Understanding the Requirements on Modelling Techniques

S.J.B.A. Hoppenbrouwers, H.A. Proper, and Th.P. van der Weide

Institute for Computing and Information Sciences, Radboud University Nijmegen,
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, EU
{S.Hoppenbrouwers, Th.P.vanderWeide, E.Proper}@cs.ru.nl

Abstract. The focus of this paper is *not* on the requirements of an information system to be developed, but rather on the requirements that apply to the *modelling techniques* used during information system development. We claim that in past and present, many information systems modelling techniques have been developed without a proper understanding of the requirements that follow from the development processes in which these techniques are to be used. This paper provides a progress report on our research efforts to obtain a fundamental understanding of the requirements mentioned. We discuss the underlying research issues, the research approach we use, the way of thinking (*weltanschauung*) that will be employed in finding the answers, and some first results.

1 Introduction

In past and present, many information systems modelling techniques have been, and are being, developed [1, 2, 3, 4]. With the term *modelling technique* we (roughly) refer to the combination of a modelling language/notation and procedures/guidelines for the creation of models. This definition is in line with definitions that can be found in e.g. [3, 4, 5]. The authors of this paper have themselves contributed their fair share of modelling techniques [6, 7, 8, 9, 10, 11]. The plethora of modelling techniques that is available to developers of information system has, in the past, already been referred to as "a jungle" [3]. This jungle leaves developers of information systems with the burden of selecting modelling techniques that are apt for the modelling tasks at hand.

The UML [12] aims to provide a standardisation of some of the diagramming techniques used. However, to a large extent it is still up to the modelers to choose the right diagramming technique for a given task, and the best way to apply it to this task. What is more, there are bound to be modelling tasks for which none of the UML standard techniques suffice. There is no silver bullet [13] for modelling techniques.

This leads us to the key questions with which the research, as reported in this paper, is concerned:

- *What are the purposes that modelling techniques should serve when used in information system development?*

O. Pastor and J. Falcão e Cunha (Eds.): CAiSE 2005, LNCS 3520, pp. 262–276, 2005.

- *What requirements does this set for the modelling techniques to be used?*
- *How should one go about selecting and applying modelling techniques for a given task in a development process?*

In our view, selecting and applying modelling techniques depends strongly on the goals of the tasks that are to be executed in the development process. These goals dictate the requirements that should be set for the modelling technique and their usage. In this paper, we will show how these requirements depend on a multitude of factors. In doing so, we will base ourselves both on theoretical considerations and on input from interviews with practitioners.

In finding answers to the questions raised above, we employ the *action research paradigm* [14]. This entails that our work will progress (evolutionary) through two major stages (taken from [15]):

Diagnostic stage – *This stage involves a collaborative analysis of the social situation by the researcher and the subjects of the research. Theories are formulated concerning the nature of the research domain.*

Therapeutic stage – *This involves collaborative change experiments. In this stage changes are introduced and the effects are studied [16].*

We are currently in transition from the diagnostic stage to the therapeutic stage. In the execution of the diagnostic stage, we have used a three-pronged approach, which has also inspired the structure of the remainder of this article:

Articulate way of thinking – This sub-stage refers to the way of thinking (or *weltanschauung*) concerning information system development we will embrace in seeking the answers to the questions raised above. The current status of this sub-stage is discussed in section 2.

Define conceptual framework – The focus of this sub-stage is on a further refinement and concretisation of the way of thinking in terms of a conceptual framework. This framework is needed to position and "code" the empirical results that will follow from the execution of the next stage (the therapeutic stage). The current status of this sub-stage is addressed in section 3.

Initial findings – The elaboration of the conceptual framework took place in conjunction with a number of interviews with experienced modelers (in particular enterprise architects)[1]. These interviews and discussions already produced some results that provide a more practical perspective on the theoretical framework. These results are presented in section 4.

2 Communication-Driven Knowledge Transformation

In this section we discuss our fundamental way of thinking with regard to system development. It provides a frame of thought against which one can better understand the (communicative) requirements posed on modelling techniques.

