Requirements Eng (2001) 6:183–199
© 2001 Springer-Verlag London Limited

Requirements Engineering

Modelling Off-the-Shelf Information Systems Requirements: An Ontological Approach

Pnina Soffer[a], Boaz Golany[a], Dov Dori[a,b] and Yair Wand[c]
[a]Technion – Israel Institute of Technology, Haifa, Israel; [b]Massachusetts Institute of Technology, Cambridge, Massachusetts, USA; [c]University of British Colombia, Vancouver, Canada

Requirements for choosing off-the-shelf information systems (OISR) differ from requirements for development of new information systems in that they do not necessarily provide complete specifications, thus allowing flexibility in matching an existing IS to the stated needs. We present a framework for OISR conceptual models that consists of four essential elements: business processes, business rules, information objects and required system services. We formalise the definitions of these concepts based on an ontological model. The ontology-based OISR model provides a framework to evaluate modelling languages on how appropriate they are for OISR requirements specifications. The evaluation framework is applied to the Object-Process Methodology, and its results are compared with a similar evaluation of ARIS. This comparison demonstrates the effectiveness of the ontological framework for evaluating modelling tools on how well they can guide selection, implementation and integration of purchased software packages.

Keywords: Conceptual model; Object-Process Methodology; Off-the-shelf information systems; Ontology; Requirements specification

1. Introduction

As enterprises increasingly purchase their information systems rather than develop them from scratch, off-the-shelf information systems play an increasingly important role in corporate information systems. These large-scale information systems are typically generic software packages, such as ERP and CRM, that are designed to serve a variety of enterprise types. The selection and implementation of such packages are based on requirements defined by the enterprise. During the evaluation and selection, the enterprise verifies that the software package is able to satisfy a limited set of major requirements. The implementation of the selected set of one or more software package entails a process of customising them to the specific needs of the enterprise and possibly the integration of several packages. The customisation involves defining the software configuration, parameter values and sometimes the enhancement of its functionality. Often, the enterprise needs to adapt its business processes to the software package rather than the other way round. Thus, the customisation process might involve Business Process Reengineering [1]. This concurrent process of software configuration and business solution design might affect significantly both the success of the implementation project and the future business practices of the enterprise [2].

Extant support tools for this mutual adaptation process do not explicitly provide for stipulating the enterprise requirements. Rather, they frequently incorporate 'reference models' that supposedly represent 'best business practice', represented by enterprise modelling tools such as Integrated Architecture of Information Systems (ARIS) [3,4]. The enterprise is expected to select and adopt such a standard model for its future business practice 'as is', or with minimal changes. The selection and refinement of a standard model into the business solution are based solely on available solutions. This process is sometimes referred to as 'blueprinting' and its outcome is the 'business blueprint' [4]. The business blueprint is then regarded as the new enterprise requirements, leaving no record of the original require-

Correspondence and offprint requests to: P. Soffer, Technion – Israel Institute of Technology, Technion City, Haifa 32000, Israel. Email: Pnina@tx.technion.ac.il

ments or the requirements engineering process, which have provided the basis for both the decision making process and the final outcome.

In cases where the enterprise finds no standard model satisfactory, or where major deviations from such a model are required, the customisation process is not systematically supported. Rather, it is typically done in an ad hoc manner that requires a considerable effort.

Software vendors often claim that their standard solutions are applicable without any change. Nevertheless, the only reported attempt to investigate this claim has shown a different picture. Daneva [5] defined a reuse metrics associated with the SAP R/3 reference model in order to measure requirements reuse in implementations of this product. This metrics was applied to four different cases of enterprise that were included in the implementation blueprint. The results indicated that full reuse was not achieved, although in some cases the rate of reuse was remarkably high.

Supporting both standard enterprises and unique ones requires a systematic decision process that emphasises the enterprise requirements rather than the system's capabilities and the standard solutions. The first step in such a decision process is a requirements engineering (RE) process, which leads to a requirements specification document. This specification can then be compared against and matched with the software package capabilities, whose representation is discussed in Soffer [6], and provide a business solution via a structured process [7].

The requirements should be specified in a manner that supports further analysis and matching with software packages. However, here lies a difficulty. Software is usually described in terms of functions, data, queries and transactions. In contrast, business needs are understood in terms of business concepts. Thus, incompatibility is built into this matching process due to the 'mismatch' between the terminology used to describe the software and that used to describe the business requirements. It makes little sense to adapt the description of business needs to the language of describing information systems functionality using terms related to data structures or query definitions. To remedy this fundamental problem, we propose that Off-the-shelf Information Systems Requirements (OISR) be described using *business concepts and terms*. This is especially important if we are to adapt the enterprise to the software capabilities.

The way we describe a business conceptually is termed a conceptual model (CM). Constructing a CM has been defined by Mylopoulos as 'The activity of formally describing some aspects of the physical and social world around us for the purpose of understanding and communication' [8]. We must not confuse the enterprise CM, discussed below, with conceptual models of databases, which deal with a specific set of aspects relevant for the purpose of developing and managing the supporting information system (in particular, the database).

The resulting CM, which serves as the basis for selection, matching and adaptation of a software package, depends on the quality of the conceptual modelling method being used, underscoring the importance of selecting an effective modelling method.

Early OISR- related works focused on the RE process [9,10] and the application of OISR to the selection [11–14] or customisation [4] of a software package. However, although the way the application domain was modelled could determine the coverage of the OISR model to a great extent, the modelling method itself was rarely considered.

In this work we propose a process for selecting an appropriate CM method through careful evaluation of the candidate methods. Specifically, our approach is as follows. (1) Propose a generic model that relates the business view and the information system. (2) Based on the generic model, define the OISR framework. (3) Apply an ontological model to precisely define the OISR concepts. We use the Bunge–Wand–Weber (BWW) ontological model, which has already been employed in various settings [3,15,16]. (4) Evaluate a given CM method to see how well it represents the constructs identified. In particular, we will examine the analysed method for missing, ambiguous and redundant constructs. (5) Compare CM methods with respect to the outcome of the analysis performed.

The remainder of this paper is organised as follows. In Section 2 we provide the generic model and develop the OISR framework. Section 3 presents an ontological analysis of OISR. This analysis forms the basis for evaluating the expressive power of a given modelling method to be applied for an OISR CM. Section 4 applies the ontological evaluation framework to the Object-Process Methodology (OPM) [17], which serves as the modelling platform in the ERP customisation process described in Soffer [7]. The results are compared with a similar evaluation of ARIS [3]. Section 5 provides our conclusions.

2. Developing the OISR Framework

2.1. Related Work

While requirements for developed IS have been frequently addressed in the RE literature, the attention paid to OISR has been relatively small. Recently, however, OISR has become a subject of research. The OISR models in earlier works has encompassed *scenarios* [11–13], *maps* [9,18] and *enterprise modelling tools* [4,5,10]. These are different representation techniques, each specifying a different set of aspects of the desired IS and its environment.

Scenarios, which specify business processes and their required use of the IS, are employed mainly for the selection of software products [11–13]. Maiden and Ncube [11,12] advocate an iterative process of requirements acquisition and product selection phases, where partial requirements are used and each iteration increases the level of details in the requirements specification and narrows the set of candidate software products on the basis of the current set of requirements. They specify a list of functional, behavioural and non-functional requirements for the IS to be selected, formally represented as a part of their process guide engine [12]. In Maiden and Ncube [11], the functional and behavioural requirements are verified through scenarios. Scenario representation of the OISR is also used by Feblowitz and Greenspan [13] for analysing the impact of commercial off-the-shelf (COTS) systems on business practices of enterprises as part of the acquisition decision. Scenarios seem to be sufficient for differentiating products based on their capabilities. However, the customisation of the selected IS in the implementation phase should discriminate not only among different products but also among different configuration options within the product, for which scenarios are not suited. Requirements specification, on the other hand, is capable of expressing the entire suite of issues involved and predicting situations that are not considered by the main scenarios.

