Concept Covering for Automated Building Blocks Selection based on Business Processes Semantics

Francesco di Cugno, Tommaso Di Noia,
Eugenio Di Sciascio, Azzurra Ragone
Politecnico di Bari,
via Re David 200,
70125 BARI, Italy
{f.dicugno, t.dinoia, disciascio, a.ragone}@poliba.it

Francesco M. Donini
Università della Tuscia,
via S. Carlo 32,
01100 VITERBO, Italy
donini@unitus.it

Abstract

In this paper we present a novel approach and a system for automated selection of building blocks, by exploiting business processes semantics.

The selection process is based on a novel greedy concept covering algorithm, which computes, given a description of the request, the set of needed building blocks. If such a set does not exist the algorithm returns the covered subset and explanation information on both the left uncovered part and conflicting part of the request description, exploiting non-standard inference services. The requested description can be expressed as conjunction of mandatory requirements and preferences. The approach has been deployed in a system specifically designed for SAP R/3 best practices reusability, which is fully functional and is currently being evaluated in an industrial setting.

1. Introduction

ERPs play at present a critical role in several enterprises, as they allow to improve in a substantial way the quality and efficiency of their business processes, both administration and production oriented. As a consequence more and more companies, even medium/small sized tend to move to ERPs to reengineer and integrate their processes. nevertheless it is well-known that the customization and adaptation process is usually much more expensive than the actual purchase of the software licences [16]. SAP R/3, for example, provides a huge number of parametric customizations in order to adapt the system to every particular organization context, and usually consultants, or consulting firms are hired to provide the needed expertise in such reengineering process. In SAP R/3 terms, the configuration process is called *Customizing* [1, 14]. The customizing process plays a fundamental role for the satisfaction of needs using a SAP R/3 solution. To reduce the analysis period various new methodologies were developed. Among others we mention: Global Accelerated SAP (GASAP), Accelerated SAP (ASAP) and Best Practices [14, 12]. These methodologies aim at reusing results obtained using the customized implementations. They provide the needed support to the SAP customizer, showing the implementation roadmap, with technical and organizational optimizations. Central to the best practices approach is the Building Block (BB) concept[12]. The basic idea is the modularization of a vertical solution, *i.e.*, in SAP terminology a complete SAP R/3 solution developed for a well defined organization scenario, identifying and extracting all its client independent information. BB contents in SAP Best Practices are defined considering from the start the possibility of their reuse from an implementation point of view. Basically, the BB content is defined by the identification of which Business Process parts can be reused within a predefined solution. The BB Library [15] provided by SAP in fact aims at sharing SAP knowledge within the community of developers. It is also possible to develop specific BBs able to provide particular solutions within a company context. Nevertheless, because of the rapid growth of the BBs number, choosing the correct BB in order to satisfy part of a specific Business Process, is increasingly expensive in terms of time, as the selection is driven only by the developers experience.

In this paper we show how technologies borrowed from the Semantic Web initiative can help both in modeling – using semantic annotation– building blocks descriptions and business processes, and in supporting consultants sharing their knowledge, by allowing automated selection of BBs from available business processes to compose new ones.

The remaining of the paper is structured as follows: next section provides the theoretical framework and algorithms used; then we present the system in section 3. Then we present the system and an illustrative example together with a numerical evaluation of system performances. Last Section draws the conclusions.

2. Semantic-Based Framework

The formal framework we adopt here borrows from Semantic Web (www.w3.org/2001/sw/) Languages and Description Logics (DLs)[2]. We assume the reader familiar with basics of both of them. Semantic annotation is usually adopted to execute standard inference services, namely subsumption and satisfiability, which return a boolean answer. Formally, such services can be defined as follows:

Concept Satisfiability: given an ontology $\mathcal{T}$ and a concept $\mathbf{C}$, does there exist at least one model of $\mathcal{T}$ assigning a non-empty extension to $\mathbf{C}$?
Subsumption: given an ontology $\mathcal{T}$ and two concepts $\mathbf{C}$ and $\mathbf{D}$, is C more general than D in any model of $\mathcal{T}$?

Given an ontology $\mathcal{T}$ modeling the investigated knowledge domain, a request description $\mathbf{C}$ and a resource description $\mathbf{D}$ *e.g.,* a building block, using subsumption it is possible to evaluate either (a)if the building block completely fulfills the request – $\mathcal{T} \models \mathbf{D} \sqsubseteq \mathbf{C}$, full match – or (2)if they are at least compatible with each other – $\mathcal{T} \models \mathbf{C} \sqcap \mathbf{D} \not\equiv \bot$, potential match – or (3)not – $\mathcal{T} \models \mathbf{C} \sqcap \mathbf{D} \equiv \bot$, partial match [9]. It is easy to see that in case of full match all the information specified on $\mathbf{C}$ is expressed, explicitly or by means of the ontology $\mathcal{T}$, also in $\mathbf{D}$. Nevertheless there are several cases when we are looking for approximate matches to a given request, which should be provided keeping into account the semantic annotations, as pointed out *e.g.,* in [11]. In our approach we use non-standard inference services to this purpose, whose rationale and definition are hereafter briefly recalled. Let us consider two concepts $\mathbf{C}$ and $\mathbf{D}$; if their conjunction $\mathbf{C} \sqcap \mathbf{D}$ is unsatisfiable in the TBox $\mathcal{T}$ representing the ontology, *i.e.,* they are not compatible with each other, we may want, as in a belief revision process, to retract requirements in $\mathbf{D}$, G (for *Give up*), to obtain a concept K (for *Keep*) such that $K \sqcap \mathbf{C}$ is satisfiable in $\mathcal{T}$.

