Faculty of Arts and Science

Thesis No FiF-a 30

# Pragmatization of Information Systems
## – A Theoretical and Methodological Outline

by

**Pär J. Ågerfalk**

Submitted to the Faculty of Arts and Science at Linköpings universitet in partial fulfilment of the requirements for the degree of Licentiate of Philosophy

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköping 1999

# Pragmatization of Information Systems
## – A Theoretical and Methodological Outline

by

Pär J. Ågerfalk

Sep 1999
ISBN 91-7219-610-6
Faculty of Arts and Science FiF-a 30
ISSN 1401-4637

ABSTRACT

Pragmatization of information systems means to climb the semiotic ladder from the syntactic and semantic levels, and take the pragmatic use of language as a starting point for understanding information and information systems as social phenomena. This thesis concerns theory and method for conceiving and developing information systems based on such an understanding. We propose an action theory of information systems, which is founded in the information system methods tradition, based on a language action perspective, and influenced by current and important trends within the fields of information systems, software engineering, and human-computer interaction. As a methodological consequence of the theory, a re-design of an existing systems engineering method has been performed as a combination of theoretical work and action research. Both the theory and the method, as well as the empirical work, are presented and elaborated upon in this thesis.

*This work has been financially supported by Örebro University and The Swedish National Board for Industrial and Technical Development (NUTEK).*

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

# Foreword

Information systems development is a discipline within the Faculty of Arts and Science at Linköping University. Information systems development is a discipline studying human work with developing and changing computer-based information systems in organisational settings. It includes theories, strategies, models, methods, co-working principles and tools concerning information systems development. Different development/change situations can be studied as planning, analysis, specification, design, implementation, deployment, evaluation, maintenance and redesign of information systems and its interplay with other forms of business development. The discipline also includes the study of prerequisites for and results from information systems development, as e.g. studies of usage and consequences of information systems.

This work, Pragmatization of information systems – a theoretical and methodological outline, is written by Pär J. Ågerfalk at Örebro University. He is also a member of research group VITS and the Centre for studies on humans, technology and organization (CMTO) at Linköping University. He presents this work as his licentiate thesis in information systems development, Department of Computer and Information Science, Linköping university.

Linköping September 1999

Göran Goldkuhl
Professor

# Doctoral Dissertations in Information Systems Development

1. Karin Axelsson (1998): Metodisk systemstrukturering – att skapa samstämmighet mellan informationssystemarkitektur och verksamhet

2. Stefan Cronholm (1998): Metodverktyg och användbarhet – en studie av datorstödd metodbaserad systemutveckling

3. Anders Avdic (1999): Användare och utvecklare – om anveckling med kalkylprogram

# Licentiate Theses in Information Systems Development

1. Owen Eriksson (1994): Informationssystem med verksamhetskvalitet – utvärdering baserat på ett verksamhetsinriktat och samskapande synsätt

2. Karin Pettersson (1994): Informationssystemstrukturering, ansvarsfördelning och användarinflytande – en komparativ studie med utgångspunkt i två informationssystem-strategier

3. Stefan Cronholm (1994): Varför CASE-verktyg i systemutveckling? – En motiv- och konsekvensstudie avseende arbetssätt och arbetsformer

4. Anders Avdic (1995): Arbetsintegrerad systemutveckling med kalkylprogram

5. Dan Fristedt (1995): Metoder i användning – mot förbättring av systemutveckling genom situationell metodkunskap och metodanalys

6. Malin Bergvall (1995): Systemförvaltning i praktiken – en kvalitativ studie avseende centrala begrepp, aktiviteter och ansvarsroller

7. Mikael Lind (1996): Affärsprocessinriktad förändringsanalys – utveckling och tillämpning av synsätt och metod

8. Carita Åbom (1997): Videomötesteknik i olika affärssituationer – möjligheter och hinder

9. Tommy Wedlund (1997): Att skapa en företagsanpassad systemutvecklingsmodell – genom rekonstruktion, värdering och vidareutveckling i T50-bolag inom ABB

10. Boris Karlsson (1997): Metodanalys för förståelse och utveckling av system-utvecklingsverksamhet – analys och värdering av systemutvecklingsmodeller och dess användning

11. Ulf Melin (1998): Informationssystem vid ökad affärs- och processorientering – egenskaper, strategier och utveckling

12. Marie-Therese Christiansson (1998): Inter-organisatorisk verksamhetsutveckling – metoder som stöd vid utveckling av partnerskap och informationssystem

13. Fredrik Öberg (1998): Object-oriented frameworks – a new strategy for CASE tool development

14. Ulf Seigerroth (1998): Integration av förändringsmetoder – en modell för välgrundad metodintegration

15. Bengt EW Andersson (1999): Samverkande informationssystem mellan aktörer i offentliga åtaganden – en teori om aktörsarenor i samverkan om utbyte av information

16. Pär J. Ågerfalk (1999): Pragmatization of information systems – a theoretical and methodological outline

# Acknowledgements

I am indebted to the many people who have made this thesis possible. Among them, I would specifically like to thank:

My supervisors Prof. Göran Goldkuhl and Dr. Stefan Cronholm for endless support and encouragement. Your involvement in the work presented in this thesis, both as academic advisors and fellow researchers, has been invaluable for its success. Thank you.

Owen Eriksson and Pär Fahlén who have both contributed, in several different ways, to the work presented in this thesis.

Kenneth Åhlgren for intricate discussions about systems engineering methods and methodology in general, and a host of specific research topics during the years.

Karin Hedström for commenting on the final manuscript, and for everything else that I owe you thanks for.

Anders Avdic. Without your engagement I would probably never have got involved in graduate studies in the first place.

All the colleges in the Research group VITS who have suffered through the many seminar presentations and given valuable feedback throughout the entire research process.

Tom Lacy for the fast proof-reading.

All the anonymous referees of the papers upon which this thesis is based, and all the active participants at the conferences where I have presented them. Your questions, comments and suggestions have been most valuable. I would specifically like to mention Mareike Schoop, Andreas Opdahl and Neil Maiden.

Torsten Gille, Lasse Gustafsson, Fredrik Gustafsson, Stefan Stråle and Magnus Fröberg for making the empirical work pleasant and valuable.

This work has been financially supported by Örebro University and The Swedish National Board for Industrial and Technical Development (NUTEK) through the MCC-project (Management system for change-over efficiency in complex business and production processes).

Örebro September 1999

Pär J. Ågerfalk

# Contents

# Chapter 1

# Introduction to the Thesis

*Mechanical engineering is like looking for a black cat in a lighted room. Chemical engineering is like looking for a black cat in a dark room. Software engineering is like looking for a black cat in a dark room in which there is no cat. Systems engineering is like looking for a black cat in a dark room where there is no cat and someone's yelling –'I got it'!*

Internet saying

This chapter starts by presenting the background of the thesis work in order to establish its legitimacy. Subsequently, the aim of the thesis is presented and motivated. Finally, demarcations are stated and reading instructions, including a thesis outline, is presented.

## 1.1 Background and Motivation

There are many different methods and approaches to support systems development. Even though they all aim at the same thing – building 'good' information systems – they often differ in several ways. Sometimes the differences are subtle (for example different object-oriented methods that share the same concepts and notations) but sometimes the underlying values and perspectives are certainly discriminating, i.e. their respective notions of what constitutes a 'good information system'. We believe that foundational values and perspectives have vital impacts on the development of information systems as well as on the developed information systems *per se*, which this thesis will try to show.

This thesis concerns re-design of a systems engineering method[1] referred to as VIBA/SIMM[2] (or VIBA for short). The method was first developed by Göran Goldkuhl and published in a Swedish book: Verksamhetsutveckla datasystem (Goldkuhl, 1993). The method was well received within academia and has been used extensively in Informatics courses at several Swedish universities ever since. One of the most in-

[1] More specifically a method for specifying requirements on information systems and their business environment, i.e. requirements engineering.

[2] Versatile Information and Business requirements Analysis according to the Situation adaptable work and Information systems Modelling Method.

teresting benefits of VIBA was its foundation in human infology: a theoretical perspective that emphasizes the linguistic and social character of information systems (Goldkuhl & Lyytinen, 1982; Goldkuhl, 1993). VIBA has also been used in industry to some extent.

The main problem targeted by VIBA was that information systems often seem to disharmonize with the business. Users feel that information systems are hard to understand and regard them as belonging to the IT department instead of their own workplace, i.e. they feel alienation and helplessness (cf. Computer Sweden, 1999, p. 6). The customer value is often neglected and information systems are often used as an excuse for poor customer service instead of being regarded as service enablers. To solve these problems VIBA promotes a business-oriented approach ('verksamhetsinriktad' in Swedish). That is, the business and the business' actors are in focus when developing systems. Systems development is considered a special kind of business development and the aim is to build information systems that:

- are usable,
- are integrated and congruent with the business,
- are understandable and predictable,
- have visible actors, and
- enable and support good working conditions.

During the last few years, the method has been criticized and challenged in several ways. The sources of criticism represent both theoretical work, as seen in several evaluations made by students during systems engineering classes and examination projects, e.g. Överström (1994), Vallgren (1992); and more practical assessments, as in for example the fact that different instructors have modified the method to better fit their particular needs, e.g. Nilsson *et al.*, (1997).

One criticism concerns the complex handling of user interaction modelling and the lack of (or at least unclear) support for experimental and iterative development. It has also been challenged as not making use of the benefits of object orientation, which has gained more and more interest (and success) during the last few years. Another criticism, which can be directed towards many systems engineering methods, is unclear statements on the applicability of the method, i.e. the method context (see below for discussion of method context). With this in mind we started working on ways to re-design VIBA. We wanted to make the method even more explicit about its foundations in a further developed action theory for information systems and to take advantage of recent developments and (successful) trends within the areas of information systems, software engineering, and human-computer interaction.

The reason for taking VIBA/SIMM as the point of departure for the work was VIBA's previously discussed foundation in human infology, which matched our intentions of emphasizing the action character of information and information systems (cf. sections 1.2 and 1.3), i.e. that to conceive information systems as action is a key factor in the success of systems engineering. Of course, some other method based on similar theoretical foundations, such as, for example, the Commodious method (Holm, 1996), would have sufficed. The reasons for favouring VIBA, in this respect, were simply the availability of the method and the method's rationale; the creator of the original VIBA/SIMM was actually part of the research project (cf. section 1.10), which, of course, influenced the choice.

## 1.2 Aim of Research

The aim of this thesis is to present and elaborate on the design of a method for information systems engineering based on an Action Theory of Information Systems (ATIS).

By 'present and elaborate' we refer to an argumentative presentation of a method that has been grounded both theoretically (i.e. it is consistent with the Action Theory of Information Systems) and practically (i.e. it has been tested empirically).

What, then, do we mean by method and theory? These topics are examined more carefully in chapter 2. But to give an indication of what this thesis is about we can summarize our view of method as being a combination of five parts:

1. Modelling concepts and notations used to conceptualize and document the problem domain.
2. Contexts in which the method is applicable.
3. Values that imply what problems to solve and hove to solve them. For example, what is a good IS and a good development process?
4. Development models under which methods are supposed to be used.
5. Techniques to handle complexity of the problem domain, the solutions and the problem solving process.

A theory of information systems is, in short, a conceptualization of interesting properties of information systems and information systems development forming a coherent basis for the five parts of a method.

The method design has been based on two assumptions. The first assumption is that *to conceive information systems as action* is a key fac-

tor in the success of future development efforts. By viewing information systems and information systems development as mainly social phenomena, in the sense of communicative action, we begin to understand the nature and problems of systems engineering and are hopefully able to cope with them in a better way. The title of this thesis is 'Pragmatization of information systems'. Pragmatization of information systems means to climb the semiotic ladder (cf. Falkenberg *et al.*, 1998) from the syntactic and semantic levels, and take the pragmatic use of language as a starting point for understanding information and information systems.

The second assumption is that *information systems are subject to change at an ever-increasing speed*. One implication is that specified requirements are valid for shorter and shorter periods in order to keep information systems up-to-date with rapidly changing business environments.

The work has also been deliberately influenced by five recent and, as we believe, important knowledge domains of information systems. These are:

- business process orientation,
- object modelling (object orientation),
- usability,
- rapid (application) development (including prototyping) and
- software requirements management.

In sections 1.3 to 1.5 we will discuss these assumptions and knowledge domains with the purpose of explaining what they stand for and why these were chosen.

## 1.3 An Action Theory of Information Systems

One early, and important, step towards a theory of information systems can be dated back to 1966 and the book 'Theoretical Analysis of Information Systems' (THAIS) by Prof. Börje Langefors (1966). Ever since it was first published THAIS has played a major role in the conceptualization of, and in the way people perceive and talk about, information systems, at least in Scandinavia (cf. Iivari & Lyytinen, 1998). THAIS treats information and information systems as special cases of general systems theory. One basic assumption is that parts that are related to each other in some way, and can be described in a formal (mathematical) way, constitute a system. Another important part of THAIS is the fundamental principle of systems work that states how to cope with the complexity of a system: it has to be recursively broken down in smaller parts until each part is small enough to overview. Each part can then be analysed with

respect to its internal and external properties. At a generic level, the smallest parts of information are assumed to be elementary messages (e-messages). An e-message consists of three parts: object (the thing referenced by the e-message); property (used to predicate something about that thing); and a point in time or time interval during which the e-message is valid. The *kind* of information represented by an e-message is referred to as an e-concept, hence '*any e-message will be regarded as an instance of an e-concept*' (Langefors, 1966, p. 296). E-messages are used to build up information, which is thought of as interpreted data. This is formalized in the infological equation, which states that the information content I = i(D, S, t) is the result of an interpretation process i applied to the data D during the time interval t, given the pre-knowledge S. Information is thus defined as interpreted data that add knowledge to the interpreter. If it does not add knowledge, the information is considered redundant.

This way of understanding information and information systems according to the THAIS-perspective is still prevalent in the information and software systems communities. For example, the OPR(t)-model (Sundgren, 1992) is founded on the concept of the e-message and the compositional way of working with system analysis (e.g. Yourdon, 1989) is in parallel with the fundamental principle of systems work. Another example is object orientation, which builds explicitly on the notion of a system as having internal and external properties (c.f. Langefors, 1995).

Criticisms of the THAIS-perspective were brought up already in the 70's. One source of criticism was the *human infological* perspective on information and information systems. Human infology, which treated information systems as mainly social phenomena[3], led to new ways of understanding the social and organizational implications of information systems development and usage (Goldkuhl, 1980). With such a perspective information systems are considered mainly as formalizations of the professional language used within a business (Lyytinen, 1983). This language approach to information has evolved (and converged with other theories) into what is now called the *Language Action Perspective* (LAP). LAP is a theoretical framework for understanding communication and information usage based on speech act and communicative action theories, i.e. sociological, linguistic and language-philosophical theories (Dignum *et al.*, 1996; Dignum & Dietz, 1997; Goldkuhl *et al.*, 1998; 1999). With such a perspective information systems are not restricted to

[3] According to Goldkuhl & Lyytinen (1982) are information systems '*social systems only technically implemented*'.

being 'containers of facts' or 'tools for information transfer' but rather vehicles for communication among people and organizations.

A key difference between the THAIS perspective's and the Language Action Perspective's views of information systems is their respective notions of smallest informational content (i.e. their axioms regarding information). With a language action perspective, messages carry more then mere facts (as is the case with e-messages). Messages also carry the communicating actor's intention or 'action mode', i.e. what speaker does in relation to listener (e.g. Austin, 1962). People *do* things while communicating; for example making promises, requests, proposals, *et cetera*. Among other things, this leads to an extension of the concept of e-message to what is called 'action elementary message' (cf. chapter 4).

## 1.4 The Increasing Speed of Business Change

The customer-oriented businesses of today are extremely susceptible to changes in the environment (e.g. Keen, 1997). To satisfy disperse customer needs, businesses have to be both flexible and adaptable (*ibid.*). In order to manage such organizations, information systems must be able to develop 'fast' and be 'easy' to adapt to changes. DuPont (1989)[4] makes the following comment (as quoted by Martin, 1991):

> '*It is speed that is the deciding factor in most competitive situations–in identifying a new end use, in getting products to the market, in implementing new services, in resolving problems that reduce waste, in responding to fashion trends, in designing better processes, in making effective organizational changes, in controlling inventories and distribution, and in scenario simulations to optimize the machine/product mix. ...A top criterion for I.S. is that information systems should never interfere with the business' ability to seize an opportunity*'.

Paradoxically, it seems that in order to cope with fast development of flexible information systems developers need to be even more careful during early phases and to adopt structured engineering methods and practices to support development and evolution (Martin, 1991).

[4] According to Martin (1991): '*a top management document describing future information systems strategy*'.

## 1.5 Influential Knowledge Domains

The purpose of this section is to present important knowledge domains that we wanted to influence the re-design of VIBA/SIMM and to discuss why we believe they are important aspects to consider in systems development.

The presentations follow a two-phased structure. First, we establish our interpretation of what the domain is and why it is important to consider in systems engineering (our intention in using it). Second, we discuss how the domain will affect the re-design of VIBA (our intended use of it).

### 1.5.1 Business Process Orientation

Business process orientation (BPO) means that businesses are viewed as consisting of value adding processes that cut through traditional functions; see Fig 1-1. It is the customers that judge the amount of added value and BPO thereby implies a strong customer orientation.

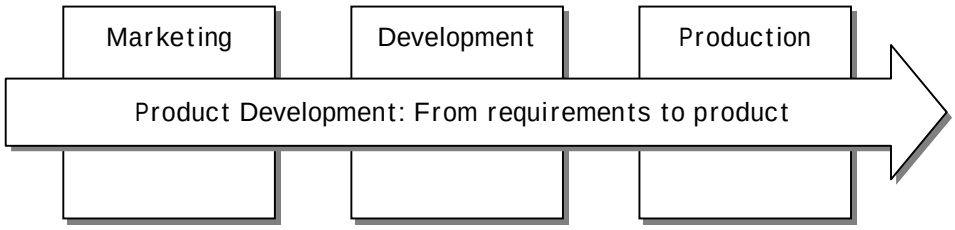



**Fig 1-1:** Business processes cut through traditional functions.

Davenport (1993) explains the difference between traditional hierarchical organization and process organization when asserting that: '*Whereas an organization's hierarchical structure is typically a slice-in-time view of responsibilities and reporting relationships, its process structure is a dynamic view of how organizations deliver value*'.

However, as pointed out by Jacobson *et al.* (1995), the concept of process is nothing new. Within organizations there have always been processes. Applying BPO simply implies that the business' processes are emphasized and taken as a starting point for business modelling rather than the traditional hierarchical division into functions. Nevertheless, a functional division might very well exist in parallel with a process organization.

One reason to favour BPO is according to Jacobson *et al.* (1995, p. 3) that '*Processes arise more naturally than hierarchies because they come into being when people realize that they must cooperate to achieve the result promised to the customer*'. Hence, taking business processes as a starting point for business design would yield (1) a better match be-

tween formal and informal structures, and (2) a smoother implementation of the new design. Another argument for BPO is that it makes it easier to estimate and measure, for example, cost, time, quality, and customer satisfaction (Jacobson *et al.*, 1995).

The applicability of a method is highly related to the problem domain(s) it addresses. The applicability might also depend on desirable types of solutions to problems in a certain domain. For example, a function-oriented method (e.g. SASD, Yourdon, 1989) is probably better suited for adoption to a function-oriented organization than some workflow approach, even if they share the same problem domain (business information systems development). We refer to these two aspects of methods as 'method context'. The method context was previously not well articulated in VIBA. With the re-design we wanted to be more explicit. The method context of VIBA is hence business supporting information systems within process-oriented organizations.

### 1.5.2 Rapid Development

A method is always used together with some framework[5]. A framework is a high-level structure that dictates what to be done without the details of, and specific techniques for, how to do it (cf. section 2.3.2.4). Martin (1991) presented an alternative to traditional development models and in doing so he coined the expression 'Rapid Application Development' or RAD for short.

We interpret rapid development as 'developing information systems fast', following McConnel (1996), and believe that different development models can be used to support such development. Rapid development is often associated with incremental development models, such as, for example, Boehm's (1988) spiral model, and tools for rapid application development, for example Borland Delphi and Microsoft Visual Basic, used with some prototyping approach (e.g. Smith, 1991). But, as McConnel (1996) points out, which development model to chose depends on the development situation and available tools that support rapid development.

According to Martin (1991) RAD:

- avoids specifications becoming obsolete before cutover;
- needs user involvement in design workshops;
- needs prototyping, which helps meet user needs;
- needs user involvement in the construction phase;
- needs CASE tool support;

[5] We will in this thesis use the terms 'framework', 'development model', and 'lifecycle model' interchangeably.

- needs code generators that produce bug-free code; and
- tests the evolving design concurrently with construction.

Martin (*ibid.*, p. 5) further argues that RAD will lead to higher quality, and that '*high quality, lower cost and rapid development, thus, go hand-in-hand if an appropriate development methodology is used*'.

This relation between rapid development, quality and cost is visualized by Fig 1-2. In this context, Martin (*ibid.*) defines quality as: '*meeting the true business (or user) requirements as effectively as possible at the time the system comes into operation*'.