[1] This part of the research was conducted within the context of the ArchiMate project (`http://archimate.telin.nl`), a research projects that aims to provide concepts and techniques to support enterprise architects in the visualisation, communication and analysis of integrated architectures.

2.1 Communication-Driven

Key to our view on the utility of modelling techniques is their role as a means of communication in system development. In the past we have already taken a communication-driven perspective on modelling activities in information system development [11, 17, 18, 19, 20], as well as on the act of system development itself [21]. We are certainly not alone in doing so [22, 23].

To understand the role of modelling techniques in system development, we have extended our communicative perspective to cater for the fact that the communication taking place during during system development leads to the creation and dissemination of *knowledge*. In essence, we regard system development as a communication-driven *knowledge transformation* process whereby conversations are used to share and create knowledge pertaining to the system being developed as well as the development process as such. The notion of *conversation* should be interpreted here in the broadest sense, ranging from a single person producing a model (description), via one-on-one design/elicitation sessions, to workshops with several stakeholders, and even the broad dissemination of definitive system designs.

Our aim of viewing information system development as a knowledge transformation process is to use this perspective on system development to better understand and articulate the requirements that should be set for modelling techniques. From this perspective, modelling techniques should be regarded as a means (a language) to an end (system development), not unlike a functional perspective (*What is it to be used for?*) on language [24].

2.2 Development Community

Given our focus on communication, it is important to identify the *actors* and *objects* that could play a role in the communication that takes place during the system development process. The actors are likely to have some stake with regards to the system being developed. Examples of such actors are: problem owners, prospective actors in the future system (such as the future 'users' of the system), domain experts, sponsors, architects, engineers, business analysts. The actors, however, are not the only items playing an important role in system development. In addition, consider a number of objects: the many different documents, models, forms, etc., that *represent* bits and pieces of knowledge pertaining to the system that is being developed. Actors and objects combined, and the different roles they can play, is what we shall refer to as a *system development community*.

The *actors* in a system development community will (typically as a consequence of their personal *goals* and *stakes*) have some specific interests with regard to the system being developed. This interest implies a sub-interest regarding (the contents of) the system descriptions that are communicated within the community. This interest is, in line with [25], referred to as the *concern* of a stakeholder. Some examples of concerns are:

- The current situation concerning the computerized support of a business process.
- The requirements of a specific stakeholder with regard to the desired situation.

– The improvements/benefits which a new system may bring to a pre-existing situation in relation to the cost of acquiring the system.

2.3 System Development Knowledge

The system development community harbours knowledge about the system being developed. To be more precise, the members of the system development community can be regarded as *knowledge carriers* harbouring knowledge pertaining to (their view on) a sub-domain within the system being developed (and/or its development process). In this vein, the communication occurring within a system development community essentially aims to create, further, and disseminate this knowledge. Importantly, the actual knowledge can pertain to the system being developed, as well as the development process as such. In the next section, we will provide a more elaborate discussion on the kinds of knowledge that may (have to) be communicated.

Depending on the concerns of a stakeholder, she will be interested in different knowledge topics pertaining to the system being developed. For example: a financial controller will be interested in an investment perspective on the overall scope of a future system, a designer will be interested in all aspects of the design chain from different perspectives, etc.

2.4 Transformations of Knowledge

During the development of a system, the knowledge about the system and its development will evolve. New insights emerge, designs are created, views are shared, opinions are formed, design decisions made, etc. Consequently, the knowledge as it is present in a development community can be seen to evolve through a number of *knowledge states*. At present, we identify two dimensions for the knowledge states of the development community: (1) level of sharing, and (2) level of explicitness.

Knowledge needs to be *introduced* into the community first, either by creating the knowledge internally or importing it from outside of the community. Once the knowledge has been introduced to a community, it can be *shared* among different knowledge carriers. Sharing knowledge between different knowledge carriers may progress through a number of stages. Of the two kinds of knowledge carriers (objects and actors) in a development community, only the actors are "allowed" to cast judgement on the level of sharing between two knowledge carriers. We actually distinguish three major stages of knowledge sharing:

Aware – An carrier may become "aware" of (possible) knowledge by way of the sharing by another carrier (possibly from outside the community), or by creating it themselves.

