

Selecting Software Components with Multiple Interfaces

CBSE @ EUROMICRO' 2002

Luis Iribarne¹, José María Troya² and Antonio Vallecillo²

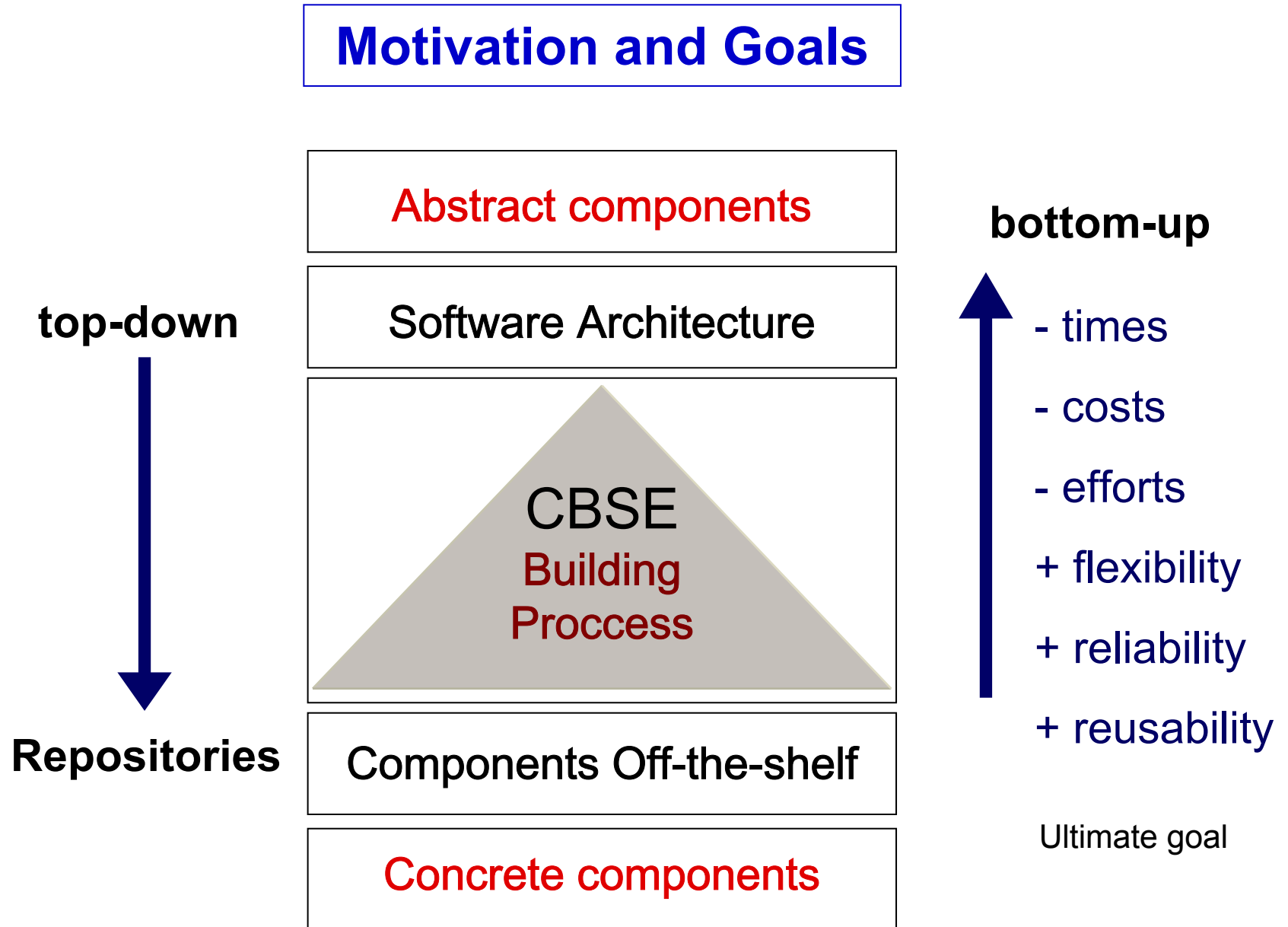
¹ Dpt. Lenguajes y Computación
University of Almería, Spain
liribarne@ual.es

