# Software Reuse and Reusability based on Requirements: Product Lines, Cases and Feature-Similarity Models

**Hermann Kaindl, Mike Mannion**

1

## Structure

- Introduction
- Part 1: Motivation for Retrieving Similar Products in Software Product Lines
- Part 2: Feature Model Based Development
- Part 3: Case-Based Reasoning
- Part 4: Similarity Matching in Software Product Line Development
- Summary and Conclusion

2

GCU Glasgow Caledonian University TU WIEN

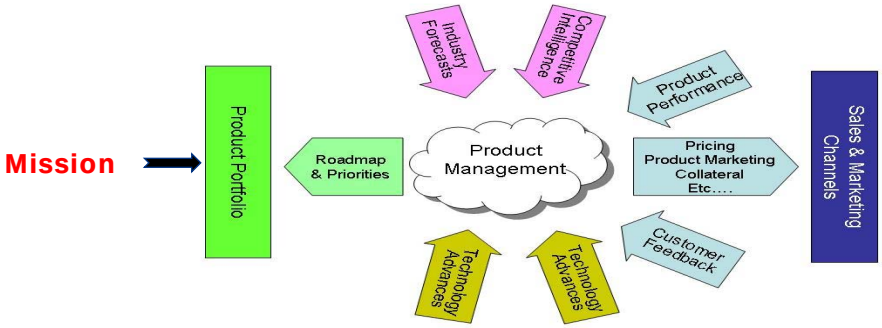
# Part 1

- Motivation for Retrieving Similar Products in Software Product Lines

3

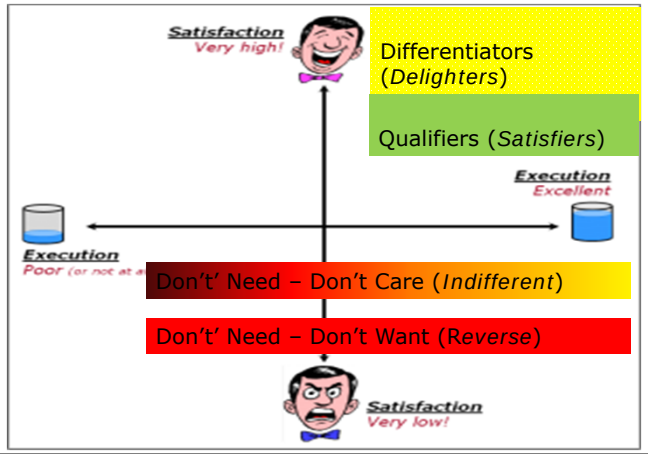
GCU Glasgow Caledonian University TU WIEN

## What Products to Build ?



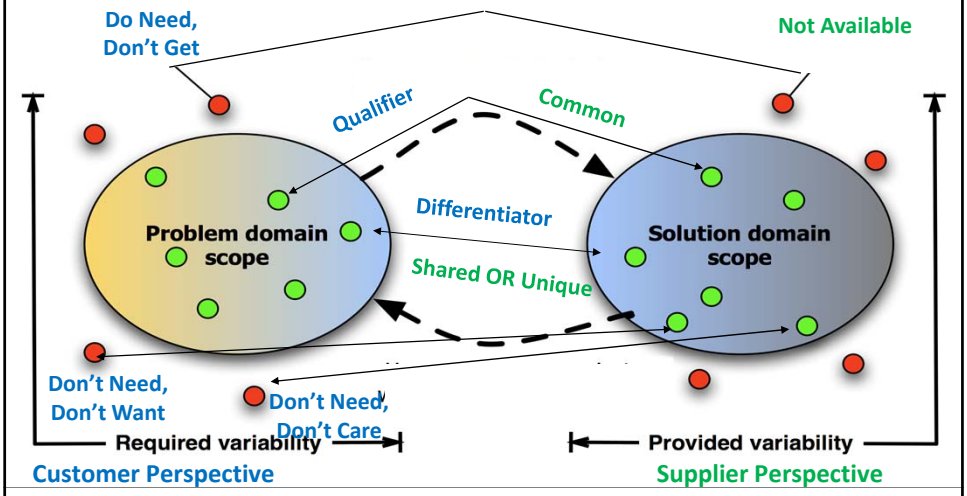

4

## Matching Supply & Demand (Kano Model)




5

## Matching Supply & Demand

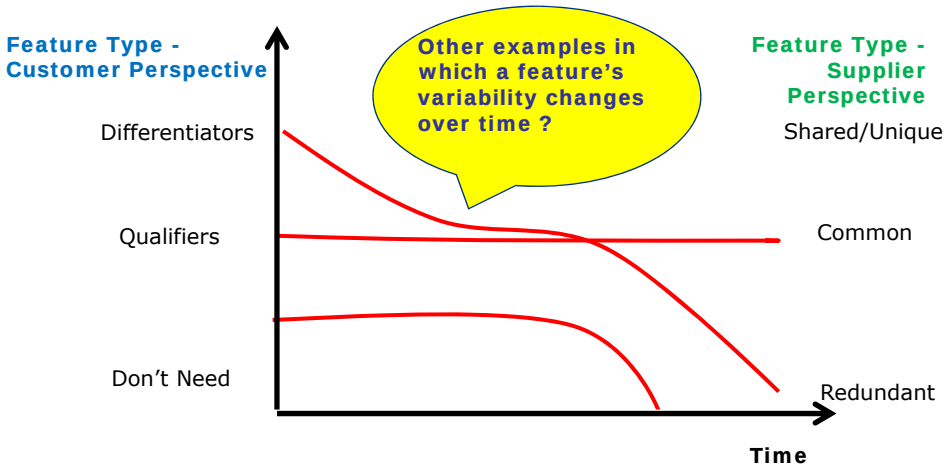



6

6

GCU Glasgow Caledonian University TU WIEN

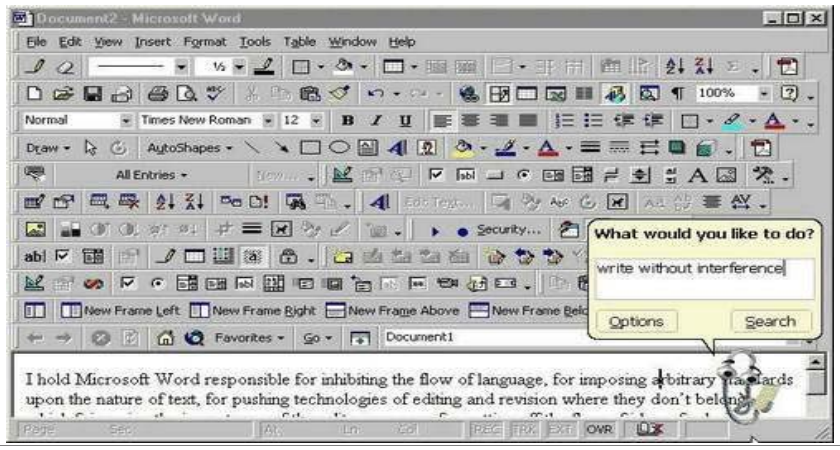
## Matching Supply & Demand - Feature Evolution




