METHOD CHUNK FEDERATION

ISABELLE MIRBEL

LABORATOIRE I3S

Route des Lucioles, BP 121

06903 Sophia Antipolis, Cedex

France

isabelle.mirbel@unice.fr

EMMSAD 2006

OUTLINE

- Need for customization
- Situational Method Engineering
- Organization-wide standard approaches

- Merging of project specific methods
  - Capture & understanding of all the project specific methods by method engineers
  - Acceptation & understanding of the organization wide method by method users

- Make project specific method federable
  - Break down method into meaningful atomic parts
  - Qualify meaningful atomic parts
- Federate federable method parts

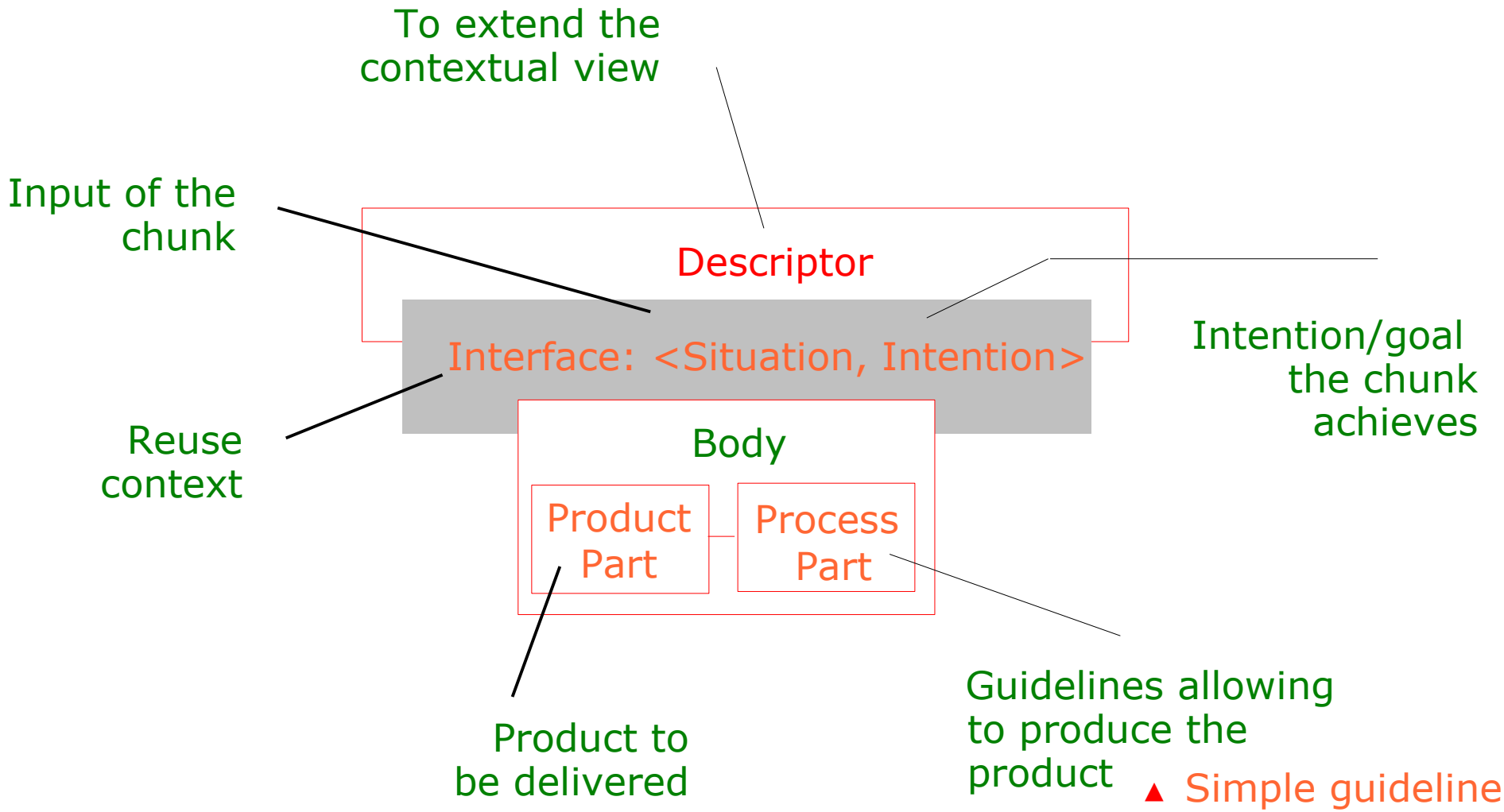
OUTLINE

METHOD MEANINGFUL ATOMIC PARTS (1)

- Method chunk from J. Ralyté
  - Process & product dimensions
  - Assembly techniques provided

METHOD CHUNK: Autonomous & coherent part of a method supporting the realization of some specific Information System & Software Development (ISSD) activities

METHOD: Set of loosely coupled method chunks expressed at different levels of granularity


To extend the contextual view
Input of the chunk
Descriptor
Interface: <Situation, Intention>
Intention/goal the chunk achieves
Reuse context
Body
Product Part
Process Part
Product to be delivered
Guidelines allowing to produce the product
Simple guidelines
Tactical guidelines
Strategic guidelines

△ Method chunk Behavior Investigation

<User Interface Description, Standardize UI business rules behavior>

UML notation:

- Activity diagram
- Class diagram

Some business rules may be generic and associated to the whole software while called in several use-cases and windows. To be coherent through the whole UI specification, these business rules have to be isolated in a specific *class diagram* and explicitly used in the diagrams describing each concerned windows.

