# Enterprise Resource Planning Requirements Process: The Need for Semantic Verification

**Peter Bollen**

**Abstract** This paper reviews the relevance of requirements determination in the commercial-off-the-shelf (COTS) enterprise software era. State-of-the-art requirements determination methods must contain, facilities for allowing semantic verification. We will introduce a conceptual modelling approach that fulfills this requirement and that can be used in the process of ERP configuration and requirements determination in general. The fact-based conceptual modelling approach that we will use in this paper is CogNIAM.

## 1 Introduction

The London Stock exchange automated trading system Taurus, had to be withdrawn before it ever was used [38]. The failure of National Insurance Recording System in England lead to tax overpayments by 800,000 people [40]. These are examples of organizations that have become victims of an unsatisfactory user requirements determination process. Unsatisfactory user requirements determination is one of the most prevalent reasons for faulty information systems or information systems that turn out to be overdue and too costly. Requirements determination is the least well-defined phase in the systems development process [17] and: "has been widely recognized as the most difficult activity of information systems development." ([10], p. 224). Failures in the requirements determination process represent one of the leading causes of system failure: "Given an appropriate design, most information systems departments can successfully implement a system. The big problem is correctly determining information requirements and designing the right system." ([45], p. 52). Many IS failures can be attributed to a

P. Bollen (✉)
Maastricht University, Maastricht, Netherlands
e-mail: p.bollen@maastrichtuniversity.nl

F. Piazolo and M. Felderer (eds.), *Innovation and Future of Enterprise Information Systems*, Lecture Notes in Information Systems and Organisation 4,
DOI: 10.1007/978-3-642-37021-2_6, 

lack of clear and specific information requirements." ([11], p. 118). "The major reason that IS does not meet user expectation is a failure to obtain the correct and complete set of user requirements." ([47], p. 412) "Often, much of post-delivery maintenance work can be traced to requirements which had been poorly or falsely described in the system requirements specification (SRS), or were missed altogether." ([22], p. 161). Errors in the requirements specification caused by a faulty requirements determination process can remain latent until the later stages in the IS development process ([43], p. 666) and will cost a manifold to fix in these later stages [3, 4].

The information systems development market place changed in the early nineties of the last century when the *product software*-suppliers, e.g. MFG/PRO, IFS, ORACLE, SAP, BAAN, Marshal, Peoplesoft ([1], p. 369, [34], p. 387–389) started to sell their enterprise solutions on the waves of the Business Process Reengineering (BPR) sea [15, 20]. These product software solutions, promised to solve many problems that were caused by the software crisis and were considered to be an attractive investment option in ICT for the large (Fortune 500) companies. The implementation of, for example, ERP systems in a company, however, in most cases meant that the business process had to be reengineered or redesigned to fit the 'reference-model' that underlies the ERP package. This reengineering process turned out to be feasible for standard application functionality, for example, accounting, payroll, human resource management, inventory control. However, company-specific, functionality remained a problem in the first generation ERP-solutions. The second generation ERP-solutions, however, tried to redefine the concept of company-specific functionality, by developing 'standardized' software solutions for specific 'branches', for example, health-care, utilities, retail and so forth [8]. An example is Customer–Relationship Management (CRM) by Siebel [26]. The development of the additional functionality in these second generation ERP systems, implied, in many cases, additional reengineering efforts on these specific application domains before an implementation could take place. In spite of the availability of the second –generation ERP solutions, many companies needed customized modules and interfaces that allows them to support the specific parts of their business [37]. In the last decade firms have added modules that address inter-firm activities [25] and cross-organizational coordination [13]. We will call these ERP-implementations third generation ERP systems.

## 2 Roles in Requirements Determination

The improvement of the requirements determination processes for enterprise applications is still a relevant research subject within the field of business information systems because improving the state of the art in requirements determination methods to be applied in these requirements determination processes will have the following impact on organizations:

- It will enable them to express their (information) requirements using less (human) resources (more efficient).
- It will enable them to express their (information) requirements in a more precise, consistent and complete way.

If we now look back at the development in the development of (business) information systems over the past 60 years we can distinguish a number of roles in the requirements determination process:

1. The role of *user* or (business *domain expert*), these roles involve the knowledge of the business domain as it exist with the knowledge workers in the enterprise, for example the knowledge on how to process an invoice or how to approve a loan.
2. The role of the *analyst*, this role involves the knowledge on how to elicitate the knowledge of a knowledge worker in the focal enterprise in a format that can be used by a developer to develop an application system. The result of the work of the analyst we will call a requirements specification.
3. The role of the systems *developer,* this role involves the knowledge on how to transform an information systems specification into a working information system that complies with the *functional requirements* as embedded in the *requirements specification.*

In Fig. 1 we have illustrated the general relationships between the aforementioned roles.

