PERGAMON

Information Systems 28 (2003) 673–690

www.elsevier.com/locate/infosys

ERP modeling: a comprehensive approach

Pnina Soffer, Boaz Golany*, Dov Dori

Faculty of Industrial Engineering and Management, Technion-Israel Institute of Technology, Technion city, Haifa 32000, Israel

Received 20 February 2002; accepted 18 October 2002

Abstract

We present a generic reverse engineering process, aimed at developing a model that captures the available alternatives at different application levels of an Enterprise Resource Planning (ERP) system. Such a model is needed when ERP systems are aligned with the needs of the enterprise in which they are implemented. In order to support the ERP implementation process, the model should describe the entire scope of the ERP system's functionality and the alternative business processes it supports, as well as the interdependencies among them. We analyze the desired properties a modeling language should satisfy to be applied in constructing an ERP system model. This analysis, which follows the Cooperative Requirements Engineering With Scenarios classification framework in its adapted ERP modeling form, results in a set of criteria for evaluating modeling languages for this purpose. Using these criteria, we evaluate the Object–Process modeling Methodology and apply it for generating a detailed ERP system model. The generic process and detailed criteria we develop can serve for comprehensive ERP modeling, as well as for obtaining a model of other process-supportive off-the-shelf systems that are of generic and configurable nature.

Keywords: Enterprise resource planning; Reverse engineering; Object–process methodology; Modeling languages

1. Introduction

Enterprise Resource Planning (ERP) systems are off-the-shelf software packages that support most of the key functions of an enterprise, such as logistics, sales, and financial management. These systems are generic, and the functionality they provide can serve a large variety of enterprises.

The implementation of an ERP system involves a process of customizing the generic package and aligning it with the specific needs of the enterprise. The alignment process simultaneously defines the software configuration and the enterprise business solution. Due to the need to adapt the enterprise to the software package rather than the other way around, it often results in redesigned business processes [1,2]. Enhancing the system's functionality through software customizations, although not desired [3], is sometimes required. The business process alignment, affected by various environmental aspects, such as existing information systems prior to the ERP implementation [2] and organizational culture [4], bears crucial consequences on the success of the ERP implementation project and on the future business practice of the enterprise [5].

*Corresponding author. Tel.: +972-4-8294512; fax: +972-4-832-5194.

E-mail addresses: pnina@tx.technion.ac.il (P. Soffer), golany@ie.technion.ac.il (B. Golany), dori@ie.technion.ac.il (D. Dori).

0306-4379/03/$ - see front matter
PII: S0306-4379(02)00078-9

This work is motivated by the need to provide support for the alignment process in enterprises whose business processes are unique and form their competitive edge. Such enterprises do not necessarily wish to standardize all their processes due to ERP implementation, as is often the case. Common tools that support the alignment process refer to predefined "best practice" models and configurations [6–8], and therefore do not support the needs of these enterprises. Preserving their unique processes may require these enterprises to make software customizations in the ERP system, and take risks of a long implementation time and high costs for maintenance and upgrades in the future [3]. However, the functionality of ERP systems is in many cases rich enough and capable of supporting business processes that are not included in the "best practice" solutions.

A requirement-driven alignment approach [9–11] enables the enterprise to identify the ERP configuration that satisfies stated requirements that do not necessarily match any predefined "best practice" solution, and the gaps—the requirements that are not satisfied by the system.

Such an approach matches specifications of the enterprise requirements with a model, specifying the ERP system capabilities. This requires the model of the ERP system capabilities to represent the entire scope of options available in the ERP system in a manner that enables matching with the enterprise requirements. A common basis of semantics and representation is needed for both the ERP system model and the enterprise requirements.

The enterprise requirements, discussed in detail in [11], are obtained through a requirements engineering process and represented formally in order to be matched with the ERP system capabilities. Some ERP systems, such as SAP and Baan, have embedded enterprise models, which represent their capabilities and "best practice" solutions [6,8]. The model embedded in an ERP system may serve as a basis for matching the system with the requirements. Thus, it makes sense to represent the requirements using the same modeling approach. However, some ERP systems have no embedded business model, and others apply different modeling conventions, addressing different aspects of the enterprise. The Architecture of Integrated Information Systems (ARIS) [6,12], applied in the reference models of SAP, incorporates five representation views. These include a function view, decomposing activities in a top-down manner, a business process view, represented as Event-driven Process Chains (EPC), a resource view, representing organizational units and other resources, a data view represented by Entity-Relationship Diagrams (ERDs), and an output view, representing physical inputs and outputs. The Dynamic Enterprise Modeling (DEM) [8], applied in the Baan reference models, incorporates a business control view, which represents business functions, their structure and interaction, an organizational structure view, an enterprise structure view, showing geographically distributed inner supply chain, and a business process view, represented as Petri Nets [8].

Due to the lack of a standard modeling language, the enterprise requirements must be represented differently when implementing different ERP systems in order to be matched with the ERP system capabilities. Furthermore, in order to be matched, the issues addressed by the enterprise requirements must correspond to the issues addressed by the modeling method applied in the package, which is being implemented. This does not make much sense, since the issues addressed by the enterprise requirements are independent of the specific ERP package implemented.

Rather than relying on the models embedded in the various ERP packages, our goal was to establish a proper ERP model and determine what modeling language is most appropriate for representing the ERP system capabilities to be matched with the enterprise requirements. For this purpose the desired model should represent business processes and their corresponding underlying information objects [11], and capture the entire scope of alternative options provided by the ERP system and their interdependencies.

Constructing an abstract model of an existing system is a process of reverse engineering. Reverse engineering processes are generally composed of three main activities: restructuring, comprehension, and production of an abstract model of the system under investigation [13]. Restructuring,

which is the process of transforming the structure of a system without changing the level of abstraction, is applied when error correcting or recovery of structure and consistency is needed, which is not our case. Existing reverse engineering approaches (e.g., [14,15]) that provide methodologies for obtaining a model from an information system, typically address legacy systems, and therefore do not focus on capturing optionality. Other works, addressing complex and configurable systems [16] and product families [17], deal with the system's architecture in support of its evolution, rather than its possible configuration options. Providing a reverse engineering focus on optionality, which is essential for ERP modeling, is one of the contributions of the present work.