Another approach to OISR representation is presented by Rolland [9], whose model maps the OISR as ‘intentions’ and ‘strategies’. The intentions represent goals of the enterprise, while the strategies express possible ways of achieving them. The detail level of the maps is controllable. The maps are further extended in Rolland and Prakash [18] to include ‘situations’, defining preconditions in terms of data. However, this is a semi-formal representation, which is not aimed at providing an explicit definition of the data structures. Therefore, the combination of intentions, strategies and situations, which seems useful for representing most of the OISR in a general manner, may require considerable effort when representing specific detailed requirements that involve compound conditions and a detailed data design.

Enterprise modelling techniques, which are in common use for the blueprinting process, encompass a range of aspects, depending on the specific modelling technique. For example, ARIS [19] includes a function view, a process view, a resource view, a data view and an output view; Enterprise Knowledge Development (EKD [20]) includes an objectives view, a business rules view, a process view, an information view and a resource view. Some of these views are redundant [3] since they provide information that is already included in one or more of the other views (e.g., the function view of ARIS includes information that is already included in the process view). Others are not needed as a part of the OISR. In particular, the objectives view of EKD is an important step in the process of requirements definition, but is in itself too abstract to be a part of the OISR. The resource or organisation view (in both ARIS and EKD), although important for understanding the enterprise, is out of the IS scope and is actually independent of the IS functionality. Both the IS and the organisation structure may change independently without affecting each other. Thus, the links between this business aspect and the information system description might not be well established. Similarly, other models that represent physical flows and outputs (ARIS) are out of the OISR scope.

Despite the contribution of all these works to the understanding and practice of OISR, they do not provide an in-depth discussion of OISR in terms of both content and scope.

2.2. Relating the Business and IS Views

RE processes usually start in developing an understanding of the business needs and then deriving the IS requirements from them. In order to develop an understanding of how to express OISR we examine IS requirements in general and then point out the differences between OISR and the general requirements for IS development.

Nilsson [21] discusses different levels of modelling in the RE process, and argues that the same fundamental structure is repeated at different *levels of abstraction*. Figure 1, adapted from [21], presents two of these modelling levels: *operational business modelling* and *IS modelling*. Their fundamental structure consists of *intentions*, *processes*, *rules* and *objects*. The intentions motivate and are satisfied by *processes*, which are controlled by the rules, expressing the intentions. The processes create and consume *information objects*, whose states are regulated and defined by the rules.

The interface between the two levels is in the *business messages* created by the business processes and the *system services* provided by the system in response to the business messages. The intention of the IS is therefore providing services in relation to the input messages. Note, while the word ‘services’ implies an action, here we will usually refer to the outcome of the action. The outcome observable to the business will usually be an information object (newly created or modified). Also, the

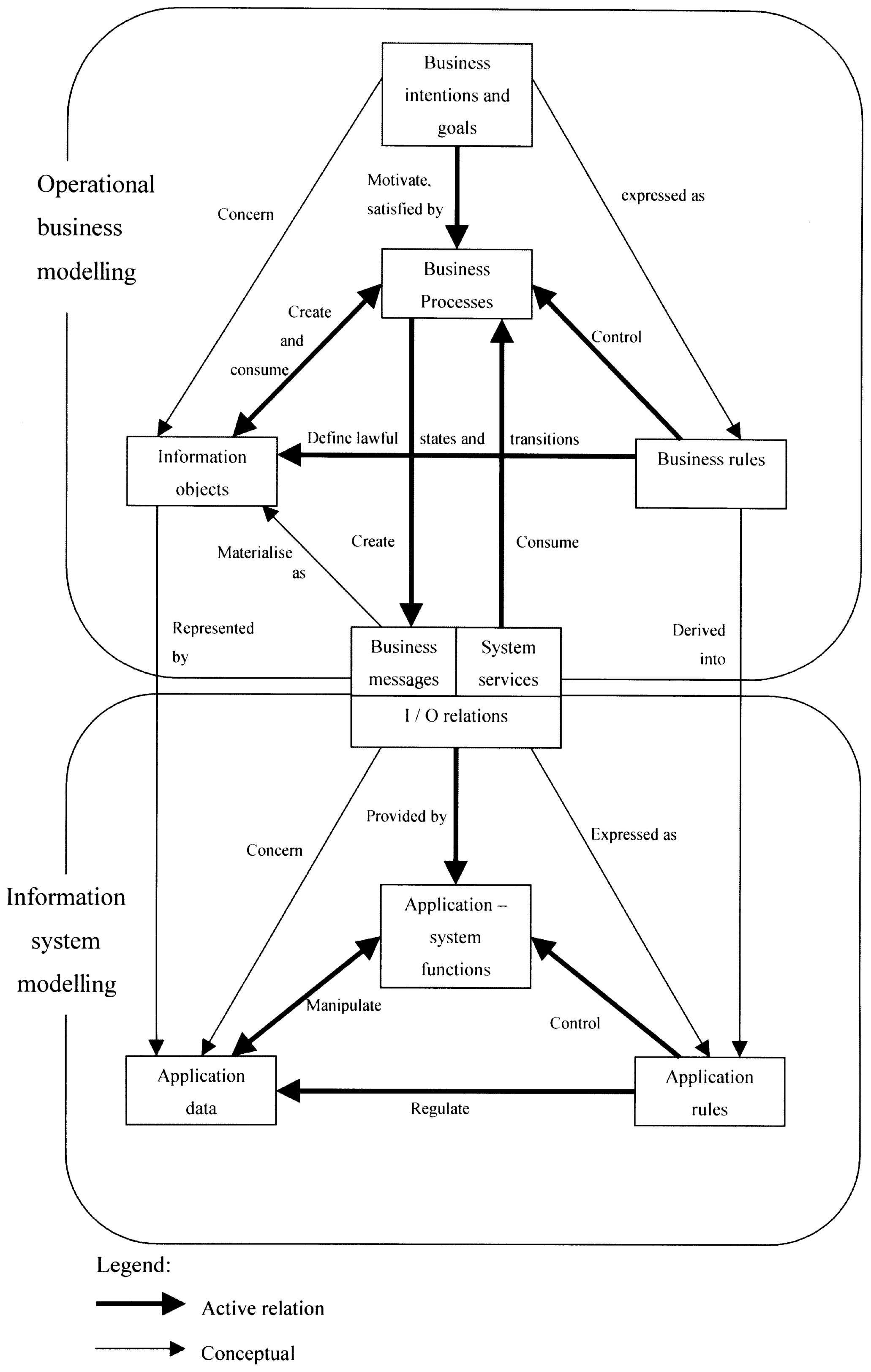

Fig. 1. Modelling levels in the RE process.

way business processes can send messages to the IS is via information objects. As a result, the IS might change its state (that is, values of the application data). Thus, the messages and the system responses (services) materialise as information objects that are created, modified and used by the business processes. In this way, the role of the IS can be described as taking charge of manipulating information objects to satisfy the needs of the business processes. The information system is notified about these needs via the messages, and its responses comprise the system services.

2.3. OISR Framework

The ultimate requirement posed on the information system is to provide the system services in response to the business messages, as defined in the interface between the business model and the IS model. (See also Davis [22], where requirements are defined as specification of the responses of the system to given input.)

