Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers in Industry 56 (2005) 639–662
COMPUTERS IN INDUSTRY
www.elsevier.com/locate/compind

Aligning an ERP system with enterprise requirements: An object-process based approach

Pnina Soffer [a,*], Boaz Golany [b], Dov Dori [b]

[a] *Haifa University, Carmel Mountain, Haifa 31905, Israel*
[b] *Technion—Israel Institute of Technology, Technion City, Haifa 32000, Israel*

Received 29 March 2004; received in revised form 14 December 2004; accepted 13 March 2005
Available online 17 June 2005

Abstract

One of the main problems in ERP implementation projects is how to align an off-the-shelf software package with the business processes of the enterprise implementing it. The paper proposes a requirement-driven approach, which benefits from reusing the business process design without being restricted by predefined solutions and criteria.

The approach applies an iterative alignment process, which employs an algorithm that matches a model of the enterprise requirements with a model of the ERP system capabilities. The algorithm identifies possible matches between the two models and evaluates the gaps between them despite differences in their completeness and detail level. It provides the enterprise with a set of feasible combinations of requirements that can be satisfied by the ERP system as a basis for making implementation decisions. We use Object-Process Methodology (OPM) to model both the ERP system and the enterprise requirements, and utilize the pair of resulting OPM models as input for the matching algorithm. The alignment algorithm has been tested in an experimental study, whose encouraging results demonstrate the ability of the approach to provide a satisfactory solution to the problem of aligning an ERP software package with an enterprise business model.

Keywords: Enterprise Resource Planning; Requirement-driven; Similarity; Business process design

1. Introduction

During the past decade, Enterprise Resource Planning (ERP) systems have become the leading type of information systems in industrial enterprises. ERP implementation raises several critical issues, reviewed by [1]. One such issue is the alignment of an off-the-shelf software package with the business processes of the enterprise implementing it. An ERP system is designed to serve a large variety of enterprises. As such, it has many options for supporting various business processes used in different types of enterprises. The system configuration, defined by the values assigned to the system's control parameters during the implementation, determines the exact operations and processes supported by the system in the specific enterprise [4,17,22].

* Corresponding author. Tel.: +972 4 8294512; fax: +972 4 8235194.
E-mail address: spnina@is.haifa.ac.il (P. Soffer).

0166-3615/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.compind.2005.03.002

The implementation of an ERP system is often accompanied by a Business Process Reengineering (BPR) that changes the way the enterprise operates [22]. Unlike traditional BPR, referred to as ''fundamental rethinking'' [20], when applying some ERP package, the business processes need to be designed within the framework of the target ERP system, preferably without resorting to extension of the system's capabilities.

The alignment problem, also known as ''gap analysis'', exists virtually in every ERP implementation project. While an initial gap analysis is performed as part of the system selection process, detailed gap analysis and alignment is performed during the system implementation, and determines the customization and configuration of the system. Solving this problem is critical to the success of an ERP implementation project, since it determines the future processes of the enterprise and the way the ERP system will support them. Adopting standard business processes may adversely influence the competitive advantage the enterprise may be enjoying, and should therefore be carefully considered to ensure that the enterprise does not lose it [7]. However, unnecessary software customizations may consume resources that exceed the planned schedule and budget of the ERP implementation project and may harm the system's integrity, especially through future upgrades [2,21].

Common tools that support the alignment process take a solution-driven approach. The ASAP method of Rapid SAP R/3 implementation, SAP's Business Engineer [4,16] and Baan's DEM [30,43] refer to predefined ''best practice'' models and configurations. Based on the premise that the enterprise has to adapt itself to the package rather than the other way around, one of the ''best practice'' solutions is to selected and adopted ''as is'' or, at best, with minimal changes. The selection of a solution to be reused is based on predefined reuse criteria, which vary from a rough logistic characterization of the enterprise to a detailed questionnaire, addressing a variety of issues. The actual requirements of the enterprise are not explicitly considered in this process. Rather, they exist only as part of the human knowledge and reasoning underlying the solution selection. Gaps between the enterprise needs and the system are basically ignored or solved by requiring the enterprise to adapt to an available solution.

This solution-driven approach speeds up the implementation, reduces its cost and provides a high quality, bug-free solution. However, these benefits are worthwhile only if and when the enterprise indeed finds a solution that suits its needs. In cases where one or more core processes of the enterprise are unique to the extent that they are not satisfactorily addressed by the predefined reuse criteria, the alignment process cannot be supported. Rather, it is typically done in an ad-hoc, intuitive manner, which requires considerable efforts. Due to the high complexity of ERP systems, even a small deviation from a given configuration is risky, and requires an extensive verification effort [16].

Despite the common wisdom that suggests that enterprises can and should standardize their processes in alignment with ''best practice'' solutions [7,22], research reports indicate that it is not always the case in practice. Daneva [5], who measured requirements reuse in SAP R/3 implementations, found that full reuse was not achieved, although in some cases the rate of reuse was remarkably high. Daneva [6] analyzed the application practice of the ASAP method, and reported that the lack of change impact analysis in the observed projects led software customizations that became unanticipatedly complex and exceeded the planned schedule.

This paper proposes a requirement-driven approach to the alignment problem. Ideas and aspects related to requirement-driven approaches are presented and discussed in [33,34–36,38,40,41]. By facilitating the reuse without imposing a predefined set of reuse criteria, a requirement-driven approach provides a systematic support for both standard enterprises and unique ones. While the requirement-driven approach presented in [33–36] and the related fitness relationship [37] provide a systematic basis for human reasoning, the requirement-driven alignment process presented in this paper employs an automated matching algorithm between the enterprise requirements and the ERP system capabilities. Empirical results demonstrate the feasibility of the approach and its potential capability of providing a satisfactory solution to the alignment problem.

A requirement-driven alignment approach emphasizes the enterprise requirements rather than the ERP system's capabilities and standard solutions. The requirements themselves serve as reuse criteria. These are matched against the capabilities of the ERP system

in order to identify the required solution parts and the remaining gaps. The approach enables a systematic examination of the ERP capabilities beyond the predefined "best practice" solutions. It addresses the ERP options as a set of components or building blocks to be assembled for a suitable solution. Hence, though unable by itself to increase the flexibility of the existing system, it utilizes this flexibility up to its limits.

The matching is carried out between two models, one representing the enterprise requirements and the other—the ERP system capabilities. These two models should therefore have common modeling conventions that would serve as a basis for the issues addressed. In our approach, both models relate to business concepts. This is a natural way for an enterprise to express its needs and is also applied by solution-driven tools. Furthermore, it is especially important to express the system capabilities in business terms if we are to adapt the enterprise to the software package rather than the other way around.

While the ERP system capabilities are modeled once, the resulting model can be used repeatedly for any number of implementation projects. The requirements model, in contrast, needs to be constructed for each implementation separately. Moreover, it cannot be expected to remain static through the alignment process. Rather, the alignment is an iterative process, where requirements are matched against the system model, reformulated on the basis of the matching results, and matched repeatedly, until a satisfactory solution is obtained. The initial requirements model is formed on the basis of a prior gap analysis, performed in the course of the package selection. The result of this prior analysis, which is an initial mapping of the enterprise entities to the entities that exist in the ERP system, sets the terminology and entity names in the requirements model. The reformulation of some requirements is aimed at refining them so they can be met by the available system capabilities, and may include different entity mappings. In the course of this process, some requirements may be altered, while others can possibly be abandoned. The requirements reformulation is the result of human reasoning, and it is verified and complemented by an automated matching algorithm, which matches the two models and computes their similarity.

The remainder of the paper is organized as follows: Section 2 introduces the modeling concepts and discusses the inputs to the alignment process, which are the enterprise requirements and the ERP system model. Section 3 develops the iterative alignment process and the details of the matching algorithm. Empirical results are provided and analyzed in Section 4, and a discussion in Section 5 summarizes the potential benefits and drawbacks of our approach.

2. The inputs to the alignment process

Since both the ERP system model and the enterprise requirements model relate to business issues, they should be represented in the same modeling language to enable their matching. Rolland and Prakash [33–36] suggest that a map representation be applied in both cases. We apply Object-Process Methodology (OPM) [12] to model both the ERP system and the enterprise requirements. The choice of modeling language is justified in our earlier work. In [41] an ontological evaluation framework, based on a requirements ontology, is presented and applied for evaluating OPM's expressive power as a requirements specification language. Selection criteria for ERP modeling languages are presented in [40], applied to OPM, and assess its suitability for this purpose as well. We first provide a brief introduction to OPM and then discuss the enterprise requirements and the ERP system model in OPM terms

2.1. Object-Process Methodology

Object-Process Methodology, described in detail in [12], has been applied for various purposes, such as computer integrated manufacturing [9], image understanding [10], modeling research and development environments [25], algorithm specification [44], document analysis and recognition [8], and modeling electronic commerce transactions [11]. It employs a CASE tool that supports a variety of stages in the system lifecycle [13].

OPM's building blocks are two equally important classes of entities: objects and processes, which are connected by procedural links and structural relations. Most object-oriented modeling methods and enterprise modeling methods require the use of a set of models (also called views), each with its diagramming symbols and conventions, to describe different aspects

of the system. OPM, in contrast, uses a single graphic tool, the Object-Process Diagram (OPD) set, to model the major system's aspects, structure and dynamics. Using a single view eliminates the model multiplicity problem from which object oriented modeling methods suffer. Solving the model multiplicity problem requires considerable efforts to integrate the various views into a coherent system model and to keep consistency among them [29]. Single-view representation is also much more convenient when two models are being matched.

While using a single model representation, OPM keeps simplicity through two abstracting-refinement mechanisms that control the visibility of the system details. Unfolding of entities (objects and processes) and zooming into them enables top-down analysis, yielding a hierarchical OPD set, which specifies the structure and behavior of the system at a spectrum of abstraction-refinement levels. To enable middle-out model construction, which is common practice in many real-life systems analysis projects, the reverse operations, folding and out-zooming are also possible.

As an example, Fig. 1(a) shows the process *Purchase Order Handling*, which zooms into (includes) the processes *Purchase Order Maintaining*, *Subcontracting*, *Purchased Goods Receiving*, and *Purchase Order Closing*. This process affects objects such as *Purchase Order* and *Inventory by Item*, and is enabled by objects, such as *Supplier* and *Item*. In Fig. 1(b), the process *Purchased Goods Receiving* is unfolded, revealing three specializations. The occurrence of the specialized processes *Location Controlled Receiving* and *No Location Controlled Receiving* is conditioned by the states of the Boolean object "*Location Controlled*?", which characterizes a *Warehouse*, while the *Non-inventory Item Receiving* is conditioned by *Item* being a *Non-inventory Item*. Note, that for the sake of brevity, some objects that appear in Fig. 1(b), such as *Inventory by Location*, are omitted from Fig. 1(a). Each of the entities can be further refined (unfolded or in-zoomed) in other OPDs. The entire OPD set constructed in this way constitutes a complete specification of the system.