² Dpt. Lenguajes y Ciencias de la Computación
University of Málaga, Spain
{troya,av}@lcc.uma.es

COMPONENT-BASED SOFTWARE ENGINEERING

Introduction

- ❑ **CBSE is moving organizations from applications development to applications assembly.**
- ❑ **The development effort now becomes one of gradual discovery about the components, their capabilities, their internal assumptions, and their incompatibilities when used in concert.**
- ❑ **Software component search and retrieval have now become crucial activities for building applications.**
- ❑ **Abstract specifications vs. Concrete specifications.**



Traditional Approaches

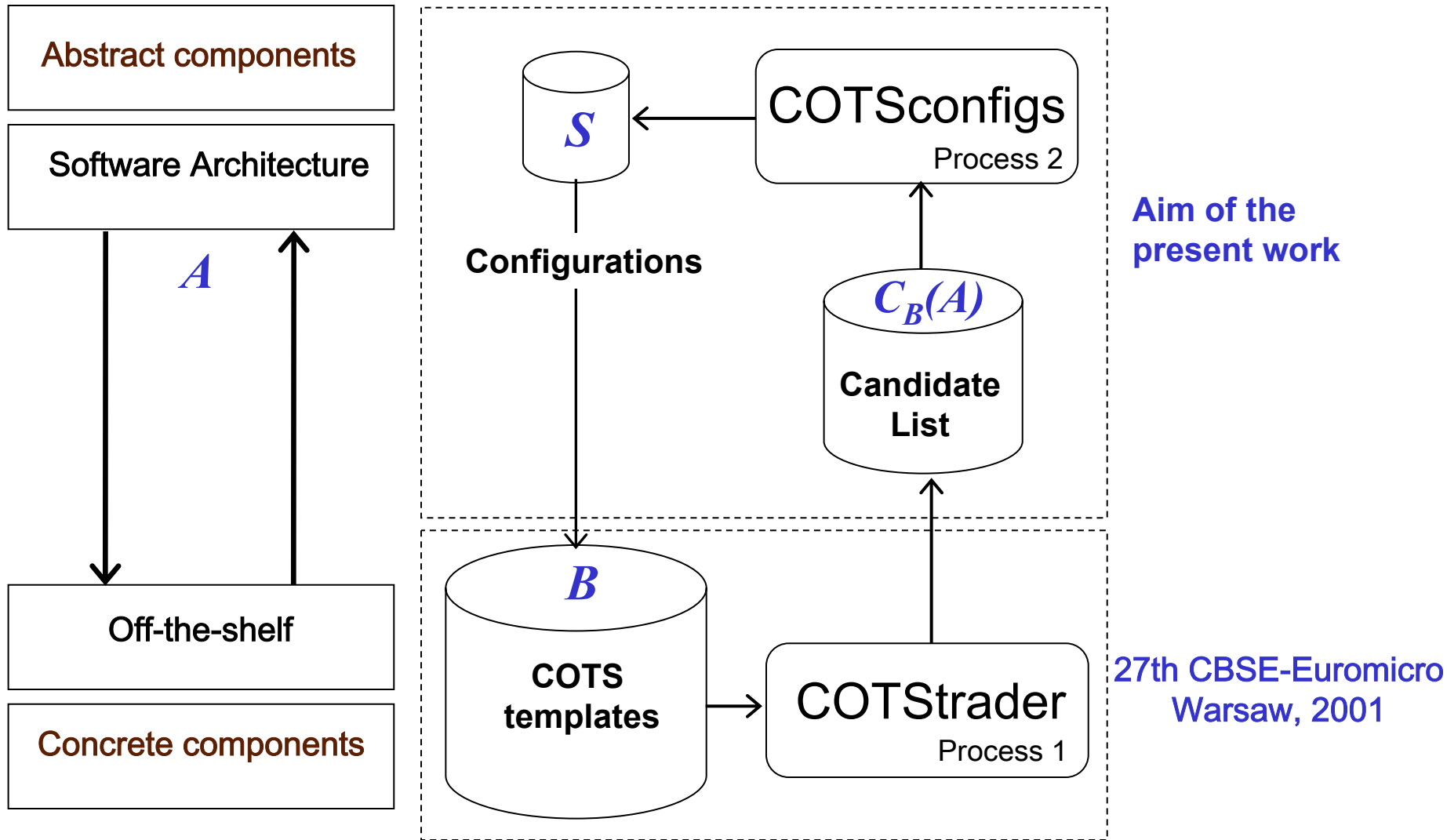
- ❑ Components with only one interface.**
- ❑ One-to-one selection processes.**
- ❑ Matching processes obtain non-required interfaces.**