Agreed – When shared, carriers can make up their own "minds" about the shared knowledge, and decide wether or not to *agree* to the knowledge shared.

Committed – Carriers who "agree" to a specific knowledge topic may decide to actually commit to this knowledge. In other words, they may decide to adopt their future behaviour in accordance to this knowledge.

There is no way to objectively and absolutely determine the levels of awareness, agreement, and commitment of a given set of knowledge carriers. It is in the eyes of the beholder.

The actual knowledge that is harboured by a knowledge carrier can also not be taken into account *explicitly* since the knowledge that is available from/on/in a knowledge carrier is subjective and context-dependent by nature [26]. The harbouring of a knowledge topic by some knowledge carrier may occur at different levels of formality, completeness, executability, etc. In the field of knowledge management, a key distinction is made between *explicit* and *tacit* knowledge [27]. *Explicit knowledge* refers to knowledge that can be externalised in terms of some representation. In representation of knowledge, we refer to the process of encoding knowledge in terms of some language on some medium. Our focus is on the communication of system development knowledge by way of *explicit* representations. In other words, *explicit knowledge*, where the representations pertain to an existing or future system; its design, the development process by which it was/is to be created, the underlying considerations, etc.

3 Conceptual Framework

Following the general way of thinking as discussed in the previous section, in this section we present a conceptual framework. This framework will be used in the further development of our theories based on its use in practical settings.

3.1 System Development Knowledge

We start by briefly exploring the kinds of knowledge that are relevant to a system and its development, in other words: the knowledge topics that can be discerned. During system development, members of the system development community will create and exchange knowledge pertaining to different topics. A first distinction can be made between:

Target domain – Knowledge pertaining to the *system* being developed.
Project domain – Knowledge about the *development process* that brings forth the system.

We have borrowed the terms *target domain* and *project domain* from the Information Services Procurement Library (ISPL) [28]. For both of these knowledge domains, further refinements can be made with regards to the possible topics. One can identify the following additional characterizations:

Perspective – Artifacts, such as systems, can be considered from different perspectives. Some examples are: (1) Business, application, and infrastructure aspects of a (computerized) information system; (2) Social, symbolical, and physical aspects of a system; (3) Process, information, actors, and technology featuring in a system.

Scope – Given a domain, such as a system or a development project, several scopes can be identified when approaching the domain. Some examples are: (1) enterprise wide; (2) department specific; (3) task specific.

Design chain – When considering the design of some artifact, a distinction can be made between: (1) the **purpose** for which an artifact is needed; (2) the **functionality** which the artifact should provide to its environment; (3) the **design** of the artifact, i.e. *how* it should realize the functionality; (4) the **quality** of the artifact, i.e. *how well* it should do so; (5) the **cost** at which it will/may do so, and at which it may be constructed.
Based on these distinctions, knowledge topics can be characterized in terms of their focus on, for example, functionality or quality in *isolation*, or their focus on bridging the gaps between purpose, functionality and design in terms of the underlying *design rationale*.

Historical perspective – Given an artefact with a design, one may consider different versions of this design over time. One could, for example, make a distinction between a strategic (5-10 years), a tactical (1-5 years), and an operational perspective (now).

Abstraction level – When considering a domain, one may do so at several levels of abstraction. Various forms of abstraction can be distinguished, for example type-instance, generalisation, is-a-kind-of, encapsulation, and the hiding/encapsulation of implementation details.

In general, each of the above characterizations of knowledge topics applies to both target and project domains. As mentioned before, depending on their concerns, stakeholders may be interested in different knowledge topics.

3.2 Explicitness of Knowledge

Given our focus on system development, a more precise classification can be made with regard to the *level of explicitness* as mentioned in section 3.2. Based on [28, 29], the following dimensions of explicitness for representations of system development knowledge (pertaining to both target domain and project domain knowledge) can be identified:

Level of formality – The degree of formality indicates the type of representation language used. Such a language could be formal, in other words a language with an underlying well-defined semantics in some mathematical domain, or it could be informal –not mathematically underpinned; typically texts in natural language, graphical illustrations, animations, etc.