Definition 1 *Let $\mathcal{L}$ be a DL, $\mathbf{C}$, $\mathbf{D}$, be two concepts in $\mathcal{L}$, and $\mathcal{T}$ be a set of axioms in $\mathcal{L}$, where both $\mathbf{C}$ and $\mathbf{D}$ are satisfiable in $\mathcal{T}$. A* Concept Contraction Problem *(CCP), identified by $\langle \mathcal{L}, \mathbf{D}, \mathbf{C}, \mathcal{T} \rangle$, is finding a pair of concepts $\langle G, K \rangle \in \mathcal{L} \times \mathcal{L}$ such that $\mathcal{T} \models \mathbf{D} \equiv G \sqcap K$, and $K \sqcap \mathbf{C}$ is satisfiable in $\mathcal{T}$. We call K a* contraction *of $\mathbf{D}$ according to $\mathbf{C}$ and $\mathcal{T}$.*

$\mathcal{Q}$ is a symbol for a CCP, and $SOLCCP(\mathcal{Q})$ denotes the set of all solutions to $\mathcal{Q}$. Obviously, there is always the trivial solution $\langle G, K \rangle = \langle \mathbf{D}, \top \rangle$ to whatever CCP $\mathcal{Q}$, that is give up everything of $\mathbf{D}$. When $\mathbf{C} \sqcap \mathbf{D}$ is satisfiable in $\mathcal{T}$, the "best" possible solution is $\langle \top, \mathbf{D} \rangle$, that is, give up nothing — if possible. Hence, a *Concept Contraction* problem is an extension of a satisfiability one. Since usually one wants to give up as few things as possible, some minimality in the contraction must be defined [10]. In most cases a pure logic-based approach could be not sufficient to decide between which beliefs to give up and which to keep. There is the need of modeling and defining some extra-logical information. One approach is to give up minimal information [3]. Another one considers some information more important than other and the information that should be retracted are the least important ones [7]. When subsumption does not hold *i.e.,* a full match is unavailable, one may want to hypothesize some explanation on which are the causes of this result. In [8] the Concept Abduction Problem (CAP) was introduced and defined as a non standard inference problem for DLs to provide a logic-based answer to such question.

Definition 2 *Let $\mathbf{C}$, $\mathbf{D}$, be two concepts in a Description Logic $\mathcal{L}$, and $\mathcal{T}$ be a set of axioms, where both $\mathbf{C}$ and $\mathbf{D}$ are satisfiable in $\mathcal{T}$. A* Concept Abduction Problem *(CAP), denoted as $\langle \mathcal{L}, C, D, \mathcal{T} \rangle$, is finding a concept H such that $\mathcal{T} \not\models \mathbf{C} \sqcap H \equiv \bot$, and $\mathcal{T} \models \mathbf{C} \sqcap H \sqsubseteq \mathbf{D}$.*

$\mathcal{P}$ is a symbol for a CAP, and $SOL(\mathcal{P})$ denotes the set of all solutions to $\mathcal{P}$.
In [8] also a minimality criteria for H and a polynomial algorithm to find solutions which are irreducible, for an $\mathcal{ALN}$ DL, have been proposed.

Given a CAP $\mathcal{P}$, if H is a conjunction of concepts and no sub-conjunction of concepts in H is a solution to $\mathcal{P}$, then H is an irreducible solution. The *rankPotential* algorithm [9] allows to numerically compute the *length* of H. Recalling the definition, *rankPotential* $(\mathbf{C}, \mathbf{D})$ corresponds to a numerical measure of what is still missing in $\mathbf{C}$ w.r.t. $\mathbf{D}$. If $\mathbf{C} = \top$ we have the maximum value for *rankPotential* $(\mathbf{C}, \mathbf{D})$, that is the maximum (potential) mismatch of $\mathbf{C}$ from $\mathbf{D}$. The value returned by *rankPotential*$(\top, \mathbf{D})$ hence amounts to how specific is a complex concept expression $\mathbf{D}$ with respect to an ontology $\mathcal{T}$, what we call the *depth* of $\mathbf{D}$: $\mathtt{depth}(\mathbf{D})$. Such a measure is not trivially the depth of a node in a tree for at least two main reasons:
1. An ontology, typically, is not a simple terms taxonomy tree, *i.e.,* its structure is not limited to simple IS-A relations between two atomic concepts.
2. A complex concept is generally the conjunction of both atomic concepts and role expressions.

The solution to a CAP $\mathcal{P}$can be interpreted as what has to be hypothesized in $\mathbf{C}$, and in a second step added to, to make $\mathbf{C}$ more specific than $\mathbf{D}$, which would make subsumption result true. So, as Concept Contraction extends satisfiability, Concept Abduction extends subsumption.Previously described services can be used to infer information when we are matching a request with a given description to find a suitable one. Yet, given a request we may have to fulfill such a request by selecting several building blocks. To reach such an objective using semantics, we have to make one step beyond.

Concept Covering, originally defined in [13] for a particular set of DLs with limited expressiveness, was later extended and generalized in terms of Concept Abduction in [4]. We recall here this definition:

Definition 3 *Let **D** be a concept, $\mathcal{R} = \{S_1, S_2, ..., S_k\}$ be a set of concepts in a Description Logic $\mathcal{L}$and $\mathcal{T}$ be a set of axioms, where **D** and $S_i, i = 1..k$ are satisfiable in $\mathcal{T}$.*

1. *A* Concept Covering Problem *(CCoP), denoted as* CCoP$(\langle \mathcal{L}, \mathcal{R}, D, \mathcal{T} \rangle)$*, is finding, if it exists, a set $\mathcal{R}_c \subseteq \mathcal{R}$, such that both for each $S_j \in \mathcal{R}_c$, $\mathcal{T} \not\models \sqcap S_j \equiv \bot$, and $H \in SOL(\langle \mathcal{L}, \sqcap S_j, \mathbf{D}, \mathcal{T} \rangle)$ is such that $H \not\sqsubseteq \mathbf{D}$.*
2. *We call $\langle \mathcal{R}_c, H \rangle$ a* solution *for the CCoP $\langle \mathcal{L}, \mathcal{R}, D, \mathcal{T} \rangle$.*