However, as discussed in Section 1, the OISR serves a purpose of mutual adaptation of both the business and the IS. Therefore, it does not necessarily employ a complete interface specification. Completeness, which is one of the basic properties characterising IS development requirements, is defined by Davis [22] as follows: '(a) everything the software is supposed to do is included in the requirements specification. (b) Definitions of the responses of the software to all realisable classes of input data in all realisable classes of situations are included. (c) No sections of the specifications are marked ''To Be Determined (TBD)''...'. OISR, in contrast, is incomplete in its nature. The incompleteness of OISR has been recognised by several authors [11–14]. It is described humorously as IKIWISI ('I know it when I see it') [23]. In particular, we find that 'TBD's play a significant role in an off-the-shelf IS implementation process. Since a software package implementation involves changes in the enterprise in adaptation to standard processes which the IS supports, the rigidity of the complete requirements may prevent the detection of an exact matching solution within the software. Rather, the OISR should accurately specify the interface only where the detailed design is of considerable importance to the enterprise. The rest can be left as a 'TBD' and provide the enterprise with the flexibility and adaptability to the existing interfaces provided by the software package.

Denoting by S the set of system services needed for the operations of the enterprise and by S′ be the set of core system services to be specified in detail, the assertion $S' \subseteq S$ holds.

In order to fill the gap between S′ and S, OISR should consider the context in which the system services are to be used. We assume that system services are used through business processes (see Fig. 1). Therefore, specifying the business processes may identify required services even when these are not explicitly specified. Nevertheless, the implementation of an off-the-shelf IS may involve redesigning some of the business processes. In this context, accurate documentation of all the existing processes may have an adverse effect on flexibility and is therefore undesirable. Still, a subset of processes, which we term *core* processes, must not be changed in the course of the IS implementation. Let P be the set of all business processes in the enterprise, and P′ be the set of core processes, then $P' \subseteq P$.

In order to fill the gap between P′ and P, the OISR should provide more information about the *invariant aspects* that are not specified by the core processes. Business processes are controlled by business rules (BR), which express the business goals (see Figure 1) and are not to be changed through the IS implementation. Therefore, the role of the business rules in the OISR is to provide the underlying logic wherever the interface and the processes are not specified in detail.

To complete the definition of the OISR framework, we need to specify the information objects as the entities to which the business processes and rules apply. The information objects are created and consumed by the business processes, thus forming the system services. The states and transitions of information objects are restricted by the business rules. In the OISR we do not require completeness of the set of information objects specified. Rather, the objects to be specified are the ones that participate in the business processes in the form of system services and are specified by the rules. Let I be the set of all information objects existing in the enterprise, and let I′ be the set of information objects that participate in S′, or are manipulated by P′ or are controlled by BR, then $I' \subseteq I$.

In conclusion, the OISR can be defined by the four sets discussed above, as a tuple $<S', P', BR, I'>$, providing sufficient information for the decision making involved in the IS selection and implementation. In the next two sections, we discuss further each of the OISR elements, clarify its content and relationships to other elements, and formalise the modelling constructs for OISR representation.

Business processes. The concept of processes is now commonly accepted as a good way to think about and analyse business needs. While business processes are not the only concept applied to describe a business, they are an important one. Indeed, in the ERP domain, process models are used often. A business process is a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs: a structure for action [24]. These ordered activities affect the states of the IS by creating information objects, consuming them and changing their contents (represented as attribute values).

System services. These are messages produced by the system as a result of messages received [21]. The incoming messages can be divided into two categories: requests for information (queries) and reports of tangible changes in the state of the environment served by the IS (transactions), such as a change in the inventory level of an item. The system service provided for queries appears in the form of requested information objects and their contents (represented as attribute values). In the second case, the system records the change internally in terms of application data (e.g., inventory transaction) and provides a message that reflects this record. The provided system service depends on the incoming message as well as the current state of the system, as determined by the sequence of activities involved in the business processes (and represented in the system in terms of the application data).

Information objects. These represent entities in the business that participate in and are affected by business processes. The information objects' attributes and states, as well as how they are transformed by the business processes, should be specified in the OISR. The relations among these objects, which drive the business processes, should be modelled as well.

Business rules. This term is used for describing several different types of declarations. Different researchers [20,25,26] classify them into different categories, which can be summarised into the following two:

- structural constraints, which limit the possible states of an entity and its attribute values;
- rules of dynamics, which restrict the behaviour of the entities in the system. Specific cases of these are derivation rules, which define the way attributes change their value, transition rules that define lawful state transitions by means of pre/post conditions, and stimulus/response constraints, specifying the action that should be taken in response to certain events. A specific case of a transition rule is an existence rule, specifying the conditions for a transition between existent and non-existent states of an entity.

The following example illustrates the four OISR elements and the way they interact. Consider a production environment, with a process including the activities of *production planning*, preceding *production execution* and *production control* that are performed in parallel, followed by *posting to inventory* and finally *order closing*. Note that physical inputs and outputs are not considered as a part of this process, since they themselves do not concern the information system directly. This process uses system services, which include *order documents*, *costing report* (providing requested information) and *production report* (reflecting a change in the real world). It manipulates information objects, such as *production order*, *production schedule* and *inventory* records. The specification of a business object such as *production order* includes its attributes: *order number*, *item ID*, *quantity*, *start date*, *due date*, *routing number* and *status*. A related rule of the type 'structural constraint' states that the possible values of the attribute *status* are *planned*, *released*, *active*, *completed*, *closed*, *held* and *cancelled*. Rules of dynamics may specify these values as pre and post conditions of activities. Some activities are part of the production process while others express exceptional situations, such as holding, releasing or cancelling an order. The *Order documents*, created as a system service, are used by *Production execution*. In this example, generating an order document is a core system service, which should be fully specified. Documents of this type must be specified in detail, because missing information in these documents may cause errors in the production process.

We shall elaborate on this example in the later parts of this paper.

3. OISR Ontology

In the previous sections we have explained the need for understanding what aspects of the environment should be included in the OISR, and suggested a framework for their conceptual models. In this section we develop precise ontological definitions of the concepts involved in this framework. These definitions enable us to identify the set of modelling constructs, which are necessary and sufficient for OISR representation, resulting in an evaluation framework for OISR modelling techniques. The purpose of this evaluation framework is to assess the expressive power of modelling languages when applied to OISR representation.

3.1. The Bunge–Wand–Weber Ontology

Wand and Weber [15,16,27] have extended an ontology presented by Bunge [28], and applied it to the modelling of information systems. The Bunge ontology presents a set of high-level, abstract constructs that are intended to be a means of representing all real-world phenomena.

His work relies on the philosophical foundations of Aristotle, Aquinas, Descartes and others. Wand and Weber have based their work on Bunge's work because of its rigour and comprehensiveness. The Bunge–Wand–Weber (BWW) ontology has been used as a theoretical basis for a number of works, addressing a variety of issues, such as metrics for OO design [29], metamodelling [30], enterprise modelling [31] and data quality dimensions [32]. One of the fundamental premises of the ontological approach is that a sufficiently expressive modelling grammar is a necessary condition for a model to be a complete one. Such a modelling grammar must be able to represent all the concepts in the real world that might be of interest to users of information systems. Based on this premise, Wand and Weber developed a framework that uses the ontological representation model in order to evaluate the expressive power of modelling languages [15,16]. This framework has been applied for the evaluation of several modelling grammars, such as ER [15,16], DFD [15], ARIS [3] and others. The evaluation of a modelling language regards ontological completeness and ontological clarity [16]. Completeness is defined as a total mapping of all the ontological constructs to modelling grammar constructs. Clarity is the lack of three deficiencies: *construct overload*, when one modelling construct maps into two or more ontological constructs, *construct redundancy*, when two or more modelling constructs represent a single ontological construct, and *construct excess*, when a grammatical construct does not map into any ontological construct.

Green and Rosemann [3], who applied the BWW framework, have pointed out that it may be too general. They suggested that the purpose of modelling should be taken into account and reflected in a specialised subset of the ontology when evaluating a modelling technique.