7

GCU Glasgow Caledonian University TU WIEN

## Supply Trends – Sales-Driven Product Optimization ("Feature Creep")




8

## Domain Platform Stretching

- Boeing 737 divided into 3 platforms:
- Initial (100,200)
- Classic (300,400,500)
- NextGen (600,700,800,900)

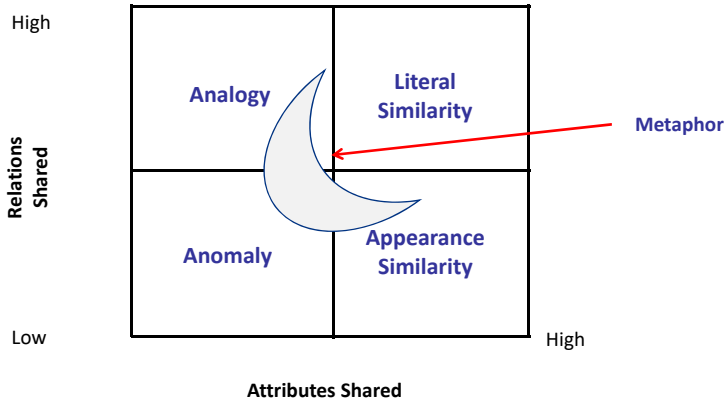
- Boeing 777 also been designed to be stretched


9

## Motivation - Similarity

10

## Similarity




11

## The Need for Similarity Support

- Compare existing products against partial target specification
- Compare similar features in different products
- Compare customer's perspective against supplier's perspectives
- Create a new product from ideas in other products

- What methods do we have ?