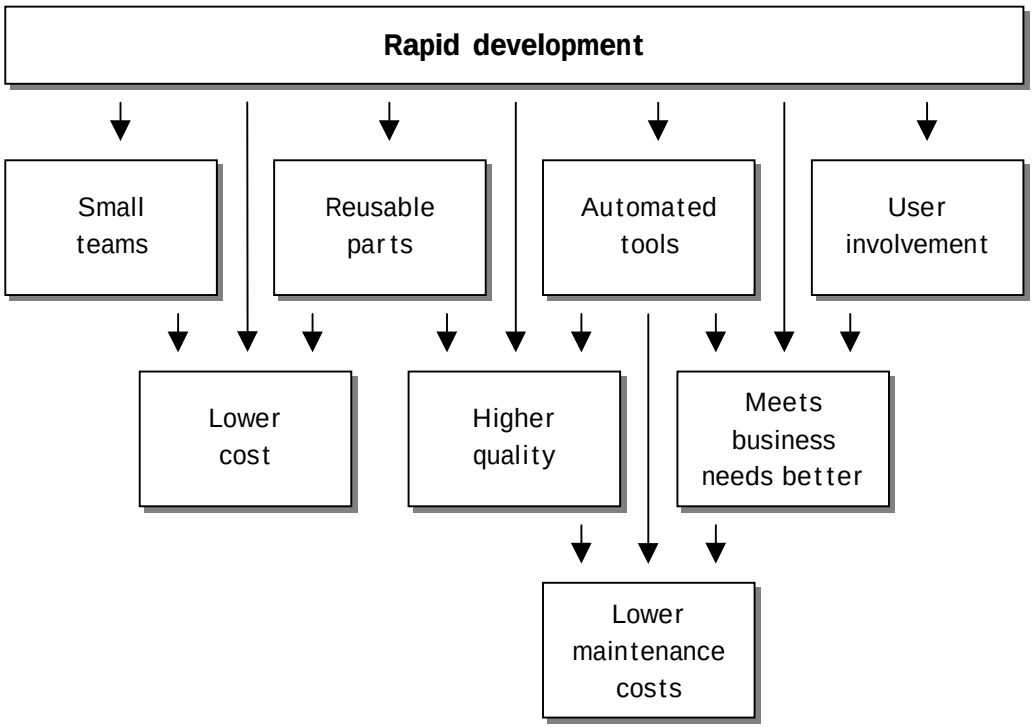



**Fig 1-2:** Rapid development yields higher quality and lower cost (Martin, 1991).

According to McConnel (1996) rapid development does not presuppose any particular development model. Instead, a crucial part of any development project is to choose a model that suits the demands of that particular development situation. For instance, a pure waterfall model might do well if requirements are well understood at the beginning of development and reliability is of high importance. The particular development model that McConnel claims will fit most rapid development projects is the spiral model (Boehm, 1988), which, in some sense, is the direct opposite of a pure waterfall. With a spiral model projects start small and expand in scope in increments. The scope is not expanded until the risks for the next increment are reduced to an acceptable level.

One recent approach to rapid development is the Dynamic Systems Development Method (DSDM), outlined by DSDM (1999). According to DSDM traditional approaches fix requirements early while allowing time and resources to vary during development. In DSDM the opposite is true. While letting functionality vary, time and resources are considered as being as fixed as possible. See Fig 1-3 for an illustration.

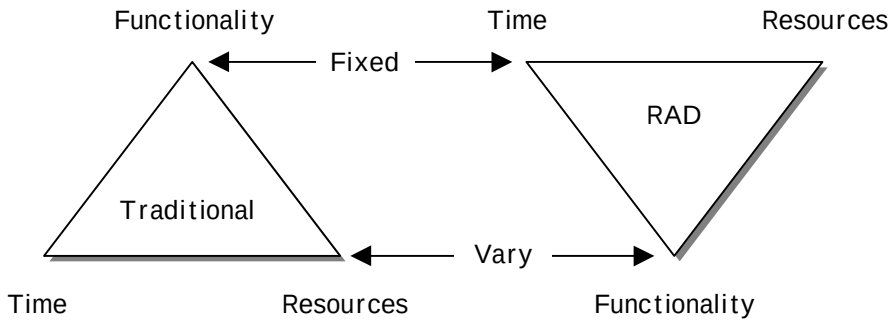



**Fig 1-3:** Traditional development compared to RAD according to DSDM (1999).

Thus, rapid development means that the best products possible, given certain amounts of time and resources, are what to strive for, and hence not 'perfect' products in a traditional sense (i.e. with all requirements fulfilled).

No matter what approach to rapid development one chooses: in order to keep information systems up to date in an ever-changing business environment, information systems must be developed as fast as possible – without decreased quality of the product, including documentation.

### 1.5.3 Object Modelling

A concept that has gained more and more attention is that of 'object'. In entity-relationship (E/R) modelling (Chen, 1976; Sundgren, 1992) an object (entity) represents a concrete or abstract phenomenon in the problem domain that the system is supposed to keep information about. The object consists of relevant information about the real world phenomenon, which can be operated on by functions and procedures as the result of queries and updates. In object orientation (Coad & Yourdon, 1991; Booch, 1994) the functionality is considered encapsulated in the objects and it is usually said that objects 'inhabit behaviour'. Industrial experiences show that object orientation is a preferable way to organize and design software systems. Lately, there have been proposals (e.g. Jacobsson *et al.*, 1995; Graham, 1998) as to how to use object orientation also during business modelling.

Due to object orientation's inherent modularization of software systems, it seems to lend itself well to the construction of systems that

are subject to change frequently – following the principles of Parnas (1972) – in that it helps structuring requirements management. As pointed out above, one key factor of rapid development is the use of automated tools and reusable parts. Contemporary tools for rapid development and graphical user interface programming (RAD-tools) are usually based on object-oriented technology. Hence, to implement rapid development it is in practice necessary, in one way or another, to make use of object-oriented technology, at least during technical implementation. Nevertheless, the object-oriented way of encapsulating behaviour and knowledge into congruent wholes (objects and classes) leads to problems when using today's RAD-tools due to their strong bias toward the relational model of data (Ågerfalk, 1999). Therefore we have chosen not to enforce object orientation during analysis but to provide a smooth transition path from requirements analysis to object-oriented analysis and design.

### 1.5.4 Usability

Advocates of usability emphasize that computerized systems should be usable when performing interactive tasks.

In the Human-Computer Interaction (HCI) literature there are several definitions of the usability concept. Löwgren (1993) defines 'usability' as: '*...the result of relevance, efficiency, attitude and learnability*'. Another similar definition states: '*Usability, a key concept in HCI, is concerned with making systems easy to learn and easy to use*' (Preece *et al.*, 1994). HCI-research is mainly focused on the interface between humans and computers. User interfaces should be unambiguous, flexible and support different categories of users with different demands. Both human and computer aspects are considered. A popular research area in HCI covers different interaction styles and forms (e.g. Preece *et al.*, 1994; Sims, 1994).

In a use-situation, consisting of a user, a task and a tool (Shackel, 1984), the usability perspective aims to cover all these three components. It is well known that all these three components have to be studied equally. Furthermore, they also have to be studied in a context (or environment). See Fig 1-4 for an illustration.

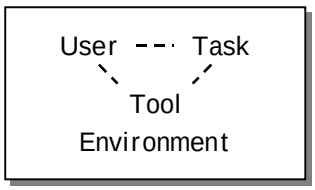



**Fig 1-4:** The four components of an IS use-situation (Shackel, 1984).

To date, the major contributions to the concept of usability have been to the relationship between the user and the tool components (in other words to the human-computer relationship). Less work has been done regarding the relationship between the tool and the task component, or about the relationship between the user and the task component.

Our understanding is also that usability is considered mainly from an individual perspective. Human-Computer Interaction has traditionally focused the dyad of one user using one computer system (Löwgren, 1995). This means that the traditional usability perspective misses the surrounding social context. There is, however, an emerging perspective on usability that consider factors such as the social organization of work and how computers can be used to support it (*ibid.*).

Traditional methods for IS engineering suffer from limitations in their treatment of cognitive and human factors, as well as in their recommendations for the analysis of different interaction styles (e.g. Lif, 1998). Within the HCI field, on the other hand, these aspects are discussed frequently (e.g. Norman, 1988). Nielsen (1993) also discusses the importance of taking into account differences in the experiences of different users. These topics are not stressed in the information systems development field and have so far been completely left out of the language action perspective. Hence, we think that it would be fruitful to combine the usability perspective with that of language action, since the latter primarily focuses on acts (tasks).

When designing IS interaction, usability is important to consider. However, we believe that the common notion of usability is too narrow, since it is often perceived as dealing only with how to design user interfaces. When designing communication through an IS, the question of *how* to interact is, of course, important. Equally important, however, is *what* to communicate and *why*. Moreover, all three aspects must be considered in a *context* where the communication is taking place – that is, a social context that is never static and fully predictable. Thus, Usability promotes a perspective that we adhere to but would like to extend in order to put more emphasis on the business context in which the interaction is taking place. Information systems should be usable not only in the context of interaction but in the context of business.

### 1.5.5 Software Requirements Management

Software requirements management has emerged as an approach to handling changes in requirements and to helping predict changes to software (and change costs) in response to business changes. The basic approach is to maximize traceability of requirements (Lam, 1998). We

believe that this can be done in two different (non-orthogonal) dimensions. The first dimension concerns traceability from business models to detailed system design models. The other dimension cuts across models at the same level of abstraction.

We believe that requirements management is important in order to ease the burden of maintenance and further enhancement. Since information systems are assumed to be subject to frequent changes, some method support is needed for *rapid enhancement* as well as for rapid development. By taking this into account from the very start of systems specification, a better foundation for change is built.

## 1.6 High-Level Re-Design Goals

We have had three high-level re-design goals when working with the redesign of VIBA/SIMM. The first goal was to make the theoretical foundations even more explicit and to take full advantage of their consequences. The theoretical foundation in this case is the Action Theory of Information Systems (ATIS). The second goal was to design a method that produces documentation that is communicable between developers and other stakeholders. The third goal was to make use of as much existing knowledge and good practice as possible. The latter includes reuse of existing modelling formalisms and notations whenever possible. Possible in the sense that they must be convenient to integrate with the proposed action view of information systems.

## 1.7 Putting the Pieces Together

Information systems often fail to meet the requirements of their users. Hence, the specification and evaluation of requirements is a crucial part of the systems life cycle. The management of requirements is a difficult task. It is hard to elicit the appropriate requirements and to formulate them in such a way that they are understandable to users and formal enough for designers to build software. A requirement specification can be seen as a contract between users and designers. The specified requirements must be accept by the users as an assignment for design. The specified requirements must also be accepted by the designers as a proper basis for the design of software.

There are many approaches to systems engineering and to requirements engineering. There are functional methods, such as, for example, Structured Analysis (Yourdon, 1989), information engineering methods (e.g. Martin, 1989) and object-oriented methods, such as, for example, OMT (Rumbaugh *et al.*, 1991).

All these approaches aim at modelling requirements. Advocates of prototyping (e.g. Smith 1991; Budde *et al.* 1992) have challenged such modelling as being too abstract. It is hard for users to understand how an information system will function from abstract models (such as data models and data flow models). The use of simple prototypes of software is claimed to enhance the understanding of future systems among users. Through experience of the behaviour of the future systems, users will come to an understanding of what it might be to use those systems. Prototypes are often used as an aid for designing user interfaces. The underlying principles of the system are however harder to visualize in a prototype. Abstract requirement models are important to use for description, analysis and critique of principles and features that go beyond the user interface. We believe that reconciliation is needed between modelling and prototyping. In requirements engineering there is a need for, on one hand, abstraction and modelling and, on the other hand, visualizing and experiences.

One important aspect to learn from the prototyping versus modelling debate is the need to go beyond static properties of a system. Many system models tend to be modelling only structural and static aspects of systems. Dynamic and active aspects of systems need also to be treated and modelled during requirements engineering.

This is also apparent if we look at the evolution of methods for requirements engineering. Classical methods for data modelling, like the entity-relationship approach (e.g. Chen, 1976), had a focus on fairly static IS aspects. Object orientation can to some extent be interpreted as a critique of such a static emphasis. In object orientation there is an attempt to model both static and dynamic aspects of IS (e.g. Rumbaugh *et al.*, 1991). The further development of object orientation goes into more dynamic aspects of information systems. The use case notion (Jacobson *et al.*, 1992) is an attempt to capture the intended behaviour of a system from its outside.

The historical evolution of requirements engineering shows an increasing interest in information systems as dynamic and active entities. Both rapid development (including prototyping) and object orientation are examples of this evolution. Unfortunately these approaches, although with many good features, lack a proper understanding of information systems as organizational phenomena. They fail to see the genuine social character of information systems. Information systems are important instruments for organizations to perform their actions. Information systems are systems for action.

Requirements engineering (RE) is really a ‘never-ending story’. The story begins with some fuzzy ideas about how a computerized IS

might support the way business is done (or perhaps ought to be done). The process of eliciting and formalizing such ideas into a requirements document (software requirements specification) is generally referred to as requirements elicitation or requirements definition and specification (Sommerville, 1996). We stress that the RE process does not end with such a document. During the life of an IS, requirements are likely to change many times in order to keep up with an ever-changing business environment. Thus, requirements change management becomes an important part of the entire RE process (c.f. Lam, 1998; Lam *et al.*, 1998).

The aim of this thesis is to present and elaborate on the design of a method for information systems engineering based on an Action Theory of Information Systems (ATIS). The scope of the thesis has been presented in terms of:

- An Action Theory of Information Systems (ATIS) that will serve as foundation for a re-designed VIBA/SIMM and hence as rationale for method design-goals.
- The assumption that organizations and businesses are subject to change at an ever-increasing speed forms the context of ATIS and hence of VIBA/SIMM.
- Influential knowledge domains of the information systems field, suitable for handling high business change rates, which are re-interpreted in the light of an action-oriented view of information systems.

Fig 1-5 shows how these parts are related.

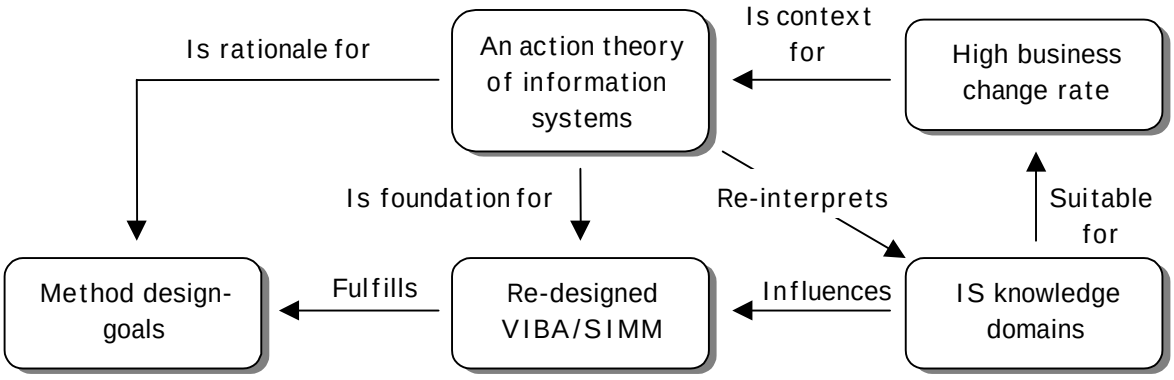



**Fig 1-5:** Scope of thesis work.

The IS knowledge domains that have influenced this work are object orientation, business process orientation, usability, rapid development and requirements management. Fig 1-6 shows how these relate in the context of high business change rate.

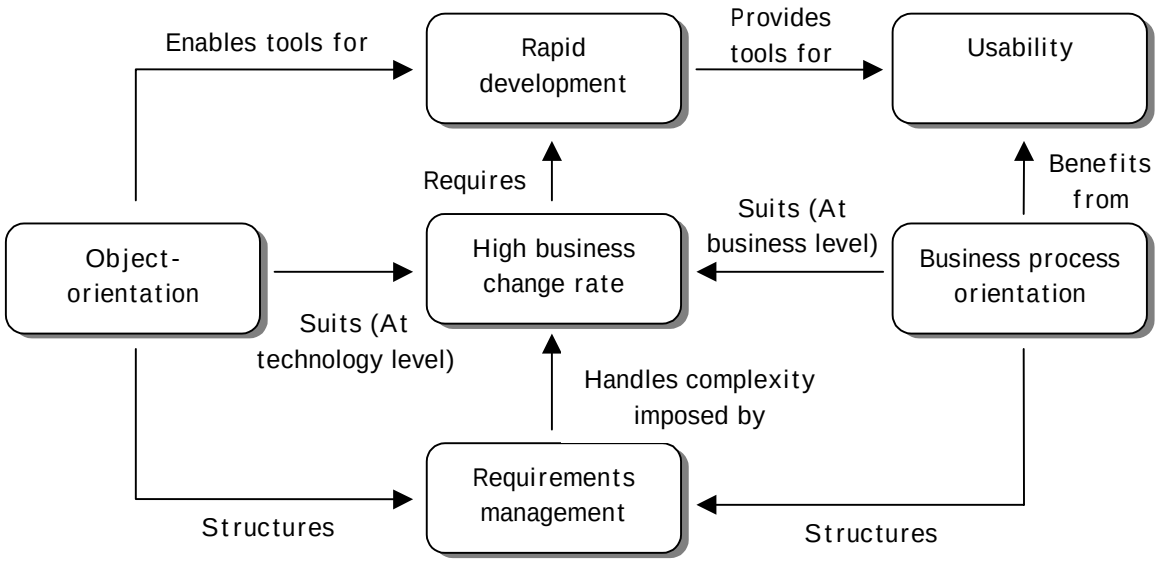



**Fig 1-6:** Influential IS knowledge domains.

The result exists on two levels: theoretical and methodological. As the title of the thesis indicates, we do not consider the result as complete. In fact, we do not think that the type of knowledge represented by this work can ever be fully complete. Just as information systems will continue to evolve after implementation, so will the knowledge of building them. The result should therefore be regarded as a well-grounded and well-argued proposal (or hypothesis for that matter). One reason for this seemingly defensiveness is the fact that all empirical work has been performed as action research and therefore the method has not been fully verified to scale to real system developing practitioners' demands.

## 1.8 Self Criticism

### 1.8.1 Why a Method Based Approach?

This work proposes a method based approach to the problem of developing information systems for rapidly changing environments. Such an approach can be challenged and there are others. One such alternative approach is to let business actors develop their systems themselves. That approach, referred to as 'end-user computing' (EUC) seems to be a possible alternative with the use of today's powerful end-user tools such as spreadsheets (e.g. Avdic, 1999). An achievement gained with the

EUC-approach is that the problems of communicating requirements between users and developers are eliminated. With our understanding of the concept of method (see chapter 2), even the EUC-approach can be regarded as a kind of method based approach that needs support in terms of conceptualizations and forms of documentation.

A criticism against a method based approach is the seeming paradox between the use of a rigorous method (which produces lots of documentation) and the need for rapid development of information systems. However, informal pre-investigations of this work as well as McConnel (1996), Lam (1998), and Martin (1991) show that, if system maintenance and evolution is taken into account, well documented systems and structured engineering practices are imperative to handle changes and to shorten development schedules.

### 1.8.2 Yet Another Method?

A reasonable question to ask is that of 'yet another method', i.e. does the systems engineering community really need a new method for specifying information systems? The question is, however, not fully applicable to our work since we do not propose a completely new method, but a redesign of an existing one with influences from other existing approaches. The reason for not settling for what has already been proposed within the systems engineering field is that contemporary methods do not seem to give the necessary support for specifying information systems in a way that emphasizes the action character of information and information systems needed to achieve the goals set up for this work. However, many of the weaknesses in contemporary methods lie in their perspectives rather than their concepts and notations (see chapter 2 for a clarification of this distinction). Therefore, many existing notations and modelling formalisms (both from the existing VIBA and other methods) have been retained and, as stated above, re-interpreted in the light of an action view of information systems.

## 1.9 Demarcations

This work, as well as the method VIBA/SIMM, is primarily concerned with the early phases of systems engineering and its implications for later phases. The focus is thus on requirements engineering, i.e. requirements elicitation, analysis, definition, and management (cf. e.g. Sommerville, 1996). The outcome of VIBA is a detailed specification of the requirements of planned systems and their business environment.

By ‘methods’ we refer to methods for information systems engineering if not otherwise explicitly stated. However, much of the general discussions ought to be applicable to other areas as well.

In this thesis we use the terms ‘information system’ and ‘computerized information system’ synonymously.

## 1.10 Contributors and Project Setting

This work has been performed in a co-operative setting and the outcome, of which this thesis is a part, is thus the result of a collaborative research effort. Active participants in the project have been, the author of this thesis Pär J. Ågerfalk, Örebro University; Göran Goldkuhl, Linköping University and Jönköping International Business School; Stefan Cronholm, Linköping University; and, to some extent, Owen Eriksson, Dalarna University. In addition, Pär Fahlén, Örebro University, participated initially in the first case study (see chapter 2). So far the work has resulted in five published scientific papers upon which this thesis is based:

- Goldkuhl G and Ågerfalk P J (1998). *Action Within Information Systems: Outline of a Requirements Engineering Method*. Presented at 4th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ’98), Pisa, Italy, June 8–9, 1998.
- Ågerfalk P J and Goldkuhl G (1998). *Elicitation and Analysis of Actability Requirements*. Presented at 3rd Australian Conference on Requirements Engineering (ACRE’98), Geelong, Vic, Australia, October 26–27, 1998.
- Cronholm S, Ågerfalk P J (1999). *On the Concept of Method in Information Systems Development*. Presented at 22nd Information Systems Research Seminar in Scandinavia (IRIS 22), Keuruu, Finland, August 7–10, 1999.
- Cronholm S, Ågerfalk P J and Goldkuhl G (1999). *From Usability to Actability*. Presented at 8th International Conference on Human-Computer Interaction (HCI International’99), Munich, August 22–27, 1999.
- Ågerfalk P J, Goldkuhl G and Cronholm S (1999). *Information Systems Actability Engineering: Integrating Analysis of Business Processes and Usability Requirements*. Presented at 4th International Workshop on the Language Action Perspective on Communication Modelling (LAP’99). Copenhagen, Denmark, September 12–13, 1999.

This thesis, and thereby its author, contributes to the project by (1) giving a coherent reconciliation of the previously published papers[6] with further elaboration on relevant topics, and (2) documenting and analysing the empirical work that has been performed.

