Formal Modelling as a Grounded Conversation

S.J.B.A. Hoppenbrouwers
Institute for Computing
& Information Science
Radboud University Nijmegen
Toernooiveld 1
6525 ED Nijmegen
The Netherlands, EU
stijnh@cs.ru.nl

H.A. Proper
Institute for Computing
& Information Science
Radboud University Nijmegen
Toernooiveld 1
6525 ED Nijmegen
The Netherlands, EU
E.Proper@cs.ru.nl

Th.P. van der Weide
Institute for Computing
& Information Science
Radboud University Nijmegen
Toernooiveld 1
6525 ED Nijmegen
The Netherlands, EU
TvdW@cs.ru.nl

September 20, 2005

Published as:

S.J.B.A. Hoppenbrouwers, H.A. (Erik) Proper, and Th.P. van der Weide. Formal Modelling as a Grounded Conversation. In G. Goldkuhl, M. Lind, and S. Haraldson, editors, *Proceedings of the 10th International Working Conference on t he Language Action Perspective on Communication Modelling (LAP'05)*, pages 139–155, Kiruna, Sweden, EU, June 2005. Link'opings Universitet and Hogskolan I Boras, Link'oping, Sweden, EU.

Abstract

As part of an ongoing, broader theoretical study concerning a communication/conversation perspective on information system development, we focus in this paper on a specific sort of conversation in IS modelling: conversations for formal modelling, which are to bridge the gap between informal (NL-based) and formal (mathematics-based) representations and interpretations. We provide a communication-based analysis of the formal modelling process, and discuss why it is crucial that the (formal) structures in the various kinds of models are somehow grounded in the structures of agreement/commitment that underly the development conversations. We explain how looking at modelling as communicative *behaviour* may help achieve grounded models, thereby improving their validity in context.

1 Introduction

In this paper, we explore application of the LAP to the field of information system development as a cooperative, communicative process. We view system development as analogous to general organization, but with some specific and highly interesting issues of its own. We focus in particular on a prominent, even fundamental issue in system development that so far does not seem to be subject of much discussion within the LAP community: the "informal-formal divide". With this term we refer *not* to well-known, paradigmatic differences concerning the nature of language use and language meaning (for this discussion, see [31, 36], among others), but to related aspects of language functionality and use that play a central role in system development *practice*: the widespread parallel (even overlapping) use of "informal" and "formal" languages in information system development. This issue is most acutely raised where "informal descriptions" are to be "formalized". This is where in some cases, LAP-like (functionalist, subjectivist) ways of thinking clash violently with their "representationalist", "positivist" counterparts, causing practical problems that originate in fundamental differences between methods and forms of cooperation.

In LAP literature, ample work exists that, directly or indirectly, aims to support or improve IS development. Also, the LAP community has brought forth several methods/techniques related to

IS development. For example, the DEMO method [32, 33] is basically a method for business process modelling, but it does provide many explicit and useful links to IS development. In particular, the combination of DEMO with ORM [17] is relevant in this respect [9]. Another example of a relevant approach can be found in [15]. More generally, many fundamental issues (mostly language- and communication related) concerning information and computer systems have been discussed within the LAP.

However, to our knowledge the LAP has never been applied directly for analysis of (information) *system development processes* as such. With this we refer to the whole of activities related to the creation of socio-technical systems: conception, definition, design, construction, deployment, and management of information systems (usually computerized) and the environment they function in (i.e. the people that use them and the organizations they support). Importantly, this *includes* highly technology-oriented domains, phases, and activities in the development process, and in relation to those, a view and use of language that is very different from the one advocated under the LAP [19, p85-7]. In a way, we thus inverse the typical (and quite acceptable) LAP attitude of turning away from the positivist/representationalist interpretation of language and communication. Rooted firmly in the LAP, we turn around and see what can be contributed to bridge the gap between the LAP and the positivist/representationalist perspective.

The diverse languages used in IS development, and the use made of them, relate to a heterogeneous group of stakeholders, including project managers, (prospective) users acting as informants, information architects, but also software engineers and programmers. For an actual computerized IS system to emerge, all of these stakeholders, and all the communication and languaging their work involves, have to play their unique part. Taking a strictly fundamental (typically "cybernetic") point of view, we can even include "automated agents" in our view; for example, compilers or CASE tools. We are the first to acknowledge that there is a crucial and at times problematic difference between "language" as "used by" a program/compiler and, say, a system user. However, given that the science of computational machinery and its use of (formal) languages undeniably has its own merits, we believe that in the context of the IS development process, it is more useful to study the complicated pragmatic relations (and possibly, similarities) between the two views rather than to emphasize the differences between them, crucial though such emphasis has been –and still is– for emancipatory reasons and for creating awareness of the complex issues at hand.

In principle, then, we include in our ongoing study of study of IS development *all communication and language use* (i.e. in the broadest sense) taking place in the "system development community" [19, p79-82]. However, in this paper we focus on formal languages and their use is IS development, in particular for "formal description". Programming languages, with their relation to both "socio-cognitive" and "operational" semantics [19, p85] are a traditionally prominent example that spring to mind, but also in less technology-oriented domains of analysis, formal languages are used, e.g. for definition of formal ontologies, formal process models, formalized business rules, etc. [20]. In the remainder of this paper, we will generally refer to "formal modelling", but use this term in the sense of "formal description" in general, hence including coding in programming langages in principle.