We start by outlining the steps that enable systematic construction of a model that captures the entire scope of alternative ERP system options and the interdependencies among them. These steps are generic and not specified for any particular modeling language. They provide a basis for discussing the nature of modeling languages suitable for our purpose. The discussion is based on the Cooperative Requirements Engineering With Scenarios (CREWS) classification framework of modeling languages [18], which was adapted for ERP representation in [10], and results in a set of evaluation criteria for a modeling language to be applied in ERP modeling. The modeling steps are refined into a modeling procedure that applies the Object–Process Methodology (OPM) [19] as a modeling language. OPM has been introduced and evaluated in [11] for modeling the requirements posed by an enterprise. Therefore, applying it for representing the ERP system provides a basis for aligning the system with the requirements. The OPM modeling procedure can be applied as is, or serve as a meta-model for constructing a modeling procedure, which is implemented with any other modeling language that fits the defined selection criteria.

The remainder of the paper is organized as follows: Section 2 outlines the generic ERP modeling procedure. The desired characteristics of a modeling language that is useful for describing an ERP model and implementing the modeling procedure are discussed in Section 3 and form selection criteria for such a language. Section 4 briefly introduces the OPM and presents an ERP modeling procedure that follows the steps outlined in Section 2. Discussion and concluding remarks are given in Section 5.

2. Generic ERP modeling steps

The alternative options provided by an ERP system appear in different levels of application. In this section, we discuss the optionality levels, and present a generic reverse engineering approach, which is based on their structure. We illustrate our approach with examples based on the Purchase and Inventory modules of the Baan IV ERP system, which concern purchasing of goods and inventory transactions in warehouses.

2.1. ERP optionality levels

Capturing the entire scope of options supported by the ERP system in a model is a challenging task, which is essential for identifying a good solution for the enterprise needs. The ERP options enable adapting the system to various enterprises with different needs, and are controlled by a set of parameters, whose values are determined through an alignment process.

Three levels of optionality are distinguished: System Configuration level, Object level, and Occurrence level.

The System Configuration optionality level affects the functionality of the ERP system throughout the entire system. The various options are controlled by the system parameters, whose values are set once when the system is implemented in an enterprise and remain static thereafter. An example of such a parameter is a Boolean parameter indicating whether warehouse location control is implemented in the system. By setting the parameter value to True, all the warehouse location-related processes, such as warehouse locations management and receiving purchased goods to locations, are enabled, while a False value of this parameter disables this functionality.

There are three types of system parameters. (1) High-level process definitions, which control the

system functionality by providing preconditions to user-interface sessions. The location control parameter is of this type. (2) Low-level process definitions that control the system's internal functionality by indicating the specific algorithms and rules to be applied when performing a specific action. For example, a parameter of this type defines what types of action require logging a record of change history by the system. (3) Process parameter definitions that provide values that determine how a specific action is performed. Examples of this parameter type are parameters defining the numerical range and structure in order numbering, and parameters indicating different planning horizons.

The Object level is an intermediate optionality level, controlled by the parameters of single data objects, and can vary in different instances. Different parameter values cause different instances to be handled in a different manner. An example of such a parameter is a Boolean attribute of a warehouse indicating whether location is controlled in this specific warehouse. When location control is implemented in the system, each warehouse may either be location controlled or not, as set by the value of this attribute. A warehouse cannot be location controlled if location control is not implemented in the system, but in a configuration including location control, there may still be one or more warehouses that are managed without location control. Since this is an attribute of a Warehouse, its True and False values enable performing inventory handling processes differently in different warehouses. Another example is a Boolean attribute of an item, indicating whether or not it should undergo a quality inspection when received and be approved before it can be used.

The Occurrence level, which is the lowest level of optionality, applies to a single occurrence of a process, and facilitates variance in its performance. For example, when registering the receipt of purchased items that need to be inspected, the user may select an option of a fast approval, and automatically approve all the delivered goods as being of good quality. Otherwise, the goods will undergo a quality inspection as a separate process. The user can decide whether to apply this option in each occurrence of the goods receiving process.

2.2. Systematic ERP modeling

The generic reverse engineering process presented here is focused on two of the three fundamental reverse engineering activities: achieving comprehension of the ERP system and its optionality controls, and abstracting it to a model. The sources of information for the process are the system database schema, as reflected in the system tables, the system on-line and off-line documentation and help facilities, the user interface, and domain knowledge of the modeler. If the system includes an embedded process model, such as SAP [6] and Baan [8], that model can serve as a basis for the business process model, but should be treated carefully and checked for consistency with the other information sources.

The process, whose summary is provided in Table 1, includes four modeling levels: a global level and three levels corresponding to the optionality levels discussed above.

Global level: The initial level in the modeling process concerns the global functionality of the system without addressing its various options. First, the ERP system's database schema is represented and abstracted in order to gain understanding of the business objects that participate in the supported business processes. The database model is obtained in a bottom-up manner, first by extracting the physical data structure and second by establishing generalized objects. The business processes are identified by grouping the user-interface sessions in the ERP modules into functional groups, and by following the order in which transformations of the status attributes of the objects may occur. In cases where the ERP system includes an embedded process model, this model can serve as a basis for the Global model. This top-level model specifies the sequence of activities that manipulate the main data objects in the system but it does not represent alternative processes.

System configuration level: The second step, which addresses the System Configuration level, concerns the alternative business processes that are

Table 1
Generic ERP modeling process

Step	Description	Information sources	Output
ERP global level	1. Constructing a data model in a bottom-up manner 2. Constructing a Global business process model	• ERP database • ERP system modules and sessions • Embedded business process model • Status transformation of objects	1. Database model 2. Global business process model
System configuration level	Identifying and modeling the alternative business processes, controlled by the system parameters	• ERP database model: system parameters • ERP system modules and sessions	System configuration-level business process model
Object level	Identifying and modeling business process alternatives and options, controlled by data object attributes	• ERP database model: system parameters and object attributes • ERP system modules and sessions	Object-level business process model
Occurrence level	Identifying and modeling activity options, controlled by user-defined options	• ERP database model: system parameters and object attributes • ERP system modules and sessions: user interface	Occurrence-level business process model

determined by the system configuration. This entails identifying the effect of the system parameters, which are included in the database model and belong to the high-level process definition set of parameters. These parameters determine the main alternatives for performing the business processes by controlling the access to user-interface sessions. Identifying their effect on the activities of the Global process model leads to refined business process alternatives in the System Configuration-level business process model.

Object level: The third step addresses the Object level, representing process variants that hold for a single instance. It involves studying the activities of the business processes and the data objects they manipulate in order to identify (a) the possible variants of the process activities, and (b) the preconditions of each such variant in terms of data object attribute values. The preconditions may be a combination of system parameters and data object attribute values. The instance-specific process variants, along with their preconditions, are represented at the Object-level business process model.