2.2. The enterprise requirements

The enterprise requirements provide the basis for both the selection of the ERP system and the alignment process. For these two purposes, completeness is not necessarily a desired property of the requirements [14,15,23,27,36]. This paper addresses the alignment of a selected package rather than the selection of a package, and therefore relates to functional requirements only, assuming that the role of non-functional requirements is mainly in the selection phase. Since the alignment process involves changes in the enterprise, which is required to adapt to processes supported by the ERP system, the requirement specification should aim at providing the enterprise with the flexibility and adaptability to these available processes, while assessing their suitability. Complete system and interface requirements may result in rigidity that would render an exact matching solution within the software package infeasible.

Based on this premise, the requirements framework presented in [41] classifies the requirements into four different types of information:

- Core system interfaces, whose detailed design is of considerable importance to the enterprise. This is a relatively small set of specified inputs and outputs, typically required for business processes involving interaction with external agents (e.g., reporting to the tax authorities). The system interfaces are modeled in OPM as detailed inputs and outputs of processes performed by the system.
- Core business processes, which must not be changed through the alignment process. The details of these processes are unique, as they generate the competitive advantage of the enterprise. These may include logistic processes, which support an outstanding supply mechanism, or quality assurance processes, which ensure an exceptional quality level. Business processes are modeled as such in OPM, including the processes and the objects they involve.
- Business rules, which express the enterprise goals (or certain external restrictions that must be followed) and control the business processes. The enterprise goals are operationalized by the business rules, which provide the underlying logic that remains invariant to changes in the business processes in the course of the alignment process. Business rules are, actually, constraints posed on the business processes. An OPM representation of a business rule shows only the necessary (partial) details of the process and objects involved.

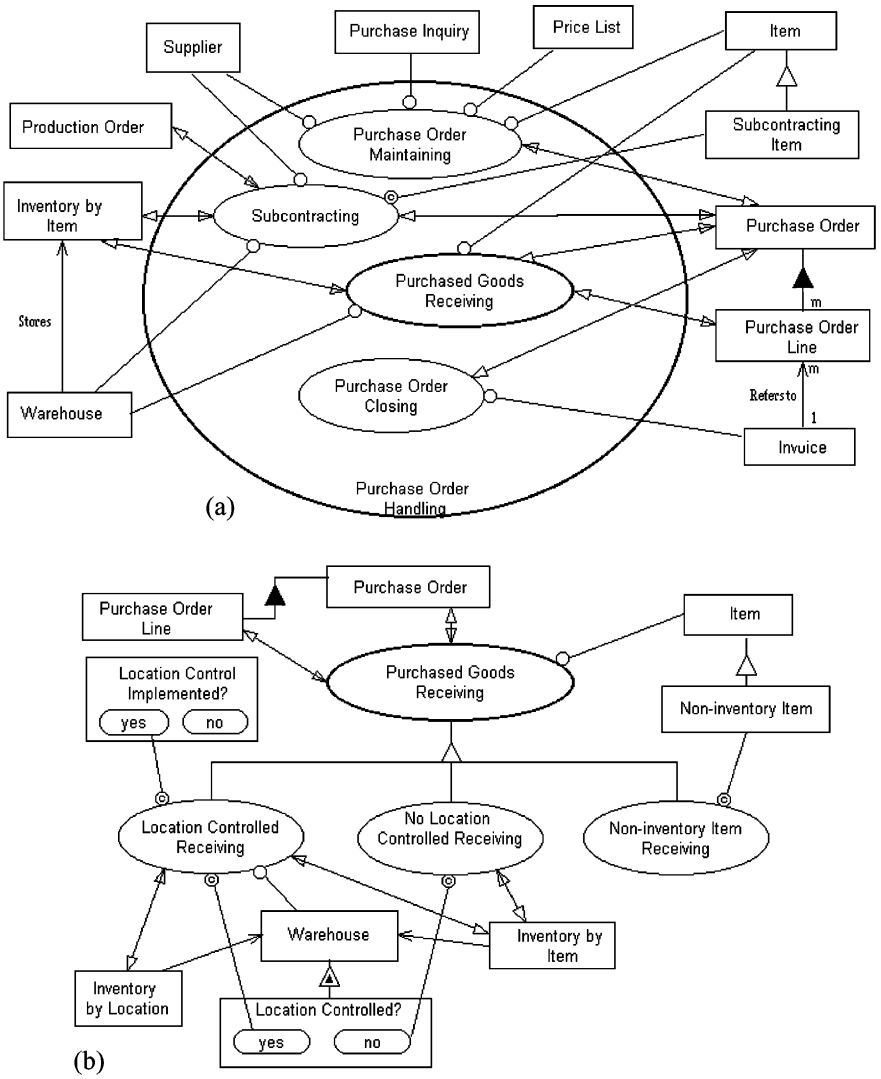

Fig. 1. An OPM representation of a Purchase Order Handling process.

- Information objects, which are manipulated by the specified business processes, controlled by the business rules, and participate in the specified interfaces. Information objects, their structure and relationship are directly modeled in OPM.

Soffer et al. [41] provides formal definitions of these four elements, and discusses the relations between them, the enterprise goals, and the system under consideration.

2.3. The ERP system model

The ERP system model is a vehicle for aligning the system with the enterprise requirements. As such, it should represent the entire scope of options and business process variants supported by the system and represent dependencies among alternative options. Some ERP packages, such as SAP and Baan, provide modeling tools and solution models as part of the system [4,43]. We decided not to use these models as a basis for our alignment approach, since relying on a model that resides within a specific package would decrease the genericity of the approach. Furthermore, while these models serve the purpose of ERP representation, they are not necessarily suitable for specifying the requirements (see, for example, the evaluation of EPC, a language used by the SAP models, in [41]). Therefore, consistent with the requirements model, the ERP system model we use is an OPM model. It is obtained through a reverse engineering process, such as the one described in [40]. The resulting model may serve as a reference as long as the modeled ERP system has not changed.

In the OPM representation, alternatives are represented by different specializations of a generic process. In Fig. 1, for example, *Location Controlled Receiving*, *No Location Controlled Receiving* and *Non-inventory Item Receiving* are alternative purchase receiving processes that the ERP system under consideration supports. The conditions specified for each alternative process reveal their dependency on parameter values or states of a Boolean object such as yes or no in the Boolean object "*Location Controlled*?". The details of each entity (object or process) may be revealed through refinement in lower-level OPDs, which may, in turn, include other alternative specializations of its entities. The refinement mechanism allows for exact specification of the ERP system at any level of detail.

The structure of the OPD set can be presented as a graph, called the system model hypergraph, where each OPD is a node with arcs connecting it to descendant OPDs that expose details of one or more of the entities in the parent OPD. One or more arcs can go from each node to its descendant diagrams. Each OPD in the OPD set (except the root, i.e., the top-level OPD) results from refining an entity of a parent OPD, and therefore has an incoming arc connecting it to that parent. Since the origin entity may appear in several OPDs, more than one arc may lead to each node. The hypergraph is therefore a directed graph (but not necessarily a tree). Fig. 2 is the system model

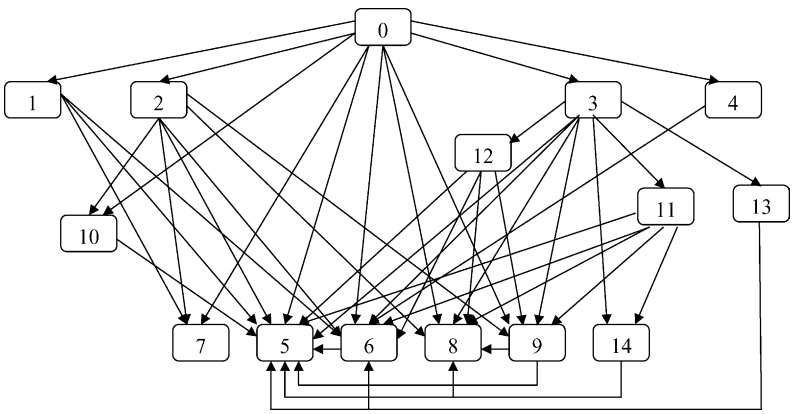

Fig. 2. The system model hypergraph of the Purchase Order Handling module, Top level OPD (OPD0) is detailed in Fig. 1a OPDs of other subprocesses is provided in Table 1.

Table 1
The OPDs in the *Purchase Order Handling* module

OPD number	OPD name	Descendant OPDs
0	Purchase Order Handling	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
1	Purchase Order Maintaining	5, 6, 7
2	Subcontracting	5, 6, 7, 8, 9, 10
3	Purchased Goods Receiving	5, 6, 8, 9, 11, 12, 13, 14
4	Purchase Order Closing	6
5	Item	
6	Purchase Order	5, 7
7	Supplier	
8	Warehouse	
9	Inventory by Item	5, 8
10	Production Order	5
11	Location Controlled Receiving	5, 6, 8, 9, 14
12	No Location Controlled Receiving	5, 6, 8, 9
13	Non-inventory Item Receiving	5, 6
14	Inventory by Location	5, 8

hypergraph, whose top-level OPD is shown in Fig. 1(a). The names of the OPDs in the hypergraph and their descendants are provided in Table 1.

3. The ERP-requirements alignment

The ERP-requirements alignment is an iterative process that receives as inputs the enterprise requirements and ERP system model. In each iteration it employs an algorithm described below for matching the two models.

3.1. The ERP-requirements matching algorithm

The options that are available in an ERP system are highly dependent on each other. Hence, a combination of features available in a specific configuration may not be available in other configurations [22]. Identifying a set of requirements that are met by the system does not guarantee that their combination is feasible in a single configuration. The system's internal dependencies form constraints on the feasible solution, which are not always clear to the implementation team members. For example, in the Baan ERP system, it is possible to allocate a specific inventory unit to a specific order (such an action is termed ''hard allocation''). However, this option is valid only if the warehouses are not location controlled, that is, in a configuration where the state of the Boolean object ''*Location Control Implemented*?'' is ''no''. Separately verifying that the system is capable of handling requirements of hard allocation and location control would yield a positive answer, despite the fact that their combination is not feasible. The matching algorithm aims at providing a solution space, which enumerates the feasible combinations of requirements satisfied by the system along with their matching scores (MS).

