Available online at www.sciencedirect.com
SCIENCE DIRECT®
Data & Knowledge Engineering 58 (2006) 4–20
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

Information modeling: The process and the required competencies of its participants

P.J.M. Frederiks [a,*], Th.P. van der Weide [b]

[a] *Philips Lighting B.V., Building EDW, Mathildelaan 1, 5611 BD Eindhoven, The Netherlands*
[b] *Faculty of Science, Radboud University, Nijmegen, The Netherlands*

Received 19 May 2005; accepted 19 May 2005
Available online 24 June 2005

Abstract

In recent literature it is commonly agreed that the first phase of the software development process is still an area of concern. Furthermore, while software technology has been changed and improved rapidly, the way of working and managing this process have remained behind.

In this paper focus is on the process of information modeling, its quality and the required competencies of its participants (domain experts and system analysts). The competencies are discussed and motivated assuming natural language is the main communication vehicle between domain expert and system analyst. As a result, these competencies provide the clue for the effectiveness of the associated process of information modeling.

Keywords: Conceptual modeling; Elicitation; Validation; Domain expert; System analyst

1. Introduction

Nowadays many methods exist for the development process of software. Following the classification in Sommerville (see [1]), a number of characteristics are:

* Corresponding author.
E-mail addresses: paul.frederiks@philips.com (P.J.M. Frederiks), th.p.vanderweide@cs.ru.nl (Th.P. van der Weide).

0169-023X/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2005.05.007

- *Iterative development* is a well known example on which the *waterfall model* [2] is based. The waterfall model is a clear model for managing the development process because it separates this process into several well-defined phases.
- *Evolutionary development* is based on the idea of developing an initial implementation (a prototype) and then refining and completing this prototype in interaction with the user. This way of working allows requirements and design decisions to be delayed.
- *Incremental development* as suggested in [3] combines the advantages of both iterative and evolutionary development leading to a clear management model with more user interaction. Another example of a hybrid, iterative model is the *spiral model* [4] which combines incremental development with explicit risk analysis thus improving the overall rate of success of the development process. The Rational Unified Process (RUP) is an example of an incremental development method [5].
- *Reuse-oriented development* focusses on reusing existing software components. COTS (Commercial Off-The-Shelf systems, [6]) is an example of industrial practise.
- *Formal systems development* is based on formal mathematical transformation. The best known example is the Cleanroom process which was originally developed by IBM [7,8].

1.1. Requirements engineering

As different as all these development processes may be, there are fundamental activities common to all these processes. One of these activities is *requirements engineering* (RE), although this activity has its own rules in each development method. RE is the process of discovering the purpose for which the software system is meant, by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, negotiation, decision-making (see [9]) and subsequent implementation. For an extensive overview of the field of RE we refer to [10].

Experts in the area of software engineering do agree that RE is the most important factor for the success of the ultimate solution for reasons that this phase closes the gap between the concrete and abstract way of viewing at phenomena in application domains (see [1,11,12]). As a consequence, during the RE process, the involved information objects from the Universe of Discourse (UoD) have to be identified and described formally. We will refer to this process as *Information Modeling* (*IM*). The resulting model will serve as the common base for understanding and communication, while engineering the requirements. Note that information modeling in this context grasps the fundamental and structural part of the meaning of information objects. Therefore it goes beyond traditional data modeling.

Where different areas of expertise meet, natural language may be seen as the base mechanism for communication. It is for this reason that each general modeling technique should support this basis for communication to some extent. As a consequence, the quality of the modeling process is bounded by the quality of *concretizing into an informal description* augmented with the quality of *abstracting from this description*. This triangular inequality is depicted in Fig. 1. We will refer to these roles within the modeling process as *domain expert* (concretization) and *system analyst* (abstraction).

In this paper focus is on information modeling as an exchange process between a domain expert and a system analyst. Our intention is to describe this exchange process, and the underlying

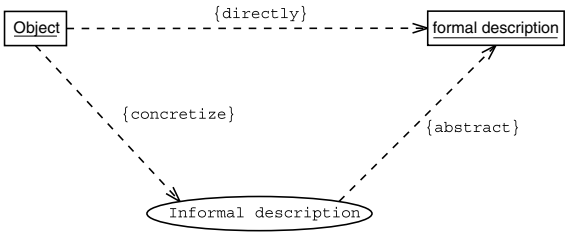

Fig. 1. Triangular inequality.

assumptions on its participants. Using natural language in a formalized way can, for example, be seen as a supplement to the concept of *use cases* (see [13]).

Roughly speaking, a domain expert can be characterized as someone with (1) superior detail-knowledge of the UoD but often (2) minor powers of abstraction from that same UoD. The characterization of a system analyst is the direct opposite. We will describe the required skills of both system analysts and domain experts from this strict dichotomy and pay attention to the areas where they (should) meet. Of course, in practice this separation is less strict. Note that as a result of the interaction during the modeling process the participants will learn from each other. The system analyst will become more or less a domain expert, while the domain expert will develop a more abstract view on the UoD in terms of the concepts of the modeling technique. This learning process has a positive influence on effectiveness and efficiency of the modeling process, both qualitative (in terms of the result) and quantitative (in terms of completion time). Related work on human competency modeling can be found in [14–16]. [14] is based on the premise that people's behavioural competencies influence the effectiveness and efficiency in the software process. In our paper we focus on the role of language during this process.

