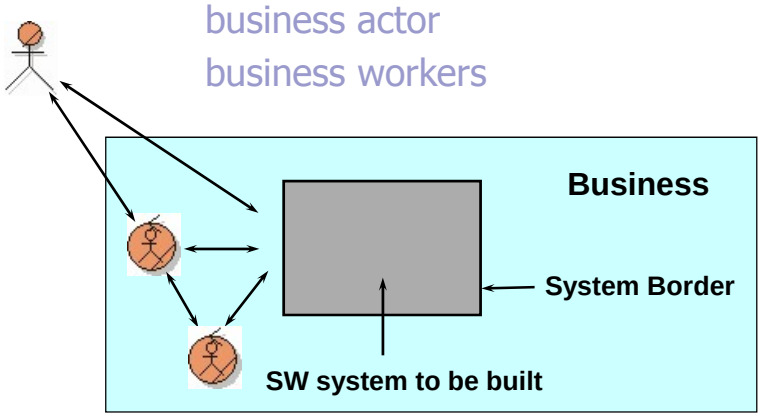
# Software Reuse and Reusability based on Requirements, Product Lines, and Business Knowledge

Institut für Computertechnik
ICT
Institute of Computer Technology

Hermann Kaindl
Vienna University of Technology, ICT
Austria

## System overview

business actor
business workers




Composite system

## Outline

- Introduction and background
- Requirements R&R in product lines
- Software R&R involving case-based reasoning
- R&R for business knowledge and software
- Contrasting these approaches
- Summary and conclusion



## Introduction

- Software reuse has many facets.
- Three approaches where the presenter has been involved:
  - Explicit representation of commonality and variability in requirements
  - Similarity metrics for requirements and design artefacts
  - Reuse driven from business process level

## Software reuse and reusability (R&R)

- "*Software reuse* is the use of existing software or software knowledge to construct new software."
- "*Reusable assets* can be either reusable software or software knowledge."
- "*Reusability* is a property of a software asset that indicates its probability of reuse."



## What are requirements?

- User wishes / needs 
- *IEEE Standard:*
  "A condition or capacity needed by a user to solve a problem or achieve an objective."
- "The <*system*> shall be able to ..."
  - system to be built
  - composite system
- *Example:* "The ATM shall accept a cash card."
- Requirements modeling

## What are requirements? – In practice

User requirements documents

Software/system requirements documents

Mostly descriptions in natural language

Representation often unstructured

Ad hoc process

Communication problem

Requirements and use cases?

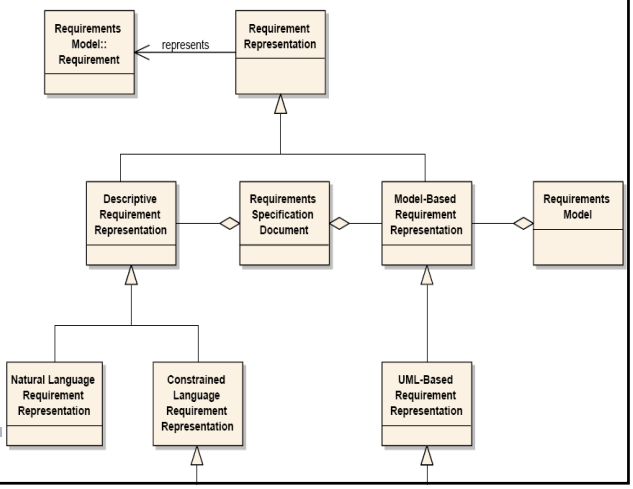


## Scenarios – Stories and narratives

- For representation of
  - cultural heritage
  - explanations of events
  - everyday knowledge
- Human understanding in terms of specific situations
- Human verbal interactions by exchanging stories

## Requirements vs. requirements representation

- Reuse of requirements representation only
- Distinction between
  - descriptive and
  - model-based
- Descriptive: need described
- Model-based: abstraction of what the system should look like






## Business knowledge

- Business Objects
  e.g., an Invoice or an Authorization
- Business Processes
  e.g., first Create Invoice and then Send Invoice.
- Business Rules
  e.g., Invoice must be authorized before being sent.
- Business Ontologies

## Ontologies

- Tom Gruber
- Actually, the old Greeks
- Domain models
- Conceptualizations of a domain
- Often using taxonomies and object-based ideas
- Ontology languages based on knowledge-representation theories
- E.g., OWL based on description logic



## Outline

- Introduction and background
- Requirements R&R in product lines
- Software R&R involving case-based reasoning
- R&R for business knowledge and software
- Contrasting these approaches
- Summary and conclusion

## Product lines

- 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
  Process that delivers software artefacts that can be reused to support the development of new products in the domain
- Commonality and variability



## Requirements reuse and reusability

- Why requirements reuse?
  - Well-understood requirements are basis for reusable architecture and components.
  - Requirements are a reusable asset.
- Commonality and variability also in requirements of a product line
- Making requirements reusable
- Reusing requirements

## Modeling requirements for a product line

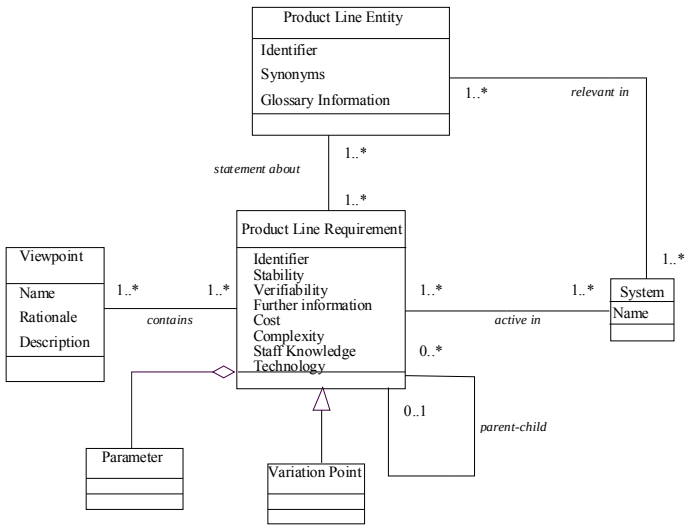
- Which requirements are common in more than one system?
- Figure out about Variation Points:
  - Is there a qualitative variation in requirements among systems?
  - Is there a quantitative variation in requirements among systems?
- Organize requirements in hierarchies (trees).



## Product Line Model

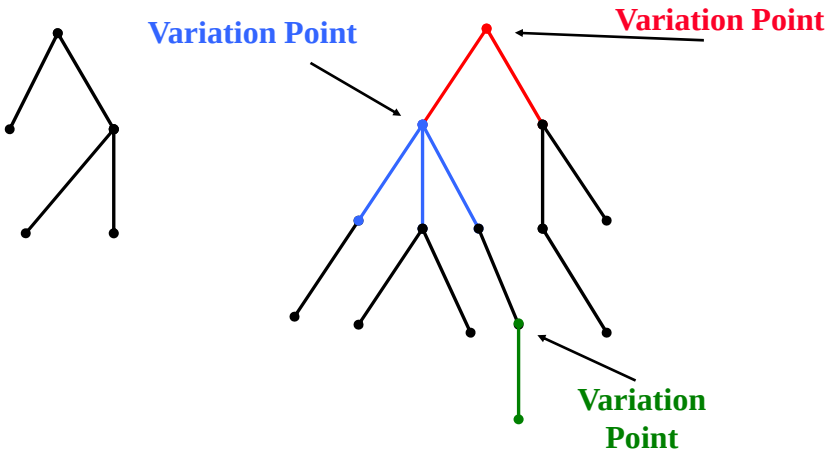
- Natural language representation of "atomic" requirements organized in tree structure
- Classification of reusable requirements:
  - common
  - variable (Variation Point)
- Mobile phone example:
  - common: There shall be the capability to make a telephone call.
  - variable: The mobile phone shall have provide TV.

## 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 a facility to add a telephone number.
- **REQ 1.2**
  There shall be a facility to search for a telephone number.
- **REQ 1.3**
  There shall be a 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"

## 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 parameterized Variation Points.



## Variation Point Types

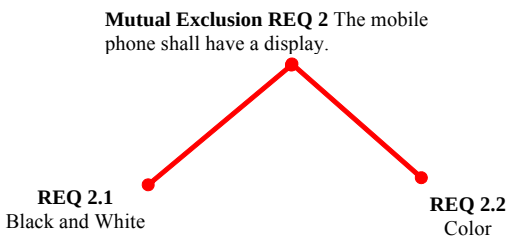
- *Mutual exclusion*: a set of mutually exclusive features from which only one can be used in any system in the domain
- *List of Alternatives*: a set of features which are optional but not mutually exclusive and at least one will be chose
- *Option*: A single optional feature
- Combination of above

## 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 color display.



## Graphical Representation of a Mutual Exclusion

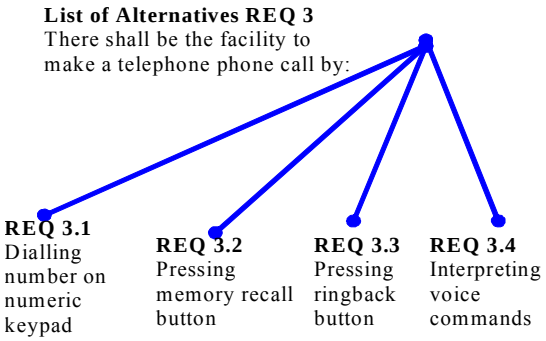


## List of Alternatives Example

- **REQ 3**: There shall be the facility to make a telephone call.
- **REQ 3.1**: The mobile phone shall allow making a telephone call by pressing the numeric digits that form a telephone number.
- **REQ 3.2:** The mobile phone shall allow making a telephone call by pressing a memory recall button.
- **REQ 3.3:** The mobile phone shall allow making a telephone call by using a ring-back facility.
- **REQ 3.4:** The mobile phone shall allow making a telephone call by using speech recognition technology.