3.1.1. Algorithm overview

The matching algorithm, depicted as a meta-model in the OPD of Fig. 3, consists of two main processes: Single Requirement Matching (SRM) and Bottom-Up Aggregation (BUA). SRM examines each requirement separately and looks for matching diagrams within the system model. It generates a matching score for each pair $\langle R, E \rangle$, where R is a requirement OPD and E is an ERP system OPD. The SRM output serves as input to BUA, which aggregates the matching scores up the system model hypergraph and identifies their feasible combinations. These feasible combinations form a solution space, denoted herein as the *Matching Option Set*.

3.1.2. Single Requirement Matching (SRM)

SRM assesses the similarity between pairs of OPD portions, each consisting of an OPD portion expressing a requirement in the Requirement model and its counterpart in the system model OPD. As discussed in

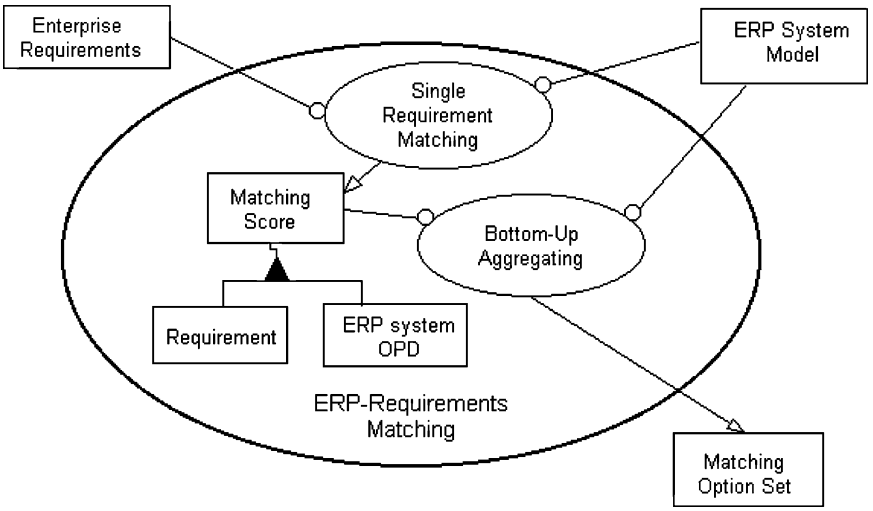

Fig. 3. An OPD of the matching algorithm.

Section 2, the requirements are generally incomplete and represent only the details that are essential to the enterprise, while the system model is usually more detailed. SRM is the part of the matching algorithm that is designed to resolve this mismatch.

SRM computes a matching score for $\langle R, E\rangle$ based on two measures: Entity Similarity (ES) and Relational Similarity (RS). ES is the proportion of entities in R that have a matching entity in E, i.e., an entity whose name and type are identical to those of the entity in R.

$$\mathrm{ES}_{\langle R,E\rangle} = (\text{number of entities in } R \text{ matched by entities in } E)/(\text{number of entities in } R)$$

ES, the Entity Similarity, is computed using a simple query, which compares the entities of each of the system model OPDs with the entities of a given requirement, and counts the entities whose type and name are identical. The query provides a set of candidate matching pairs $\langle R, E\rangle$, whose ES score exceeds a given threshold.

RS, the Relational Similarity, which is assessed for each pair in this set, measures the similarity of the link structure in a pair $\langle R, E\rangle$. An in-depth discussion of the principles underlying RS and its computation appears in [39]. RS is computed by an exhaustive search for matching of each link in R. Each link matching yields one of the following results:

(1) The link is matched by a link in E, which is of the same type and relates matching entities for both the source and the destination of the link. The Link Match (LM) is then assessed by comparing the cardinalities of the links in R and E. The cardinality of a link is determined by participation constraints defined for its source and destination. The cardinality comparison may yield one of the following: (a) The participation constraints of both the source and destination are identical, in which case LM equals 1. (b) The participation constraints of either the source or the destination are identical, in which case LM equals a constant c_1. (c) The participation constraints of both the source and destination are different, in which case LM equals a constant c_2. The constants c_1 and c_2 ($0 \leq c_2 \leq c_1 \leq 1$) are defined by the user and reflect the extent to which he wants links of different cardinalities to be considered as matching.
(2) The link has no match in E, in which case LM equals 0.
(3) The link is matched by a path in E, which is equivalent to its type and relates matching entities for both the source and the destination of the link.

3.1.2.1. Path and equivalence. A path is a sequence of links and entities, connecting a source entity to a destination entity. A path is considered equivalent to a link of specific type if it can be abstracted to a link of

this type. Equivalence is identified on the basis of equivalence rules, defined for each link type in OPM. For a given link type, equivalence rules state link types that are allowed in a path, link types that must be in a path and in some cases their required position: at the source of the path or at its destination.

The identification of a path equivalent to a given link addresses the mismatch in detail level of the two models under consideration, as illustrated in Fig. 4.

Fig. 4(a) is a model of a requirement, where *Delivered Quantity* and *Receipt Date*, which are attributes of a *Purchase Order Line*, are affected by the *Purchase Receipt Registering* process, i.e., the registration of a purchase receipt. The ERP system OPD, given in Fig. 4(b) shows that the *Purchase Receipt Registering* process updates the attributes of a *Purchase Receipt* object, which is structurally related to a *Purchase Order Line*. This structure allows several receipts to be registered for a single purchase order line. The characterization links between *Purchase Order Line* and its attributes in the requirement model OPD do not exist in the system model OPD. Nevertheless, this has no practical implication on the acceptance of the process by the enterprise, since from the user's point of view the structural relation between *Purchase Receipt* and *Purchase Order Line* is as good as having *Delivered Quantity* and *Receipt Date* as attributes of *Purchase Order Line*. Therefore, SRM identifies the path in E from *Purchase Order Line* to *Delivered Quantity* (as well as the path between the former and *Receipt Date*) through *Purchase Receipt* as being equivalent to the characterization link in the requirement model R.

3.1.2.2. Aggregated cardinality. The aggregated cardinality is the cardinality between two entities that are not linked directly, but through a path. The participation constraint of the source entity of a path is the product of all the source participation constraints of its links, and the destination participation constraint is the product of all the destination participation constraints of its links. The product of many and many ($m \times m$) is regarded as many. The aggregated cardinality of a path, identified as equivalent to a link, serves as a basis for computing the link's LM.

In the example of Fig. 4(b), the aggregated cardinality of the path from *Purchase Order Line* to *Delivered Quantity* is 1:m. Hence, the LM of the requirement characterization link, whose cardinality is 1:1, equals c_1.

The Relational Similarity (RS) of $\langle R, E\rangle$ is computed as the average of the LM of the links in R.

Having defined Entity Similarity and Relational Similarity, the overall Matching Score (MS) of a pair $\langle R, E\rangle$ is computed as a weighted sum of the two similarity measures, provided neither of them equals zero.

$$\mathrm{MS}_{\langle R,E\rangle} = W_1 \times \mathrm{ES}_{\langle R,E\rangle} + W_2 \times \mathrm{RS}_{\langle R,E\rangle};$$
$$\mathrm{ES}_{\langle R,E\rangle} > 0, \quad \mathrm{RS}_{\langle R,E\rangle} > 0, \quad W_1 + W_2 = 1$$

where W_1 and W_2 are weights assigned to ES and RS, respectively.

As an example, the computation of MS for the pair of requirement and ERP system OPDs in Fig. 4,

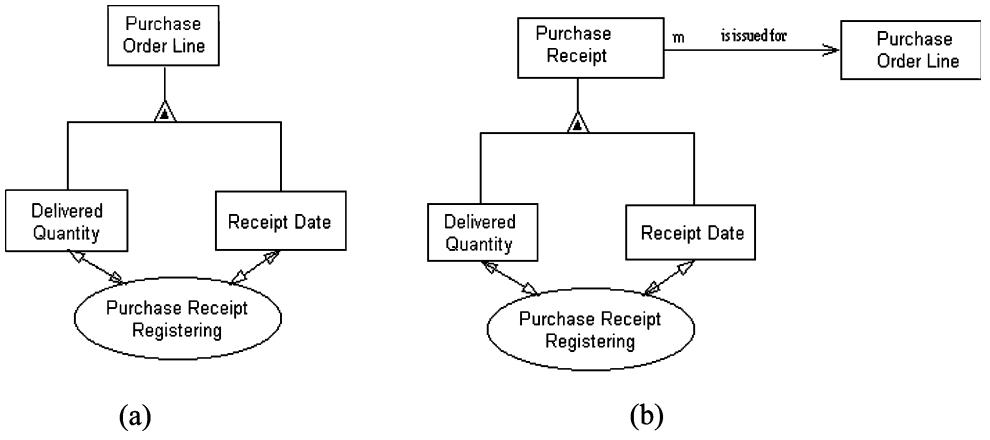

Fig. 4. An example of differences in the detail level between a requirement model, R (a), and the system model, E (b).

applying equal weights $W_1 = W_2 = 0.5$, $c_1 = 0.6$ and $c_2 = 0.5$ for the LMs, is as follows:

All the entities of Fig. 4(a) have matching entities in Fig. 4(b), therefore ES = 1.

Two of the four links of Fig. 4(a) have exact matching links in Fig. 4(b). The two characterization links, whose cardinalities are 1:1, have equivalent paths in Fig. 4(b), whose aggregated cardinalities are 1:*m*. Therefore, RS = (1 + 1 + 0.6 + 0.6)/4 = 0.8.

The overall matching score, $MS = 0.5 \times 1 + 0.5 \times 0.8 = 0.9$, indicates that there is a high degree of similarity between the requirement and the ERP model, but they are not identical.

Note, that the inequality $RS \leq ES$ always holds, since a required link can be found only if both its source and its destination are included in the OPD. Hence, the higher W_1 is the higher MS is computed, while a higher W_2 leads to a higher sensitivity of the MS to structural differences, and disregards the existence of entities in the OPD if they are not linked as required.

Many earlier works that measure similarity of models apply Entity as well as Relational Similarity [18,24,26,28,32,42]. Another approach to similarity measurement deals with an edit distance, referring to the effort required to overcome the differences between the models [3]. Relating to the classification of analogical reasoning mapping solutions [19], the SRM preserves relational structure and semantic categories. As opposed to earlier works that apply similarity measurement [31,32], our Entity Similarity measure does not apply a thesaurus-based affinity score, since the mapping of requirement entities to ERP system entities in an ERP implementation bears consequences that extend far beyond pure semantic similarity. By precise mapping of enterprise entities to ERP entities, the implementing engineer assigns control mechanisms available in the software package to the processes under consideration. Consider, for example, the terms ''Production rate'' and ''Operation rate'', which may be found related by a reasonable thesaurus. In Baan ERP system these two terms represent data of a different meaning and role. Production rate is the number of units produced per hour, while operation rate is the cost associated with a production operation. Expressing a requirement using these precise terms requires domain knowledge. Relying on a thesaurus for identifying ''similarity'' between ERP model terms and those that appear in the requirement may lead to incorrect similarity assessment. Therefore, our approach is to let the user apply knowledge and decide on the precise mapping of terms from the ERP model to the requirements. The user may browse the system entities and select their names while modeling the requirements to ensure that the terms in the requirement model comply with the terminology of the system model.