Occurrence level: The fourth and last step, addressing the Occurrence level, considers the options the user may select in each occurrence of a process, thus constructing the most detailed layer of the model. This layer expresses the alternative options available for performing each process variant. The enterprise operations can be controlled through a combination of higher-level parameters of the first three layers with low-level process occurrence parameters. In this step the user interface is studied in order to identify options that the user can determine while performing a task. The effect of these options on the business processes is defined and modeled.

The layers of the model obtained through the process steps are illustrated by the Purchase and Inventory example, in which the database model includes objects such as *Warehouse, Item, Supplier, Location, and Purchase order*, along with their relations and attributes, as well as system

parameters, such as *Location Control Implemented*. The Global business process model specifies processes such as receiving purchased goods.

In the System Configuration level, two alternative ways of receiving goods to a warehouse are identified and modeled. These are controlled by the system parameter *Location Control Implemented,* whose *False* value means that the alternative of receiving goods into a non-location controlled warehouse is selected, while a *True* value allows both alternatives, one of receiving goods into a location controlled and another—to a non-location controlled warehouse.

Two examples illustrate the Object level. In the first one the combination of the system parameter *Location Control Implemented* and the *warehouse* attribute *Location Controlled* serves as a precondition for location controlled or non-location controlled receiving. In the second example the *Item* attribute *Inspection Required* controls by itself the activation of an inspection sub-process, included as a variant in the purchase receiving process.

The Occurrence level is illustrated by the option *Fast Approval*, which the user may select when dealing with items that require inspection. This option activates an activity of automatically approving all the goods received.

The Purchase and Inventory example demonstrates the modeling steps, based on the parameters that control the ERP functionality at the different application levels. As the number of these parameters increases through the entire software package, so does the number of process alternatives and options.

The four modeling levels are independent of the modeling language applied. Nevertheless, in order to be applicable for the proposed modeling process and to properly express the system's rich and complex functionality, the modeling language should meet certain requirements. In the following section we discuss the properties of modeling languages suitable for ERP modeling.

3. Desired properties of an ERP modeling language

In this section we discuss the properties of an effective ERP modeling language. The discussion follows the CREWS classification framework of scenario representations [18], which was adapted by Rolland and Prakash [10] to classification of ERP modeling approaches. We make some modifications to their framework, whose summary is given in the Appendix A. The extended framework, which enumerates the properties of modeling languages that are of relevance to ERP representation, is the basis of our discussion. We consider the consequences of each of the properties, and indicate its desired values. Thus, we in fact transformed the adapted CREWS classification framework into an evaluation framework of modeling languages.

The adapted CREWS classifies modeling methods by four views: *Content*, *Form*, *Purpose*, and *Customization process*. Each view has a set of associated facets, and each facet is characterized by a set of relevant attributes.

The *Content* view refers to the knowledge captured in the model, defined according to four facets: Abstraction, Context, Coverage, and Argumentation.

Abstraction: The Abstraction facet is characterized by a single attribute, Abstraction, which classifies the scope of abstraction levels available in a modeling language. This scope can be Flexible, allowing different levels of abstraction in a model; Fixed, indicating that once the abstraction level is set in a model, it is not possible to change it into a higher abstraction level or a lower one; or the scope may be unique, allowing a predefined abstraction level only (e.g., details only).

For the purpose of ERP modeling Flexible abstraction is required of the modeling language, in order to support a top-down/bottom-up modeling process. Furthermore, due to the high level of complexity and broad scope of ERP systems, a fixed level of abstraction (unique or even changeable) may either provide insufficient information or become overloaded.

Context: The Context is characterized by three Boolean attributes: Internal, Interaction, and Contextual. The Internal attribute indicates whether the internal system functionality is addressed in the obtained model. The Interaction attribute indicates whether the interaction between the system and its environment is represented. The

Contextual attribute indicates whether the model represents the environmental context in which the system operates.

Since the purpose of ERP modeling is to align the ERP system with the enterprise requirements, the contextual and interaction properties are of relevance. As discussed in [11], the requirements treat the system as a black box, specifying its organizational context and partly its interaction. In order to address the same issues as the requirements, the ERP model should focus on the context of the system and its interaction.

Coverage: The Coverage facet is expressed by two attributes: Functional, which characterizes the functional information captured in the model, and Intentional, whose Boolean value indicates whether the model includes intentional information, such as objectives and goals.

While the ERP-adapted CREWS framework [10] treats the Functional attribute as a Boolean one, we chose to follow the original CREWS framework, and distinguish between structural, functional and behavioral information captured by the modeling languages. This attribute, therefore, indicates the types of functional information captured in the model. Proper representation of the ERP functionality requires the modeling language to express all three types: structural, functional and behavioral information. Intentional information is not an essential part of the ERP model, as presented in Section 2.

Argumentation: The Argumentation facet, relating to the ability of a modeling language to express optionality-related information, is expressed by three attributes: Variant, Arguments, and Variant Dependency.

The Variant attribute, indicating whether the modeling language is capable of expressing options and variants, is essential for capturing the entire scope of options and business process variants supported by the ERP system. The Arguments attribute identifies whether the modeling method provides for including arguments that support the selection. Such arguments, although possibly helpful in the alignment process, are not mandatory.

While the third attribute, the Variant Dependency, was not included in the adapted CREWS framework, we consider it as an important element of the framework. Dependencies and inter-relations play a major role in the customization decision-making process, and should therefore be reflected in the model resulting from the modeling language under consideration.

Table 2 summarizes the facets and attributes of the Content view and indicates the required attribute values of an adequate ERP modeling language. If no required value is specified in the table, every value is acceptable.

The *Form* view refers to the notation and structure applied in the modeling language, through two facets: Notation and Structure.

Notation: The Notation facet has a single attribute, Notation, which classifies the formality level of a modeling language. It should preferably be either formal or semi-formal in order to support a systematic matching with the requirements, or else it may lead to inaccuracy and ambiguity in the alignment process.

Structure: The Structure facet, addressing the types of elements and relations in a modeling language, has three attributes: Element Type, which is merely an identification of the element types included in the model, Intra-element Relationship, and Inter-element Relationship.

The Intra-element Relationship attribute indicates whether the elements of the model constitute a nested structure or a flat one. This attribute is closely related to the Abstraction attribute of the Content view, since a nested structure typically corresponds to a flexible level of abstraction, while a flat structure typically indicates a single level of abstraction. Since a flexible level of abstraction is desired in ERP modeling, the Intra-element relationship should be nested, or a combination of nested and flat structures.