Communication is not only a fact of life in IS development practice, it determines for a large part how development processes take place and whether or not a system turns out successful [22]. We believe it is indeed a good idea to view/study IS development from a communicative (language action) perspective. The basic arguments behind this belief are the following.

1. **Combine various approaches to (the functionality of) language** – Many language or information related issues lie at the core of understanding system development, including the clash between (both the creation and the use of) *formal* (say, mathematics-based) and *informal* (say, natural language-based) descriptions. This concerns central issues of (different approaches to) syntax, semantics, and pragmatics. Note that though, for the sake of the argument, we assume that indeed there *is* a fundamental difference between the formal and informal approach to language, the study of the nature of this difference, and ways of bridging the supposed gap, is part of our fundamental interests.

2. **System development as a form of social construction of reality** – Cooperation and interaction between participants/stakeholders in system development can be fruitfully approached from a LAP perspective as well. This emphasises the social interaction, agreement, and knowledge sharing aspects involved: information system development as a process of cooperative development of very specific, *grounded* representations with very different underlying requirements and goals (again including the informal/formal divide).
3. **Combine different theoretical issues under one framework** – Issues falling under 1 and 2 are ultimately intertwined (see section 2.4 below). To understand and possibly solve them, the theoretical frameworks applied should preferably be compatible (if not similar). The LAP, possibly in combination with various flavours of semiotics, seems a promising candidate framework for such an approach. Central are aspects like the crucial link between agreement about the *validity* of statements and agreement about the *meaning* of statements [23, 19], and combining the *representational* view on some meaning (a meta-linguistic representation) with a *behavioural* view on language and meaning (i.e. what people do when they discuss it and reach agreement about it). This, we feel, is very much a LAP approach.
4. **Combine organizational communication and system development issues** – Along similar lines, integration of issues of information and communication in organizations (the mainstream LAP focus) should ultimately be integrated with IS development (this concerns, for example, feedback loops and evolutionary development [21]). Again, the LAP plays a key role in enabling us to perform the required integration on a fundamental level.

In section 2, we further clarify our view of IS development as a conversation. This paves the way in particular for a discussion of arguments 1-3. Next, the 1st argument will be elaborated on (in section 3), after which we briefly present our basic view on modelling as a dialogue (section 4). This discussion will again touch issues related to argument 3. In section 5, the 2nd argument will be focused on (this is where the "grounding" of the conversations comes in). Finally, we present some plans for further research. Note that argument 4. is not further elaborated on in this paper.

2 IS development and modelling as a conversation

As said, we base our study of (information) system development on the idea that IS development processes are long, complex, and multi-faceted *conversations*, or, if you like, many interlinked sub-conversations [22]. At the center of our current analysis lie *modelling conversations*, but other types are also included in our general way of thinking.

In the past we have already taken a communication-driven perspective on modelling activities in information system development [8, 18, 14, 3, 29], as well as on the act of system development itself [34]. We are certainly not not alone in doing so [26, 10, 17]. Our main point of reference currently still is the ORM method for *domain modelling*. ORM is also extensively used in business rule formulation and requirements engineering. It is a formal modelling language ("way of modelling") that is rooted in, and can be mapped to, set theory [4] and predicate logic [16]. ORM includes a rather elablorate "way of working" that involves expressions in normalized NL phrases. This way of working has been our starting point in exploring communication-based modelling as behaviour. We intend to expand our analysis to various other flavours of formal modelling, including full blown predicate logic. We consider formal domain modelling to be an excellent "step up" for this, because it traditionally is often a first step in the formalization process.

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 [22]. 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.

We do *not* claim that viewing information system development as a knowledge transformation process is new [25]. Our aim is to use this perspective on system development to better understand, for example, the requirements that should be set for modelling languages, and the strategies that (should) drive system development conversations [22].

2.1 Knowledge in IS development

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. The system development community harbours knowledge about the system being developed. To be more precise, "objects" in the system development community (human as well as artificial) 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.

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*. We identify three dimensions for the knowledge states of the development community: (1) topic, (2) level of sharing, and (3) level of explicitness. The topical dimension is not further discussed in this paper; the other two will be elaborated on below.

Knowledge needs to be *introduced* into the development community first, either by creating the knowledge internally or by importing it from outside the community. Once knowledge has been introduced to a community, it can be *shared* among different knowledge carriers. The sharing of knowledge between different actors may progress through a number of stages. We distinguish three major stages:

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

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

Committed – Actors who agree to a specific knowledge topic may decide to *commit* to this knowledge: 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 "beholder", however, will typically be an actor in the system development community. We can, therefore, presume that actors in the system development community will be able to (and have a reason to) judge the level of sharing and explicitness of knowledge between actors, and communicate about this.

The actual knowledge that is harboured by a knowledge carrier can 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 [28]. 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.