The RS score of the SRM differs from Relational Similarity used in previous works [24,32,42] in the identification of an equivalent path, which allows matching links between entities that are not directly related in the ERP system model. This approach provides a way for resolving the mismatch in detail level between the two models.

Our similarity measure, unlike [3], does not account for edit cost, since ''editing operations'' entail software customizations to the ERP system, which cannot be computed automatically as part of the similarity measurement.

3.1.3. Bottom-Up Aggregation

The Bottom-Up Aggregation process, which is the second part of the ERP-requirements matching algorithm, provides the feasible combinations of the requirements, whose matching scores have been computed by SRM (see Fig. 3).

The following definitions and notations are used for specifying the BUA procedure.

Arc $\langle o, t \rangle$—each arc in the ERP system model hypergraph is denoted by $\langle o, t \rangle$, where o is the origin node and t is the destination node. The top-level node is denoted as the *root node*.

In the example given in Fig. 2 and Table 1, the arc $\langle 3, 9 \rangle$ relates the OPD that specifies *Purchased Goods Receiving* to the OPD specifying *Inventory by Item*.

Arc Level $L(a)$—Let a be an arc in the graph, then its level in the graph, denoted by $L(a)$, is the maximal number of arcs connecting its origin node to the root node in an a-cyclic route.

Note that the graph may contain cycles, but they do not affect the level of an arc. In the example, $L(\langle 9, 8 \rangle) = 3$, corresponding to the route $\langle 0, 3 \rangle$, $\langle 3, 11 \rangle$, and $\langle 11, 9 \rangle$, i.e., from the root to *Purchased Goods Receiving*, *Location Controlled Receiving*, and *Inventory by Item*.

Originating Arc Set $H(d)$–Let d be a node in the graph, then $H(d) = \{a | a = \langle d, t\rangle\}$ is the set of all the arcs originating in d.

In the example, $H(3) = \{\langle 3, 5\rangle, \langle 3, 6\rangle, \langle 3, 8\rangle, \langle 3, 9\rangle, \langle 3, 11\rangle, \langle 3, 12\rangle, \langle 3, 13\rangle, \langle 3, 14\rangle\}$.

Descendant Node Set $J(d)$—Let d be a node in the graph, then $J(d) = \{t | \langle d, t\rangle \in H(d)\}$ is the set of immediate descendant nodes of d.

In the example, $J(3) = \{5, 6, 8, 9, 11, 12, 13, 14\}$.

Arc Dependency Function $F_d[V]$—Let d be the origin node of n arcs and $V = (v_1, v_2, \ldots, v_n)$ a vector of strings attached to the arcs. Then the function $F_d[V]$: $V \rightarrow C$, which uses the logical operators AND, OR, and XOR to express the dependencies among the arcs originating in node d, maps V to a logical expression combining these strings.

Each arc relates the OPD embedded in d to an OPD specifying the details of one of its entities, some of which represent alternative options of the ERP system. Therefore, the dependencies that exist among the entities are also present among the respective arcs. The function represents alternative options as having an OR or a XOR operator between them, while a mandatory link is represented by an AND operator.

In the example, the OPD embedded in node 3 (*Purchased Goods Receiving*, specified in Fig. 1(b)) includes three alternative processes: *Location Controlled Receiving*, *No Location Controlled Receiving*, and *Non-inventory Item Receiving*, whose corresponding arcs are $\langle 3, 11\rangle$, $\langle 3, 12\rangle$, and $\langle 3, 13\rangle$, respectively. The operator in $F_d[V]$ that represents the alternative relations between the arcs is XOR. While the processes *Location Controlled Receiving* and *No Location Controlled Receiving* are both related to the objects *Warehouse* (whose outgoing arc is $\langle 3, 8\rangle$) and *Inventory by Item* (arc $\langle 3, 9\rangle$), the object *Inventory by Location* (arc $\langle 3, 14\rangle$) is related to *Location Controlled Receiving* only. Other objects in the OPD, such as *Item* (arc $\langle 3, 5\rangle$), and *Purchase Order* (arc $\langle 3, 6\rangle$), are related to all the alternative processes. Each such relation is represented in $F_3[V]$ by an AND operator. Therefore, if V holds the destination nodes of the arcs, i.e., $V = (5, 6, 13, 8, 9, 12, 11, 14)$, then $F_3[V] = 5$ AND 6 AND (13 XOR (8 AND 9 AND (12 XOR (11 AND 14)))).

Note that the alternative options may have different scopes, depending on the parameter that controls them. In Fig. 1(b), the object *Location Control Implemented*? is a parameter of the ERP system, defining whether location control is applied throughout the system, while the Boolean object "*Location Controlled*?" is a parameter that characterizes a warehouse and may have different values for different warehouse instances. Therefore, the scope of the alternative processes *Location Controlled Receiving* and *No Location Controlled Receiving* is a single warehouse instance. Similarly, the scope of *Non-inventory Item Receiving* is a single item instance. This is expressed in $F_3[V]$, by an index reflecting the scope of an operator. The function accounts for the scope then becomes:

$F_3[V] = 5$ AND 6 AND (13 XOR_{Item} (8 AND 9 AND (12 $\text{XOR}_{\text{Warehouse}}$ (11 AND 14)))).

Local Matched Requirements Combination $C(d)$—Let d be a node in the graph and $R = \{R_1, R_2, \ldots, R_n\}$ a set of requirements being matched against. Then $C(d)$ is a logical expression specifying the feasible combination of requirements matched by the OPD in d and all its descendants.

$C(d)$ is recursively constructed by the BUA algorithm, which starts after the SRM has computed matching scores for all the given requirements with the OPDs of the ERP system model. The computed matching scores constitute initial $C(d)$, denoted as $C^1(d)$, for a set G of nodes in the system model hypergraph, such that $C^1(d)$ is the combination of matching scores $\text{MS}_{\langle R, d\rangle}$ for each $d \in G$.

The BUA sorts the nodes of G according to their distance from the root node, starting with those whose distance is maximal. At each step it goes one level up and aggregates the results for the upper level. While climbing the set of arcs $\langle o, t\rangle$ from all the descendant diagrams t up to their mutual origin diagram o, $C(o)$ is computed by assigning all the $C(t)$ as a vector in $F_o[V]$. $C(o)$ is computed as:

$C(o) = C^1(o)$ AND $F_o[V]$, where $V = (v_1, v_2, \ldots, v_n)$, such that $v_i = C(t)$ for each $t \in J(o)$.

This procedure is repeated until the top-level diagram (the root node of the hypergraph) is reached. Its combined result C(root) holds all the requirement results aggregated up the system model hypergraph and their logical relations, providing the matching options of the ERP system with respect to the enterprise requirements. The BUA is specified in Fig. 5.

The complexity of the algorithm is $O(n^3)$, where n is the number of diagrams in the system model. Each

For every node $d \in G$ set $C^1(d)$ based on the SRM results.

** Initialization of C(*d*) **

Set L_1 as the maximal $L(a)$ for all the arcs whose $t \in G$.

** Sorting by the distance from the root node **

Set $S_1 = \{a \mid L(a) = L_1, t \in G\}$; Set $S_0 = \{a \mid L(a) < L_1, t \in G\}$.

** The set of arcs whose destination is in G is divided to S_1, the arcs whose level is L_1, and S_0, its complementary **

For every node d such that $S_1 \cap H(d) \neq \varnothing$

** Looping all the nodes whose originating arcs are in S_1 **

Set $V(d) = \{C(t), t \in J(d)\}$

** Assigning C of the descendant nodes in *V* **

If $d \in G$ then C(*d*) = (C(*d*) AND $F_d[V(d)]$)

else C(*d*) = $F_d[V(d)]$.

** Computing C(*d*) **

$G = G \cup d$.

** Adding the node to set G for the next iteration **

Set $M = \{d \mid H(d) \cap S_0 = \varnothing, d \in G\}$; G = G-M.

** Removing the nodes whose up-going arcs have already been climbed **

If $L_1 > 0$ then go back to step 2.

** Looping to a lower arc level.**

Fig. 5. The BUA algorithm.

arc in the graph is visited at most once, and the number of arcs in the graph is $O(n^2)$. The computation of $C(d)$ is linear in the number arcs originating from node d, which, for a completely flat graph, is $O(n)$.

3.2. The iterative alignment process

The alignment process involves iterations of reformulating the requirements and applying the

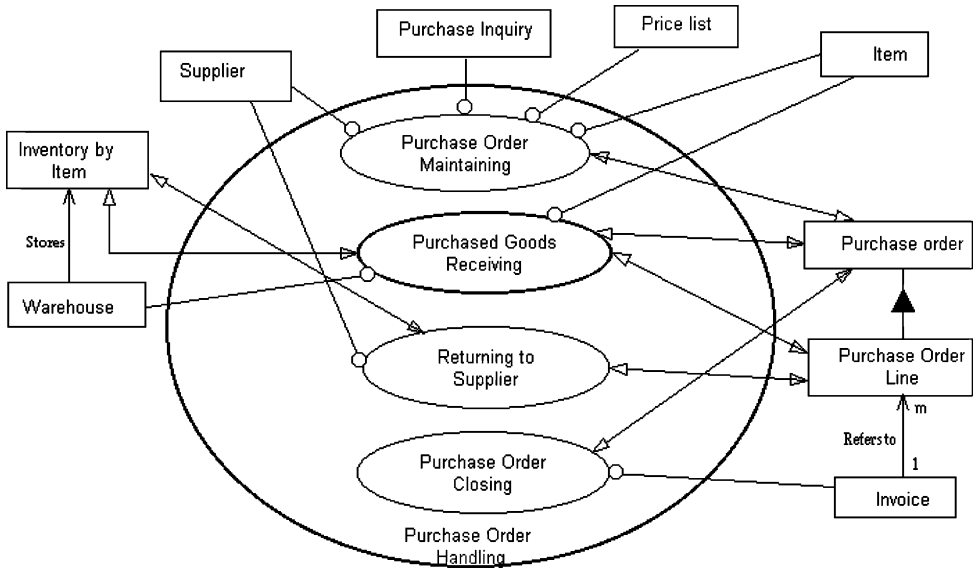

Fig. 6. A requirement model example.

matching algorithm for their verification. Reformulating the requirements involves three types of action: splitting, abandoning, and mapping.