COTS Approach

- ❑ Multiple provided and required interfaces.
- ❑ Multiple selection processes.
- ❑ Matching processes:
 - Gaps: services not covered.
 - Overlaps: overload of services.

Our approach tries to address these issues

Our Proposal Viewpoint



COTS Spec Info used by COTStrader

- ❑ COTStrader uses XML templates for documenting COTS.
- ❑ A COTS template includes four kinds of info:
 1. **Functional** (Prov&Req Interfaces+Semant.+Prot.).
 2. **Non-functional** (using properties).
 3. **Packaging** (for deployment info).
 4. **Marketing** (vendor info).

[COTS information used by COTStrader \(process 1\)](#)

COTS Spec Info used by COTSconfigs

- ❑ Only “Functional” part of a COTS template is used.**
- ❑ Functional part is described by means of interfaces.**
- ❑ A component has two kinds of interfaces:**
 - 1. Provided Interfaces**
 - 2. Required Interfaces**

COTS information used by COTSconfigs (process 2)

Component Interfaces

- $C = \{\mathcal{R}, \overline{\mathcal{R}}\}$
 - $\mathcal{R} = \{R_1, \dots, R_n\}$ the **supported** operations
 - $\overline{\mathcal{R}} = \{\overline{R}_1, \dots, \overline{R}_m\}$ the **required** operations

Component Composition

$$C_3 = C_1 | C_2 \left\{ \begin{array}{l} \mathcal{R}_3 = \begin{cases} \mathcal{R}_1 \cup \mathcal{R}_2 & \text{iff } \mathcal{R}_1 \cap \mathcal{R}_2 = \emptyset \\ \text{undefined} & \text{iff } \mathcal{R}_1 \cap \mathcal{R}_2 \neq \emptyset \end{cases} \\ \overline{\mathcal{R}}_3 = \begin{cases} \overline{\mathcal{R}}_1 \cup \overline{\mathcal{R}}_2 - \{\mathcal{R}_1 \cup \mathcal{R}_2\} & \text{iff } \mathcal{R}_1 \cap \mathcal{R}_2 = \emptyset \\ \text{undefined} & \text{iff } \mathcal{R}_1 \cap \mathcal{R}_2 \neq \emptyset \end{cases} \end{array} \right.$$

Interface Operators

- Inclusion: $\{R_1^1, \dots, R_1^s\} \subseteq \{R_2^1, \dots, R_2^t\}$
- Intersection: $\{R_1^1, \dots, R_1^s\} \cap \{R_2^1, \dots, R_2^t\}$
- Hiding: $C_1 - \{\mathcal{R}\} = \{\mathcal{R}_1 - \mathcal{R}, \overline{\mathcal{R}}_1\}$
- Replaceability: $C_1 \sqsubseteq C_2$ iff $(C_1.\mathcal{R}_1 \subseteq C_2.\mathcal{R}_2) \wedge (C_1.\overline{\mathcal{R}}_1 \supseteq C_2.\overline{\mathcal{R}}_2)$
- Equivalence: $C_1 \equiv C_2$ iff $C_1 \sqsubseteq C_2 \wedge C_2 \sqsubseteq C_1$