12

GCU Glasgow Caledonian University TU WIEN

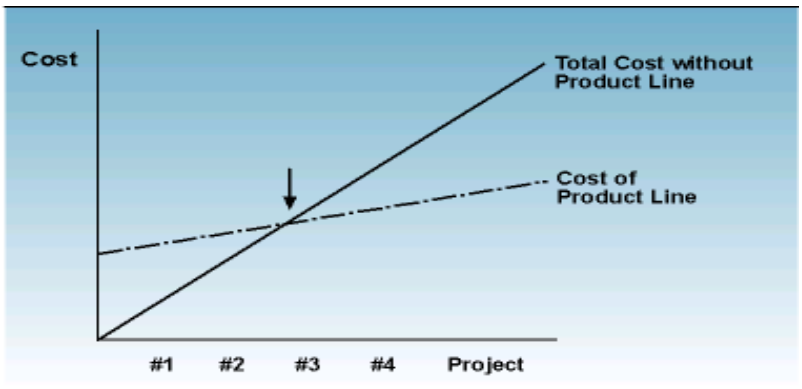
# Part 2

- Feature Model Based Development

20

GCU Glasgow Caledonian University TU WIEN

## Supply Trends - Product Lines

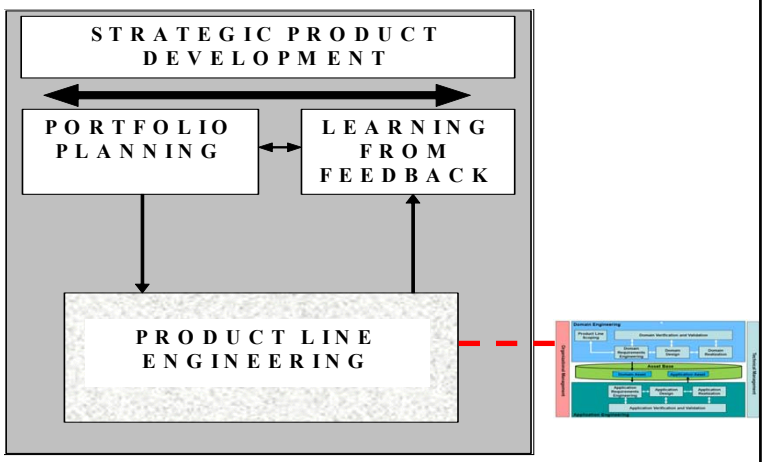



21

GCU Glasgow Caledonian University

TU WIEN

## Supply Trends – Product Line Development Framework

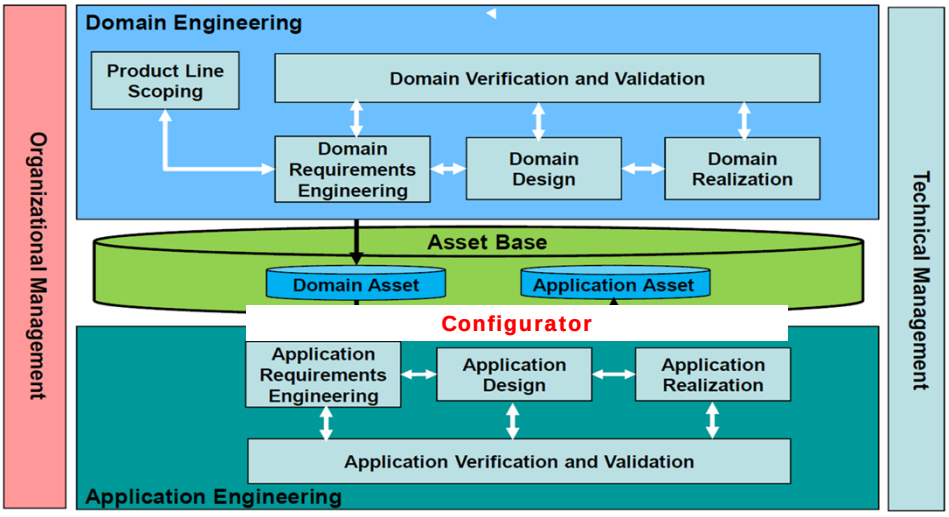



22

GCU Glasgow Caledonian University

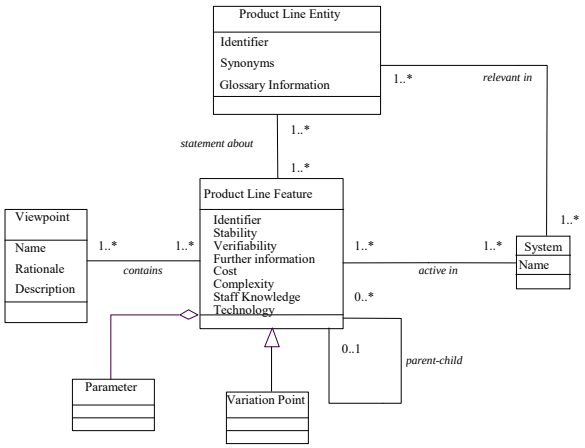
TU WIEN

## Supply Trends - Product Line Engineering




23

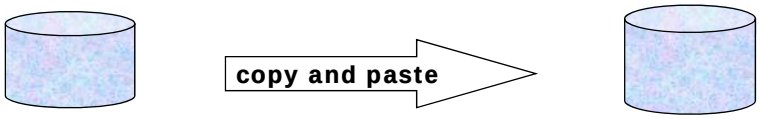
## Metamodel




24

## 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

25

GCU Glasgow Caledonian University TU WIEN

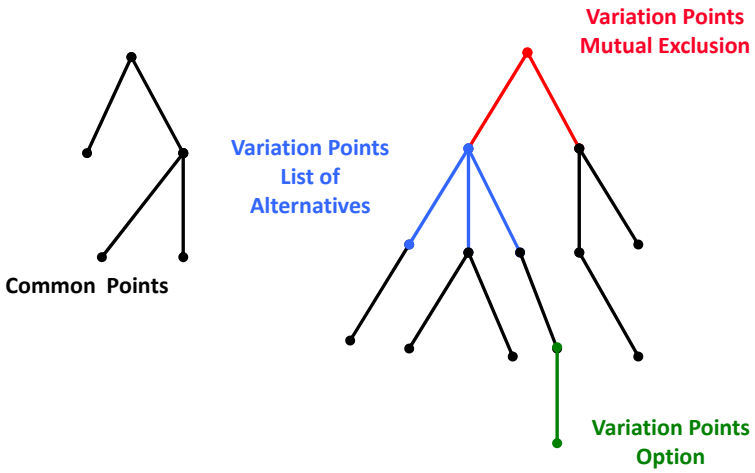
## Problems of Free Selection

- 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!!!

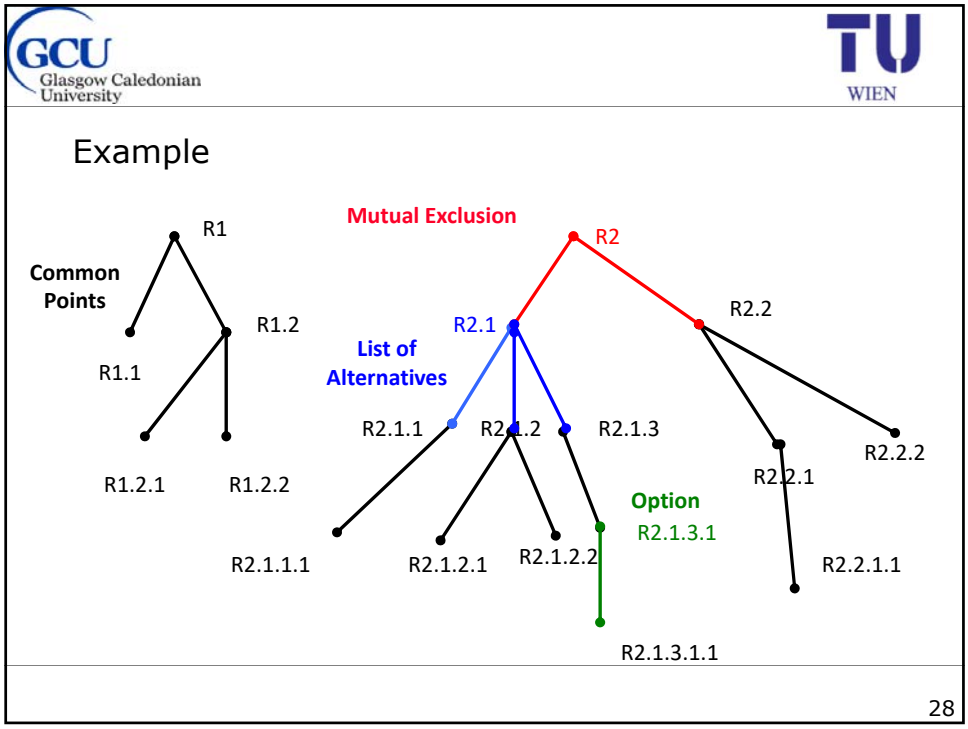
26

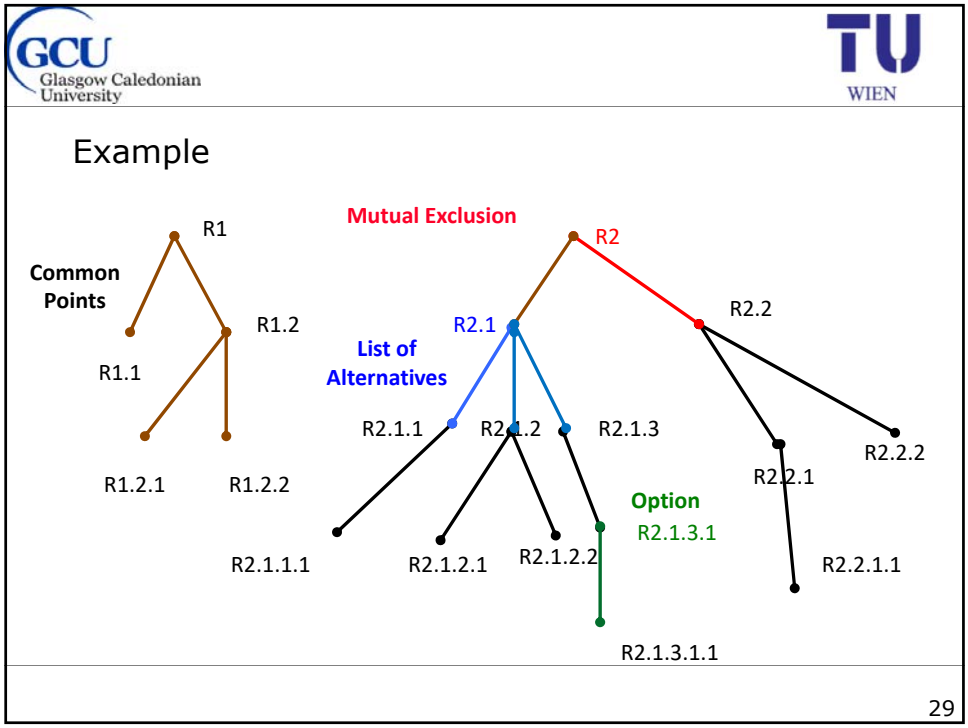
GCU Glasgow Caledonian University TU WIEN