Splitting entails splitting composite requirements into simpler ones. A composite requirement includes at least two entities that may be revealed in lower-level OPDs. A process defined in a requirement may not be defined as such in the system model, but all its subprocesses (steps) exist as parts of other processes. In such cases, it may be possible to ''assemble'' the required process as a sequence of these steps. In order to verify this possibility, each of the process steps and its required interface should be examined as a separate requirement.

As an example, consider the requirement modeled in Fig. 6. An attempt to match this requirement with the system OPD in Fig. 1(a) fails due to the absence of the required process *Returning to Supplier* in the system OPD. However, this process does exist in the system model as part of a lower level diagram, specifying the receipt of purchased goods. The human that analyzes the match results, and finds that the relatively low matching score obtained is due to this absence, should define a new requirement, including only of this process and the objects it relates to (*Supplier*, *Inventory by Item*, and *Purchase Order line*). When the matching algorithm is applied again in the next iteration, the existence of the process and its conformance to the original requirement are verified.

Abandoning requirements is the second way of reformulating requirements. When analyzing the output of the matching algorithm, some requirements may be identified as not being satisfied by the system, or as contradicting other requirements of higher priority. Depending on their importance to the enterprise, a software customization may be considered. Alternatively, if an unsatisfied requirement is of a lesser importance, or if some activities may be handled manually without involving the ERP system, that requirement can be abandoned.

The third operation of requirement reformulation is applying a different mapping. Each time a requirement is modeled and expressed using the system's terminology, some decisions of mapping the enterprise entities to entities of the system are made. The initial mappings applied in the first iteration are the result of the prior analysis performed when the specific ERP system has been selected. Different mapping decisions can be made in the next iterations in attempt to resolve gaps that have been found, since different mappings may yield different matches. Such mappings are applied frequently in manual alignments, where considerable effort is required to verify their appropriateness. Two types of mapping are

possible: (a) Textual mapping, in which an entity is renamed in order to solve a naming mismatch. (b) Substantial mapping, in which a requirement is redefined, while considering alternative mechanisms of the system for achieving a certain goal.

Consider, as an example of a substantial mapping, a case in which the enterprise requires that purchased goods be received as ''inventory on hold'', i.e., inventory that cannot be used until it is inspected and approved. When a system does not satisfy this requirement, an alternative mechanism may be considered. Such a mechanism could be receiving the goods to a reception warehouse, from which the goods cannot be issued, and transferring them to an ordinary warehouse after they are approved. When considering such an option, the requirements are reformulated, expressing the reception process and the required properties of the reception warehouse.

Substantial mapping requires the implementing team members to think creatively and to apply a high level of expertise and knowledge of the system and its internal relations. Once remapping is done, the requirement model is drawn from scratch, expressing the new entities and logic defined. It is important to apply the new mappings to all the requirements. Due to the complexity and integrative nature of ERP systems, the manual verification of the consequences of a mapping decision is a difficult task. Every decision may affect many other parts of the system, and each one of them must be checked and tested in order to ensure their correct operation. Sometimes, the effects cannot be expected, and performing all the necessary tests depends on the skills of the implementation team members.

In the iterative alignment process, new mapping decisions still rely on creative thinking, but their verification is accomplished by the matching algorithm. The matching algorithm can instantaneously scan the entire system model, identify the effects of the reformulated requirement and potential contradictions with other requirements.

The alignment process, illustrated as an OPD in Fig. 7, starts with an initial *Enterprise Requirement Set*, each represented as an OPD. These requirements are verified against the system model by the *ERP-Requirement Matching* algorithm that generates the *Matching Option Set*. The *Matching Option Analyzing* process determines whether further reformulation of the requirements is needed. Specifically, it identifies the OPD whose MS is maximal, analyzes the gaps reflected by the MS and the feasibility of combining it with other selected OPDs (on the basis of the logical expression produced by the BUA). It then identifies possible reformulations to resolve gaps and contra-

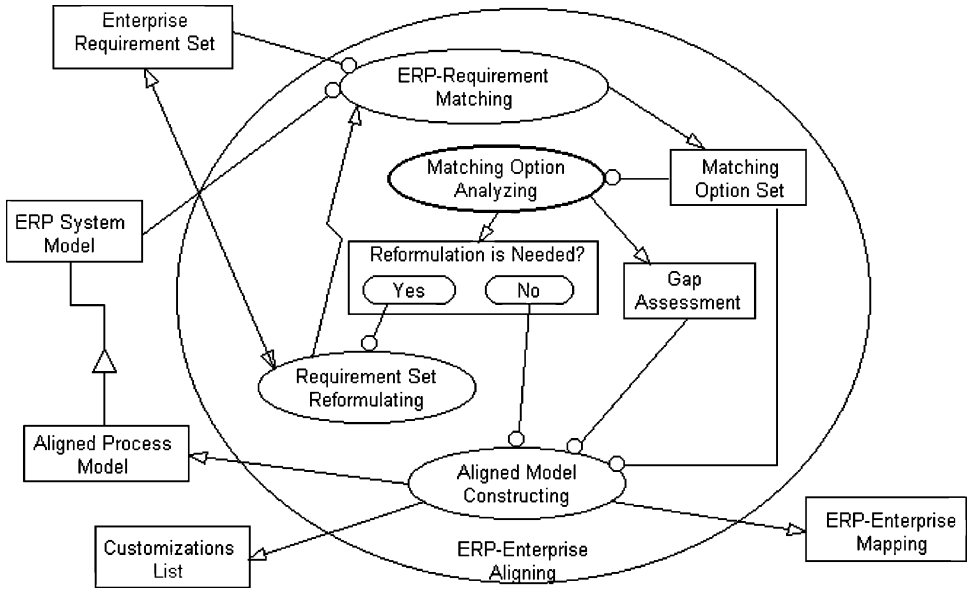

Fig. 7. An OPD of the alignment process.

dictions found. If reformulation is needed, the process of *Requirement Set Reformulating* is carried out, triggering *ERP-Requirement Matching* in another iteration of verifying the reformulated requirements. This sequence of iterations repeats, until all the requirements are satisfied, when no untested reformulation possibility is found, or when the implementation team manager decides that the results are clear enough and enable decision making. Then the *Matching Option Set* and the *Gap Assessment* serve for the *Aligned Model Constructing* process, which creates the *Aligned Process Model*, the required software *Customizations List*, and the final *ERP-Enterprise Mapping*.

The *Aligned Process Model* is a specialization of the ERP system model, consisting of the OPDs identified as satisfying the requirements. Some of these OPDs can be an ''assembly'' of parts belonging to different OPDs in the initial system model. Others may include new objects, processes and links, required as software customizations. In addition to the business process design, the aligned process model provides the ERP configuration, i.e., the ERP control parameter values, to support a set of specified processes.

The required software *Customizations List* is a list of gaps, assessed as significant, which constitute a basis for software customization decisions. The details of the gaps in different mapping scenarios provide a starting point for alternative customization designs.

The *ERP-Enterprise Mapping* corresponds to the selected solution as specified in the aligned process model. This mapping serves for planning the conversion and migration of data from the existing information system to the ERP system.

4. Validation

Validation was carried out in three experiments, whose aim was to establish the feasibility of the approach. The feasibility assessment related to four criteria:

(1) Consistency of the results through controlled changes in the input models. Assuming the matching scores reflect the level at which a given ERP system matches a given set of requirements, a different ERP package, possessing a subset of the capabilities of the current package, should achieve matching scores that are less than or (at best) equal to the current ones. We note, however, that an experiment in the other direction, i.e., extending the requirements, would not necessarily result in a consistent change in the matching scores, since a ''stronger'' or an additional requirement may have a good match in the ERP model.
(2) Sensitivity to the algorithm weights. The matching scores are computed as a linear combination of weighted Entity Similarity and Relational Similarity. Changes in these weights would linearly change the matching scores. Sensitivity analysis of the solution to these weights would identify the range of weights in which the selected solution is invariant.
(3) Validity of the solution obtained by the alignment process to real-life requirements, when compared to the solution obtained manually in a real life project. The specific validation criteria were: (a) the number of gaps identified by the process compared to the number of gaps identified in real life. (b) The number of unreal gaps indicated by the process. (c) The quality of the solutions provided by the process to identified gaps compared to the solutions defined in real life. The quality of the solution could, naturally, be based on human judgment only.
(4) Robustness to variations in the input requirements model. Whenever a domain is modeled by different modelers, the models obtained are expected to be slightly different, due to differences in the perception and modeling attitude of each individual modeler. When the alignment process is to be applied in real life, the implementation team members of each project should draw the requirements model. It is therefore crucial that the individual modeling attitudes that affect the requirements model should not influence the solution provided by the alignment process significantly.

The feasibility of the approach according to these criteria was evaluated in three experiments. As a preparatory step to these experiments, a system model had been constructed, by modeling the purchasing and inventory module of the Baan ERP system using OPM. The system model included 119 OPDs.

4.1. Experiment 1

The aim of this experiment was to test the consistency of the results through controlled changes in the input models, according to the first validation criterion. It entailed a fixed set of requirements repeatedly matched against the ERP system model, in which controlled changes have been made.

4.1.1. Settings

The experiment procedure was as follows:

1. We applied the alignment process on a given set of 23 requirements and the ERP model.
2. We eliminated an arbitrarily chosen process in the ERP system model (thus relate to a "different" and less powerful system). Naturally, the elimination of a process included all its details in lower-level diagrams and all the objects that are its results. As an example, eliminating the process of lot management included eliminating it from all the ERP model OPDs where it appeared (e.g., *Receiving Purchased Goods*), and eliminating the *Lot* object and all its attributes from all the OPDs where it appeared.
3. We applied the alignment process again to the same set of requirements. The expected result was one of the following: (a) the matching scores would remain the same as the initial ones, implying that the eliminated process had not been included in the initial solution. (b) At least one of the requirements' matching scores, whose initial value had been positive, would now be zero, implying that the eliminated process is the only possible match for that requirement. (c) At least one of the requirement's matching scores would have a positive value that is less than its initial value. If the best-fit OPD for this requirement is different than the initial one, then the eliminated process had been included in the initial solution, and the matching process has identified an alternative ERP process that matches the requirement to a lesser extent. Otherwise, the eliminated process had been a part of the solution and no alternative has been found.

This procedure was repeated five times, eliminating a different process each time.