## Graphical Representation of a List of Alternatives



## Option Example

- **REQ 4**
  The mobile phone shall have an email facility.



## Graphical Representation of an Option

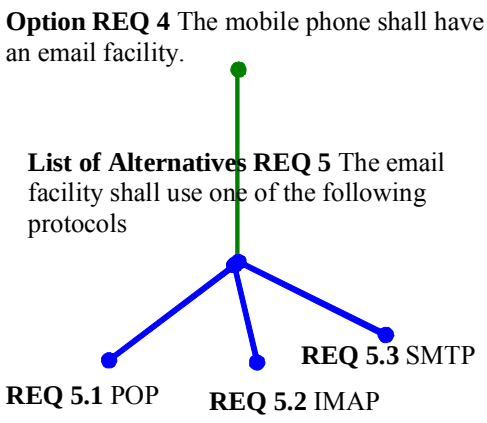
**Option REQ 4** The mobile phone shall have an email facility.

## Variation Point Combination Example

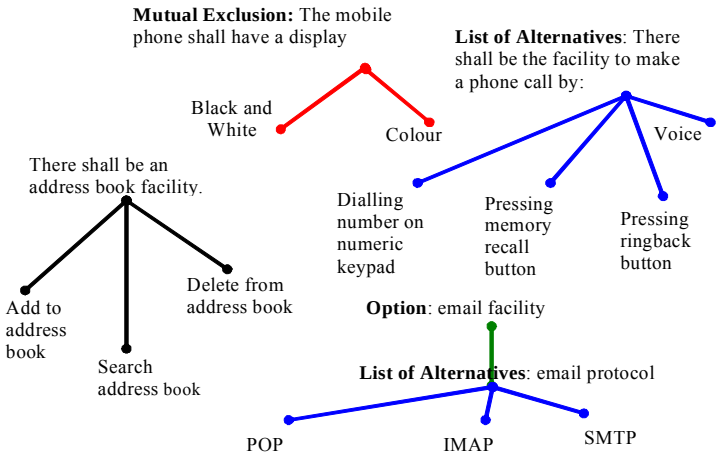
- **REQ 4 (Option)**
  The mobile phone shall have an email facility.
- **REQ 5 (Parent of List of Alternatives)**
  The email facility shall use one of the following protocols.
- **REQ 5.1**: There shall be the facility to use the Post Office Protocol.
- **REQ 5.2:** There shall be the facility to use 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






## Parameterized 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 parameterized Variation Point is a Variation Point that also happens to contain parameters.
- If parameters removed, requirement remains Variation Point.

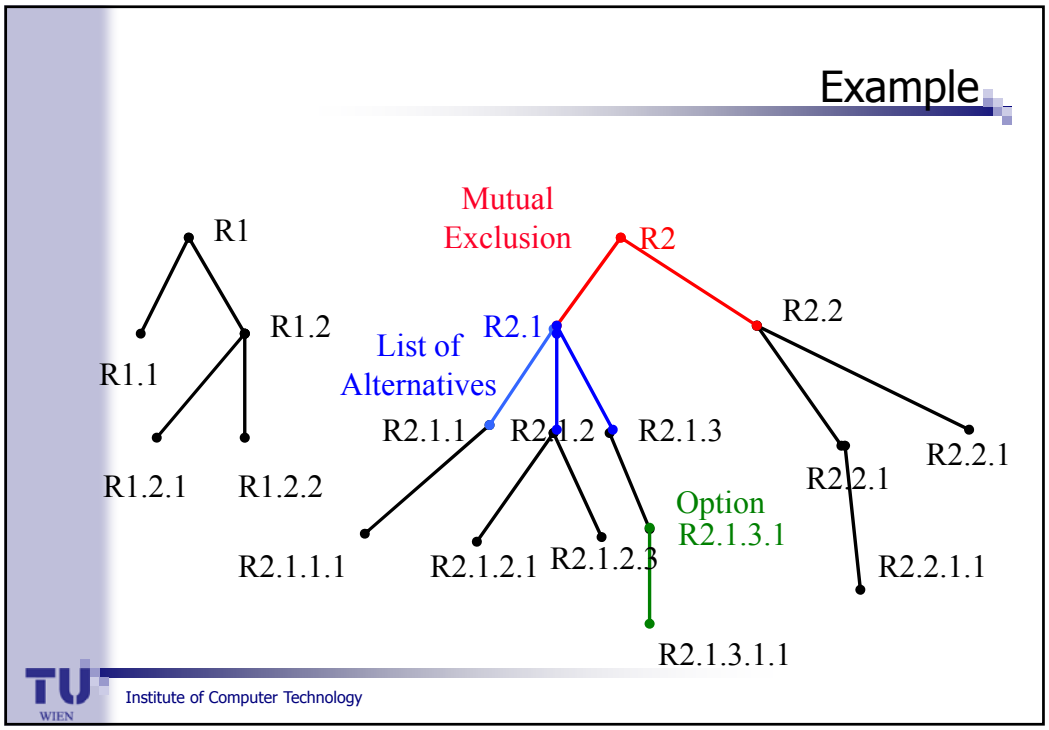
## Selecting single-system requirements for reuse

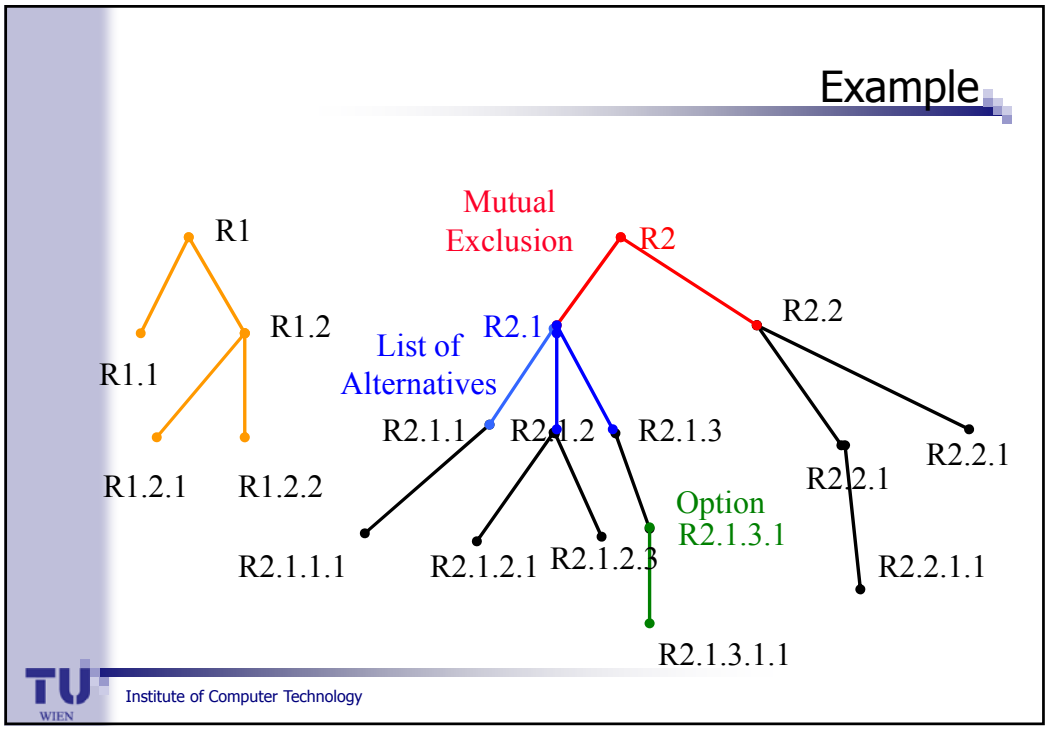
- Reuse requirements from product line for new single system
- Different approaches:
  - Variation Point-based selection
  - Free selection
- Different properties