The extent in which the role of an analyst can be played perfectly in the requirements determination process depends upon the availability of 'a way of working', 'a way of modeling' and 'a way of controlling' ([46], p. 14). A *way of modeling* refers to the model types that are required: "A way of modeling structures the models which can be used in information systems development. Several models are usually required for problem specification and solution in the application area" ([46], p. 15).

A *way of working* or a *prescriptive process model* [32]: "is a description of processes at the type level. It defines how to use the concepts defined within a product Model. A prescriptive Process Model is used to describe 'how things must/should/could be done." ([32], p. 62). The way of working refers to the process-oriented view of information system development, whereas the way of modeling refers to the product-oriented view of information system development.

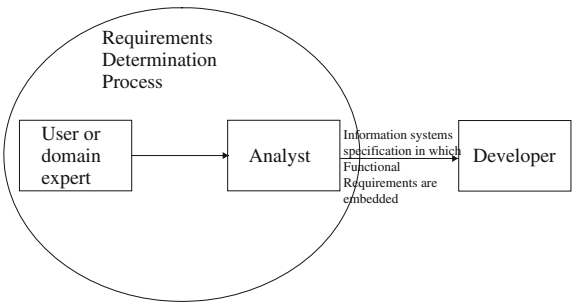


**Fig. 1** The roles in the requirements determination process in general

## 2.1 Sub-Steps in Requirements Determination

The general requirements determination process from Sect. 2.1 is generally viewed as consisting of three steps [10, 21]:

1. Information gathering (or requirements elicitation), during which an analyst elicitates requirements from (a) user (s) or domain expert(s),
2. Representation (or requirements specification), in which those requirements are specified in some modeling language by the analyst,
3. Verification (or requirements validation) in which the analyst verifies the correctness of these requirements with the user.

If we consider the aforementioned steps in the requirements determination process, then we can state that the scientific research on these steps has not exclusively taken place in the fields of Fig. 1. For example, with respect to the step information gathering or requirements elicitation, substantial research has taken place within the field of Knowledge Engineering [2] leading to knowledge acquisition methods like KADS [9]. These approaches are primarily directed at 'knowledge' green fields, i.e. those application domains that were generally considered to contain predominantly 'tacit' knowledge and these approaches were not developed for business application domains in which available knowledge has to be categorized and at most be made explicit.

With respect to the second step in the general requirements determination process: representation or requirements specification we can conclude that the definition of requirements specification languages has been a major research stream within the conceptual modeling and IS fields of study that deal with requirements determination. A Major data-oriented 'language family' in this respect is the (extended) ER language [12, 40]. As an example of a 'process-oriented' specification language we can consider Data Flow Diagrams (DFD's) [39] or Activity Diagrams (A-schemas) in ISAC [24].

With respect to the third step: requirements validation (or verification) we must make a distinction into *semantic verification* and *syntactic verification*. Semantic verification is the type of validation that is concerned with the capturing of the 'right' domain requirements in terms of the extent in which what the analyst records is what the domain user intends to express. Dullea et al. ([16], p. 171–172) define the concept semantic validity as follows: "An entity-relationship diagram is semantically valid only when each and every relationship exactly represents the modeler's concept of the problem domain". We will generalize this concept to every requirements determination method and more importantly, we will extend this concept beyond the modeler's interpretation of the application domain to the user's interpretation for the application domain, into our definition of a semantic correct specification. The outcome of a requirements determination process expressed in some specification language, therefore, should always be a semantically correct specification.

Syntactic verification, merely deals with the compliance of a specific application specification to the modeling rules that are contained in the meta-model of the specification language. We must be aware of the possibility that a semantic incorrect specification can be syntactically correct in any given situation.

The steps in the requirements determination process that cover the semantic verification are missing in the existing requirements determination methods for management information systems or business information systems ([18], p. 376). In this paper we will introduce a requirements determination method in which the semantic verification is incorporated in an explicit way.

## 2.2 Eras in Requirements Determination

In the 1970s a clear separation took place between the functional requirements and the way in which these functional requirements were coded in a specific implementation technology [42]. The distinction between an *information analyst* and *systems developer* emerged. The application of information systems development methodologies was aimed at the creation of 'tailor-made' information systems in which the needs of the domain users served as input.