2.2 IS development conversations

The knowledge transformations discussed above 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. The design conversation at large starts off with some initial (sub-)conversation, which

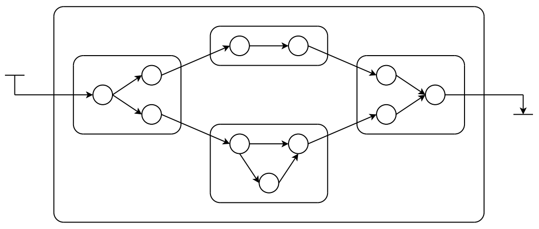

Figure 1: Example sequence of conversations

leads to further conversations (each with their own specific goal and participants). Generally, the main conversation first shows divergation (“branching off”), then convergation. Importantly, the conversations can only for a limited part be planned; which conversations are to follow up previous ones largely depends on the intermediate results, which cannot be predicted.

Each (sub-)conversation is presumed to have some *knowledge goal*: a knowledge state which it aims to achieve (or maintain). This knowledge goal can best be regarded as a multi-dimensional vector, positioning (as mentioned) the knowledge topic, the level of explicitness of the knowledge, and the level of sharing. In achieving a knowledge goal, a (sub)-conversation will follow a *conversation strategy*. Such a strategy is needed to achieve the goal of the (sub-)conversation, starting out from the current state:

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* [12]; elaboration on specific situational factors goes too far for the current discussion.

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.

2.3 The role of (formal) modelling in IS development

At various stages in system development, formal descriptions (data structures, formal protocols, process models, computer programs, etc.) may play a role. From a functional perspective, there are two main, interrelated reasons why they may be needed:

1. Because a formal description is to be used as input for, or otherwise as part of, a computation
2. Because a formal descriptions is used in a context that requires rigid logical or mathematical description and "thinking", reflecting a desire to achieve (within some clearly delimited domain or activity) a very high degree of certainty and predictability (arguably, rationality).

Quite obviously, reason 1 is for a large part the result of technology and engineering based on 2. However, in many cases the use of computational technology seems to have become a goal in itself, detached from its mathematical/rational roots.

There is an important link between "formality" and the *level of explicitness* (the third dimension of knowledge states); formal descriptions are a special flavour of highly explicit knowledge representation. Many different sorts of models and formalisms (i.e. formal languages) exist and may be used. Importantly, some formalisms are very "heavy" (for example, require a completely formalized, i.e. mathematics-based *semantics*), whereas others are much "lighter" (for example, they may just require a well-defined syntax that can be mathematically captured, but without a formally described semantics). We refrain from discussing detailed classification of formalisms here. However, we can safely state that when it comes to the use of formalisms in IS development practice, we are referring mostly to formalisms. These may be light (for example, XML) or heavy (for example, predicate logic or C++), but either serve *operational* goals in IS development, and are "texts" delivered as part of the actual development process, usually by specialist modelers (arguably including programmers) that are members of the system development community.

2.4 Uncertainty reduction as a chief goal

Another word about the goals of conversation and strategies deployed to achieve those goals. Identifying and resolving 'vagueness' is a major part of the refinement task in IS development and modelling [34]. Consequently, many conversation strategies aim for *uncertainty reduction* of some sort. Two main *types of uncertainty* can be relevantly identified [30], each raising a different class of questions within the modelling conversation:

Epistemic uncertainty This uncertainty exists in the mind of the individual expert, and reflects the incomplete knowledge a domain expert has of the domain. The uncertainty is a result of limited mental resources and limited time to investigate the domain [2].

Linguistic uncertainty This is uncertainty introduced in communication between participants, occurring when an expression in common language has more than one possible interpretation. For flexible common languages, such as natural language, this may occur frequently. Very constrained languages, on the other hand, may prevent the occurrence of multiple interpretation, at the expense of limited expressive power.

Although precise methods for handling the various types of uncertainty depend on the situation, several general approaches can be distinguished. Lipshitz and Strauss [24] investigated how decision makers handle uncertainty; they found that four general ways occured:

Reduction of uncertainty Collect additional information, e.g. by asking.

Assumption based reasoning Fill gaps in knowledge by making plausible assumptions.

Weighing pros and cons of various alternatives.

Suppression Ignore uncertainty, at least for a while.

We view these four approaches as basic *strategies* for dealing with uncertainty in modelling.

2.5 Communication about concepts: languages or models?

Since languages, texts, and the interpretations given to those texts play a key role in IS development [23, 19, 34], and much effort in IS development goes into clarifying and refining statements, language-related issues (choice of languages and concepts, formal-informal differences) are crucial for understanding system development conversations. Formal modelling by no means is the only place where systematic and precise statements are required; it is mostly that formalisms pose very specific and above all *explicit* demands concerning how a statement should be structured and what it should and should not include (and in some cases, what its meaning should essentially be composed of; in practice, formal interpretation is usually left to machines). With respect to both informal and formal descriptions, therefore, communication about "concepts" is an integral part of the system development process.

There is a terminological complication here that requires some discussion at this point. There often is confusion about the difference between "languages" (in particular, modelling languages) and "models". Though at first glance it seems clear enough that models are the descriptions as such, and languages provide the concepts to express them in, many models (in particular, "conceptual models") provide detailed descriptions of the meaning of concepts (both atomic and complex) that in one way or another are, or could also be, part of a "language". What is a "language" and what is a "model" largely depends on pragmatic choices made for particular modelling situations. Thus, models concerning quite specialized matters and used among specialists may be expressed in a highly specialized language and, given that the language is sufficiently known among the experts, may result in fairly simple models. The burden of specialization is thus put on the language. Conversely, in a modelling environment where not much pre-defined domain-specific concepts are required or available, modelers will do well to resort to generic modelling languages (e.g. the UML, ORM, or basic predicate logic). Similar differences between specialized and generic *terminologies* exist within "natural language" as a modelling language, but specialist terminologies are usually much less strictly demarcated and are commonly mixed with generic language, or other specialized terminologies [19, 158-9].