In [4] the algorithm $GREEDYsolveCCoP$ was also proposed, to perform a greedy concept covering using concept abduction. Intuitively, $\mathcal{R}_c$ is a set of concepts that completely or partially cover **D** w.r.t. $\mathcal{T}$, while the abduced concept H represents what is still in **D** and is not covered by $\mathcal{R}_c$. It is worth noticing that a greedy approach is needed for performance reasons as also the basic set covering problem is NP-Hard, but also that a Concept Covering Problem is similar, but has remarkable differences when compared to classical set covering. In fact CCoP is not trivially a different formulation of a classical minimal set covering (*SC*) problem, as an exact solution to a CCoP may not exist. Furthermore in *SC* elements are not related with each other, while in CCoP elements are related with each other through axioms within the ontology, and while the aim of an *SC* is to minimize the cardinality of $\mathcal{R}_c$ the aim of a CCoP is to maximize the covering, hence minimizing H. There can be several solutions for a single CCoP, depending also on the strategy adopted for choosing concepts in $\mathcal{R}_c$. In the basic implementation of our approach [6] we adopted as Concept Covering algorithm the $GREEDYsolveCCoP$. Examining first results of tests on users experiencing our system, we noticed that, when an exact covering is unavailable, users would anyway appreciate approximate covers, also with requirements in conflict –as it eases anyway the customizing process– as long as the system can clearly illustrate conflicting requirements. Hence we implemented an enhanced version of the Concept Covering algorithm, which is better able to take into account such needs.

The algorithm *BBComposer*$(\mathcal{R}, BP, \mathcal{T})$ –where BP is the required Business Process and $\mathcal{R} = \{BB_i\}, i = 1...n$ is a set of Building Blocks, all described w.r.t. a TBox $\mathcal{T}$– takes also concept contraction into account during the process of selection of building blocks.

```
Input: $BP$, $\mathcal{R}$, where $BP$ and all $BB_i \in \mathcal{R}$ are
       satisfiable in $\mathcal{T}$
Output: $\mathcal{R}_c, BP_{uncovered}, K_{BP}, G_{contraction}$
1  $BP_u = BP$;
2  $H_{min} = \top$;
3  $\mathcal{R}_c = \emptyset$;
4  repeat
5      $BB_{max} = \top$;
6      $d_{min} = \infty$;
7      foreach $BB_i \in \mathcal{R}$ such that $\mathcal{R}_c \cup \{BB_i\}$ covers
       $BP_u$ contraction according to
       $\mathcal{Q} = \langle \mathcal{L}, \sqcap_{BB_k \in \mathcal{R}_c} BB_k \sqcap BB_i, BP_u, \mathcal{T} \rangle$ do
8          if $\mathcal{T} \models BP_u \sqcap BB_i \equiv \bot$ then
9              $\langle G, K \rangle = contract(BB_i, BP_u, \mathcal{T})$;
10             $H = abduce(BB_i, K, \mathcal{T})$;
11         else
12             $H = abduce(BB_i, BP_u, \mathcal{T})$;
13             $K = BP_u$;
14             $G = \top$;
15         end
16         $d = \Psi(G, K, H, BB_i, BP_u)$;
17         if $d < d_{min}$ then
18             $BB_{max} = BB_i$;
19             $H_{min} = H$;
20             $d_{min} = d$;
21         end
22     end
23     if $(BB_{max} \not\equiv \top)$ then
24         $\mathcal{R} = \mathcal{R} \backslash \{BB_i\}$;
25         $\mathcal{R}_c = \mathcal{R}_c \cup \{BB_i\}$;
26         $BP_u = H_{min}$;
27     end
28 until $(BB_{max} \not\equiv \top)$;
29 $\langle K_{BP}, G_{fin} \rangle = contract(\sqcap_{BB_k \in \mathcal{R}_c} BB_k, BP, \mathcal{T})$;
30 $BP_{uncovered} = abduce(\sqcap_{BB_k \in \mathcal{R}_c} BB_k, K_{BP}, \mathcal{T})$;
31 return $(\mathcal{R}_c, BP_{uncovered}, K_{BP}, G_{fin})$;
```

Algorithm 1: *BBComposer*$(\mathcal{R}, BP, \mathcal{T})$ – Extended Concept Covering

The algorithm tries to cover BP description "as much as possible", using the concepts $BB_i \in \mathcal{R}$. If a new building block BB_i can be added to the already composed set $\mathcal{R}_c$, *i.e.*, $\mathcal{T} \not\models (\sqcap_{BB_k \in \mathcal{R}_c} BB_k) \sqcap BB_i \equiv \bot$, then an extended matchmaking process is performed (rows 8–22). If BB_i is not consistent with the uncovered part of the business process $BP_{uncovered}$, the latter is contracted and subsequently an abduction process is performed between the contracted uncovered part and BB_i (rows 8–10). If BB_i is consistent with $BP_{uncovered}$, only a concept abduction problem is solved (rows 11–15). Based on the previously computed concepts G, K and H a global score is computed as a metric to evaluate how good BP_i is with respect to the covering set (rows 16–21). The score is determined through

Ψ function. It takes into account what has to be given up (G), kept (K) in the uncovered part (BP_u) of BP in order to be compatible BB_i and what has to be hypothesized (H) in in order to fully satisfy the contracted BP_u, *i.e.,* K. For $\mathcal{ALN}$ we originally proposed in [5] the following closed form.

$$\Psi(G, K, H, BB_i, BP_u) = \left| 1 - \frac{N}{N-g} * (1 - \frac{h}{k}) \right| \quad (1)$$

where N, k, g, h represent numerical evaluation, computed via the $rankPotential$ algorithm[9], for respectively:

- BP_u the uncovered part of BP in an intermediate step of the covering process – $N = rankPotential(\top, BP_u, \mathcal{T})$.
- K: belonging to a solution of $\mathcal{Q} = \langle \mathcal{ALN}, BB_i, BP_u, \mathcal{T} \rangle$ – $k = rankPotential(\top, K, \mathcal{T})$.
- G: belonging to a solution of $\mathcal{Q} = \langle \mathcal{ALN}, BB_i, BP_u, \mathcal{T} \rangle$ – $g = rankPotential(BP_u, K, \mathcal{T})$.
- H: a solution of $\mathcal{P} = \langle \mathcal{ALN}, BB_i, K, \mathcal{T} \rangle$ – $h = rankPotential(BB_i, K, \mathcal{T})$.

