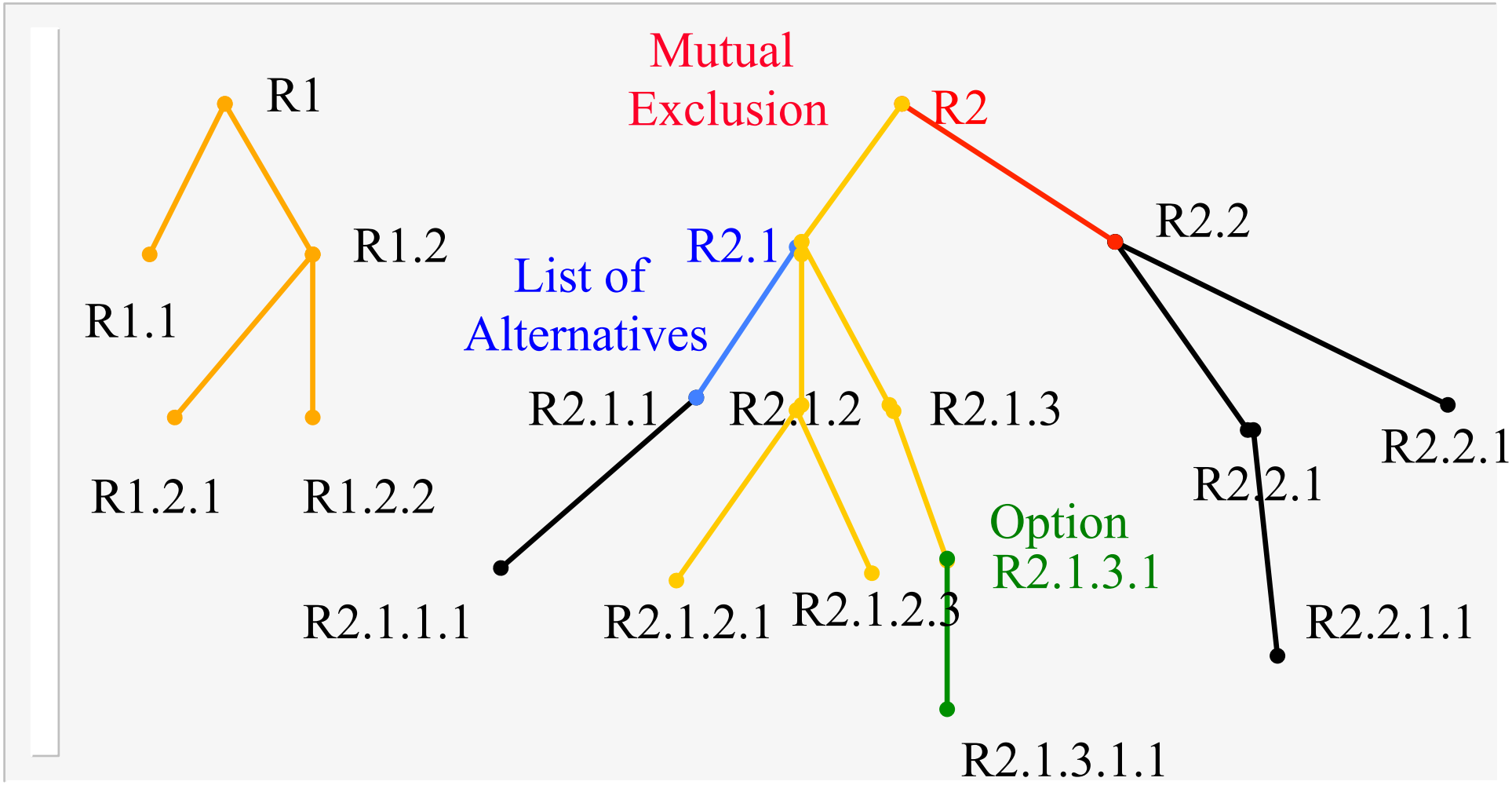
Example



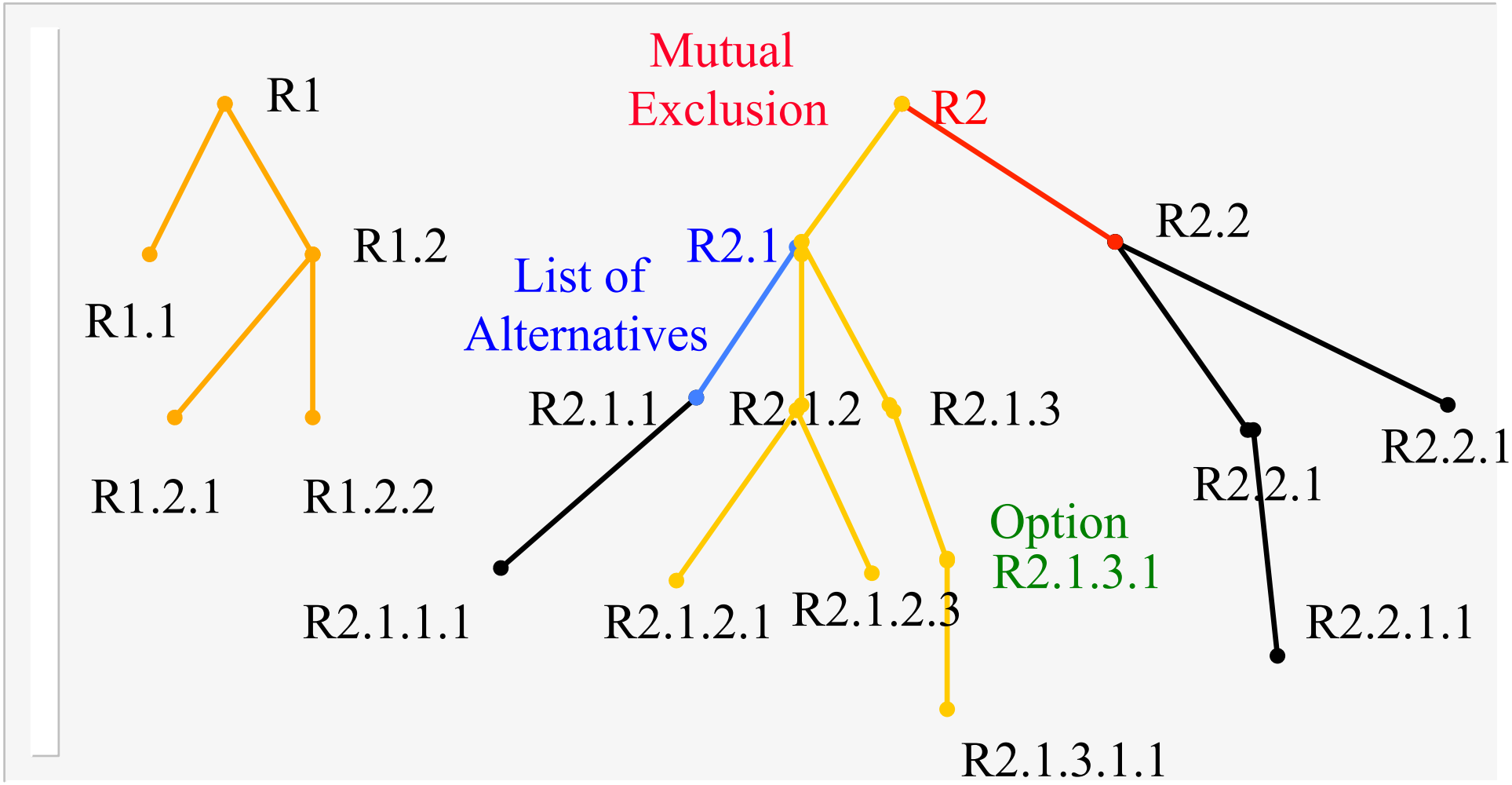
Example



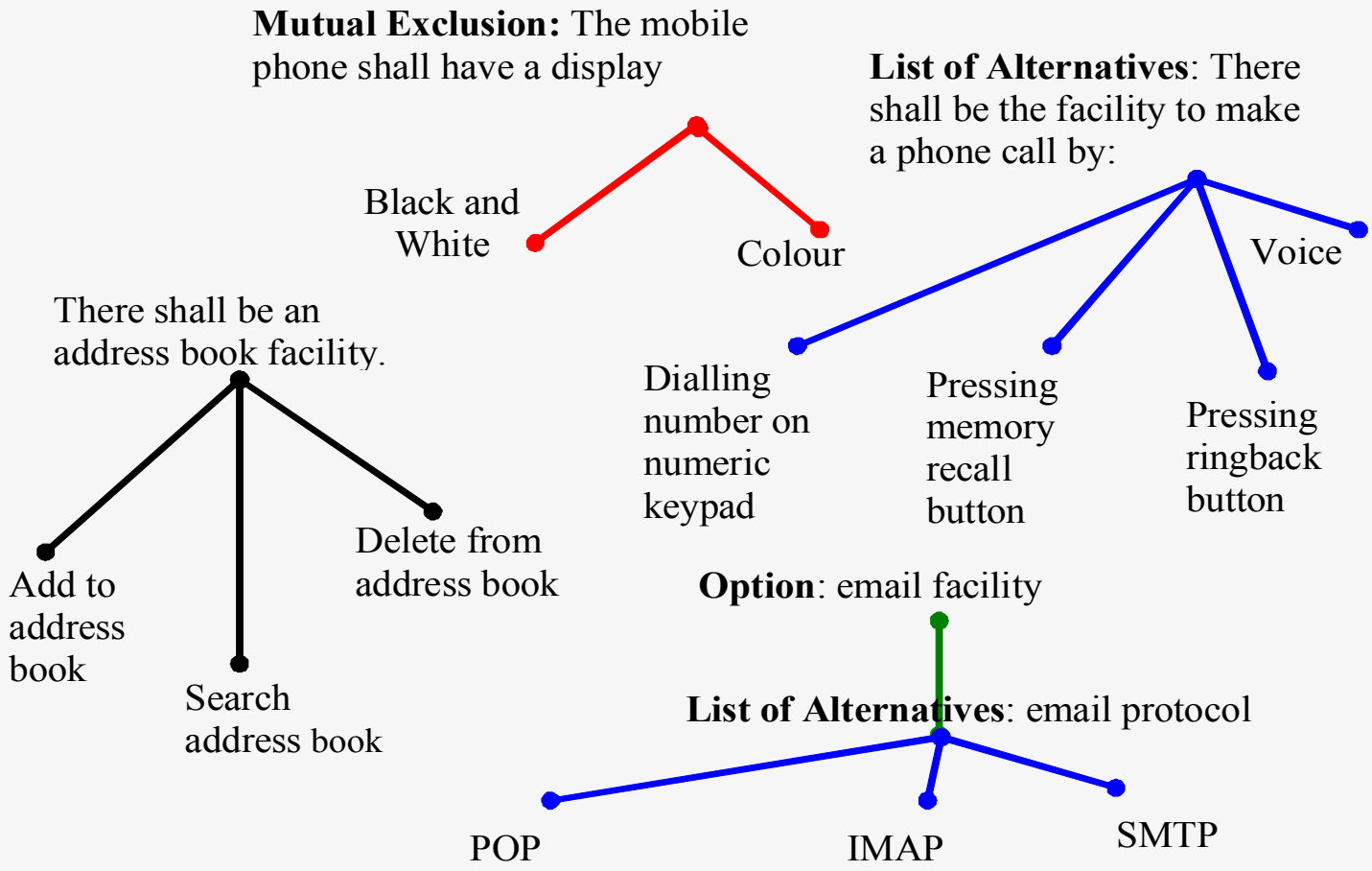
Example



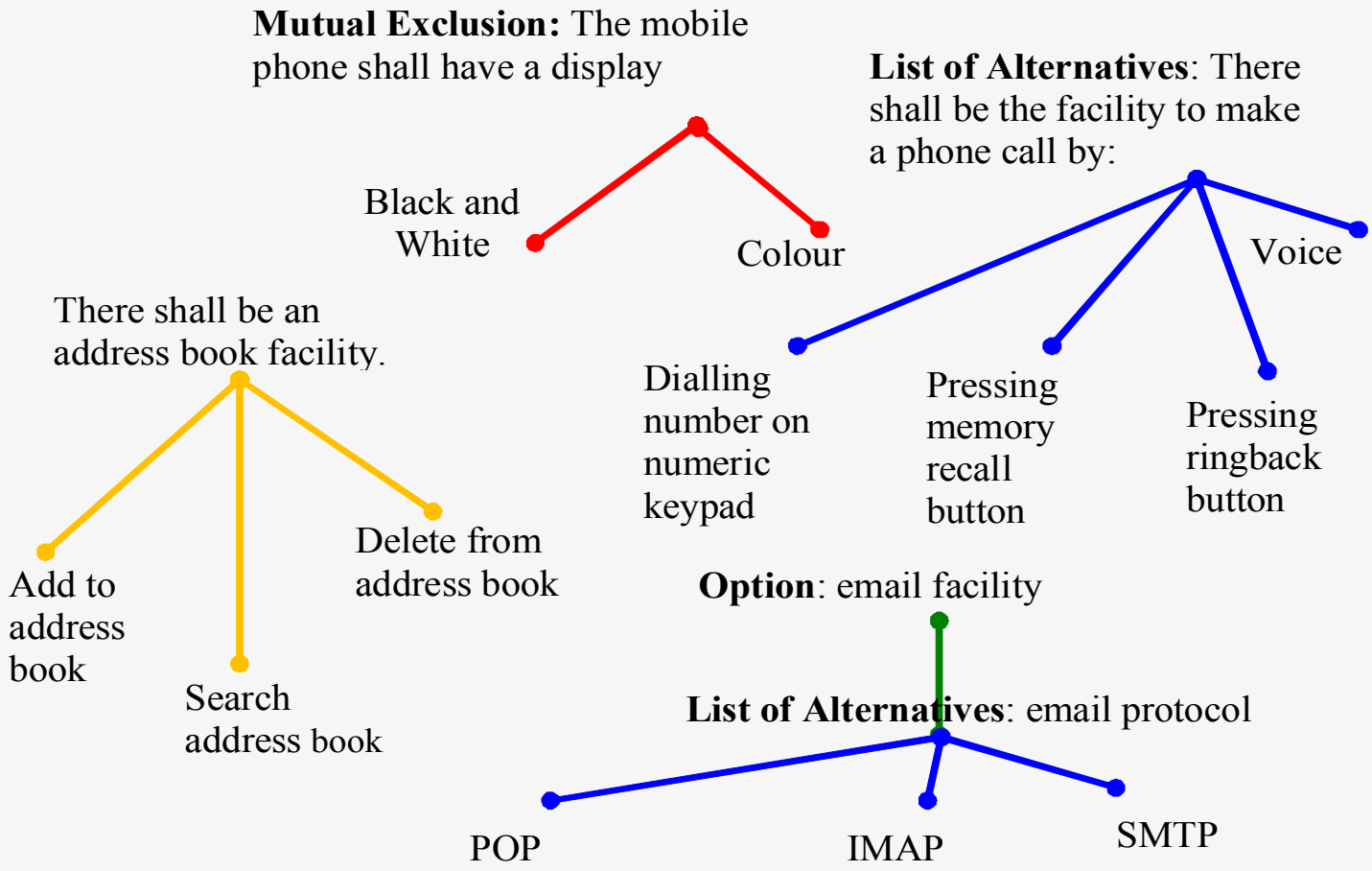
Example



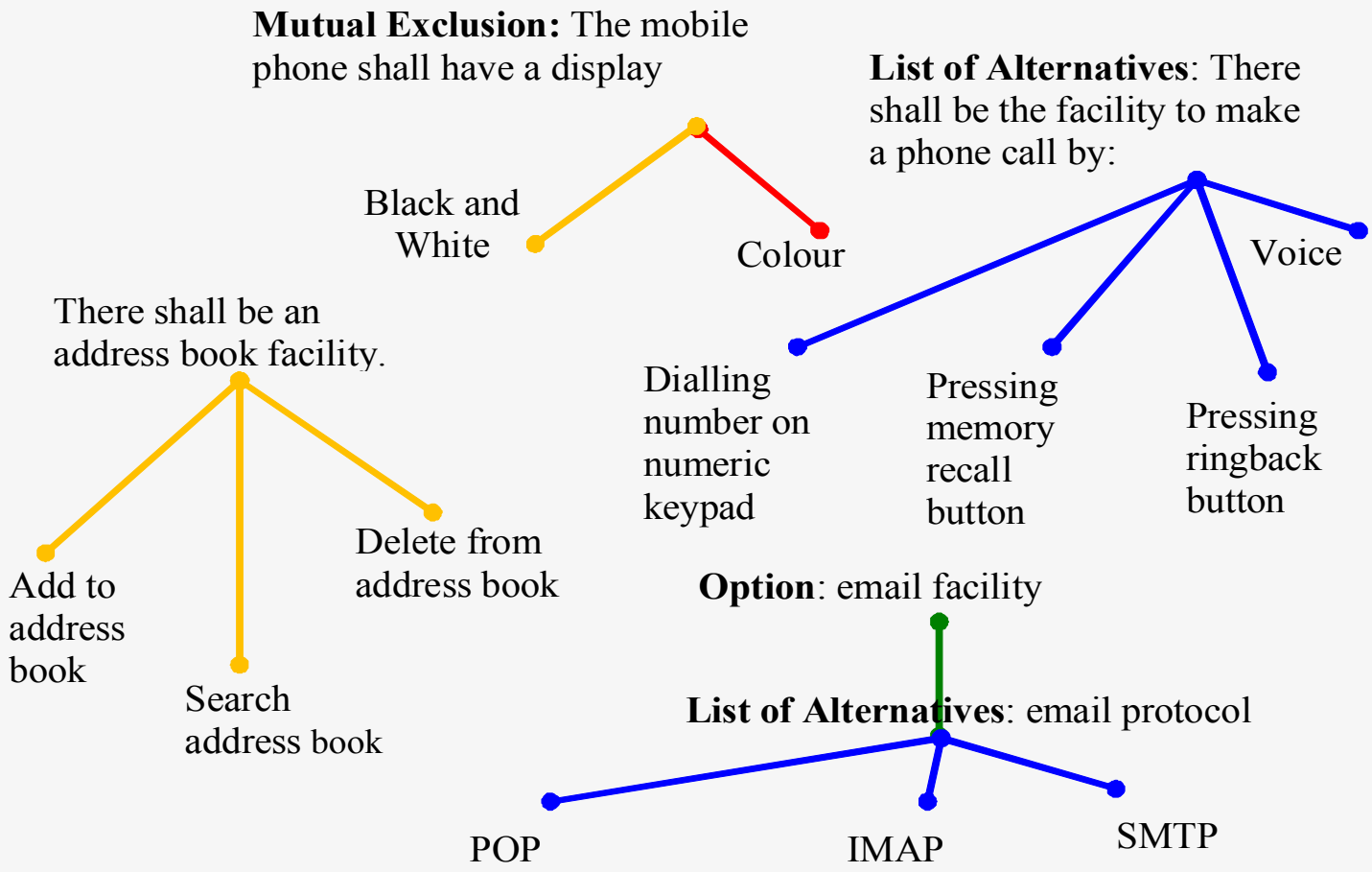
Mobile Phone Example



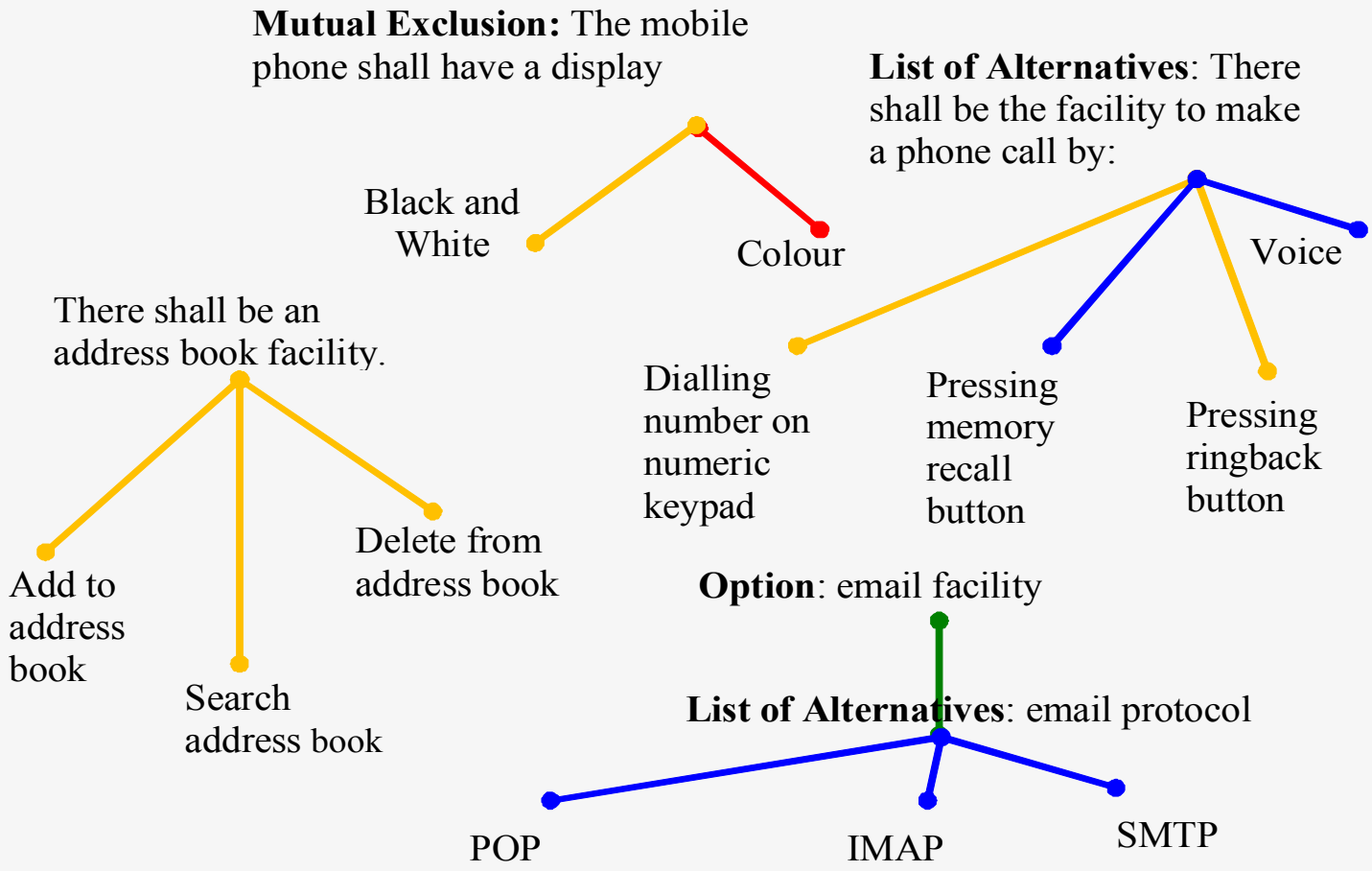
Mobile Phone Example



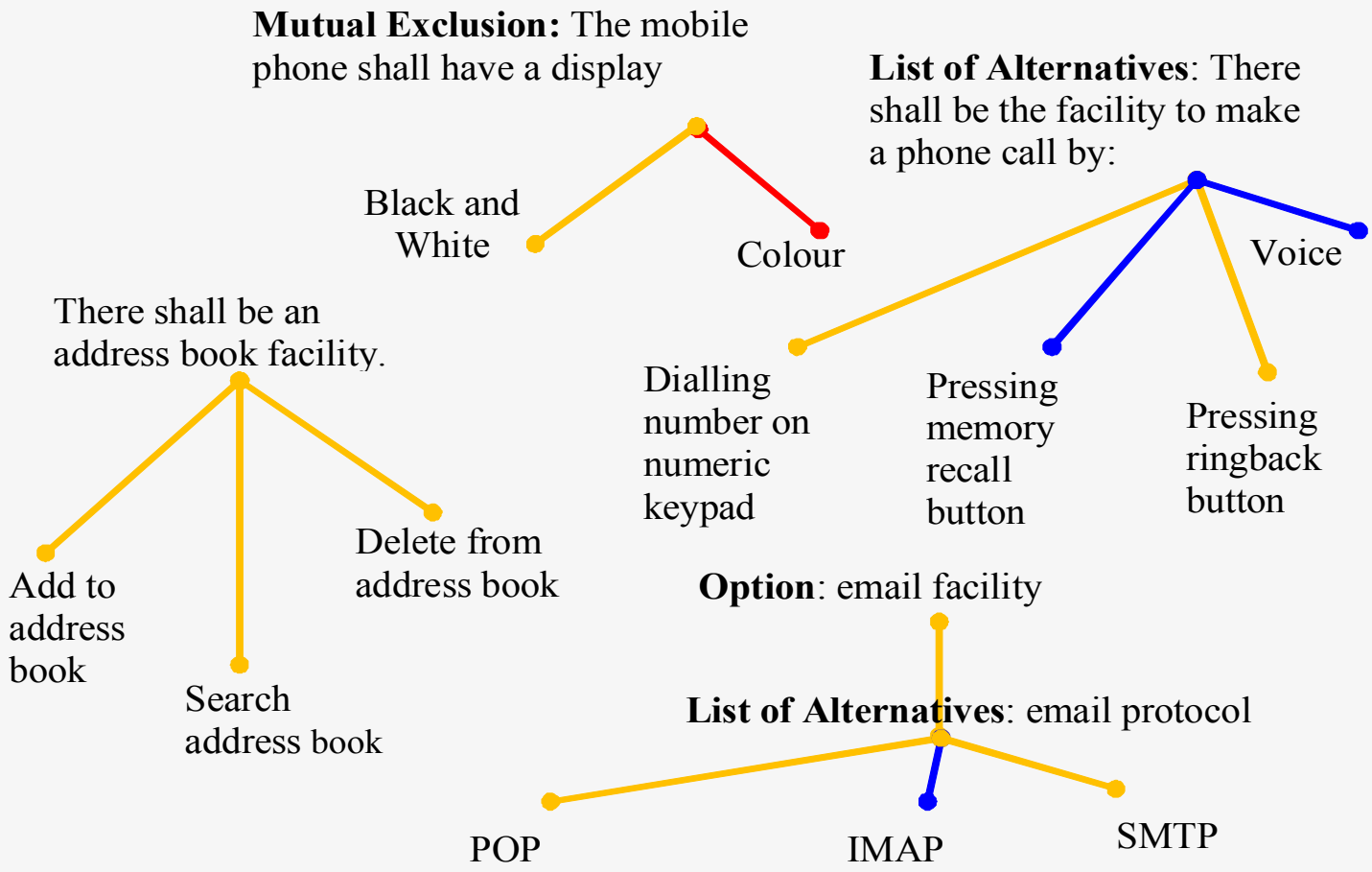
Mobile Phone Example



Mobile Phone Example



Mobile Phone Example



Case Study
SCOS 2000 Commanding Requirements

- Spacecraft commands are organised into *tasks*
  - Tasks can be grouped together into *command sub-systems*
  - Tasks have parameters, which have attributes of type and length so that each task can be executed with varying amounts of information.
- Commanding Specification organised as hierarchy of sections of requirements.
- Requirements well written: 10% re-written to make the implicit variability more explicit.
- 2000 requirements; 778 *commanding* requirements

Case Study

Reusable Requirement Type	Numbers
Common	737
Parameterised Requirement	14
Mutual Exclusion Variation Point	3
List of Alternatives Variation Point	8
Option Variation Point	15
Parameterised Variation Point	1

Advantages of Variation Point-Based Selection

- Time & effort savings
  - —max 35 choices, not 778
  - —system requirements took days not weeks.
- Proportional to number of variation points.
- Time spent on selection implications, not definition, specification, linkage, traceability.
- Selections consistent with the product-line model are made.
- Limited user input because selections pruned.