Consequently, conversations that concern the same topic may in fact involve "concepts of a language" or "part of a model" (or, of course, some combination of the two), depending on the view of the modelers and of the frame of reference they work under. Communication about language, then, is distinguishable from communication about a model only in context of a particular domain and the methodological choices made in for that domain. In any case, communication about concepts serves as a prominent strategy in modelling, and therefore also in modelling for IS development purposes. This is particularly important in view of our communication-based perspective on IS development and modelling. It also raises questions as to the fundamental difference between "linguistic uncertainty" and "epistemic uncertainty". However, from a pragmatic, contextual point of view, the difference is still relevant enough, as within context, the seperation of langage and model is usually clear enough.

3 Bridging the informal-formal divide in IS development

We have already briefly discussed the differences between "formal" and "informal" languages (section 1). We will not enter here into an elaborate discussion of the different cultural and theoretical opinions and practices underlying these two main flavours of use and description of language and meaning. Instead, we focus on a possible direction for creating not just peaceful coexistence but actual synergy between the two views, if not on a fundamental then at least on a pragmatic level.

3.1 Basic options for resolving the informal-formal divide

For the informal-formal divide to be dealt with, we see three hypothetical options:

Formalize the "informal" world This is exactly the sort of approach that has led to counter-reactive approaches like the Language Action Perspective. The representationalist/positivist view on language ignores some crucial issues that are vital in a "human-oriented" context (subjective, contextualized, social, and also more oriented on realistically human cognition). Also, there are great practical limitations to formalization of the "informal" world.

Reject the formal world, i.e. "informalize" the formal world In its radical form, this is indeed only a hypothetical option, since it would entail, among other things, rejecting all computer technology. This would be throwing away the child with the bath water. A less radical version, however, is much more realistic:

Separate informal from formal and hide the formal world We might try to place formalized structures "out of sight" and not bother "regular people" with them. In fact, this amounts to what has since long been practiced in IS development and use. In an idealized and naive version of this idea, "intelligent machines" should "simply" perform the translation (formalization) for us. A more down-to-earth view leaves real formalizations to a relatively small group of experts and generally makes do with very light formalisms in IS development, accepting the limitations and problems this entails.

It may not come as a surprise that we too work towards a viable version of the third option, and integrate it in the IS development process. What this boils down to is encouragement, where relevant and productive, of an optimally rational way of system development without use of actual formal languages where possible, complemented with support for the problematic process of formalization where required. It is the latter we focus on in the rest of this paper.

3.2 The danger of using the translation metaphor in formalization

An important misconception underlying many naive approaches to bridging the formal-informal divide in IS development stems from a limited understanding of the nature and functionality of formal languages and formalizations. This misconception is reflected in the term "translation" as in "translation from informal to formal language". Still assuming that formal and informal descriptions are fundamentally different because of the nature and use of their respective underlying semantics, both systems of semantics can perhaps be respected and *partially linked*, in a non-deterministic and *pragmatic* fashion, without imposing one on the other in any absolute sense. This is essentially different from "rephrasing an expression from one language in another, retaining the original meaning" –which is what most people associate with "translation".

Formalization has its own, specialized functional context, which is rather different from that of natural language use. We propose to study how those two functionalities can be best combined in IS development (and, ultimately, use), making optimal use of the strong points of either functionality. However, above all we propose (in a LAP-like fashion) to open the black box of the act of "formalization" (or, perhaps more correctly, "formal modelling") and study it in terms of *behaviour*. The LAP is our paradigm of choice for this approach. Formalization, viewed as a conversation, can thus be linked with and brought closer to other communicational activities, without the need for informal-formal "translation" in the narrower, misguided sense.

In addition, from a purely practical point of view, it would be highly beneficial to system development in general if formalization came within the reach of non-specialists, or at least became a less specialistic task. Also, formalization might be made more efficient and effective, and the quality of formalizations (both in terms of validity and correctness) "better".

4 Our view of the formal modelling process

Under our "behavioural" approach we aspire to explicitly express "modelling behaviour" as part of (communicative) behaviour in general. Therefore, it includes modelling strategies just as it includes conversation strategies; in fact, it views modelling strategies *as* conversation strategies. Our current analysis is rooted in *domain modelling*: the ORM "way of working".

In line with section 2.2, under our analysis a model is constructed with a more or less specific *knowledge goal* in mind. The goal determines the required properties of the model: its topic, explicitness, and level of sharing. Strictly speaking, a model is a representation of a *mental model* belonging to a particular *viewer* of a domain, instead of the domain itself. The domain expert perceives the domain, and constructs (conceives) a mental model of that domain. Therefore, the information provided by a domain expert is always subjective and reflects only the structure of the domain *as seen by the domain expert* [5].

4.1 Modelling as a dialogue

We view the modelling process as a goal-driven *dialogue* between a number of participants. Questions and answers are exchanged. Each participant has her particular view (conception) of the domain. The only way the participants can achieve their respective modelling goals is to communicate with each other, and remember and build on what has been discussed. An explicit way to do this is to keep "modelling minutes" that are agreed on by the participants [34].