The Inter-element Relationship attribute, which specifies the relationship types in the model, determines to a great extent the expressive power of a modeling language, which should be as high as possible. Nevertheless, relating this attribute to the Intra-element Relationship, a nested intra-element structure provides for generalization and refinement relationships to be reflected by the abstraction levels of the model. Therefore, within a single level of the model, the lack of generalization and refinement is acceptable.

Table 2
Required values for attributes of the Content View

Facet	Attribute	Explanation	Required values
Abstraction	Abstraction: ENUM {Unique, Fixed, Flexible}	The scope of abstraction levels supported	Flexible
Context	Internal: BOOLEAN	The model represents internal system functionality	
	Interaction: BOOLEAN	The model represents interaction between system and environment	True
	Contextual: BOOLEAN	The model represents the organizational environment	True
Coverage	Functional: SET (ENUM {Structure, Function, Behavior})	Notes the aspects of the ERP system captured in the model	{Structure, Function, Behavior}
	Intentional: BOOLEAN	The model provides intentional information	
Argumentation	Variant: BOOLEAN	Ability to represent options and variants	True
	Arguments: BOOLEAN	Argumentation guiding the variant selection	
	Variant dependency: BOOLEAN	Ability to represent dependencies among the variants	True

Table 3
Required values for attributes of the Structure View

Facet	Attribute	Explanation	Required values
Notation	Notation: ENUM {Formal, Semi-formal, Informal}	The notation formality level	Formal, Semi-formal
Structure	Element type: SET	The main element types in the model structure	
	Intra-element relationship: SET (ENUM {Flat, Nested})	Type of structure constituted by the elements	Nested
	Inter-element Relationship: SET (ENUM {Composition, Generalization, Precedence, AND, OR, XOR, Refinement, Characterization})	Possible relationship types between elements	{Composition, Precedence, AND, OR, XOR, Characterization}

Table 3 summarizes the facets and attributes of the Structure view and indicates the required attribute values of a modeling language adequate for ERP modeling.

The *Purpose* view refers to the objectives that may be satisfied by using the model. It has one facet, the Aim of Representation, expressed by three Boolean Attributes: Descriptive,

Table 4
Required values for attributes of the Purpose View

Facet	Attribute	Explanation	Required values
Aim of representation	Descriptive: BOOLEAN	The model describes one way of achieving a goal	
	Exploratory: BOOLEAN	The model describes several possible different ways of achieving a goal	True
	Explanatory: BOOLEAN	The model includes explanations of why to select a certain way	

Exploratory, and Explanatory. Since ERP systems have many alternative options supporting alternative business processes, their representation is exploratory in nature. The model may as well be explanatory and provide further assistance by suggesting guidelines for decision-making, but this is not essential for aligning it with the enterprise needs.

Table 4 summarizes the facets and attributes of the Purpose view and indicates the required attribute values of a modeling language adequate for ERP modeling.

The *Customization process* view refers to the process of aligning and customizing an ERP system to the enterprise needs on the basis of the model. It has one facet characterized by two attributes: Customization Approach (e.g., top-down, bottom-up) and Customization Paradigm (e.g., data-driven, requirement-driven).

These two attributes were originally defined in the ERP-adapted CREWS framework [10] as variables of an enumerated type, implying that a modeling language is characterized as uniquely serving a single customization approach or paradigm. However, a modeling language can be applied in different customization (or alignment) processes, taking different approaches and paradigms. We define these two attributes as sets of possible customization approaches and paradigms, for which the modeling language is applicable. Therefore, this classification view does not affect the selection of an ERP modeling language. Rather, it is a result of the selection and application of a modeling language in an alignment process.

4. Object–process ERP representation

This section introduces the OPM, evaluates it according to our adapted CREWS framework criteria, and provides an ERP modeling procedure in which OPM is the modeling language.

4.1. Object–Process Methodology

OPM, described in detail in [19], has been applied for various purposes, such as computer integrated manufacturing [20], image understanding [21], modeling research and development environments [22], algorithm specification [23], document analysis and recognition [24,25], and modeling electronic commerce transactions [26].

The basic building blocks of OPM are two equally important classes of entities: objects and processes, which are related through a variety of links among them. While most object-oriented modeling methods require the use of a set of models, each with its diagramming symbols and conventions, to describe different aspects of the system, OPM uses a single graphic tool, the Object–Process Diagram (OPD) set, as a single model of the structural, functional, and dynamic system aspects. This eliminates the model multiplicity problem [27], which requires special efforts to integrate the various views into a coherent system model and to keep consistency among them.

While using a single model representation, OPM keeps simplicity through two abstracting-refinement mechanisms that control the visibility of the system details. Unfolding objects and zooming

Table 5
Attribute values required for ERP modeling and the corresponding OPM values

View	Facet	Attribute	Required Value for ERP Modeling	OPM's Attribute Value
Content	Abstraction	Abstraction	Flexible	Flexible
	Context	Interaction	True	True
		Contextual	True	True
	Coverage	Functional	(Structure, Function, Behavior)	(Structure, Function, Behavior)
	Argumentation	Variant	True	True
		Variant dependency	True	True
Form	Notation	Notation	Formal/ Semi-formal	Formal
	Structure	Intra-element relationship	(Nested)	Nested
		Inter-element relationship	(Composition, Precedence, AND, OR, XOR)	(Composition, Precedence, AND, OR, XOR, Generalization, Refinement)
Purpose	Aim of representation	Exploratory	True	True

into processes allow a top-down analysis, yielding a hierarchical OPD set, which specifies the system at a spectrum of abstraction levels. Through folding and out-zooming, OPM may as well support a bottom-up analysis, going up in levels of abstraction.

The links employed between the entities in an OPD include structural relations, such as aggregation, generalization, and characterization, and behavioral links, classified into enabling, resulting, and triggering link types.

Aligning the ERP system with the requirements of an enterprise can be achieved when the system and the requirements are represented in the same modeling language. An evaluation of OPM's ability to represent the requirements posed by enterprises regarding ERP systems has found its expressive power adequate for this purpose [11], implying that both contextual and interaction information of the system can be represented in an OPM model. When applying the adapted CREWS framework evaluation criteria presented in Section 3, OPM is found to be suitable for ERP modeling. Therefore, OPM can serve as a modeling language in an ERP-enterprise alignment, since it is suitable for representing both the enterprise requirements and the ERP system. The evaluation of OPM's adequacy for ERP modeling is summarized in Table 5.

4.2. Applying OPM for ERP modeling

The modeling method presented in this section is based on the generic modeling steps introduced in Section 2, and includes specific instructions for representing the ERP data and functionality in OPM terms. Each modeling step is illustrated by an example.