A specialisation of the BWW for the purpose of OO enterprise modelling has already been developed in Wand and Woo [31], but the emphasis there is static, and the dynamics of business processes is not explicitly addressed. We propose a specialisation of the BWW ontology that serves the purpose of OISR modelling.

3.2. Specialised OISR Ontology

While the BWW ontology aims at representing 'the world', our goal is to represent only some of its parts that are relevant to OISR. These include the elements discussed in Section 2: processes, rules, information objects and system services. Below we present and discuss the BWW ontological constructs that provide the basis for defining the elements of the OISR tuple. These are a subset of the BWW constructs, a more complete list of which is given in Appendix 1.

Thing. The world is made of things.

According to this fundamental premise of the BWW ontology, things are the elementary real-world units. While a thing in the BWW ontology is a concrete substance that exists in the real world, the OISR addresses reflections of real-world things (such as items) and concepts (such as a production order) in an information system. Therefore a thing in the OISR is an informatical thing or a *virtual thing* (namely, a thing somebody believes to exist).

Property. Things possess properties.

A property inherently possessed by an individual thing is termed *intrinsic*. A property that is meaningful only in the context of two or more things is called *mutual*.

Attribute. Attributes are representations of real-world properties, and thus properties of the models of things, as viewed by people.

Attributes map properties of a thing into values, and may be expressed as functions.

Let t be a thing, M a non-empty set, and V a domain of values, then each attribute i of the thing is a function F_i: M $\rightarrow$ V_i. The domain M represents conditions under which the value of F_i is observed, such as time instances.

Since the OISR model addresses reflections of the real world, it includes attributes as a representation of properties.

Class. A subset of things is called a class if and only if a property exists, such that the subset is equal to the set of things that possess this property.

The common property may be a composite one, encompassing a set of behavioural and characteristic properties of the things. Since the OISR defines process patterns and general-case rules that generalise real-life occurrences, the OISR modelled unit is meaningful as a thing associated to a class rather than an arbitrary instance (e.g. we refer to 'products', 'items', 'machines' and 'order forms').

Functional schema. Let T be a set of things, all possessing a common set of properties. A functional schema X_m = <M,F> is a non-empty set M and a finite sequence F = <F_1, ... F_n> of functions defined on M; that is, F_i:M$\rightarrow$$V_i$ is a domain of values, and each attribute function represents a property of the thing.

A functional schema, in other words, represents the set of attribute functions, associated with a class. Since, as discussed in Section 2, the OISR is incomplete in its nature, a functional schema may be partially defined. The functional schema concept is the basis for the definition of a *state*.

State. A state is the set of attribute values of a thing at a specific point in time. Consider a thing X described by a functional schema <M,F>. The function F_i:M→V_i is termed the *i*th state function of the thing. The set of values F(*t*) = <$F_1(t)$ ... $F_n(t)$> at a certain time point *t* represents the state of X at *t*. (Note, recall time is a part of M.)

Interaction. Thing X acts on thing Y if and only if the states that Y traverses for a given subset of M when X is present are different from the states that Y would traverse if the thing X did not exist. Things X and Y interact if at least one acts on the other.

Interaction emerges naturally from mutual properties of the things involved. Unlike intrinsic properties of each thing, which are modelled by its attributes, mutual properties are not necessarily modelled by attributes of the things involved. Rather, they may be observed by tracking interaction patterns. Interactions were defined in terms of state changes, or *events*, which are brought about by *transformations*. These concepts are defined as follows:

Transformation. A transformation is a mapping from one state to another one.

Event. An event is a change of state of a thing, effected via a transformation. Events may be further divided into *external* events and *internal* events.

External event. An event that arises in a thing by virtue of the action of some other thing.

Internal event. An event that arises in a thing by virtue of a transformation in that thing itself.

Stable state. A state in which a thing will remain unless forced to change by virtue of the action of another thing (i.e. by an external event).

Unstable state. A state in which transformations of the thing will occur until a stable state is reached.

Relating events to states of a thing, an external event may cause an unstable state, which would lead to a sequence of internal events that continues until a stable state is reached. This is the mechanism that drives a *process*. In the OISR model, this sequence of events is not restricted to changes in the states of a single thing. This leads us to an extension of the definition of stable and unstable states for a set of interacting things. We consider a state of a thing to be stable if and only if it does not trigger a sequence of events that eventually leads to another transformation in the same thing. Such a sequence can progress via state changes of several things, interacting with each other. Each thing might, in turn, cause an external event in another thing. Since an external event might trigger a response when the affected thing is in a stable state, the sequence of events progresses on its own. We can thus view each event in that sequence as internal to the set of things involved in the process. Hence, the distinction between internal and external events is related to a process rather than to a thing.

As discussed in Section 2, business processes in the OISR model are controlled by business rules. The analogous concepts in the BWW ontology are laws, which restrict states, events and transformations.

State law. A law which restricts the values of the attributes of a thing to a subset defined by natural or human laws.

Note, a state law reflects a restriction on the properties of a thing.

Transformation law. A law which defines the *lawful transformations* that can happen to a thing, that is, the lawful events of the thing.

While state laws and lawful transformations are required to define the OISR business rules, some constraints may be posed due to structural relations among the entities in the OISR model. These are handled by the following set of concepts.

Subclass. A subset of things X is a subclass of another set of things Y if and only if X is a class and a proper subset of Y. Subclasses are defined by a conjunction of the property used initially to define the class and an additional property of interest.

Composite thing. A thing is composite if and only if it consists of at least two things.

A property of a composite thing may be *emergent*, i.e. it not possessed by any of its components, otherwise it is *hereditary*.

Composition. The things comprising a composite thing are its composition.

Decomposition. A decomposition of a composite thing is a set of things, such that every component of the composite thing is either a member of this set or is included in the composition of one of the members.

3.3. Ontological Definitions of OISR Concepts

Using the BWW constructs we formalise the concepts of the OISR tuple, namely process, rule, information object and system service.

Definition 1. A *process* is a sequence of *events*, which is triggered by an *external event*, causing an *unstable state*. This is followed by *transformations* in one or more *things*, leading into a *stable state*.

The sequence of events captures the interactions among the things involved in the process. Consider the production example discussed in Section 2. It is important to understand that the processes we refer to are the administrative processes related to production management, not the production process itself. Thus, the things involved reflect entities that are used, created and modified by such processes. In particular, the things involved in the process are *production order*, *production schedule*, *inventory*, *order documents*, *production report* and *costing report*. The external event that triggers the process is the creation of the production order in an unstable state *planned*, and the stable state reached is a *closed* order. Note that the stable state that ends the process is only of the thing whose unstable state has triggered the process, i.e. *production order*. Other things, such as *inventory*, may remain in an unstable state to be resolved through other processes.

Definition 2. *An information object* is a *thing* associated to a *class.*

By relating a thing to a class we obtain the definition of its functional schema. In our production example, each occurrence of the process relates to a single production order. Each such order is characterised by the set of attributes that represent the set of properties of the production order class members.

As discussed in Section 2, there are two types of business rules: structural constraints and rules of dynamics. Each is defined separately.

Definition 3. *A structural constraint* is a *state law*.
State laws, which restrict the possible states of a thing, may be influenced by the composition of a composite thing or by the fact that a class is a subclass of another one. These ontological concepts may therefore be needed when defining structural constraints. In the production example, structural constraints specify the lawful values of *production order status*.

Definition 4. *A rule of dynamics* specifies a *lawful transformation*.

The difference between the types of rules of dynamics discussed in Section 2 is in their reference point, which may be:

1. the event which triggers the transformation (*stimulus / response rule*);
2. the transformation as a mapping from state to state (*transition rule*);
3. the state subsequent to the transformation (*derivation rule*).