In early approaches, the modelling process was portrayed as an asymmetric flow of information from domain expert to system analyst. More recently, the role of the analyst has also been considered, with a focus on the interaction of the participants in the modelling process [14]: the analyst knows the demands posed by the formalism used, and how to fulfil them. Under such an approach, the modelling dialogue is a *symmetric* exchange of knowledge in which:

- the domain expert transfers *expert knowledge* to the system analyst
- the system analyst transfers *analysis knowledge* to the domain expert

Both domain expert and system analyst are experts in their own field, contributing knowledge to the common model.

In general, a modelling process then is a dialogue involving a set $\{A_1, \ldots, A_n\}$ of participants (see figure 2). Each actor has an internal model $\{M_1, \ldots, M_n\}$. For human actors, this is a mental model. The modelling process may be typically seen as a collaborative activity to develop a common and agreed model. To achieve this common model, the participants perform a goal-driven dialogue. Under our approach, formal modelling is captured by recording restricted aspects

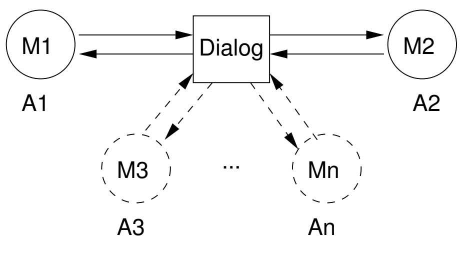

Figure 2: Symmetric view of the modelling dialogue, with two or more participating actors.

of communication between participants.

4.2 Using controlled language in the modelling dialogue

Formal and informal language may be hard to fully reconcile, but a classic meeting point between natural and formal language lies in similarities between the basics of their grammar and meaning, in particular in predicates and predication. It has since long been recognized that when we use simple, elementary sentences in natural language, we can relatively easily bridge the gap between formal and informal [13, 18], even if the bridge can only bear very light traffic. Such simple, elementary language can be described by a relatively simple grammar and can yet be realistically used in a modelling conversation. We refer to it as *controlled language*. Our notion of controlled language is related to that of *simplified English*; see [11, 1].

Importantly, the information covered by a single statement in controled language will be quite limited. A formalization requires a fairly large number of interrelated, well validated statements to reach a degree of explicitness and agreement adequate for the target formalization.

4.3 Explicit statements with various implicit interpretations

We presume a participant not only to be able to represent (parts of) her conception of the universe, but also to be able to represent (relevant parts of) the language/concepts they use in producing their conception of the universe. Taking a communicative perspective on information modelling, the goal of the modelling process can thus be described as: trying to reach a state where all participants agree that they have some degree of common understanding. The participants try to reach consensus about a group semantics of the model that is based on, and is compatible with, the semantic systems and the interpretations of all group members.

Participants will be convinced this goal has been achieved if they have validated their assumptions to contentment of everyone involved. For example, a system analyst will be convinced that the derived model is complete if the model has been validated against the real situation. In our view, this means that the domain expert, harbouring the semantics of her conception of the universe, has positively responded to the description of the model provided by the system analyst, which may be rooted in a formal language. Various semantic systems and interpretations come into play; the shared, controlled language (which may in part cooperatively constructed as part of the modelling process) performs an intermediary function.

The goal of the dialogue can thus be seen as the construction of (1) a grammar for representations that are acceptable to all participants, and (2) semantic interpretation(s) in terms of some model(s). The grammar produced in the interaction is a generative device; the grammar is correct when all possible (valid) sample sentences can be generated. In addition, the grammar can be used as a parsing device. From the point of view of the system analyst, the target model is restricted by the (formal) semantics of the modelling technique used. From the point of view of the domain expert, validation of the model may be seen as assigning meaning (interpretation) to the representations generated by the system analyst. A more symmetric way of putting this is that for each party (a), the other party (b) agrees with the controlled language statements provided by party (a).

4.4 Sketch of the dialogue structure

Because of editorial limitations on this paper, we cannot discuss modelling strategies or the structure of modelling dialogues at any lenghth here. We can only say that we distinguish six straightforward dialogue actions:

Propose(a, s)	Actor a proposes statement s. It does not become part of the common model until every actor accepts it.
Withdraw(a, s)	Actor a withdraws statement s. Withdraw is the opposite of a propose.
Accept(a, s)	Actor a accepts statement s as a valid statement; it may eventually become part of its internal model M_a. A statement can only be accepted after it is proposed.
Reject(a, s)	Actor a rejects statement s, because a finds s unacceptable even for further consideration. Reject is the counterpart of accept.
Ask(a, q)	Actor a asks question q, to be answered by some actor. Queries can be withdrawn or answered.
Answer(a, q, s)	Actor a answers question q with statement s; an answer functions as a special Propose.

As a mere illustration of the principle, consider this very simple example, that reflects a core aspect of domain modelling: application of what we might call the "delayed specificity" strategy. A more elaborate, formalized analysis of this strategy can be found in [6].