A typical example of these generic business rules is the control among two dates (to be sure a first date is before the second one, for instance).

Give a *class diagram* and *activity diagrams* to summarize the general business rules.

△ Taken from the JECKO methodology (I3S/Amadeus)
http://www.i3s.unice.fr/~mirbel/jecko/jecko.html

OUTLINE

- Classification & retrieval techniques
  - Based on user intention & structural relationships
    - Method dependent
  - Based on user intention & application domain
    - Domain dependent

- Based on user intention & ISSD knowledge
  - The REUSE FRAME

- Ontology dedicated to ISSD activities
- Shared by all projects & project members
- Method Chunk descriptor: set of at least one *keyword* taken from the Reuse Frame
- Reuse Frame content proposal

- Universal/critical IS aspects, meaningful reuse *keywords*
  - Human

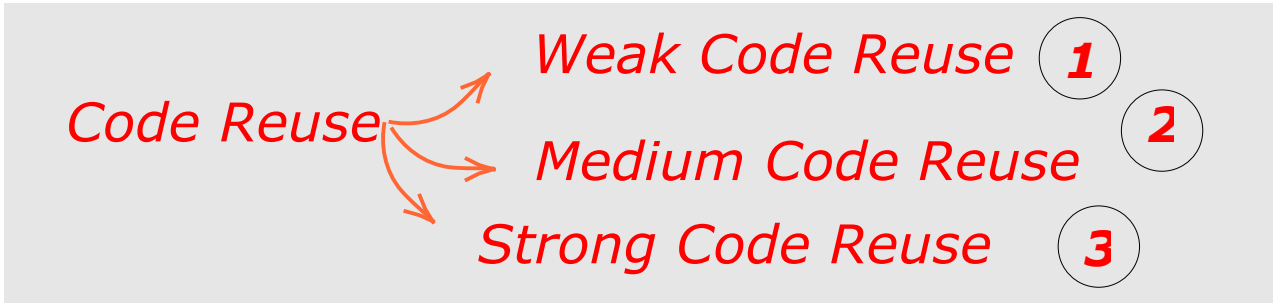
    Expert Analyst
    - Different method users (developer, designer, product definition responsible, test manager, ...)
    - Different levels of expertise (expert, beginner)
  - Organizational

    Business Modelling
    - Contengency factors
    - Project characteristics, goals & assumptions
    - System Engineering Activities
  - Application Domain

    Legacy System
    - Application type
    - Source system
    - Application technology

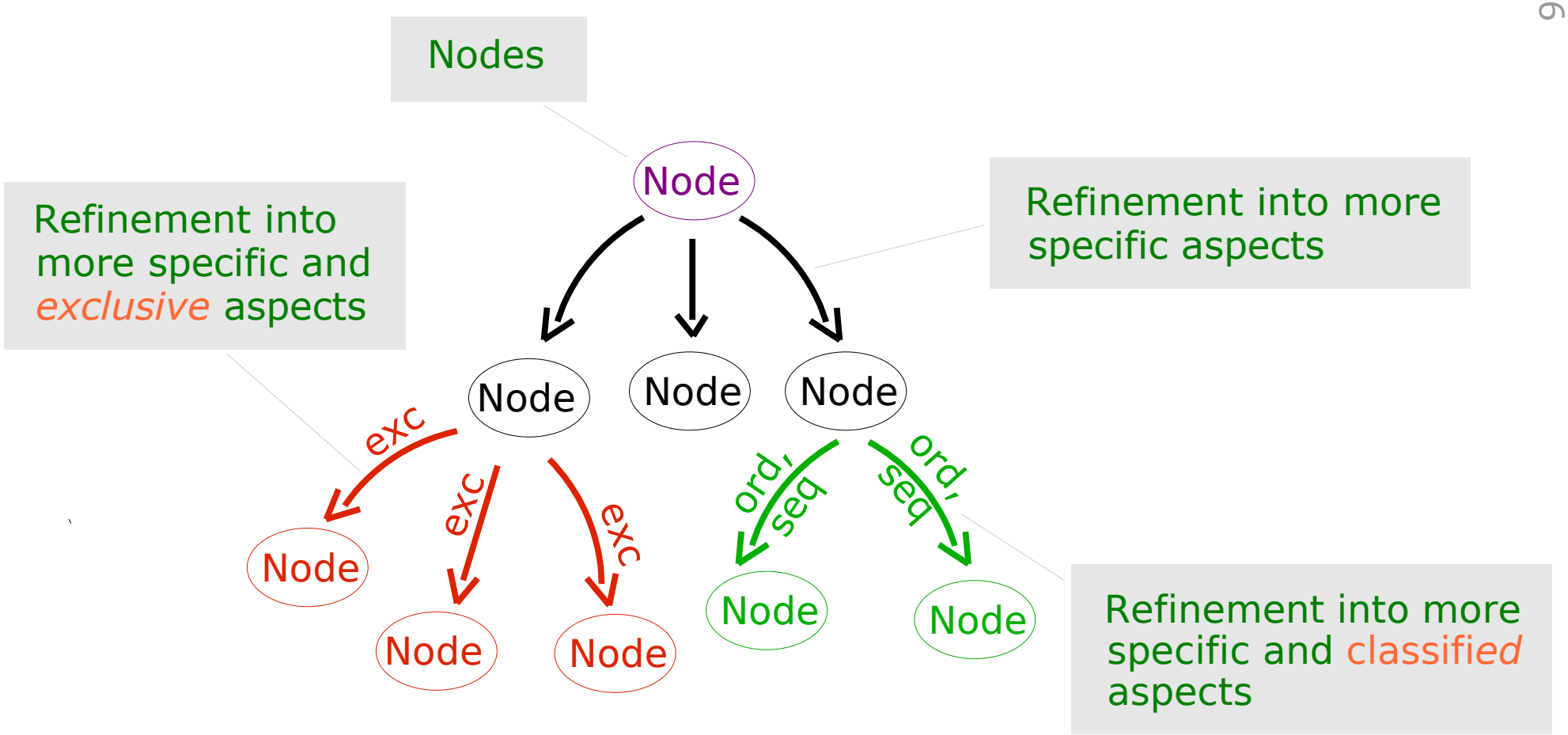
- Refinement relationships
  - Refinement into *more specific* aspects