In addition to the five papers mentioned above, two further papers have served as background for the thesis:

- Ågerfalk P J and Åhlgren K (1999). *Modelling the Rationale of Methods.* Presented at 1999 Information Resources Management Association International Conference. Hershey, PA, USA, May 16–19, 1999.
- Ågerfalk P J (1999). *Encapsulation or Availability: On the Combination of Objects and Relations in Systems Development.* Presented at 22nd Information Systems Research Seminar in Scandinavia (IRIS 22), Keuruu, Finland, August 7–10, 1999.

Due to the fact that much of the work has been performed as co-operative knowledge development, and the inherent impossibility of pinpointing the source of every specific knowledge fragment, the author has chosen to adopt the convention of using 'we' in favour of 'I' throughout the entire thesis.

## 1.11 Reading Instructions

### 1.11.1 Intended Audience

This thesis has been written with two large target groups in mind. Research colleagues within the fields of requirements engineering, information systems and human-computer interaction constitute the first group. Practitioners out there, struggling with the kinds of problems addressed by the thesis, constitute the other. With the varying theoretical and formal backgrounds this audience probably represents, trying to write clearly and concisely without being either too inconsiderate or too prosaic has been one of the most intricate pedagogical problems during this work. However, the presentation of the material assumes that readers possess a general knowledge of systems engineering and are used to following discussions and argumentation at a fairly abstract level. Existing, well-documented, formalisms and notations are not described in detail. Instead references are given to relevant sources. In particular the presentation of VIBA/SIMM (chapters 5 to 7) assumes familiarity with the Unified Modelling Language (UML) and its notation for class models

[6] Due to this circumstance, the papers mentioned are not explicitly cited but used and incorporated in the text throughout the thesis.

(Class diagrams) and state models (Statecharts); please see, for example, Booch *et al.* (1999) for an introduction.

### 1.11.2 Notational conventions

Throughout the thesis we will use an informal notation, as of figures 1 and 2, called Concept diagrams to show conceptual relationships. The arrows of Concept diagrams are used to indicate intended direction of reading. Solid arrows indicate a relationship between two concepts and dotted arrows indicate the special relationship 'is a' in the sense that the 'from-concept' constitutes a subset of the 'to-concept'; i.e. a generalization/specialization relationship.

### 1.11.3 Thesis Outline

The thesis is conceptually divided into five parts.

The first part (chapters 1 and 2) presents some background and establishes the aim of the work as well as the research approach adopted.

The second part (chapters 3 and 4) elaborates on the Action Theory of Information Systems (ATIS), i.e. the theoretical framework, as a reconciliation of the presented assumptions and influences.

The third part (chapters 5 to 7) presents the re-designed VIBA/SIMM as a result of methodological consequences of part two.

The fourth part (chapter 8) evaluates the two cases of VIBA usage that has been performed as parts of the method engineering process.

Finally, the fifth part (chapter 9) concludes the work and points at some possible future research directions.

# Chapter 2

# Field of Study and Research Approach

*For every complex problem there is an answer that is short, simple and wrong.*

H. L. Mencken

The purpose of this chapter is to present and discuss the research approach used. The aim of the thesis is to articulate a theory of information systems as action and to show the design of a method for systems engineering based on that theory. To be able to reason about theories and method development we must first establish what is to be meant by 'theory' and 'method' in the context of information systems. We therefore start by discussing what a theory for information systems is and relate it to other notions from the information systems literature. We then present our concept of method and discuss how methods can be constructed. Finally the research approach is presented and related to the previous discussions.

## 2.1 Development Versus Engineering

This thesis is about development of theory and method for systems development. When using the term *development* we refer to the design or construction of some artefact, for example a system (system development) or a method (method development). Development is performed whenever an artefact is created, in whatever way. When using the term *engineering* we refer to a special kind of development. When development follows some kind of structured and planned practice we say that we are engineering, i.e. that engineering is taking place. Engineering is thus the opposite of *ad hoc* construction.

## 2.2 Information Systems Theory

Different authors have discussed information systems development in terms of *theory* (e.g. Magoulas & Pessi, 1991; Langefors, 1966), *perspective* (Goldkuhl, 1994; Opdahl, 1998), *tradition* (Näslund, 1996), and *paradigm* (Nurminen, 1988). In its most general form, a theory can be thought of as a set of definitions and axioms, and theorems derived from

the definitions and axioms. Euclidean geometry is an example of such an 'ideal' theory. In this thesis a somewhat more generous interpretation of the concept of theory is used. A theory of information systems is here thought of as a set of properties that constitutes what is meant by information and information systems. This concept of theory is similar to the concept of perspective. According to Opdahl (1998) a perspective is a mental representation of the world in terms of interesting/purposeful[1] *things* and a selection of interesting/purposeful *properties* of those things. Opdahl stresses that a perspective is subjective and that different people might see different things when looking at the same piece of reality. Thus, a theory of information systems is an externalized, possibly inter-subjective, conceptualization of interesting properties of information, information systems, and information systems development. Put another way, a theory emphasizes certain kinds of properties and directs attention to particular kinds of things and properties of those things. When perceiving a phenomenon based on such a theory, a perspective is applied.

When several people share the same perspective and work according to a common theory their ways of working eventually become institutionalized into a tradition. People within a tradition thus share a common set of values and conceptualizations and '*take certain aspects more or less for granted*' (Näslund, 1996). Examples of traditions that Näslund mentions are *the information system methods tradition, the software engineering tradition*, and *the human-computer interaction tradition*. Näslund further claims that traditions are similar to scientific paradigms, in the sense of Törnebohm (1976). We do however believe that traditions differ from paradigms since many people work across the borders of two or more traditions, i.e. what Näslund (*ibid.*) refers to as '*boundary spanners*', and thus are members of many traditions. Nurminen (1988) makes a more accurate interpretation of paradigm in the discussion of *the system theoretical paradigm*, *the socio-technical paradigm* and the *humanistic paradigm*. The emergence of each of these paradigms follows the kind of revolutionary change scientific paradigms are supposed to follow according to Kuhn (1970).

Tradition and theory are not equivalent classification schemes. Within a tradition there might exist different theoretical assumptions that form groups within a tradition. Groups might even span several traditions, just like individual tradition members. Within a tradition there are assumptions (explicit or implicit) about interesting problem domains and desirable types of solutions to problems. Different tradi-

[1] That is '*interesting/purposeful for the situation or activity at hand*' (Opdahl, 1998).

tions might however regard the same or similar problems as interesting but approach them in different ways. For example the field of requirements engineering is highly relevant to both to the information system method tradition and the software engineering tradition.

Using the classification by Näslund (1996), this work has been carried out within the information system method tradition, but with inspiration from the software engineering and the human-computer interaction traditions.

## 2.3 The Concept of Method

The concept of method has been discussed through several decades (e.g. Bubenko & Källhammar, 1969; Checkland, 1981; Goldkuhl & Lyytinen, 1982; Jayaratna, 1994). Information systems engineering (ISE) methods are often used during development of information systems in order to guide and support the ISE process, i.e. ISE methods can be though of as normative conceptualizations that direct method users' attention to certain kinds of phenomena. Hence methods are created and used to support ISE actors to perform different tasks aiming to reach some goals (Ågerfalk & Åhlgren, 1999).

However, the meaning of the concept of method is not always clear. When studying different theories about methods or looking into different practical manuals it is obvious that several definitions of the concept of method exist, and there are also new emerging concepts. Furthermore, there exist several related concepts – often with 'method' as a prefix and a following noun (method alliances, method components, method fragments).

When looking more closely at different concepts, one can identify that there are different concepts (and terms) used for the same phenomenon and also the same concept (and term) for different phenomena. The definition of 'method' chosen in this thesis originates from several researchers who have all contributed to the understanding, and to the development, of the concept of method. The aim of this section is to discuss different method concepts and explain how these are related to each other to serve as rationale for the use of the concept of method in this thesis.

### 2.3.1 Method Versus Methodology

Jayaratna (1994) defines 'method' as '*an explicit way of structuring one's thinking and actions. Methodologies contain model(s) and reflect particular perspectives of 'reality' based on a set of philosophical paradigms. A methodology should tell you 'what' steps to take and 'how' to perform*

*those steps but most importantly the reasons 'why' those steps should be taken, in a particular order*'. As we can see Jayaratna uses the term 'methodology'. Methodology is a Greek term meaning the study of methods. The Oxford English Dictionary defines methodology as '*the study of systematic methods of scientific research*'. Jayaratna justifies the use of methodology claiming: '*however, the term methodology is pragmatically well established within the field of information systems to mean the same as method*'.

Another example of the use of 'methodology' synonymously with 'method' is that of Stamper (1988), when stating that '*I use the term 'methodology' under protest bowing only to customary usage. It would be better, as in philosophy of science, to speak of 'methods' when referring to specific ways of approaching and solving problems, and to reserve 'methodology' for comparative and critical study of methods in general; otherwise this vital field of study is nameless*'.

A third example of the misuse of the term methodology is by Brinkkemper (1996), who states that '*the misuse of the term methodology standing for method is a sign of the immaturity of our field, and should consequently be abandoned*'. Brinkkemper (*ibid.*) further defines method as an '*approach to perform systems development project, based on a specific way of thinking, consisting of directions and rules, structured in a systematic way in development activities with corresponding development products*'.

A third definition of the concept of method is that by Röstlinger & Goldkuhl (1994). Their definition reads: '*methods are prescriptions for human actions and methods are normative and guide the ISE process*' (our translation to English).

Another definition from Checkland (1981) runs: '*a methodology will lack the precision of a technique but will be a firmer guide to action than a philosophy. Where a technique tells you 'how' and a philosophy tells you 'what', a methodology will contains elements of both 'what' and 'how' '*. Checkland also uses the term methodology when he actually means method. Furthermore, Checkland uses the concept 'technique'. Exactly what he means by technique is not defined, but we believe that he is referring to a diagramming technique, such as data flow diagramming or entity-relationship diagramming.

When examining the method definitions above, it is clear so far that the term 'methodology' is often used when what is actually referred to is 'method'. Method is descended from the Greek language, meaning 'way of investigation'. The meaning of 'method' seems to answer the question how ISE shall be performed.

### 2.3.2 Other Method-Related Concepts

#### *2.3.2.1 Method Types*

Nilsson (1991) presents the concept of 'method type' (in Swedish: 'metodik', there is no corresponding established English term). Nilsson distinguishes between a method type and a method. Nilsson defines a method type as general concept (a type of methods) and a method as a specific concept (an instance). In other words, a method is a concrete representation of a method type. Following Nilsson's definition, object orientation is an example of a method type and the Object Modeling Technique (OMT) is an example of a method of that type.

#### *2.3.2.2 Method chains and alliances*

Fåhraeus (1986) talks about 'method chains' as consisting of several methods linked to each other. Further, the result from a method used in an earlier step shall be used in a following step.

Nilsson (1999) has further developed the concept of method chain. Nilsson's definition runs: '*Integration of methods between different levels of development work. This approach to combine methods is a kind of vertical integration*'. Nilsson points out that there are several (abstraction) levels of development work. For example there could be a higher level dealing with conceptual modelling, followed by object modelling performed in a lower level. The object model can in turn be used when defining the database schema. Nilsson's division into a vertical integration makes sense when he also introduces the concept 'method alliance'.

A method alliance is an integration of methods within the same level of abstraction. This is a horizontal integration of methods. Nilsson states that alliances are motivated by the need '*to tackle several problems or perspectives in concrete situations*'. That is, method alliances cover several aspects of a problem domain at a specific level. In our opinion a method should cover several phases and aspects within ISE. We thus think of 'method' as addressing the ISE into a congruent whole. We interpret what Fåhraeus and Nilsson mean by 'method' rather as method fragments or method components (defined below).

#### *2.3.2.3 Perspective*

Another method-related concept is 'perspective'. A perspective is a theory of how ISE shall be performed (Nurminen, 1988). Such a theory shall be normative, explanative and classifying. Mathiasen (1982) defines perspective as a conceptual abstraction of a view of a specific phenomenon (cf. section 2.2). Jayaratna (1994) says, '*methodologies ... reflect*

*particular perspectives of 'reality' based on a set of philosophical paradigms*'.

In other words, the method constructor's perspective is based on how he or she perceives the world. The method constructor's values and beliefs thus influence the method user when performing ISE. A perspective implies, for example, what primitives to use and these primitives in turn influence method users (cf. Ågerfalk & Åhlgren, 1999). The character of the influence can be either governed or supported. The perspective is not necessary made explicit in the method. The method constructor's perspective is often implicit and taken for granted. One can say that a method is always based on a perspective with which follows, for example:

- principles,
- values,
- conceptions,
- experiences,
- categories, and
- definitions.

To sum up, the perspective, explicit or implicit, influences the method user in one way or another. We can distinguish between internal and externalized perspective of a method creator (or any human being). The internal perspective is constituted by the parts of the conception of the world that are hard (or even impossible) to externalize. The externalized perspective, on the other hand, is constituted by inter-subjective values, beliefs, *et cetera*, to which the method creator adheres. Examples of existing externalized perspectives in ISE are business orientation, object orientation and user centred design.

#### *2.3.2.4 Framework/Model*

Another method-related and sometimes confusing term is 'model'. What do we actually mean when talking about models? According to Yourdon (1989) a model is used to '*highlight, or emphasize, certain critical features of a system, while simultaneously de-emphasizing other aspects of the system*'. Examples of classical notations to express models are data flow diagrams and entity-relationship diagrams (*ibid.*). Rumbaugh *et al.* (1991) define 'model' as '*an abstraction of something for the purpose of understanding it before building it*'.

Jayaratna (1994) defines 'framework' as a static model, which provides a structure to help connect a set of models or concepts. Goldkuhl (1991) defines 'model' as a structure for the ISE process. Further, a model defines and delimits specific areas within ISE that form related

phases. A model answers the question of *what* to be done but not *how* it should be done. Examples of such models are the classical Swedish SIS/RAS model and the LOGIC model.

What makes the definitions above confusing is that they are referring to different domains. When examining Yourdon's definition, it is obvious that he is referring to a model of an IS. The same goes for Rumbaugh *et al.*. However, when Goldkuhl talks about models he is referring to a model of the ISE process. In other words, they are using the same term but referring to different concepts. Jayaratna (1994), similarly to Goldkuhl, refers to the ISE process whilst the others refer to the product of such a process.

To avoid confusion we think that it is better to, as Jayaratna does, use the concept of 'framework' when referring to the ISE process. In Röstlinger & Goldkuhl (1994) the framework concept is also used as a synonym to model. The concept 'framework' is well defined in the software engineering community but not fully applicable in the information systems community.

One definition from the software engineering community runs (Öberg, 1998): '*A framework is a generic design solution to a certain problem or a certain domain. The framework describes the different design elements involved in the solution, as well as their relations*'. If one changes the term 'design solution' to 'ways of performing ISE' and the term 'design elements' to 'phases' the definition becomes similar to Röstlinger & Goldkuhl's (1994) definition of 'framework/model'.

#### *2.3.2.5 Method Components and Method Fragments*

A method can be perceived as a 'whole' consisting of different 'parts'. Therefore we also need a concept for the parts of a method. During the last few years, concepts such as 'method components' (Röstlinger & Goldkuhl, 1994) and 'method fragments' (Harmsen, 1997) have been proposed to talk about method parts. A reason for this is a move from viewing methods as monoliths to a generic flexibility (Röstlinger & Goldkuhl, *ibid*.) suited for situational method engineering (Harmsen, 1997; Brinkkemper *et al.*, 1998).

The concept 'method fragment' is defined by Harmsen (1997) as '*... a description of an IS engineering method, or any coherent part thereof*'. From this definition, a complete method, for example OMT, is a method fragment and so is any single concept used within OMT, as for example 'object'. To sort this out, a method fragment is said to reside on a certain *layer of granularity*, of which five are possible: method, stage, model, diagram, or concept. Thus, 'object' resides on the concept layer and 'OMT' on the method layer. Furthermore, a method fragment is either a

*process fragment* or a *product fragment*. Process fragments represent the activities, stages, *et cetera*, that are to be carried out and product fragments represent deliverables, diagrams, *et cetera*, that are to be produced, or that are required, during development.

Röstlinger & Goldkuhl (1994) view methods as constituted by exchangeable and reusable components. Each component consists of descriptions for ways of working (a process), notations, and concepts. A process describes rules and recommendations for the ISE and informs the method (component) user what actions to perform and in what order. Notation means semantic, syntactic and symbolic rules for documentation. Concepts are categories included in the process and the notation. A method component can be part of a method chain or a method alliance. A method component or fragment can also be used separately and independently from other components. Each method component addresses a certain aspect of the problem at hand. Examples of method components are 'use case analysis' (Jacobson *et al.*, 1992) and 'finding classes & objects' (Coad & Yourdon, 1991), which are both parts in a whole (a method).

Thus, a method component can be thought of as the smallest *meaningful* assembly of method fragments to address a certain aspect of a problem (cf. Brinkkemper *et al.*, 1998) and consists of product fragments (notation), process fragments (process) and concept fragments (concepts) used in the other two types of fragments. Note that a method component *per se* is a method fragment at some intermediate layer of granularity.

#### *2.3.2.6 Co-operation Forms*

The Scandinavian tradition of performing ISE often means that several actors are involved in the ISE process. Hägerfors (1994) describes the ISE process as a group process with actors who interact, discuss, learn, agree, disagree and argue. Several research reports argue for a strong user (business actor) participation. This means that methods also should support co-operation forms. According to Goldkuhl *et al.* (1997) co-operation forms describe how different persons interact and co-operate when performing method guided work. Co-operation also has to do with roles and division of work. One can say that co-operation forms deal with the meta-question of who is to ask the questions during ISE (*ibid.*). Examples of co-operation forms are brainstorming sessions, interviews and modelling sessions. Harmsen (1997) distinguishes between two different domains that are in focus during ISE. Some ISE activities belong to the *target domain* and some to the *project domain*. The target domain consists of activities directly addressing ISE and the project domain consists

of activities addressing management of ISE. Co-operation forms thus seem to belong to the project domain.

### 2.3.3 Relating the Concepts

In this section we relate the previously identified concepts (sections 2.3.1 and 2.3.2) to each other (see Fig 2-1).

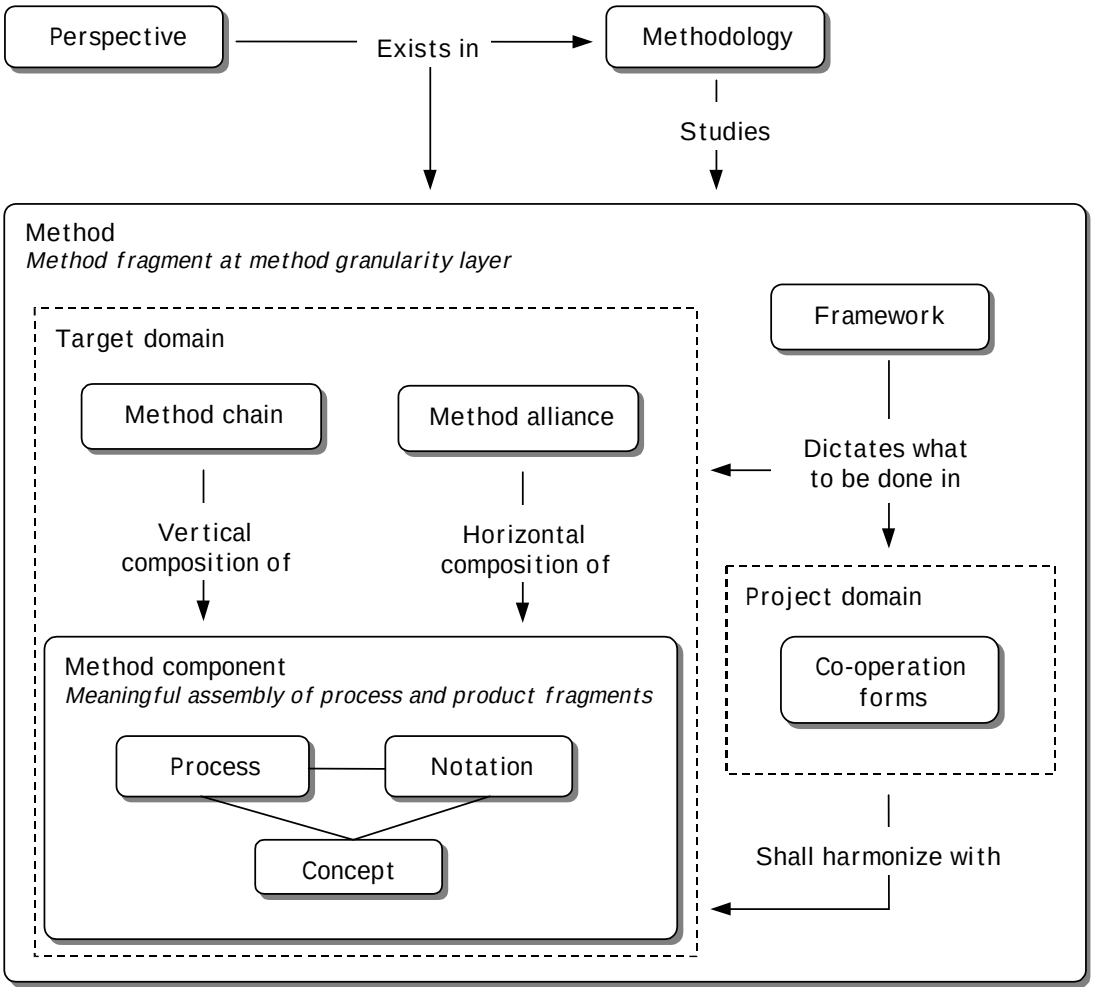



**Fig 2-1:** The concept of method and other related concepts.

The analysis shows that there is a need for a distinction between 'method' and 'methodology'. As mentioned above, the relation between them is that methodology *studies* methods. A method can be thought of as a method fragment at the 'method' layer of granularity. Two distinguishable focal areas of methods have been identified as the *project domain* and the *target domain*. The recommended co-operation forms of a method belong to the project domain. Note that the project domain itself contains methods for project management, process improvement, techniques for gathering data, *et cetera*; this yields a recursive relation not

explicitly shown in Fig 2-1. Such project domain activities are, however, outside the scope of this thesis. It is important, however, that the project domain shall *harmonize* with the target domain.