All operations defined on top of the traditional “ \sqsubseteq ” operator
(syntactic, protocol, or semantic substitutability)

Concepts

- **Candidates** (from a repository \mathcal{B}):

$$C_{\mathcal{B}}(A) = \{B \in \mathcal{B} \mid A.\mathcal{R} \cap B.\mathcal{R} \neq \emptyset\}$$

- **Configurations:**

- A set of candidate components S_1, \dots, S_n with no service gaps or service overlaps.

- **Closure:**

- A transitive closure of a configuration S wrt $C_{\mathcal{B}}(\mathcal{A})$

Building Apps with components

Software Architecture $\longrightarrow A = \{ A_1, A_2, \dots, A_n \}$

1) $A = A_1 \mid \dots \mid A_n$ and then $A = \{ A.\mathcal{R}, A.\overline{\mathcal{R}} \}$

2) **Candidates selection: $C_B(A)$**

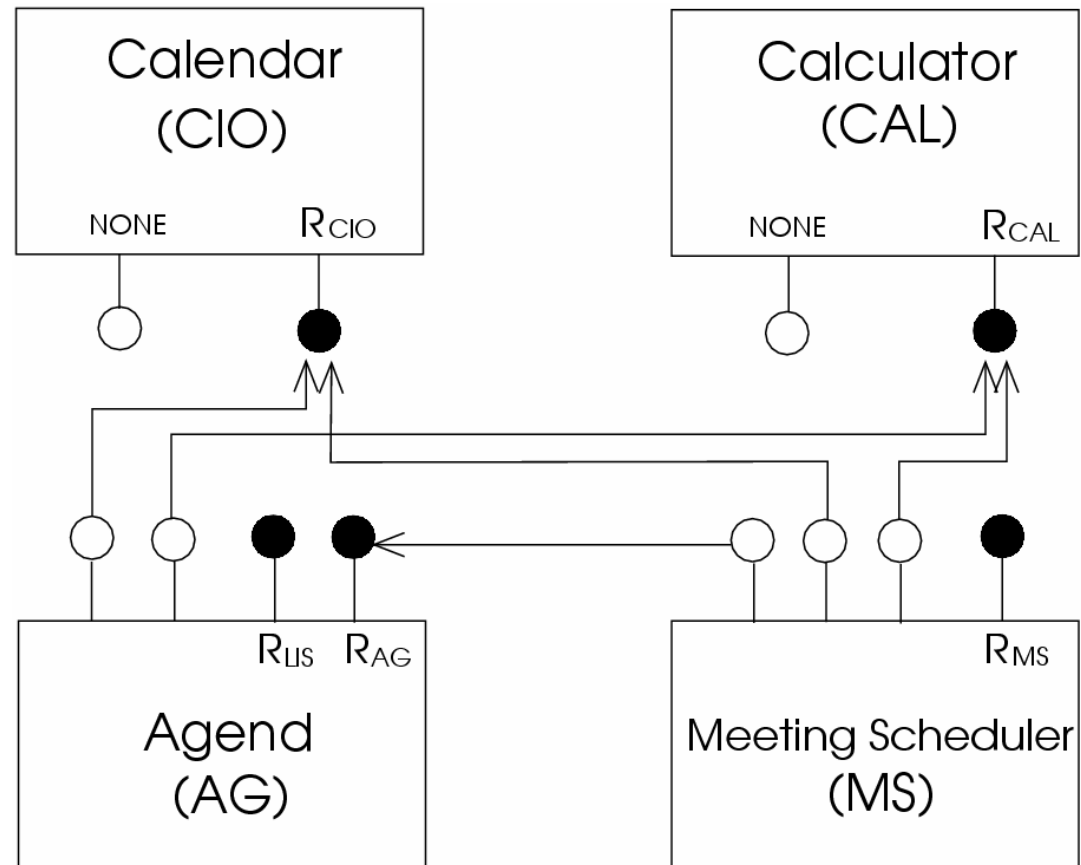
COTStrader
(process 1)