4.1.2. Results

The results, presented in Table 2, demonstrate that the expected consistency was achieved. The table lists

Table 2
Results of Experiment 1

Eliminated process	Requirement	Before elimination		After elimination		Explanation
		OPD	Score	OPD	Score	
Lot management	Receipt and inspection	28	0.77	28	0.75	Lot management is a basic ERP functionality. It has no alternative.
	Receipt registering	30	0.55	30	0.39	
	Item object	117	0.56	117	0.5	Its elimination violated the lot object requirement, and reduced the match of other requirements
	Lot object	90	0.75		0	
Alternative supplier by item management	Alternative supplier management	66	1	113	0.7	The mechanism of alternative coding systems is identified as a substitute to the original match, although the score is lower
Supplier discounts management						The issue was not addressed in the requirements
Generate outbound data	Sub-contracting	52	0.58	53	0.58	The match was not reduced, since the alternative process, of manually inserting outbound data, received the same score as the original automatic one
Order history management	Purchase order line inserting	24	0.5	24	0.27	No alternative was found.
	Order history	72	0.7		0	

the eliminated processes, the affected requirements, and for each requirement its initial and final best-fit ERP model OPD and matching score. In one case (supplier discount management) the eliminated process was not included in the requirements, therefore no effect has been made. In two other cases (alternative supplier management and outbound data generation) the alignment process has identified alternative solutions that exist within the ERP system and satisfy the requirements, although in one case (alternative supplier management) the match is to a lesser extent than the original one.

The capability of the alignment process in identifying alternative solutions for the requirements supports our premise that ERP systems are basically more flexible than the ''best practice'' solutions they often promote, and that our approach can utilize this flexibility up to its limits.

4.2. Experiment 2

The aim of this experiment was to analyze the sensitivity of the solution to the algorithm weights, according to criterion 2. As discussed in Section 3, since RS $\leq$ ES, the higher the ES weight (W_1) is, the higher the MS is. The change in the score is linear with respect to the change in the weights. The interesting question here is whether different weights yield different solutions, that is, different system model OPDs are identified as best match for a given requirement.

The experiment repeatedly matched a set of 23 requirements against the system model, changing the weights in the matching scores.

The results are presented in Table 3, that shows the best-fit OPD for each requirement and its MS for different sets of weights. The table shows the linearity of MS with respect to the weights, where the slope depends on the difference between MS and RS. Requirement 5, for example, has a relatively high ES and a relatively low RS, therefore the change in weights affects the MS significantly. Requirements 18 and 21, on the other hand, have equal ES and RS, therefore the MS is invariant to the weights.

In most cases, despite of the changes in the MS, the best-fit OPD selected for the solution was not changed. However, in requirements 7 and 9, high weights of ES ($W_1 \geq 0.8$) resulted in a different best-fit OPD selection for the solution. Note, that high weights of RS did not distract the selected solution.

Table 3
Results of Experiment 2

Requirement	$W_1 = 0.9$, $W_2 = 0.1$		$W_1 = 0.8$, $W_2 = 0.2$		$W_1 = 0.7$, $W_2 = 0.3$		$W_1 = 0.6$, $W_2 = 0.4$		$W_1 = 0.5$, $W_2 = 0.5$		$W_1 = 0.4$, $W_2 = 0.6$		$W_1 = 0.3$, $W_2 = 0.7$		$W_1 = 0.2$, $W_2 = 0.8$		$W_1 = 0.2$, $W_2 = 0.8$	
	OPD	MS	OPD	MS	OPD	MS	OPD	MS	OPD	MS	OPD	MS	OPD	MS	OPD	MS	OPD	MS
1	1	0.36	1	0.33	1	0.29	1	0.26	1	0.22	1	0.19	1	0.15	1	0.12	1	0.09
2	8	0.96	8	0.93	8	0.9	8	0.86	8	0.83	8	0.8	8	0.76	8	0.73	8	0.7
4	52	0.8	52	0.74	52	0.69	52	0.64	52	0.58	52	0.53	52	0.47	52	0.42	52	0.36
5	28	0.87	28	0.85	28	0.82	28	0.8	28	0.77	28	0.75	28	0.72	28	0.7	28	0.67
6	30	0.91	30	0.82	30	0.73	30	0.64	30	0.55	30	0.46	30	0.37	30	0.28	30	0.2
7	39	0.78	39	0.72	40	0.65	40	0.58	40	0.51	40	0.45	40	0.38	40	0.31	40	0.24
8	36	0.77	36	0.7	36	0.64	36	0.58	36	0.51	36	0.45	36	0.39	36	0.32	36	0.26
9	30	0.7	30	0.63	117	0.59	117	0.58	117	0.56	117	0.55	117	0.54	117	0.52	117	0.51
10	76	0.97	76	0.96	76	0.95	76	0.94	76	0.93	76	0.92	76	0.91	76	0.9	76	0.89
12	98	0.79	98	0.79	98	0.78	98	0.78	98	0.77	98	0.77	98	0.76	98	0.76	98	0.75
13	94	0.91	94	0.83	94	0.75	94	0.66	94	0.58	94	0.5	94	0.41	94	0.33	94	0.25
14	60	0.96	60	0.92	60	0.88	60	0.85	60	0.81	60	0.77	60	0.73	60	0.7	60	0.66
16	67	0.98	67	0.96	67	0.94	67	0.92	67	0.9	67	0.88	67	0.86	67	0.84	67	0.82
17	30	0.34	30	0.32	30	0.3	30	0.28	30	0.26	30	0.24	30	0.22	30	0.2	30	0.18
18	24	0.5	24	0.5	24	0.5	24	0.5	24	0.5	24	0.5	24	0.5	24	0.5	24	0.5
19	72	0.74	72	0.73	72	0.72	72	0.71	72	0.7	72	0.7	72	0.69	72	0.68	72	0.67
20	10	0.95	10	0.9	10	0.85	10	0.8	10	0.75	10	0.7	10	0.65	10	0.6	10	0.55
21	66	1	66	1	66	1	66	1	66	1	66	1	66	1	66	1	66	1
22	61	0.85	61	0.85	61	0.85	61	0.84	61	0.84	61	0.84	61	0.84	61	0.83	61	0.83
23	90	0.95	90	0.9	90	0.85	90	0.8	90	0.75	90	0.7	90	0.65	90	0.6	90	0.55

It may, therefore, seem that it is preferable to use a high W_2. Nevertheless, the low matching scores obtained with a high W_2 are not necessarily an accurate reflection of the match. For example, requiring the process of *Inserting Purchase Order* to use the *Supplier* object is represented by an Instrument link between the object and the process. However, one of the attributes of *Supplier* in the ERP model, *Order Balance*, is affected when a purchase order is inserted, thus the link in the system model is an effect link. This difference, while reflected in the MS, does not violate the requirement in practice. Hence, applying a high RS weight might result in distorted matching scores.

Note, that sensitivity analysis could address the controlled changes in the input as performed in Experiment 1, that is, measuring the effect of the weights on the changes in MS when the input model is changed. The linearity of the matching scores makes it straightforward to see that this effect would be linear as well. Assume that for a given W_1 we got MS, which is the result of ES and LS. After changing the input model we got MS^1, based on $ES^1 = ES - \alpha$, and $RS^1 = RS - \beta$. Now changing W_1 to $W_1 + \delta$, would yield $MS^2 = MS^1 + \delta(\alpha - \beta)$, independently on the initial W_1.

4.3. Experiment 3

This experiment addressed the third and fourth validation criteria, validation against real-life results, and robustness to variations in the requirement models. Prior to the experiment, the ERP system had been implemented in a telecommunication company employing 150 workers. The requirements in this project were defined textually, without applying any formal or systematic RE process. We applied the alignment process using OPM representation of these enterprise requirements in a post-hoc manner, and compared the obtained outputs with the decisions made in the real life project. The OPM representation of the requirements was created by different modelers in parallel, and the effect of the variations among these representations on the solution was investigated.

A secondary objective of this experiment was to establish the effectiveness of each of the reformulation operations and to measure its effect separately. To this end, we performed the iterative process applying a single reformulation type in each iteration and measuring the effect on the matching scores.

4.3.1. Settings

The textual enterprise requirements were given to four different graduate students who are OPM experts and teach OPM in undergraduate courses. The modelers were instructed to represent the requirements as OPM models without adding any information other than what is in the text. They were also given a list of entities used in the system model, and instructed to select entity names from this list in their model as much as possible. The alignment process was then applied to each of these four models.

Since we wanted to isolate the effect of different reformulation operations, we did not follow the entire iterative alignment process. Instead, we applied three iterations. The first iteration used the initial requirement model, the second one applied only splitting of requirements, and the third one applied only new mapping decisions. We applied the same set of mapping decisions to the four requirement models. This ensured that any inconsistency in the results was caused by the initial modeling differences only, and provided a rough evaluation of the robustness of the matching to different modeling attitudes that may have been taken by different modelers.

The recommendations obtained by the resulting alignment were qualitatively compared with the decisions made in the project. This comparison enabled us to understand the differences between the results obtained when applying the four requirement models as well as their sources. The alignment recommendations that resulted from the study were validated through an interview with the project manager.

4.3.2. Results

The results of the experiment indicate that the alignment process is capable of providing a solution and identifying the gaps between the requirements and the system's capabilities, as compared to manual decisions. Table 4 presents a comparison between the recommendations made in the study regarding 13 significant gaps identified after the first two iterations and the actual decisions made in the real project. Over 75% of these gaps were resolved in the real project in a manner identical to that of the study. These include

Table 4
Summary of the comparison between Experiment 3 and the real project

Experiment recommendation	Project decision			
	Software customization made	Solution by mapping	Abandon requirement	Gap does not exist
Software customization	15%	8%	8%	
Solution by mapping		46%		
Abandon requirement			15%	
Identification difficulty				8%

customization decisions, mapping decisions, and abandoned requirements. The remaining 25% relate to requirements that were abandoned in the manual decision-making and to one unreal gap. In what follows we discuss the details of these findings with respect to our validation criteria.

4.3.2.1. Identification of gaps. All the gaps that had been identified in the real project, which resulted in software customizations, were also identified in the experiment. This indicates that the alignment process is capable of detecting gaps between the requirements and the system's capabilities.

4.3.2.2. Unreal gaps. One of the 13 gaps indicated by the automated matching was not real, and was caused by a situation, in which the details of a requirement are satisfied by a generic system solution, which has a customizable control mechanism. In such cases, the matching algorithm is not able to identify the match, since the structure of the requirement model is entirely different than that of the ERP model. To resolve this problem the requirement may either be remapped or verified manually.

4.3.2.3. Quality of solution. Comparing the recommendations made in the experiment with the real-life decisions we relate to mapping decisions and to abandoned requirements.