  - Refinement into more specific and classified aspects



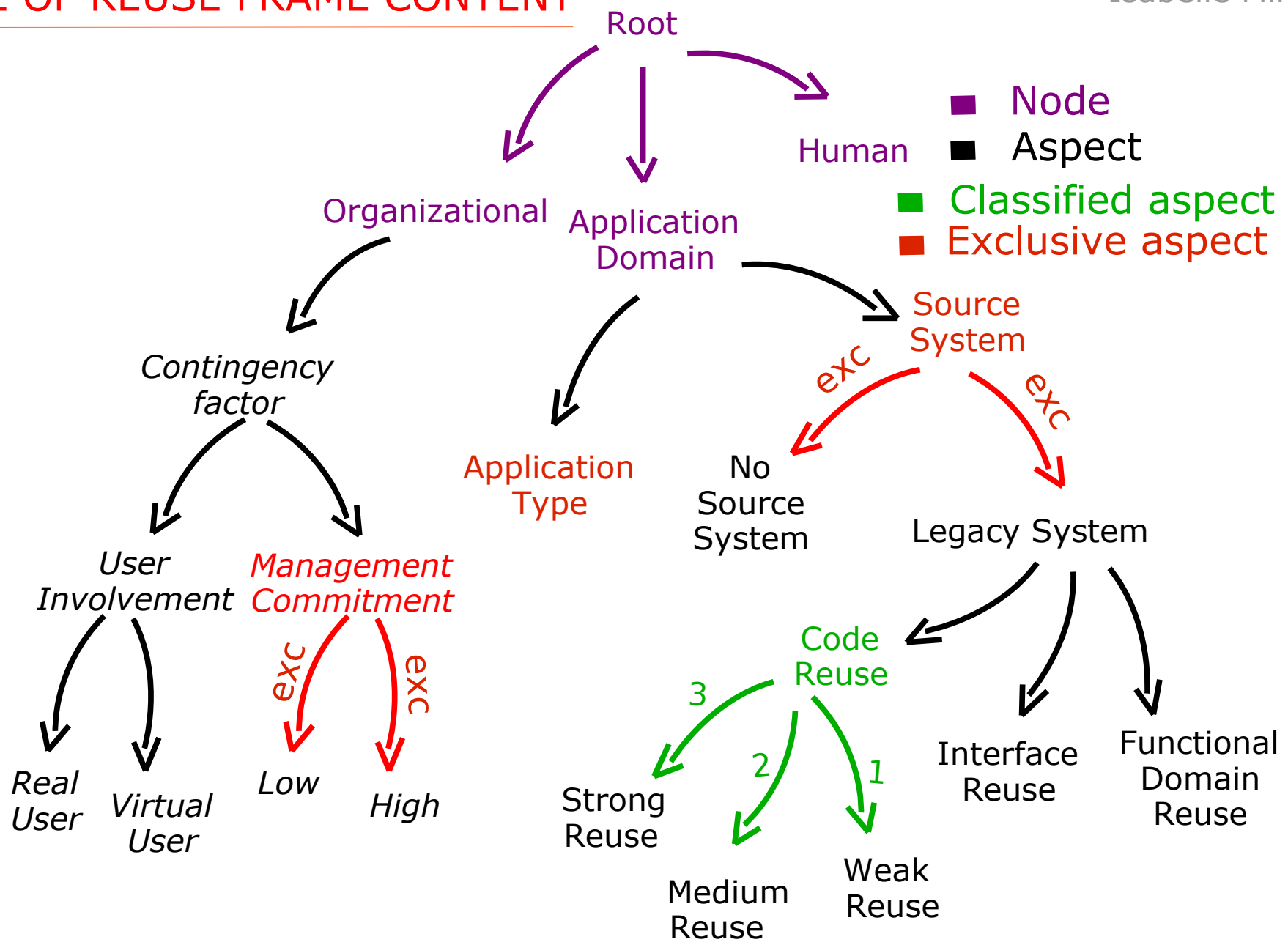
  - Refinement into more specific and *exclusive* aspects