In the ERP era (1990 and onwards) the roles of the user (or domain expert), analyst and developer were becoming more iterative instead of the linear sequence in which those roles were performed in the 1970 and 1980s. Because the implementation of ERP-systems usually is linked to business process redesign [14, 33] or a business process reengineering exercise ([35], p. 72), the role of the user or domain expert becomes more complex. In cooperation with the ERP-analyst the domain expert has to evaluate a number of proposed ways of working that will be supported by the specific ERP system in the company ([36], p. 183).

The roles that we have depicted in Fig. 1 have deliberately different names in Fig. 2, because an ERP analyst is not only modeling the user requirement of a proposed (or 'to-be') business process but in addition has to confront the user or domain expert with the different possible (or 'to-be') business logics or best practices that are available in the chosen ERP system. The business, therefore, is expected to select and adapt a reference model, based on available solutions with minimal changes and leaving no record of the enterprise's original requirements ([36], p. 183). On the other hand, even when they decide to implement an ERP system some organizations (for example Reebok) still choose to customize ([23], p. 417) and enhance the standard functionality of the ERP system [37]. We remark, that the focus of the requirements determination in this article is on the conceptualization of the information and decision rules that must be contained in an (ERP) application. The available functionality in the templates of an ERP product, however determines the 'boundaries of practice' for the organization that wants to implement an ERP system [44].

## 3 A Method for ERP Requirements Determination and Semantic Verification

In this section we will introduce a conceptual modeling approach that has proven successfully for the creation of (IS)-specifications that require a built-in semantic verification process. This approach is called the fact-based conceptual modeling approach and has evolved over 35 years from an architecture for databases [27] towards a versatile methodology for specifying knowledge bases, business rules and business processes [28]. Currently the fact-based approach is embedded in two main methods : Object Role Modeling (ORM-2) [19] and CogNIAM [28, 29]. Both methods take a single fact encoding modeling construct as a starting point. Both methods also apply a rigid 'way of working' for creating a conceptual schema for the data perspective. These methods, differ, however in terms of focus. In ORM-(2) a very large selection of constraints to model business rules in the data perspective has been introduced. In CogNIAM the focus is on a generic knowledge architecture that also covers the process and behavioural perspectives in conceptual modeling. In this article we will illustrate the application of the fact-based approach by using CogNIAM's knowledge architecture and notational convention for fact type diagrams. A theoretical foundation for CogNIAM can be found in [5–7].

In fact-based modeling we will use tangible documents or 'data-use cases' as a starting point for the modeling process. In most, if not all cases, a verbalizable knowledge source is a document that often is incomplete, informal, ambiguous, possibly redundant and possibly inconsistent. As a result of applying the fact-oriented knowledge extracting procedure (KEP), we will yield a document that only contains structured knowledge or a knowledge grammar which structures verbalizable knowledge into the following elements (*knowledge reference model(KRM)*) ([29], p. 766):

1. Knowledge domain sentences.
2. Definitions and naming conventions for concepts used in domain sentences.
3. Knowledge domain fact types including sentence group templates.

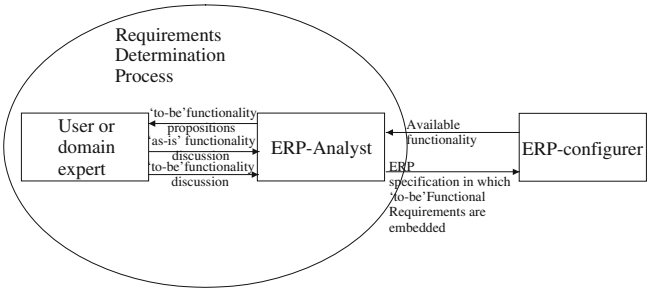


**Fig. 2** The roles in the ERP requirements determination process

4. Population state (transition) constraints or validation rules for the knowledge domain.
5. Derivation rules that specify *how* specific domain sentences can be derived from other domain sentences.
6. Rules that specify *what* fact instances can be inserted, updated or deleted.
7. Event rules that specify *when* a fact is derived from other facts or when a fact must be inserted, updated or deleted.

A KRM of a complete organization would contain hundreds, possibly thousands of concept definitions, naming conventions, fact types, population constraints, derivation rules and event rules. In [30] a successful application of fact-based modeling using the NIAM2007 method (a predecessor to CogNIAM) for requirements analysis is documented. In this project 125 fact types were derived, 1260 concept definitions were created, 704 (business rule) constraints were modeled and 20 derivation rules were derived.