1.2. Information modeling

According to [17] the IM process is refined into four more detailed phases, see Fig. 2. Note that this process is not necessarily to be completed before other phases of system development, i.e. design and implementation, can start. At each moment during this process the formal specification may be realized. Still the IM process may be recognized in the various methods for system development as discussed in the beginning of this section.

In order to initiate the information modeling process the *system analyst* must elicit an initial problem specification from *domain experts*. This is referred to as *requirements elicitation*. In this phase domain knowledge and user requirements are gathered in interactive sessions with domain experts and system analysts. Besides traditional techniques, more enhanced techniques may be applied, for example *cognitive* or *contextual* techniques. For more elicitation techniques, see [10] or [18].

The requirements elicitation results in an *informal specification*, also referred to as the *requirements document*. As natural language is human's essential vehicle to convey ideas, this requirements document is written in natural language. In case of an evolutionary development, the previous requirements document will be used as a starting point.

In an iterative process of *modeling*, *verification* and *validation* the informal specification evolves to a complete *formal specification*, also referred to as a *conceptual model*. The primary task of the

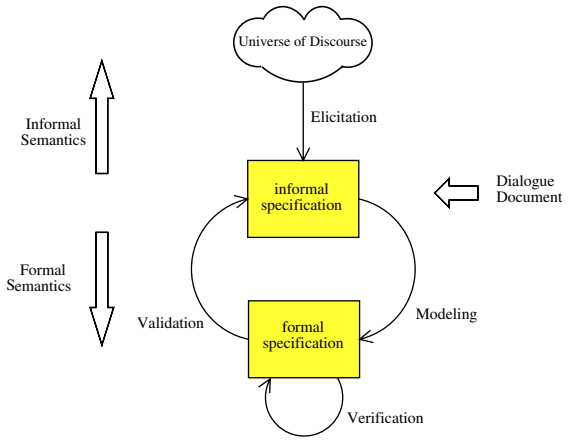

Fig. 2. Information modeling process.

system analyst is to map the sentences of this informal specification onto concepts of the particular conceptual *modeling technique* used. As a side product, a sample population of the concepts derived from the example instances may be obtained. Using the formal syntax rules of the underlying modeling technique, the formal specification can be verified. The conceptual model in turn can be translated into a comprehensible format. For some purposes a prototype is the preferable format, for other purposes a description is better suited. In this paper we restrict ourselves to a description in terms of natural language sentences that is to be validated by the domain expert. The example population serves as a source when instantiated sentences are to be constructed, thereby creating a feedback loop in the IM process. This translation process is called *paraphrasing*.

Basically, the conceptual model may be seen as a generative device (grammar) capable to generate not only the informal specification, but also all other feasible states of the UoD.

The *correctness* of this way of working depends on whether the formal specification is a proper derivate of the informal specification which in its turn must be a true reflection of the UoD. Being a proper derivate is also referred to as the *falsification principle*, which states that the derived formal specification is deemed to be correct (adequate) as long as it does not conflict with the informal specification. The care for this principle is a responsibility for the system analyst. Being a true reflection is referred to as the *completeness principle*. It is falsified when a possible state of the UoD is not captured, or when the grammar can derive an unintended description of a UoD state. It is the responsibility of the domain expert to guarantee this principle not to be violated.

These two principles require the participants of the modeling process to have some specific competencies. For instance, a domain expert should be able to come up with significant sample information objects. On the other hand a system analyst should be able to make a general model out of the sample information objects such that the model describes all other samples. Section 3 discusses these competencies in more detail.

The *effectiveness* of this way of working depends on how well its participants can accomplish their share, i.e. (1) how well can a domain expert provide a domain description, (2) how well

can a domain expert validate a paraphrased description, (3) how well can a system analyst map sentences onto concepts, and (4) how well can a system analyst evaluate a validation. At least one iteration of the modeling loop is required, a bound on the maximal number of iterations is not available in this simple model.

Usually modeling techniques tend to focus on modeling concepts and an associated tooling but less on the process of modeling, i.e. the way of working. To minimize and to control the number of iterations, this way of working requires methods that support this highly interactive process between domain experts and system analysts. Furthermore, guarantees for the quality of this process and the required competencies of the participants involved during RE are insufficiently addressed by most common methods.

Generally spoken, the way of working of a modeling technique should be such that:

> The probability of a model being inadequate, as a function of the dialogue length, tends to zero for the combination of a qualified domain expert and a qualified system analyst.

The goal of the system analyst may be described as follows:

> Find a minimal (information) grammar capable to generate/accept the sentences of the informal specification.

Being minimal is to be motivated from the informal specification itself, in the sense that each formal concept is motivated from the informal specification. A forteriori, the introduction of each formal concept then can be related to the specific items in the dialogue document.

In this paper focus is on the IM process, its quality and the required competencies of the participants in this process. Section 2 describes the IM process in more detail. The competencies of the participants of the IM process is subject of discussion in Section 3. In Section 4 the correctness of this way of working is verified and sketched. Finally, our conclusion and directions of future research are presented in Section 5.

2. The information modeling process

In order to make a more fundamental statement about the quality of IM, we describe in Fig. 4 the modeling process in more depth by further elaborating the activities of *elicitation*, *modeling* and *validation*. We will also make more explicit what elements can be distinguished in a *formal specification*. In the next section we will make explicit what competencies are required from its participants, and use these competencies to verify this process (Fig. 3).