THE REUSE FRAME STRUCTURE (2)



EXAMPLE OF REUSE FRAME CONTENT

THE REUSE FRAME



△ Full description available at :
http://www.i3s.unice.fr/~mirbel/reuse-frame/html/rf.html

Inclusion between aspects

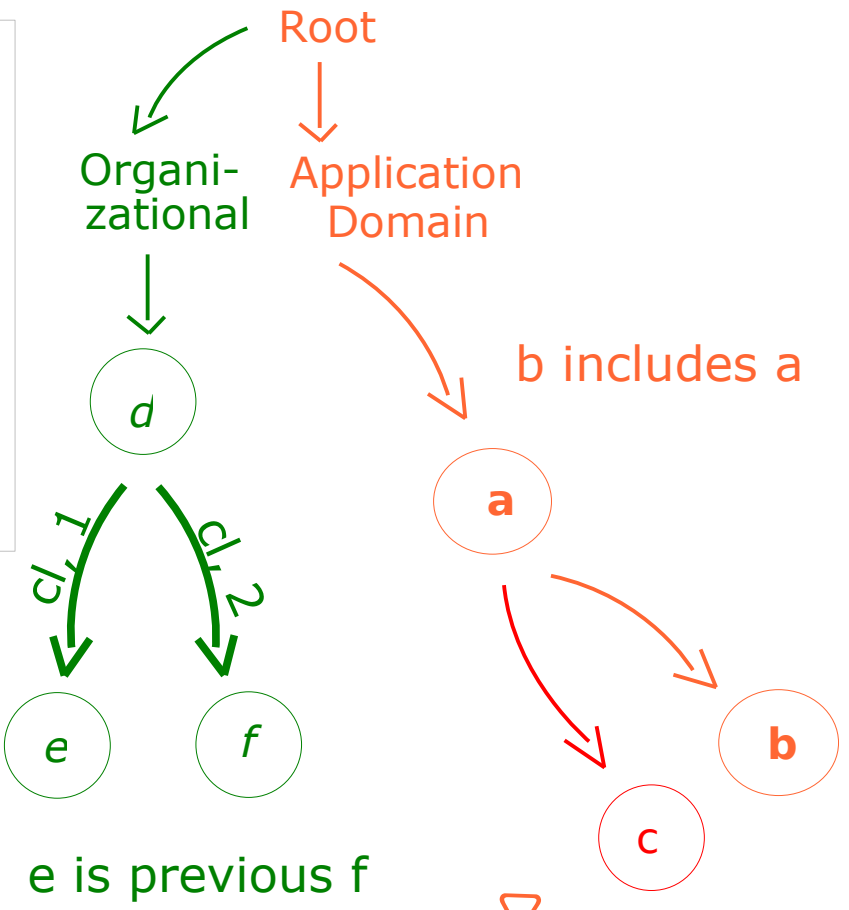
An aspect a1 is included in an aspect a2 if the path from the root node to a1 is a sub-path of the path from the root node to a2.

An aspect a1 includes an aspect a2 if the path from the root node to a2 is a sub-path of the path from the root node to a1.