- *Mapping decisions*: 46% of the gaps identified in the first two iterations were resolved by mapping decisions in the third iteration. Out of these, 31% were identical to the mapping decisions made in the real life project, while the remaining 15% involved mappings to different entities. The mappings suggested in the experiment were presented to the project manager, who confirmed they were good solutions that could have been implemented in the project instead of the ones that were actually selected.
- *Recommendations to abandon requirements*: These reflect unresolved gaps. In the experiment, we recommended abandoning two requirements, where the ERP solutions were very close to the requirements, but not identical. The decisions in the real project matched our recommendation. In real life a number of additional requirements were abandoned, where we recommended customizations. The actual customization/abandoning decision involves other factors, such as cost and risk, which are out of the scope of our alignment process

4.3.2.4. Robustness to differences in the initial model. In order to assess this robustness, we performed the alignment applying four different models of the same set of requirements. Due to different modeling attitudes and insufficient domain knowledge of the modelers, the initial models were quite different from each other. Nevertheless, there was consistency in identifying the main gaps. In the second iteration, where the requirements were split without changing their content, the variance among the models increased, because the differences in modeling attitudes increased the affected number of requirements. A high level of consistency, reflected by low variance of the matching scores of the best-fit

Table 5
Average matching scores and variance of the four models in the three iterations

Iteration	Average matching score	Variance among models
Initial model	0.325	0.028
Splitting iteration	0.422	0.044
Mapping iteration	0.576	0.014

OPDs included in the solution, was achieved after applying the mapping decisions in the third iteration. This result indicates that the matching result is quite robust to differences in the initial model. The variance of the matching scores in the four models along the three iterations is given in Table 5.

Analysis of the inconsistencies among the results of the four different modelers indicated their main cause had been differences in modeling attitudes. The different attitudes reflect attempts to apply different design logic to the requirements, rather than to model them as they are. An example of a matching problem

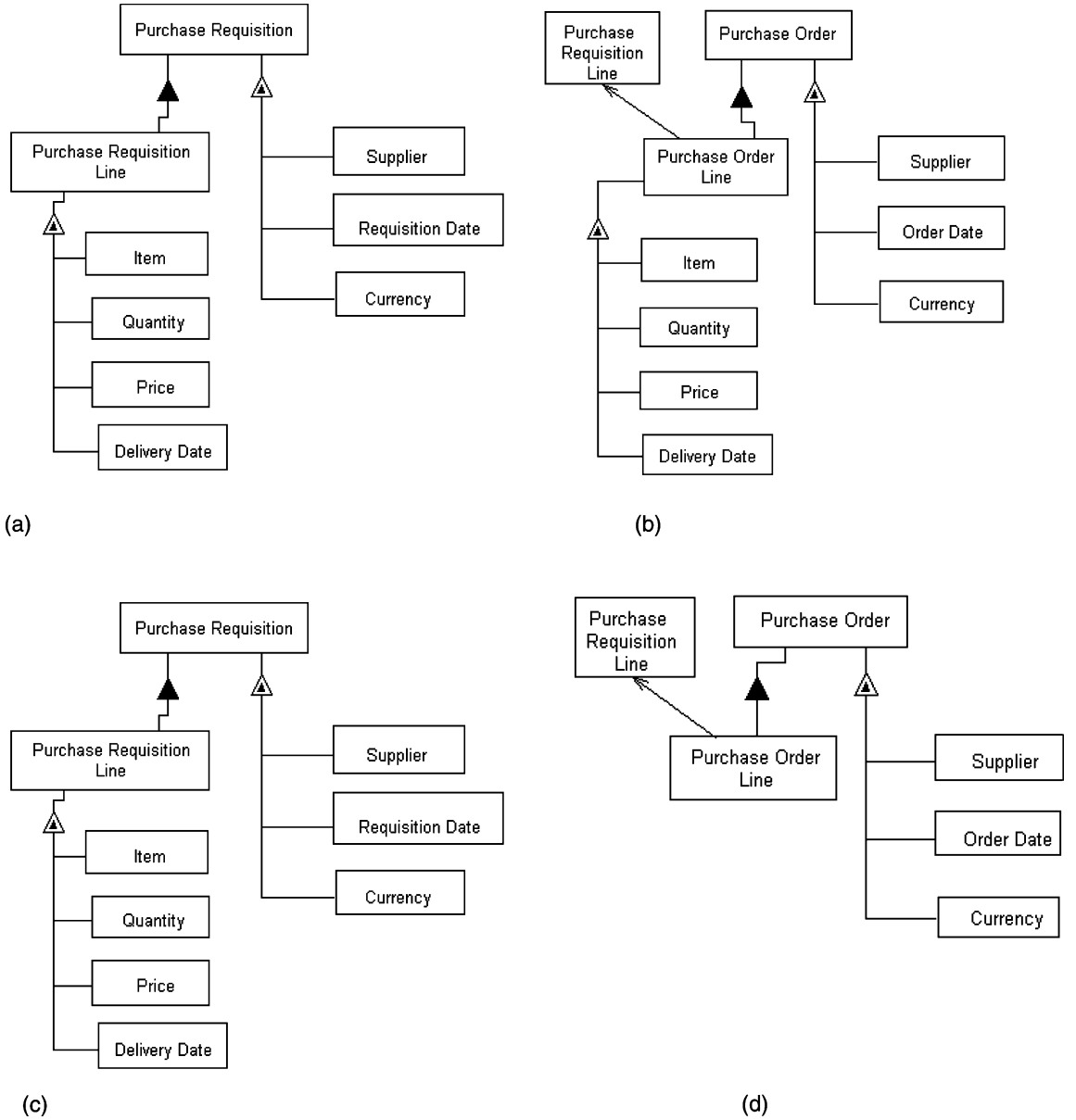

Fig. 8. Model-based matching problem. (a) Purchase requisition model; (b) purchase order model; (c) purchase requisition model; and (d) purchase order model.

that arises due to different modeling attitudes is illustrated in Fig. 8. Two separate requirements, specifying the objects *Purchase Requisition* and *Purchase Order*, are modeled in Fig. 8(a) and (b), respectively. One of the modelers assumed that the two objects should point at each other, and modeled them as illustrated in Fig. 8(c) and (d). This created dependency between the two requirements. Since the *Purchase Requisition* object does not exist in the system, the *Purchase Order* requirement could not be matched either. Based on these findings, we have designed modeling guidelines, which facilitate the matching procedure. A modeling guideline intended to prevent the problem illustrated in Fig. 8 is: ''Represent each requirement separately, including all the specified details. Do not rely on details specified in other requirements''. This guideline is expected to prevent the dependency illustrated in Fig. 8(c) and (d), since all the specified attributes of *Purchase Order* should be represented, the way they are in Fig. 8(b).

4.3.2.5. Other lessons learned. These include understanding the effect of the different reformulation operations and gaining indications as to the applicability of the approach in practice.

4.3.2.6. The effect of reformulation operations. The effect of splitting and mapping was isolated by applying them in different iterations in the experiment. The results indicate that both these reformulation types improved the matching scores, as reflected in the average matching scores achieved at each iteration, presented in Table 5.

4.3.2.7. Applicability evaluation. Representing the requirements in OPM took approximately 15 h. Applying the matching algorithm and analyzing the results took approximately 10 h for each iteration. These figures can hardly serve for estimating the duration of the alignment process in a real life project, since the study was carried out only on a single module of the ERP system, and served also for refining the methodology and the analysis method. Nevertheless, the manual decision making regarding this specific module took six weeks, and was performed by three full-time analysts and other stakeholders part time. It therefore seems that significant time saving can be achieved by applying our approach. The modelers who participated in the study noted that browsing in search for system terms to be used in the requirements model was inconvenient and this slowed the modeling process.

5. Concluding discussion

The problem of identifying and analyzing the gaps between an ERP system and the requirements of an enterprise, and aligning the system to the needs of the enterprise, is at the heart of ERP systems implementation. Adequate solution to this problem is crucial for a successful implementation and the competitive edge of the implementing enterprise.

The approach presented in this paper for solving this gap analysis and alignment problem provides a systematic support for the alignment process in both standard enterprises and unique ones. Unlike solution-driven methods, which apply a predefined set of reuse criteria for the process selection, our approach facilitates reuse on the basis of the enterprise requirements. This approach benefits from reuse without being restricted by a predefined set of criteria and standard solutions.

A requirement driven approach to the alignment problem, matching requirements model with an ERP model, is suggested also by [35,36]. However, the matching applied there is based on human reasoning while we apply an automated matching.

Matching between a model of the enterprise requirements and a model of the ERP system capabilities, the matching algorithm addresses two difficult problems. One is identifying a match between an incomplete requirements model and a complete and detailed system model. The other problem is the existence of dependencies among the ERP options, which poses constraints on the feasibility of combining all the satisfied requirements in a single configuration of the system. While our matching algorithm is designed for OPM models, the two above problems are generic, inherited in the nature of the alignment process, and should be addressed regardless of the modeling methodology.

The main limitations of our approach are the use of OPM, which is not a common standard in industry, and the need to create a system model as a preparation to

applying the approach these limitations can also be viewed as strengths. The OPM representation, which requires a set up effort in constructing a system model, has the advantage of a high expressive power while preserving an ease of matching due to the single view taken. The set up effort, though intensive, is a one-time effort and once the model is constructed it can be used for any number of times. The OPM representation, as opposed to representation methods that exist in specific packages, makes our approach generic and suitable for any ERP system rather than a specific package.

Two difficulties have been identified in the matching algorithm. One is that the scope of a process and refinement into lower-level diagram do not have definite rules, and are determined by the modeler based on individual judgment. It is therefore not necessarily consistent with the system model. This difficulty can be resolved by splitting the requirements to their basic ingredients. The second difficulty is in identifying the semantic similarity of entity names. As discussed in Section 3, the matching algorithm does not apply affinity measurement due to the high level of human reasoning required for mapping. While being resolvable through mapping iterations in our alignment process, our observations in Experiment 3 show that modelers find browsing for entity names tiresome and time consuming. A possible solution is an interactive entity naming, where, while matching the entities the algorithm would present the user possible entity names in the system for approval.

We have tested the alignment algorithm in three experiments that complement each other in assessing the feasibility of the approach. The third experiment reported here was closest to simulating the process in a real-life situation, and its results demonstrate the ability of the process to provide an adequate solution to the problem. Still, being a single case study it can be considered as a demonstration rather than full validation. However, in combination with the other two experiments, we were able to show that different input data would affect the solution in a consistent and predictable direction, and that the selection of the algorithm weights within a reasonable range would not change the solution obtained by the process. We also showed that the solution is quite robust to variations in the requirements model. Combining these findings leads to the conclusion that the viability of the approach has been shown.