Step 1: Convert the database to an object model. In this step, the data model is constructed based on the database tables. It involves two activities: representation and generalization. First, each of the system's tables is represented by an OPD. Then, structurally similar objects are generalized into higher-level objects. The representation activity converts each database table into an OPD, in

which the table's title is an object characterized by each one of its fields through characterization links. Foreign keys in a table are represented either by a tagged structural relation (e.g., *Purchase order* Is_Supplied_by *Supplier*) or by an aggregation link (e.g., *Purchase order line* to *Purchase order*), depending on the context.

Second, the objects that share common features (attributes and operations) are identified as specializations of a generalized object. Then the generalized object, its features and inheritance relations constitute one OPD, and each of the specialized objects with its unique features is represented in a separate OPD.

As an example, the OPDs in Fig. 1(a) and (b) are obtained in the initial representation activity from the database tables *Standard Item* and *Customized Item*. For the sake of clarity, the OPDs in the Figure, as well as in the other examples, are simplified. Since both objects share many common features, the generalization activity defines the object *Item* as their abstraction, whose features, shown in Fig. 1(e), are the features that are common to both tables. Each table is modeled as a specialized object of *Item* and exhibits only its unique features, as shown in Fig. 1(c) and (d).

Step 2: Construct the Global business process model. In this step, both domain knowledge and ERP system knowledge are applied in order to define the high-level generic business processes supported by the system. The sessions of the system are grouped into functional groups, such as ''purchase order creating'' or ''purchased goods receiving''. Each such functional group is represented as an OPM process. The objects that enable and are manipulated by the processes are identified and their links with the processes are established. Objects included in the model are preferably generalized objects, or else the main objects related to database tables. Attributes are not included in the model at this step.

This step is illustrated by the process *Purchase Goods Receiving* in Fig. 2.

This top-level process includes three sub-processes, *Purchase Receipt Registering, Goods Inspecting and Approving* and *Warehouse Allocating*. The instruments used by this process are the objects *Item, Warehouse*, and *Supplier*, while the other objects, such as *Purchase Order* and *Inventory by Item*, are objects that the process *Purchase Goods Receiving* affects by changing the state or value of one or more of their attributes.

Note that the refinement of OPDs does not necessarily correspond to the modeling steps. Rather, a modeling step may yield several levels of the OPD set representing the system, or merely add details to one or more OPDs obtained in earlier steps.

Step 3: Identify System Configuration-level business process alternatives. In this step, the system parameters in the data model are investigated and categorized. The high-level process-defining parameters are identified as preconditions to some of the user-interface sessions. The alternative business processes are represented in OPM as specializations of the processes in the Global-level model. Each process alternative is related by condition links to the relevant states of the control parameters, represented by objects. The sub-processes modeled for each business process alternative are the sessions it includes.

This step is illustrated in Fig. 3, which shows the alternative specializations, obtained when zooming into the *Warehouse Allocating* process. These alternative processes are controlled by the system parameter *Location Control Implemented?*, whose *Yes* state is related by a condition link to *Location Controlled Warehouse Allocating.* This modeling step may also include zooming into the alternative process specializations in order to represent their sub-processes.

Step 4: Identify Object-level variants of the business processes. This step identifies variants of the business processes, which are controlled by the parameters of single data objects, either by themselves or in combination with the system parameters. The attributes of the objects that participate in the modeled processes are studied to identify their influence on the process activities. The outcome in the OPM model can be specialized processes conditioned by object attribute states, or as specific sub-processes of a process, each of which is related by a condition link to an attribute state.

This step is illustrated by the OPDs in Fig. 4 and 5. In Fig. 4, the attribute, represented by the