Precedence between aspects

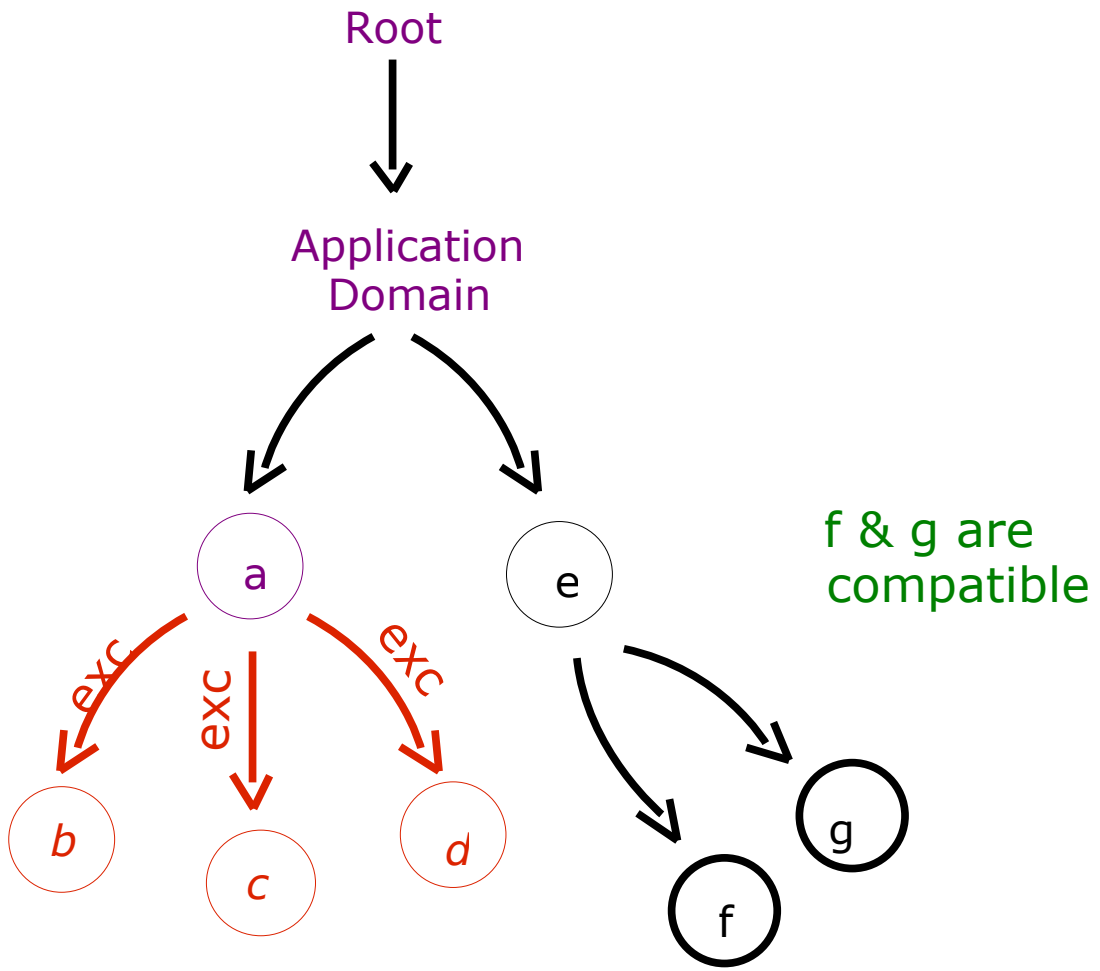
An aspect a1 is previous an aspect a2 if they have the same father node and cl(a1) < cl(a2)

An aspect a1 is next an aspect a2 if they have the same father node and cl(a1) > cl(a2)



△ Compatibility between aspects

Compatible aspects do not share in their path (from the root node) a node with exclusive leaving edges.