The processes represented by the arrows labelled 4–8 are suitable for automation support. A formal theory for these processes and corresponding models is elaborated in [19] and may be used as the underlying framework for building a supporting tool.

2.1. Elicitation

The stage *elicitation* is refined in Fig. 4 into the following substages (the numbers refer to the numbers in the figure):

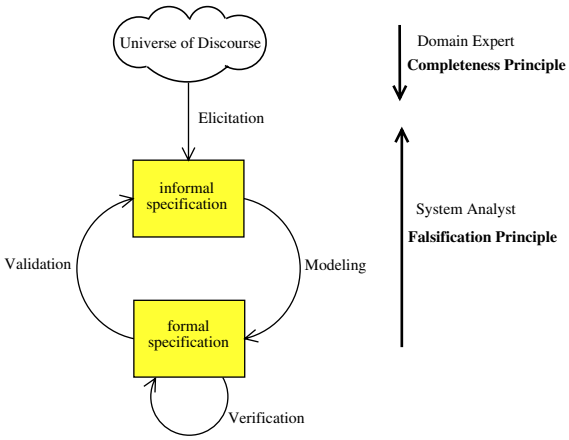

Fig. 3. Information modeling process: roles and responsibilities.

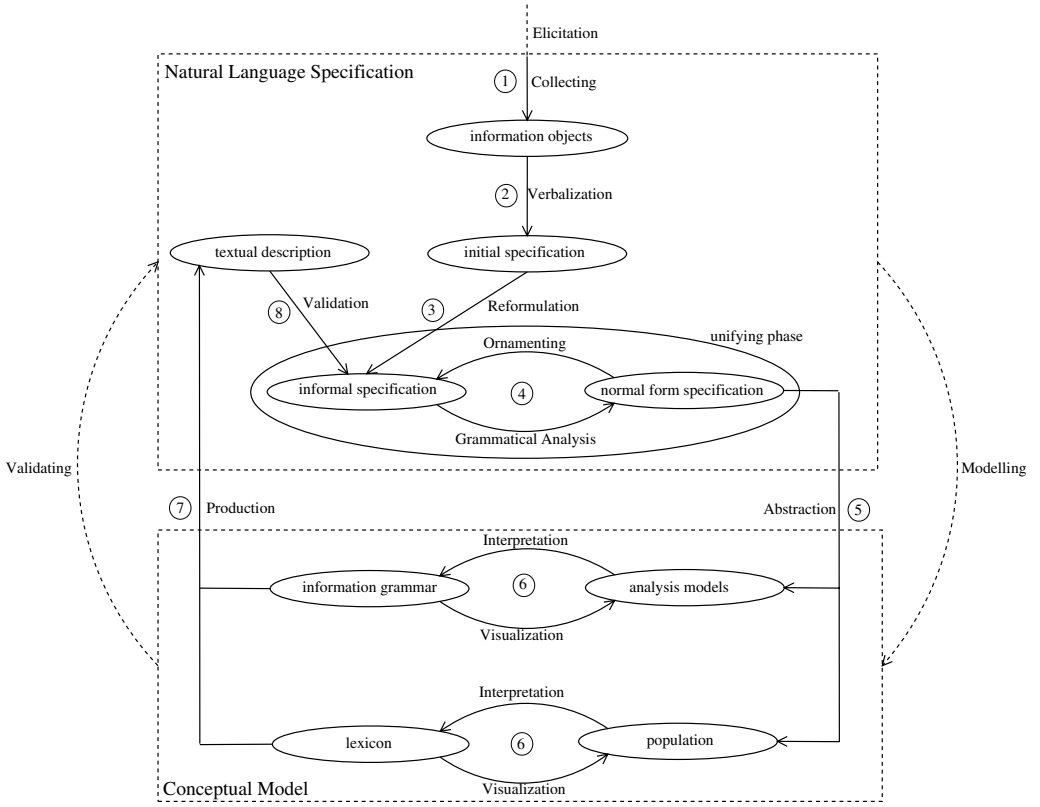

Fig. 4. The IM process in more detail.

1. Collect significant information objects from the application domain.
2. Verbalize these information objects in a common language.
3. Reformulate the initial specification into a unifying format.

The communication in the UoD may employ all kinds of information objects, for example text, graphics, etc. However, a textual description serves as a unifying format for all different media. Therefore the so-called *principle of universal linguistic expressibility* [20] is a presupposition for this modeling process:

> All relevant information can and must be made explicit in a verbal way.

The way of working in this phase may benefit from linguistic tools, see [17], for example to detect similarities and ambiguities between sentences.

Independent of what modeling technique used, the (sentences in the) initial specification are to be reformulated in accordance with some unifying formatting strategy, leading to the informal specification. At this point the modeling process may start, during which the involvement of the domain expert is not required.

Only few conceptual modeling techniques provide the domain expert and system analyst with clues and rules which can be applied on the initial specification, such that the resulting informal specification can actually be used in the modeling process. Examples of such modeling techniques are NIAM [21], Object-Role Modeling [22] and PSM [23].

2.2. Modeling

The intention of the modeling phase is to transform an informal specification into a formal specification. This phase can be decomposed into the following substages (see Fig. 4):

4. Discover significant modeling concepts (syntactical categories) and their relationships.
5. Match sentence structure on modeling concepts.