Part IV
Verifying a Single System Model

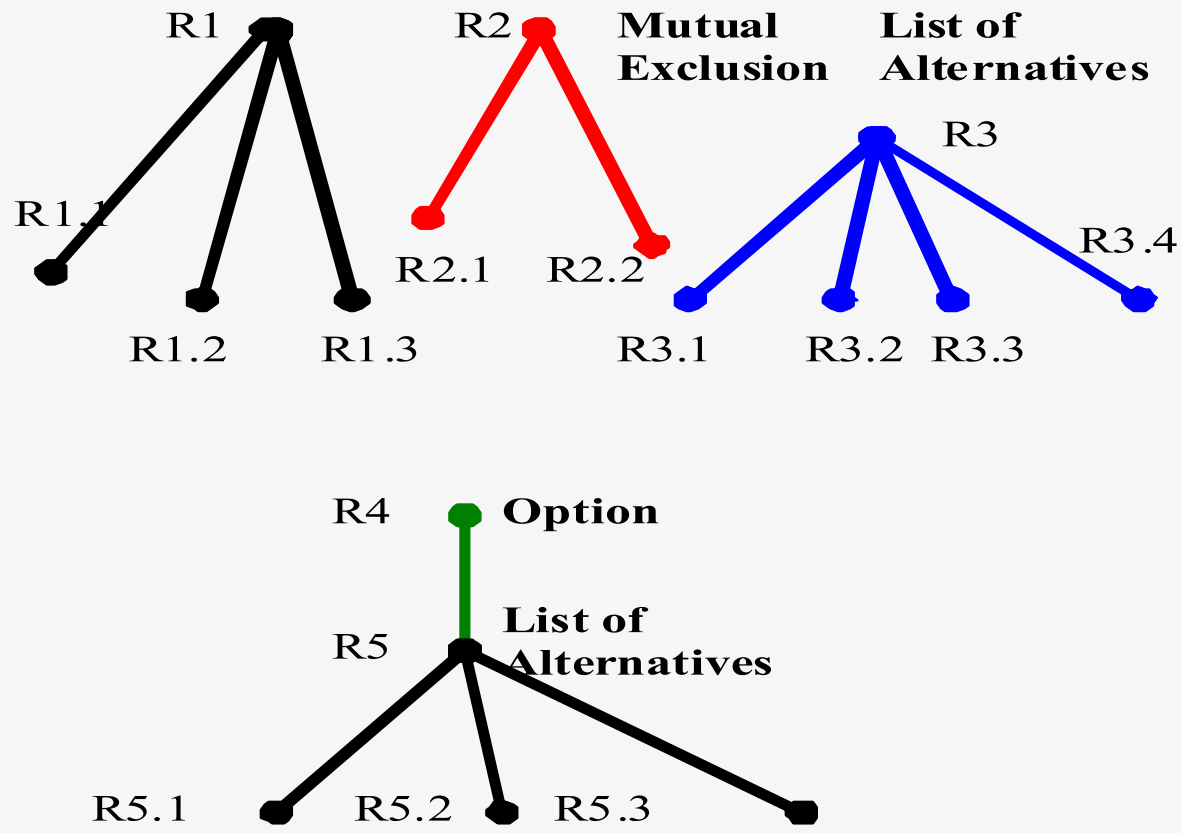
Product Line Model Relations and Formal Definitions

Product Line Relation	Formal Definition
Product line model	$T_1 \wedge T_2 \wedge \ldots \wedge T_n$
Tree (T)	$a_1 \text{ Я } a_2 \text{ Я } \ldots \text{ Я } a_n$
Parent-Child	$a_i \wedge a_j$
Mutual Exclusion Variation Point	$a_i \oplus a_j$
List of Alternatives Variation Point	$a_i \vee a_j$
Option Variation Point	$(a_i \vee \neg a_j)$ if i=j, $(a_i \leftrightarrow a_j)$ if i≠j

Product Line Model Definition

- For a product line model P of product line requirements a logical expression can be defined as
- $E(P) = \{T_1 \wedge T_2 \wedge \ldots \wedge T_n \mid \{T_i = a_{i1}\ Я_{i1}\ a_{i2}\ Я_{i2}\ a_{i3}\ Я_{i3}\ \ldots\ Я_{i(n-1)}\ a_{in};\ a_{ij} = s(r_{ij})$
  - where r_{ij} must be a directly reusable requirement or Variation Point;
  - and $Я_{ij} \in \{Я_{pc}, Я_{sa}, Я_{ma}, Я_{o}\}\ \}$

Example



Mobile Phone Product Line Model

$$((R1 \wedge (R1.1 \wedge R1.2 \wedge R1.3)) \wedge \ldots\ldots\ldots\ldots (T_1)$$

$$(R2 \wedge (R2.1 \oplus R2.2)) \wedge \ldots\ldots\ldots\ldots\ldots\ldots (T_2)$$

$$(R3 \wedge (R3.1 \vee R3.2 \vee R3.3 \vee R3.4)) \wedge \ldots . (T_3)$$

$$(R4 \leftrightarrow (R5 \wedge (R5.1 \vee R5.2 \vee R5.3))) \ldots\ldots . (T_4)$$

Free Selection: Example 1

Suppose the selected requirements are:

(R1, R1.1, R1.2, R1.3, R2, R2.1, R3, R3.1, R3.2, R4, R5, R5.1)

The product line logical expression becomes:

(TRUE ∧ (TRUE ∧ TRUE ∧ TRUE)) ∧……..(T_1)

(TRUE ∧ (TRUE ⊕ FALSE)) ∧……..(T_2)

(TRUE ∧ (TRUE ∨ TRUE ∨ FALSE ∨ FALSE)) ∧……(T_3)

(TRUE ↔ ((TRUE ∧ (TRUE ∨ FALSE ∨ FALSE)))……..(T_4)

(T_1), (T_2), (T_3) and (T_4) each evaluate to TRUE.

Hence $T_1 \wedge T_2 \wedge T_3 \wedge T_4$ evaluates to TRUE.