Level of quantifiability – Different aspects of the designed artefact, be it (part of) the target or the project domain, may be quantified. Quantification may be expressed in terms of volume, capacity, workload, effort, resource, usage, time, duration, frequency, etc.

Level of executability – The represented knowledge may, where it concerns artefacts with operational behaviour, be explicit enough so as to allow for execution. This execution may take the form of a simulation, a prototype, generated animations, or even fully operational behaviour based on executable specifications.

Level of comprehensibility – The knowledge representation may not be comprehensible to the intended audience. Tuning the required level of comprehensibility of the representation, in particular the representation language used, is crucial for effective communication. The representation language may offer special constructs to increase comprehension, such as stepwise refinements, grouping/clustering of topically related items/statements, etc.

Level of completeness – The knowledge representation may be complete, incomplete, or overcomplete with regard to the knowledge topic (see previous subsection) it intends to cover.

3.3 Conversation Strategies

The knowledge transformations as discussed in section 2.4 are brought about by conversations. The scope of these conversations may range from 'atomic' actions involving a small number of actors, via discussions and workgroups, to the development process as a whole. This has been illustrated informally in figure 1.

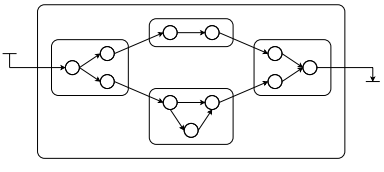

Fig. 1. Example sequence of conversations

Each conversation is presumed to have some *knowledge goal*: a knowledge state which the conversation aims to achieve (or maintain). This knowledge state can best be regarded as a multi-dimensional vector, positioning: (1) the knowledge topic (see section 3.1); (2) the level of explicitness of the knowledge (see section 3.2); (3) the level of sharing (see section 2.4).

In achieving a knowledge goal, a conversation will follow a *conversation strategy*. Such a strategy is needed to achieve the goal of the conversation, starting out from the current state:

Knowledge goal – The knowledge goal; a desired knowledge state which the conversation will aim to achieve/contribute towards.

Initial state – The initial knowledge state as it holds at the start of the conversation.

Conversations take place in some situation in which resources may or may not be available for execution of the conversation. A conversation situation may be characterised further in terms of *situational factors* [28]. We identify three classes of situational factors:

Availability of resources – Refers to the availability of resources that can be used in a conversation. The availability of resources can be refined to more specific factor such as: time for execution, actors present, intellectual capacities required from the actors, and financial means.

Complexity – The resources needed for the conversation, the knowledge being conversed about, etc., will exhibit a certain level of complexity. This complexity also influences the conversation strategy to be followed. Examples of such complexity factors (inspired by [28]) are: heterogeneity of actors involved, quantity of actors involved, complexity of technology used, complexity of knowledge being conversed about, and size of the gap between the initial knowledge state and the desired knowledge state.

Uncertainty – In determining a conversation strategy fit for a given situation, assumptions will have to be made about the knowledge goal, the initial state, the availability of resources, and the complexities of these factors. During the execution of a conversation, some assumptions may prove to be wrong. For example: the commitment of certain actors may be lower than anticipated (initial state); materials needed for a workshop may not be available on time (resources); during a requirements elicitation session it may emerge that the actors involved do not (yet) have enough knowledge about the future system and its impact to formulate/reflect on the requirements of the future system (initial state). Typical uncertainty factors could relate to: the knowledge goal, the initial state, the abilities of actors involved, the availability of resources, and the complexities as discussed above.

In formulating a conversation strategy, all of the above factors should be taken into account. A conversation strategy should typically cover at least the following elements:

Execution plan – As mentioned before, a conversation can be composed of sub-conversations. Each of these sub-conversations focusses on a sub-goal, but they all contribute towards the goal of the conversation as a whole. The execution plan of a (composed!) conversation consists of a set of sub-conversations, together with a planned execution order.

Description languages – The description languages to be used in the conversation(s).

Media – The kind of media to be used during the conversation(s).

Cognitive mode – The *cognitive mode* refers to the way in which knowledge is processed/gathered by the collective of actors involved in a conversation. Typically, a distinction is made between an analytical and experimental approach.