3) **Configurations generation: S**

COTSconfigs
(process 2)

4) **Hiding and Closing configurations**

A simple example



Software Architecture: $A = \{ \text{CAL, CIO, AG, MS} \}$

Abstract component specs vs. concrete candidate component specs

A Architecture
(Abstract specifications)

$CIO = \{R_{CIO}\}$
 $CAL = \{R_{CAL}\}$
 $AG = \{R_{AG}, R_{LIS}, \bar{R}_{CAL}, \bar{R}_{CIO}\}$
 $MS = \{R_{MS}, \bar{R}_{AG}, \bar{R}_{CAL}, \bar{R}_{CIO}\}$



Step 1: Composition

$A.R = \{R_{CIO}, R_{CAL}, R_{AG}, R_{LIS}, R_{MS}\} \quad A.\bar{R} = \{\}$

$C_B(A)$: Candidate components
(Concrete specifications)

$C_1 = \{R_{CIO}\}$
 $C_2 = \{R_{CAL}\}$
 $C_3 = \{R_{AG}, R_{CIO}, \bar{R}_{CAL}\}$
 $C_4 = \{R_{LIS}\}$
 $C_5 = \{R_{MS}, R_{AG}, \bar{R}_{CIO}\}$
 $C_6 = \{R_{CAL}, R_{LIS}, \bar{R}_P\}$

Step 2: Candidate selection

COTStrader

Step 3: Configurations generation (COTSconfigs)

```
1  function configs(i, Sol,  $\mathcal{S}$ )
2      //  $1 \leq i \leq \text{size}(C_{\mathcal{B}}(A))$  traverse the repository
3      // Sol is the configuration being built
4      //  $\mathcal{S}$  contains the set of all valid configurations .
5      if  $i \leq \text{size}(C_{\mathcal{B}}(A))$  then
6          for  $j := 1$  to  $\text{size}(C_i.\mathcal{R})$  do // all service in  $C_i$ 
7              // we try to include  $C_i.R_j$  service in Sol
8              if  $\{C_i.R_j\} \cap \text{Sol}.\mathcal{R} = \emptyset$  then //  $C_i.R_j \notin \text{Sol}.\mathcal{R}$ ?
9                   $\text{Sol} := \text{Sol} \cup \{C_i.R_j\}$ ;
10                 if  $A.\mathcal{R} \subseteq \text{Sol}.\mathcal{R}$  then // Is Sol a configuration?
11                      $\mathcal{S} := \mathcal{S} \cup \{\text{Sol}\}$ ; // if so, it is included in  $\mathcal{S}$ 
12                 else // but if there are still service gaps ...
13                     configs(i, Sol,  $\mathcal{S}$ ); // search in  $C_i$  ...
14                 endif
15                  $\text{Sol} := \text{Sol} - \{C_i.R_j\}$ ;
16             endif
17         endfor
18         configs(i + 1, Sol,  $\mathcal{S}$ ); // Next in  $C_{\mathcal{B}}(A)$ .
19     endif
20 endfunction
```