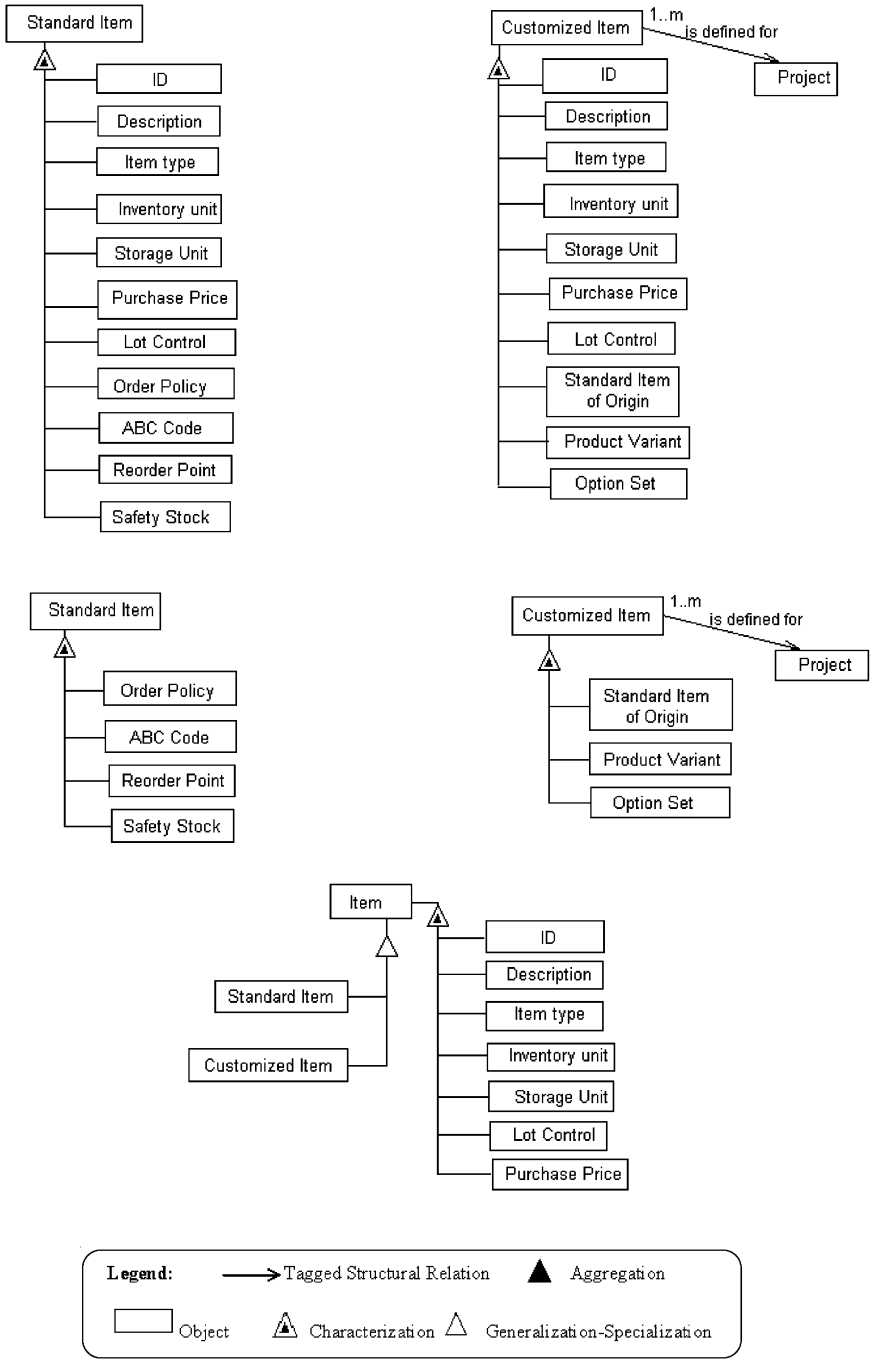

Fig. 1. A part of the database model.

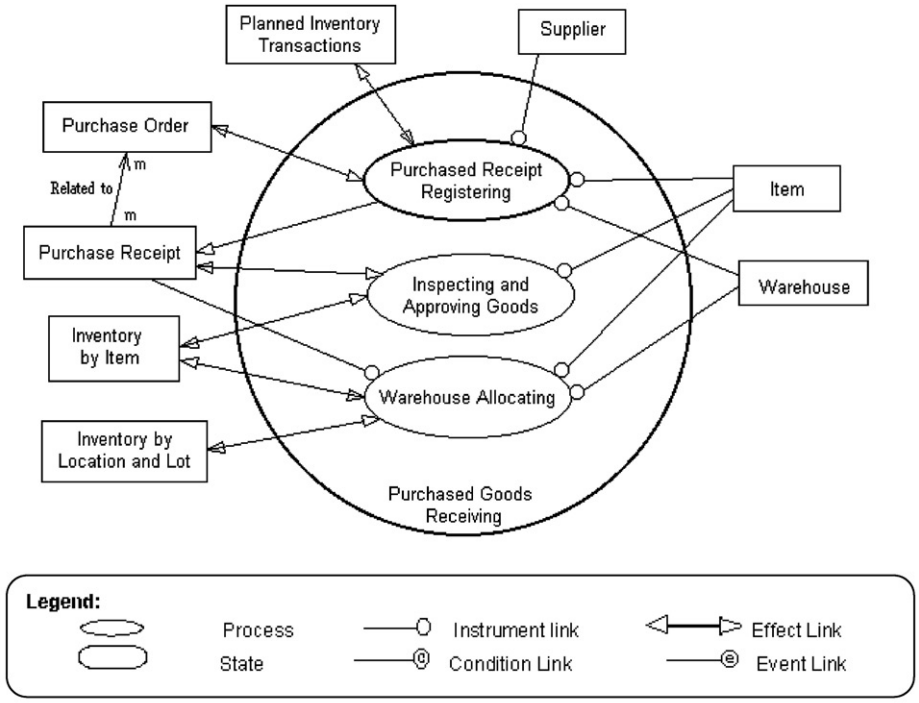

Fig. 2. A top-level business process model.

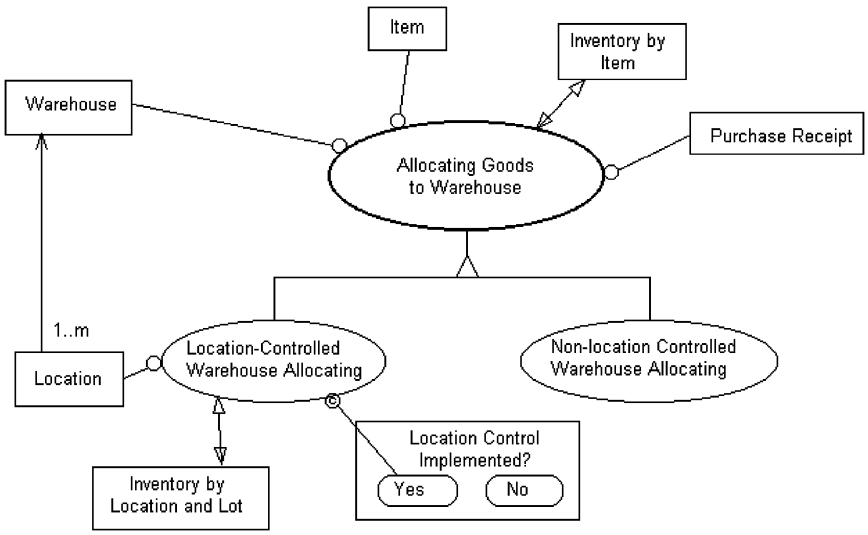

Fig. 3. System configuration-level business process model.

Boolean Object *Location Controlled?* of a *Warehouse*, allows for performing different processes in different *Warehouse* instances. The process is controlled by the combination of the system parameter *Location Control Implemented?* and the *Warehouse* attribute values.

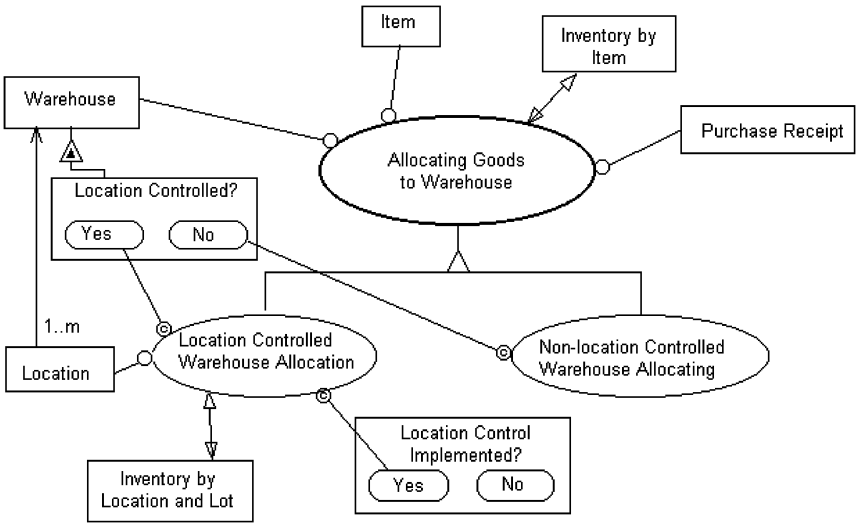

Fig. 4. Business process variants controlled by an attribute of a warehouse.