Free Selection: Example 2

Suppose the selected requirements are:

(R1, R1.1, R1.2, R2, R2.1, R3, R3.1)

R1.3 not selected

Product line logical expression is:

(TRUE ∧ (TRUE ∧ TRUE ∧ FALSE)) ∧……..(T_1)

(TRUE ∧ (TRUE ⊕ FALSE)) ∧……..(T_2)

(TRUE ∧ (TRUE ∨ FALSE ∨ FALSE ∨ FALSE)) ∧……..(T_3)

(FALSE ↔ (FALSE ∧ (FALSE ∨ FALSE ∨ FALSE)))……..(T_4)

(T_2), (T_3) and (T_4) each evaluate to TRUE but (T_1) evaluates to FALSE because the directly reusable requirement R1.3 was not selected. Hence $T_1 \wedge T_2 \wedge T_3 \wedge T_4$ evaluates to FALSE.

Verification Technique

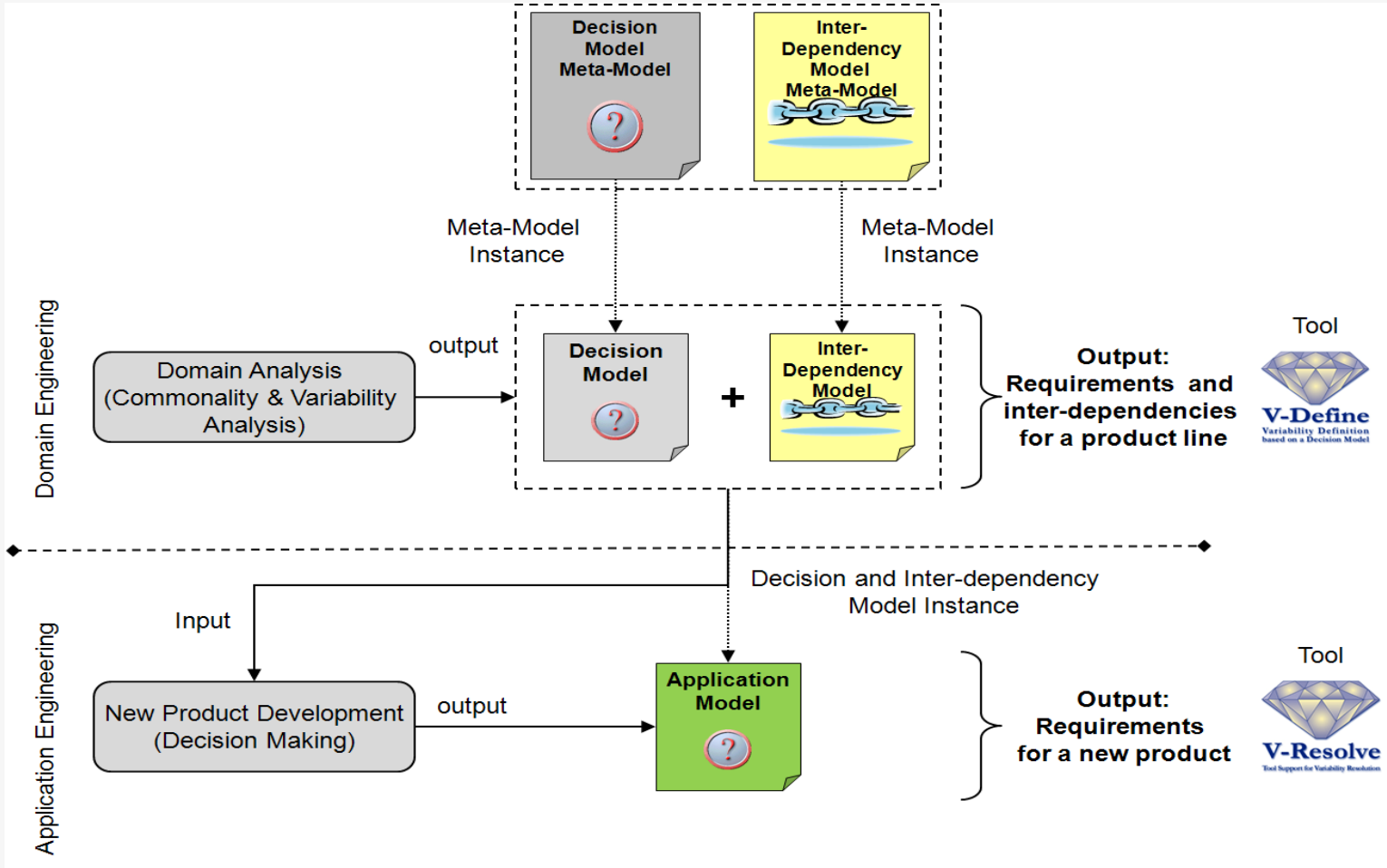
- Helps verify whether the application requirements satisfy the constraints of the product line model.
- Variation Point-based selection always evaluates to TRUE for any resulting requirements selection of a single system.
- Debugging is a matter of isolating the tree in which the wrong selection combinations have been made.
- Easy to automate.
- Invites the engineer to consider reworking the model.