The fact-based knowledge extracting procedure (KEP) specifies *how* we can transform a possibly informal, mostly incomplete, mostly undetermined, possibly redundant and possibly inconsistent description of business domain knowledge into the following classes: *informal comment*, *non-verbalizable knowledge* and *verbalizable knowledge* to be classified into types 1 through 7 of the KRM. We note that the knowledge extraction procedure that is needed to instantiate the elements 1 through 5 (of the KRM) is an extension of ORM's conceptual schema design procedure (CSDP) [19]. In business domains, furthermore, we can capture the dynamic aspects by defining the exchange rules (element 6 of the KRM) and the event rules (element 7 of the KRM).

## *3.1 Knowledge Domain Sentences*

The first element in the KRM is the group of sentences that represent an elementary fact (ground fact) in the domain. In our EOQ determination domain we encounter following example elementary sentences (that represent elementary (ground) facts):

```
The quantity of 1500 is the annual demand for the item ab3456
The quantity of 2500 is the annual demand for the item ab9876
The item ab3456 has an ordering cost of 25 euros
The item ab9876 has an ordering cost of 55 euros
The item ab3456 has a unit holding cost of 0,5 euros
The item ab9876 has a unit holding cost of 0,6 euros
The Item ab3456 has an economic order quantity 387
The Item df4567 has an economic order quantity 677
```

These sentences have a meaning for the people working in the logistics department. However, as soon as people communicate with people outside of this

department, additional semantics have to be captured. For example, it should be agreed upon to what time-frame the amount of holding costs for a product refers: a day, a week, a month, a year ?. These agreements should be part of a list of concept definitions in which it will be recorded that the term *unit holding cost* always refers to the *unit holding costs per year*. Another semantic issue that has to be defined in a list of concept definitions is the naming conventions for domain concepts. For example of what name class is 'ab3456' an instance? In practice when there is inter-organizational communication it must be crystal-clear which name classes can be used in communication. In case more two or more alternative name classes exist in the domain, it must be agreed upon to explicitly qualify the names used in the communication with the name classes:

```
The item with item code ab3456 has an ordering cost of 25
euros
The item with EAN bar code 8734576287465 has an ordering cost
of 55 euros
```

## 3.2 Concept Definitions and Naming Conventions for Concepts Used in Domain Sentences

In order to be able to grasp the meaning of sentences in the business domain it was argued that when two or more actors are involved in a communication process, semantic consistency can only be achieved if the different actors have the same understanding of concepts and naming conventions. This will be established in CogNIAM by creating (and maintaining) a list of concept definitions. An instance of such a list of concept definitions for our running example is given in Table 1.

## 3.3 Knowledge Domain Fact Types

The next step in CogNIAM is the generalization of the ground facts into fact type forms. The example ground facts from Sect. 3.1 will lead to the following fact type forms by replacing the variable parts in those sentence by 'placeholders' (<..>):

```
The quantity <Quantity> is the annual demand for the item
<Item>
The item <Item> has an ordering cost <Cost>
The item <Item> has a unit holding cost <Cost>
```

Next to the difference in naming conventions, inter-organizational and even intra-organizational communication might necessitate the existence of two or more

**Table 1** The list of concept definitions fort he EOQ

| List of definitions for economic order quantity business process | |
|---|---|
| Item | An individual product that has an identifying item code and is held in inventory somewhere along the value chain.synonym: stock keeping unit |
| Item code | An {Item code} is a unique signification for an [item] that enables us to identify a specific [Item] within the set of all [Item]s within the context of a business organization |
| EAN bar code | An {EAN bar code} is a unique signification for an [Item] that enables us to identify a specific [Item]within the set of all [Item]s within the context of a business organization in Europe |
| Lot | A {lot} is a quantity of [Item]s that are processed together |
| Cost | A sacrifice or expenditure |
| Ordering cost | The [Cost] of preparing a purchase order for a supplier or a production order for shop |
| Inventory holding cost | The sum of the [Cost] of capital and the variable [Cost]s of keeping [Item]s on hand, such as storage and handling, taxes, insurance and shrinking, for a time period of a year |
| Cycle inventory cost | The portion of [Inventory Holding Cost] that varies directly with [Lot] size |
| Economic order quantity | An {Economic Order Quantity} is the quantity of a [Lot] that minimizes total annual [Cycle Inventory Cost] and [Ordering Cost] for a given [item] |
| Annual demand | The yearly total demand for a given [Item] |
| Natural number | A unique signification for an [Economic Order Quantity] or [Annual Demand] that enables us to identify a specific quantity within the set of all [Economic Order Quantity]s or [Annual Demand]s |
| Unit holding costs | The costs for holding one unit of an [Item] in inventory for a year |
| Dollar amount | A unique signification for a [Cost] that enables us to identify a specific [Cost] within the set of all [Cost]s |

fact type forms to communicate instances of the same fact type in a *target group* specific way. For example for the fact type *Annual Demand Quantity* the following two fact type forms might exist together, each serving a different target group within or outside) the organization:

```
1: The quantity <Quantity> is the annual demand for the item
<Item>
2: Item <Item> has an annual demand of quantity <Quantity>
```

In Fig. 3 we have graphically shown the fact type *Annual Demand Quantity* together with the defining fact types for the object types that play the 'variable' roles in the fact type: *Item* and *Quantity*.