Social mode – The *social mode* is the way in which the actors executing the system development process collaborate with the actors from the business domain. We distinguish between an expert-driven and a participatory approach.

Communication mode – A small number of basic patterns of communication can be distinguished, as covered by combinations of the some basic factors:

speaker-hearer ratio, requirements on hearer response, allowed time-lag, locality, and persistency. Combinations of these factors can be used to typify many different modes of communication, which can have a major impact on the resources required for communication and the likelihood a knowledge goal is achieved.

4 Guidelines for the Usage of Modelling Techniques

This section concerns guidelines that should help practitioners in selecting modelling techniques for communication tasks at hand. These guidelines are based on interviews with a number of architects. In general, the use of a modelling technique will pass through a number of phases. These phases are:

Phase 1 – Scoping: Select (an) appropriate modelling technique, select the (sub)domain that needs to be represented/modelled in terms of (a) model(s), and determine the constraints that apply to the domain being modelled.

Phase 2 – Creation of models: Create/select the actual content of the models. This can pertain to the selection of a part of a larger (pre-existing) model, or the creation/refinement of a part of a model.

Phase 3 – Validation: Validate the resulting model. Do the stakeholders agree to the fact the model is a correct representation of the actual/intended situation?

Phase 4 – Obtaining commitment: If *agreement* has been reached among the key stakeholders involved, the next step will be to create *commitment* for the results. In other words, do the stakeholders commit themselves to the (potential) impact of what is described by the model?

Phase 5 – Informing: Inform other stakeholders of the results. These stakeholders will be those members of the development community whose explicit commitment has, in a *conscious* decision, been considered not to be crucial.

Note that these phases will not necessarily be executed in a linear order. Practical circumstances usually dictate a more evolutionary approach. Any modelling techniques is to be used in activities from each of the above phases. The guidelines resulting from the interviews are categorised according to these phases, and are discussed below.

4.1 Scoping

The importance of focussing on the concerns of stakeholders and the extent to which a specific modelling technique addresses these concerns, was confirmed by the outcomes of the interviews:

- *Quite a number of important business-related concerns are left out by the regular (technology-oriented) modelling techniques. Most notable are legal, ethical and economical aspects of a system.*
- *When communicating to business managers, only those models are needed that enable a discussion of factors deserving special attention. Typically,*

these are factors that have a high impact if they fail and also have a high risk of indeed failing. For communication with the actual software developers, on the other hand, more detailed models are crucial.

The selection of modelling techniques should be deliberate and based on rational considerations. What is more, architects stated that this decision, and its rationalisation, must be readily available for communication during the different phases:

- *The modelling techniques to be used, the goals for which the models are to be created, as well as the underlying rationale, should be known beforehand.*
- *It is quite possible that a stakeholder (usually a technology-oriented one) will ask for more detail in a model than you can give her, or want to give her, in that particular phase of the project. A modeller/designer should be prepared to better clarify the goals of the particular model and phase, and why the requested details are not yet relevant (or even harmful).*
- *Even if the stakeholders do not always need to know why a certain model or way of modelling is used, the modeler should know this perfectly well, at all times.*

4.2 Creation of Models

During the creation of a model, in particular when actual modelling (i.e. not elicitation etc.) is concerned, it is considered sensible to limit the number of participants in a conversation:

- *Graphical models may or may not be used in communication with stakeholders, but most actual modelling is done by individuals, two people at most. Genuine group modelling sessions are very rare.*

During the early stages of system design, it is often considered a bad thing to "think" in terms of "solutions". According to some of the interviewed architects, however, it is sometimes defendable to let this "thinking in terms of examples" run its course, as long as the results are expressed at the correct level of abstraction:

- *When detailed modelling takes place in a cooperative setting, give informants some room to think in terms of "solutions" even if pure requirements thinking (what, not how) does not officially allow for this.*
  Most people just think better in terms of in concrete solutions; it is a vital part of their creativity. Just be sure that requirements thinking is returned to in due time.