Part V
Selection Decision Interdependencies

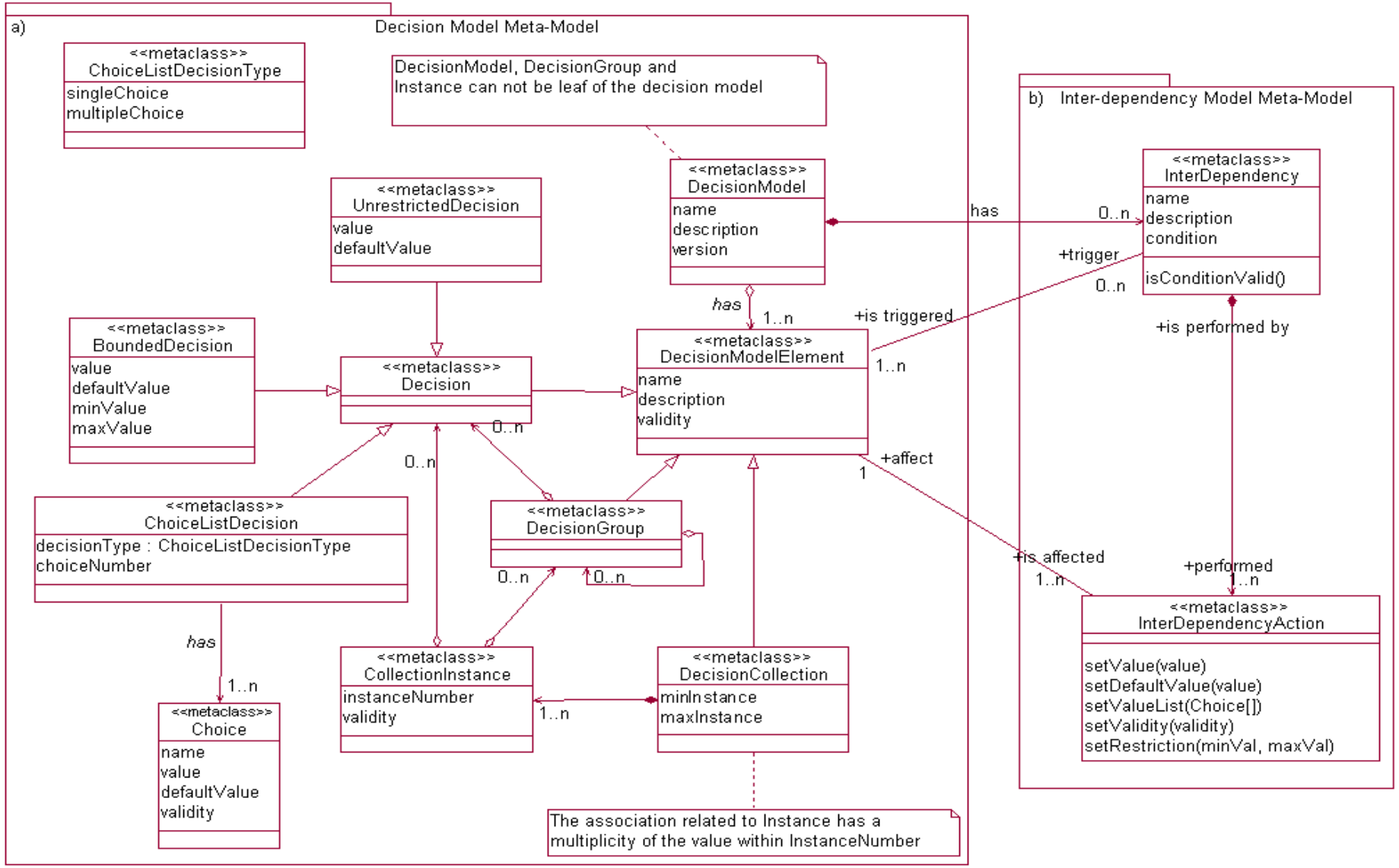
Decision-Making

- large number of variation points and inter-dependencies makes holistic view difficult to see
- inter-dependencies between variation points, between variants, and between variation points and variants
- groups of selection decisions
- groups of inter-dependencies
- different reasons for an inter-dependency
- decision makers require additional information and advice about decisions already taken to inform decisions to be taken
- the transparent and consistent application of selection methods
- different priorities of variation points and variants
- populating of variation points with default variant values before a decision model is used
- making clear the scope and impact of a selection decision i.e. there can be interacting decisions within the same set of artefacts (e.g. features) and interacting decisions between different sets of artefacts (e.g. feature and design and components).

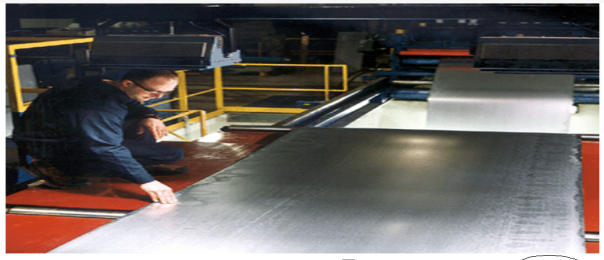
Decision Model Construction & Decision Making Process