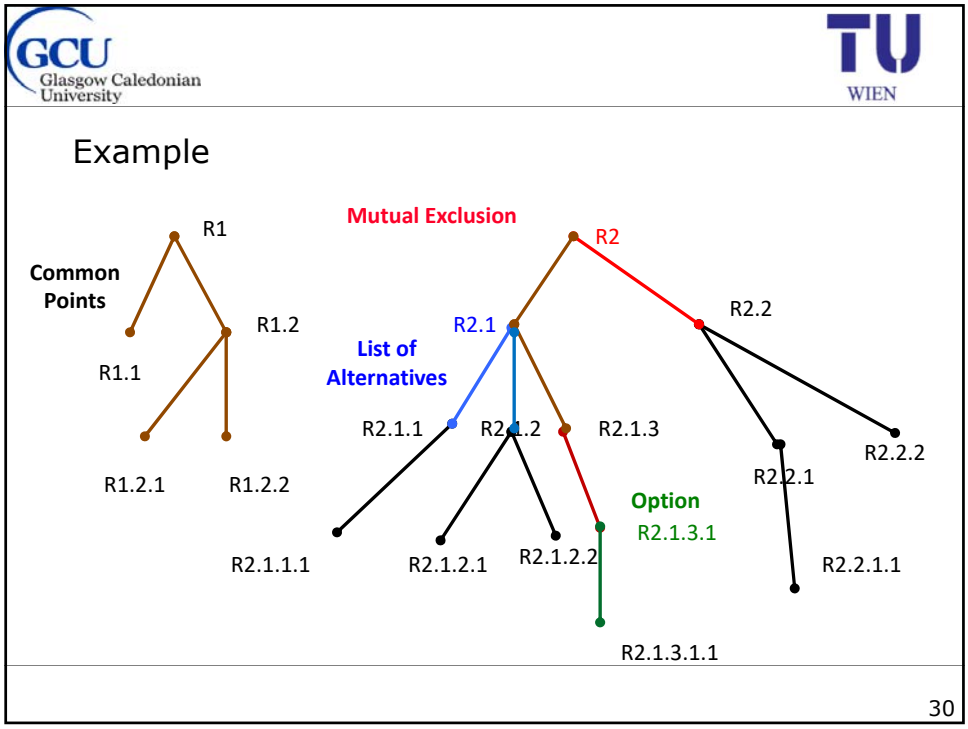
## Constraint-Based Selection

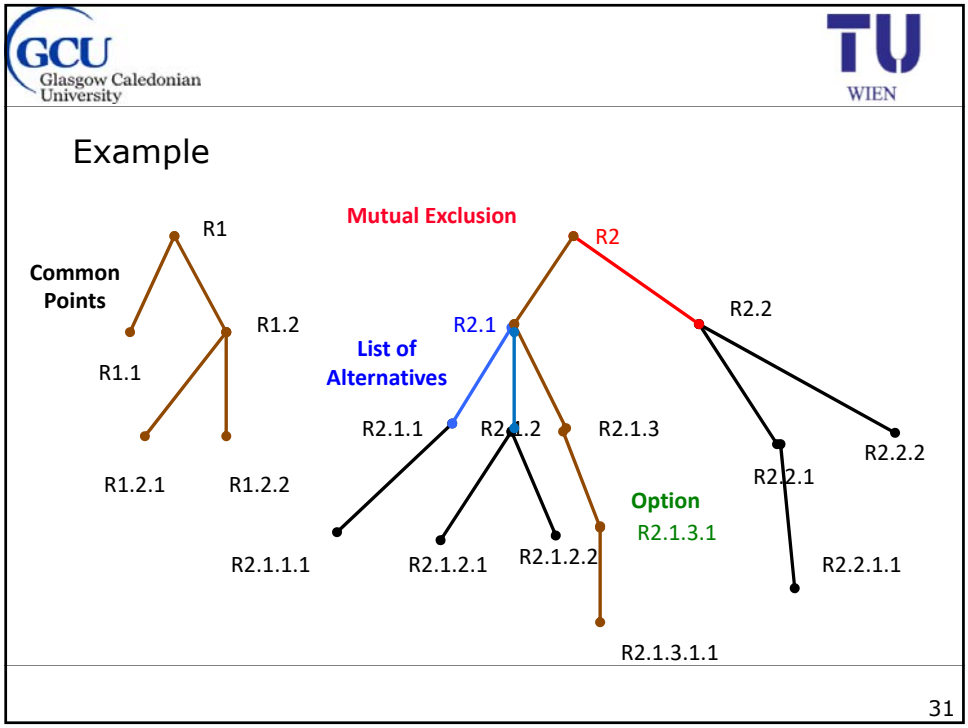



27


GCU Glasgow Caledonian University
TU WIEN
Example
Common Points
R1
R1.1
R1.2
R1.2.1
R1.2.2
Mutual Exclusion
R2
R2.1
R2.2
List of Alternatives
R2.1.1
R2.1.2
R2.1.3
R2.2.1
R2.2.2
Option
R2.1.3.1
R2.1.1.1
R2.1.2.1
R2.1.2.2
R2.2.1.1
R2.1.3.1.1
28


GCU Glasgow Caledonian University
TU WIEN
Example
Common Points
R1
R1.1
R1.2
R1.2.1
R1.2.2
Mutual Exclusion
R2
R2.1
R2.2
List of Alternatives
R2.1.1
R2.1.2
R2.1.3
R2.2.1
R2.2.2
Option
R2.1.3.1
R2.1.1.1
R2.1.2.1
R2.1.2.2
R2.2.1.1
R2.1.3.1.1
29