A method implies a perspective. As discussed previously, a perspective is either explicit or implicit in the method (or methodology for that matter).

Furthermore, a framework *dictates what to be done* during ISE and thus relates to both the target domain and the project domain. As we see it, a many-to-many relationship exists between 'framework' and 'target domain' as well as between 'framework' and 'project domain'. Thus, one method chain or method alliance can be used in different frameworks. There is also a many-to-many relation between method chains and method components as well as between method alliances and method components. A method chain is a *vertical composition* of one or more method components and one method component can be used in different method chains. Similarly, a method alliance is a *horizontal composition* of one or more method components, in which each method component might be used in several method alliances. Finally, a method component is a meaningful assembly of process fragments and product fragments and the concepts used within those fragments. By meaningful, we refer to appropriateness for addressing a certain aspect of a problem at a potentially re-usable granularity layer.

The focus of this work is on the method component part of the model in Fig 2-1, and its relation to perspective (and hence theory).

### 2.3.4 Existence Levels of Methods

Goldkuhl (1994) proposes to classify methods according to their level of existence. With that classification methods exist on (1) subjective level, (2) inter-subjective level, (3) linguistic level, (4) action level, or (5) consequence level. The subjective level refers to knowledge and skill held by some individual. The inter-subjective level refers to two or more individuals sharing the same knowledge. The linguistic level refers to externalized prescriptions for action. The action level refers to actions that follow method prescriptions. Finally, the consequence level refers to materialized effects of method-following actions. Two special cases of levels 3 and 4 occur when methods are being implemented in software tools (e.g. CASE tools). Level 3 is then formalized and implemented in software and a computer performs some of the actions of level 4 (possibly in interaction with human method users).

This thesis deals primarily with methods at the linguistic level, action level, and consequence level; i.e. levels 3, 4, and 5.

## 2.4 Method development

Method development means developing methods. Following the discussion in section 2.1, method engineering refers to the process of conceptualising and externalising a method in a structured way. Construction of methods that resides only at subjective level (in someone's mind) is thus not regarded as method engineering. In this thesis we use the two terms interchangeably and refer, if it is not explicitly stated, to structured development of methods at the linguistic or action levels.

Methods direct method users' attention to certain kinds of phenomena. Some of these are recognised as primitives, i.e. kinds of modelling elements. That is, the focus during method use is determined by the questions asked to abstract analytically. Methods include prescriptions that direct attention towards such primitives. The prescriptions are, in some sense, realizations of the methods goals (perspective), which are always based on some values. The action prescriptions are then instantiated in actions during method use, which leads to some effect. Fig 2-2, which is based on Goldkuhl (1994), illustrates these concepts.

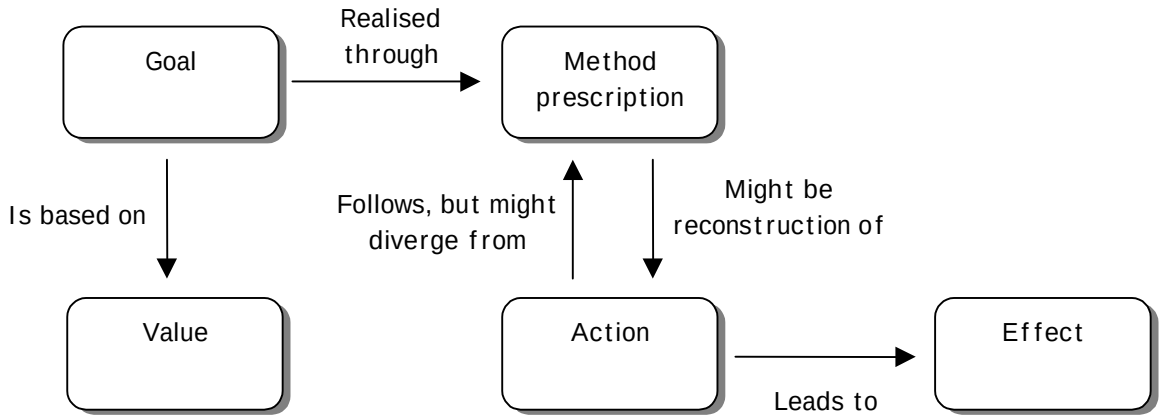



Fig 2-2: Method as action (Ågerfalk & Åhlgren, 1999).

One way of constructing methods is by reconstructing successful actions (that is actions that lead to desired effects). Method construction can thus be a kind of conceptualization and externalization of already institutionalized implicit methods, in the sense of Berger & Luckmann (1989). Another approach is to begin with some goals or theory and deduce method prescriptions that are consistent with the theory.

The rationale of a method is the argumentation that connects the parts of Fig 2-2 into a congruent whole. With this view method engineering is the process of conceptualizing (creating or assembling) processes, concepts and notations into a congruent whole with respect to

some problem domain and desired kinds of solutions (effects). Method engineering also includes the process of externalizing the method in, for example, a method handbook and/or tool-support.

According to Goldkuhl (1994), the argumentation that constitutes method rationale should be built up in two stages. The first stage – internal grounding – concerns how the pieces of a method fit together. That is, an argumentation from a theoretical point of view. This includes legitimization of method prescriptions by relating them to goals and values (theory grounding). The second stage – external grounding – concerns empirical tests of the method in use. These theoretical/empirical activities might very well be performed in parallel with a continuing change of focus between them. Internal arguments are built up partly with results from empirical tests and empirical tests are performed in the light of the theoretical argumentation in a dialectic process.

## 2.5 Research Approach

An ideal but naive research process could have been performed in three stages. Firstly, the Action Theory of Information Systems (ATIS) could have been designed (articulated and externalized) based on general language action theory and the influential knowledge domains. Secondly, ATIS could have been used as theoretical foundation for the re-design of VIBA/SIMM. Finally, empirical tests of VIBA could have been performed to verify both VIBA and ATIS. Fig 2-3 illustrates this process. In the figure, solid arrows indicate information flows and dotted arrows indicate flow of activities.

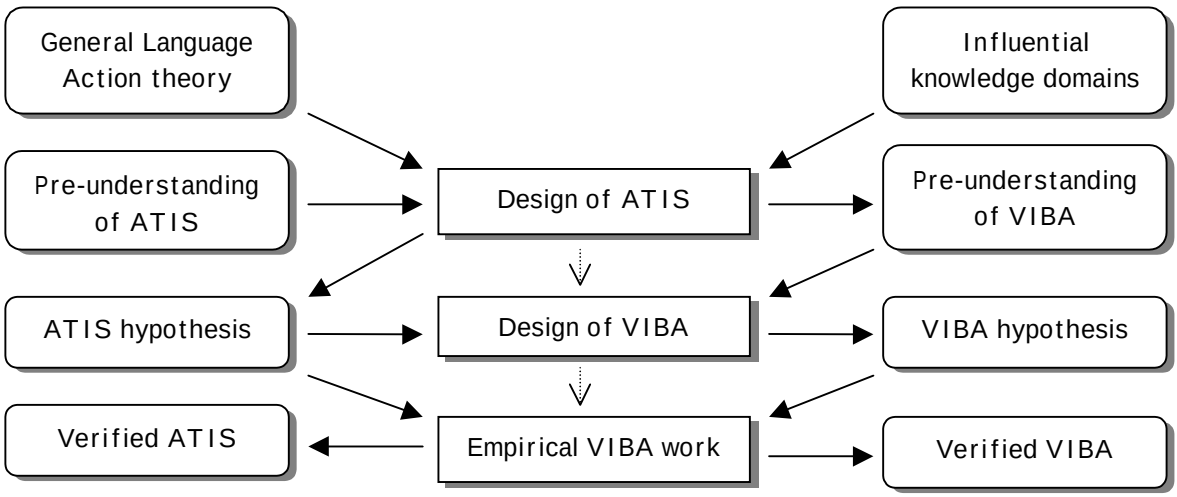



**Fig 2-3:** An ideal but naive research process.

Following the research process of Fig 2-3 would have restricted us from letting the empirical work influence the design of ATIS. Hence, in the spirit of hermeneutics we wanted to be open for changes in the design of both ATIS and VIBA as far as possible. The main message of hermeneutics is that knowledge development is cumulative and that the understanding of a certain phenomenon depends on the observers pre-knowledge at the time of observation. Thus, by re-evaluating the 'current' understanding with the new understanding as pre-knowledge, we gain further knowledge of the phenomenon studied. This is usually described by the so-called 'hermeneutic spiral'.

This led to the actual research process as shown in Fig 2-4 and discussed below.

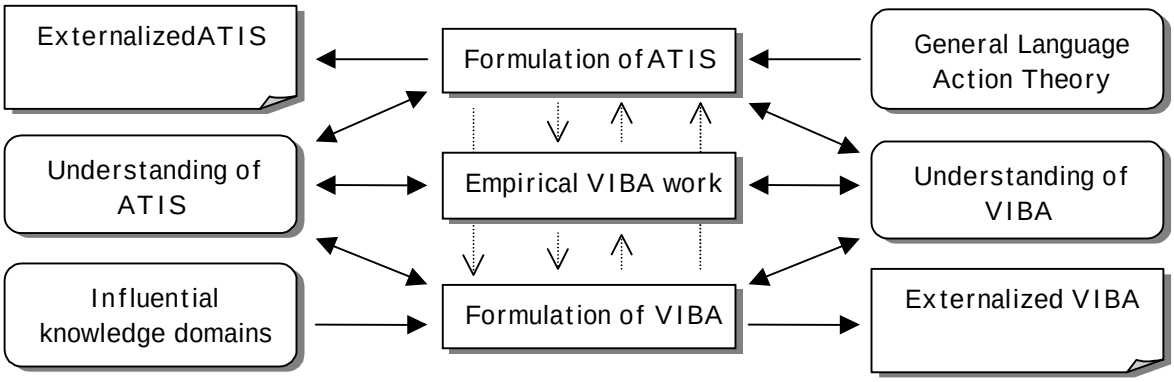



**Fig 2-4:** The actual research process.

Following the discussion in section 2.4, the re-design of VIBA has been performed in two primary focal areas. The first main focus was method construction and internal grounding. The second main focus was external method grounding, which also led to further method development and internal grounding (the 'hermeneutic spiral'). As a tool for internal grounding we have formalized parts of ATIS. To express the theory in a formal (mathematical) way helped us to reveal inconsistencies and forced us to 'think through' the different concepts used, and their relationships. The formalization is, however, not fully complete, and was never intended to be. It is consequently not included in this thesis.

As initial inputs we had the existing VIBA, the previously described prerequisites in terms of assumptions, design-goals and influences, and the theoretical foundations of information systems as action systems. We thus were in a position that lent itself to a mixed approach of deductive and inductive method construction. We had the existing VIBA with some prescriptions that were proven useful and that we wanted to keep. This is an example of an inductive approach, even

though we did not have to reconstruct the method since it was already documented. On the other hand, we had the prerequisites that had to be incorporated in the new VIBA in a deductive fashion.

The initial work was carried out with the use of an existing system that was modelled and analysed (Case 0). During that work we tried different suggestions and techniques and performed an integrated internal and external validation process.

The following work was carried out as two development projects in collaboration with industry, referred to as Case I and II. During specification of a production planning system for mechanical engineering (Case I), and specification and realization of a material administration system for a paper mill (Case II), we tested the new method proposal in industrial settings and continuously refined the method to comply with real life demands.

The positioning in time of the cases is shown in Fig 2-5.

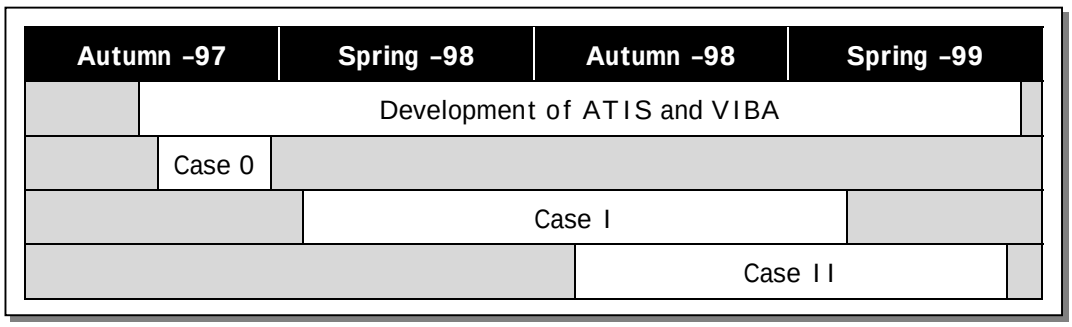



**Fig 2-5:** Time span of the different research activities.

The two cases served somewhat different purposes. When Case I was initiated we had only some vague ideas about the re-designed VIBA. The purpose was hence to test initial ideas and continuously refine VIBA and ATIS. When Case II was initiated we felt that the method was quite stable. The purpose was now to test, more thoroughly, the re-designed VIBA and to be more careful with documenting observations. Hence, whilst Case I was more of an explorative study, Case II was more of hypothesis testing. Nevertheless, both VIBA and ATIS continued to change during both Case I and II.

As indicated previously, the empirical work has produced output of two different kinds. Fig 2-6 illustrates these kinds of empirical output.

The first kind of output, which we refer to as *ideas*, represents reflections made during the empirical work that has been fed back to a continuos design process, and immediately resulted in changed 'versions' of VIBA and ATIS. The other kind of output, which we refer to as *data*, represent more systematically treated data, gained through observations

and interviews. Although ideas, by nature, are more volatile than data, some reasons underlying new ideas have been documented in logbooks and hence constitute data.

The dashed arrow from 'result' to 'design' indicates future re-design based on the results.

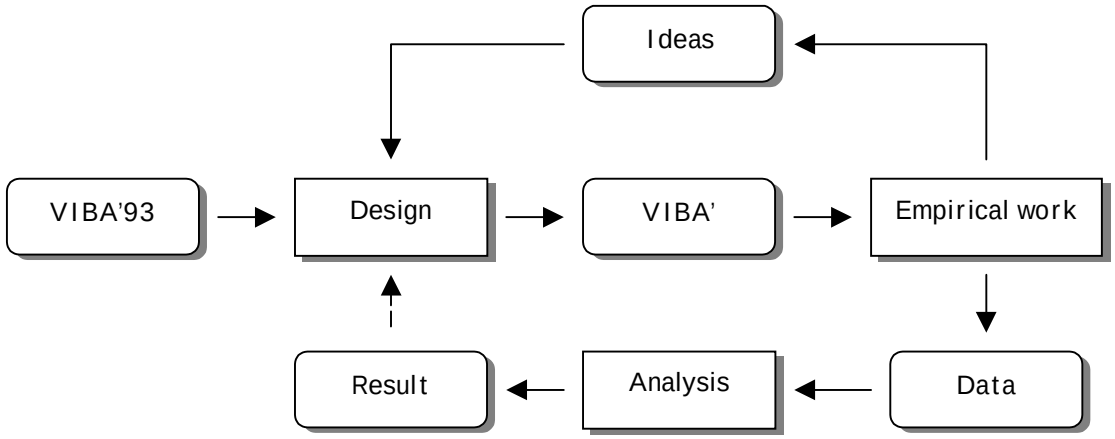



**Fig 2-6:** The Empirical work as generator of ideas and data.

Hence, the basic approach during the empirical work has been to (1) continuously improve VIBA and ATIS, and (2) collect data of different kinds (see section 2.5.2 subsequently) and do a 'formal' analysis to try to understand what worked well and what did not, and why. It is important, though, to realize that the results are not generelizable, in a traditional sense, and that more empirical work is still needed.

The empirical work has been performed according to a qualitative research tradition. The reasons for this are inherent in a combination of the aims of the studies and in research constraints in terms of allotted time and resources. Since the first case was, in essence, an explorative study, the choice between quantitative and qualitative approach was, sort of, given from the start. It was simply not possible, or at least feasible, to try to do a survey or other kind of quantitative research. We felt that we needed to try the initial ideas by ourselves, and refine them, before even trying to quantify. Hence, Case I was performed as action research in a minimal setting. How to perform the second study, when the re-design had stabilized, at least to some extent, was a more open question. However, due to practical reasons, we felt that another case study would probably be the most promising approach. A reason to favour qualitative action research was also inherent in the research problem as such. We wanted to gain experience, and draw conclusions, from working with a systems engineering method. Such methods are complex phenomena *per se*. In addition, the method was under development and continu-

ously changing. Such circumstances require closeness between the observer and the observed in order to gain nuanced data. Furthermore, the circumstances made it difficult to specify expected results in advance.

However, during Case II we wanted to also involve other people to actively work with the method proposal, in order to get, some kind of, a 'second opinion'. This led to Case II being carried out as a mixture of action research and observations. We (the researchers) created a specification of an IS which we handed over to an IT consultant firm for realization[2]. In this way we could continue to refine VIBA whilst doing the requirements elicitation and analysis. And we could test whether the consultants understood and appreciated the resulting specification. There was a big risk involved in this research design. The systems engineering had to follow a waterfall process, at least from analysis to design, which seemed to be in contradiction with our rapid development aim. Anyway, we decided to take that risk since it was also an opportunity to test if the documentation were communicable, in the extreme.

Our interest in Case I was the specification. We had not planed to study the following phases due to uncertainties as to whether financial support for realization was available. As it turned out, the project was cancelled (or at least put to rest) even before the specification was fully completed, due to this circumstance. However, we where able to collect quite a lot of data during the case and the fact that it was never completed did not effect the appropriateness of its inclusion in this study.

Due to circumstances outside of our control, the IS built during Case II was never implemented in the host organization. Otherwise, we had planned to follow up the implementation to see whether the action orientation actually became visible in the final product. This important issue consequently remains to be studied further (cf. chapter 9).

In the following sections we will present the two cases and discuss how data collection and analysis have been performed.

### 2.5.1 About the Cases

This subsection gives a brief description of the two cases. The purpose of the descriptions is to give a background, and a 'feeling' for the systems engineering efforts and their contexts.

#### *2.5.1.1 Case I: Production Planning in Mechanical Engineering*

The first case study was performed at AB Örebro Mekaniska Verkstad – a Swedish mechanical engineering firm supplying parts to other companies (i.e. contract manufacturing). The firm is a small company with

[2] By 'realization' we refer to the design, construction, and implementation of an IS.

about 15 employees and no formal organization chart. Instead the managing director, who is also the owner, handles both sales, production planning, and all other managerial tasks. As it turned out, the MD, whose workload was continuously increasing, felt that some computerized tool was needed to ease the burden. The main problem was to match production schedules with customer demands and to reach higher precision in predicting when a particular product could be delivered. The goal was to create an information system that kept track of production schedules, and that could be used interactively to calculate delivery dates while still talking to a customer. At that time, this was impossible due to many circumstances. First, the MD was not always updated on the actual state of the production process. Second, everything needed to calculate production times had to be memorized and calculations were performed manually. Some information was not even possible to retrieve instantly. Instead the MD had to ask some shop-floor workers about the current capabilities. The system thus aimed to cover the sales process from negotiation, via order, to fulfilment and delivery. Hence it was a combined 'sales support system' and a 'production planning system'. Furthermore, it was planned to replace current systems for 'customer administration' and 'deliverance administration'. It was also supposed to co-operate with the administrative systems used for traditional bookkeeping. However, the production planning was at the heart of the planned system, and the key to the success of its other parts.

#### *2.5.1.2 Case II: Material Administration in Paper Production*

The second case study was performed at Duni Paper Mills Kisa – a paper mill that produces paper tissue from pulp; as well as different laminate products, of which tissue is an important ingredient.

The IS to be developed was a quite 'standard' material administration system. The goal was to keep track of products and product categories, raw material, warehousing, suppliers, supplier orders, and deliveries of raw material. One part of the system was aimed at helping the production planner to create production schedules. Another part was aimed at helping the buyer to create supplier orders, based on the need implied by the scheduled production. A third part was aimed for shop-floor workers to register deliveries and raw material usage. At that time, the buying and planning activities were supported by stand-alone spreadsheet and document processing applications. The warehousing was performed and maintained manually. However, much of the data needed for the new IS was already contained in different databases used for statistical and quality assurance purposes. So, there was a need to

integrate the new system with a host of legacy applications and data bases.

### 2.5.2 Data collection

This section describes how the cases were selected and how data was recorded during the work.

#### *2.5.2.1 Selection of Cases*

Both cases were selected 'opportunistically' in the sense that they were both 'stumbled across' and, after evaluation, seemed to match our purposes. Another approach might have been to actively search for cases, but we did not feel that there was any need for that. Both cases represented the kind of problem domains that we were addressing: business supporting information systems where communication problems where obvious, and where a business process-oriented approach was straightforward.

#### *2.5.2.2 Records*

The collection and recording of data was focused primarily on positive and negative aspects of the systems engineering work and of the forms of documentation suggested by the re-designed VIBA.

The sources of data, i.e. the (roles of the) actors participating in the cases, are summarized in Table 2-1.

| Role | Case | Referred to as |
|---|---|---|
| Researcher, requirements engineer | I and II | R1 |
| Researcher, requirements engineer | Part of I | R2 |
| Researcher, requirements engineer | II | R3 |
| Systems co-ordinator | II | SC |
| IT Consultant, Project leader and designer | II | C1 |
| IT Consultant, Designer and programmer | II | C2 |

**Table 2-1:** Actors providing data and ideas.

The studies can be characterized as combinations of *action research* and *formative evaluation*, using the terminology of Patton (1990). In this terminology action research is focused on solving some problems within an organization and formative evaluations are used to improve, for example, a program or product. Hence, the cases led to results on two levels: information systems (specifications) for the participating organizations, and improved method and theory as primary research result.