During the grammatical analysis, syntactical variation is reduced to some syntactical normal form. The next step is to abstract from the sentence structures within the normal form specification and to match these sentences structures onto concepts of the modeling technique used. On the one hand, this step results in a structural framework (the conceptual model) and the rules for handling this framework (the constraints). Furthermore, the formal specification describes how these concepts can be addressed. Without loss of generality, we can state that this part of the specification contains a *lexicon* and rules to describe compound concepts (the *grammar rules*). As a side product, a *sample population* is obtained and put at the disposal of the *validation* activity.

The elementary and structure sentences of the normal form specification provide a simple and effective handle for obtaining the underlying conceptual model of so-called *Snapshot Information Systems* (see e.g. [24]), i.e. information systems where only the current state of the UoD is relevant. However, even though these informal specifications are an important aid in modeling information systems, they are still too poorly structured. One of the missing elements is the order and history of events. The mutual order of the sentences in an informal specification is lost, the analyst has to reconstruct this order. Other missing structural UoD properties are for instance related to the associates involved in events, and the role in which they are involved (see for example [25]).

2.3. Validation and verification

The validation phase is further refined in Fig. 4 into the following stages:

6. Produce by paraphrases a textual description of the conceptual model using the information grammar and the lexicon.
7. Validate the textual description by comparing this description with the informal specification. Finally, we consider the verification phase.
8. Check formal specification for internal consistency, e.g., check model for flexibility.

Once the information grammar is obtained as part of the formal specification, this grammar may be used to communicate the derived models to the domain experts for validation purposes. Two ways of validating the analysis models are considered:

(1) producing a textual description of (part of) an analysis model using the information grammar,
(2) obtaining a parser from the information grammar which provides the domain expert with the ability to check whether sentences of the expert language are captured by the (current version of the) information grammar. Note that this will be very useful when prototyping is used.

For an example of how to construct such a validation mechanism, see [26].

3. Information modeling competencies

The previous section describes IM from a process point of view. In this section, we discuss the actors and investigate the competencies that are required to provoke the intended behavior. Note that the processes labelled 4, 5, 6, and 8 have associated a single actor, i.e. the system analyst, while the other processes (labelled 1, 2, 3, and 7) have more actors. We will not consider any algorithmic synchronization aspects of cooperation between actors. Whether domain expert or system analyst actually are teams, is also not considered (Fig. 5).

There are some interesting issues that are not discussed in this paper. For instance, social skills, such as negotiating and communicating, are out of scope. How the competencies are measured, enforced in practice, and how to check if they are applied, is also not discussed.

3.1. Domain experts

A first base skill for a domain expert is the *completeness base skill*:

D-1 Domain experts can provide a complete set of information objects.

As obvious as this skill might seem, its impact is rather high: the skill is the foundation for correctness of the information system and provides insight into the requirements for those who communicate the UoD to the system analyst during step 1. Furthermore, the skill is required to fulfill the

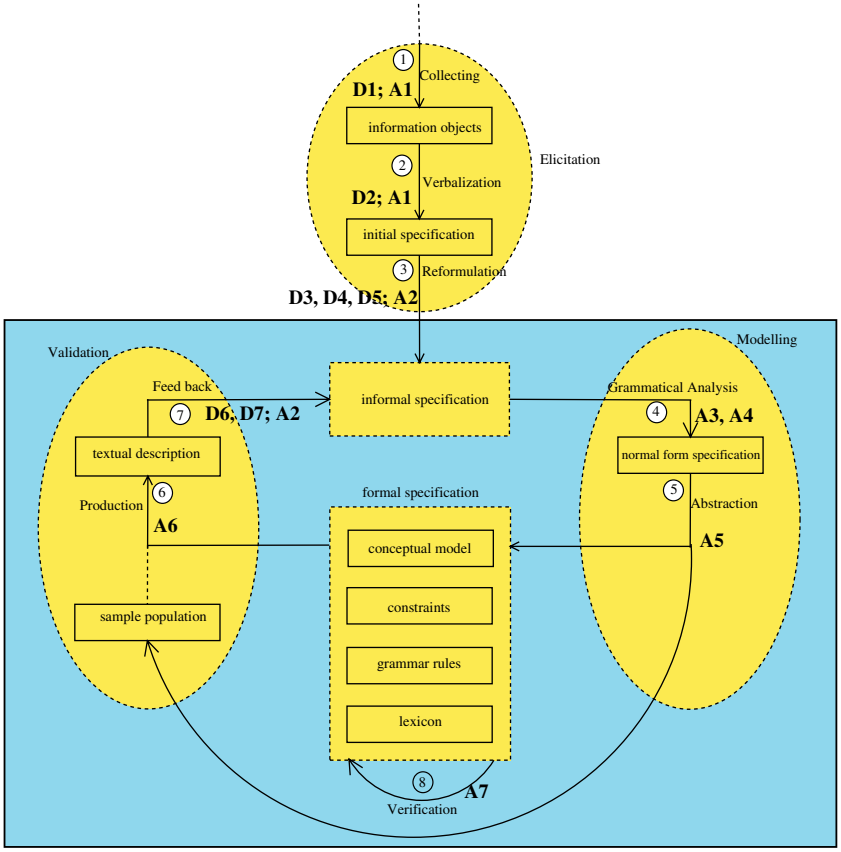

Fig. 5. Overview of competencies.

completeness principle as introduced in Section 1. A second foundation for correctness, which is also introduced in Section 1, is the *falsification principle*, which states that the derived formal specification is deemed to be correct if it does not conflict with the informal specification. To support this principle system analysts should have suitable skills, which are the topic of the next subsection.