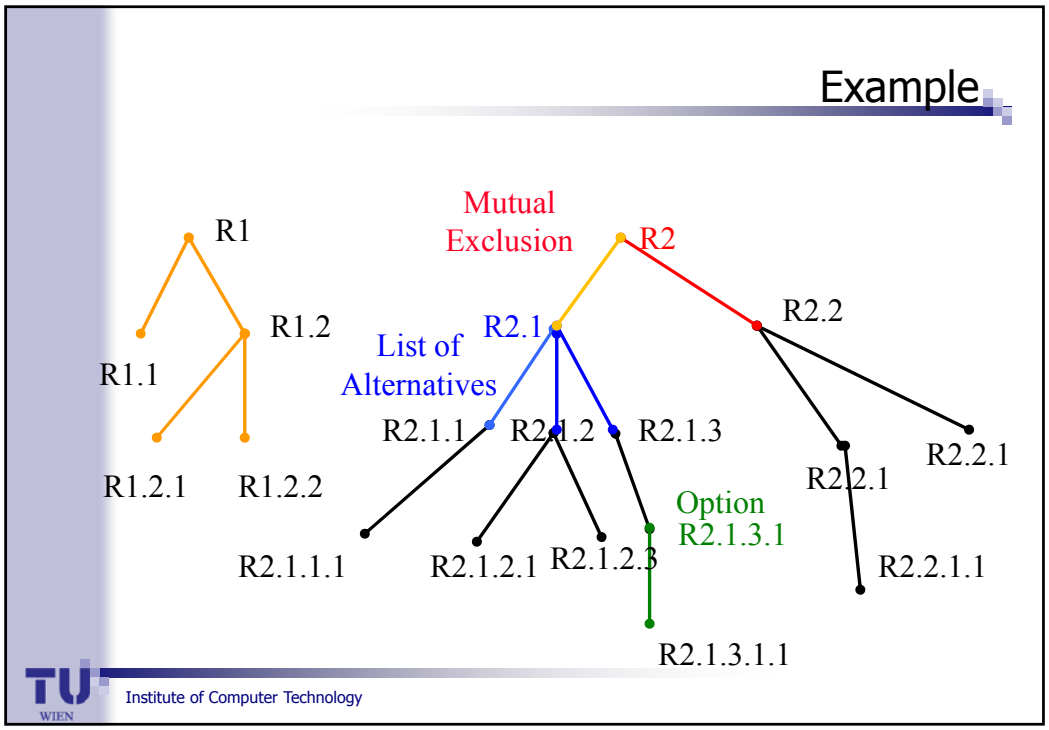


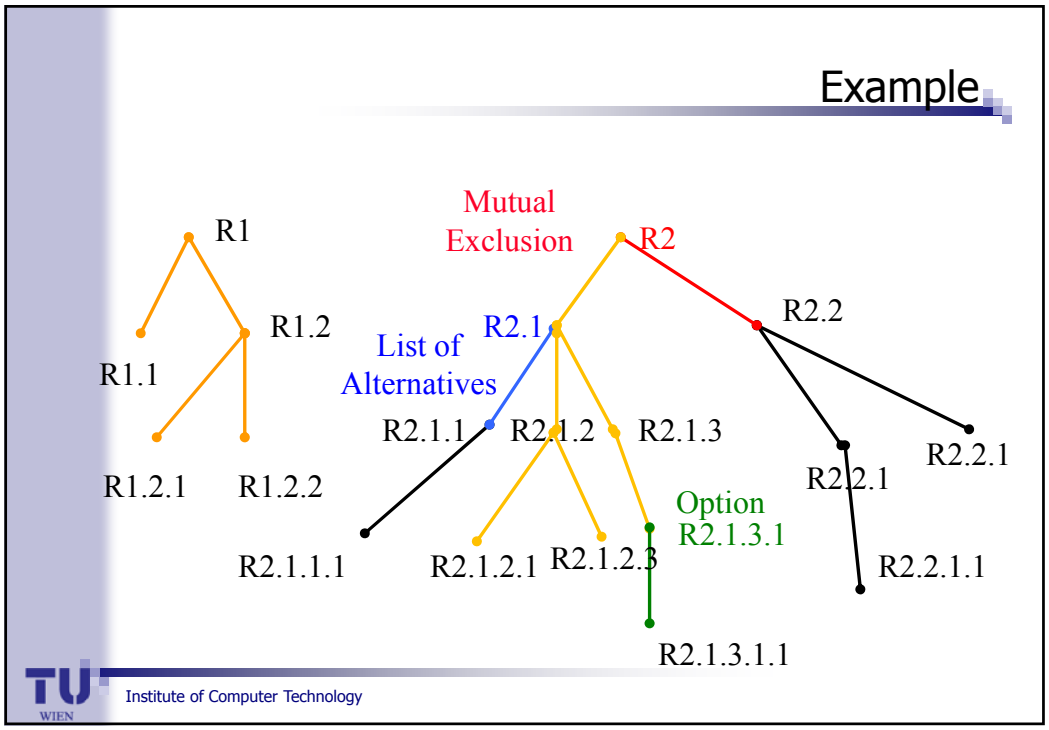
## Variation Point-based selection

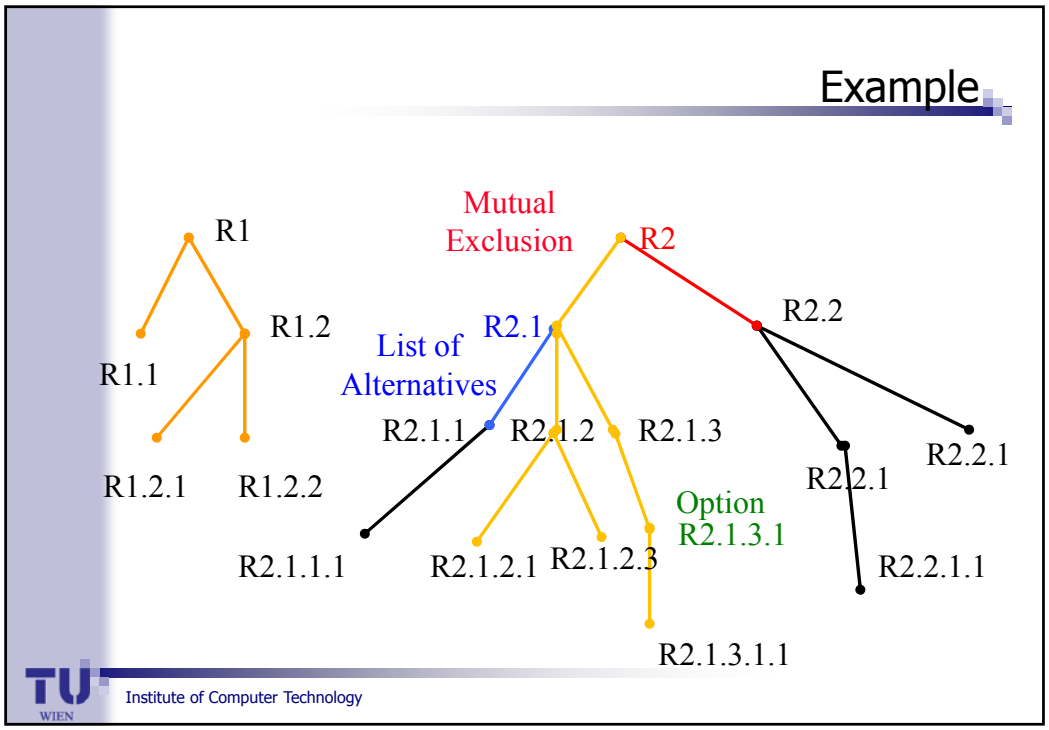
- Use tree structure and Variation Points to direct requirements selection.
- Start at one of the roots.
- Traverse depth first.
- Ask user to make a choice at each Variation Point.
- Common requirements are automatically selected if their parents are already selected or if they are a root node.

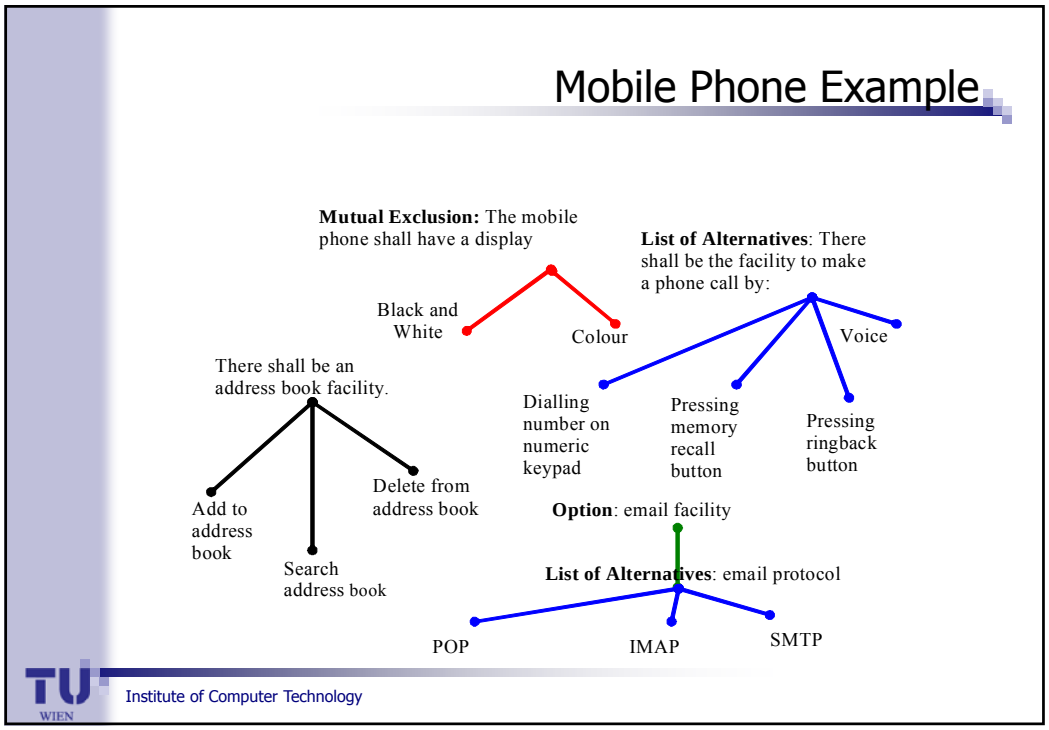

Example
Mutual
Exclusion
R1
R2
R1.2
R2.1
R2.2
List of
Alternatives
R1.1
R2.1.1
R2.1.2
R2.1.3
R2.2.1
R2.2.1
R1.2.1
R1.2.2
Option
R2.1.3.1
R2.1.1.1
R2.1.2.1
R2.1.2.3
R2.2.1.1
R2.1.3.1.1
TU WIEN
Institute of Computer Technology