Data collection was performed throughout the case studies and recorded in research logs. Logs were also created by the IT consultants

(C1 and C2) during their realization work in Case II. The system coordinator at Duni (SC) also documented his experiences during the realization in a log. In addition to the logs, C1 and C2 were asked to fill out forms with comments during their work. The forms had pre-specified columns addressing positive and negative aspects of each type of document in the specification. Furthermore the forms contained two pages addressing 'missing information' and 'non-useful information' in the specification that they might occasionally find during the work.

A total of four interviews have been conducted, all during Case II. The first two interviews were performed with the IT consultants after approximately 50% of the realization had been performed. As suggested by Patton (1990), the interviews followed an interview guide, specifying the topics and issues to be penetrated; see Fig 2-7.

| **General** | Opinion | How did that feel | Evolution over time |
|---|---|---|---|
| Work tasks | | | |
| Overall judgement | | | |
| Understanding of the business | | | |
| Understanding of the System | | | |
| Guiding/freedom | | | |

| **Documentation** | How used | Why | Positive | Negative |
|---|---|---|---|---|
| Action diagrams | | | | |
| Document definitions | | | | |
| ISP List | | | | |
| I-Tables | | | | |
| Prototypes | | | | |
| Sequence restrictions | | | | |
| Class diagrams | | | | |
| Class definitions | | | | |

| **Other documentation** | How would it have been used | Why | How did instead | Opinion (what to do) |
|---|---|---|---|---|
| …filled out during interviews… | | | | |

**Fig 2-7:** Interview guide used during I1 and I2.

The interview guide was divided into three parts where each part was formed as a matrix of headings (y-axis) and dimensions (x-axis) from which to discuss the topic suggested by the heading. During the interviews each 'cell' was marked when its area had been discussed.

The two last parts of the interview guide mirrored the forms discussed previously, but allowed for greater flexibility. The forms represent a standard open-ended inquiry whilst the interviews allowed for

more reflection and follow-up questions. In accordance with the recommendations of Patton (1990) the interview guide was used as a check-list and did not direct the ordering of questions, *et cetera*. Instead, the interviews where performed more like conversations, allowing for different twists and turns to occur. These first two interviews were recorded on tape.

The third and fourth interviews were performed as unstructured open-ended interviews. That is, no interview guides where used, and the respondents got to speak freely and reflect upon the work performed. During these interviews notes were taken.

A summary of the different records is shown in Table 2-2.

| What | Source | When and in what form |
|---|---|---|
| Research log 1 (RL1) | R1 | Case I. Written chronologically. |
| Research log 2 (RL2) | R1 | Case II, specification. Written, chronologically. |
| Research log 3 (RL3) | R1 | Case II, realization. Written, chronologically. |
| Consultant log 1 (CL1) | C1 | Case II, realization. Written, chronologically. |
| Consultant log 2 (CL2) | C2 | Case II, realization. Written, chronologically. |
| Interview 1 (I1) | C1 | Case II. Transcription of taped interview performed when approximately 50% of the realization was completed. |
| Interview 2 (I2) | C2 | Case II. Transcription of taped interview performed when approximately 50% of the realization was completed. |
| Written comments 1 (WC1) | C1 | Case II. Submitted during I1, Pre-structured form. |
| Written comments 2 (WC2) | C2 | Case II. Submitted during I2, Pre-structured form. |
| Interview 3 (I3) | C1 and C2 | Case II. Written notes from interview performed after the realization. |
| Interview 4 (I4) | SC | Case II. Written notes from interview performed after the realization. |
| System owner log (SL) | SC | Case II, the realization part. Submitted after I4. Written, chronologically. |

**Table 2-2:** Summary of the collected data and its origin.

### 2.5.3 Data Analysis

Analysis of qualitative data can be performed in several different ways. At first we planned to use Grounded theory (Straus & Corbin, 1998) as a source for structuring the work. However, after some initial analysis (open coding and part of axial coding) we started to realize that Grounded theory was not an optimal approach to our case. The reason

was that the analysis tended to be to broad[3]. One assumption of Grounded theory is that the analyst should be as open-minded as possible and not pre-categorize. Instead categories are to be built up, and related to each other, based on empirical data. Our interest in the data was thus too focused to actually fit within the Grounded theory framework. In fact, much of the data was inherently pre-categorized and semi-structured, such as the forms with positive and negative aspects of document types, and the interview guides. Therefore we decided to favour an approach that made use of the structures already created. This was following Patton (1990, p. 376), who states: '*...answers from different people can be grouped by topics from the guide ...The interview guide actually constitutes a descriptive analytical framework for analysis.*'. When the data had been structured in this way, we tried to find patterns in the answers and see if there were generelizable and reoccurring ideas. It is important to observe that in the analysis we treat our own experiences during the action research as data. This implies a great risk of bias towards our own opinions and beliefs. This is, however, a risk in all interpretative studies, and we have kept the sources separated throughout analysis in order to make the results traceable.

It is important to be aware that after the specification was handed over to the consultants in Case II, as well as after the interviews where performed, some rounds in the 'ideas loop' of Fig 2-6 (page 35) have been performed, partly as a consequence of the fact that Case I was still ongoing. Actually we have had three major externalized versions of VIBA: the original VIBA (VIBA'93), the version that was manifested in the documentation handed over to the consultants in Case II (VIBA'98), and the version reported in this thesis (VIBA'99); see Table 2-3.

| | |
|---|---|
| VIBA'93 | Point of departure |
| VIBA'98 | Basis for Case II |
| VIBA'99 | Reported in this thesis |

**Table 2-3:** The different externalized versions of VIBA.

In essence, this means that the results of the analysis do not refer directly to the latest versions of VIBA and ATIS (as presented in chapters 3 to 7). As we shall se in chapter 8, some 'modifications' suggested by the analysis had already been performed.

Due to the transient state of VIBA when we entered Case I, we do not refer to any particular externalized version as the basis for that case.

[3] Broad in the sense of 'unfocused'. Of course, interesting information was revealed, but much of that information was outside of the scope of this thesis.

Instead, VIBA'93, modified with ideas from Case 0 and Case I, forms VIBA'98. As shown in Table 2-2, data collected during Case I have been used in the analysis reported in chapter 8.

Differences between VIBA'93 and VIBA'99 are discussed in chapters 5 to 7. Differences between VIBA'98 and VIBA'99 are discussed in chapter 8.

# Chapter 3

# Action and Communication as Theoretical Foundation

*What I shall have to say here is neither difficult nor contentious;*
*the only merit I should like to claim for it is that of being true, at least in parts.*

John L. Austin

This chapter presents and elaborates on a theory of information systems based on a language action perspective of organizations and information. The aim is to establish a baseline from which to elaborate further in chapter 4. The chapter is divided into four sections. The first section (3.1) establishes some preliminaries regarding information and messages from a language action perspective. The second section (3.2) discusses organizations and performance of business as a special case of language action. The third section (3.3) discusses and defines the concept of information system as being a language action system for performing business. Finally, the fourth section (3.4) brings up some criticisms against using language action as foundation for the design of information technology.

## 3.1 Doing and Speaking

Many approaches and theories within the field of information systems are based on a strict representational view of information. With such a view, a main purpose of requirements elicitation and analysis is to get an accurate 'image' of the problem domain in order to have that piece of reality properly represented in the future (data base of the) system. Such a strict representational view seems to be an instance of what Austin (1962) calls a 'descriptive fallacy' and can be challenged in several ways. Admittedly, one obvious application of language use is to describe things; i.e. uttering statements about the nature and state of the world. However, language is also used for many other purposes. Language is used to command, request, question, promise, warn, assign, *et cetera* (Austin, 1962; Searle, 1969; Habermas, 1984). A descriptive fallacy is thus said to occur when language is considered as used *only* for descrip-

tive purposes, which seems to be the case with the strict representational view of information; cf. Holm (1996) on '*the technological version of the descriptive fallacy*'. Just as language, information systems are used for other purposes than pure description. From this language action point of view, information systems should not be considered as mere 'fact repositories'. Information systems should instead be seen as vehicles for communication among people and organizations.

This thesis takes the standpoint of language action theory, which emanates from speech act theory (Austin, 1962; Searle, 1969; 1979) and communicative action theory (Habermas, 1984). Such an approach emphasizes that people *act* when communicating. This theoretical approach has a growing impact on the IS community. Flores & Ludlow (1980) and Goldkuhl & Lyytinen (1982) made early contributions and Winograd & Flores (1986) performed seminal work. One possible conclusion is that made by Whitaker (1992): '*The greatest impact of alternative linguistic models on IT has been that of Austin's theory, as elaborated by Searle and evangelized by Winograd & Flores*'.

The main manifesto of the Language Action Perspective is that an utterance (message) is a combination of a propositional (informational) content and an illocutionary force (an action mode). The propositional content is what is talked about and the illocutionary force means what kind of action is performed (Searle, 1969). The illocution used is a result of the intention behind the communication. When we communicate, we formulate propositional contents and embed these in communicative action types.

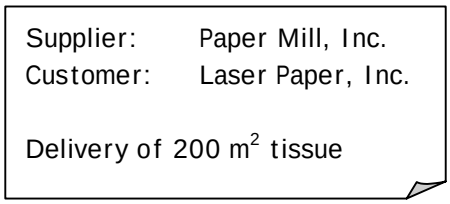



**Fig 3-1:** Example of a propositional content, which can be embedded in different types of communicative acts.

A propositional content, as described in Fig 3-1, can be used in many different utterances (messages). It can be used in an offer, an order, a promise, a contract, a prediction, a question and a report and also possibly other types of communicative acts. Several attempts to classify speech acts with respect to illocutionary force[1] have been made (e.g.

[1] Actually with respect to illocutionary point, i.e. the purpose of the illocution, which is only one component of the illocutionary force. Other components are: the *degree of strength* of the illocutionary point; *propositional content conditions* imposed by the

Searle, 1979; Habermas, 1984). Such classification is not the key point in our adoption of speech act theory since we do not want to restrict our model to a predefined classification, because (1) in our opinion a single message might have several communication functions[2] and (2) we want to allow for particular professional business languages to be used when naming the classes. However, Table 3-1 shows some examples of how the propositional content of Fig 3-1 can be embedded in different types of illocutionary acts, i.e. with a different action mode. In the examples, the same 'things' are referenced but the meanings of the utterances are quite different.

| **Action mode** | **Example utterance** |
|---|---|
| Offer | Paper mill, Inc. offers to deliver 200 $m^2$ of tissue to Laser Paper, Inc. |
| Order | Laser Paper, Inc., directs Paper mill, Inc. To deliver 200 $m^2$ of tissue. |
| Promise | Paper mill, Inc. will deliver 200 $m^2$ of tissue to Laser Paper, Inc. |
| Contract | Laser Paper, Inc. and Paper mill, Inc. agree on the delivery of 200 $m^2$ of tissue. |

**Table 3-1:** Examples of different action modes.

A message[3] is communicated in some context by some actor playing some role and is interpreted by some (possibly another) actor playing (possibly another) role. Fig 3-2 illustrates this discussion. Note that chapter 4 offers a more careful examination of these topics.

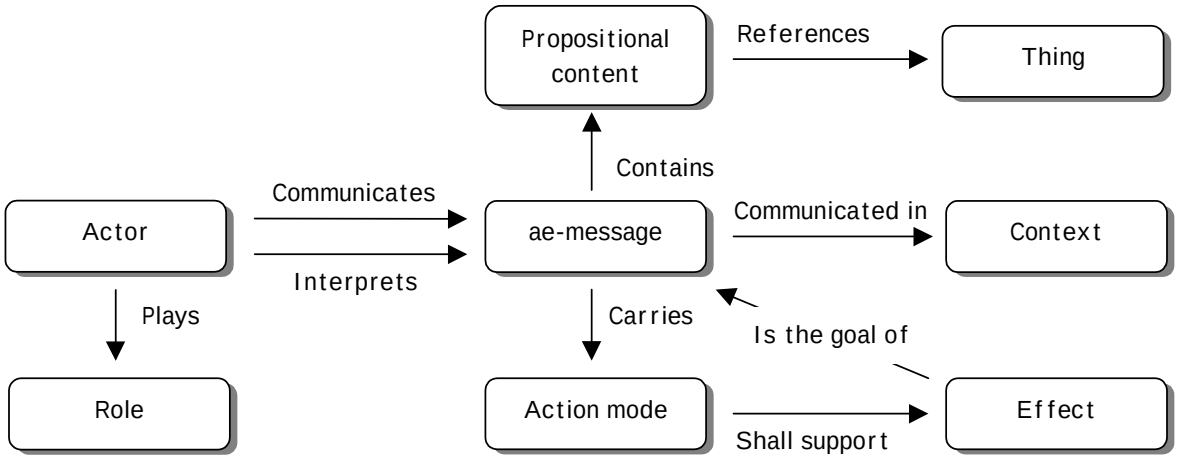



**Fig 3-2:** The concept of ae-message in communicative acts.

type of illocutionary force; and *preparatory conditions* that represent presuppositions for certain type of speech acts.

[2] Hence, the action mode of a message might embed several communication functions.

[3] A message is, in this context, referred to as an action elementary message, or ae-message for short. Please see chapter 4 for a more careful examination of the concept of ae-message and its relation to 'traditional' e-messages.

The references to concepts of Fig 3-2 represent what the message concerns, i.e. the propositional content. The action mode (illocutionary point) is determined by the message's (and thus its communicators) intention and hence intended effects.

In Fig 3-3 the concept of actor together with the concept of information system has been generalized (super-classed) to a concept of performer. A performer is something that is capable of performing action. However, the distinction between actors and information systems is important when communication is performed through information systems, which we will discuss further below.

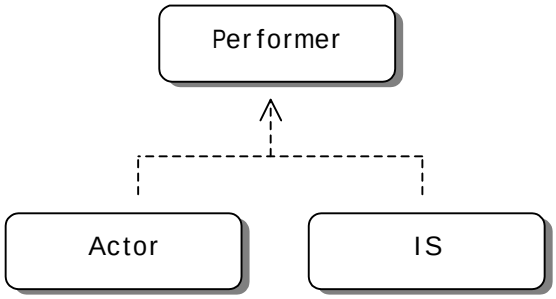



**Fig 3-3:** Performers, actors and information systems.

Our concept of performer is similar to what Graham (1998) refers to as 'agents' and what Falkenberg *et al.* (1998) and Jacobson *et al.* (1995) refer to as 'actors', which is also the UML-term. We do however believe that the term 'agent' should be used exclusively when talking about performers acting on someone else's behalf, which is certainly not the case with all performers.

### 3.1.1 Messages and references

A message might concern a performer, another message, an earlier action, or some other entity in the world. It is however important to distinguish between real world phenomena and the inter-subjective conceptualizations of such phenomena used when talking about, and within, the world. As showed in Fig 3-4, concepts are thought of as such conceptualizations of real world phenomena (things).

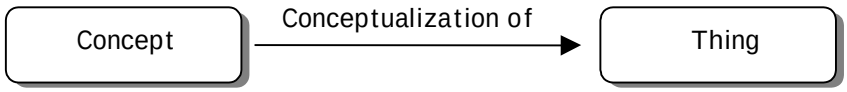



**Fig 3-4:** Concepts are conceptualizations of real world phenomena.

Whenever a message's propositional content references, for example, an earlier action, it is not the action *per se* (the thing) that is referred to, but a conceptualization of that action. That is, it is a limited subset of the properties of the action, which then constitutes a concept, that is referenced. Concepts are referenced by expressions (or terms) that form the propositional content. This view corresponds to the so-called Ogden's triangle (e.g. Falkenberg *et al.*, 1998) as shown in Fig 3-5. A concept, for example a person, is a conceptualization of the real-world phenomenon of a human being. Another conceptualization of the same phenomenon might be, for example, a human resource.

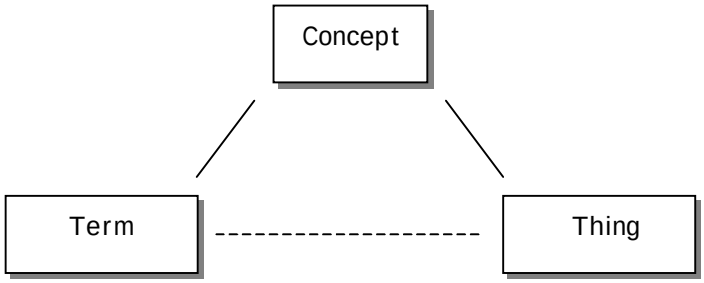



**Fig 3-5:** Our version of Ogden's triangle.

The same concept might be referenced by different terms (or expressions). For example 'Peter's boss' and 'the head of department A' might reference the same concept. Depending on the context, these two terms might very well refer to different conceptualizations of the same real world phenomena.

Hence, a propositional content refers to things indirectly via a conceptualization of the things.

### 3.1.2 Types and instances

When modelling a business, or any other system, it is inconvenient to always enumerate all single instances of a particular phenomenon. In business modelling it would actually be impossible since, for example, all actual actions performed might be infinite and possibly uncountable during design. Hence, there is a need to introduce a distinction between types and instances. This is the kind of distinction made, for example, between a class and the instances (objects) of that class in object-oriented analysis. Hence, any 'concept' is an instance of a 'concept type'[4].

[4] From a philosophical point of view 'concept types' are actually conceptualizations of 'phenomena types', which are phenomena in their own rights. Therefore a 'concept type' is actually a 'concept' as well. We will, however, not dig into such 'academic' discussions in this thesis.

Likewise, 'messages' are instances of 'message types', and 'actions' are instances of 'action types'. See Fig 3-6 for an illustration[5].

So far so good. When it comes to 'performers', however, we say that they play a professional 'role' within the business. A 'role' might thus be viewed as a 'type of performer' but this does not conform to the traditional use of the type/instance distinction. The non-conformance is that the same performer might play different roles at different points in time whilst the relationship established by traditional instantiation is static.

Hence, a performer that performs an action plays the role dictated by the type of that action. Similarly, the propositional content of the resulting message refers to instances of entities whose types are dictated by references to 'concept types' in the 'message type' of the message.

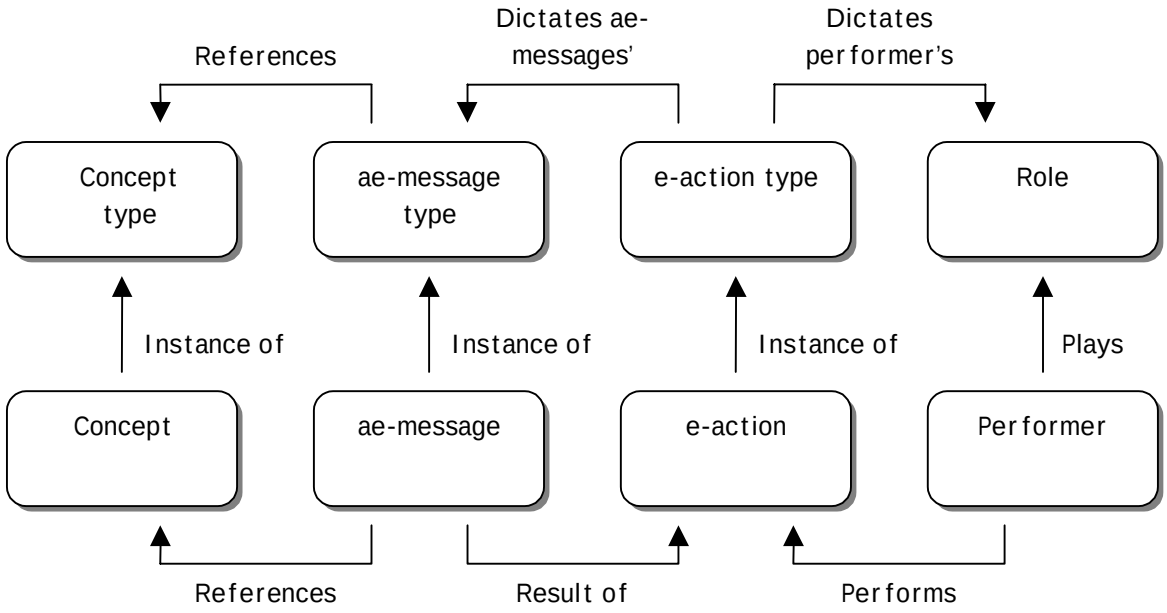



**Fig 3-6:** Types and Instances.

### 3.1.3 Language Acts and Other Action

We have so far argued that language use, in the form of utterances, can (and should) be regarded as action. This is, however, not the only way communication is performed. The concept of 'speech act' can be regarded as a special case of the more general concept of 'semiotic act' (Graham, 1998). When two performers communicate, they use some inter-subjective conventions of signs and signals. Such signals between per-

[5] In the figure the terms 'e-action' and 'ae-message' are used instead of 'action' and 'message'. These concepts are examined more carefully in chapter 4. For now, they may be labeled 'action' and 'message'.

formers are called semiotic acts and are always carried by some material substratum. If the substratum is verbal or written then the semiotic act is a speech act (*ibid.*). This implies that also other substrata might be used to form communication, for example gestures and other forms of 'body language' are ways to communicate. The key point is that not just speech acts but all semiotic acts share the properties discussed above; i.e. they are manifested by messages having propositional contents and an action modes. This is the reason why we prefer to talk about communicative acts rather than speech acts, as in Fig 3-2.

## 3.2 Organizations as Action

An enterprise carrying on business buys and sells; i.e. it is an actor in the market place. The concept of actor of Fig 3-3 can therefore be further specialized into human actors and organizations, as shown in Fig 3-7.