Results of the configs algorithm

	C ₁	C ₂	C ₃	C ₄	C ₅	C ₆	Configurations
-	<i>R_{CIO}</i>	<i>R_{CAL}</i>	<i>R_{AG}</i>	<i>R_{LIS}</i>	-	-	NONE: <i>R_{MS}</i> missing (gap)
1	<i>R_{CIO}</i>	<i>R_{CAL}</i>	<i>R_{AG}</i>	<i>R_{LIS}</i>	<i>R_{MS}</i>	-	C ₁ , C ₂ , C ₃ -{ <i>R_{CIO}</i> }, C ₄ , C ₅ -{ <i>R_{AG}</i> } RC
2	<i>R_{CIO}</i>	<i>R_{CAL}</i>	<i>R_{AG}</i>	-	<i>R_{MS}</i>	<i>R_{LIS}</i>	C ₁ , C ₂ , C ₃ -{ <i>R_{CIO}</i> }, C ₅ -{ <i>R_{AG}</i> }, C ₆ -{ <i>R_{CAL}</i> } R
3	<i>R_{CIO}</i>	<i>R_{CAL}</i>	-	<i>R_{LIS}</i>	<i>R_{MS}</i> , <i>R_{AG}</i>	-	C ₁ , C ₂ , C ₄ , C ₅ C
4	<i>R_{CIO}</i>	<i>R_{CAL}</i>	-	-	<i>R_{MS}</i> , <i>R_{AG}</i>	<i>R_{LIS}</i>	C ₁ , C ₂ , C ₅ , C ₆ -{ <i>R_{CAL}</i> } R
5	<i>R_{CIO}</i>	-	<i>R_{AG}</i>	<i>R_{LIS}</i>	<i>R_{MS}</i>	<i>R_{CAL}</i>	C ₁ , C ₃ -{ <i>R_{CIO}</i> }, C ₄ , C ₅ -{ <i>R_{AG}</i> }, C ₆ -{ <i>R_{LIS}</i> } R
6	<i>R_{CIO}</i>	-	<i>R_{AG}</i>	-	<i>R_{MS}</i>	<i>R_{CAL}</i> , <i>R_{LIS}</i>	C ₁ , C ₃ -{ <i>R_{CIO}</i> }, C ₅ -{ <i>R_{AG}</i> }, C ₆
-	<i>R_{CIO}</i>	-	<i>R_{AG}</i>	-	-	<i>R_{LIS}</i>	NONE: <i>R_{CAL}</i> , <i>R_{MS}</i> missing (gaps)
7	<i>R_{CIO}</i>	-	-	<i>R_{LIS}</i>	<i>R_{MS}</i> , <i>R_{AG}</i>	<i>R_{CAL}</i>	C ₁ , C ₄ , C ₅ , C ₆ -{ <i>R_{LIS}</i> }
8	<i>R_{CIO}</i>	-	-	-	<i>R_{MS}</i> , <i>R_{AG}</i>	<i>R_{CAL}</i> , <i>R_{LIS}</i>	C ₁ , C ₅ , C ₆
-	<i>R_{CIO}</i>	<i>R_{CAL}</i>	-	-	-	-	NONE: <i>R_{AG}</i> , <i>R_{LIS}</i> , <i>R_{MS}</i> missing (gaps)
9	-	<i>R_{CAL}</i>	<i>R_{AG}</i> , <i>R_{CIO}</i>	<i>R_{LIS}</i>	<i>R_{MS}</i>	-	C ₂ , C ₃ , C ₄ , C ₅ -{ <i>R_{AG}</i> } C
10	-	<i>R_{CAL}</i>	<i>R_{AG}</i> , <i>R_{CIO}</i>	-	<i>R_{MS}</i>	<i>R_{LIS}</i>	C ₂ , C ₃ , C ₅ -{ <i>R_{AG}</i> }, C ₆ -{ <i>R_{CAL}</i> }
-	-	<i>R_{CAL}</i>	<i>R_{AG}</i>	<i>R_{LIS}</i>	<i>R_{MS}</i>	-	NONE: <i>R_{CIO}</i> missing (gap)
11	-	<i>R_{CAL}</i>	<i>R_{CIO}</i>	<i>R_{LIS}</i>	<i>R_{MS}</i> , <i>R_{AG}</i>	-	C ₂ , C ₃ -{ <i>R_{AG}</i> }, C ₄ , C ₅ C
12	-	<i>R_{CAL}</i>	<i>R_{CIO}</i>	-	<i>R_{MS}</i> , <i>R_{AG}</i>	<i>R_{LIS}</i>	C ₂ , C ₃ -{ <i>R_{AG}</i> }, C ₅ , C ₆ -{ <i>R_{CAL}</i> }
-	-	<i>R_{CAL}</i>	-	-	<i>R_{MS}</i>	-	NONE: <i>R_{CIO}</i> , <i>R_{AG}</i> , <i>R_{LIS}</i> missing (gaps)
13	-	-	<i>R_{AG}</i> , <i>R_{CIO}</i>	<i>R_{LIS}</i>	<i>R_{MS}</i>	<i>R_{CAL}</i>	C ₃ , C ₄ , C ₅ -{ <i>R_{AG}</i> }, C ₆ -{ <i>R_{LIS}</i> }
14	-	-	<i>R_{AG}</i> , <i>R_{CIO}</i>	-	<i>R_{MS}</i>	<i>R_{CAL}</i> , <i>R_{LIS}</i>	C ₃ , C ₅ -{ <i>R_{AG}</i> }, C ₆
15	-	-	<i>R_{CIO}</i>	<i>R_{LIS}</i>	<i>R_{MS}</i> , <i>R_{AG}</i>	<i>R_{CAL}</i>	C ₃ -{ <i>R_{AG}</i> }, C ₄ , C ₅ , C ₆ -{ <i>R_{LIS}</i> }
16	-	-	<i>R_{CIO}</i>	-	<i>R_{MS}</i> , <i>R_{AG}</i>	<i>R_{CAL}</i> , <i>R_{LIS}</i>	C ₃ -{ <i>R_{AG}</i> }, C ₅ , C ₆
-	-	-	-	-	-	<i>R_{CAL}</i> , <i>R_{LIS}</i>	NONE: <i>R_{CIO}</i> , <i>R_{AG}</i> , <i>R_{MS}</i> missing (gaps)