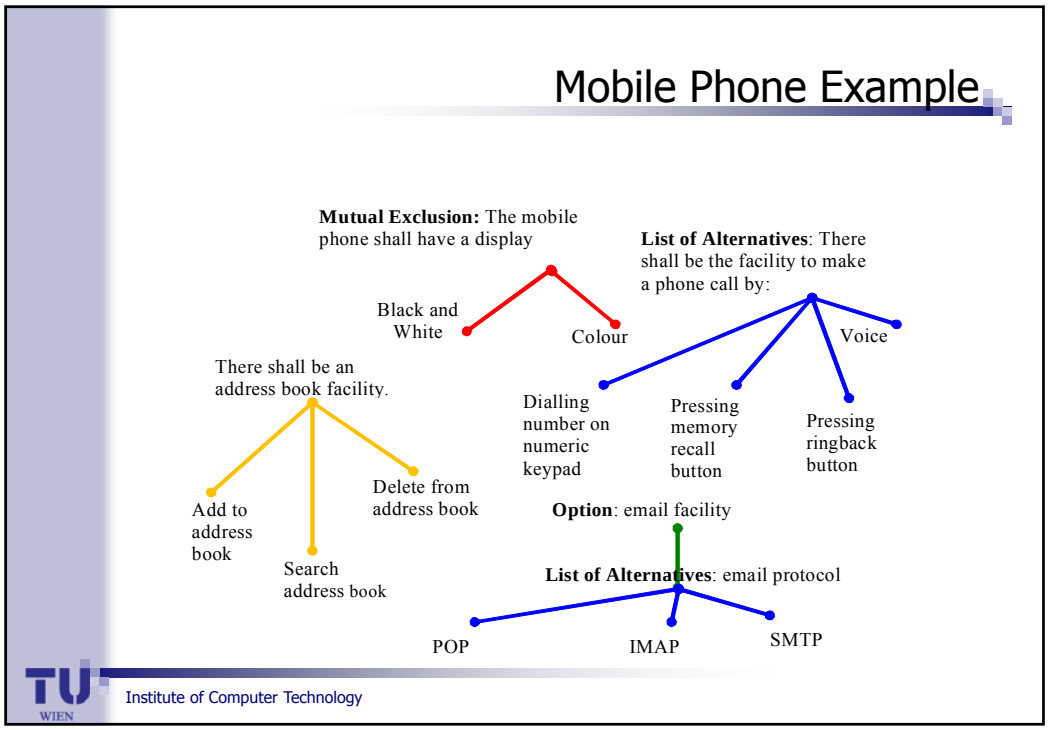

Example
Mutual
Exclusion
R1
R2
R1.2
R2.1
R2.2
List of
Alternatives
R1.1
R2.1.1
R2.1.2
R2.1.3
R2.2.1
R2.2.1
R1.2.1
R1.2.2
Option
R2.1.3.1
R2.1.1.1
R2.1.2.1
R2.1.2.3
R2.2.1.1
R2.1.3.1.1
TU WIEN
Institute of Computer Technology

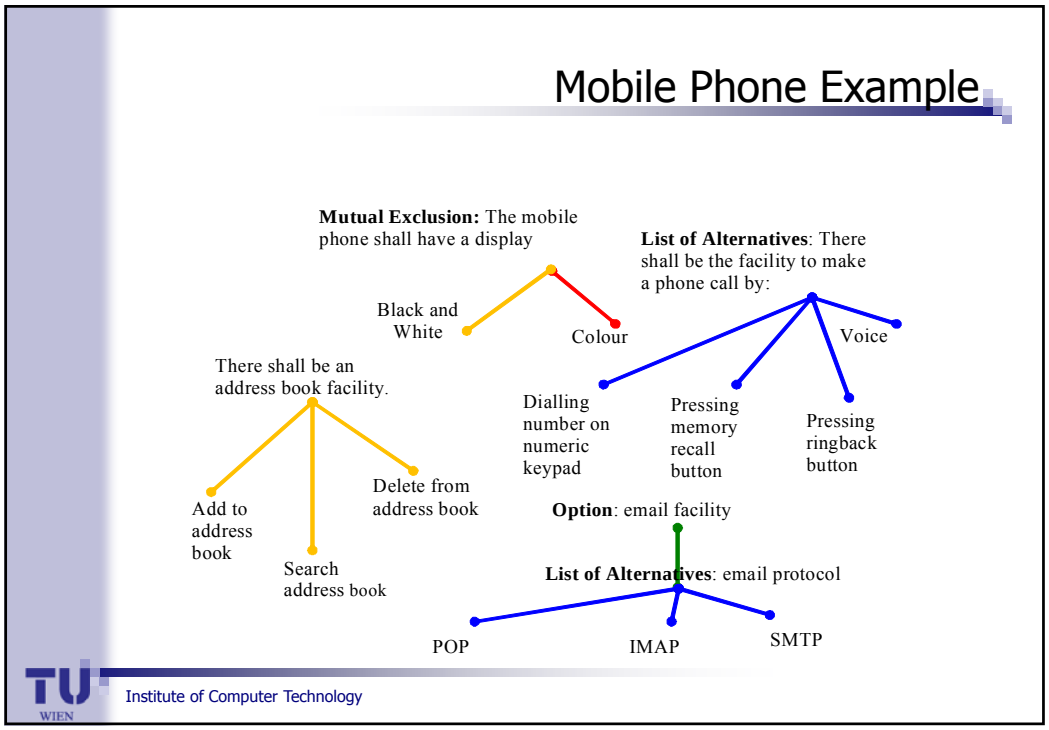

Example
R1
R1.1
R1.2
R1.2.1
R1.2.2
Mutual
Exclusion
R2
R2.1
List of
Alternatives
R2.1.1
R2.1.2
R2.1.3
R2.1.1.1
R2.1.2.1
R2.1.2.3
Option
R2.1.3.1
R2.1.3.1.1
R2.2
R2.2.1
R2.2.1
R2.2.1.1
Institute of Computer Technology

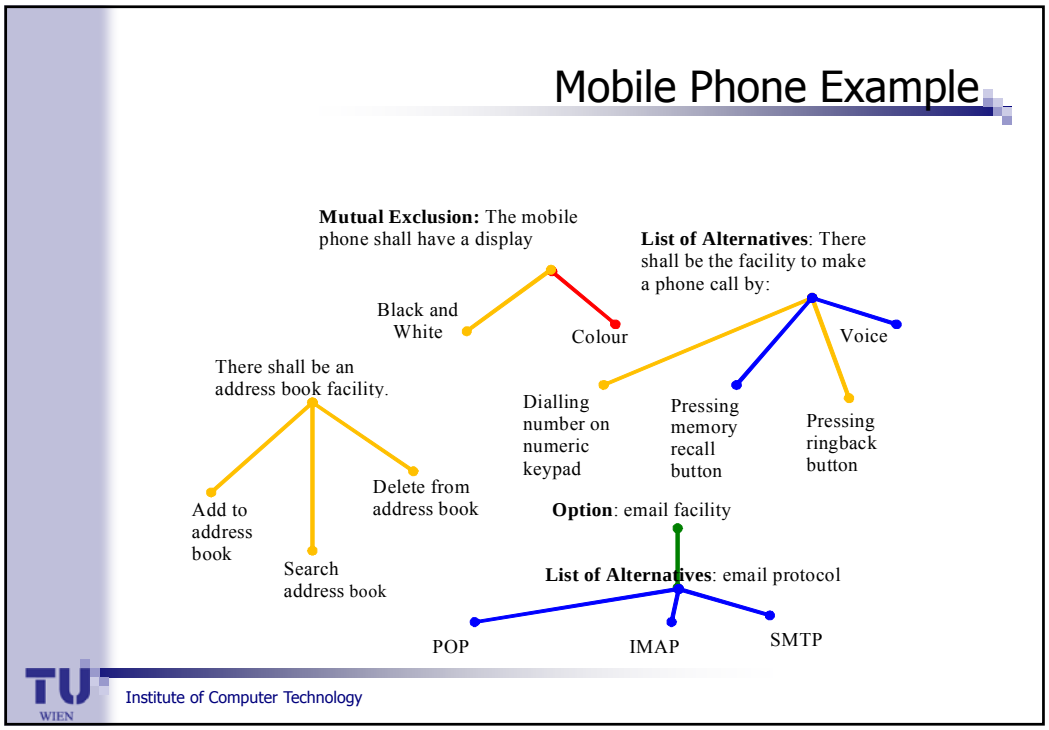

Example
R1
R1.1
R1.2
R1.2.1
R1.2.2
Mutual
Exclusion
R2
R2.1
List of
Alternatives
R2.1.1
R2.1.2
R2.1.3
R2.1.1.1
R2.1.2.1
R2.1.2.3
Option
R2.1.3.1
R2.1.3.1.1
R2.2
R2.2.1
R2.2.1
R2.2.1.1
Institute of Computer Technology