Considering the production example, the rule of dynamics specifies the lawful transformations of *production order status*.

According to the OISR model presented in Section 2, system services are the responses of the IS to messages, generated in the course of the business processes. The messages may be reporting a change in the real world to be recorded, in which case the system service is a message that confirms the recording of this change. Alternatively, they can be requests for some processing to be performed, resulting in the generation of a new information object. The new information object can materialise either a query result or a message about the changes made to the data as a result of the processing performed by the IS (e.g. calculated costs or generated planned production orders). The messages provided by the system in either case are information objects, i.e. things, which are created for this purpose. The system service, therefore, is the creation of these things through a transformation.

Definition 5. *A system service* is a *transformation* that creates a *thing*.

This definition provides the ontological description of a system service, but it is not a unique identification of one. Every system service is a transformation that creates a thing but there may be such transformations that are not system services. The difference is that system services reside in the interface between the information system and its environment; namely, they are made available to business processes. However, the OISR model regards the real world as it should be reflected in an information system without modelling the system itself. Hence, the ‘interface’ has no specific meaning in ontological view of OISR. Therefore, an ontological definition is able to provide necessary conditions for identifying a system service, but not sufficient ones.

In the production example presented in Section 2, the creation of *production report* is a system service, provided in response to reported changes in the real world. The creation of *Order documents* is a response to an information request, and the creation of *Costing report* is a result of an internal computation performed when closing an order.

3.4. Evaluation Framework

Based on the definitions of the OISR tuple elements, we introduce an evaluation framework, which is a specialisation of the BWW evaluation framework [15,16] that is aimed at assessing the expressive power of modelling methods to be applied for OISR representation. The

Table 1. OISR ontological evaluation framework

Ontological construct	Explanation
Thing	A thing is the elementary unit in the ontological model. The real world is made up of things. A composite thing is a combination of at least two things
Attribute	Things possess properties. A property is modelled via an *attribute* function that maps the thing into some value
Class	A class is a set of things that possess a common property
State	The vector of values for all attribute functions of a thing is the state of the thing
State law	A state law restricts the values of the properties of a thing to a subset defined by natural or human laws
Interaction	A thing acts on another thing if its existence affects the state of the other thing. The two things are said to be coupled or interact
Event	An event is a change of state of a thing, effected via a transformation (see below)
Transformation	A transformation is a mapping from one state to another one
Lawful transformation	A lawful transformation defines which events in a thing are lawful
External event	An external event is an event that triggers a process
Internal event	An internal event is an event that occurs in the course of a process
Stable state / Unstable state	A state of a thing is stable if and only if it does not trigger a sequence of events that will eventually lead to another transformation in the same thing. Otherwise it is unstable
Subclass	A subset of things X is a subclass of another set of things Y if and only if X is a proper subset of Y
Composition	The things in a composite thing are its composition
Decomposition	A decomposition of a composite thing is a set of things such that every component of the composite thing is either a member of this set or is included in the composition of one of the members

difference between the OISR evaluation framework and the original BWW framework lies in the set of ontological constructs that need to be represented in a modelling method in order for it to be ontologically complete. Our ontological construct set, presented in Table 1, is a subset of the original BWW set, necessary and sufficient for OISR representation. Note that some of the ontological constructs discussed above, such as property and functional schema, are not necessary for OISR representation. They were introduced in order to provide a basis for the understanding of other constructs, such as attribute and state.

Regarding ontological clarity, this framework follows the original BWW framework and identifies three types of clarity deficiencies: construct overload, construct redundancy and construct excess.

4. Applying the Ontological Evaluation Framework

In this section we demonstrate the evaluation framework by applying it to the Object-Process Methodology (OPM) in order to assess OPM's appropriateness for OISR modelling. OPM is applied as a modelling platform in the requirement-driven ERP customisation process described in Soffer [7]. The usefulness and generality of the framework are further demonstrated by using it for comparing OPM and ARIS. ARIS is not usually regarded as a conceptual modelling tool, as it is less rigorous and formal than other tools. However, since it is the modelling tool embedded in SAP R/3, and is common in other ERP implementations, its evaluation as an OISR modelling tool is of interest. The evaluation of ARIS is partially based on an earlier evaluation presented in Green and Rosemann [3].

4.1. Object-Process Methodology

Object-Process Methodology, described in detail elsewhere [17,33–35], employs a single graphic model, a set of *Object-Process Diagrams* (OPDs), to capture both the structural and the dynamic aspects of a system. OPM's fundamental building blocks are two equally important classes of entities: *objects* and *processes*, which are related by *structural* and *procedural links*. The processes of OPM generalise business processes, activities and transformations.

Most of the object-oriented modelling methods and enterprise modelling methods require the use of a set of models, 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, as a single model for the structural and dynamic system aspects. This eliminates the model multiplicity problem of these modelling methods [36]. The singularity of the model eliminates the effort required to integrate the various

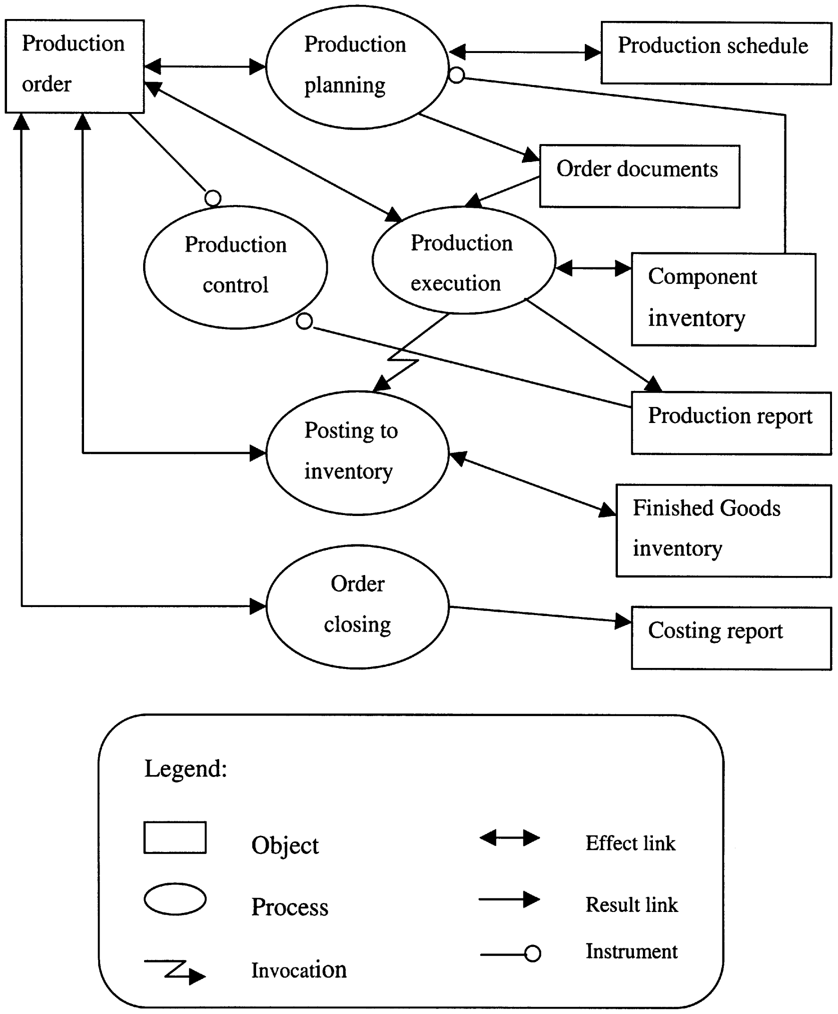

Fig. 2. An OPM representation of the production example.

views into a coherent system model and to keep consistency among them. Thus, in OPM all aspects of the requirements are integrated into one coherent representation. In an OPD, there is a *timeline* that flows from the top of the diagram to its bottom, so the sequence of processes is represented by their vertical order, as well as by event-triggered processes. The system services are expressed by result links between processes and objects. Information objects, their structure, attributes and relations are also integrated into the OPDs. Business rules, both structural constraints and rules of dynamics, are integral parts of an OPD.