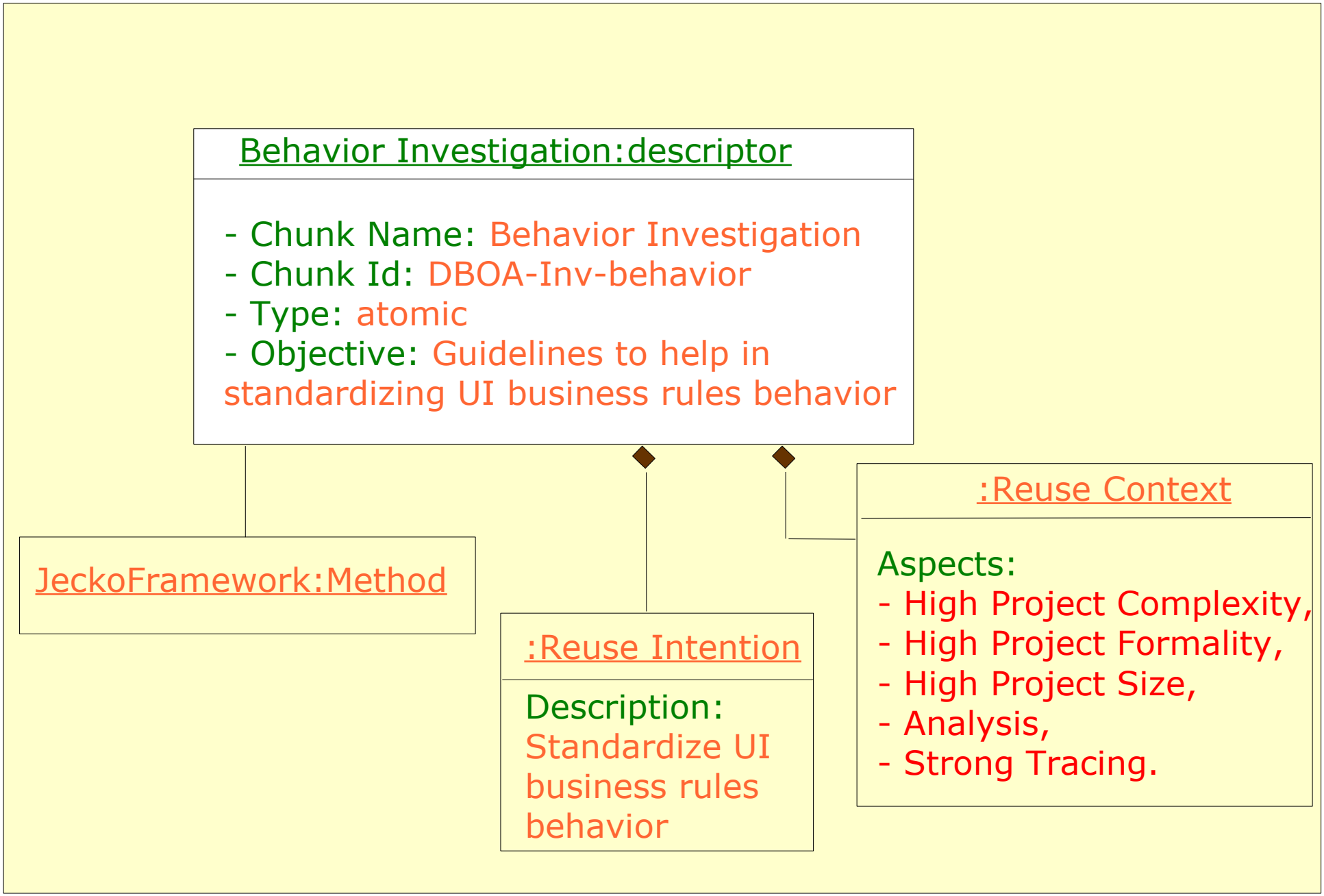


OUTLINE

- Mechanism(s) to extract meaningful method chunks from the federation
- Meaningful method chunks for project members
  - To cope with ISSD activities covered by the project-specific method ▸ Alternative guidelines
    - Similarity Metrics to compare method chunks
  - To cope with ISSD activities not (well) covered by the project-specific method ▸ Complementary guidelines
    - Similarity Metrics to compare method chunks to user need

- A set of at least one aspect to qualify a method chunk
  - Aspects can not be included one in the others
  - All aspects must be compatible among them
- Method chunks providing general guidelines should be qualified with the help of less refined aspects (i.e. close to the root node).
- Method chunks providing specific guidelines should be qualified with the help of more refined aspects (i.e. close to the leaf nodes or leaf nodes themselves).

SUPPORTING METHOD CHUNK FEDERATION



OUTLINE

- A set of at least one aspect to specify necessary aspects
  - All aspects must be compatible among them
  - Searches for general guidelines
    - less refined aspects (close to the root node)
  - Searches for specific guidelines
    - more refined aspects, that is to say aspects close to the leaf nodes or leaf nodes themselves.
- A set of aspects to specify forbidden aspects
  - All aspects must be compatible among them

- Constraints between necessary & forbidden aspect sets
  - No common aspects between necessary & forbidden aspects
- For optimisation purpose (to avoid redundancies)
  - No inclusion between necessary & forbidden aspects
    - It is not possible by definition to find two aspects included one in the other in the same method chunk context
  - No incompatibility between necessary & forbidden aspects
    - It is not possible by definition to find two incompatible aspects in the same methodchunk context

SUPPORTING METHOD CHUNK FEDERATION

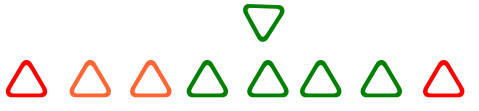
: Method User Situation

Necessary Criteria:

- Graphical User Interface
- Database
- High formality
- Analyst

Forbidden Criteria:

- Design Eng. Activity
- High time pressure



OUTLINE

- Between method chunks
  - Number of common aspects
  - 0 < SM < 1
- Between method chunk & user situation
  - Number of common aspects / necessary aspects
  - Number of common aspects / forbidden aspects
  - SM < 1
  - Does not take into account the number of aspects in the context

- Comparison between method chunks

An aspect from the method chunk context

$$sm(c1,c2) = \frac{\Sigma^{i=1..n}\, d(a_{RCc1_i}, CA_{c2})}{max(card(CA_{c1}), card(CA_{c2}))}$$

Method chunk context

$d(a,A) = 1$ if $a \in A$, 0 else

- $0 < sm(c1,c2) < 1$

An aspect from the forbidden aspect set (Method user situation)

Method chunk context

$$sm(c,s) = \frac{\Sigma^{i=1..n} d(a_{NAsi}, CA_c) - \Sigma^{j=1..m} d(a_{FAsj}, CA_c)}{card(NA_s)}$$

An aspect from the necessary aspect set (Method user situation)

Necessary aspect set (Method user situation)

- $sm(c,s)>0$ ▸ More necessary aspects than forbidden ones
- $sm(c,s)<0$ ▸ More forbidden aspects than necessary ones
- $sm(c,s)=1$ ▸ Perfect adequation between situation & context
- $\frac{-card(CA_c \cap FA_s)}{card(NA_s)}$ ▸ Worst situation with adequation only on forbidden aspects