The above observation is actually a concretisation of a more abstract observation:

- *When discussing models with stakeholders and informants, in particular when trying to establish a common understanding, it is sensible to discuss different scenarios and alternatives to the model being considered. Doing so leads to an exploration of the meaning and impact of the model taking shape, and also leads to improved mutual understanding.*

The use of concrete examples is but a way to make the different potential scenarios more tangible.

The graphical notation that is part of a modelling technique should be approached flexibly when it comes to communicating with stakeholders (in particular non-technical ones):

- *If people are not used to or prepared to deal with abstract graphical models, do not use them. It is pointless. Use other forms of visualization; anything. Iconised diagrams work particularly well. However, be prepared to point out the relation between the alternative visualisation and your abstract models if asked to. Also do use your models for your own insight, and for generating extra questions you can ask your stakeholders.*

Finally, during a modelling session, several things may come to the fore that will influence the further process. External events may occur that are a threat to the process as a whole:

- *Be prepared to stop modelling if executive commitment is withdrawn. It may be frustrating, but from a business perspective it may also be crucial. It is simply part of a flexible project setup.*
- *If the informants turn out to be less informed than expected, it is better to stop than to try to "make the best of it" and produce an ill-conceived model.*

4.3 Validation

In validation, a clear difference should be made between validation of content (qualitative validation, by modelers and experts) and validation in terms of commitment (by executives). Both are crucial, but very different. Obtaining (and validating) commitment is discussed in the next subsection.

Whether good mutual communication and understanding about a model is being reached is often a matter of intuition:

- *If people involved have a mutual feeling that "their thoughts are well in sync", then dare to trust that feeling. However, if the opposite is the case, be prepared to invest in substantial discussion of concrete examples – or face the dire consequences of poor validation.*

Validation is an activity that should be conducted in limited groups:

- *"Feedback Rounds" involving a larger number of people, by e-mail or printed documentation, do not really work. If you want feedback that is worth something, find key people and discuss the models, preferably face to face.*

4.4 Obtaining Commitment

Obtaining commitment for a specific architectural design involves obtaining commitment for the *impact* of this design on the future system and its evolution, as well as the *costs/resources* needed to arrive at this future system. This means that the message that one needs to get across to the stakeholders involves:

- What are the major problems in the current situation?
- How bad are these problems (to the concerns and objectives of the stakeholders)?
- How will this improve in the new situation? (Benefits)
- At what costs will these improvements come?

When discussing costs and benefits with stakeholders, it is important to realize the following:

- *Make costs and benefits as SMART (Specific, Measurable, Attainable, Realisable, and Time-bound) as possible.*
- *Make sure that the stakeholders agree, up front, to the criteria that are used to express/determine costs and benefits. It is their commitment that is needed. They will be the judge. Let them also decide what they want to base their judgement on! Create shared responsibility towards the outcomes.*

Selecting the stakeholders that should be involved when obtaining commitment is also of key importance. Involving the wrong stakeholders, or leaving out important ones, will have obvious repercussions. At the same time, selecting too large a group of stakeholders may make the process bog down.

- *Though ideally "everyone" should be heard, this is generally a practical impossibility. Therefore, choose your experts carefully. Aim for the opinion leaders, and also accept that you cannot please everyone. Be aware that some people will not be perfectly satisfied, prepare for it, and deal with it.*
- *People who actually make the decisions are usually those who are just outside the group of people who really know what is going on. Make sure that the former people are* also *involved and aware of what is happening.*
- *Careful reflection about the stakeholders that should be involved in obtaining commitment often lacks in practice. There is a natural tendency to involve only like-minded stakeholders, i.e. to "preach to the converted".*

The architects interviewed also noted some potential pitfalls in obtaining commitment:

- *Communication between a representative and her group is vital, but may also do damage if conducted clumsily. Interestingly, too much communication may be a bad thing: it may create unnoticed and uncontrolled discussion outside the main discussion, leading to twisted conceptualisations and expectations. Therefore, communication between representatives and their group should at least be monitored.*
- *Getting executive commitment may actually be technologically dictated. If their business is heavily technological, business people do not see technology as secondary, and will only commit to anything if they are assured that "their organization will be able to run it".*
- *Sharing design decisions and their underlying considerations at a late stage has a negative impact on the commitment of stakeholders. Start commitment building early on in the process.*

The latter point clearly confirms that the linear ordering of the "technique use phases" as provided at the start of this section should not be applied strictly.