The ORM formalism requires relational structures to be *specific*: enough information about entities must be provided, including of what *type* an entity is. Compare the first, "specific" sentence with the second, non-specific one:

```
1) person with name John lives in city with name Nijmegen
2) John lives in Nijmegen
```

The first sentence may be preferable from the point of view of the formalism, but it is a highly unlikely statement to be produced in one go by a domain expert. If a 1:1 translation from an NL statement to the formalism were required, sentence 2) would be unacceptable. However, another way out is a guided dialogue which leads to step-wise approach to gathering of the information needed:

- Detect non-specific statement (the grammar/parser can be of help here)
- Solve non-specificity by asking the domain expert for missing information
- Alternatively, solve non-specificity by assuming the missing information. This requires the sub skill of creating plausible information; suggestions (to be validated!) could be provided by means of a generic lexicon/ontology.

Note that the approach above relates to some basic strategies for uncertainty reduction mentioned in section 2.4. A complete example diogue may then be as follows (keeping to the traditional roles in domain modelling, "DE" is domain expert, "SA" is system analyst):

```
    propose(DE    John lives in Nijmegen)
        ask(SA    What kind of thing is John?
    propose(DE    John is a person)
ask/propose(SA    Do we distinguish John from other persons by means of his name?
     accept(DE,SA Yes [we distinguish John from other persons by means of his name])
ask/propose(SA    Do you agree that John is a person with name John?)
     accept(DE,SA Yes [John is a person with name John])
```

For each "accept", both DE and SA are registered as agreeing with the statement. In two cases, the action of asking and proposing is collapsed in one statement. As a result of this small sub-dialogue, the statement "John is a person with name John" is added to the seperate set of "agreed statements", on the basis of which a complete formalization can later be compiled. In addition, by means of the "defoleating strategy" (typical for "fact based modelling"), a generalization can be derived, suggested, and confirmed:

```
ask/propose(SA    Do persons generally have names?)
     accept(DE,SA Yes [persons generally have names])
```

5 Grounding formal models in the social world

Let us now briefly consider how the modelling dialogues sketched in the previous section relate to the grounding of models. In the representationalist/positivist tradition, formal models are generally supposed to be decontextualized (and therefore unambiguous). This is a disputable view at best. If in addition, we view system development as a conversation, then not only are models undeniably produced and used in context and with some goal, but the specifics of a context are known and can be recorded alongside the models. Assuming that a model is built up through the gathering of statements, and that it is somehow recorded who puts forward each statement and who agrees with it (or even commits to it), we can place formal models in a *social context* –which opens up all sorts of possibilities, both practically and theoretically.

5.1 Validity of models

The quality of formal models is generally considered to be determined by two main fators: their *correctness* (checked by means of verification) and their *validity* (checked by means of validation). However, whereas the literature on formal verification is absolutely vast, from a representationalist/positive perspective, not much can be said about, nor done about, validity. Most approaches boil down to "letting a participant read a description (possibly translated to NL), and letting her decide whether it is valid". Our dialogue-based, contextualized approach enables us in principle to deal with validation in a nuanced, varied, goal-driven, and controllable way. It is arguably an attempt to achieve for validation what representation-based grammar checking and proofing techniques have achieved for verification. Though no "absolute" validation can be achieved because of the fundamentally subjective nature of the validation process, detailed and contextualized validity of a model can be systematically checked and re-checked. What is more, a model is typically not validated as a whole, but statement by statement, using whatever strategie are required to reach the required degree of agreement and explicitness for the target formalism.

5.2 Some basic aspects of validity

The LAP literature provides many interesting links to validity issues in modelling. Most prominently, the *validity claims* (Power, Truth, Sincerity, Justice, Comprehensibility –or some comparable constellation) as distinguished by Searle and Habermas (among others; see [31, p51-]) apply to modelling statements just as they do to other sort of statements, and can be used to create and select specific strategies for specific sorts of validation. Note that when validation/agreement on the semantics of statements is concerned, the claim to comprehensibility can be challenged and discussed. For more on this, see [19, p60-70]. In addition, the stages/levels of knowledge sharing listed in section 2.1 can be included. Within LAP literature, a relation presents itself between those levels and, for example, illocutions as distinguished in [31] or even the DEMO method. In fact, van Reijswoud's analysis of business conversations as reflected in his Transaction Process Model (in particular, its discussion layer) provides an interesting framework for further understanding and even guiding of modelling dialogues.

Importantly, a behavioural approach working towards validity (and possibly even to formal correctness) of a model does not necessarily "look back" on a completed (part of) a model, it strives to guide the modelling process and along the way creates a valid/correct model. This means the process itself is, at many levels and in many different ways, geared towards "quality" –as defined in context. It also suggests that that processes for valid/correct modelling can more easily be integrated in dynamic approaches to system development. In view of the whole IS development process as an integrated conversation, the bottom line is that not just some insular (formal) models (being *representations*) are developed on the basis of 'rational" strategies, but that the whole process can be shaped as an optimally "rational process" [34, 37].

5.3 Including design/modelling rationales

An aspect we have not considered in any depth yet, but that seems promising enough, is that of recording *rationales* (argumentation) behind modelling decisions. In IS development, these often are design decisions, and keeping track of them is a long cherrished wish in design and system development. Indeed it seems quite possible to add a particular set of strategies to modelling dialogues that elicit and record such rationales, and whether there is agreement or even disagreement about them, and what alternatives were considered, and who took the final decision. Within LAP literature, an inspiring framework that could well be used for structuring and categorizing rationales (and, possibly, the semantics of modelling dialogues in general, though we have to refrain from discussion of that issue here) is that of Weigand and de Moor [35], who provide a formal framework (argumentation semantics) in context of communicative action. This framework is partly based on Conklin's Issue Based Information Systems (IBIS) paradigm [7].