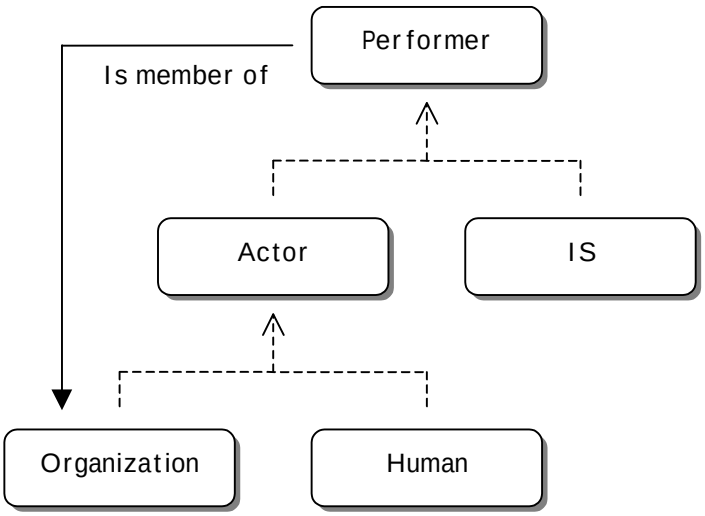



**Fig 3-7:** A hierarchy of performers.

An organization will usually act in different (meta-) roles: as a supplier trying to sell its products, and as a customer purchasing products in order to create conditions for selling. Organizations can be viewed from a language action perspective. Business interaction means besides exchange of value also business communication. Proposals are given from customers and suppliers. These proposals are evaluated and possibly transformed into orders and contracts, which are mutual commitments. A business transaction means an exchange of value, i.e. a product (goods or services) is delivered and payment is made in return. Business communication means exchange of proposals and commitments in a conversation between customer and supplier.

It is important to notice that an organization must always have at least one human actor member, who is responsible for, and performs the actions of, the organization.

### 3.2.1 Generic Business Frameworks

Generic business frameworks are general descriptions (conceptions) of business activities and business communication. Such frameworks can be used as templates to direct analysts' attention to certain (important) aspects when modelling a business. The probably most well-known generic business framework is the Action Workflow approach (Action Technologies, 1993; Denning & Medina-Mora, 1995). A more thorough model, though, is the more recent Business Action Theory (BAT) (Goldkuhl, 1998). These two frameworks build explicitly on the Language Action Perspective and hence they help us to emphasize the actions performed by, and within, organizations.

Generic business frameworks differ from traditional modelling techniques such as data flow diagramming (Yourdon, 1989) and use case analysis (Jacobson *et al.*, 1992; 1995). The difference is that traditional modelling techniques are, what could be called, notation driven. That is, they provide a set of diagramming techniques with associated syntactic and semantic conventions (rules) to describe business processes. Generic business frameworks, on the other hand, are theory driven. In addition to providing modelling techniques they also provide a theory of business activity, and hence enable a more critical and reflective way of working with business modelling. Usually, generic business frameworks have more restricted syntax of their notations and allow for different notation techniques to be used.

#### *3.2.1.1 The Action Workflow Approach*

The Action Workflow approach (Action Technologies, 1993; Denning & Medina-Mora, 1995) can be used for description of business interaction between a customer and a supplier. The underlying idea is that whenever a business activity is performed, in order to satisfy a customer, a generic pattern of speech acts occurs.

This generic pattern (or schema) of 'conversation for action' has been described by Winograd & Flores (1986) in terms of states and transitions (see Fig 3-8) where each transition corresponds to a speech act and each state corresponds to a state of the conversation.

According to the conversation for action schema, a business conversation is initiated by a request from a customer A (the initial speaker), which specifies some conditions of satisfaction. The supplier B (the initial hearer) then has the choice of accepting the conditions

(promise to satisfy the request), rejecting it, or making a counteroffer. If and when the parties have agreed the supplier eventually asserts that the conditions of satisfaction have been met (reporting completion). Now the customer can either declare that, in his or her opinion, the conditions have not been met, or declare satisfaction (which would end the conversation happily). During the conversation both the supplier and the customer can withdraw at any point and thus cancel the conversation sequence (in an unhappy manner).

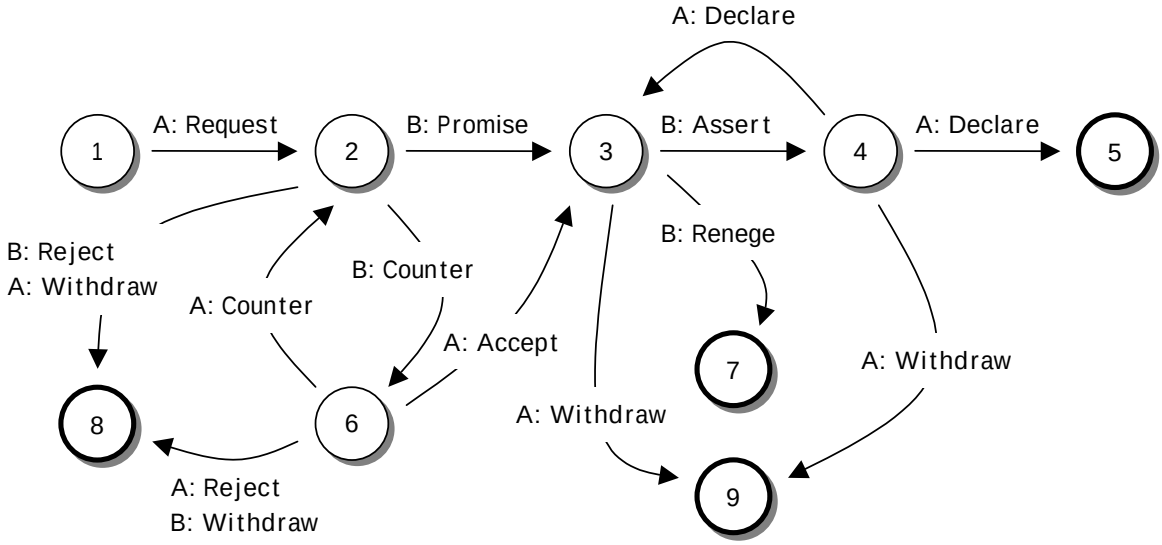



**Fig 3-8:** The basic conversation for action (Winograd & Flores, 1986).

Building on the generic speech act pattern of the conversation for action, the Action Workflow approach describes business interaction as consisting of four phases: (1) preparation, (2) negotiation, (3) performance, and (4) acceptance. As before the roles are pre-defined as customer and performer. These phases and roles are described by the so-called Action Workflow loop; see Fig 3-9.

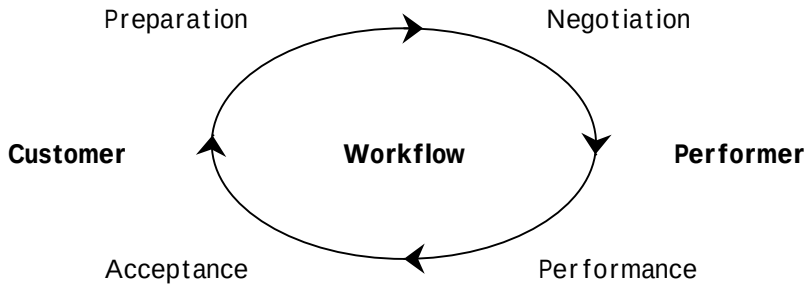



**Fig 3-9:** The Action Workflow loop (Action Technologies, 1993).

The Action Workflow loop can thus be regarded as a generic conception of the structure of business activity used to direct analysts' attention to the action character of doing business.

#### *3.2.1.2 Business Action Theory*

Similar to the conversation for action schema and the Action Workflow loop, the generic business model of Business Action Theory (BAT) (Goldkuhl, 1998) shows the inherent business logic of two parties doing business together. The model describes performance of business based on generic business actions of communicative as well as material character (*ibid*.). Hence, BAT differs from Action Workflow in that language action as well as material action are given equal emphasis. BAT also emphasizes the mutual co-ordination inherent in business exchanges: it admits and attempts to clarify common, as well as conflicting, interests (*ibid*.).

According to BAT, doing business means an exchange of value between two parties. Thus, an 'exchange relationship' is established between a customer and a supplier. The generic business actions, and hence the exchange of value, is performed in six phases: (1) Business prerequisites phase, (2) Exposure and contact search phase, (3) Contact establishment and proposal phase, (4) Contractual phase (order and delivery promise), (5) Fulfilment phase (delivery and payment), and (6) Completion phase (acceptance or dissatisfaction, with possible claims). The model is illustrated by Fig 3-10.

The first phase (the Business prerequisites phase) means the establishment of prerequisites (of both supplier and customer) for the subsequent interactive business performance. The supplier is supposed to have a certain 'supplying potential' in terms of knowledge, capacity, and a range of goods and/or services. In turn, the customer has needs and demands that match the potential of the supplier.

The other five phases mean exchanges of a different character. These exchanges imply actions directed towards the other party, i.e. exchange of (2) interests, (3) proposals, (4) commitments, (5) value, and (6) acceptances or claims. Phases two and three together form a business interest stage.

During the second phase (the Exposure and contact search phase) both parties are searching for contact. The supplier exposes its supplying potential to the market and the customers' needs and demands give rise to desires and wishes that direct the search for products and suppliers.

When the supplier and the customer eventually make contact, a negotiation is possibly carried out. This is where the third phase starts (the Contact establishment and proposal phase). Offers and counterof-

fers are communicated while the customer's desires and the supplier's potential are made more explicit.

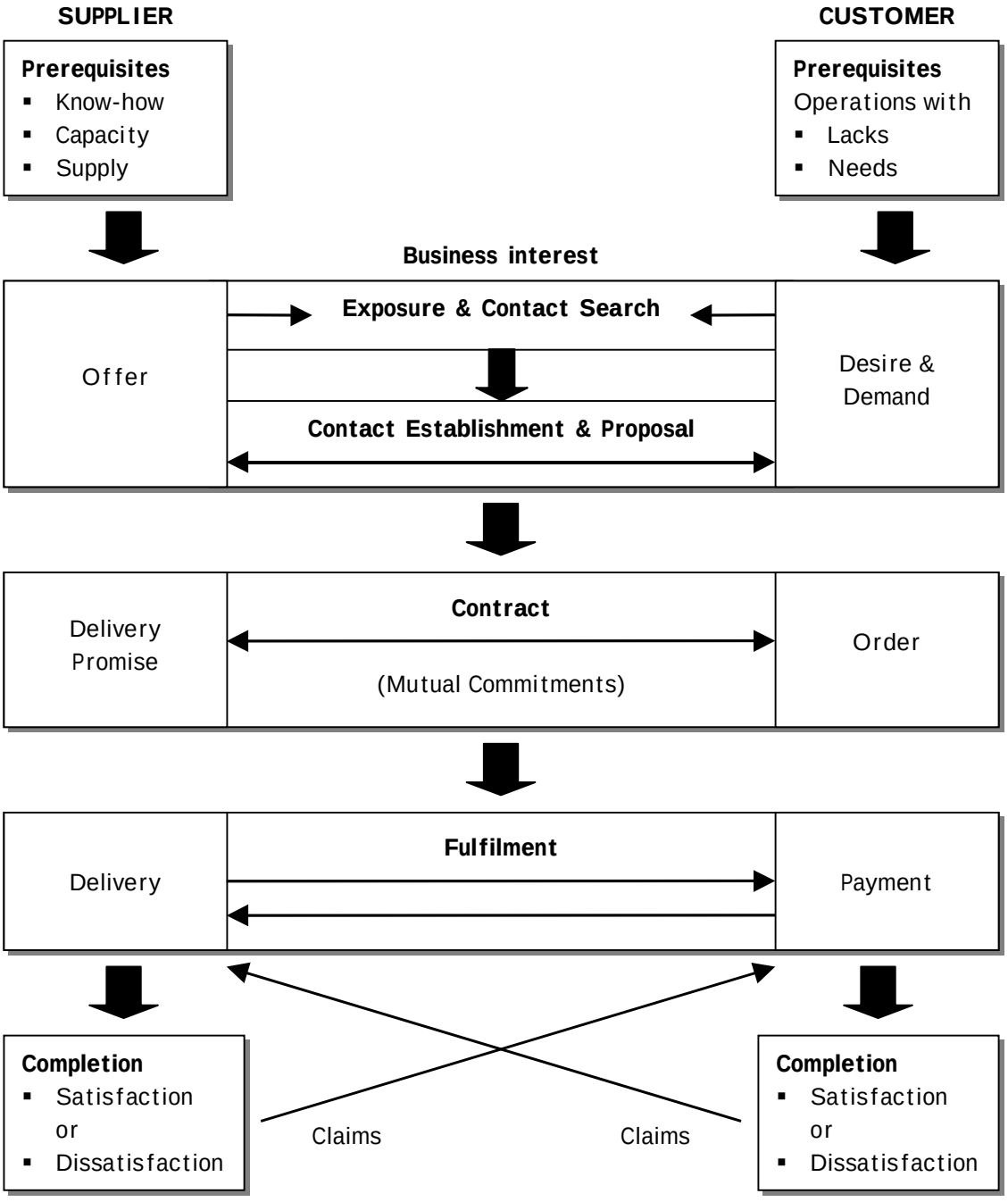



**Fig 3-10:** Business Action Theory: A phase model (Goldkuhl, 1998).

The negotiations might eventually end up in a contract, and the business interaction enters the fourth phase (the Contractual phase). The key word within this phase is *agreement* – the mutual agreement to perform the business transaction. The contract is a mutual commitment

that dictates future actions to be taken by the parties. It is a promise of delivery from the supplier and a promise of payment from the customer.

The fifth phase (the Fulfilment phase) means fulfilment of the promises stated during the contractual phase, including delivery and payment.

The sixth and final phase (the Completion phase) means that both parties follow up and evaluate the business transaction, which leads to satisfaction or dissatisfaction for each party. During this phase both customer and supplier might make complaints. The client might, for example, claim that the products delivered did not satisfy the agreement, and the supplier might have a different opinion than the customer regarding payment details.

As we can see, the BAT model differs from (and thereby extends) the Action Workflow approach in at least three important respects.

First, it puts equal emphasis on both language action and material action. The latter is left out of the conversation for action schema, which simply states that the supplier shall assert that the request has been satisfied. The argument by Winograd & Flores (1986, p. 66) that '*The actual doing of whatever is needed to meet the conditions of satisfaction lies outside of the conversation*', does not seem convincing in this respect. Since the 'doing' when doing business consists of a mixture of material and communicative acts, both should be considered in a generic business framework. As stated previously, sometimes the material acts implies (or equals for that matter) communicative acts. Therefore, directing attention to material acts as well, increases the possibility that no important communicative acts are omitted when the framework is applied.

Second, it ranges over a larger part of the business process than the conversation for action schema does. The latter basically starts in phase three of the BAT model and thus misses the important activities performed before the actual business negotiation starts. In this respect leaving the pre-sales and marketing activities outside the business interaction.

Third, following Searle's (1979) original classification of speech acts, Action Workflow assumes that all language acts can be identified with one, and only one, illocutionary point (action mode). However, as recognized by BAT, some acts have more than one function. An offering is, for example, both a presentation of the suppliers potential and a promise to actually deliver the products offered and/or services in accordance with the offer.

### 3.2.2 Concluding Remarks

To sum up, organizations are business actors performing business action. A business action involves to a large extent communicative action directed towards other organizations (i.e. towards customers and suppliers). Such communication is not mere information transfer. It can include intentional influence and the making of commitments. In a business setting, the recipient (interpreter) of a 'business message' must trust what is said (and thus communicatively done) by a speaker (communicator), cf. Habermas (1984).

## 3.3 Information Systems as Action

Information systems usage in a business firm must be understood within a framework of the organization as a business actor. Every information system should be possible to relate to one or several of the six phases of business interaction (described above). By the *business value imperative* (Porter, 1985) and advocates of BPR, e.g. Hammer & Champy (1993), it is understood that all the activities of a business should directly or indirectly contribute to the value for the customer. An information system should therefore include activities that contribute to the creation of such value.

An IS can be used directly in a business interaction. Information systems might be used for:

- Exposing the capacity and products of the firm.
- Making offers.
- Giving delivery promises.
- Governing the deliveries.
- Invoicing.
- Reclaiming payments.

In such situations, the firm is using information systems to perform parts of its business action. An information system can also be used for other purposes than direct business interaction. It can be used for background activities supporting the direct business interaction.

This language action view of information systems emphasizes that such system are more than 'information containers'. An information system is not restricted only to holding information about its environment, but is used to perform communicative acts important to business.

From our point of view an information system is interpreted to be an 'action system'. This means that the system is capable of performing actions. It is important to add that we do not presuppose computers to have human properties of consciousness and ethical responsibility. In-

formation systems perform actions that are predefined by human actors. Information systems do not find out what kind of actions to perform. This is always predefined to the system. We do not blame a computer for errors made. We blame those humans who have given the system faulty instructions.

This view helps us to see information systems as information agents in an organization (Verharen, 1997; Graham, 1998). There can be different agents acting and interacting communicatively in an organization, i.e. human agents and IT artefacts (information systems). Both information systems and human actors can thus perform actions on behalf of the organisation of which they are members.

Thus, there are two levels of actions involved in information systems (1) development, and (2) usage. There are (1) *regulative actions* performed by systems engineers when defining a discourse for the (2) *interactive actions*, *automatic actions*, and *consequential actions* performed by the information system's users. This view of information systems as action is visualized in Fig 3-11.

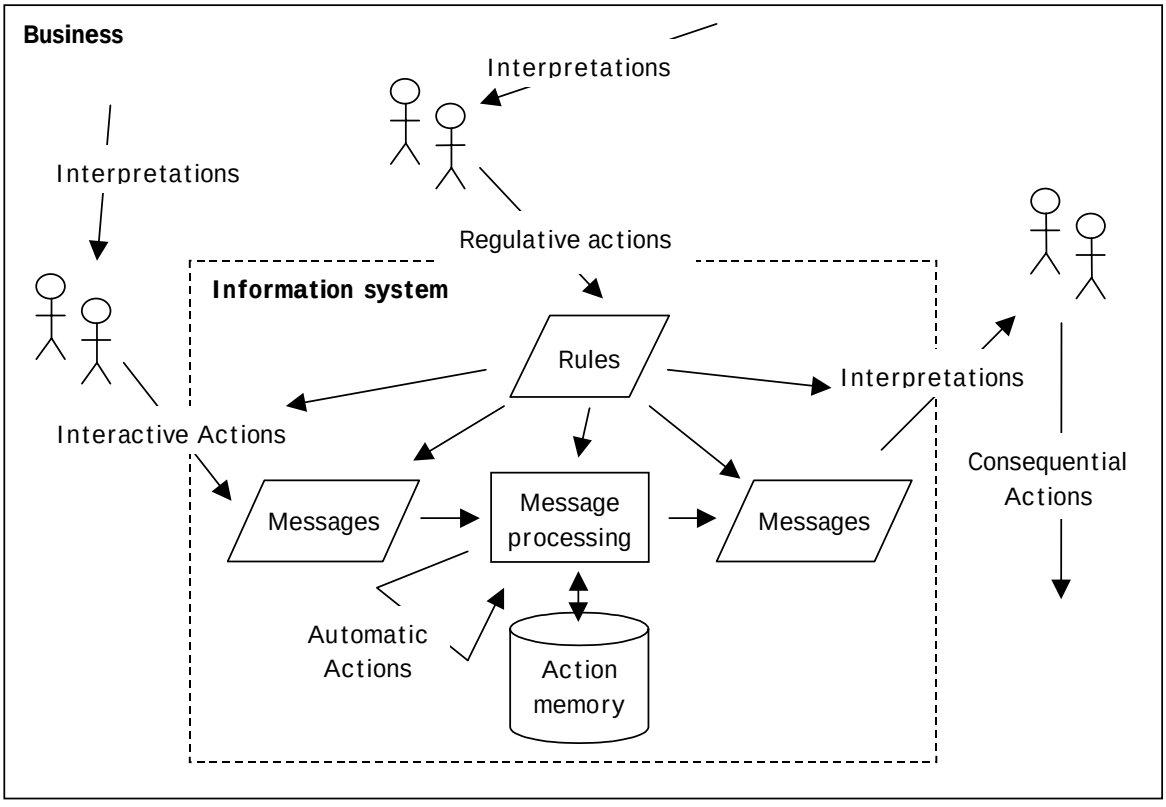



**Fig 3-11:** Information system as action.

An information system has thus possibilities to perform communicative actions, which are governed by specified rules. Those rules can be

interpreted as the *action potential* of the system. An IS has an *action repertoire* predefined by such action rules. When performing IS actions it is many times necessary to have access to other earlier actions. For example when checking an incoming delivery, it should be possible to access the corresponding order. An information system must hold a *memory of actions* already performed, and other important prerequisites for action.

As a system for communicative action it will of course hold a vocabulary of the action topic. Different concepts and their associated terminology are part of the system's action potential. This means a conceptualization of what is talked and talked about.

An information system has a dual action character. It can both (1) be an instrument (tool) for users to perform action and (2) perform action independently of its users (but of course not independently of its predefined rules). A user (for example an order clerk) can put an order assisted by an order information system (1). An IS can also perform actions by itself, for example checking for delivery possibilities (2).

We refer to these two types of IS usage as *interactive usage situations* and *automatic usage situations*. In situation (1) as well as (2) the information system can (should) be furnished with an ability to keep record of communicative actions performed (in its action memory) and be able to use them for other purposes later on. This kind of indirect IS usage is referred to as *consequential usage situations*. Fig 3-12 exemplifies these kinds of usage situations in an Action diagram (the notation used in Action diagrams is described in detail in chapter 6).[6]

This language action view of information systems may have several functions. It can serve as a new and fresh way to *conceive* information systems, i.e. a way to interpret 'information system'. It can also be used to *evaluate* existing information systems. Moreover, this view can be used to *design* new information systems and to *redesign* existing ones. We do not propose that existing information systems (which are not designed according to this paradigm) do not involve any IS action. They do, but perhaps in an obscure way. One major problem of many existing information systems is probably that their action character is not visible enough for their users, i.e. the actions of the system may be implicit. Hence, old information systems might be redesigned to be more action transparent.