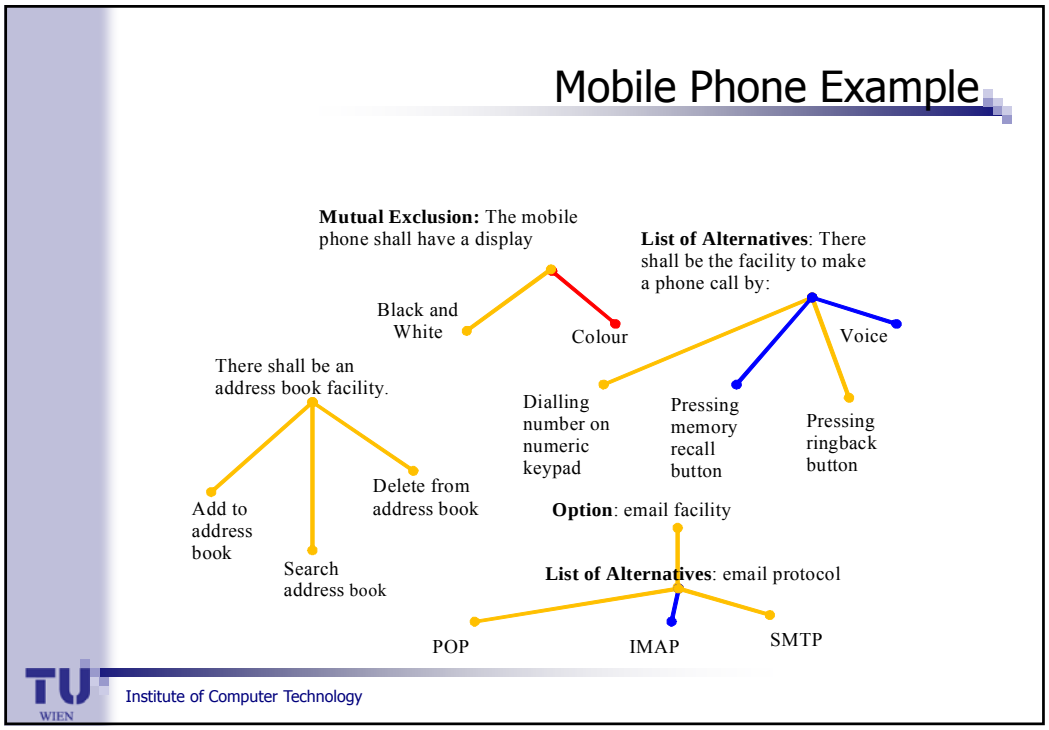

Example
R1
R1.1
R1.2
R1.2.1
R1.2.2
Mutual Exclusion
R2
R2.1
List of Alternatives
R2.1.1
R2.1.2
R2.1.3
R2.1.1.1
R2.1.2.1
R2.1.2.3
Option R2.1.3.1
R2.1.3.1.1
R2.2
R2.2.1
R2.2.1
R2.2.1.1
Institute of Computer Technology


Mobile Phone Example
Mutual Exclusion: The mobile phone shall have a display
Black and White
Colour
List of Alternatives: There shall be the facility to make a phone call by:
Voice
Dialling number on numeric keypad
Pressing memory recall button
Pressing ringback button
There shall be an address book facility.
Add to address book
Search address book
Delete from address book
Option: email facility
List of Alternatives: email protocol
POP
IMAP
SMTP
Institute of Computer Technology


Mobile Phone Example
Mutual Exclusion: The mobile phone shall have a display
Black and White
Colour
List of Alternatives: There shall be the facility to make a phone call by:
Voice
Dialling number on numeric keypad
Pressing memory recall button
Pressing ringback button
There shall be an address book facility.
Add to address book
Search address book
Delete from address book
Option: email facility
List of Alternatives: email protocol
POP
IMAP
SMTP
Institute of Computer Technology

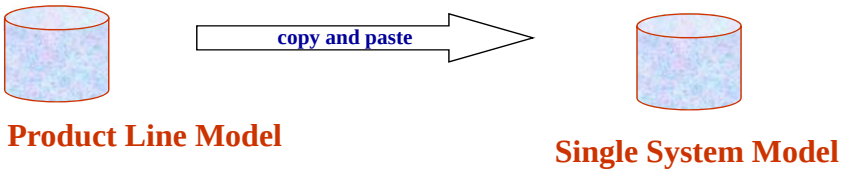

Mobile Phone Example
Mutual Exclusion: The mobile phone shall have a display
Black and White
Colour
List of Alternatives: There shall be the facility to make a phone call by:
Voice
Dialling number on numeric keypad
Pressing memory recall button
Pressing ringback button
There shall be an address book facility.
Add to address book
Search address book
Delete from address book
Option: email facility
List of Alternatives: email protocol
POP
IMAP
SMTP
Institute of Computer Technology


Mobile Phone Example
Mutual Exclusion: The mobile phone shall have a display
Black and White
Colour
List of Alternatives: There shall be the facility to make a phone call by:
Voice
Dialling number on numeric keypad
Pressing memory recall button
Pressing ringback button
There shall be an address book facility.
Add to address book
Search address book
Delete from address book
Option: email facility
List of Alternatives: email protocol
POP
IMAP
SMTP
Institute of Computer Technology


Mobile Phone Example
Mutual Exclusion: The mobile phone shall have a display
Black and White
Colour
List of Alternatives: There shall be the facility to make a phone call by:
Voice
Dialling number on numeric keypad
Pressing memory recall button
Pressing ringback button
There shall be an address book facility.
Add to address book
Search address book
Delete from address book
Option: email facility
List of Alternatives: email protocol
POP
IMAP
SMTP
Institute of Computer Technology

## 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.






## Problems of free selection

- Selecting a single requirement is often not sufficient.
- Random choice can mean illegal choice
  - e.g., two mutually exclusive requirements
  - e.g., not choosing common requirement
- Untenable number of choices
- However, engineers like freedom of choice!

## Product Line Model relations and formal definitions

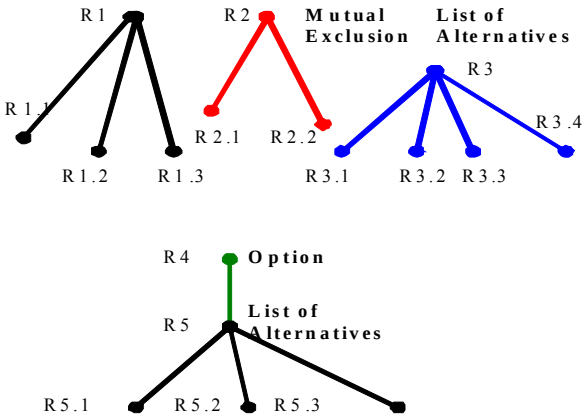
| Product Line Relation | Formal Definition |
|---|---|
| Product line model | $T_1 \wedge T_2 \wedge \ldots \wedge T_n$ |
| Tree (T) | $a_1$ Я $a_2$ Я ... Я $a_n$ |
| Parent-Child | $a_i \wedge a_j$ |
| Mutual Exclusion Variation Point | $a_i \oplus a_j$ |
| List of Alternatives Variation Point | $a_i \vee a_j$ |
| 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 $E(P) = \{T_1 \wedge T_2 \wedge \ldots \wedge T_n$ ¦ $\{T_i = a_{i1}$ Я$_{i1}$ $a_{i2}$ Я$_{i2}$ $a_{i3}$ Я$_{i3}$ ... 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$_{ij}$ ϵ {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)
- 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.

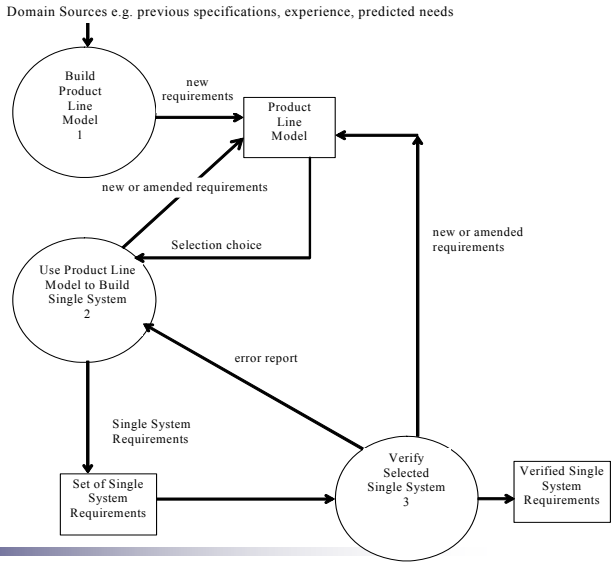
*R1.3 not selected*

## Verifying consistency

- Easy to automate based on Propositional Logic
- Helps verifying 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.
- Invites the engineer to consider reworking the model.



## MRAM process model



## Summary of our product line approach

- Our product line approach is based on *Feature-Oriented Domain Analysis* (FODA).
- It supports representing commonality and variability of requirements explicitly.
- It covers both reuse (by selecting from) and reusability (by creating a repository of product line requirements).