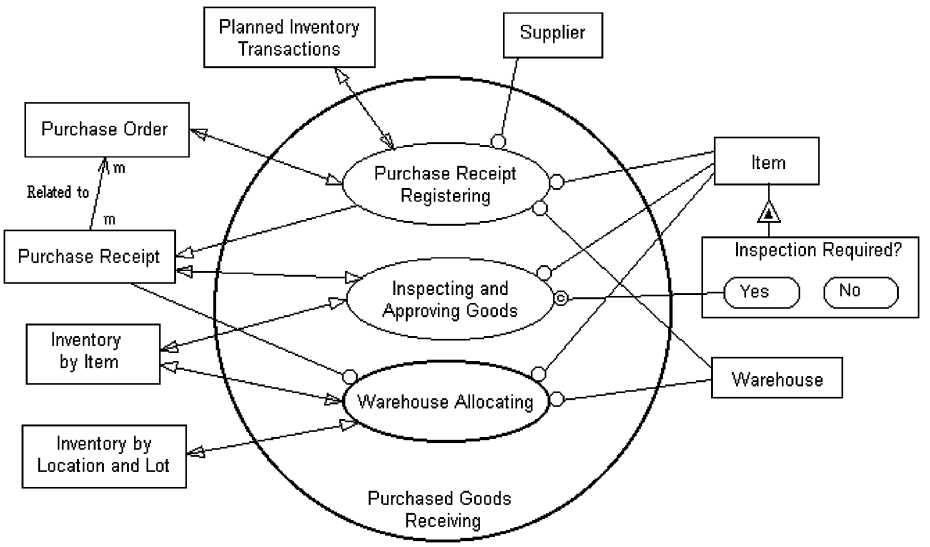

Fig. 5. A business process variant controlled by an attribute of an item.

In Fig. 5, the process *Goods Inspecting and Approving* is conditioned by the Boolean attribute *Inspection Required?* of *Item*.

Step 5: Expose the Occurrence-level business process options. In this step, the options available to the user in each of the sessions represented in the model are investigated. These options are determined by the user in real time, and allow a limited control of the process' course. In the OPM model, a user-controlled option is an object inside a process, whose states are events that trigger sub-processes.

This step is illustrated by the example in Fig. 6, which shows the option *Fast Approval?* available to the user in the process *Purchase Receipt Registering.*

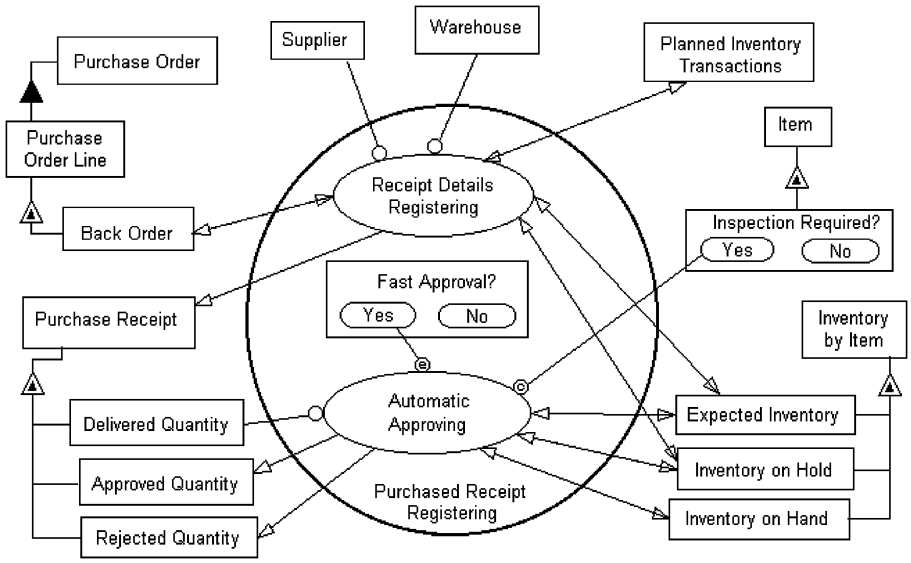

Fig. 6. Process option model.

By selecting the state *Yes* of the *Fast Approval* option, the user triggers a step of *Automatic Approving* as part of the registering process.

For the purpose of being matched against the enterprise requirements, the ERP model needs to be modular, so that alternative processes are represented in different OPDs. Then each OPD in the system model can be evaluated separately, so that the alignment rules out the OPDs that specify the ERP parts that do not match the requirements, and selects the ones that comply with them. The following modeling principles ensure that the constructed model is modular and complete.

- Separation of concerns: Each OPD in the model deals with a single issue and can be given a single title that fully expresses its content. This principle addresses the tendency of modelers to try to squeeze as much information as possible in an OPD (e.g., a process affecting an object together with the structure of the object). According to this principle, different issues (process, object data structure) need to be represented in different OPDs.
- Separation of alternatives: When there are several alternative specializations of a specific process, whose selection is controlled by a system parameter or an object attribute, their details should be specified in separate lower-level OPDs.
- Completeness of low-level OPDs: Where links between an object and a process appear in the lowest-level OPDs (the "leafs" of the OPD set), all the attributes that participate in the link (either as enablers or as objects which are affected) must be specified. According to the separation of concerns principle, structural information of an object, rather than being included in OPDs showing processes in which it is involved, is represented in a separate OPD. However, in the lowest level OPDs the complete details of the relation are provided by showing the specific object attributes that participate in the processes. This principle is demonstrated in Fig. 6, which, as a low-level OPD, provides all the affected attributes of the objects participating in the process.

5. Conclusions

The ERP modeling approach suggested in this paper addresses the construction of an ERP model, for the purpose of aligning an ERP system with the needs of an enterprise.

The contribution of the paper is threefold. First, we introduce generic ERP modeling steps, aimed at capturing the entire scope of process variants supported by the ERP system and the interdependencies among them. These generic steps may be applied using a variety of modeling languages. However, the complexity of the problem poses requirements on properties that a modeling language should satisfy to be suitable for this task. The second contribution of the paper is the analysis and characterization of these properties, which results in a set of evaluation criteria for modeling languages. This analysis extends the ERP-adapted CREWS classification framework [10]. The main addition to the framework is the consideration and representation of possible dependencies among variants and options in the ERP system. The analysis and evaluation criteria established led to the selection of OPM as the modeling language of choice for ERP modeling. Since OPM has been evaluated in [11] and found adequate also for representation of enterprise requirements, it is now shown to be applicable as a basis for ERP-enterprise alignment. Finally, the third contribution is a detailed ERP modeling procedure that we present in Section 4. The modeling procedure can be used as is, or serve as an example showing how the generic modeling steps are refined into a detailed procedure.