[6] Admittedly, the usage situations of Fig 3-12 are idealized and these three kinds are combined in most practical cases.

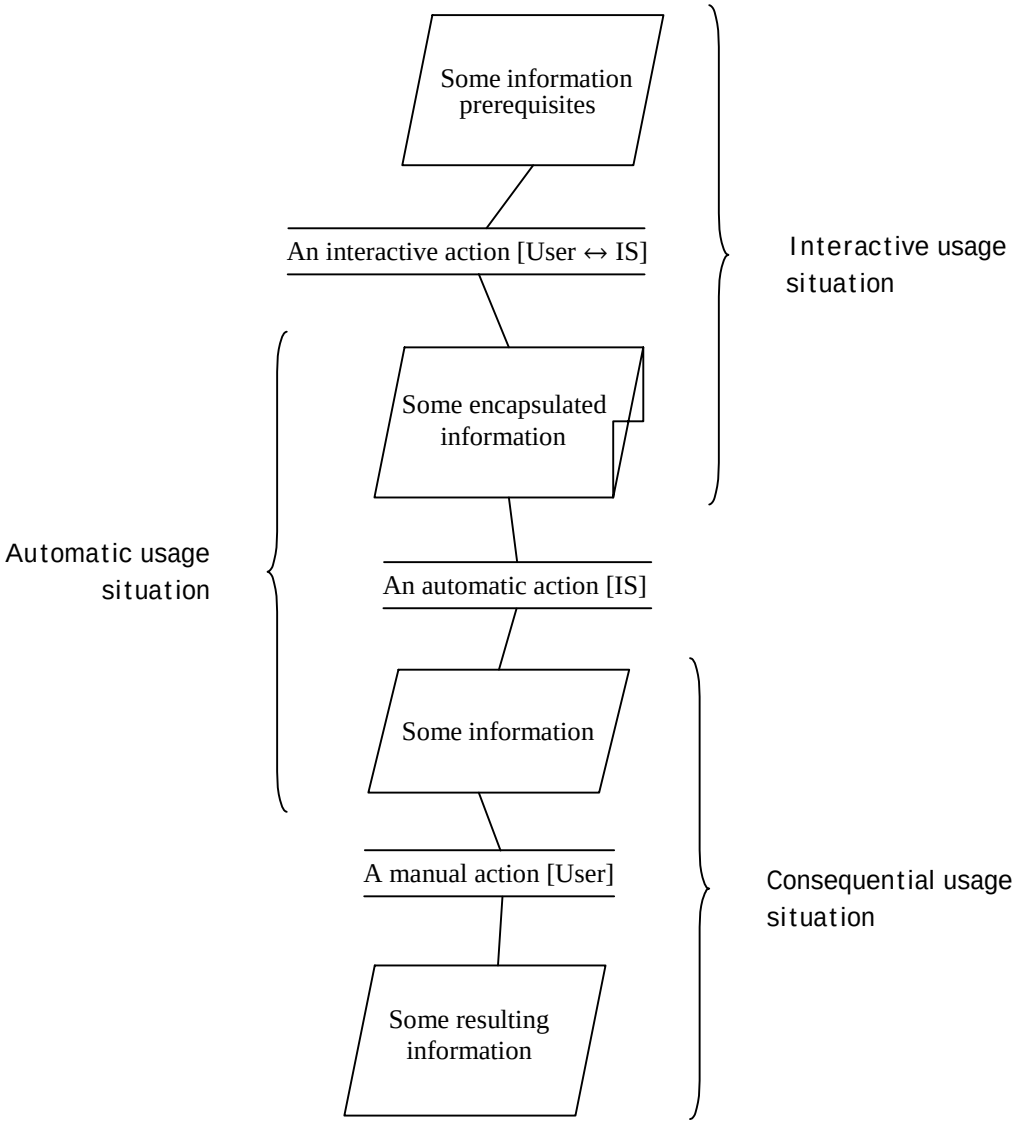



**Fig 3-12:** Interactive, automatic and consequential IS usage situations.

To summarize our view of information systems, such a system consists of:

- an action potential (a repertoire of actions and a vocabulary),
- a memory of earlier actions and action prerequisites, and
- actions performed interactively by the user and the system and/or automatically by the system.

Another way to conclude is to say that information systems have action ability – *actability*. We argue that new systems should deliberately be designed for actability of high quality, which we claim, is the most important quality of any information system. We define actability as an information system's ability to perform actions and to permit,

promote and facilitate users to perform their actions both through the system and based on messages from the system, in some business context.

In the HCI literature (e.g. Preece *et al.*, 1994) it is well argued that information systems must be usable (possess usability). We do not deny this, but we will argue that the most fundamental property of an information system is its actability. Usability can be considered as part of actability. A good actability means, among other things, that the actions of the system are transparent and accessible to its user. The action potential and the action memory of the system must be possible for the users to access.

In chapter 4 we will elaborate more carefully on the concept of actability, but first we will present and discuss some of the criticisms that have been raised against using speech act theory as a foundation for design of information technology.

## 3.4 Criticisms of Speech Act Theory

Even though the Language Action Perspective has gained much attention, and proved successful over the years, it has also been criticized. Ljungberg & Holm (1996) have investigated the pros and cons of adopting speech act theory as a foundation for design of information technology and identified two main sets of criticisms. The first set of criticisms concerns the problem of theoretical abstraction and can be broken down into two 'lines' of criticism: (a) the insufficiency of any theoretical abstraction, and (b) the insufficiency of speech act theory in particular. The second set of criticisms concerns the problem of a rationalistic design of work.

We believe that the second set of criticisms and the second line of the first set in this classification are intertwined. In all modelling efforts some abstractions are made. As pointed out earlier, any method or model directs attention to certain kinds of phenomena (cf. Ågerfalk & Åhlgren, 1999). This was claimed as an argument for the use of generic business frameworks in that they help analysts to identify important aspects of the modelled piece of reality. Of course, if the underlying theory is not able to capture the important aspects, then we are out on deep water. One can also use the arguments of Wittgenstein (1958) and conclude, from a philosophical point of view, that '*any attempt to produce a theory of meaning fails, simply because it is a theory*' (Ljungberg & Holm, 1996). However, as pointed out by Ljungberg & Holm, there is a difference between a descriptive theory used by passive observers (such as philosophers) and a normative theory used by active designers (such as systems

engineers). Ljungberg & Holm further argue that '*The crucial question concerns what needs to be articulated about work and communication to improve current praxis in the development and usage of IT*'. These kinds of criticisms are not specific to speech act theory and the Language Action Perspective. Rather they can be directed towards any theory, method or model for systems engineering. As we have argued in chapter 1, well-documented systems and structured engineering practices are imperative to handle changes and to shorten development schedules. Hence, we need to abstract and to use models. The critical issue thus seems to be to use theories that direct attention to the 'right' phenomena and, most important, that developers and other system stakeholders understand the importance of, and discuss and reflect upon, underlying theoretical foundations, and that an inter-subjective understanding of 'right' is reached.

One criticism belonging to the 'second line of criticisms' of the first set (i.e. the insufficiency of speech act theory) according to Ljungberg & Holm (1996) is the question of 'discourse versus conversation'. This topic is further elaborated by Holm & Jungberg (1996) who discuss whether systems development based on speech acts imposes discourses on businesses when conversations[7] are what to strive for in order to gain flexibility. To make this distinction they resort to two different linguistic schools for analysis of texts: discourse analysis and conversation analysis. From this point of view, a discourse is basically a larger part of a text than the sentence. A discourse is thus constituted by a pattern of discourse-parts (i.e. speech acts) into a well-defined discourse schema. A business model based on speech acts is thus claimed to be restricted to the particular action sequences imposed by the globally managed discourse pattern (or schema). The actors participating in a conversation, on the other hand, locally manage the conversation. Each actor can choose to continue (answer) a course of utterances, or to change the conversation in another direction. Alternatively a speaker can choose to invite another actor to continue the conversation, or start a new one. The problem is, according to Ljungberg & Holm (1996), that work in conversation analysis has shown that conversations are not usually structured in the way indicated by the conversation for action schema. Rather, conversation is usually built up of pairs of utterances as, for example, question-answer, offer-acceptance, *et cetera*.

Holm & Ljungberg (1996) further argue that business rules often require that a globally managed discourse is necessary but that the use

[7] Note that with the distinction between discourse and conversation, the conversation for action schema of Action Workflow describes a discourse and not a conversation.

of information systems within such discourses should be regarded as conversations. They propose a design method (the Commodious method) that treats business processes as discourses and human-human-computer interaction as conversations that conform to conversation schemas.

We believe that the distinction between discourse and conversation is valid in the context of human-human interaction. Consequently, Business Action Theory, which we use as our generic abstraction, does not impose this restriction; which, for example, the Action Workflow loop and the conversation for action schema do. However, since we are dealing with machines with finite instruction repertoires there can be no 'true' conversations (or Wittgensteinian language games for that matter) when it comes to human-computer interaction. At every given point in time there is a well-defined finite set of user actions possible to perform. That is, in every interactive usage situation there is a schema that the course of actions has to conform to. Thereby all human-computer interaction conforms to a discourse and the discourse is defined during systems development. Of course, there are often good reasons for keeping action sequence restrictions at a minimum and thereby providing more flexible user interfaces. At other times, business rules require that certain action sequences are either fulfilled or 'cancelled'. In other words, human-computer interaction is always conformant to a discourse but a rule of thumb would be to keep users as unaware of that as possible, and strive for a conversational discourse as illustrated in Fig 3-13.

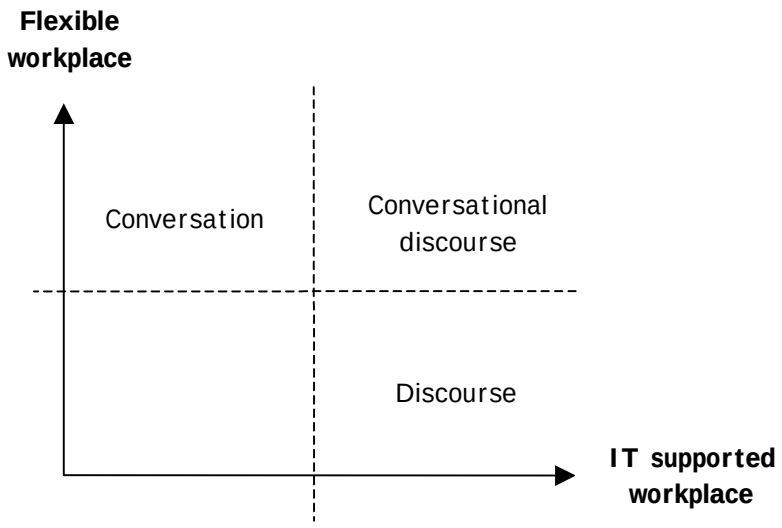



**Fig 3-13:** Flexible IT support as a conversational discourse.

Another criticism brought up by Ljungberg & Holm (1996) is that of the multi-functional nature of communicative acts. This issue was dis-

cussed previously (section 3.2.1.2) and concluded to be one of the benefits of BAT over Action Workflow.

A further criticism concerns the ignorance of the information content of speech acts in the conversation for action schema. The focus is on who is communicating when, and not on what is being communicated. This is probably an (over)reaction against the 'traditional' descriptive approaches to information systems and a position that we do not adopt. As discussed in chapter 4, the propositional content plays an important role within our approach, as well as in the works of Searle (1969; 1979).

To summarize this section we would like to conclude that there are issues in applying communicative action to the field of information systems that it is important to be aware of and to actively discuss. The important thing is never to believe that any theory is all-inclusive and the theory with a capital T. Just as businesses change so does our understanding of doing business and building information systems. Our proposal for an Action Theory of Information Systems shall therefore be viewed as representing what we know at the time of writing of this thesis, and will (hopefully) look different the day after it is finished.

# Chapter 4

# Information Systems Actability

*It's not the nature of the thing that matters. It's how we communicate with that thing, through whatever interfaces it shows us.*

Kraig Brockschmidt

In this chapter we discuss the reconciliation of business process modelling and usability-oriented design of information systems. The concept of actability – an information system's ability to perform actions and to permit, promote and facilitate users to perform their actions both through the system and based on messages from the system, in some business context – as enabler of such reconciliation is further elaborated, as is the concept of interactive usage situation.

## 4.1 The Requirements Engineering Gap

When developing information systems it is common to begin with some kind of business modelling in order to get an understanding of the business context in which new systems are to be used (e.g. Jacobson *et al.*, 1995; Bubenko & Kirikova, 1999). What is not so common is to make explicit use of the results of such modelling when defining system requirements, especially not requirements concerning human-computer interaction and usability. Information systems are used to perform and support business actions and therefore human-computer interaction models ought to be derived explicitly from business models.

There are approaches to information systems engineering that show awareness of the connection between business and information systems (e.g. Jacobson *et al.*, 1995; Jackson & Twaddle, 1997). Awareness in the sense that they make use of conceptual business modelling when defining requirements on system information content as well as derive interaction points from identified business tasks. What they seem to miss is that the intentional actions forming business tasks are crucial when designing user interfaces. A system that is unclear about what business actions it is able to perform, in interaction or automatically, is most likely unusable.

Available models for usability design, on the other hand, seem to be too narrow and focus too much on the interaction *per se*, without ap-

propriately and sufficiently relating it to the business context. Usability is often thought of as a property of the artefact in terms of, for example, relevance, learnability, flexibility and efficiency (Löwgren, 1993; Preece *et al.*, 1994). Relevance is defined by Löwgren (1993) as: '*...how well the system serves the users' needs*'. We believe that the users' needs must be judged in the light of the social context in which the users act. This context awareness is emphasized within the field of computer supported cooperative work (CSCW), partly as a reaction against the system focused usability of usability engineering (Löwgren, 1995). However, it requires more than collaboration among actors to form the social context of a business. In order to design for usability we must have a proper understanding of the business, its action structure as well as internal and external actors and agents and their professional language use.

We have thus identified a requirements engineering gap, or at least mismatch, between business modelling and systems modelling (see Fig 4-1).

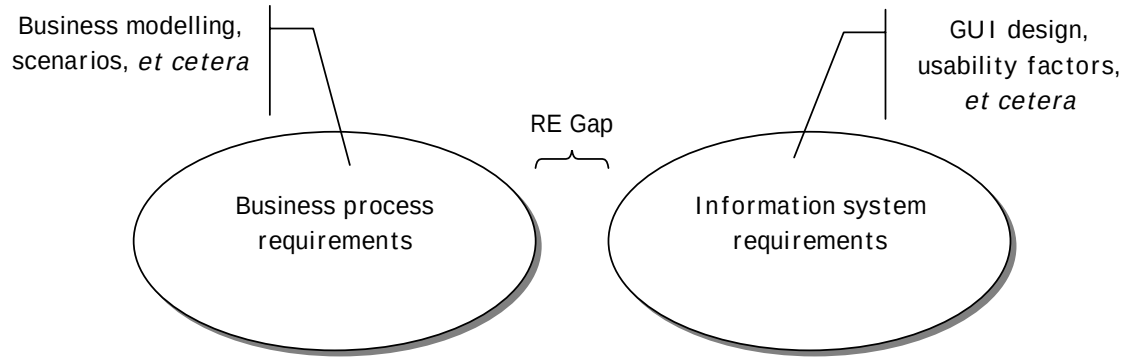



**Fig 4-1:** The requirements engineering gap.

In order to bridge the RE gap we propose that (1) user interaction should be regarded as a crucial part of doing business and its design should therefore be based on business modelling and (2) business processes should be designed with usability in mind. In other words, we would like to extend (1) business modelling to incorporate system interaction and (2) usability to incorporate design of business processes.

## 4.2 Interactive Usage Situations

In this subsection the concept of interactive usage situation (IUS) as performing language action through information systems is elaborated in further detail and related to the notions of task (e.g. Preece *et al.*, 1994; Shneiderman, 1998; Graham, 1998) and use case (Jacobson *et al.*, 1992; 1995). An action-oriented model of human-computer interaction, the

elementary interaction loop, is presented and a discussion of the role of IS documents is given.

### 4.2.1 Performing Language Action…

To say something is, according to Austin (1962), to perform three simultaneous acts: one locutionary act, one illocutionary act and one perlocutionary act. The locutionary act is the act of uttering certain words in a certain sequence with a certain sense and reference (i.e. meaning). When performing a locutionary act an illocutionary act is also performed simultaneously, i.e. we do something by speaking, for example, making a promise or warn (cf. chapter 2). Doing something will probably also (usually) have some effect on the hearer of the utterance, for example changing her knowledge or causing him to act in response. Causing these effects is to perform a perlocutionary act.

Searle (1969) adopted Austin's concepts of illocutionary and perlocutionary acts but refined the concept of locutionary act, which led to a concept of speech act as consisting of four different acts:

a) Uttering words = performing *utterance acts*.
b) Referring and predicating = performing *propositional acts*.
c) Stating, questioning, commanding, promising, *et cetera*. = performing *illocutionary acts*.
d) Causing effect in hearers = performing *perlocutionary acts*.

Note that the first three parts (a, b and c) are not separate things that a speaker does simultaneously. Likewise, a and b are not means to achieve c. Rather, as Searle (1969) puts it: '*utterance acts stand to propositional acts and illocutionary acts in the way in which, e.g., making an "X" on a ballot paper stands to voting*'.

However, following Norman's (1988) 'seven stages of action model' (see section 4.2.3), we argue that before performing an utterance act, the speaker forms the goal and intention of the utterance as well as formulates the sequence of words to be uttered ('*specifying the action*' in Norman's terminology). This is to say that based on the desired perlocution (effect) the speaker chooses illocution (what to do) and formulates the utterance (word sequence) concerning something (propositional content) in his mind before the actual uttering, i.e. thinking before talking. This should not be mistaken for a rationalistic view of human action and communication. Certainly, one can perform an utterance without thinking through what to say in advance. Our point is that if an important speech act is to be performed, such as a business action, a speaker *should* be aware of what he or she is actually doing by that act. Fur-

thermore, as we will show subsequently, an IS can (and should, in order to be actable) help the speaker to reach such awareness by supporting both the formulation and the uttering (execution) of speech acts (communicative actions). Hence, our view is normative, in the context of business action, and not descriptive in a general context of language use.

In sections 3.1, 3.2.1.2, and 3.4 we argued that a communicative action can have several communicative functions embedded in its action mode. As an implication, Searle's (1969) concept of speech act needs to be extended to allow for several illocutionary acts and corresponding perlocutionary acts to be performed within the same communicative act. This extension, together with the above discussion of formulation of speech acts, leads to a concept of 'communicative act' as consisting of five different acts:

a) Performing a *formulation act*.
b) Performing an *utterance act*.
c) Performing a *propositional act*.
d) Performing one or several *illocutionary acts*.
e) Performing one or several *perlocutionary acts*.

### 4.2.2 …through information systems

Before going into details, we will first establish some terminological conventions that we believe are more convenient for use in the context of information systems than those of 'pure' speech act theory.

By *action elementary message* (ae-message) we refer to the result of performing a speech act. Thus, an ae-message is what is being sent through an IS when performing an *elementary action* (e-action). To perform an e-action is referred to as *execute the e-action* or, alternatively, *send the ae-message*. Some *performer* (agent) acting on the *communicator*'s behalf might perform the e-action. A performer might be a human, in the case of interactive usage situations, or an artefact, in the case of automatic usage situations. The process of creating an ae-message is referred to as *formulate the ae-message*, which, just as sending, can be done interactively or automatically. An ae-message consists of a *propositional content* and an *action mode* corresponding to the propositional act and the illocutionary acts. The propositional content of an ae-message refers to one or more concepts, predicates attributes to those concepts and corresponds to the propositional act; see Fig 4-2.

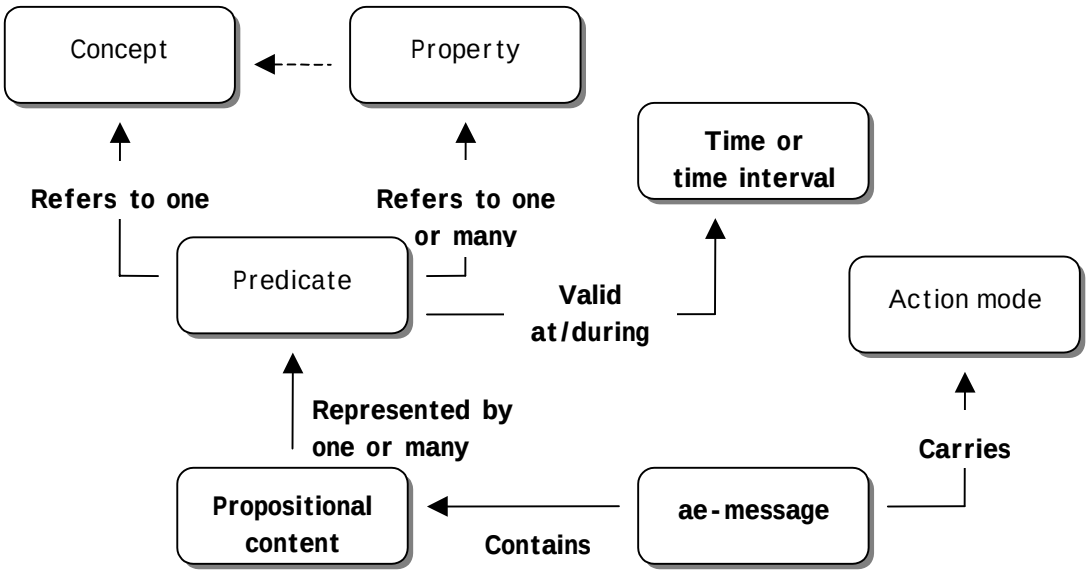



**Fig 4-2:** The concepts of ae-message and propositional content.

Since there is a one-to-one correspondence between ae-messages and e-actions, the action mode of an ae-message implies what *communication functions* the e-action serves, i.e. what kinds of relations that are created between the communicator (the speaker) and the interpreters (the hearers). By *communication effects* we refer to the perlocutionary acts of an e-action.