## Outline

- Introduction and background
- Requirements R&R in product lines
- Software R&R involving case-based reasoning
- R&R for business knowledge and software
- Contrasting these approaches
- Summary and conclusion

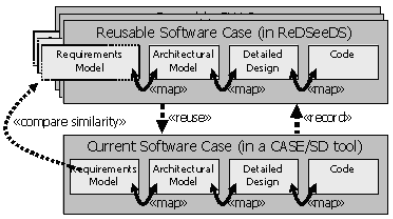
## Context of this research

- European Union ReDSeeDS project
  - Requirements Driven Software Development System
  - Contract number IST-2006-033596
  - www.redseeds.eu
- Scenario-driven development method
- Reuse and tool support
- Case-based approach

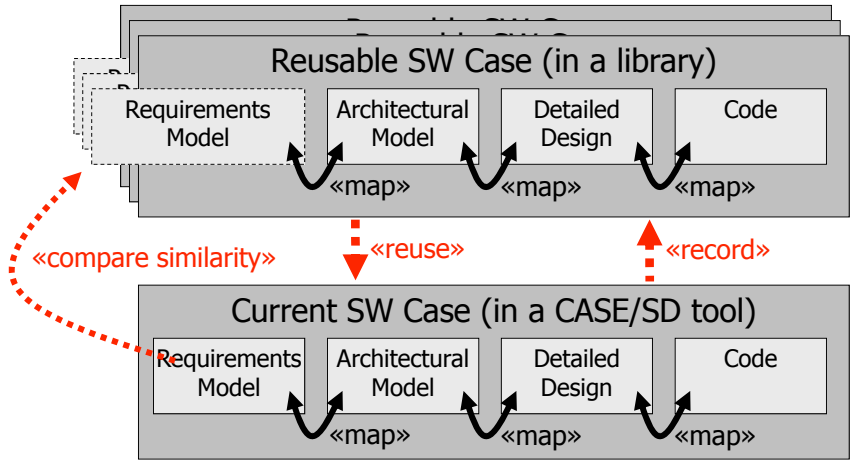



## Essence

- Requirements reuse organized around specific software cases stored in repositories
- Employs similarity metrics for finding good candidates
- Kind of *Case-based Reasoning*
- Works even for *partially* developed requirements



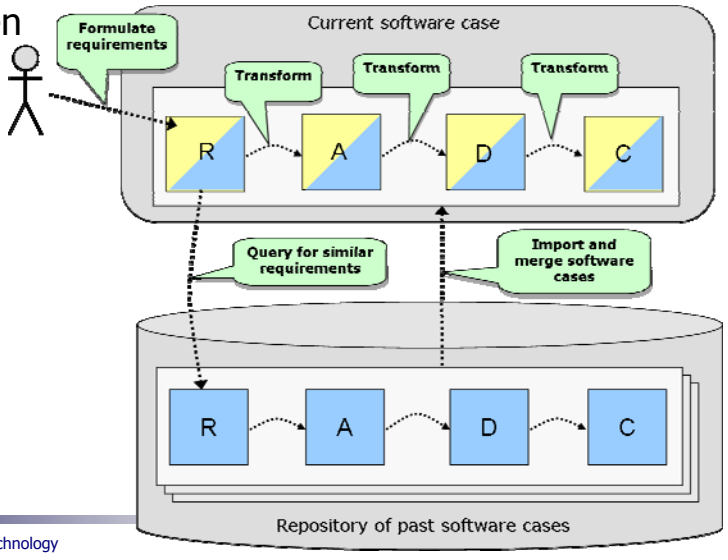
## Reuse Approach




Institute of Computer Technology

## Software cases

- Model-driven and
- Case-based combined



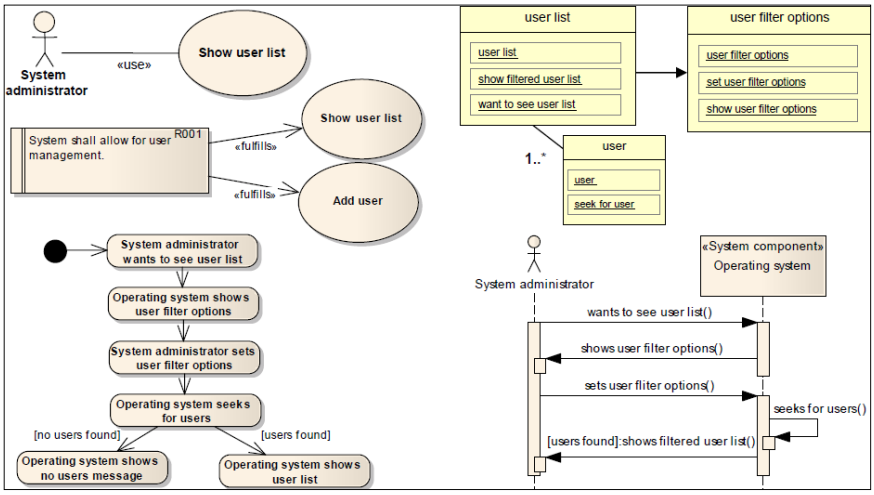

Institute of Computer Technology

## Requirements-based reuse utilizing similarity

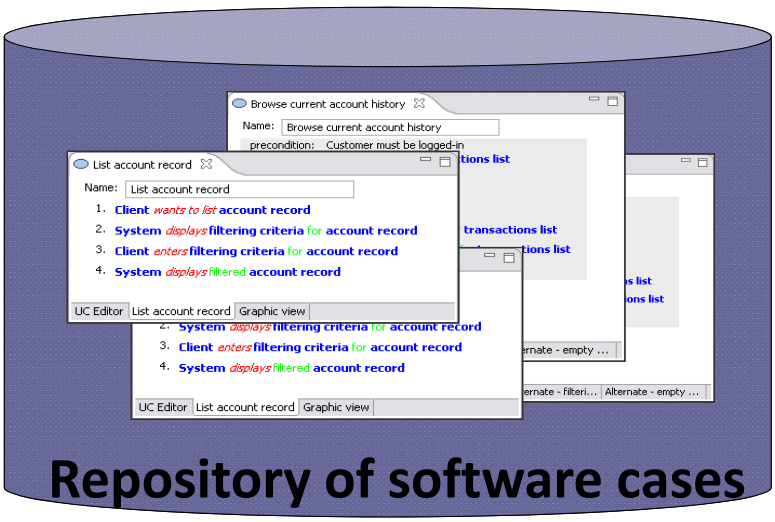
- Definition of similarity based on
  - graph similarity
  - "semantic" similarity
- Similarity actually measured on representations (in specific languages)
- Still, as much "semantics" used as possible