At a first glance the right formulation for g, would seem to be $g = rankPotential(\top, G, \mathcal{T})$, as done for K. Using a trivial example we will show that the former formulation in more appropriate than the latter. Imagine a simple $\mathcal{T} = \{\mathsf{C} \sqsubseteq \mathsf{B} \sqsubseteq \mathsf{A}; disj(\mathsf{E}, \mathsf{C})\}$ with $BB_i = \mathsf{E}$ and $BP_u = \mathsf{C} \sqcap \mathsf{F}$.

For $\mathcal{Q} = \langle \mathcal{ALN}, BB_i, BP_u, \mathcal{T} \rangle$ we have $\langle G, K \rangle = \langle \mathsf{C}, \mathsf{B} \sqcap \mathsf{F} \rangle \in SOLCCP(\mathcal{Q})$.

Now, $rankPotential(\top, \mathsf{C}, \mathcal{T}) = 3$ while $rankPotential(\mathsf{C} \sqcap \mathsf{F}, \mathsf{B} \sqcap \mathsf{F}, \mathcal{T}) = 1$[1]. It is easy to see that the latter result better takes into account how much BP_u has been contracted in order to be compatible with BB_i.

The rationale of the closed form expression (1) is given in [5]. By choosing the candidate descriptions minimizing Ψ, the algorithm takes into account both g and h, *i.e.,* a numerical measure of how much it has to be given up in the uncovered request BP_u and how much to hypothesize in the building blocks analyzed at each stage. Notice that we do not need to change the definition of Concept Covering Problem provided in [4]: the problem we solve is still the one detailed in Definition 3 if we consider $\mathsf{D} = K_{BP}$. The distinguishing element between the two solving algorithms $GREEDYsolveCCoP$ and *BBComposer* is in the choice criterion among the BB_i greedy selected to compose the set. In $GREEDYsolveCCoP$ such choice was made on the basis of a minimality criterion on H, the solution of an abduction problem on $BP_{uncovered}$– the part of the needed process yet to cover at each stage of the algorithm– and the selected building block BB_i. In *BBComposer* the choice criteria is the minimization of the function Ψ, which takes into account also a measure of how much the request should be contracted before the selection, thus improving the selection process. The outputs of *BBComposer* are:

- $\mathcal{R}_c$: the set of BBs selected to compose the business process
- $BP_{uncovered}$: the part of the business process description that has –in case no exact cover exists– not been covered
- K_{BP}: the contracted BP
- G_{fin}: the part of the business process description given-up at the end of the whole composition process

The enhanced covering algorithm previously presented takes into account also the contraction of the request during the search, thus providing approximated results also in the presence of conflicting requirements. Nevertheless there can be cases when users are not willing to give up from the begininning to characteristics they consider of paramount importance in their request. To satisfy such a reasonable need we offer users the possibility to explicitly express mandatory requirements $\mathcal{MND}$ and preferences $\mathcal{PRF}$ within their request as $BP \equiv \mathcal{MND} \sqcap \mathcal{PRF}$.

Based on such specification we pre-select the building blocks to be selected if and only if their description is compatible with $\mathcal{MND}$. The pre-selection process is based on the following simple check: Where $\mathcal{R}_{\mathcal{MND}}$ is the set of

```
1 if ($\mathcal{T} \not\models \mathcal{MND} \sqcap BB_i \equiv \bot$) then
2     $\mathcal{R}_{\mathcal{MND}} = \mathcal{R}_{\mathcal{MND}} \cup \{BB_i\}$;
3 end
```

BB_i which are compatible with the mandatory requirements $\mathcal{MND}$of the user. Considering $\mathcal{MND}$ and $\mathcal{PRF}$ as conjunctions of $\mathcal{ALN}$ concepts, it is obviously possible to check users specification coherence, using non-standard inference services. This incoherency can be detected by checking if there exists an overlap between $\mathcal{MND}$ and $\mathcal{PRF}$. Using Concept Abduction [8] or its numerical version [9] it is possible to compute how dissimilar $\mathcal{PRF}$ is from $\mathcal{MND}$ as we will show in the following. Obviously, if $\mathcal{MND}$ and $\mathcal{PRF}$ overlap, the user can be asked to reconsider her requirements.

3. Proposed System

The framework and system we built is able to select the BBs to be installed with respect to a given business process and it has been designed with the objective of allowing a developers community to share the knowledge they

1 Roughly speaking, in this example, $\mathcal{T}$can be seen as a tree and $rankPotential(C, D, \mathcal{T})$ measuring the hops to go from D to C.

acquire, together with the one provided by SAP documentation. The approach is based on an architecture, which is sketched in Fig 1, comprising:

- a **Knowledge Base Repository** composed by an **Ontology Repository** (OR) and **Instance Repository** (IR). In OR all the ontologies shared within the system are stored in OWL files – we use different ontologies interacting with each other via the *owl:imports* TAG. IR contains all the concept descriptions representing both BBs capability descriptions and Business Processes. IR is a DBMS-based repository for efficiency reasons. IR and OR can be obviously stored on different machines. The idea behind this decentralized structure is based on the observation that different consultants and companies may refer to the same ontologies in order to describe their own BBs or Business Processes. Hence, the companies share the ontologies on a single OR but there are different IRs, one for each company.
- a **Client GUI**: a user friendly interface allowing the user to interact with the system in order to query, tell new knowledge, examine the available modeled knowledge. The client allows to:

  a) browse the ontology;

  b) describe a new BB and store its ontology-based description within the Instance Repository, see Fig. 2;

  c) query the system and compose the request using the knowledge modeled with the ontology, see Fig. 3.
- a **Reasoner** – MAMAS-tng, used to perform inferences based on formal semantics of the language used to model both the ontologies and the individuals. MAMAS-tng is a DL reasoning engine endowed both of standard reasoning services and non-standard inferences. It is a multi-user, multi-ontology Java servlet based system and exposes an enhanced DIG 1.1 interface. It is available as an HTTP service at `http://dee227.poliba.it:8080/MAMAS-tng/DIG`.