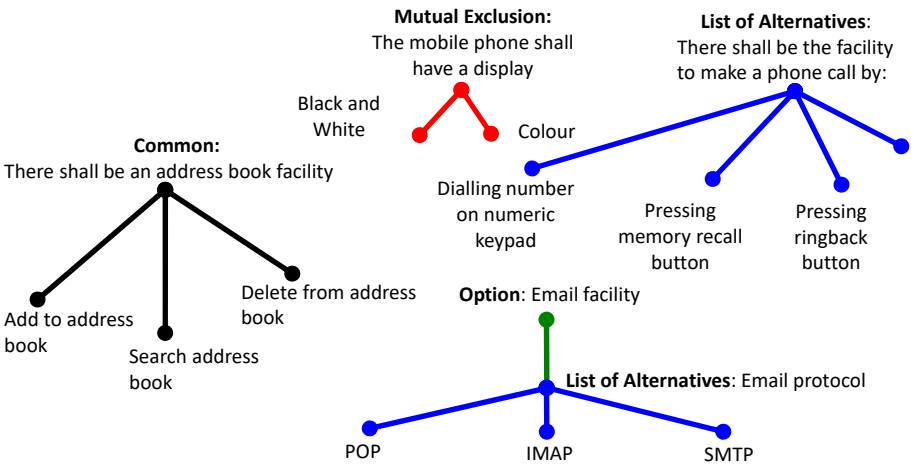



## Product Line Model using Formal Representations

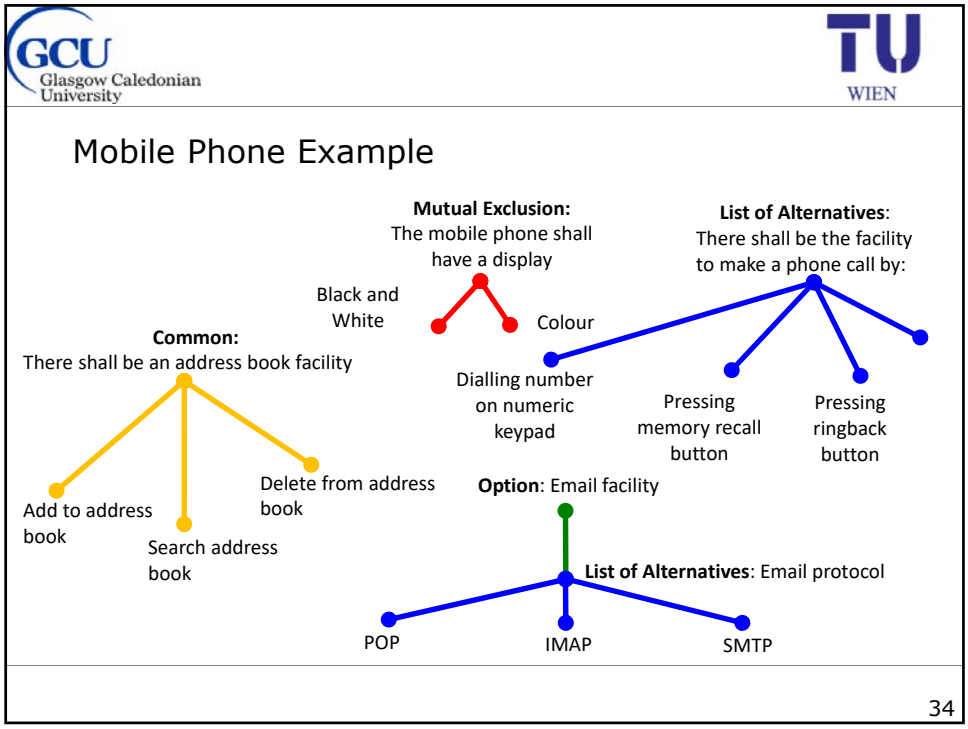
- For a product line model P of product line requirements a logical expression can be defined as
  $E(P) = \{T_1 \wedge T_2 \wedge \ldots \wedge T_n \mid \{T_i = a_{i1}\ Я_{i1}\ a_{i2}\ Я_{i2}\ a_{i3}\ Я_{i3}\ \ldots\ Я_{i(n-1)}\ a_{in};\ a_{ij} = s(r_{ij})$
  - where $r_{ij}$ must be a directly reusable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