Future Issues

- ❑ **Metrics and heuristics for configurations**
- ❑ **Branch-and-bound method**
- ❑ **Extend *COTSconfigs* to other info**

Selecting Software Components with Multiple Interfaces

CBSE @ EUROMICRO' 2002

<http://www.cotstrader.com>

Luis Iribarne Dpt. Lenguajes y Computación
University of Almería, Spain
liribarne@ual.es

José María Troya Dpt. Lenguajes y Ciencias de la Computación
University of Málaga, Spain
troya@lcc.uma.es

Antonio Vallecillo Dpt. Lenguajes y Ciencias de la Computación
University of Málaga, Spain
av@lcc.uma.es

Appendix A. COTS template example

```
<COTScomponent name="Calendar"  
xmlns="http://www.cotstrader.com/COTS-XMLSchema.xsd">  
  
  <functional>...</functional>  
  
  <properties>...</properties>  
  
  <packaging> ...</packaging>  
  
  <marketing> ... </marketing>  
  
</COTScomponent>
```

Appendix A. COTS template example

```
<COTScomponent name="Calendar">
  <functional>
    <providedInterfaces>
      <interface name="Event">
        <description notation="CORBA-IDL"> ... </description>
        <exactMatching href="http://..."> <softMatching href="http://...">
        <behavior notation="JML"> <!-- Java Larch -->
          <description href="http://...">
            <exactMatching href="http://..."> <softMatching href="http://...">
          </behavior>
        </interface>
      </providedInterfaces>
      <requiredInterfaces> ... </requiredInterfaces>
      <serviceAccessProtocol>
        <description notation="pi-protocols" href="http://... ">
        <exactMatching href="http://..."> <softMatching href="http://...">
      </serviceAccessProtocol>
    </functional>
    <properties notation="W3C">
      <property name="run"><type>xsd:bool</type><value href="http://..."></property>
    </properties>
    <packaging><description notation="CCM-softpkg" href="http://..."></packaging>
    <marketing>
      <license href="http://..."><expirydate>...</expirydate><vendor>...</vendor>...
    </marketing>
  </COTScomponent>
```