While using a single model representation, OPM keeps simplicity through a set of scaling mechanisms that control the *visibility* of the system details. It allows a top-down analysis by unfolding objects and zooming into processes in increasingly detailed OPDs, constructing a hierarchical OPD set which specifies the system's structure and behaviour at various levels of detail.

To illustrate the OPM representation and modelling, the production environment example given in Section 2 is modelled in Figs 2, 3 and 4.

Figure 2 is an OPD of the production process, representing the activities (processes), objects and various links among them. The *production planning* activity, for example, is related by an effect link with the objects *Production order* and *Production schedule*, uses the object *Component inventory* and creates the *Order documents* as a system service. Each of the entities (objects and processes) in the OPD may be unfolded in other OPDs to reveal more details. Figure 3 is an unfolding of the *Production order* attribute *Status*, and is a model of the rule of dynamics

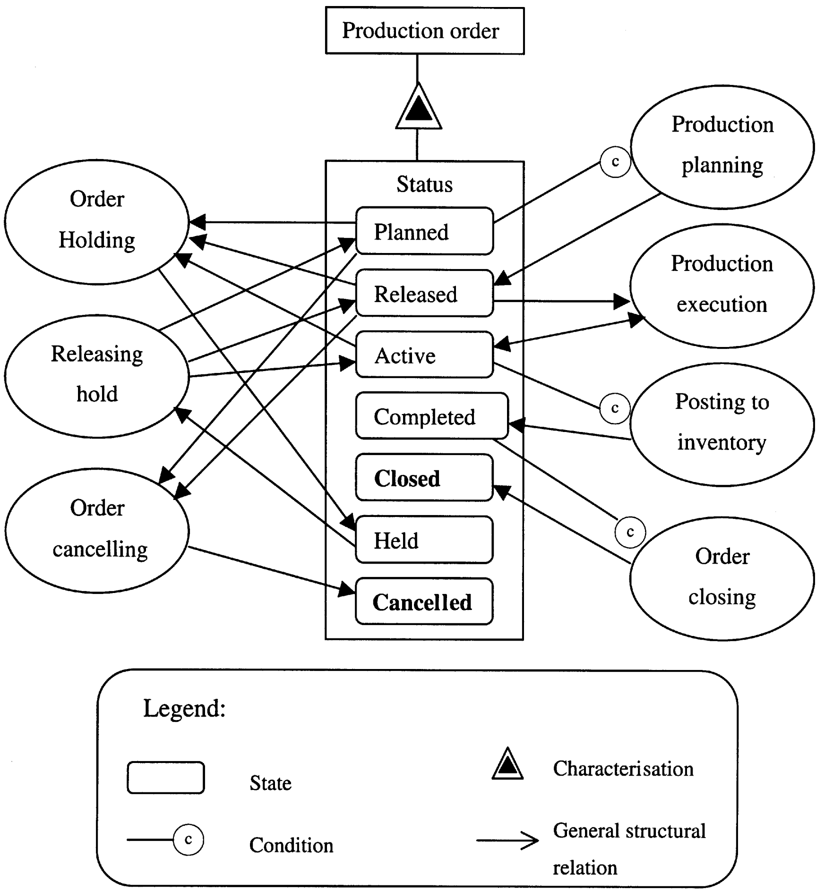

Fig. 3. An OPD representing the lawful transformations of production order status.

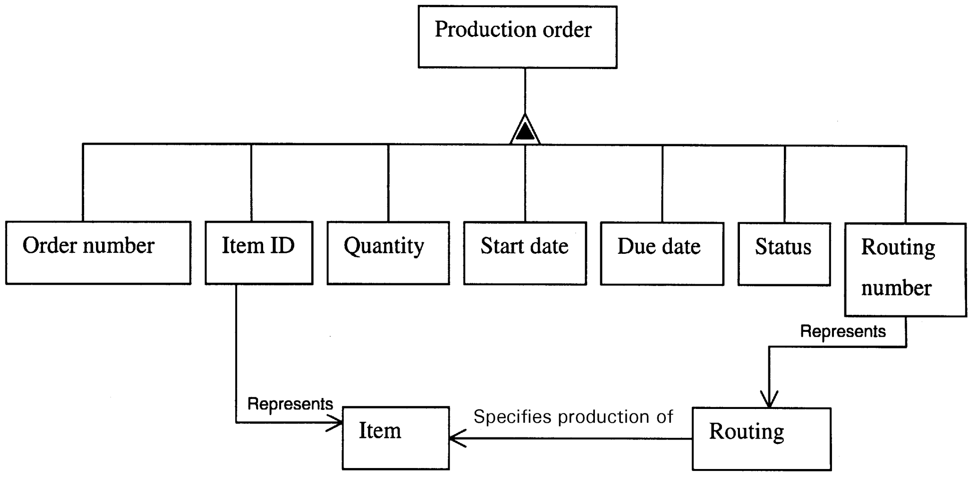

Fig. 4. An OPD unfolding the attributes of production order.

specifying its lawful transformations. States in bold type in the OPD denote persistent states (e.g., order closed or canceled).

The attribute set of *production order* is given in Fig. 4. The attributes *item ID* and *routing number* represent relations to the objects *item* and *routing*.

Table 2. OPM representation of OISR ontological constructs

Ontological construct	OPM representation
Thing	Instance
Attribute	An attribute is an object related to another object by a characterisation link
Class	Object class
State	State (separately modelled for each attribute)
State law	A state law is a specification of the possible states of an object, including distinction of transient and persistent states
Interaction	The sequence <Object A /state → condition / event / effect / instrument / consumption link → process → effect / result link → Object B/ state> expresses interaction between objects A and B
Event	The event of changing state A to state B is represented by the sequence <State A → consumption link → process → result link → state B>
Transformation	Process
Lawful transformation	A set of objects / states linked to a process by a condition / event / effect / consumption / instrument link. The process is linked to another set of objects / states by an effect / result link
External event	Object / state → event link → process
Internal event	Process → effect / result link → object / state
Stable state	A persistent state, or any other state, which is not unstable (see below)
Unstable state	State A in the sequence <state A → condition / event / consumption link → process → result link → state B> is an unstable state
Subclass	An object class, which is related to another class by a specialisation link
Composition	Composition and decomposition are given by the sequence <object → aggregation link → set of objects>.
Decomposition	The composite thing is linked at the vertex of the aggregation symbol and its components at the bottom

4.2. OPM Evaluation

We apply our ontological evaluation framework to evaluate OPM's adequacy as a modelling language for OISR. The framework evaluates the ontological completeness of OPM with respect to the specialised OISR ontology, and its ontological clarity in terms of construct overload, construct redundancy and construct excess. Table 2 provides the OPM representation of the ontological constructs required for OISR modelling. Alternative representation options are separated in the table by a '/'.