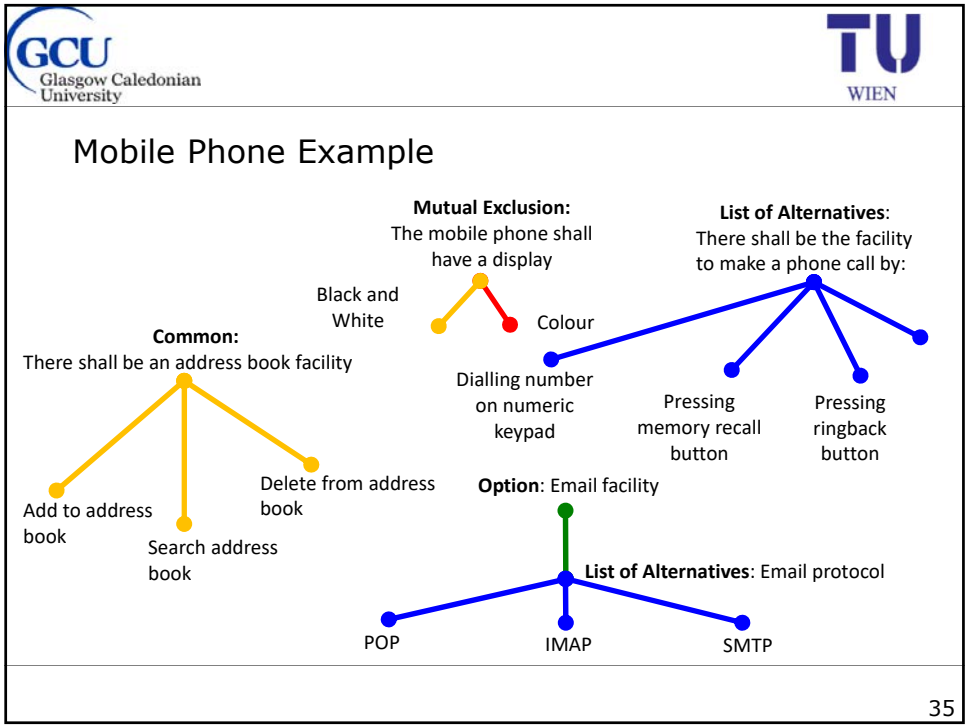
32

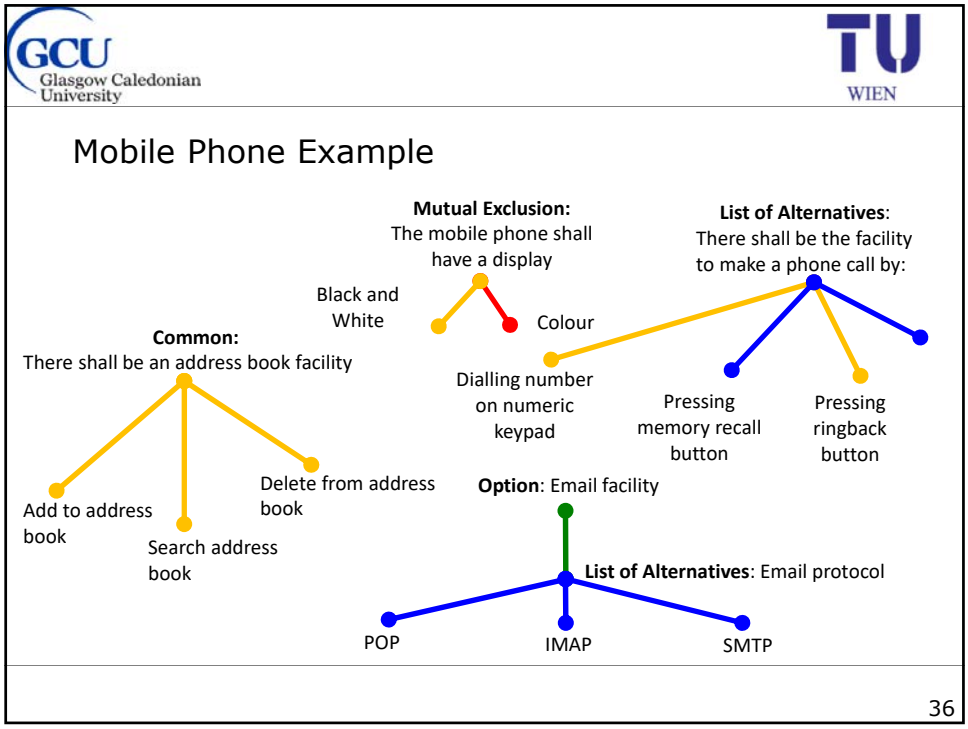
## Mobile Phone Example

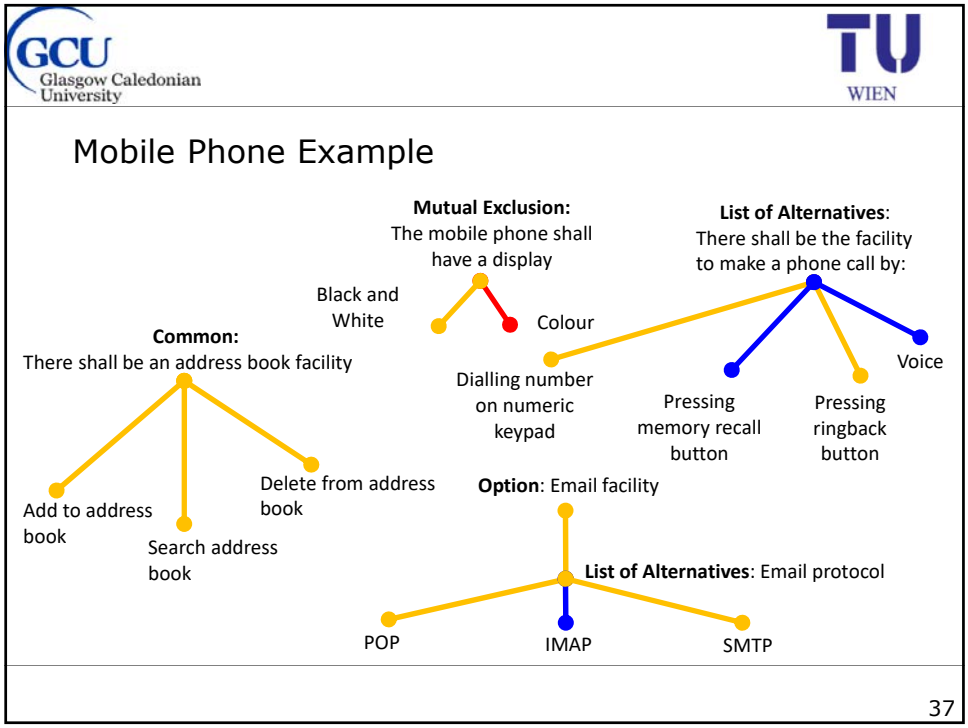



33





## Product Line Model using Formal Representations

- 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 $Я_{ij} \in \{Я_{common}, Я_{mutex}, Я_{list\_alts}, Я_{option}\}\ \}$

38

## Mobile Phone Example – Valid Selection

Propositional Logic Expression Evaluates to **TRUE** or **FALSE** depending on selections made.

New Product = $T_1 \wedge T_2 \wedge T_3 \wedge T_4$

((**AddressBook** $\wedge$ (**Add** $\wedge$ **Search** $\wedge$ **Delete**)) $\wedge$ .....($T_1$)
(**Display** $\wedge$ (**B&W** $\oplus$ **Colour**)) $\wedge$ .....($T_2$)
(**CallTypes** $\wedge$ (**Keypad** $\vee$ **Memory** $\vee$ **Ringback** $\vee$ **Voice**) $\wedge$ .....($T_3$)
(**Email** $\leftrightarrow$ (**EmailProtocol** $\wedge$ (**POP** $\vee$ **IMAP** $\vee$ **SMTP**))).....($T_4$)

i.e.

((**TRUE** $\wedge$ (**TRUE** $\wedge$ **TRUE** $\wedge$ **TRUE**)) $\wedge$ .....($T_1$)
(**TRUE** $\wedge$ (**TRUE** $\oplus$ **FALSE**)) $\wedge$ .....($T_2$)
(**TRUE** $\wedge$ (**TRUE** $\vee$ **FALSE** $\vee$ **TRUE** $\vee$ **FALSE**)) $\wedge$ .....($T_3$)
(**TRUE** $\leftrightarrow$ (**TRUE** $\wedge$ (**TRUE** $\vee$ **FALSE** $\vee$ **TRUE**))).....($T_4$)

which evaluates to **TRUE**

39

## Mobile Phone Example – Invalid Selection

New Product = $T_1 \wedge T_2 \wedge T_3 \wedge T_4$

((AddressBook ∧ (Add ∧ Search ∧ **Delete**)) ∧ .....($T_1$)
(Display ∧ (B&W ⊕ **Colour**)) ∧ .....($T_2$)
(CallTypes ∧ (Keypad ∨ **Memory** ∨ Ringback ∨ **Voice**) ∧ .....($T_3$)
(Email ↔ (EmailProtocol ∧ (POP ∨ **IMAP** ∨ SMTP))).....($T_4$)

Not Selected

i.e.

((TRUE ∧ (TRUE ∧ TRUE ∧ FALSE)) ∧ .....($T_1$)
(TRUE ∧ (TRUE ⊕ FALSE)) ∧ .....($T_2$)
(TRUE ∧ (TRUE ∨ FALSE ∨ TRUE ∨ FALSE)) ∧ .....($T_3$)
(TRUE ↔ (TRUE ∧ (TRUE ∨ FALSE ∨ TRUE))).....($T_4$)

which evaluates to **FALSE because $T_1$ is FALSE**

40

## 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.

41

## Automated Feature Model Analysis

- Does the configured product satisfy the feature model constraints ?
- How many product configurations (if any) satisfy these constraints ?
- How many products satisfy a given set of features ?
- Any anomalies in the feature model e.g. contradictions, redundancy ?
- To what degree (expressed as a numeric value) has a feature model variable or common features.

**Feature models (regardless of notation) not suited to compare similarity.**

Benavides, D., Segura, S., Ruiz-Cortes, A., 2010 Automated Analysis of Feature Models 20 years Later: A Literature Review. *Information Systems*, 35(6):615–636.

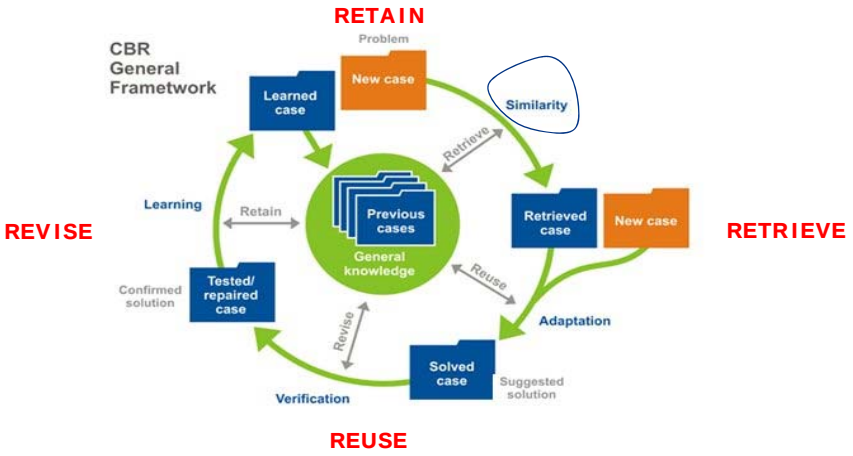
43

## Part 3

- Case-Based Reasoning

44

## Case-Based Reasoning




Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, A Aamodt, E Plaza, AI Communications,Vol. 7 Nr. 1, March 1994, pp 39-59