4.5 Informing

Once commitment from the opinion leaders has been obtained, other stakeholders may be informed about future plans and their impact. In doing so, it still makes sense to concentrate on cost/benefit considerations when trying to "sell" the new system. Below, we have gathered some observations that apply to the informing phase. However, due to their general communicative nature, some of these observations are also applicable to the creation, validation, and commitment phases.

- *Do not impose presumed information system terminology on true business people. Use* their *terminology. Even a concept like "service" is suspect because it is relatively technology-oriented and often unknown by stakeholders that are strictly on the business side.*
- *Models are particularly important in giving stakeholders a feeling that they are "part of the larger whole". Often, just knowing where in the model "they can be found" is important to stakeholders, even if they do not understand the fine points of the model.*
- *Even if people are willing to and capable of reading models thoroughly, text (spoken or written) needs to be added. Models alone never suffice.*

5 Conclusion

We presented a progress report on one of our ongoing research efforts. We discussed our way of thinking regarding system development as a communication-driven knowledge transformation process, and refined this way of thinking in terms of a conceptual framework. Finally, some first results have been discussed. These results are guidelines based on interviews with (enterprise) architects, which were conducted as part of the "diagnostic" stage of the action research paradigm.

We are currently in the process of initiating the "therapeutic" stage of the action research paradigm. Our plan is to participate in selected activities in development processes taking place in large Dutch cooperations and/or governmental agencies.

References

1. Bubenko, J.: Information System Methodologies - A Research View. In Olle, T., Sol, H., Verrijn-Stuart, A., eds.: Information Systems Design Methodologies: Improving the Practice. North-Holland/IFIP WG8.1, Amsterdam, The Netherlands, EU (1986) 289–318.
2. Avison, D., Wood-Harper, A.: Information Systems Development Research: An Exploration of Ideas in Practice. The Computer Journal **34** (1991) 98–112.
3. Avison, D.: Information Systems Development: Methodologies, Techniques and Tools. 2nd edn. McGraw-Hill, New York, New York, USA (1995). ISBN 0077092333