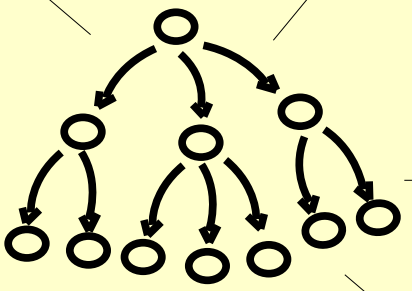
Method Chunk 2

Real User Involvement
Low Time Pressure
Small Project Size

Method Chunk 1

Reuse Context

Real User Involvement
Low Time Pressure
Huge Project Size



sm(MC1,MC2)=0.66
sm(A,MC1)=1
sm(A,MC2)=0.5
sm(B,MC1)=-0.5
sm(B,MC2)=0

User Situation A

Necessary aspects

Real User Involvement
Low Time Pressure

Forbidden aspects

Small Project Size
Low Project Formality

User Situation B

Real User Involvement
Strong Project Traceability

Huge Project Size
Low Time Pressure

OUTLINE

- Federation interest : Ability to propose new & meaningful method chunks
  - Partial matching to be considered
  - Method chunk context included in the method user situation
  - Exploitation of the Reuse Frame relationships
    - more generic aspects
    - more specific aspects
    - previous aspects
    - next aspects

Extension possibilities

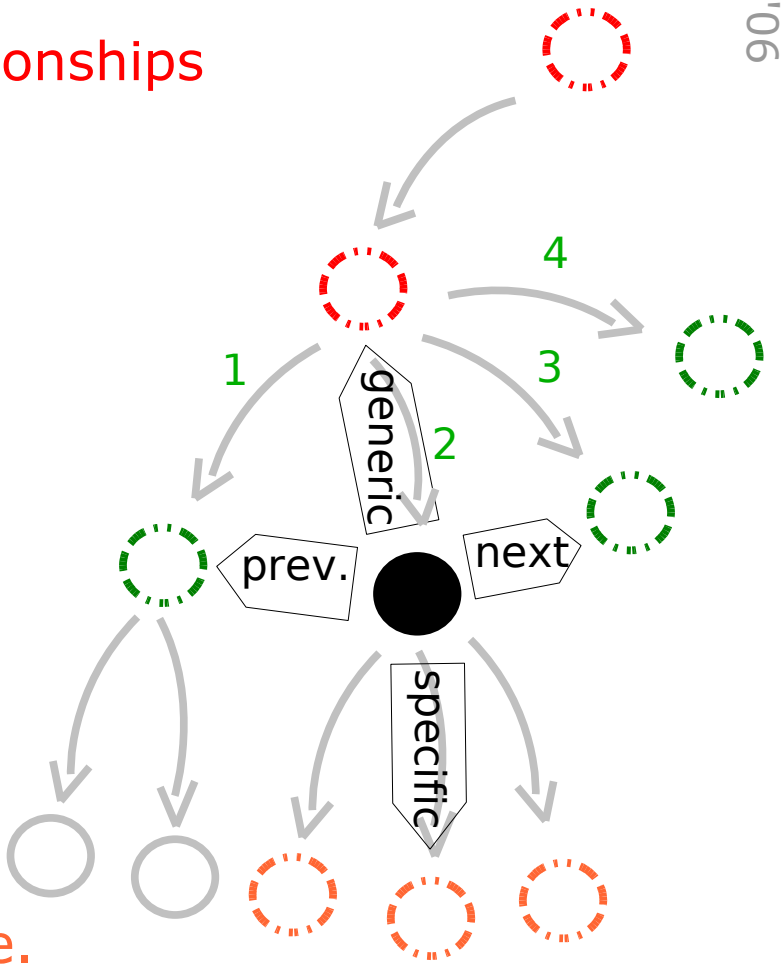
		Extended Matching		
	Exact Matching	Less refined aspects	More refined aspects	before/after aspects
Necessary aspects	to search for method chunks	*to retrieve more method chunks*		
		More general chunks	More specific chunks	Adjacent method chunks
Forbidden aspects	to avoid method chunks	*to retrieve less method chunks*		
		to avoid full branches of the Reuse Frame	to avoid too specific chunks	to avoid adjacent / overlapping chunks

- Closeness between aspects
  - Exploitation of the Reuse Frame relationships

    - Aspect extension

      The extension ext(a) of an aspect a is the set of all the aspects:
      - more generic
      - more specific
      - previous it
      - next it

  - Quantification of the distance with regards to the reference situation (i.e. aspects given by the method user)