A common terminology can be shared by the community of SAP consultants within OR, the Ontology Repository. We modeled all the ontologies used within the system using the $\mathcal{ALN}$ subset of OWL DL, thus allowing to exploit Concept Abduction and Concept Contraction service via MAMAS-tng and perform a Concept Covering of the user requests. The main competency questions for the ontologies we developed are basically related to BB functionalities and Business Process activities. Some of them are:

- Which are the activities a Business Process must perform?
- Which scenario better approximate a given Business Process and which are the BBs used to implement it?

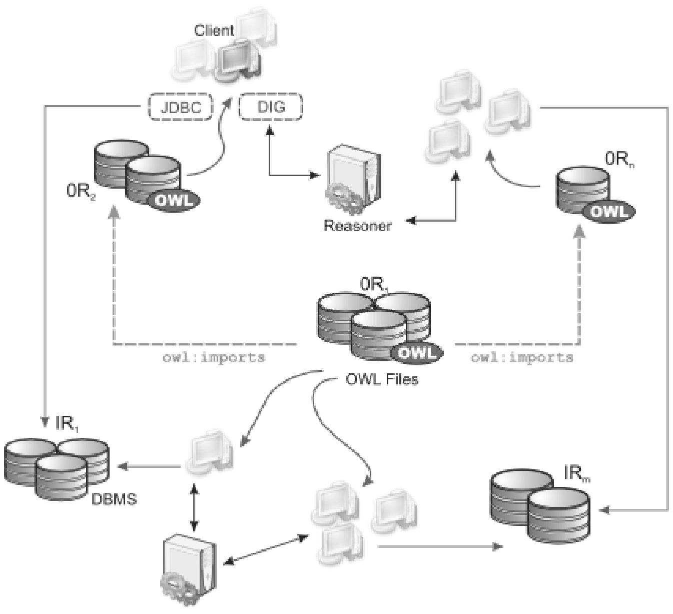
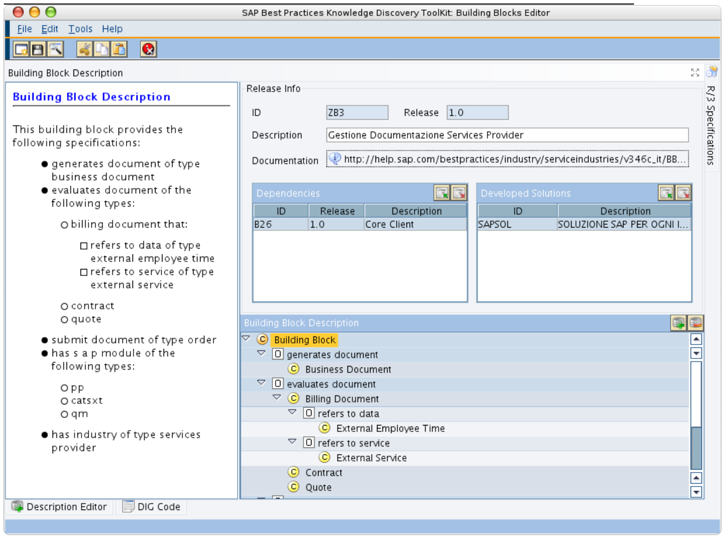


Figure 1. System Architecture


SAP Best Practices Knowledge Discovery ToolKit: Building Blocks Editor
File Edit Tools Help
Building Block Description
R/3 Specifications
Description Editor
DIG Code

Building Block Description

This building block provides the following specifications:

- generates document of type business document
- evaluates document of the following types:
  - billing document that:
    - refers to data of type external employee time
    - refers to service of type external service
  - contract
  - quote
- submit document of type order
- has s a p module of the following types:
  - pp
  - catsxt
  - qm
- has industry of type services provider

Release Info

ID: ZB3 Release: 1.0

Description: Gestione Documentazione Services Provider

Documentation: http://help.sap.com/bestpractices/industry/serviceindustries/v346c_it/BB...

Dependencies

ID	Release	Description
B26	1.0	Core Client

Developed Solutions

ID	Description
SAPSOL	SOLUZIONE SAP PER OGNI I...

Building Block Description

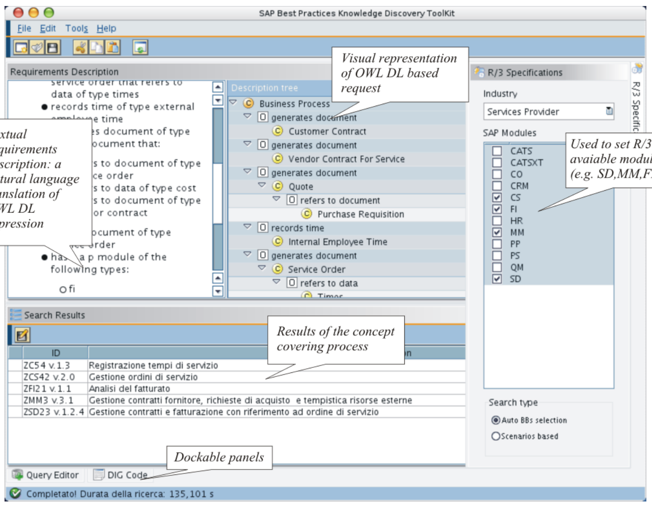
```
Building Block
  generates document
    Business Document
  evaluates document
    Billing Document
      refers to data
        External Employee Time
      refers to service
        External Service
    Contract
    Quote
```

Figure 2. A snapshot of a Building Block description

- Which are the BBs needed in order to manage a particular Business Process?