## Requirements Specification Language

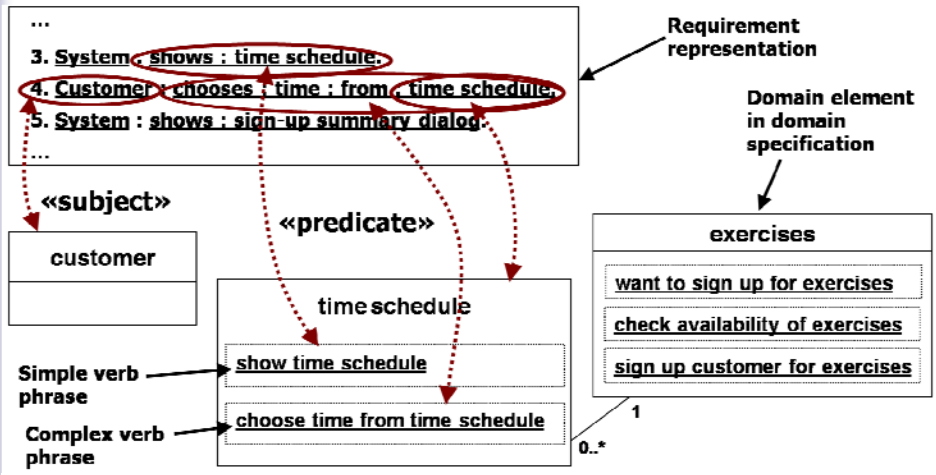


## Software cases

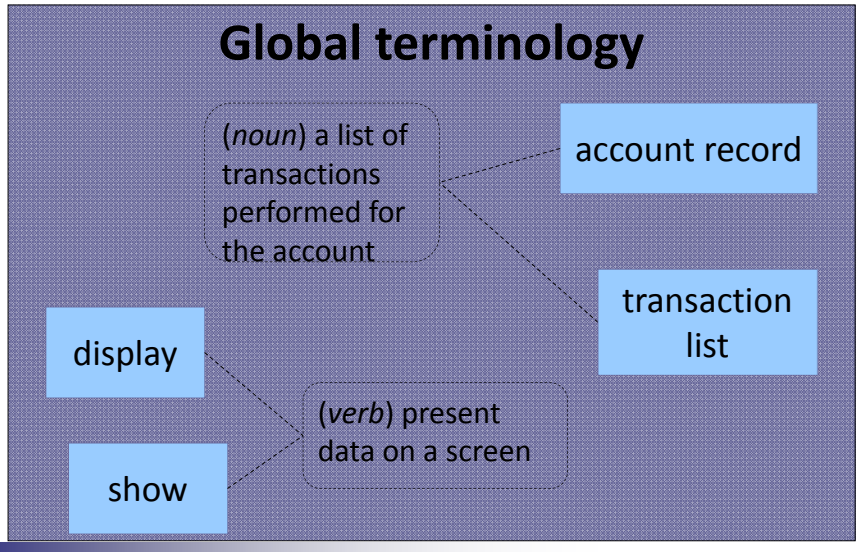





## Requirements representations referring to vocabulary elements



## Links to terminology

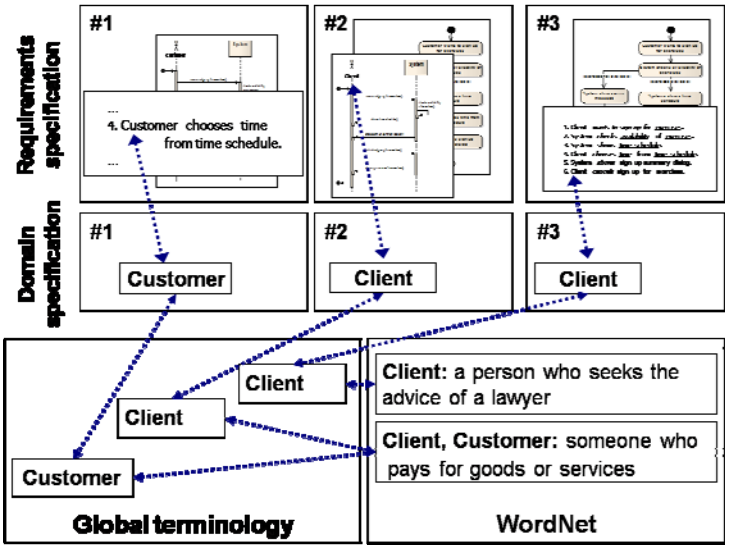





## Integration of WordNet

- **Problem:** To make requirements comprehensible to humans, the used terminology must be defined precisely. This is a complex and time-consuming task.
- **Solution:** Reuse terminology definitions from the semantic lexicon WordNet
- **Advantages**
  - Most of the used words of the English language are predefined.
  - Knowledge about semantic relations between the words is available.

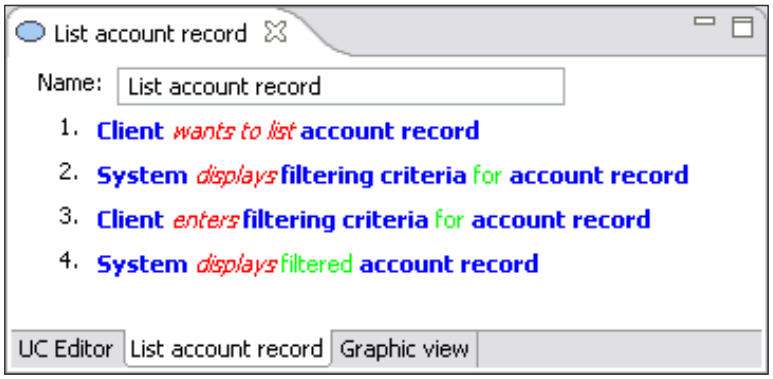
## Integration of WordNet



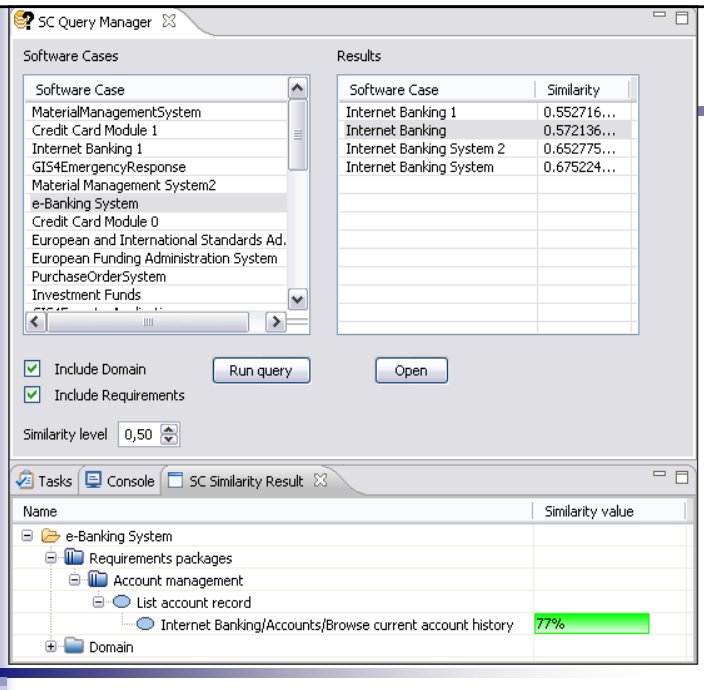



## Partial requirements specification / scenario

- Finding cases works even for a single scenario



## Tool




Institute of Computer Technology

## How to use found software cases

- 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.

Institute of Computer Technology

## Summary of case-based approach

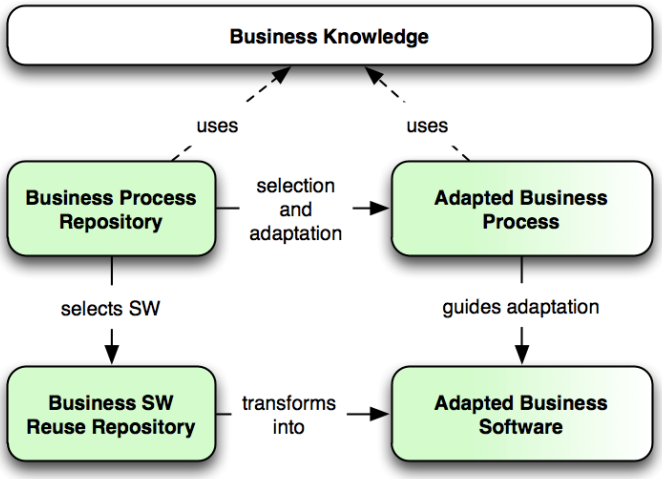
- Our case-based approach allows reuse without the usual and significant effort for making software explicitly reusable.
- Supporting the reuse for only *partially* developed requirements is important, since it allows reuse already without the need to develop a "complete" requirements specification first.
- It even facilitates the *reuse of requirements*.
- *Even a single new scenario* may be sufficient for finding relevant cases automatically.



## Outline

- Introduction and background
- Requirements R&R in product lines
- Software R&R involving case-based reasoning
- R&R for business knowledge and software
- Contrasting these approaches
- Summary and conclusion

## Business software reuse approach

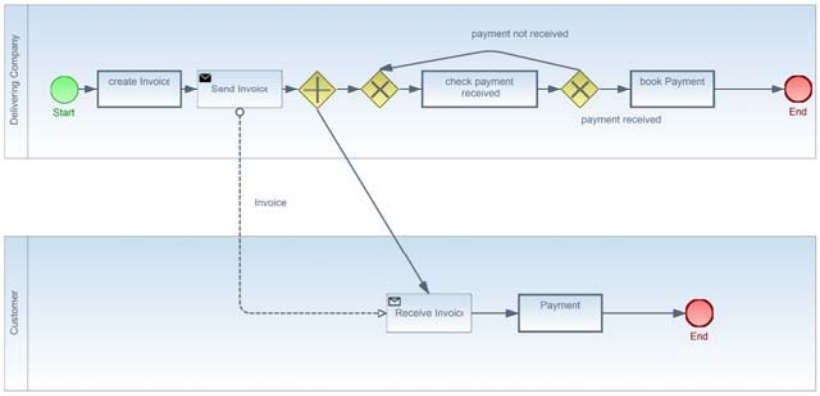





## BPMN (Business Process Model and Notation)

- Merged the two worlds of
  - operating department and
  - IT department
- BPMN 2.0
  - Graphical representation and
  - standardized XML-based format
- Execution using attached objects or services?

## Reference process example in BPMN






## Automated adaptation based on Business Rules

- Through model transformations specifying Business Rules
- When business process models try to capture all details, they are complex and need to deal with variability according to context.
- Also changing over time

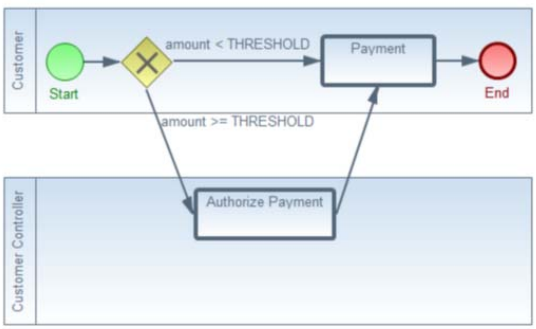
## Business rule with Sub-Process creation

- Payment is to be handled differently in a given business depending on the amount to be paid.
- Substitution of Activity named Payment with Sub-Process named Authorized Payment.



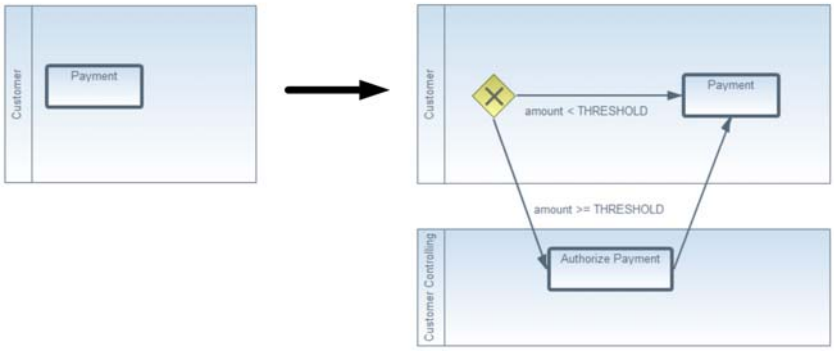

## Sub-Process Authorize Payment

- If this amount exceeds a defined threshold, another business actor needs to authorize the payment before its execution.