4. Bernus, P., Mertins, K., Schmidt, G., eds.: Handbook on Architectures of Information Systems. International Handbooks on Information Systems. Springer, Berlin, Germany, EU (1998). ISBN 3-540-64453-9
5. Olle, T., Hagelstein, J., Macdonald, I., Rolland, C., Sol, H., Assche, F.v., Verrijn-Stuart, A.: Information Systems Methodologies: A Framework for Understanding. Addison-Wesley, Reading, Massachusetts, USA (1988). ISBN 0-201-54443-1
6. Bommel, P.v., Hofstede, A.t., Weide, T.v.d.: Semantics and verification of object-role models. Information Systems **16** (1991) 471–495.
7. Hofstede, A.t., Weide, T.v.d.: Expressiveness in conceptual data modelling. Data & Knowledge Engineering **10** (1993) 65–100.
8. Bronts, G., Brouwer, S., Martens, C., Proper, H.: A Unifying Object Role Modelling Approach. Information Systems **20** (1995) 213–235.
9. Creasy, P., Proper, H.: A Generic Model for 3-Dimensional Conceptual Modelling. Data & Knowledge Engineering **20** (1996) 119–162.
10. Campbell, L., Halpin, T., Proper, H.: Conceptual Schemas with Abstractions – Making flat conceptual schemas more comprehensible. Data & Knowledge Engineering **20** (1996) 39–85.
11. Hoppenbrouwers, J., Vos, B.v.d., Hoppenbrouwers, S.: Nl structures and conceptual modelling: Grammalizing for KISS. Data & Knowledge Engineering **23** (1997) 79–92.
12. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modelling Language User Guide. Addison-Wesley, Reading, Massachusetts, USA (1999). ISBN 0-201-57168-4
13. Brooks, R.: Studying programming behavior experimentally: The problems of proper methodology. Communications of the ACM **23** (1980) 207–213.
14. Avison, D., Lau, F., Meyers, M., Nielsen, P.: Action research. Communications of the ACM **42** (1999) 94–97.
15. Baskerville, R.: Investigating Information Systems with Action Research. Communications of the Association for Information Systems **2** (1999).
16. Blum, F.: Action research – a scientific approach? Philosophy of Science **22** (1955) 1–7.
17. Derksen, C., Frederiks, P., Weide, T.v.d.: Paraphrasing as a Technique to Support Object-Oriented Analysis. In Riet, R.v.d., Burg, J., Vos, A.v.d., eds.: Proceedings of the Second Workshop on Applications of Natural Language to Databases (NLDB'96), Amsterdam, The Netherlands (1996) 28–39.
18. Frederiks, P., Weide, T.v.d.: Information modeling: the process and the required competencies of its participants. In Meziane, F., Métais, E., eds.: 9th International Conference on Applications of Natural Language to Information Systems (NLDB 2004). Volume 3136 of Lecture Notes in Computer Science., Manchester, United Kingdom, EU, Springer-Verlag, Berlin, Germany, EU (2004) 123–134.
19. Bleeker, A., Proper, H., Hoppenbrouwers, S.: The role of concept management in system development – a practical and a theoretical perspective. In Grabis, J., Persson, A., Stirna, J., eds.: Forum proceedings of the 16th Conference on Advanced Information Systems 2004 (CAiSE 2004), Riga, Latvia, EU, Faculty of Computer Science and Information Technology, Riga Technical University, Riga, Latvia, EU (2004) 73–82. ISBN 9984-9767-0-X
20. Proper, H., Hoppenbrouwers, S.: Concept evolution in information system evolution. In Gravis, J., Persson, A., Stirna, J., eds.: Forum proceedings of the 16th Conference on Advanced Information Systems 2004 (CAiSE 2004), Riga, Latvia, EU, Faculty of Computer Science and Information Technology, Riga Technical University, Riga, Latvia, EU (2004) 63–72.

21. Veldhuijzen van Zanten, G., Hoppenbrouwers, S., Proper, H.: System Development as a Rational Communicative Process. Journal of Systemics, Cybernetics and Informatics **2** (2004). http://www.iiisci.org/Journal/sci/pdfs/P492036.pdf
22. Embley, D., Kurtz, B., Woodfield, S.: Object-Oriented Systems Analysis – A model-driven approach. Yourdon Press, Englewood Cliffs, New Jersey, USA (1992). ASIN 0136299733
23. Halpin, T.: Information Modeling and Relational Databases, From Conceptual Analysis to Logical Design. Morgan Kaufman, San Mateo, California, USA (2001). ISBN 1-55860-672-6
24. Cruse, A.: Meaning in Language, an Introduction to Semantics and Pragmatics. Oxford University Press, Oxford, United Kingdom, EU (2000). ISBN 0-198-70010-5
25. The Architecture Working Group of the Software Engineering Committee, Standards Department, IEEE: Recommended Practice for Architectural Description of Software Intensive Systems. Technical Report IEEE P1471-2000, The Architecture Working Group of the Software Engineering Committee, Standards Department, IEEE, Piscataway, New Jersey, USA (2000). ISBN 0-738-12518-0 http://www.ieee.org
26. Peirce, C.: Volumes I and II – Principles of Philosophy and Elements of Logic. Collected Papers of C.S. Peirce. Harvard University Press, Boston, Massachusetts, USA (1969). ISBN 0-674-13800-7
27. Nonaka, I., Takeuchi, H.: The knowledge-creating company. Harvard Business Review (1991) 97–130.
28. Franckson, M., Verhoef, T., eds.: Managing Risks and Planning Deliveries. Information Services Procurement Library. ten Hagen & Stam, Den Haag, The Netherlands (1999). ISBN 9076304831
29. Proper, H., ed.: ISP for Large-scale Migrations. Information Services Procurement Library. ten Hagen & Stam, Den Haag, The Netherlands, EU (2001). ISBN 9076304882