As seen in Table 2, OPM is ontologically complete for the sake of OISR modelling. However, examining the ontological clarity reveals some deficiencies:

- *Construct overload.* OPM modelling constructs do not precisely map into the BWW ontological constructs. Instead, the mapping is of combinations of OPM constructs through production rules (see [16]). Considering single modelling constructs, there is obviously a construct overload (for example, an OPM process may represent both a transformation and a sequence of events). But when considering the combinations of modelling symbols rather than each single symbol, we have not found any construct overload in OPM.
- *Construct redundancy.* Table 2 reveals some cases of construct redundancy. For example: the construct 'lawful transformation' may be expressed by several OPM combinations, depending on the type of link between the objects and the process. This is because a lawful transformation always involves OPM's *transformation* links, but may also be triggered by OPM *event* links, or restricted by OPM's *enabling* links, which define its preconditions. OPM, which attempts to capture the dynamics of the world and classify events in a detailed manner, differentiates between types of transformation links (effect, result and consumption), enabling links (condition, instrument and agent), and event links (event, invocation, exception). Indeed, this may lead to several alternative ways of modelling a domain.
- *Construct excess.* OPM employs some modelling constructs that do not map into ontological constructs by themselves or in combination with other modelling constructs. These include agent and invocation links.

The agent link denotes an enabling link between an intelligent physical object (usually human) and a process. The agent link stands for the human resource modelling, which, as discussed in Section 2, is not a part of the OISR. Therefore applying the agent link is not recommended in OISR modelling, as it represents a thing that is out of the modelled domain (see [16]). The invocation link denotes an immediate triggering of a process by the completion of another process. What actually triggers the process is a transformation of some object – the creation, consumption or change in the state of that object, which is not modelled explicitly. The invocation link is a shorthand notation that bypasses this object transformation when it is considered unimportant or irrelevant to the system. When that transformation is modelled explicitly, the invocation link is not required.

Despite the deficiencies in OPM's ontological clarity, there are two reasons for applying it for OISR modelling. The first one is its ontological completeness and the second one is its single model representation, which is convenient when attempting to match between the OISR model and a system model. The lack of ontological clarity may be overcome by defining clearer modelling rules that would prevent ambiguity. For example, such a rule may require that whenever a process B is triggered by the completion of process A, the sequence $<$process A $\rightarrow$ result link $\rightarrow$ object / state $\rightarrow$ event link $\rightarrow$ process B$>$ should be used rather than an invocation link.

4.3. Ontological Comparison of OPM and ARIS

As noted in Section 3, the BWW evaluation framework has been applied to several modelling languages [3,15,16]. ARIS, presented in detail in Scheer [19], has become popular as the modelling language of the SAP R/3 reference model, which serves for the SAP 'blue-printing' process [4] described in Section 1. Therefore, it is interesting to compare the OISR ontological evaluation of ARIS to that of OPM.

ARIS employs five related *views*, the central of which is the *process view*. Processes are modelled using the EPC (event-driven process chains) grammar, which represents processes by bipartite graphs with active nodes (functions) and passive nodes (events). In addition, ARIS has an *organisation view*, representing organisational units and resources; a *function view*, representing the activities performed through the processes; a *data view*, which addresses the data manipulated through the processes represented by ER diagrams; and an *output view*, which represents physical outputs and output data.

The ontological evaluation of ARIS [3] has found it ontologically incomplete regarding the original BWW framework. That work had suggested that the findings might have been different if, regarding the purpose of modelling, a subset of the framework was applied. Using the evaluation of ARIS in Green and Rosemann [3] as a basis, we apply our OISR subset of ontological constructs to evaluate the expressive power of ARIS for OISR modelling. The ARIS views considered here are the process view, the data view and a part of the output view. We disregard the organisation view and the physical part of the output view since they are out of the OISR scope (see Section 2). The function view is disregarded since it has been found ontologically redundant in Green and Rosemann [3]. Applying the specialised OISR ontology, ARIS is still ontologically incomplete, although to a lesser extent compared with the original BWW evaluation. While discussing these findings, we focus on the consequences they bear on the OISR model.

Composition and *decomposition* of things, which are not explicitly represented in ARIS, can be represented through an ordinary structural relation in the ARIS data view. Therefore this may be resolved easily without really affecting the information captured in a model.

Thing, which should be the basic modelled unit, has no representation. Surprisingly, this is not very harmful, bearing in mind that classes represent their instances. This can therefore be considered a clarity deficiency rather than ontological incompleteness.

Unstable state is not identified explicitly. We may assume that each state within a process in ARIS is unstable and the only stable states are in the end of a process, but this is not consistent with our definitions of stable and unstable states. Therefore, the ARIS model does not provide for unambiguous identification of unstable states.

State law and *lawful transformation* are not represented in ARIS. The absence of these two constructs makes it impossible to represent both structural constraints and rules of dynamics in ARIS. This crucial deficiency prevents the representation of a basic element in the OISR tuple.

In conclusion, the ontological incompleteness of ARIS makes it inappropriate for OISR representation, as opposed to OPM, which is ontologically complete for this purpose.

Considering ontological clarity, examples of all three deficiencies (construct overload, construct redundancy and construct excess) were found in ARIS [3]. Since similar clarity deficiencies exist also in OPM, no conclusive superiority can be identified in this sense. Table 3 is a summary of the ontological completeness and clarity comparison between OPM and ARIS. The ontological completeness assessment in Table 3 is expressed regarding each of the elements of the OISR tuple.

In view of the findings of the ontological incompleteness of ARIS, one may wonder how this affects the SAP R/3 reference model, for which ARIS is used as a modelling platform. The answer is that the ARIS models of SAP represent the system capabilities in an implementation process, which, as explained in Section 1, does not explicitly account for the requirements at all. Since the system capabilities model is assumed to be complete, the business rules are not important, and the

Table 3. A comparison of OPM and ARIS

Evaluation	Issue	OPM	ARIS
Ontological completeness	Process model	Complete	Unstable state missing
	System services	Complete	Complete
	Business rules	Complete	Lacking essential constructs: state law and lawful transformation.
	Information objects	Complete	Thing represented only by its association to a class.
Ontological clarity	Construct overload	Exists only when regarding each construct separately	Exists
	Construct redundancy	Exists	Exists
	Construct excess	Exists	Exists only when regarding each construct separately

absence of state law and lawful transformation, which are crucial for the OISR model, is not critical in the SAP models.

5. Conclusions

This paper has addressed fundamental issues related to modelling of requirements for off-the-shelf information systems (OISR). While most of the work in this area suggests a specific method for describing the requirements or the requirements definition process, we do not propose here another method. Rather, our intention has been to provide a method for evaluating modelling techniques and selecting an appropriate one for OISR representation.

A generic model for understanding the OISR domain has been introduced, resulting in an OISR framework. The framework specifies four types of elements to be included in an OISR specification: business processes, business rules, information objects and system services. The BWW ontological framework provided a sound and precise theoretical basis for the formal definitions of these elements. As a result, a subset of ontological constructs relevant for OISR modelling has been identified. This subset constitutes a specialisation of the BWW for OISR modelling, and serves as a framework for evaluating modelling languages to be applied for this purpose. The evaluation framework has been demonstrated by applying it to the OPM, finding it to be ontologically complete but revealing some clarity deficiencies. The effectiveness of the evaluation framework has been further demonstrated by applying it for a comparison between OPM and ARIS. ARIS has been found to be ontologically incomplete, with severe ontological deficiencies. This work therefore shows that OPM is superior to ARIS for the purpose of OISR modelling.

The contribution of the OISR framework is in providing a sound theoretical basis for the OISR requirement engineering process and requirements specification. The resulting evaluation framework provides a tool specialised for this purpose, which can be applied whenever a modelling method should be selected. The tool provides information about the capability of the selected method to represent the entire suite of OISR elements, thereby satisfying a necessary condition for the requirements model to be effective.

Several directions for future research are possible. An OISR RE process, which seeks to identify the details of the four OISR elements in a given situation, may be developed on the basis of the OISR framework. The ontological evaluation framework can be applied to other modelling languages in order to assess their suitability for OISR modelling and gain insights into the effectiveness of the evaluation framework itself. Regarding OPM, the clarity deficiencies identified here should be worked out through future research, possibly by the development of appropriate modelling rules.