The next step, labelled with verbalization, is the composition of a proper description of the information objects. It requires the domain expert to be familiar with the background of these objects, and being capable to describe them in every aspect. This is the purpose of the *provision base skill*:

D-2 Domain experts can provide any number of significant sample sentences in relation to relevant information objects.

This competence is a bottleneck for verbalization. As this base skill does not state that a domain expert can provide a *complete* set of significant examples by a single request from the system analyst, some more base skills that describe aspects of the *communication* between domain experts and system analysts are necessary.

A prerequisite for conceptual modeling is that sentences are elementary, i.e. not splittable without loss of information. As a system analyst is not assumed to be familiar with the semantics of sample sentences, it is up to the domain expert to judge about splitting sentences. This is expressed by the *splitting base skill*:

D-3 Domain experts can split sample sentences into elementary sentences.

A major advantage *and also* drawback of natural language is that there are usually several ways to express one particular event. For example, passive sentences can be reformulated into active sentences. By reformulating all sample sentences in a uniform way a system analyst can detect important syntactical categories during grammatical analysis of these reformulated sample sentences. The domain expert is responsible for reformulating the sample sentences, which is expressed by the *normalization base skill*:

D-4 Domain experts can reformulate sample sentences into a unifying format.

In order to capture the dynamics of the UoD the sentences need to be ordered. As a result, the domain expert has to be able to order the sample sentences. This is captured in the *ordering base skill*:

D-5 Domain experts can order the sample sentences according to the dynamics of the application domain.

The skills D-3, D-4 and D-5 are essential for reformulation (step 2 of the IM process).

During the modeling process, the information grammar (and thus the conceptual model) is constructed in a number of steps. After each step a provisional information grammar is obtained, which can be used in the subsequent steps to communicate with domain experts. First a description of the model so far can be presented to the domain experts for validation. In the second place, the system analyst may confront the domain expert with a sample state of the UoD for validation. The goal of the system analyst might be to detect a specific constraint or to explore a subtype hierarchy. This is based on the *validation base skill*:

D-6 Domain experts can validate a description of their application domain.

This skill is essential for the validation in step 7 of the IM process. During this step of the IM process, the system analyst may check completeness by suggesting new sample sentences to the domain expert. This is based on the *significance base skill*:

D-7 Domain experts can judge the significance of a sample sentence.

The performance of the analysis process and the quality of its result benefits from the capability of the domain expert to become familiar with the concepts of the modeling technique. This is expressed by the *conceptuality base skill*:

D-8 Domain experts have certain abilities to think on an abstract level.

In contrast with the former skills, this latter skill is not required but highly desirable. This skill has a positive effect on all steps of the information modeling process.

3.2. System analysts

Besides studying the cognitive identity of domain experts it is necessary to investigate the *cognitive identity* of system analysts.

Domain experts tend to leave out (unconsciously) those aspects in the application domain which are experienced as generally known and daily routine. An inexperienced system analyst may overlook such aspects leading to discussions on the explicit aspects, which usually addresses exceptions rather than rules. In order to detect implicit knowledge, the system analyst should not take things for granted, but should rather start with a minimal bias with a clean slate. This is expressed by the *tabula rasa base skill*:

A-1 System analysts can handle implicit knowledge.

This base skill is most related to the effectivity of the modeling process and the quality of its result, and is imperative during steps 1 and 2.

A next skill for system analysts is the so-called *consistency base skill*:

A-2 System analysts can validate a set of sample sentences for consistency.

The system analyst will benefit from this base skill especially during step 3. A major step in a natural language based modeling process is the detection of certain important syntactical categories. Therefore a system analyst must be able to perform a (possibly partially automated) grammatical analysis of the sample sentences. The result of this grammatical analysis is a number of related syntactical categories (step 4). This leads to the *grammar base skill*:

A-3 System analysts can perform a grammatical analysis on a set of sample sentences.

A system analyst is expected (during step 4) to make abstractions from detailed information provided by domain experts. By having more instances of some sort of sentence, the system analyst will get an impression of the underlying *sentence structure* and its appearances. The sentence structures all together form the structure of the information grammar, used as a normal form for sentences of this sort. This competence is addressed in the *abstraction base skill* rule, which states that system analysts should be able to abstract from the result of the grammatical analysis:

A-4 System analysts can abstract sentence structures from a set of related syntactical categories.

Then (during step 5) a system analyst must be able to match sentence structures found with the abstraction base skill with the concepts of a particular conceptual modeling technique. This is expressed by the *modeling base skill* rule:

A-5 System analysts can match abstract sentence structures with concepts of a modeling technique.

As abstraction of sentences is based on the recognition of concrete manifestations, the system analyst must be able to generate new sample sentences which are validated by the domain experts. This enables the system analyst to formulate hypotheses via sample sentences. This way the system analyst has a mechanism to check boundaries, leading to data model constraints. Being able to generate sample sentences is expressed by the *generation base skill*:

A-6 System analysts can generate new sample sentences.

For example the system expert might resolve a case of ambiguity by offering (in step 7) well suited sample sentences to the domain expert for validation (see also [27] for an inspection technique for detecting ambiguities in informal requirement documents). This base skill is related to the ability of the system analyst to get acquainted with the nature of the application domain, i.e., the counterpart of base skill D-8 for the domain expert. Note that base skill A-6 is required to give execution to the falsification principle as introduced in Section 1.