We note that we can define as many fact type forms for a fact type as are needed by the domain(s), e.g. we might add fact type forms in German, Russian, French and Spanish if a company's international business contacts require this. The black rectangles in the low-right corner of a 'variable' denote that a value must exist in order to get a correct sentence.

### 3.4 State (Transition) Constraints or Validation Rules

In Fig. 3 we have shown the model for the fact type *Annual Demand Quantity*. The next element of the KRM is the detection of those business rules that can be expressed as constraints or validation rules on the possible populations of the fact type(s), e.g. business rules that specify which actual sentence combinations are allowed to exist at any point in time, and which transitions between sentence combinations are permitted. In fact based modeling a large number of constraint types exist: uniqueness- mandatory role-, value, set-comparison-, ring constraints [19]. In this section we will give an illustration of uniqueness constraints and referential constraints.

Uniqueness Constraints

As an example of the application of a semantic verification process we will illustrate how we can meticulously derive all uniqueness constraints for a given fact type, by starting with sentence instances that represent ground facts:

`The quantity of 1500 is the annual demand for the item ab3456 (sentence 1)`

We will now create a second example sentence in which the value of the 1st variable or placeholder has changed:

`The quantity of 1200 is the annual demand for the item ab3456 (sentence 2)`

We confront the domain expert with these two example sentences and ask him/her whether these sentences can exist in combination at any point in time. The answer of the domain expert is: No, these sentences can not exist in combination, because at a given point in time there exists (at most) one specific value for the annual demand for a given item. This finding now has lead us to the detection of uniqueness constraint *C1* defined as an arrow covering the variable *Item* of fact type *AnnualDemandQuantity* in Fig. 4. We will now create a third example sentence by changing the value of the 2nd variable (of sentence 1):

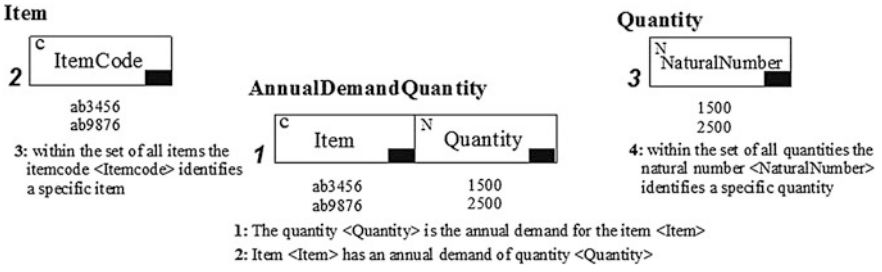


**Fig. 3** Domain model for fact type annual demand quantity including object types

`The quantity of 1500 is the annual demand for the item cd6457 (sentence 3)`

In this case the domain expert will confirm that it is possible that sentences 1 and 3 coexist at any point in time. This means that there does not exist an uniqueness constraint that is defined on the role *Quantity* of the fact type *AnnualDemandQuantity* (see Fig. 4). Uniqueness constraints *C2* and *C3* on the object defining fact types *Item* and *Quantity* are implied because these fact types are unary, i.e. they contain exactly one variable.

Referential Constraints

A second group of constraints or validation rules is concerned with the issue of which object type is referenced by a variable in a fact type. The variable *Quantity* in fact type *Annual Demand Quantity* is played by the object type *Quantity*, hence the subset constraint *c5* departing from the variable role quantity in fact type *Annual Demand Quantity* and ending in the variable role *NaturalNumber* from the object defining fact type *Quantity* (see Fig. 4). Implying that at any point in time the set of annual demand quantities has to be a subset of all quantities.

We note that the referential constraint for the variable item from the fact type *Annual Demand Quantity* to the object type item is the equality constraint *C4* (see Fig. 4). This means that for every item that exists an annual demand quantity must be known. Similar reasoning for constraints *C7* and *C11*: for every item a holding cost and ordering cost must be recorded.

## 3.5 Derivation Rules

In Fig. 5 we have given a complete domain specific fact type model for the EOQ domain area in which we have added all uniqueness and referential constraints.