References

1. Davenport TH. Putting the enterprise into the enterprise system. Harvard Business Rev 1998;121–131
2. Holland CP, Light B. A critical success factors model for ERP implementations. IEEE Software 1999;16:30–35
3. Green P, Rosemann M. Integrated process modeling: an ontological evaluation. Inform Syst 2000;25:73–87
4. Curran TA, Ladd A. SAP R/3 Business blueprint: understanding enterprise supply chain management (2nd edn). Prentice-Hall, Englewood Cliffs, NJ, 1999
5. Daneva M. 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. Soffer P. Process and language requirement for ERP modeling. In: EMMSAD'01, Proceedings of the sixth CaiSE/IFIP8.1 international workshop on evaluation of modeling methods in systems analysis and design, Interlaken, Switzerland, 2001, pp X-1–X-8

7. Soffer P. A requirement driven approach to business application implementation. In: Proceedings of USP roundtable conference 2000 on E-enterprise structure and analysis, Haifa, Israel, 2000 (on CD)
8. Mylopoulos J. Conceptual modeling and Telos. In: Loucopoulos P and Zicari R (eds). Conceptual modeling, databases, and CASE. Wiley, New York, 1992, pp 49–68
9. Rolland C. Requirements engineering for COTS based systems. Inform Software Technol 1999;41:985–990
10. Daneva M. Practical reuse measurement In: Wangler B, Bergman L (eds). Advanced information systems engineering. Lecture notes in computer science 1789. Springer, Berlin, 2000, pp 309–325
11. Maiden NA, Ncube C. Acquiring COTS software selection requirements. IEEE Software 1998;15(2):46–56
12. Ncube C, Maiden NAM. Guiding parallel requirements acquisition and COTS software selection. In: RE'99: Proceedings of the IEEE international symposium on requirements engineering. IEEE Press, Washington, DC, 1999, pp 133–140
13. Feblowitz MD, Greenspan SJ. Scenario-based analysis of COTS acquisition impacts. Requirements Eng 1998;3:182–201
14. Finkelstein A, Spanoudakis G, Ryan M. Software package requirements and procurement. In: Proceedings of the eighth international workshop on software specification and design. IEEE Press, Washington, DC, 1996 pp 141–145
15. Wand Y, Weber R. An ontological evaluation of systems analysis and design methods. In: Falkenberg ED, Lindgreen P (eds). Information system concepts: an in-depth analysis. North-Holland, Amsterdam, 1989, pp 79–107
16. Wand Y, Weber R. On the ontological expressiveness of information systems analysis and design grammars. J Inform Syst 1993;3:217–237
17. Dori D. Object process methodology: a holistic systems paradigm. Springer, Berlin, 2001 (in press)
18. Rolland C, Prakash N. Bridging the gap between organizational needs and ERP functionality. Requirements Eng 2000;41(3):180–193
19. Scheer AW. ARIS: business process frameworks. Springer, Berlin, 1999
20. Bubenko JA, Brash D, Stirna J. EKD user guide. Dept. of Computer and Systems Science, KTH and Stockholm University, Sweden, ESPRIT Project No. 22927, ELECTRA, Document, 1998
21. Nilsson BE. On why to model what and how: concepts and architecture for change. In: Nilsson AG, Tolis C, Nellborn C (eds). Perspectives on business modelling. Springer, Berlin, 1999, pp 269–303
22. Davis AM. Software requirements: objects, functions and states. Prentice-Hall, Englewood Cliffs, NJ, 1993
23. Bohem B. Requirements that handle IKIWISI, COTS, and rapid change. Computer 2000;33(7):99–102
24. Davenport TH. Process innovation: reengineering work through information technology. Harvard Business School Press, Boston, MA, 1993
25. Eriksson HE, Penker M. Business modeling with UML: business patterns at work. Wiley, New York, 2000
26. Rosca D, Greenspan S, Feblowitz M, Wild C. A decision making methodology in support of business rules lifecycle. In: RE'97: Proceedings of the third IEEE international symposium on requirements engineering. IEEE Press, Washington, DC, 1997 pp 236–246
27. Wand Y, Storey VC, Weber R. An ontological analysis of the relationship construct in conceptual modeling. ACM Transactions on Database Systems 1999;24:494–528
28. Bunge M. Treatise on basic philosophy. Vol. 3: Ontology I: the furniture of the world. Reidel, Boston, MA, 1977; Vol. 4: Ontology II: a world of systems. Reidel, Boston, MA, 1979
29. Chidamber SR, Kemerer CF. A metrics suite for object oriented design. IEEE Trans Software Eng 1994;20(6):476–493
30. Moser S. Metamodels for object-oriented systems: a proposition of metamodels describing object-oriented systems at consecutive levels of abstraction. Software Concepts Tool 1995;16(2):63–80
31. Wand Y, Woo C. Ontology-based rules for object-oriented enterprise modeling. Working paper 1999, Faculty of Commerce and Business Administration, University of British Columbia
32. Wand Y, Wang RY. Anchoring data quality dimensions in ontological foundations. Commun ACM 1996; 39(11):86–95
33. Dori D. Object process analysis: maintaining the balance between system structure and behavior. J Logic Comput 1995:227–249
34. Dori D, Goodman M. From object process analysis to object process design. Ann Software Eng 1996;2:20–25
35. Peleg M, Dori D. Extending the object-process methodology to handle real time systems. J Object Oriented Programming 1999;11(8):53–58
36. Peleg M, Dori D. The model multiplicity problem: experimenting with real-time specification methods. IEEE Trans Software Eng 2000;26(8):742–759

Appendix: a list of BWW ontological constructs

Ontological construct	BWW explanation
Thing	A thing is the elementary unit in the ontological model. The real world is made up of things. A composite thing may be made up of other things
Property	Things possess properties. A property is modelled via an attribute function that maps the thing into some value
Class	A class is a set of things that possess a common property
Subclass	A subset of a class, defined by a conjunction of properties
Functional schema	The set of attribute functions associated with a class
State	The vector of values for all attribute functions of a thing is the state of the thing
State law	A state law restricts the values of the properties of a thing to a subset defined by natural or human laws
Event	An event is a change of state of a thing, effected via a transformation (see below)
Transformation	A transformation is a mapping from one state to another one
Lawful transformation	A lawful transformation defines which events in a thing are lawful
Coupling	A thing acts on another thing if its existence affects the state of the other thing. The two things are said to be coupled or interact
Composition	The things in a composite thing are its composition
Decomposition	A decomposition of a composite thing is a set of things such that every component of the composite thing is either a member of this set or is included in the composition of one of the members
Level structure	Defines a partial order over the things in a decomposition to show which things are components of other things
External event	An event that arises in a thing, subsystem or system by virtue of the action of some thing in the environment on the thing, subsystem or system
Internal event	An event that arises in a thing, subsystem or system by virtue of lawful transformations in the thing
Stable state	A state in which a thing, subsystem or a system will remain unless forced to change by virtue of the action of a thing in the environment (an external event)
Unstable state	A state that will be changed into another state by virtue of the action of transformations in the system
Well defined event	An event in which the subsequent state can always be predicted given that the prior state is known
Poorly defined event	An event in which the subsequent state cannot be predicted given that the prior state is known
Kind	A set of things that possess two or more common properties
Conceivable state space	The set of all states that the thing might ever assume
Lawful state space	Set of states of a thing that complies with the state law of the thing
Event space	The set of all possible events that can occur in a thing
Lawful event space	The set of all events in a thing that are lawful
History	The chronologically ordered states of a thing
System	A set of things so that for any bipartitioning of it coupling still exists among the things in the two sets
System environment	Things that are not in the system but interact with things in the system
System structure	The set of couplings in a system and its environment
Subsystem	A system whose composition and structure are subsets of the composition and structure of another system