Finally, the system analyst is expected to control the quality of the analysis models against requirements of well-formedness, i.e. the system analyst must satisfy the *fundamental base skill*:

A-7 System analysts can think on an abstract level.

The skills presented for the system analysts focus on a natural language based modeling process. Of course, system analysts and *system designers* also need expertise for using and understanding modeling techniques. In [28] a conceptual framework is proposed for examining these types of expertise. The components of this framework are applied to each phase of the development process and used to provide guidelines for the level of expertise developers might strive to obtain.

Starting from a *tabula rasa*, the performance of the analysis process and the quality of its result benefits from the capability of the system analyst expert to become familiar with the application domain. This is expressed by the *impersonation base skill*:

A-8 System analysts can impersonate oneself in the world of the domain expert.

This base skill will enable the system analyst to help the domain expert to find requirements which the domain expert may not be aware of in the first place. An additional advantage is that future requirements (and thus future changes) may be anticipated.

In contrast with the former skills, this latter skill is not required but highly desirable. This skill has a positive effect on all steps of the information modeling process.

4. Controlling natural language

As stated in [29], natural language is the vehicle of our thoughts and their communication. Since good communication between system analyst and domain expert is essential for obtaining

the intended information system, the communication between these two partners should be in a common language. Consequently natural language can be seen as a basis for communication between these two partners. Preferably natural language is used in both the modeling process as well as the validation process. In practice, system analyst and domain expert will have gained some knowledge of each other's expertise, making the role of natural language less emphasized moving informal specification towards formal specification.

For the modeling process, natural language has the potential to be a precise specification language *provided* it is used well. Whereas a formal specification can never capture the pragmatics of a system, an initial specification in natural language provides clear hints on the way the users want to communicate in the future with the information system.

Since modeling can be seen as mapping natural language concepts onto modeling technique concepts, paraphrasing can be seen as the inverse mapping intended as a feedback mechanism. This feedback mechanism increases the possibilities for domain experts to *validate* the formal specification, see e.g. [26,30,31]. Besides the purpose of validation, paraphrasing is also useful to (1) lower the conceptual barrier of the domain expert, (2) to ease the understanding of the conceptual modeling formalism for the domain expert and (3) to ease the understanding of the UoD for the system analyst. In [32] more arguments for paraphrasing a conceptual model in natural language are given.

Up to now we focussed on the positive aspects of the usage of natural language, but in practice there are not many people who can use natural language in a (1) complete, (2) non-verbose, (3) unambiguous, (4) consistent way, (5) expressed on a uniform level of abstraction. In the sequel of this section we will make it plausible how the base skills for domain experts and systems analysts can reduce the impact of these disadvantages, as these are the main critical success factors of IM, of which the efficiency and effectiveness is a direct consequence.

The completeness and provision base skills (D-1 and D-2) anticipate on the completeness problem of natural language specifications. Skill D-2 states that domain experts can provide any number of significant sample sentences based on these information objects. Assuming that each UoD can be described by a finite number of structurally different sample sentences, the probability of missing some sentence structure decreases with each new sample sentence generated by the domain expert. Furthermore, the system analyst may generate sample sentences for validation in order to test correctness and completeness aspects of the formal specification sofar. These interactions are triggered, controlled and guided by the system analyst, as stated in the tabula rasa base skill A-1 and the consistency base skill A-6, to aim at convergence.

Specifications in natural language tend to be verbose, hiding essentials in linguistic variety. Complex (verbose) sentences will be fed back to the domain expert for splitting (splitting base skill D-3) and judging significance for the problem domain (significance base skill D-7). A natural language specification may also get verbose by exemplification, providing examples (instantiations) of the same sentence structure. The grammar base skill A-3 reflects the ability of the system analyst to recognize the underlying similarity whereas base axiom A-4 provides the ability to abstract from superfluous sample sentences. Furthermore, the ability of domain experts to reformulate sentences in a unifying format (base axiom D-4) and to order sample sentences (base axiom D-5) is also helpful to eliminate the woolliness of initial specifications.

An often raised problem of natural language usage is ambiguity, i.e. sentences with the same sentence structure yet having a different meaning. The system analyst should have a nose for

detecting ambiguities. The consistency base skill A-2 provides the system analyst with the required quality. A typical clue comes from establishing peculiarities in instantiated sentences. In order to decide about a suspected ambiguity, the system analyst will offer the domain expert these sentences for validation (generation base skill A-6 and validation base skill D-6).

On the other hand, the system analyst may also wish to elicit further explanation from the domain expert by requesting alternative formulations or more sample sentences with respect to the suspected ambiguity (provision base skill D-2).

The consistency base skill A-2 guarantees that the system analyst is equipped with the ability to verify a natural language specification for consistency. Just like the entire conceptual modeling process, consistency checking of natural language specifications has an iterative character. Furthermore, consistency checking requires interaction with the domain expert, as a system analyst may have either a request for more sample sentences (provision base skill D-2), or a request to validate new sample sentences (generation base skill A-6, validation base skill D-6 and significance base skill D-7).