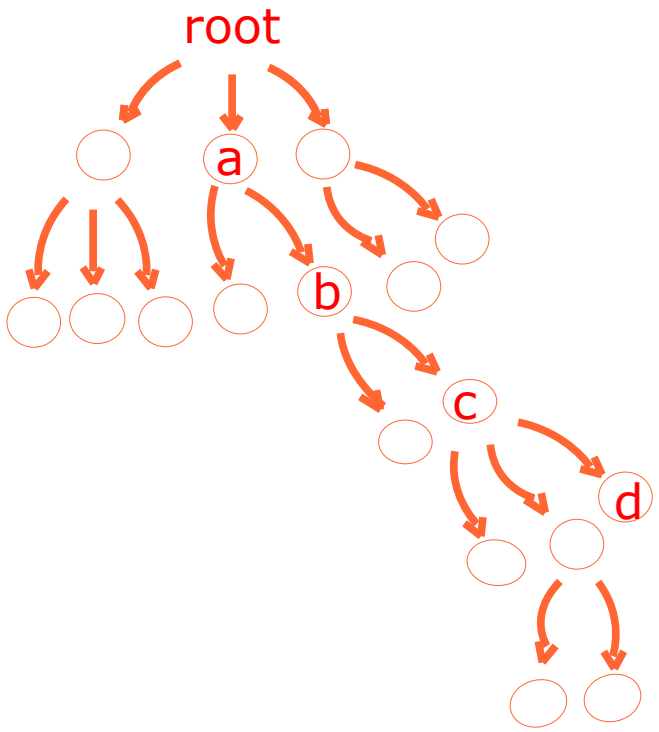
- Example of closeness distance
  - Closeness distance for more generic aspects

a ▸ cd(a,d) = 0.25

b ▸ cd(b,d) = 0.5

c ▸ cd(c,d) = 0.75

d ▸ cd(d,d) = 1

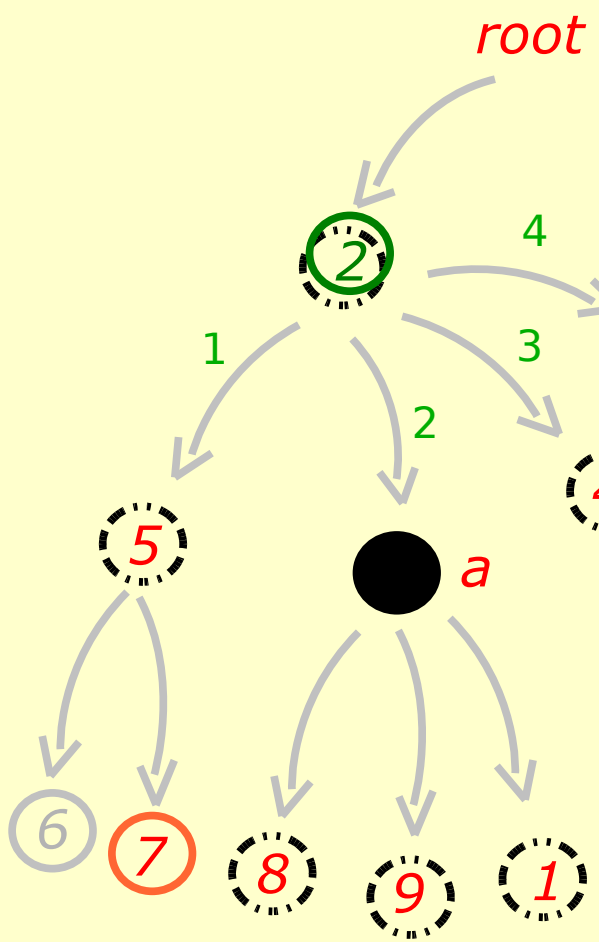


$$sm(c,s) = \frac{\Sigma^{i=1..n} ed(a_{NAsi}, CA_c) - \Sigma^{j=1..m} ed(a_{FAsj}, CA_c)}{card(NA_s)}$$

- if $ext(a) \cap A = \emptyset$, $ed(a,A) = 0$
- else $ed = max\ cd(y,z)$ with
  - $y \in ext(a) \cap A$
  - $z \in A$

EXAMPLE OF EXTENDED SIMILARITY METRICS

SUPPORTING METHOD CHUNK FEDERATION



MC1

2, 14, 23

$ext(a) \cap C_{mc1} = \{2\}$
$ed(a, C_{mc1}) = cl(2,a) = 0,5$
$sm(C_{mc3}, C_{mc1}) = 0,5$

MC2

7, 11

$ext(a) \cap C_{mc2} = \varnothing$
$ed(a, C_{mc2}) = 0$
$sm(C_{mc3}, C_{mc2}) = 0$

US

a

Ext(a) = {root, 2, 5, a, 4, 3, 8, 9, 1}

OUTLINE

- Method Chunk Federation
  - Situational Method Engineering
  - Organization-wide standard approaches
- The Reuse Frame
  - Method Chunk Context
  - Method User Situation
  - Similarity Metrics

- About projects
  - Project specific view & vocabulary layer on the Reuse Frame
  - Dynamic filtering of new and meaningful method chunks
    - Turning context & situation into profiles
    - To compare user profiles
- About method chunks
  - Presentation & integration of the retrieved method chunks in the project-specific methods
  - Method chunk comparison (based on guidelines, ...)
- About the Reuse Frame
  - To move the ontology from the federation to
    - Project level
    - Method chunk level