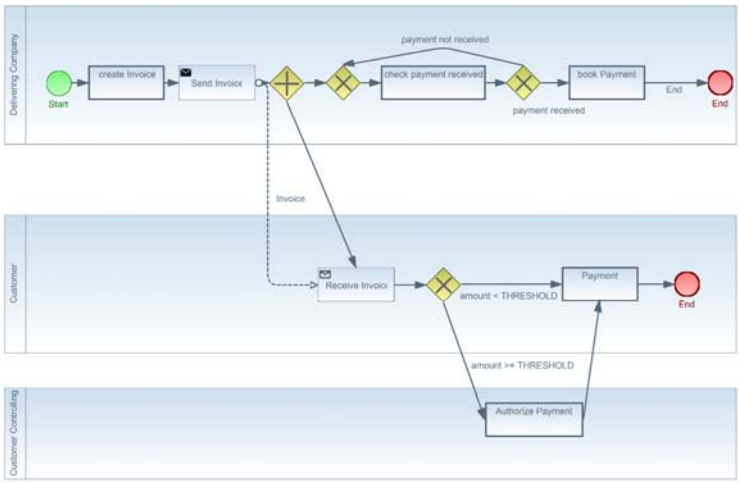
## Business Rule with "in-situ" substitution

- At the same place
- Of given Payment Activity with the conditional payment authorization






## In-situ changed process

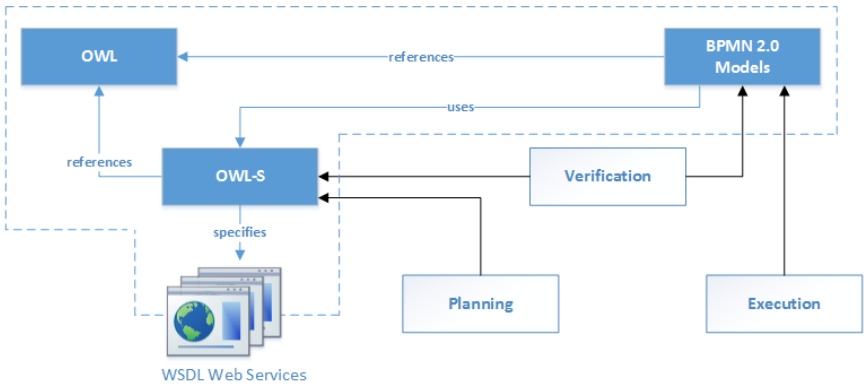


## Using model transformations

- Automatic adaptation of reference processes through model transformations
- Representation of Business Rules, e.g., in ATL
- ATL engine does not by itself allow for substituting something in a whole model.
- Explicit transformations for generating the unchanged parts of the model, in addition to rules for changes
- 'Generic' transformation rule that can be auto-matically instantiated for the unchanged model part

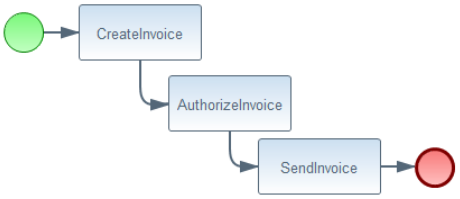


## Bigger picture



## Use of business ontologies

- Reference ontologies (represented in OWL)
- Semantic specification of services, e.g., using OWL-S
  - Input and Output (corresponding, e.g., to WSDL)
  - Precondition and Postcondition
- Example:
  *Send Invoice*
  Input: Invoice
  Output: none
  Precondition: none
  Postcondition: sent (Invoice, true)






## Business process verification using Logic

- Representation in Fluent Calculus
- Tool FLUX
- For a given business process defined as a composed service, automated verification against the specifications of the single services
- Problem of potential over-specification of services leading to mismatch with service implementation

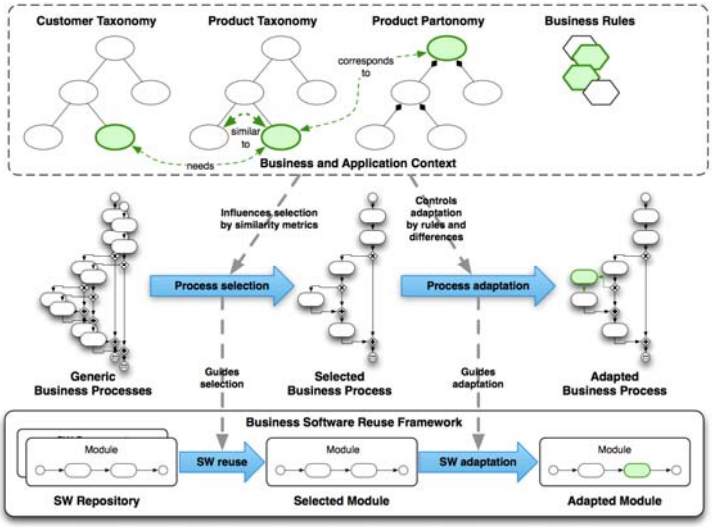
HICSS'15

## Business process generation through planning

- Automatic planning in FLUX tool based on the same service specifications
- For a given goal condition, composition of services achieving it
- Verifyably correct, but not necessarily valid business processes



## Vision of Business Process-driven automated software development and reuse



## Summary of business knowledge approach

- Automated adaptation of business processes based on Business Rules
- Business process verification and generation using
  - business ontologies (in OWL)
  - OWL-S for semantic service specification
  - Fluent Calculus based on formal Logic
- Work in progress



## Outline

- Introduction and background
- Requirements R&R in product lines
- Software R&R involving case-based reasoning
- R&R for business knowledge and software
- Contrasting these approaches
- Summary and conclusion

## Reusable assets

- Product lines
  - Requirements
  - Features
- Case-based
  - Requirements
  - Software artefacts
- Business knowledge and software
  - Business processes
  - Services



## Reuse approaches

- Product lines
  - Systematic selection from product line
  - Possibly semi-automatic selection
- Case-based
  - Finding similar cases
  - Adaptation of most similar one(s)
- Business knowledge and software
  - Adaptation of business processes
  - Service composition

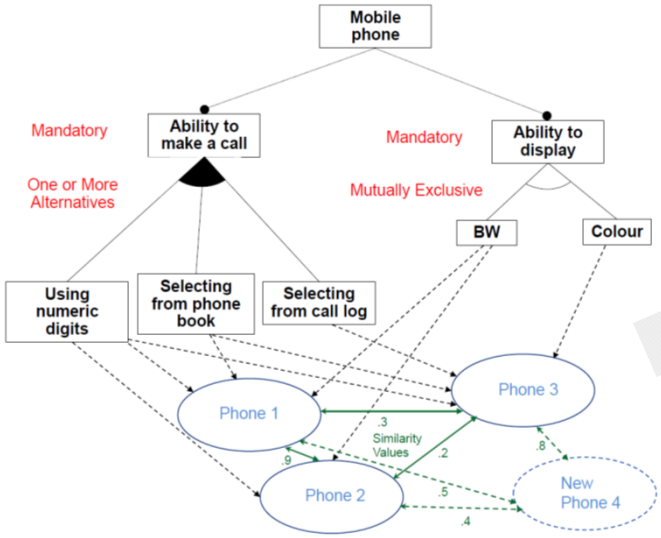
## Costs vs. benefits of case-based approach and product lines

| | SPLE | CBR |
|---|---|---|
| **Costs of Making *Reusable*** | Substantial | Negligible |
| **Benefits for *Reuse*** | Facilitates automated product derivation | Facilitates finding similar cases for reuse |

ICSR'15



## Integration of case-based approach with product lines – Feature-Similarity Model




ICSR'15

## Outline

- Introduction and background
- Requirements R&R in product lines
- Software R&R involving case-based reasoning
- R&R for business knowledge and software
- Contrasting these approaches
- Summary and conclusion



## Summary and Conclusion

- Commonality and variability of requirements explicitly represented for reuse and reusability in the context of product lines
- Reuse of requirements and software artefacts using case-based reasoning
- Automated adaptation, verification and generation of business processes
- Software reuse based on business processes and requirements
- Manifold reuse possibilities on high conceptual level

Thank you for your attention!

???

Institute of Computer Technology

## Selected work of this tutorial presenter


- 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 (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.


Institute of Computer Technology

## Selected work of this tutorial presenter (cont.)


- M. Mannion, B. Keepence, H. Kaindl and J. Wheadon, Reusing Single System Requirements from Application Family Requirements, in *Proc. Twenty-first International Conference on Software Engineering (ICSE-99)*, Los Angeles, CA, May 1999, 453–462, ACM.
- M. Mannion, O. Lewis, H. Kaindl, G. Montroni and J. Wheadon, 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'00), 2000, 153–169.
- Popp, R., Kaindl, H.: Automated adaptation of business process models through model transformations specifying business rules. In Nurcan, S., Pimenidis, E., Pastor, O., Vassiliou, Y., eds.: *Proceedings of the CAiSE'14 Forum at the 26th International Conference on Advanced Information Systems Engineering (CAiSE)*, Thessaloniki, Greece June, 2014. Volume 1164 of CEUR Workshop Proceedings.
- K. Wolter, M. Smialek, D. Bildhauer, H. Kaindl, Reusing Terminology for Requirements Specifications from WordNet, in *Proceedings of the 16th IEEE International Requirements Engineering Conference (RE 2008)*, 2008, 325-326.