The ERP modeling process has been applied and proven useful in obtaining a model of the parts of the Baan ERP system as part of a requirement-driven alignment experiment. The model construction was labor-intensive and required studying the details of the ERP system and the processes it supports. Nevertheless, this is a one-time effort, which produces a model that can serve repeatedly for any number of implementation projects.

The modeling process, although developed for ERP systems, is not limited only to this type of systems. The levels of optionality included in ERP systems appear also in other off-the-shelf customizable information systems. Therefore, the modeling process may be relevant to any process-supportive generic off-the-shelf information system. It may also be of importance to reverse engineering of product families, which include parametric variability.

Future research addresses the application of the ERP model in ERP-enterprise alignment. It can also apply the evaluation criteria for evaluating and comparing other modeling languages, and develop detailed modeling procedures for those languages that are found suitable on the basis of the generic modeling steps. Another research direction is to relate our modeling process, which deals with optionality, to the architectural-oriented reverse engineering approaches of product families.

Appendix A

See Table 6.

Table 6
ERP-adapted CREWS framework

View	Facet	Attribute
Content	Abstraction	Abstraction: ENUM {Unique, Fixed, Flexible}
	Context	Internal: BOOLEAN Interaction: BOOLEAN Contextual: BOOLEAN
	Coverage	Functional: BOOLEAN Intentional: BOOLEAN
	Argumentation	Variant: BOOLEAN Arguments: BOOLEAN
Form	Notation	Notation: ENUM {Formal, Semi-formal, Informal}
	Structure	Element type: SET (STRING) Intra-element relationship: SET (ENUM {Flat, Nested}) Inter-element relationship: SET (ENUM {Composition, Generalization, Precedence, AND, OR, XOR, Refinement})
Purpose	Aim of representation	Descriptive: BOOLEAN Exploratory: BOOLEAN Explanatory: BOOLEAN
Customization process	Customization process	Approach: ENUM {Top-down, Bottom-up, Middle-out, Single-level, Breadth-first, Depth-first} Paradigm: ENUM {Data-driven, Requirements-driven, Quality-driven, Behavior –driven}

References

[1] T.H. Davenport, Putting the enterprise into the enterprise system, Harv. Business Rev. 76 (1998) 121–131.

[2] C.P. Holland, B. Light, A critical success factors model for ERP implementations, IEEE Software 16 (1999) 30–35.

[3] B. Light, The maintenance implications of the customization of ERP software, J. Software Maintenance: Res. Practice 13 (2001) 415–429.

[4] M. Krumbholz, N. Maiden, The implementation of enterprise resource planning packages in different organisational and national cultures, Inf. Systems 26 (2001) 185–204.

[5] J. Ghosh, SAP Project Management, McGraw-Hill, New York, 2000.

[6] T.A Curran, A. Ladd, SAP R/3 Business Blueprint: Understanding Enterprise Supply Chain Management, 2nd edition, Prentice-Hall, Englewood Cliffs, NJ, 1999.

[7] 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.

[8] H.A. Post, R. Van Es (Eds.), Dynamic Enterprise Modeling: a Paradigm Shift in Software Implementation, Kluwer, Dordrecht, 1996.

[9] C. Rolland, Intention driven component reuse. In: S. Brinkkemper, E. Lindencrona, A. Solvberg (Eds.), *Information Systems Engineering: State of the Art and Research Themes*, Springer, Berlin, 2000, pp. 197–208.

[10] C. Rolland, N. Prakash, Bridging the gap between organizational needs and ERP functionality, Requirements Eng. 41 (2000) 180–193.

[11] P. Soffer, B. Golany, D. Dori, Y. Wand, Modelling off-the-shelf information systems requirements: an ontological approach, Requirements Eng. 6 (2001) 183–199.

[12] A.W. Scheer, ARIS—Business Process Frameworks, Springer, Berlin, 1999.

[13] H. Yang, X. Liu, H. Zedan, Abstraction: a key notion for reverse engineering in a system reengineering approach, J. Software Maintenance: Res. Practice 12 (2000) 197–228.

[14] W.J. Heuvel, M. Papazoglou, M.A. Jeusfeld, Configuring Business Objects from Legacy Systems, in: Proceedings of the CAiSE '99 (LCNS 1626), Springer, Berlin, 1999, pp. 41–56.

[15] H. Lee, C. Yoo, A form driven object-oriented reverse engineering methodology, Inf. Systems 25 (2000) 235–259.
[16] R.L. Krikhaar, Reverse engineering approach for complex systems, in: Proceedings International Conference on Software Maintenance, IEEE, Los Alamitos, CA, 1997, pp. 4–11.
[17] B. Bellay, H. Gall, Reverse engineering to recover and describe a system's architecture, in: Development and Evolution of Software Architectures for Product Families, ARES '98 (LNCS 1429), Springer, Berlin, 1998, pp. 115–122.
[18] C. Rolland, C. Ben Achour, C. Cauvet, et al., A proposal for a scenario classification framework, Requirements Eng. 3 (1998) 23–47.
[19] D. Dori, Object Process Methodology—a Holistic Systems Paradigm, Springer, Berlin, Heidelberg, New York, 2002.
[20] D. Dori, Object–process analysis of computer integrated manufacturing documentation and inspection functions, Int. J Comput. Integrated Manuf. 9 (1996) 339–353.
[21] D. Dori, Analysis and representation of the image understanding environment using the object–process methodology, Object-Oriented Programming 9 (1996) 30–38.
[22] D. Meyersdorf, D. Dori, The R&D universe and its feedback cycles: an object–process analysis, R&D Manage. 27 (1997) 333–344.
[23] L. Wenyin, D. Dori, Object–process diagrams as an explicit algorithm specification Tool, J. Object-Oriented Programming 12 (1999) 52–59.
[24] D. Dori, Arc segmentation in the machine drawing understanding environment, IEEE Trans. Pattern Anal. Mach. Intell. 11 (1995) 1057–1068.
[25] L. Wenyin, D. Dori, A generic integrated line detection algorithm and its object–process specification, Comput. Vision-Image Understanding (CVIU) 70 (1998) 420–437.
[26] D. Dori, Object–process methodology applied to modeling credit card transactions, J. Database Manage. 12 (2001) 2–12.
[27] M. Peleg, D. Dori, The model multiplicity problem: experimenting with real-time specification methods, IEEE Trans. Software Eng 26 (2000) 742–759.