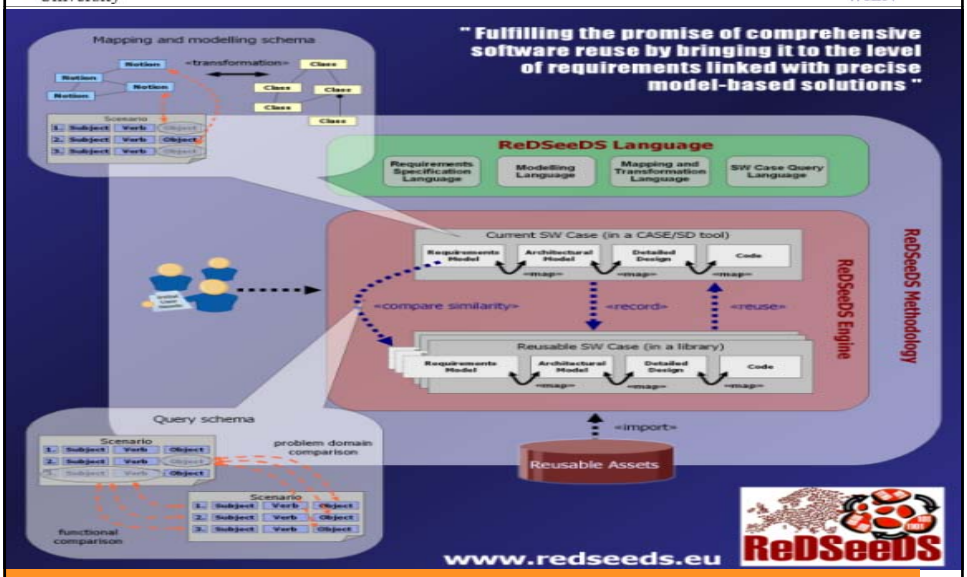
45

## CBR Characteristics

| Task | Issues |
|---|---|
| **RETAIN** | Concrete cases or Generalised cases;<br>Central knowledge units or distributed units<br>Indexed or flat or hierarchical<br>General and/or domain specific ontologies<br>Rich information beyond feature vectors |
| **RETRIEVE** | Use 1 or more similarity metrics e.g. K nearest Neighbour<br>Guided or not by deep model of general knowledge<br>Sequentially or in parallel |
| **REUSE** | Facilities to use directly or modified retrieved case |
| **REVISE** | Editing facilities to create a new "solved" case<br>(i) the end user does it OR<br>(ii) automated procedure: risk is that system makes poor judgement, yet it is added to the case-base which becomes progressively degraded. |

46

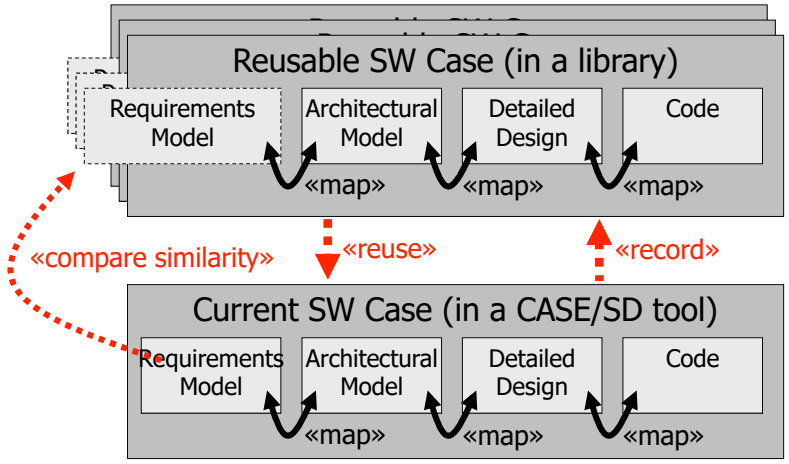
## CBR Example: ReDSeeDS




Version 0.7.0 released June 2016: Infovide S.A., Warsaw University of Technology (PL), HITeC e.V., University of Koblenz-Landau, Fraunhofer - Gesellschaft (Institute for Experimental Soft-ware Engineering), PRO DV Software AG (G), Institute of Mathematics and Computer Science, University of Latvia (LV), Vienna University of Technology (AU), Algoritmu sistemos,UAB (LT),

47

## RETAIN




Similarity-Driven Reuse, Bildhauer, D., Horn, T., Ebert, J., CSVM'09 17 May 2009

48

GCU Glasgow Caledonian University

TU WIEN

## RETAIN


**Figure 1. Sample scenario written in RSL**

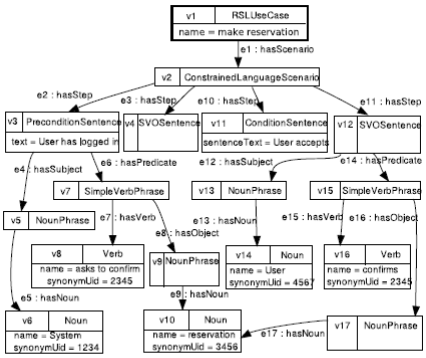


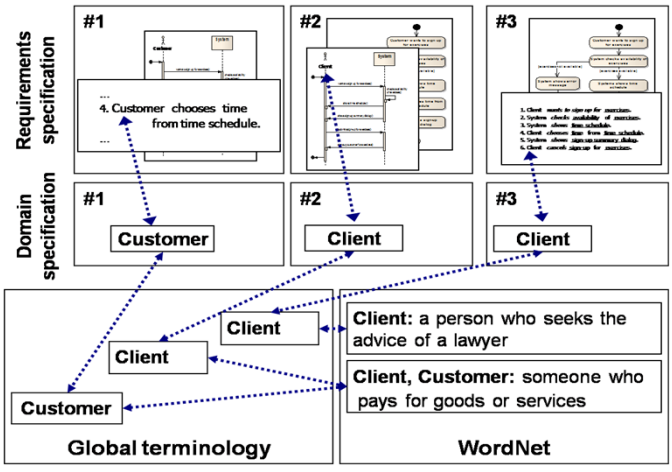

**Figure 2. Graph for the sample scenario**

49

GCU Glasgow Caledonian University

TU WIEN

## RETAIN

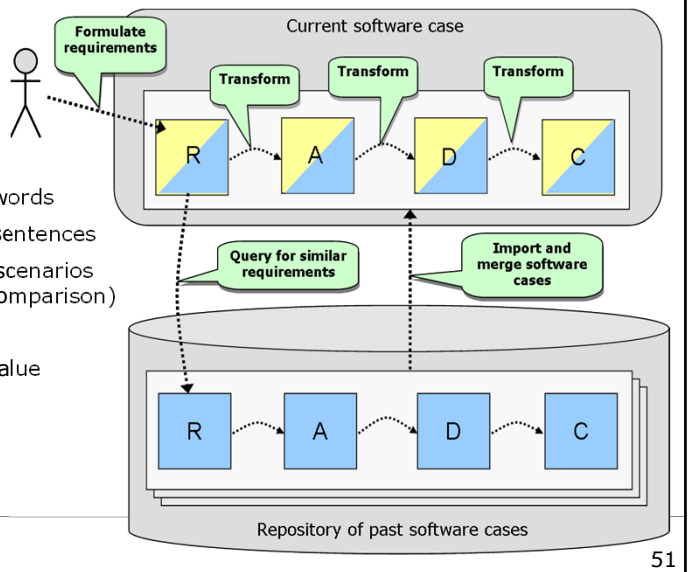



https://wordnet.princeton.edu/

50

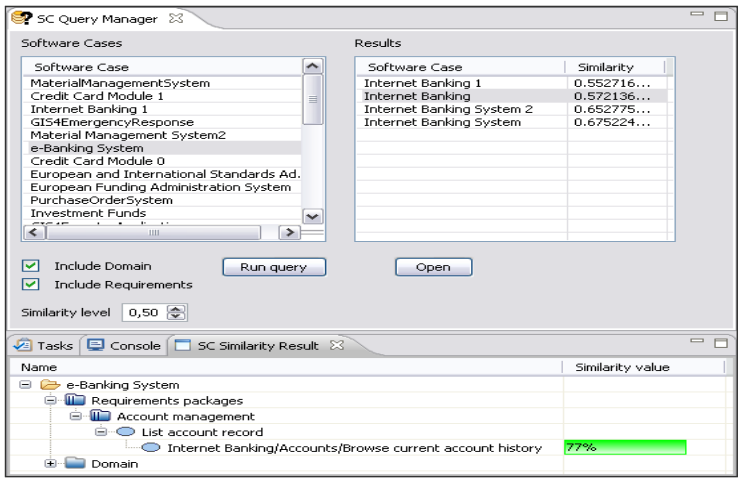
## RETRIEVE

- similarity between words
- similarity between sentences
- Similarity between scenarios (structured graph comparison)
- A single similarity value calculated