Sentences of a natural language specification are often on a mixed level of abstraction. As a system analyst has limited detail knowledge, and thus also limited knowledge at the instance level, a prerequisite for abstraction is typing of instances (grammar base skill A-3 and abstraction base skill A-4) and map these types on the concepts of a modeling technique (modeling base skill A-5). The analysis of instances within a sentence is in fact a form of *typing*, attributing types to each of its components. As an example of such a sentence consider: *The Rolling Stones record the song Paint It Black*. Some instances will be typed by the domain expert (the song *Paint It Black*) while others are untyped (*The Rolling Stones*). This may be resolved by applying a *type inference mechanism* to untyped instances (see [25]). Typed sentences can be presented to the domain expert for validation (base skill D-6).

Although a formal mathematical proof cannot be provided the above line of reasoning makes it plausible that the disadvantages of natural language usage can be overcome if the participants in the information modeling process do have the required skills.

5. Conclusions and future research

After discussing the information modeling process a number of base skills have been presented for both domain experts and system analysts using natural language. The relation between the skills of domain experts and system analysts have been made. Furthermore, these skills are related to the several phases of the information modeling process. It has been made plausible that the disadvantages of using natural language in the information modeling process are weakened by these skills.

Further research is required in order to investigate the consequences of the base skills for the education of both domain experts and system analysts. The question of whether these base skills can be measured and learned seems to be justified. Furthermore, more research is necessary to get a better understanding of the cognition of domain experts and system analysts.

Once these skills have been obtained by domain expert and system analyst, a next research issue is how to enforce their application. A simple example of how in practice system analysts may try to apply the tabula rasa base skill is to enforce active voice and to avoid nominalisation. In both

cases verb valency is more fully exploited to gain as much knowledge on actors or objects involved, etc. as possible. Another technique is to use fully qualified sentences. Future research may also go beyond full qualification. This allows incomplete descriptions to be part of the formal model. In [33] an open modeling concept is introduced to handle incomplete information.

An integration between the framework of this paper (syntax-oriented) and semantics-oriented methods (e.g. the COLOR-X method, see [17,34]) can provide handles to get more grip on the modeling process.

In this paper we have restricted to the participation of domain experts and systems analysts. In the context of the ArchiMate project [35]. Requirements Engineering is studied from a broader point of view, including other participants (stakeholders) and their competencies. This makes information modeling a process of negotiation and decision-making, requiring extensions to the information modeling process [9]. A more sophisticated dialogue model is required to describe this communication. The *Chatbox Model* may be a good candidate for this purpose. The Chatbox Model is a recursive, sentence oriented dialogue model. This model can describe a dialogue between any number of participants, and allows participants to have discussions in smaller groups.

Natural language as a communication mechanism played a prominent role in this paper. Information modeling can be more effective by allowing other communication vehicles without formal semantics. For example, a domain expert may draw a sketch without providing a rigorous semantics. The ArchiMate project also focusses on such multimedia extensions of the communication between domain experts and system analysts, allowing intermediate specifications having assigned a partial semantics only. See for example [33].

Acknowledgments

The second author has worked on this paper in the context of the ArchiMate. This paper results from the ArchiMate, a research initiative that provides concepts and techniques to support an architect in the visualization, communication and analysis of integrated architectures. The ArchiMate consortium consists of ABN AMRO, ABP, the Dutch Tax and Customs Administration, Ordina, Telematica Instituut, Centrum voor Wiskunde en Informatica, University of Nijmegen, and the Leiden Institute of Advanced Computer Science.

The authors would like to thank Denis Verhoef and Marc Lankhorst for reviewing this paper and providing very useful comments.

Furthermore, the authors are grateful to the anonymous referees for their valuable suggestions.

References

[1] I. Sommerville, Software Engineering, sixth ed., Addison-Wesley, Reading, Massachusetts, 2001.
[2] W. Royce, Managing the development of large software systems: concepts and techniques, in: 9th International Conference on Software Engineering, IEEE WESTCON, Los Angeles, California, 1970, pp. 1–9.
[3] H. Mills, D. O'Neill, The management of software engineering, IBM Systems Journal 24 (2) (1980) 414–477.
[4] B. Boehm, A spiral model of software development and enhancement, IEEE Computer 21 (1988) 61–72.
[5] G. Booch, I. Jacobson, J. Rumbaugh, The Rational Unified Process, An Introduction, Addison-Wesley, 2000.
[6] B. Boehm, Cots integration: plug and pray? IEEE Software 32 (1) (1999) 135–138.