6 Towards an environment for studying and supporting formal modelling

We have presented our over-all view on information system development and modelling as a multi-faceted conversation within the development community, grounded in knowledge sharing, agreement, and commitment. Thus, we take a Language Action Perspective on the IS development process, viewing representations produced in the process as texts that reflect ongoing, social interaction between participants. If such texts include *formalizations*, then the functionality of both formal and informal texts and languages should be taken into account, and somehow combined. We have proposed a dynamic way of achieving this, based on dialogues, and provided a simple example of how we might analyse an aspect of domain modelling as a focused information gathering conversation.

We are now in the process of deepening our analysis and formalizing part of it. This has already provided us with a preliminary conceptual framework for describing modelling *dialogues* and *strategies.* We are also making some initial progress in applying our framework to various flavours of modelling, including information modelling, pre-negotiation ontological modelling, architecture modelling, and requirements modelling. All this will be reported on before long.

One of our core goals is to extensively study modelling strategies for various sub-fields of information system development (indeed, for such a study we need a sound conceptual framework). The conceptual framework, but in particular the strategies, will have to be studied and validated *empirically.* In order to make this possible, we are planning to create an experimental *environment for modelling dialogues.* By studying how participants in the modelling process interact, we first hope to discover and confirm how modelling dialogues work, and in particular what strategies are used in them. Once we have a sufficient grasp of the strategies involved, we intend to add some functionality to the environment enabling it to actively *guide* the dialogues (in view of particular modeling goals). Eventually, it is not unthinkable for our environment for studying and guiding modelling dialogues to evolve into an environment for the support of modelling conversations in practice. This would possibly amount to a hybrid system, part Groupware, part CASE tool, part Truth Maintenance System. However, whether we ever get there remains to be seen, and it is not currently our target to support modelling dialogues in an industrial setting.

References

[1] AECMA - The European Association of Aerospace Industries. *AECMA Simplified English – A guide for the preparation of aircraft maintenance documentation in the international maintenance language*, January 2001. Issue 1, Revision 2.

[2] John R. Anderson. *Cognitive psychology and its implications.* W.H. Freeman and Company, San Francisco, California, USA, 1990.

[3] A.I. Bleeker, H.A. (Erik) Proper, and S.J.B.A. Hoppenbrouwers. The Role of Concept Management in System Development – A practical and a theoretical perspective. In J. Grabis, A. Persson, and J. Stirna, editors, *Forum proceedings of the 16th Conference on Advanced Information Systems 2004 (CAiSE 2004), Riga, Latvia, EU*, pages 73–82, Riga, Latvia, EU, June 2004. Faculty of Computer Science and Information Technology.

[4] P. van Bommel, A.H.M. ter Hofstede, and Th.P. van der Weide. Semantics and verification of object–role models. *Information Systems*, 16(5):471–495, October 1991.

[5] S. Bosman and Th.P. van der Weide. A case for incorporating vague representations in formal information modeling. In *Conferentie Informatiewetenschap 2003*, 2003.

[6] S. Bosman and Th.P. van der Weide. Towards formalization of the information modeling dialog. Technical Report ICIS–R05012, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands, EU, 2004.

[7] J. Conklin. *The IBIS Manual: a short course in IBIS methodology.* Touchstone, 2003.

[8] C.F. Derksen, P.J.M. Frederiks, and Th.P. van der Weide. Paraphrasing as a Technique to Support Object–Oriented Analysis. In R.P. van der Riet, J.F.M. Burg, and A.J. van der Vos, editors, *Proceedings of the Second Workshop on Applications of Natural Language to Databases (NLDB'96)*, pages 28–39, June 1996.

[9] J.L.G. Dietz and T.A. Halpin. Using DEMO and ORM in Concert: A Case Study. In *Advanced Topics in Database Research, Vol. 3*, pages 218–236. 2004.

[10] D.W. Embley, B.D. Kurtz, and S.N. Woodfield. *Object–Oriented Systems Analysis – A model–driven approach.* Yourdon Press, New York, New York, USA, 1992.

[11] G. Farrington. *An Overview of the International Aerospace Language.* 1996.

[12] M. Franckson and T.F. Verhoef, editors. *Managing Risks and Planning Deliveries.* Information Services Procurement Library. ten Hagen & Stam, Den Haag, The Netherlands, EU, 1999.

[13] P.J.M. Frederiks. *Object–Oriented Modeling based on Information Grammars.* PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU, 1997.

[14] P.J.M. Frederiks and Th.P. van der Weide. Information Modeling: the process and the required competencies of its participants. In F. Meziane and E. M'etais, editors, *9th International Conference on Applications of Natural Language to Information Systems (NLDB 2004), Manchester, United Kingdom, EU*, volume 3136 of *Lecture Notes in Computer Science*, pages 123–134, Berlin, Germany, EU, 2004. Springer.

[15] G. Goldkuhl and P.J. Agerfalk. Action Within Information Systems. Technical Report 1998–2, Jonkoping International Business School, 1998. Accepted to the Fourth International Wokshop on Requirements Engineering: Foundations of Software Quality (REFSQ'98), June 8–9 1998, Pisa, Italy.