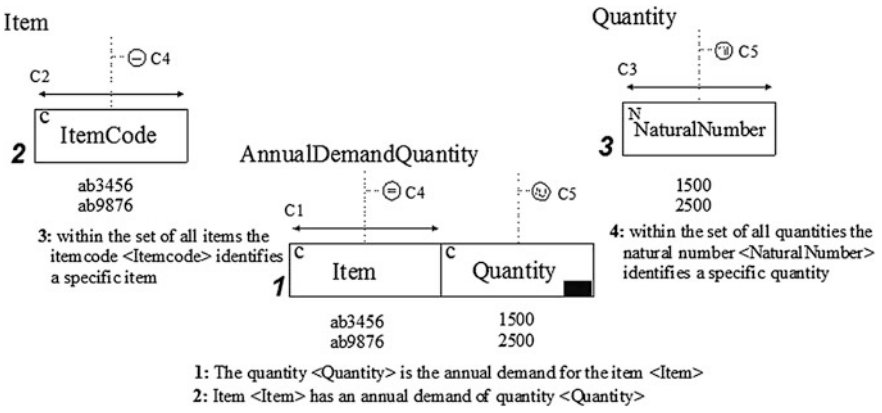


**Fig. 4** Fact type annual demand quality including object types and uniqueness- and referential constraints

Furthermore, it can be noticed from the referential constraints that we do not record an Economic Order Quantity for every item. But what we do know is that for those items that fulfill the EOQ assumptions we will calculate the economic order quantity using the EOQ formula and rounding it to the next integer number. We will model this in fact-based modeling using derivation rules. In CogNIAM a derivation rule is signified by an 'f' box connected to the variables in the fact type that will be derived using the logic from the derivation rule (see Fig. 5).

To specify the logic of the derivation rule we use the notational convention (definition style) from ORM-2 ([19], p. 99) in Fig. 6.

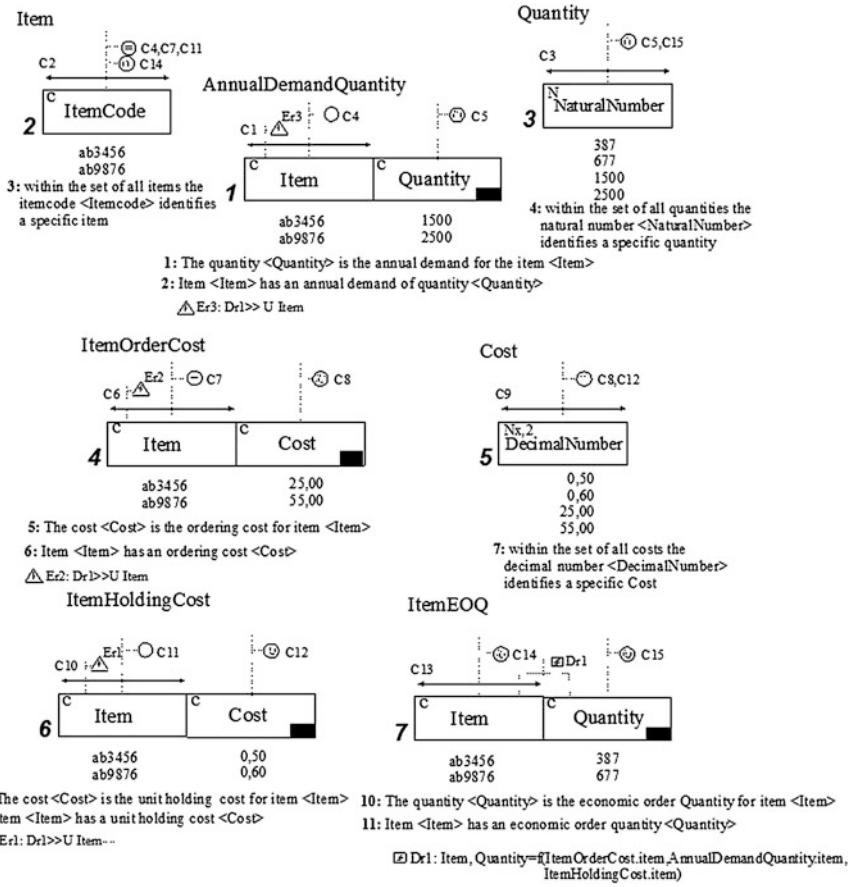


**Fig. 5** Complete domain specific model for economic order quantity

```
{EOQ = ROUND(SQRT((2*annual demand*ordering cost)/unit holding cost))}
define  item has EOQ(units)
    as  item has annual demand    and
        item has ordering cost    and
        item has unit holding cost and
      EOQ= ROUND(SQRT((2*annual demand*ordering cost)/unit holding cost))
```

**Fig. 6** Derivation rule logic for derivation rule Dr 1

### *3.6 Exchange Rules*

In the former section we explained how instances of a derived fact type can be 'calculated' by using a derivation rule. For those fact types that are non-derivable or asserted we need to specify how, instances can be added, updated or deleted from the information base. Basically we distinguish between an insert (I), an update (U) and a delete (D) exchange rule.