[7] H. Mills, M. Dyer, R. Linger, Cleanroom software engineering, IEEE Software 4 (5) (1987) 19–25.
[8] R. Linger, Cleanroom process model, IEEE Software 11 (2) (1994) 50–58.
[9] A. Aurum, C. Wohlin, The fundamental nature of requirements engineering activities as a decision-making process, Information and Software Technology 45/14 (2003) 945–954.
[10] B. Nuseibeh, S. Easterbrook, Requirements engineering: a roadmap, in: 22nd International Conference on Software Engineering, ACM, Limerick, Ireland, 2000, pp. 35–46.
[11] R. Pressman, Software Engineering, fifth ed., McGraw-Hill, London, England, 2000.
[12] G. Kotonya, I. Sommerville, Requirements Engineering—Processes and Techniques, Wiley, UK, 1998.
[13] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Addison-Wesley, 1999.
[14] S.T. Acuna, N. Juristo, Modelling human competencies in the software process, in: ProSim 03, 2003.
[15] R. Boam, P. Sparrow, Designing and Achieving Competency: A Competency-based Approach to Developing People and Organization, McGraw-Hill, 1992.
[16] J.L. Moses, W.C. Byham, Applying the Assessment Center Method, Pergamon, 1997.
[17] J. Burg, Linguistic Instruments In Requirements Engineering, Ph.D. Thesis, Free University, Amsterdam, The Netherlands, 1996.
[18] I. Bray, An Introduction to Requirements Engineering, Addison-Wesley, Edinburg Gate, United Kingdom, 2002.
[19] P. Frederiks, T.v.d. Weide, Deriving and paraphrasing information grammars using object-oriented analysis models, Acta Informatica 38 (2002) 437–488.
[20] P. Adriaans, Language Learning from a Categorial Perspective, Ph.D. Thesis, University of Amsterdam, Amsterdam, The Netherlands, 1992.
[21] G. Nijssen, T. Halpin, Conceptual Schema and Relational Database Design: A Fact Oriented Approach, Prentice-Hall, Sydney, Australia, 1989.
[22] T. Halpin, A. Bloesch, Data modeling in uml and orm: a comparison (1999). URL available from: <citeseer.nj.nec.com/article/halpin99data.html>.
[23] A.t. Hofstede, T.v.d. Weide, Expressiveness in conceptual data modelling, Data and Knowledge Engineering 10 (1) (1993) 65–100.
[24] C. Date, An Introduction to Data Base Systems, seventh ed., Addison-Wesley, Reading, Massachusetts, 1999.
[25] P.v. Bommel, P. Frederiks, T.v.d. Weide, Object-oriented modeling based on logbooks, The Computer Journal 39(9).
[26] C. Derksen, P. Frederiks, T.v.d. Weide, Paraphrasing as a technique to support object-oriented analysis, in: R.v.d. Riet, J. Burg, A.v.d. Vos (Eds.), Proceedings of the Second Workshop on Applications of Natural Language to Databases (NLDB'96), Amsterdam, The Netherlands, 1996, pp. 28–39.
[27] D.B.E. Kamsties, B. Paech, Detecting ambiguities in requirements documents using inspections, in: Workshop on Inspections in Software Engineering (WISE'01), 2001.
[28] V. Storey, C. Thompson, S. Ram, Understanding database design expertise, Data and Knowledge Engineering 16 (2) (1995) 97–124.
[29] W. Quine, Word and object—Studies in communication, The Technology Press of the Massachusetts Institute of Technology, Cambridge, Massachusetts, 1960.
[30] H. Dalianis, Aggregation, formal specification and natural language generation, in: Proceedings of the First International Workshop on Applications of Natural Language to Databases (NLDB'95), Versailles, France, 1995, pp. 135–149.
[31] H. Dalianis, Aggregation in natural language generation, Journal of Computational Intelligence 15 (4) (1999) 384–414.
[32] H. Dalianis, A method for validating a conceptual model by natural language discourse generation, in: P. Loucopoulos (Ed.), Proceedings of the Fourth International Conference CAiSE'92 on Advanced Information Systems Engineering, Lecture Notes in Computer Science, vol. 593, Springer-Verlag, Manchester, United Kingdom, 1992, pp. 425–444.
[33] S. Bosman, T. v. d. Weide, A case for incorporating vague representations in formal information modeling, in: Conferentie Informatiewetenschap 2003, TU Eindhoven, The Netherlands, 2003.
[34] A. Steuten, R. van de Riet, J. Dietz, Linguistically based conceptual modeling of business communication, Data and Knowledge Engineering 35 (2000) 121–136.

[35] H. Jonkers, R.v. Buuren, F. Arbab, F.d. Boer, M. Bonsangue, H. Bosma, H.t. Doest, L. Groenewegen, J. Guillen Scholten, S. Hoppenbrouwers, M.-E. Iacob, W. Janssen, M. Lankhorst, D.v. Leeuwen, H. Proper, A. Stam, L.v.d. Torre, G. Veldhuijzen van Zanten, Towards a language for coherent enterprise architecture descriptions, in: M. Steen, B. Bryant (Eds.), 7th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2003), IEEE Computer Society Press, Brisbane, Australia, 2003.

P.J.M. Frederiks is the IT Demand Manager for the region Europe at Philips Lighting B.V. within the Business Group Lighting Electronics. Before his current job he has worked for Philips Semiconductors, Edmond Research and Development and the University of Nijmegen. In his job at the University of Nijmegen he has written his Ph.D. thesis named "Object Oriented Modeling using Informal Language". His Ph.D. research included a formal approach to the subjects of object orientation, information modeling and the usage of natural language in the information modeling process. Whereas the focus then was mainly on the formalization of models, i.e. the way of modeling, currently his interest is on the way of working and the human factor in the modeling process.

Th.P. van der Weide received his masters degree at the Technical University Eindhoven, The Netherlands in 1975, and the degree of Ph.D. in Mathematics and Physics from the University of Leiden, The Netherlands in 1980. He is currently professor in the Nijmegen Institute for Computing and Information Sciences at the Radboud University in Nijmegen, The Netherlands, and head of the department IRIS (Information and Knowledge Systems). The IRIS department focusses its research on Information and Knowledge Systems. The group contributes to the educational activities of the Informatics program and the Information Science program, both on bachelor and master level. His main research interests include information systems, information retrieval, hypertext and knowledge based systems. Natural language based information modelling in the context of information-system development, including the modeling of the temporal evolution of information objects is another strand of his research.