Practical implications, such as the expected implementation time saving in a real life project, are hard to estimate at this point. However, the results of the third experiment indicate that compared to a manual alignment, which is currently performed when standard solutions are not applicable, significant time saving as well as a high quality solution can be achieved.

A large-scale study or an action research, in which the alignment process would be applied in a real-life project may yield better indications. Before such a study can be conducted, several improvements should be made to the support tool, especially regarding the semantic similarity assessment. We also intend to develop a mechanism to support the creation of new mapping decisions, in which ‘‘mapping opportunities’’ will be identified through generalization hierarchies.

References

[1] M. Al-Mashari, Process orientation through enterprise resource planning (ERP): a review of critical issues, Knowledge and Process Management 175 (2001) 185.
[2] P. Bingi, M. Sharma, J. Godla, Critical issues affecting an ERP implementation, Information Systems Management 7 (1999) 14.
[3] H. Bunke, Error correcting graph matching: on the influence of the underlying cost function, IEEE Transactions on Pattern Analysis and Machine Intelligence 917 (1999) 922.
[4] T.A. Curran, A. Ladd, SAP R/3 Business Blueprint: Understanding Enterprise Supply Chain Management, second ed., Prentice Hall, NJ, 1999.
[5] M. Daneva, Measuring reuse in SAP requirements: a model-based approach, in: SSR’99, Proceedings of the Fifth Symposium on Software Reusability, ACM Press, New York, 1999, pp. 141–150.
[6] M. Daneva, Using maturity assessment to understand the ERP requirements engineering process, in: Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE’02), IEEE Press, Los Alamitos, CA, 2002.
[7] T.H. Davenport, Putting the enterprise into the enterprise system, Harvard Business Review 121 (1998) 131.
[8] D. Dori, Arc segmentation in the machine drawing understanding environment, IEEE Transactions on Pattern Analysis and Machine Intelligence 1057 (1995) 1068.
[9] D. Dori, Object-process analysis of computer integrated manufacturing documentation and inspection functions, International Journal of Computer Integrated Manufacturing 339 (1996) 353.
[10] D. Dori, Analysis and representation of the image understanding environment using the object-process methodology, Journal of Object-Oriented Programming 30 (1996) 8.
[11] D. Dori, Object-process methodology applied to modeling credit card transactions, Journal of Database Management 4 (2001) 14.

[12] D. Dori, Object Process Methodology—A Holistic Systems Paradigm, Springer Verlag, Heidelberg, New York, 2002.
[13] Dori D, Reinhartz-Berger I, Sturm A. OPCAT—a bimodal CASE tool for object-process based system development. In: Proceedings of the Fifth International Conference on Enterprise Information Systems, vol. 3, 2003, pp. 286–291.
[14] M.D. Feblowitz, S.J. Greenspan, Scenario-based analysis of COTS acquisition impacts, Requirements Engineering 182 (1998) 201.
[15] Finkelstein ACW, G. Spanoudakis, M. Ryan, Software package requirements and procurement, in: Proceedings of the 8th International Workshop on Software Specification and Design, IEEE Press, Los Alamitos, CA, 1996, pp. 141–145.
[16] J. Ghosh, SAP Project Management, McGraw-Hill, New York, 2000.
[17] N. Gibson, C.P. Holland, B. Light, Enterprise resource planning: a business approach to systems development, in: Proceedings of the 32nd Hawaii International Conference on System Sciences, 1999.
[18] H. Gomaa, L. Kerchberg, V. Sugumaran, C. Bosch, I. Tavakoli, L. O'Hara, A knowledge-based software engineering environment for reusable software requirements and architectures, Automated Software Engineering 285 (1996) 307.
[19] R.P. Hall, Computational approaches to analogical reasoning: a comparative analysis, Artificial Intelligence 39 (1989) 120.
[20] M. Hammer, J. Champy, Reengineering the Corporation, Harper Collins, New York, 1993.
[21] C.P. Holland, B.A. Light, Critical success factors model for ERP implementations, IEEE Software 30 (1999) 5.
[22] C. Koch, BPR and ERP: realising a vision of process with IT, Business Process Management Journal 258 (2001) 265.
[23] N.A.M. Maiden, C. Ncube, Acquiring COTS software selection requirements, IEEE Software 46 (1998) 56.
[24] P. Massonet, A.V. Lamsweerde, Analogical reuse of requirements frameworks, in: RE'97, Proceedings of the Third IEEE Symposium on Requirements Engineering, IEEE Press, Los Alamitos, CA, 1997, pp. 26–37.
[25] D. Meyersdorf, D. Dori, The R&D universe and its feedback cycles: an object-process analysis, R&D Management 333 (1997) 344.
[26] H. Mili, F. Mili, A. Mili, Reusing software: issues and research directions, IEEE Transactions on Software Engineering 528 (1995) 561.
[27] C. Ncube, N.A.M. Maiden, Guiding parallel requirements acquisition and COTS software selection, in: RE'99: Proceedings of the IEEE International Symposium on Requirements Engineering, IEEE Press, Los Alamitos, CA, 1999, pp. 133–140.
[28] E. Ostertag, J. Hendler, R. Prieto Diaz, R.C. Braun, Computing similarity in a reuse library system: an AI-based approach, ACM Transactions on Software Engineering and Methodology 205 (1992) 228.
[29] M. Peleg, D. Dori, The model multiplicity problem: experimenting with real-time specification methods, IEEE Transactions on Software Engineering 742 (2000) 759.
[30] H.A. Post, R. Van Es (Eds.), Dynamic Enterprise Modeling: A Paradigm Shift in Software Implementation, Kluwer, Dordrecht, 1996.
[31] E. Rahm, P.A. Bernstein, A survey of approaches to automatic schema matching, The VLDB Journal 334 (2001) 350.
[32] J. Ralyte, C. Rolland, An assembly process model for method engineering, in: Proceedings of the 13th International Conference on Advanced Information Systems Engineering (LNCS 2068), Springer-Verlag, Berlin, 2001, pp. 267–283.
[33] C. Rolland, Requirements engineering for COTS based systems, Information and Software Technology 985 (1999) 990.
[34] C. Rolland, Intention Driven Component Reuse. Information Systems Engineering: State of the Art and Research Themes, Springer-Verlag, Berlin, 2000, pp. 197–208.
[35] C. Rolland, N. Prakash, Bridging the gap between organizational needs and ERP functionality, Requirements Engineering 180 (2000) 193.
[36] C. Rolland, N. Prakash, Matching ERP system functionality to customer requirements, in: Proceedings Fifth IEEE International Symposium on Requirements Engineering, IEEE Press, Los Alamitos, CA, 2001, pp. 66–75.
[37] C. Salinesi, C. Rolland, Fitting business models to system functionality exploring the fitness relationship, in: Proceedings of the 15th International Conference on Advanced Information Systems Engineering (LNCS 2681), Springer-Verlag, Berlin, 2003, pp. 647–664.
[38] P. Soffer, Aligning an enterprise system with enterprise requirements: an iterative process, in: Proceedings of the Fifth International Conference on Enterprise Information Systems, vol. 3, 2003, pp. 147–155.
[39] P. Soffer, Refinement equivalence in model-based reuse: overcoming differences in abstraction level, Journal of Database Management 16 (3) (2005) 21–39.
[40] P. Soffer, B. Golany, D. Dori, ERP modeling: a comprehensive approach, Information Systems 673 (2003) 690.
[41] P. Soffer, B. Golany, D. Dori, Y. Wand, Modelling off-the-shelf information systems requirements: an ontological approach, Requirements Engineering 183 (2001) 198.
[42] A. Sutcliffe, N.A. Maiden, The domain theory for requirements engineering, IEEE Transactions on Software Engineering 174 (1998) 196.
[43] R. Van Es, Dynamic Enterprise Innovation, Baan Business Innovation B.V., The Netherlands, 1998.
[44] L. Wenyin, D. Dori, Object-process diagrams as an explicit algorithm specification tool, Journal of Object-Oriented Programming 52 (1998) 9.

Pnina Soffer is a lecturer at the Management Information Systems in Haifa University, Israel. She has a BSc (1991) and MSc (1993) in Industrial Engineering and a PhD (2002) in Industrial Engineering and Information Systems from the Technion – Israel Institute of Technology. Her research areas are requirements engineering (specifically: requirements modeling, goal analysis, requirements for process-supportive information systems), off-the-shelf information systems (specifically, ERP systems requirements, alignment, and customization) and business process mod-

eling and design (specifically: development of a formal goal-oriented modeling approach).

Boaz Golany is a Professor at the Industrial Engineering and Management Faculty and the holder of Samuel Gorni Chair in Engineering in the Technion – Israel Institute of Technology. He has a BSc in Industrial Engineering and Management from the Technion (1982), and a PhD from the Business School of the University of Texas at Austin (1985). He was awarded the Naor Prize of the Israeli Operations Research Society in 1982 and the Yigal Alon Fellowship from the Israeli Education Ministry in 1986. In 1991 he was a recipient of the Technion Academic Excellence Award and in 1994 he became a Senior Research Fellow of the IC^2 Institute at the University of Texas at Austin. He has served as the Associate Dean of the IE&M faculty in 1994–1999. He has also served as an Area Editor and member of the editorial Board for the *Journal of Productivity Analysis, IIE Transactions* and *Omega*. Prof. Golany has published over 80 papers in academic journals and books. His publications are in the areas of Industrial Engineering, Operations Research and Management Science. He has been active in various professional societies including The Institute of Management Science (in the US) and the Israeli Operations Research Society (where he served as the Treasurer in 1989–1992 and as the vice-President 1996–2002).

Dov Dori is Head of the Information Systems Engineering Area at the Faculty of Industrial Engineering and Management, Technion, Israel Institute of Technology, Research Affiliate at MIT, and CEO of OPCAT, Inc. Between 1999–2001 he was Visiting Faculty at MIT's Engineering Systems Division and Sloan School of Management. Professor Dori received his BSc in Industrial Engineering and Management from the Technion in 1975, MSc in Operations Research from Tel Aviv University in 1981, and PhD in Computer Science from Weizmann Institute of Science, Israel, in 1988. His research interests include Systems Development Methodologies, Information Systems Engineering, Computer-Aided Software Engineering and Document Analysis and Recognition. Between 1999–2001 Prof. Dori was Associate Editor of IEEE Transaction on Pattern Analysis and Machine Intelligence (T-PAMI). He is author/co-editor of four books and author of over 130 publications. He is Fellow of the International Association for Pattern Recognition (IAPR) and Senior Member of IEEE. He has been consultant and invited lecturer for companies, including Pratt and Whitney, Ford, FAA, NASA, Kodak, and NIST.