[16] T.A. Halpin. *A logical analysis of information systems: static aspects of the data–oriented perspective*. PhD thesis, University of Queensland, Brisbane, Queensland, Australia, 1989.

[17] T.A. Halpin. *Information Modeling and Relational Databases, From Conceptual Analysis to Logical Design*. Morgan Kaufmann, San Mateo, California, USA, 2001.

[18] J.J.A.C. Hoppenbrouwers, B. van der Vos, and S.J.B.A. Hoppenbrouwers. NL Structures and Conceptual Modelling: Grammalizing for KISS. *Data & Knowledge Engineering*, 23(1):79–92, 1997.

[19] S.J.B.A. Hoppenbrouwers. *Freezing Language; Conceptualisation processes in ICT supported organisations*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU, 2003.

[20] S.J.B.A. Hoppenbrouwers. Formele Informatiekunde: overbrugging tussen de informele en de formele wereld. Technical report, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands, EU, 2005. In Dutch.

[21] S.J.B.A. Hoppenbrouwers and H.A. (Erik) Proper. A Communicative Perspective on Second Order Information Systems. In G.E. Lasker, editor, *Proceedings of the 16th International Conference on System Research, Informatics and Cybernetics, Baden–Baden, Germany*. IIAS, 2004.

[22] S.J.B.A. Hoppenbrouwers, H.A. (Erik) Proper, and Th.P. van der Weide. Understanding the Requirements on Modelling Techniques. In O. Pastor and J. Falcao e Cunha, editors, *Proceedings of the 17th Conference on Advanced Information Systems 2005 (CAiSE 2005), Porto, Portugal, EU, Berlin, Germany, EU*, pages 262–276, Berlin, Germany, EU, 2005. Springer.

[23] S.J.B.A. Hoppenbrouwers and H. Weigand. Meta–communication in the Language Action Perspective. In *Proceedings of the Fourth International Workshop on the Language Action Perspective on Communication Modelling (LAP 2000)*, Aachener Informatik-Berichte, Aachen, Germany, EU, 2000. RWTH Aachen.

[24] R. Lipshitz and O Strauss. Coping with uncertainty: a naturalistic decision–making analysis. *Organizational Behaviour and Human Decision Processes*, 2(69):152–154, 1997.

[25] J. Mylopoulos. Techniques and Languages for the Description of Information Systems. In P. Bernus, K. Mertins, and G. Schmidt, editors, *Handbook on Architectures of Information Systems, Berlin, Germany, EU*. Springer, Berlin, Germany, EU, international handbooks on information systems edition, 1998.

[26] G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design: a fact oriented approach*. Prentice–Hall, Englewood Cliffs, New Jersey, USA, 1989.

[27] I. Nonaka and H. Takeuchi. The Knowledge–Creating Company. *Harvard Business Review*, (November–D):97–130, 1991.

[28] C.S. Peirce. *Volumes I and II – Principles of Philosophy and Elements of Logic*. Collected Papers of C.S. Peirce. Harvard University Press, Boston, Massachusetts, USA, 1969.

[29] H.A. (Erik) Proper and S.J.B.A. Hoppenbrouwers. Concept Evolution in Information System Evolution. In J. Gravis, A. Persson, and J. Stirna, editors, *Forum proceedings of the 16th Conference on Advanced Information Systems 2004 (CAiSE 2004), Riga, Latvia, EU, Riga, Latvia, EU*, pages 63–72, Riga, Latvia, EU, June 2004. Faculty of Computer Science and Information Technology.

[30] H.M. Regan, B.K. Hope, and S Ferson. Analysis and portrayal of uncertainty in a food web exposure model. *Human and Ecological Risk Assessment*, 8(7):1757–1777, 2002.

[31] V.E. van Reijswoud. *The Structure of Business Communication: Theory, Model and Application.* PhD thesis, Delft University of Technology, Delft, The Netherlands, EU, 1996.

[32] V.E. van Reijswoud and J.G.L. Dietz. *DEMO Modelling Handbook*, volume 1. Delft University of Technology, Delft, The Netherlands, EU, 2nd edition, 1999.

[33] V.E. van Reijswoud, J.B.F Mulder, and J.L.G. Dietz. Commucation Action Based Business Process and Information Modelling with DEMO. *The Information Systems Journal*, 9(2):117–138, 1999.

[34] G.E. Veldhuijzen van Zanten, S.J.B.A. Hoppenbrouwers, and H.A. (Erik) Proper. System Development as a Rational Communicative Process. *Journal of Systemics, Cybernetics and Informatics*, 2(4), 2004.

[35] H. Weigand and A. de Moor. Argumentation semantics of communicative action. In M. Aakhus and M. Lind, editors, *Proceedings of the 9th Interntional Working Conference on the Language–Action Perspective in Communication Modelling (LAP 2004)*. Rutgers University, New Brunswik, USA, June 2004.

[36] T. Winograd and F. Flores. *Understanding Computers and Cognition.* Ablex Publishing, Norwood, New Jersey, USA, 1986.

[37] H. Wupper and A.H. Mader. System Design as a Creative Mathematical Activity. Technical Report CSI–R9919, Institute for Computing and Information Science, Radboud University Nijmegen, 1999.