In order to satisfy the needed prerequisites, a BP description will contain the list of the activities to be performed by such process. Notice that in a BP, from the application point of view, only activities requiring particular functionalities from R/3 are relevant and then modeled. As an example, let us suppose we want to compose BBs in order to model the Business Process described in the following:

> *The process creates a service contract, which would reflect the terms of service agreed upon with the customer. A vendor contract is set up to define the*


SAP Best Practices Knowledge Discovery ToolKit
File Edit Tools Help
Requirements Description
Description tree
R/3 Specifications
Industry
Services Provider
SAP Modules
Search type
Auto BBs selection
Scenarios based
Search Results
Query Editor
DIG Code
Visual representation of OWL DL based request
Textual requirements description: a natural language translation of OWL DL expression
Used to set R/3 avaiable modules (e.g. SD,MM,FI,..)
Results of the concept covering process
Dockable panels

ID	
ZC54 v.1.3	Registrazione tempi di servizio
ZCS42 v.2.0	Gestione ordini di servizio
ZFI21 v.1.1	Analisi del fatturato
ZMM3 v.3.1	Gestione contratti fornitore, richieste di acquisto e tempistica risorse esterne
ZSD23 v.1.2.4	Gestione contratti e fatturazione con riferimento ad ordine di servizio

Completato! Durata della ricerca: 135,101 s

Figure 3. The GUI used to compose the requested Business Process

agreement for external services. The service order is then used for planning and executing the work to be performed by both internal and external resources. A purchase requisition is automatically generated from the service order for the external service requirement and a purchase order is created from it. The Cross-Application Time Sheet (CATS) is used for recording employee hours and the service entry sheet captures contracted labor. The costs associated with internal work are transferred to the service order. Resource-related billing, results analysis, and profitability analysis can be performed on a periodic basis or at the end of the project.

Now suppose that such BP belongs to professional service and that transactions needed in order to complete all the activities belong to FI, MM, CATS, SD and CS modules provided by R/3 [2]. Using DL syntax, with respect to the reference ontology, we model the previous BP as shown in Figure 4.

The description has to be easily understandable even by an inexperienced user. This is very important in order to increase the friendliness during the query formulation. To this aim they are graphically represented using a tree structure where `BusinessProcess` is the root node and its children represent the BP activities (see Fig. 3). The GUI also simply allows the user to communicate with the KB and use the knowledge modeled and stored within it. In an equivalent way, a BB is described with respect to the functionalities it provides, representing the set of activities which can be performed using the BB itself with respect to a BP. Fig. 2 shows a snapshot of the interface for BBs description.

Demand:

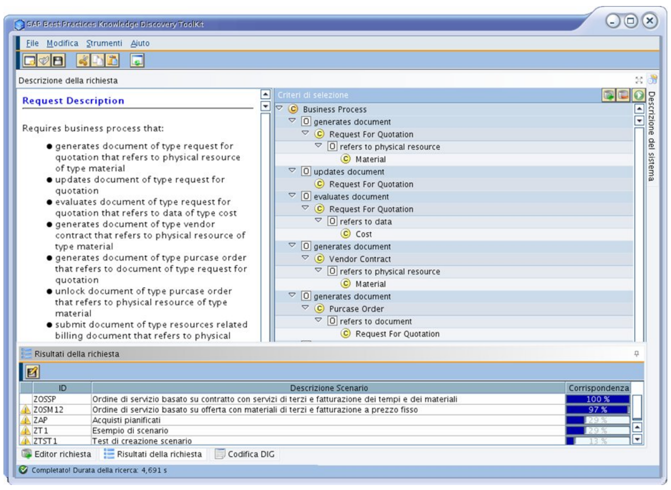
```
∀ activity:generatesDocument
    resources:CustomerContract
    ⊓ resources:VendorContractForService
    ⊓ resources:ServiceOrder
    ⊓ resources:Quote

⊓ ∀ resources:refersToDocument
    resources:PurchaseRequisition
    ⊓ resources:ServiceOrder
    ⊓ ∀ resources:refersToData
        resources:Times

⊓ ∀ activity:recordsTime
    resources:ExternalEmployeeTime
    ⊓ resources:InternalEmployeeTime

⊓ ∀ activity:generatesDocument
    resources:BillingDocument
    ⊓ ∀ resources:refersToDocument
        resources:ServiceOrder
        ⊓ resources:VendorContract

    ⊓ ∀ resources:refersToData
        resources:Cost

⊓ ∀ activity:analyzesDocument
    resources:BillingDocument

⊓ ∀ hasIndustry
    bp:ProfessionalServices

⊓ ∀ hasSAPModule
    bp:FI
    ⊓ bp:SD
    ⊓ bp:CATS
    ⊓ bp:CS
    ⊓ bp:MM

⊓ ∃ hasIndustry
```

Figure 4. DL formalization of the BP request

4. Illustrative Example

Given a BP description, our toolkit allows to perform either a semantic search based on already known business scenarios or a fully automated search of a set of BBs that can implement the BP described in R/3, exploiting algorithms and inferences previously described. Fig. 5 shows

2 For conciseness we straightforwardly adopt SAP naming conventions [14]



SAP Best Practices Knowledge Discovery ToolKit

File Modifica Strumenti Aiuto

Descrizione della richiesta

Request Description

Requires business process that:

- generates document of type request for quotation that refers to physical resource of type material
- updates document of type request for quotation
- evaluates document of type request for quotation that refers to data of type cost
- generates document of type vendor contract that refers to physical resource of type material
- generates document of type purcase order that refers to document of type request for quotation
- unlock document of type purcase order that refers to physical resource of type material
- submit document of type resources related billing document that refers to physical

Criteri di selezione

```
Business Process
  generates document
    Request For Quotation
      refers to physical resource
        Material
  updates document
    Request For Quotation
  evaluates document
    Request For Quotation
      refers to data
        Cost
  generates document
    Vendor Contract
      refers to physical resource
        Material
  generates document
    Purcase Order
      refers to document
        Request For Quotation
```

Descrizione del sistema

Risultati della richiesta

ID	Descrizione Scenario	Corrispondenza
ZOSSP	Ordine di servizio basato su contratto con servizi di terzi e fatturazione dei tempi e dei materiali	100 %
ZOSM12	Ordine di servizio basato su offerta con materiali di terzi e fatturazione a prezzo fisso	97 %
ZAP	Acquisti pianificati	29 %
ZT1	Esempio di scenario	29 %
ZTST1	Test di creazione scenario	13 %

Editor richiesta | Risultati della richiesta | Codifica DIG

Completato! Durata della ricerca: 4,691 s

Figure 5. Results for a Basic Search

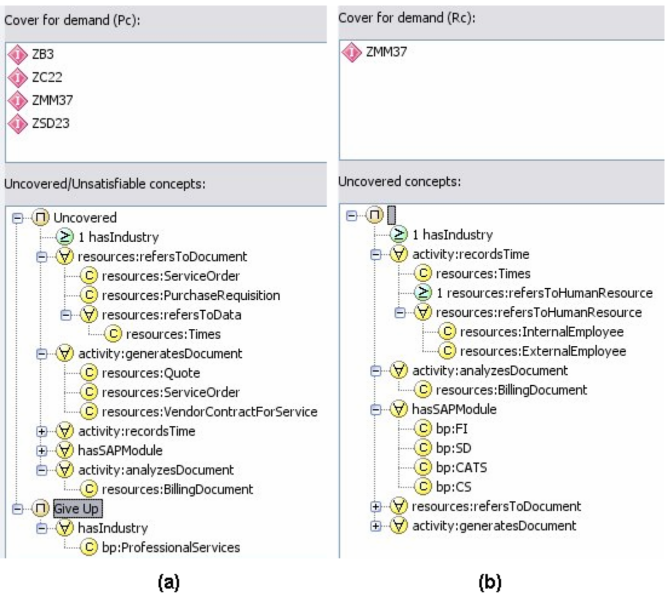
results for a simple semantic search of a business scenario.

In order to find a set of BBs whose composition satisfies the requested BP, their description are retrieved from the KB/repository where they are stored. Notice that, since a BB description represents the functionalities it provides w.r.t. a BP, the structure of such description is obviously similar to a BP request one. Given an ontology $\mathcal{T}$, a set of BB descriptions in a repository, $\mathcal{R} = \{BB_1, BB_2, ..., BB_n\}$, and a Business Process request BP_d, where $\mathcal{T} \not\models BB_i \equiv \bot$ and $\mathcal{T} \not\models BP_d \equiv \bot$ – both each $BB_i \in \mathcal{R}$ and BP_d are consistent w.r.t. to an ontology $\mathcal{T}$ – we compute a Concept Covering of BP_d w.r.t. $\mathcal{R}$. In other words, given a set of BB descriptions and a BP request BP_d, we find a subset of the available BBs, such that their conjunction is a cover of BP_d or its contracted version, that is they are able to perform the task required within (contracted) BP_d. As an example, suppose to have four BBs: ZB3, ZC22, ZSD23, ZMM37 [3] whose description is modeled with respect to the reference ontology and a BP_d description as depicted in Figure 4. Running the BBs selection procedure we obtain that:

- if the user is willing to contract her request on $\forall$`hasIndustry.ProfessionalService` (see Figure 6(a)) we obtain a covering solution involving all the available BBs.
- if the user set as $\mathcal{MND}$ the characteristic $\forall$`hasIndustry.ProfessionalService`, then we obtain a solution where only ZMM37 results to be compatible with $\mathcal{MND}$ (see Figure 6(b)).

3 again we adopt R/3 naming conventions

In the former case we have the computed solution, covering BP_d for a 22% more than in the latter case. As expected, relaxing some constraints on BP_d, we can compute a better solution from a covering point of view.



(a)

Cover for demand (Pc):

ZB3
ZC22
ZMM37
ZSD23

Uncovered/Unsatisfiable concepts:

```
⊓ Uncovered
  ≥ 1 hasIndustry
  ∀ resources:refersToDocument
    resources:ServiceOrder
    resources:PurchaseRequisition
    ∀ resources:refersToData
      resources:Times
  ∀ activity:generatesDocument
    resources:Quote
    resources:ServiceOrder
    resources:VendorContractForService
  ∀ activity:recordsTime
  ∀ hasSAPModule
  ∀ activity:analyzesDocument
    resources:BillingDocument
⊓ Give Up
  ∀ hasIndustry
    bp:ProfessionalServices
```

(b)

Cover for demand (Rc):

ZMM37

Uncovered concepts:

```
⊓
  ≥ 1 hasIndustry
  ∀ activity:recordsTime
    resources:Times
    ≥ 1 resources:refersToHumanResource
    ∀ resources:refersToHumanResource
      resources:InternalEmployee
      resources:ExternalEmployee
  ∀ activity:analyzesDocument
    resources:BillingDocument
  ∀ hasSAPModule
    bp:FI
    bp:SD
    bp:CATS
    bp:CS
  ∀ resources:refersToDocument
  ∀ activity:generatesDocument
```

Figure 6. Building Blocks Selection

Quantitative experiments have been carried out on the system, to test performances and scalability, thus numerically validating theoretical complexity results, and practical feasibility of our approach. We generated a set of 5000 descriptions, where $\mathtt{depth}(BB)$ of the descriptions follows a Gauss distribution with $\mu = 10$ and $\sigma = 20$.