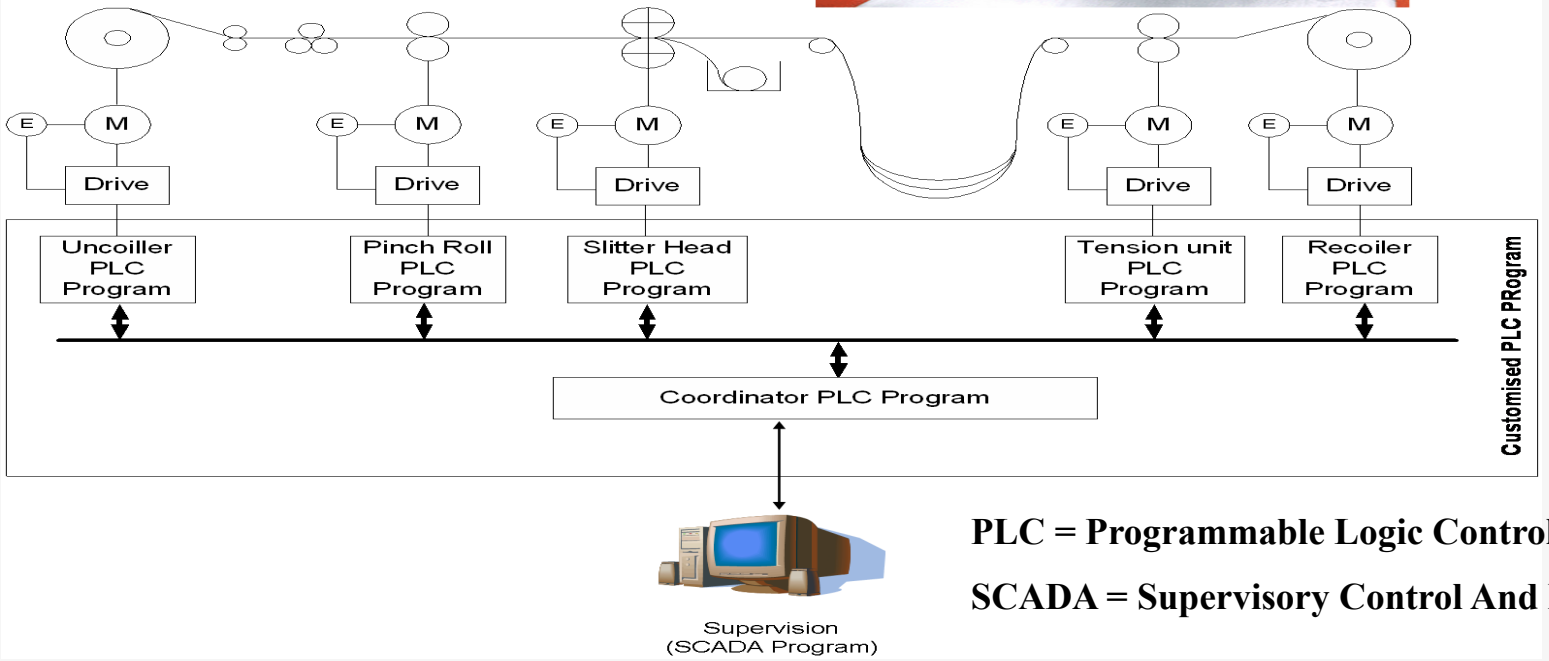


Selection Decision Metamodel



Metal Processing Lines





PLC = Programmable Logic Controller

SCADA = Supervisory Control And Data Acquisition

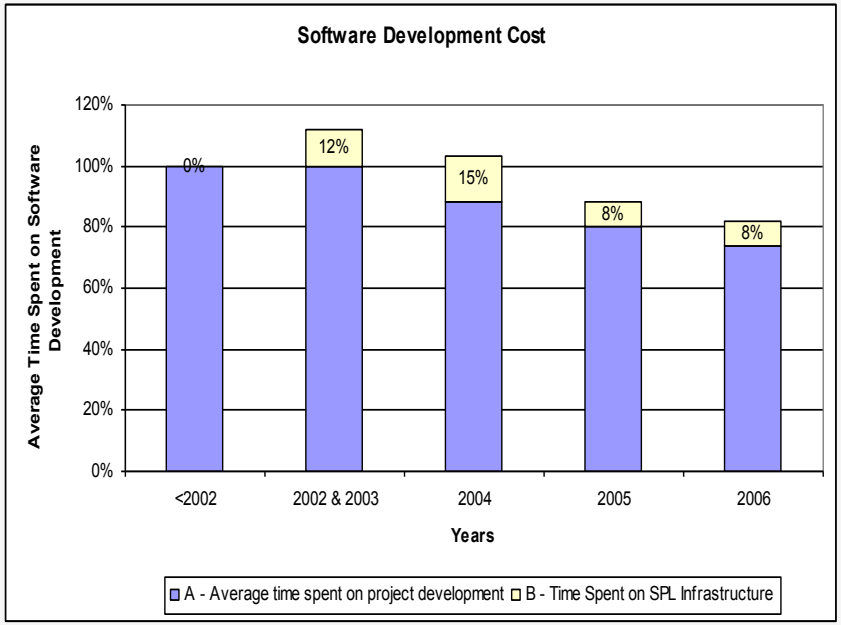
Case Study: Metal Processing Lines Mondragón Sistemas de Información (MSI)

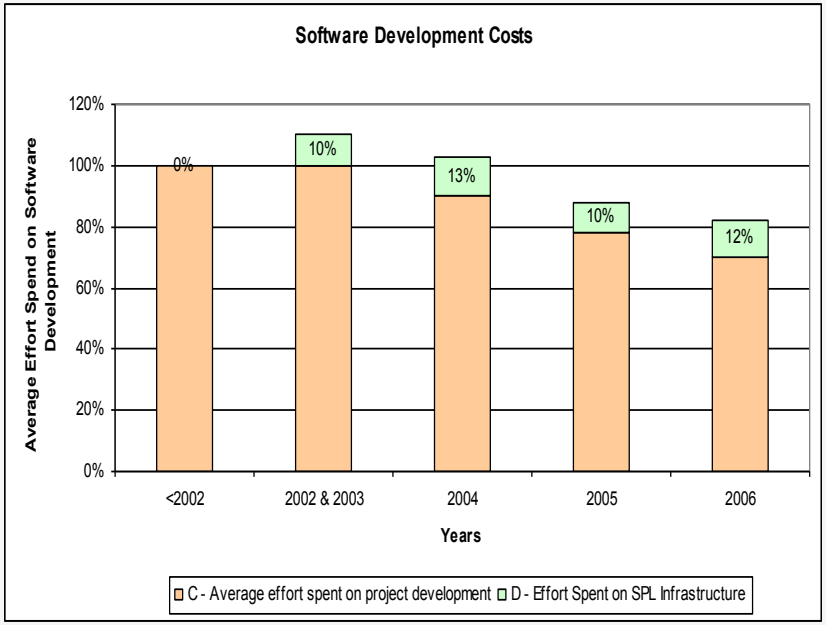
- Product line for the coordinator PLC program
- 9 person-months to develop the decision model with the tools.
- 510 requirements
- 201 common requirements
- 309 variable requirements

Metal Processing Line Decisions

Decision Model Element type	Number of Elements
Decisions	**309**
Unrestricted Decision	8
Bounded Decision	168
Single Choice List Decision	108
Multiple Choice List Decision	25
Group	**59**
Collection	**1**
Choices	**464**
InterDependencies	**198**
Inter-Dependency Actions	**319**

Cost-Benefits





Time spent on projects

Effort spent on projects

Time spent on PL infrastructure

Effort spent on PL infrastructure

Part VI
Tool Support

Tools

- Gears (BigLever)
- Pure:: Variants (Pure Systems)
- X-Feature (ETH-Zürich)
- RequiLine (RWTH Aachen University)
- T-REK (Thalès/Telelogic)
- DecisionKing (Christian Doppler Laboratory for Automated Software Engineering Johannes Kepler University)
- V-Define, V-Resolve (European Software Institute)
- TRAM (ESOC)

Tool Evaluation

Criteria	Pure:Variants	X-Feature	RequiLine	TREK	V-Define/V-Resolve
Attributes management	++	--	++	++	-
Feature meta-modelling	++	+	+	-	++
Feature metamodel security	+	++	+	-	++
Application Modelling	+	+	+	+	++
Traceability	--	-	+	++	+
Impact Analysis	+	+	+	--	+
Reporting	++	--	--	--	+
Access Mode	++	--	++	++	--
Technical Environment	++	-	++	++	--
Usability	+	-	+	+	++
Requirements & Relationships presentation	+	+	+	+	++
Automatic Filters	+	--	++	++	-
Alerts management	++	-	++	++	+

Lessons Learned

- Metamodels enable consistent information
- Rework model if it breaks during selection or if new requirements to be added
- Role of Product Line board
- Tool support essential
- Complex models – visualisation aids
- Managing Changing Variability

Summary

- Describe a product line engineering lifecycle.
- Build a product line model.
- Develop a single system model from a product line model.
- Use a product line engineering support tool.

Selected work of presenters

SELLIER, D., MANNION, M., MANSELL, J., Managing Requirements Inter-dependency for Software Product Line Derivation, *Requirements Engineering Journal* 13(4): 299–313, 2008

MANNION, M., KAINDL, H., Using Parameters and Variation Points for Product Line Requirements', *Systems Engineering*, Volume 11, Issue 1, Spring 2008

SELLIER, D., MANNION, M., Visualizing Product Line Requirement Selection Decisions, *Proc of Workshop on Product Line Visualisation at 11th Software Product Line Conference*, Kyoto, 10-14 September 2007

SELLIER, D., MANNION, M., BENGURIA, G., URCHEGUI, G., Introducing Software Product Line Engineering for Metal Processing Lines in a Small to Medium Enterprise, *Proc of 11th Software Product Line Conference*, Kyoto, 10-14 Sept 2007

MANNION, M., Using First Order Logic for Product Line Model Validation", *Proc of 2nd Software Product Line Conference*, San Diego, 19-22 August 2002, 176-187.

MANNION, M., LEWIS, O., KAINDL, H., MONTRONI, G., WHEADON, J., Representing Requirements of Generic Software in an Application Family Model". 2000. Lecture Notes in Computer Science (LNCS 1844), *Software Reuse: Advances in Software Reusability*, ed W Frakes, 6th International Conference, ICSR-6, Vienna, Austria, 27-29 June 2000, 153-169.

MANNION M., KEEPENCE B., KAINDL H., WHEADON J., Reusing Single Requirements From Product line Requirements, In *Proceedings of 21st IEEE International Conference on Software Engineering (ICSE'99)*, May 1999, 453-462.