51

## RETRIEVE




52

## REUSE & REVISE

- Select one of the better rated software cases.
- Import it to currently developed software case.
- May include design and implementation artefacts, but also requirements and domain descriptions.
- Merge reused case with currently developed one.

53

## CBR Summary Characteristics

| Task | Issues |
|---|---|
| **RETAIN** | Concrete cases or Generalised cases;<br>Central knowledge units or distributed units<br>Indexed or flat or hierarchical<br>General and/or domain specific ontologies<br>Rich information beyond feature vectors |
| **RETRIEVE** | Use 1 or more similarity metrics e.g. K nearest Neighbour<br>Guided or not by deep model of general knowledge<br>Sequentially or in parallel |
| **REUSE** | Facilities to use directly or modified retrieved case |
| **REVISE** | Editing facilities to create a new "solved" case<br>(i) the end user does it OR<br>(ii) automated procedure: risk is that system makes poor judgement, yet it is added to the case-base which becomes progressively degraded. |

54

GCU Glasgow Caledonian University TU WIEN

# Part 4

- Similarity Matching in Software Product Line Development

55

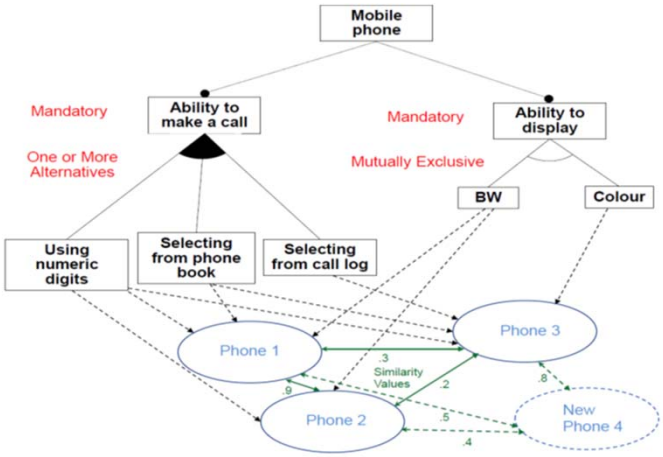
GCU Glasgow Caledonian University TU WIEN

## Software Product Line Engineering v Case-Based Reasoning

| | Feature Modelling | Case-based Reasoning |
|---|---|---|
| **Model Structure Construction** | Complex | Straightforward in principle; But structural considerations can enhance search |
| **Model Content** | Detailed, Precise | Good-enough; supported by vocabularies Watch for poor old cases |
| **Product Derivation** | Constrained facilitated automated product derivation | Similar cases automatically retrieved. Monitor search performance & accuracy |
| **Costs of making *reusable*** | Substantial | Small but Adaptation Knowledge needed |
| **Benefits for *reuse*** | Facilitates automated product derivation | Facilitates finding similar cases for reuse |

56

## Feature-Similarity Model




H. Kaindl and M. Mannion, A Feature-Similarity Model for Product Line Engineering, in *Proc. of the 14th International Conference on Software Reuse (ICSR'15)*, LNCS 8919, 2015, 34–41.

57

## Construction Process

- Construct a feature model.
- Allocate features to products.
- Calculate similarity values between features.
- Calculate similarity values between specific feature combinations in different product specifications.
- Calculate similarity values between entire product specifications.

58

## Exercise - Similarity in RE

| Task | Focus | Value of Similarity Matching ? |
|---|---|---|
| Product Line Scoping | Identify features that:<br>- distinguish the product line<br>- are important for target market<br>- Mark product line boundary. | ? |
| Domain Engineering | Specify the common and variable domain requirements of agreed product line | ? |
| Application Engineering | Generate or specify application specific requirements | ? |

59

## Similarity in RE

| Task | Focus | Purpose of Similarity Matching |
|---|---|---|
| Product Line Scoping | Identify features that:<br>- distinguish the product line<br>- are important for target market<br>- Mark product line boundary. | Should product be in PL or not ?<br>If so, where to position it ? |
| Domain Engineering | Specify the common and variable domain requirements | Reduce model updates if similar requirements exist<br>Evaluate platforms before merger or separation |
| Application Engineering | Specify application specific requirements | Compare proposed product with existing products |

60

- Summary & Conclusion

61

## Motivation for Similarity

- Customers want to personalise
- Suppliers
  - want to be distinctive
  - want to be more efficient
- Match supply and demand

62

## Research Challenges

- How can similarity matching be factored into existing process models for Product Line Scoping, Domain Engineering, and Application Engineering ?
- When to compute a similarity between two products: at product definition or on demand
- What are the thresholds for "similar" and for "different"?
- What are the thresholds for analogy anomaly ?
- Similarity Metrics
  - What combinations are worth computing e.g. pearson coefficient, cosine similarity, euclidean distance, k-nearest neighbour algorithm.
  - Use caution and prudence - best when used with data from other reference points.
  - Be clear on what you are using the metric for, get general agreement in the organization on which metrics to use, and focus on only a few metrics – less is more.

63

## Selected Work of Presenters


- R. Hoch, H. Kaindl, R. Popp, D. Ertl, and H. Horacek, Semantic Service Specification for V&V of Service Composition and Business Processes, in *Proceedings of the 48nd Annual Hawaii International Conference on System Sciences (HICSS-48)*. Piscataway, NJ, USA: IEEE Computer Society Press, 2015.
- H. Kaindl and M. Mannion, A Feature-Similarity Model for Product Line Engineering, in *Proceedings of the 14th International Conference on Software Reuse (ICSR'15)*, LNCS 8919, 2015, 34–41.
- H. Kaindl, M. Smialek and W. Nowakowski, Case-based Reuse with Partial Requirements Specifications, in *Proceedings of the 18th IEEE International Requirements Engineering Conference (RE 2010)*, 2010, 399–400.
- H. Kaindl and D. Svetinovic, On confusion between requirements and their representations, *Requirements Engineering*, vol. 15, 2010, 307–311.
- M. Mannion and H. Kaindl, Using Parameters and Discriminants for Product Line Requirements. *Systems Engineering*, vol. 11, no. 1, 2008, 61–80.


64