Experiments were carring out using requests having depth 5, 11, 15, 24, and sets between 10 and 500 descriptions, all randomly generated for each iteration. In Fig. 7 experimental results are presented, showing how $t/\mathtt{depth}(\mathbf{D})$ (time per request depth) changes w.r.t. the number of descriptions in the repository (N), while performing a Concept Covering of a request and confirms the polynomial behavior of the algorithm adopted.

5. Conclusions

In this paper we have presented an approach and a system, fully exploiting technologies and languages mutuated from the Semantic Web initiative, for automated building blocks selection within SAP R/3. The system exploits to such aim non-standard inference services, useful to provide

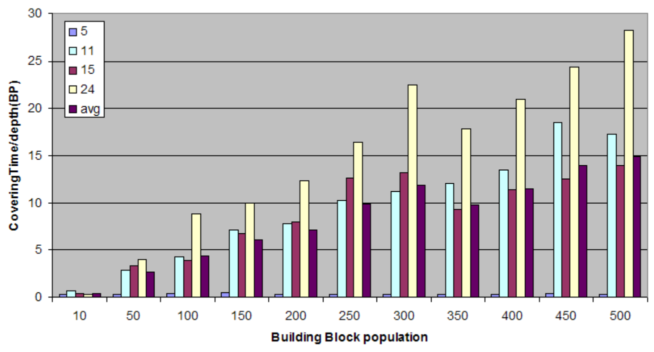


Figure 7. System performances

approximate matches when an exact solution is unavailable. For the OWL-DL subset we use, algorithms adopted are polynomial in time, as for Concept Covering we adopt greedy algorithms. It is in fact well known that also basic set covering is NP-Hard, and from numerical tests carried out, computing a non-greedy Concept Covering would prevent any practical use of our approach.
While consultants evaluation –the end users of the system– is ongoing, we already received extremely useful feedback. As detailed in the previous sections, such feedback made us modify the Concept Covering algorithm introducing also concept Contraction and the possibility to explicit mandatory requirements. Also noteworthy is that we discovered a further practical use (we honestly had not ever thought about) of the toolkit in a different stage of the consulting process. That is in the initial price determination for the customer. In practice, by adding in the BBs descriptions costs and value-added prices, we are able to simplify the price evaluation of the consultants intervention, and the cost of modules that if not available in the client packages have to be purchased. Such add-on –quite simple in theory– is currently under construction.

Acknowledgments

We wish to acknowledge partial support of Apulia regional project ACAB-C2 (System for Automated Composition of business processes in ERPs using Abduction-Based Concept Covering), PON project PITAGORA, and EU-FP-6-IST STREP TOWL.

References

[1] SAP AG. Contract-based Service Order with Third-Party Services and Time & Material Billing, SAP Best Practices for Service Providers. http://help.sap.com/bestpractices/industry/ serviceindustries/v346c_it/html/ index.htm, 2003.

[2] F. Baader, D. Calvanese, D. Mc Guinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2002.

[3] S. Colucci, T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. Concept Abduction and Contraction in Description Logics. In *Proc. of the 16th Intl. Workshop on Description Logics (DL'03)*, volume 81 of *CEUR Workshop Proceedings*, September 2003.

[4] S. Colucci, T. Di Noia, E. Di Sciascio, F.M. Donini, G. Piscitelli, and S. Coppi. Knowledge Based Approach to Semantic Composition of Teams in an Organization. In *Proceedings of the 20th Annual ACM (SIGAPP) Symposium on Applied Computing SAC-05*, pages 1314–1319. ACM, 2005.

[5] S. Colucci, T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. Concept Abduction and Contraction for Semantic-based Discovery of Matches and Negotiation Spaces in an E-Marketplace. *Electronic Commerce Research and Applications*, 4(4):345–361, 2005.

[6] F. Di Cugno, T. Di Noia, E. Di Sciascio, F.M. Donini, and E. Tinelli. Building-Blocks Composition based on Business Process Semantics for SAP R/3. In *ISWC 2005 Workshop Semantic Web Case Studies and Best Practices for eBusiness (SWCASE05)*, 2005.

[7] T. Di Noia, E. Di Sciascio, and F.M. Donini. Extending Semantic-Based Matchmaking via Concept Abduction and Contraction. In *Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2004)*, LNAI. Springer, October 2004.

[8] T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. Abductive matchmaking using description logics. In *Proc. of the 18th Intl. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, pages 337–342. MK, 2003.

[9] T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. A system for principled Matchmaking in an electronic marketplace. *Intl. Journal of Electronic Commerce*, 8(4):9–37, 2004.

[10] P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. Bradford Books, MIT Press, Cambridge, MA, 1988.

[11] A. Goderis, U. Sattler, P. Lord, and C. Goble. Seven Bottlenecks to Workflow Reuse and Repurposing. In *proc. 4th International Semantic Web Conference (ISWC 2005)*, volume 3279 of *LCNS*, 2005.

[12] R/3 Simplification Group. *Best Practices for mySAP All-in-One, BUILDING BLOCKS CONCEPT*, June 2003.

[13] M-S. Hacid, A. Leger, C. Rey, and F. Toumani. Computing Concept Covers: a Preliminary Report. In *DL'02*, volume 53 of *CEUR Workshop Proceedings*, 2002.

[14] J. A. Hernández. *SAP R/3 Handbook*. McGraw Hill, 2nd edition, 2000.

[15] SAP Building Block Library. http://help.sap.com/bestpractices/ BBLibrary/bblibrary_start.htm.

[16] August-Wilhelm Scheer and Frank Habermann. Making ERP a success. *Communications of the ACM*, 43(4):57–61, 2000.