### *3.7 Event Rules*

The final element in the knowledge reference model (KRM) are the event rules. An event rule basically determines when (an) exchange rule(s) and/or derivation rules(s) must be executed. In the domain model from Fig. 5 three event rules are contained. Event rule *Er1* (Dr1 ≫ U Item) tells us that whenever an update (U) takes place on the holding cost for a specific item, derivation rule *Dr1* must be executed. In semantic terms this means that a new EOQ is calculated as soon as the holding cost of an item change. A similar reasoning applies to event rules *Er2* and *Er3* for a change in *order costs* and *annual demand quantity*.

## 4 Conclusion

What we can conclude is that in spite of the trends in information systems development from 'tailor-made' towards 'commercial-off-the-shelf' (COTS) software implementations, the *requirement determination process* still is a significant process in the development life cycle of information systems. Moreover, the increase in complexity of the requirements determination process due to the use of 'pre-fabricated' software with its numerous implementation options (see the discussion on *configuration tables* in [14]) has basically increased the need for *requirements determination methods* that have a way of modeling that can capture the complete set of user requirements and which way of working will guide the analyst in extracting all relevant business entities and business rules for a specific application domain. The steps in the requirements determination process that cover the semantic verification are missing in the existing requirements determination

methods for management information systems or business information systems ([18], p. 376). We have shown that there exists generic RE models [31], i.e. CogNIAM [28] that fill this void in semantic-oriented coordination [13] and that can deliver a semantically verified requirements specification by guiding ERP-analysts and ERP-configurers in their task.

## References


1. Bansal, V., Negi, T.: A metric for ERP complexity. LNBIP **7**, 369–379 (2008)
2. Barrett, A., Edwards, J.: Knowledge elicitation and knowledge representation in a large domain with multiple experts. Exp. Syst. Appl. **8**(1), 169–176 (1995)
3. Boehm, B.: Software Engineering Economics. Prentice-Hall, Englewood Cliffs (1981)
4. Boehm, B: Software Risk Management. IEEE computer society press, Los Alamitos (1989)
5. Bollen, P.: The Natural Language Modeling Procedure'. In: Halevy, A., Gal, A. (eds.) Proceedings Fifth Workshop on Next Generation Information Technologies and Systems (NGITS'2002), Lecture Notes in Computer Science 2382, pp. 123–146. Springer, Berlin (2002)
6. Bollen, P.: On the applicability of requirements determination methods. Ph.D thesis. Faculty of Management and Organization. Rijksuniversiteit Groningen (2004)
7. Bollen, P.: Natural language modeling for business application semantics. J. Inf. Sci. Technol. **2**(3), 18–48 (2005)
8. Boudreau, M. ERP Implementation and Forms of Organizational Change. Working paper Georgia State University (1999)
9. Breuker, J., Wielinga, B.: Knowledge acquisition as modeling expertise; The KADS methodology. Paper presented at the 1st European workshop on knowledge acquisition for knowledge based systems. Reading University (1987)
10. Browne, G., Rogich, M.: An empirical investigation of user requirements elicitation: comparing the effectiveness of prompting techniques. J. Manag. Inf. Syst. **17**(4), 223–249 (2001)
11. Byrd, T., Cossick, K., Zmud, R.: A synthesis of research on requirements analysis and knowledge acquisition techniques. MIS Q. **16**(1), 117–138 (1992)
12. Chen, P.: The entity-relationship model: towards a unified view of data. ACM TODS **1**(1), 9–36 (1976)
13. Daneva, M., Wieringa, R.: A coordination complexity model to support requirements engineering for cross-organizational ERP. Requirements Engineering, 14th IEEE International Conference, pp. 311–314 (2006)
14. Davenport, T.: Putting the enterprise into the enterprise system. Harvard Bus. Rev. **76**(4), 121–131 (1998)
15. Davenport, T., Short, J.: The new industrial engineering: information technology and business process redesign. Sloan Manag. Rev. **31**(4), 11–27 (1990)
16. Dullea, J., Song, I.-Y., Lamprou, I.: An analysis of structural validity in entity-relationship modeling. Data Knowl. Eng. **47**, 167–205 (2003)
17. Flynn, D.: Information Systems Requirements: Determination and Analysis. McGraw-Hill, London (1992)
18. Goldin, L., Berry, D.: Abstfinder, a prototype natural language text abstraction finder for use in requirements elicitation. Aut. Softw. Eng. **4**, 375–412 (1997)
19. Halpin, T., Morgan, T.: Information Modeling and Relational Databases 2nd edn. Morgan Kaufmann Publishers, San Francisco (2008)
20. Hammer, M.: Reengineering work: don't automate, obliterate. Harvard Bus. Rev. **68** (4), 104–112 (1990)

21. Lalioti, V., Loucopoulos, P.: Visualisation of conceptual specifications. Inf. Syst. **19**(3), 291–309 (1994)
22. Lang, M., Duggan, J.: A tool to support collaborative software requirements management. Requir. Eng. **6**, 161–172 (2001)
23. Light, B.: The maintenance implications of the customization of ERP software. J. Softw. Maint. Evol. Res. Pract. **13**, 415–429 (2001)
24. Lundeberg, M., Goldkuhl, G., Nilsson, G.: A systematic approach to information systems development. Inf. Syst. **4**, 1–12, 93–118 (1979)
25. Madapusi, A., D'Souza, D.: The influence of ERP system implementation on the operational performance of an organization. Int. J. Inf. Manag. **32**, 24–34 (2012)
26. Molenaar, T.: Siebel zet in op personeelsbeheer. Computable 43: 26 oktober: p. 11 (2001)
27. Nijssen, G.M.: On the gross architecture for the next generation database management systems. In: Gilchrist, B., (ed.) Information Processing'77, pp. 327–335 (1977)
28. Nijssen, G.M., Le Cat, A.: Kennis Gebaseerd Werken: de manier om kennis productief te Maken. PNA Publishing, Heerlen (2009)
29. Nijssen, M., Lemmens, I.: Verbalization for business rules and two flavors of verbalization for fact examples. LNCS **5333**, 760–769 (2008)
30. Nijssen, M., Lemmens, I., Mak, R.: Fact-orientation applied to develop a flexible employment benefits system. LNCS **5872**, 745–756 (2009)
31. Niu, N., Easterbrook, S.: Exploiting COTS-based RE methods: an experience report. LNCS **5030**, 212–216 (2008)
32. Nurcan, S., Rolland, C.: A multi-method for defining the organizational change. Inf. Softw. Technol. **45**, 61–82 (2003)
33. Rolland, C., Prakash, N.: Bridging the gap between organisational needs and ERP functionality. Requir. Eng. **5**, 180–193 (2000)
34. Siriginidi, S.: Enterprise resource planning in reengineering business. Bus. Process Manag. **6**(5), 376–391 (2000)
35. Skok, W., Legge, M.: Evaluating enterprise resource planning (ERP) systems using an interpretive approach. Knowl. Process Manag. **9**(2), 72–82 (2002)
36. Soffer, P., Golany, B., Dori, D., Wand, Y.: Modelling off-the-shelf. Information systems requirements: an ontological approach. Require. Eng. **6**, 183–199 (2001)
37. Soffer, P., Golany, B., Dori, D.: ERP modeling: a comprehensive approach. Inf. Syst. **28**(6), 673–690 (2003)
38. Stock exchange kills projects to focus on Taurus. (1989). Editorial
39. Computing NoSystem problems leave Inland revenue with £ 20 of taxpayers' cash (2002). Computer Weekly. February 14
40. Theory, T., Yang, D., Fry, J.: A logical design methodology for relational databases using the extended E-R model. ACM Comput. Surv. **18**(2), 197–222 (1986)
41. Tsichritzis, D., Klug, A.: The ANSI/X3/SPARC DBMS framework. Info. Syst. **3**, 173–191 (1978)
42. Viller, S., Bowers, J., Rodden, T.: Human factors in requirements engineering: a survey of human sciences literature relevant to the improvement of dependable systems development processes. Interact. Comput. **11**(6), 665–698 (1999)
43. Wagner, E., Scott, S.V., Galliers, R.: The creation of 'best practice' software: myth, reality and ethics. Inf. Organ. **16**, 251–275 (2006)
44. Wetherbe, J.: Executive information requirements: getting it right. MIS Q. **15**(1), 51–65 (1991)
45. Wijers, G.: Modelling support in information systems development. Doctoral thesis. Technical University Delft (1991)
46. Wu, I.-L., Shen, Y.-C.: A model for exploring the impact of purchasing strategies on user requirements determination of e-SRM. Inf. Manag. **43**, 411–422 (2006)
47. Yourdon, E., Constantine, L.: Structured Design. Prentice Hall, (1979)