It is important to distinguish the concept of 'ae-message' from that of 'e-message' used in traditional (infological) information systems theory (e.g. Langefors, 1966; Sundgren, 1973). An ae-message is **not** simply an e-message with an augmented action mode. The propositional content of an ae-message refers to a subset of the universe of discourse (UoD), as do e-messages. However, the illocutionary forces of an utterance are often associated with a larger subset of the UoD than single e-messages. An e-message is the smallest unit of information that adds knowledge to an interpreter. In the same fashion, an ae-message is the smallest information unit that carries illocutionary forces (cf. Goldkuhl, 1995). Langefors (1995) refers to this as the *modality* of a sentence, '*...being a fact statement or an instruction or order to be executed*'. He further argues that '*It is a design decision whether to represent this in the message structure or in the environment*'. That is, illocutionary forces could be described by other e-messages referencing the e-messages that constitutes the propositional content of an utterance. We believe that such an approach is inappropriate in that it treats illocutionary forces as describable pieces of reality (belonging to the object system) rather than constituents of communicative action (as parts of the discourse).

Sundgren (1973) distinguishes between *property e-messages* and *relational e-messages*; see Fig 4-3.

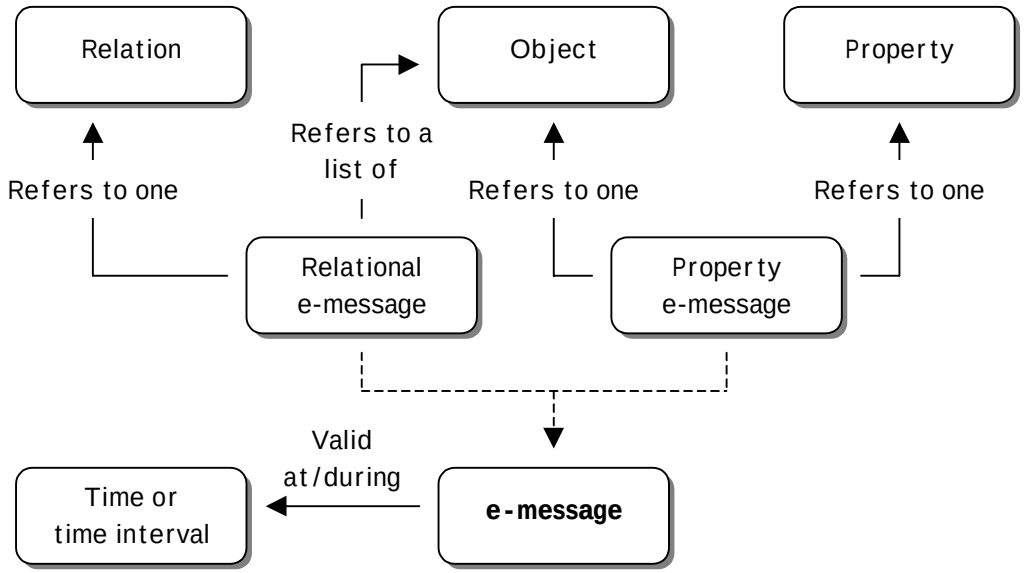



**Fig 4-3:** The concept of e-message.

A property e-message is used to predicate properties to concepts (called objects) and can be represented as a triple (o, p, t) where o is a reference to an object, p is a reference to a property of that object, and t is a point in time or a time interval during which the message is valid. A relational e-message is used to relate two or more objects and can be represented as a triple $((o_1, o_2, \ldots, o_n), r, t)$ where $(o_1, o_2, \ldots, o_n)$ is a list of references to objects, r is a reference to a relation, and t is a point in time or a time interval. Note that relations can be 'objectified', i.e. considered to be objects. This is necessary, for example, to allow for predicating properties to relations described by earlier e-messages (cf. *link attributes* of the OMT method). Example 4-1 illustrates the concept of e-message.

> **Example 4-1:** As an example, consider a property e-message describing that Peter was 26 years old at 09/06/1999. That e-message might look like: $e_1$ = (Peter, is 26 years old, 09/06/1999). Furthermore, a relational e-message describing that Peter owned the car with registration number ADB123 during the time interval 01/01/1997 – 01/05/1998 might look like: $e_2$ = ((Peter, ADB123), Owns, 01/01/1997 – 01/05/1998). The second e-message represents a binary relationship between Peter and the car. Sometimes there is a need to describe higher order relations, of which ternary relations are most common (Sundgren, 1973). A relational e-message describing that Peter sold the car to Jane might thus look like: $e_3$ = ((Peter, Jane, ADB123), Sold car,

01/05/1998). To exemplify predication of properties to a relation, we assume that the price of the car, when sold by Peter to Jane was SEK 98.000:-. This might be described by a second property e-message referencing the previously established relationship: ($e_3$, Agreed price is SEK 98.000:-, 01/05/1998). □

As we can see, there are two main differences between e-messages and ae-messages. The first difference is that an ae-message's propositional content may consist of several predicates, as discussed previously. Hence the propositional content corresponds to a c-message[1] (consolidated message) in the terminology of Sundgren (1973). The second difference is that we do not consider 'relation' as a meta construct. Instead we propose to view relations as special kinds of properties. This conforms with Searle's (1969) original description of the propositional act as referring and predicating. It is also consistent with an object-oriented worldview. Allowing for something external to all objects represent the fact that two or more objects are aware of each other obviously breaks the idea of encapsulation, which is one of the key concepts of object orientation (Graham, 1998). In essence, this resembles what has historically been called a 'binary model' (Griethuysen, 1982). Example 4-2 illustrates how the propositional contents, corresponding to the e-messages of Example 4-1 might look like.

**Example 4-2:** The propositional content representing that Peter was 26 years old at 09/06/1999 might look like: $p_1$ = {(Peter, is 26 years old, 09/06/1999)}. The propositional content representing that Peter owned the car with registration number ADB123 during the time interval 01/01/1997 – 01/05/1998 might look like: $p_2$ = {(Peter, Owned ADB123, 01/01/1997 – 01/05/1998)}. To represent a relation of order n, there is a need for n predicates. Hence, the propositional content describing that Peter sold the car to Jane for the price of SEK 98.000:- might look like: $p_3$ = {((Peter, Sold ADB123 to Jane), (Jane, Bought ADB123 from Peter), ADB123, Was sold by Peter to Jane), ($p_3$, Price was SEK 98.000:-)), 01/05/1998) }. The last propositional content might however be described in an alternative way. If we consider the actual mutual agreement of performing the selling of the car as a concept (which seems rational from an action perspective) we get the propositional content: $p_3$ = {((This Contract, Peter is seller), (This Contract, Jane is buyer), (This Contract, ADB123 is the car sold), (This Contract, SEK 98.000:- is the agreed price)) , 01/05/1998)} □

[1] '*… a c-message could be any combination of e-messages … A priori any message …will be considered to be a c-message, which, after analysis, may be broken down into e-messages…*' (Sundgren, 1973, p. 97).

Please note that we will sometimes use the term 'message' instead of 'ae-message' when it is clear from the context that we refer to communicative action.

We suggest to analyse human-computer interaction on a 'per IUS basis' as an integral part of business modelling. This is necessary in order to cope with complexity (by analysing the parts of the system separately) and still maintain a holistic view of the business context, in which the interaction is taking place, i.e. part of bridging the RE-gap. Furthermore, we suggest that the analysis of each IUS is performed on a 'per ae-message basis', for similar reasons.

In essence, actability-oriented requirements engineering is concerned with three questions (see Fig 4-4).

- Which ae-messages have to be sent to achieve the desired business effects, i.e. what e-actions have to be performed to produce the desired output?
- For each ae-message, what is its propositional content, its communication functions, its communication effects, its communicator and performer, its intended interpreters?
- What computer support is required to formulate and send the ae-messages and, what information from previously sent ae-messages is required to formulate this one?

**Fig 4-4:** The three questions of actability-oriented requirements engineering.

The questions concern three aspects of ae-messages (which ae-messages are relevant for the business context at hand; what constitutes each relevant ae-message; and what is needed, in terms of previously sent ae-messages and computer support, to formulate each relevant ae-message?).

Thus, an IUS consists of the formulation and sending of one or more ae-messages. The process of formulating an ae-message should be supported by the IS and can be viewed as a dialogue between the user and the IS. Such a dialogue consists of interaction cycles of user actions, IS actions and user interpretation acts, referred to as elementary interactions (e-interactions), in accordance to the Elementary InterAction Loop (defined below). Before any user action is performed, the IS might perform some initial IS actions that give the user initial information about possible kinds of e-actions to be performed, and/or suggested content of an ae-message to send. Both user actions and IS actions result in a special type of message referred to as interactive messages (ia-

messages) used 'IUS-internally'. A special kind of e-interaction is the one triggered by the user action that sends the actual ae-message and thus executes the e-action.

Fig 4-5 illustrates these concepts with ia-messages being sent between the user and the computer and the ae-message, sent through the IS, causing some business effect(s).

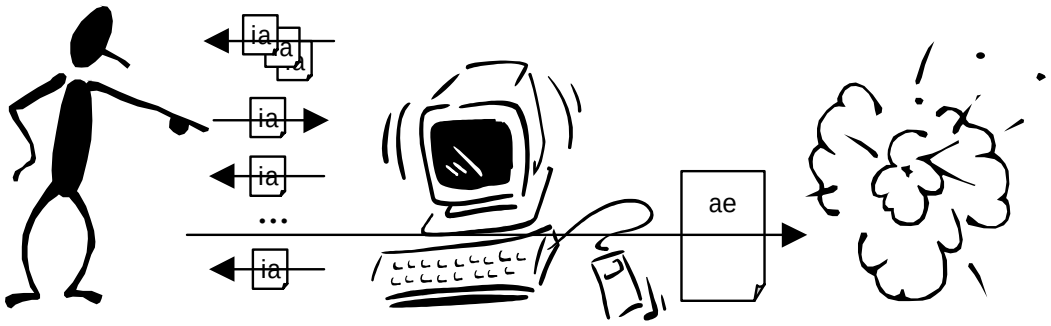



**Fig 4-5:** Action elementary messages are formulated during e-interactions in dialogue with, and sent through, the IS in order to cause some business effect(s).

Messages can be visualized as *documents*. An 'Invoice document' used to command customers to pay, for example, would typically be used to carry the ae-message 'Customer invoice' and thus corresponds to the e-action 'Send invoice to customer'.

We can distinguish three categories of document types that call for somewhat different treatment. There are *paper documents*, such as reports and lists; there are *electronically transmitted documents*, such as EDI-documents (Electronic Document Interchange); and there are *screen documents*. Screen documents can be further classified as *interactive* or *static*. Interactive screen documents permit direct manipulation on the computer screen, for example forms and dialog boxes, which their static counterparts do not, for example simple http-documents. These document categories relate to the three types of usage situations discussed in chapter 3. However, since usage situations overlap, all three categories must be considered when talking about actability.

Let us consider an example (Example 4-3) to illustrate how these concepts relate.

> **Example 4-3:** To make a customer pay (communication effect), a business firm (communicator) might send the EDI-document 'Customer invoice' to command (action mode) the customer to do so (communication effect). The e-action of sending the invoice is performed during an IUS called 'Invoicing' by some office clerk (performer). To perform the e-action, an interactive document 'Invoice'

is used. 'Invoice' consists, among other things, of a button labelled 'Send invoice' used to execute the e-action. When entering the IUS the IS presents information about the current customer order (propositional content) in the interactive document (initial IS actions). But, before executing the e-action, the invoice must be manually checked for correctness and, perhaps, discounts need to be manually chosen given some different options (e-interactions). Finally, when the 'Send invoice' button is clicked, the EDI-document is sent to the customer and a pop-up window notifies the user about this (that is, the special kind of e-interaction that executes the e-action). □

### 4.2.3 The Elementary InterAction Loop

To analyse human-computer interaction we propose the concept of elementary interaction (e-interaction). An e-interaction is a human-computer interaction that cannot reasonably be decomposed into smaller interactions. To decide the appropriate granularity of e-interactions we follow Graham's (1998) rule of thumb for deciding the granularity of task-objects. Translated to our terminology, his advice is to decompose interactions until further decomposition would introduce terminology that is not part of the business' ontology. That is, when computer terminology, such as 'move the mouse', is needed to do one more decomposition the appropriate granularity is reached.

Basically, an interactive usage situation is constituted by a set of e-interactions with some sequence restrictions applied. Each e-interaction (initiated by a user action) is matched by one or more actions performed by the system. That is, an actor performs a user action (usually by use of an interactive screen document) whereupon the system answers by performing its corresponding IS actions. After the IS action(s) are performed the actor interprets and evaluates what the system has achieved. This sequence of actions constitutes an Elementary InterAction Loop (EIAL) as shown in Fig 4-6.

During the execution of an instance of such a loop the interactive situation is in a transient state and the loop must be completed before a new e-interaction can occur. Actually, there are three different states involved: an *initial state* ($S_0$) before the user action, a *waiting for system to respond state* ($S_1$) between the user action and the IS action(s), and an *IS action accomplished state* ($S_2$) after the IS action(s). It is possible that the actor has to switch to another e-interaction in order to complete the interpretation act. For example to check some details in another document. Therefore, each e-interaction might be infinitely recursively expanded. Such sub-interactions must be permitted but analysis must ex-

plore the possibility that two or more e-interactions might have been regarded as one, in which case these should be separated and treated independently. Note that IUSs are extracted from business process modelling and how the EIAL thereby connects business requirements with functional requirements and usability requirements of the planned IS.

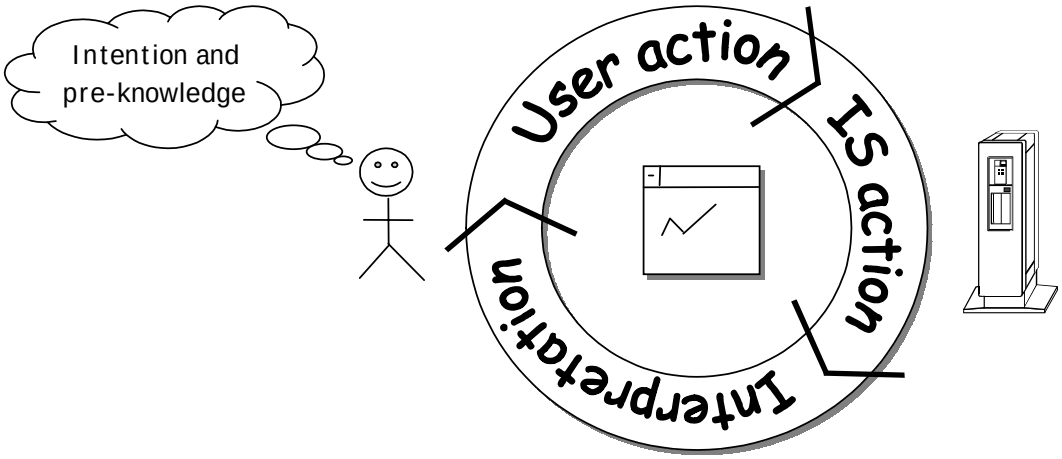



**Fig 4-6:** The Elementary InterAction Loop.

The EIAL bears considerable resemblance to Norman's (1988) 'seven stages of action' (SSA) model of human-'everyday thing' interaction. With his model, Norman distinguishes between seven stages of actions that a user goes through when trying to use, for example, information systems:

1. Forming the goal.
2. Forming the intention.
3. Specifying the action.
4. Executing the action.
5. Perceiving the system state.
6. Interpreting the system state.
7. Evaluating the outcome.

One of Norman's (*ibid.*) contributions with the SSA is that he is (as pointed out by Shneiderman, 1998, p. 57) '*placing his stages in the context of cycles of action and evaluation*'. That is in contrast to other models '*which deal mainly with the knowledge that must be in the user's mind*' (*ibid.*). We argue that the SSA stages 1 to 3 corresponds to the 'precondition-cloud' of the EIAL (Fig 4-6), that the SSA stage 4 corresponds to the user action of the EIAL, and that the SSA stages 5 to 7 corresponds to the EIAL interpretation act. If we agree to these correspondences, we can see how the EIAL takes the SSA even further by taking the system action into account as well (a stage 4.5). This is an

important property of human-computer interaction from a language action perspective – there is no point in analysing user actions and IS actions separately, without explicitly relating them to each other. Doing so would be to analyse single utterances outside their discourse, which would certainly be meaningless. Consider, for example, the trivial case of an IS action constituted by getting a file name from disk and asking whether or not the user wants to save changes before exiting. Determining whether such IS behaviour seems rational requires that we know whether the user is really trying to exit something or not, i.e. we must know something about the discourse.

Norman's (*ibid.*) model is general in the sense that it accounts for actions at 'all' levels, i.e. for single actions as well as for action aggregates (what we refer to as activities). We can therefore use the model to understand both the sending and formulation of ae-messages, by use of information systems. From a strict human-computer interaction point of view, the only thing that differs between the two is the user's goal and thereby intention. In the first case (e-actions) the goal is to achieve the communication effect and in the latter to create a part of the ae-message (a sub-goal). From a system analysis perspective, on the other hand, it is important to distinguish between them. The importance lies in the fact that a system, in order to be actable, should be clear and understandable regarding what e-interaction that is used to really execute the e-action. Up until that point in the interaction the user can, without obligations, cancel the ae-message; after that point, it cannot be undone.

Table 4-1 shows how our notion of performing e-actions and e-interactions can be related to Norman's (*ibid.*) SSA model. Remember that 'Formulating the ae-message' as well as 'Executing the e-action' are performed as e-interactions: the first as a sequence of one or more e-interactions, and the latter as a single one.

As discussed previously, e-interactions are constituted by sending ia-messages between the user and the IS and each ia-message is communicated using an interactive screen document (ISD). Actually, each ia-message is communicated by use of a piece of an ISD, referred to as an interactive screen document part (ISD-part). Each ISD can thus be viewed as an aggregate of ISD-parts.

| Stage | Norman | e-actions | e-interactions | |
|---|---|---|---|---|
| 1 | Forming the goal | Deciding communication effect | User Action | Forming the goal |
| 2 | Forming the intention | Choosing action mode | | Choosing User action |
| 3 | Specifying the action | Formulating the ae-message | | Specifying the User action. |
| 4 | Executing the action | Executing the e-action | | Performing the User action |
| 4.5 | | | | IS action(s) performed |
| 5 | Perceiving the system state | Perceiving the business state | Interact. | Receiving the ia-message (response) from the system |
| 6 | Interpreting the system state | Interpreting the business state | | Performing the interpretation act |
| 7 | Evaluating the outcome | Evaluating the performed action and the IS | | Evaluating what the system has achieved |

**Table 4-1:** Norman's SSA model compared with execution of e-actions and formulation of ae-messages by e-interactions.

Consider, for example, an order-ISD with a section of information about the customer who placed the order. That customer-section would typically constitute an ISD-part that is used to carry the propositional content of an E-interaction 'Change customer name'. Such an ISD-part might also be used in other ISDs where customer information is used. Thus, both ISD-parts as well as e-interactions are potentially reusable.

### 4.2.4 Interactive Screen Documents and Action Potential

All documents have an action potential, i.e. they can be used for action. In the case of interactive screen documents the, action potential is materialized as the functionality offered to the interacting actor at any given point in time (i.e. the possible e-interactions to initiate) by, for example, screen items to click.

An interactive screen document's action potential varies over time as the result of (1) the (type of) interactive usage situation (IUS) at hand and (2) the state of that IUS. A 'Product information' interactive document might, for example, offer different action potential during the IUS 'Create customer order' and the IUS 'Stocktaking'. During 'Create customer order' the same document might also require that some particular customer details have been entered (a certain state is reached) before the order can be registered. Such changes in action potential can, and should, be visible to the user, which could be done by, for example, disable buttons that cannot be used in a certain state.

In particular, the IS should help users to handle the seven stages of interaction (cf. Table 4-1). That is, to

1. decide which communication effects to strive for and to choose what e-action to perform (at e-action level); and to
2. form the goal, choose user action, specify the user action, perform the user action, receive the ia-message from the system, perform the interpretation act, and evaluate what the system has achieved (at e-interaction level); and finally to
3. perceive, interpret, and evaluate the e-action and its business effect(s).

The visibility of action potential is similar to what Norman (1988) refers to as 'affordance', i.e. '*the perceived and actual properties of the thing, primarily those fundamental properties that determine just how the thing could possibly be used*'. The point of taking affordance, and hence action potential visibility, into account is that, in a perfect world, '*the user knows what to do just by looking: no picture, label, or instruction is required*' (*ibid.*).

### 4.2.5 Interactive Usage Situations versus Tasks and Use Cases

Within the field of Human-Computer Interaction (HCI), IS usage is often thought of as performance of 'tasks' and interface design thus ought to be based on task analysis (Preece *et al.*, 1994; Shneiderman, 1998). A task is something that a user wants to accomplish using the computer (including software). '*That task includes the universe of real-world objects with which users work to accomplish their intentions and the actions that they apply to those objects*' (Shneiderman, 1998, p. 61). In the object-oriented community the phrase 'use case' is often favoured instead of 'task' (cf. Lauesson & Younessi, 1998). A use case is defined by Jacobson *et al.* (1992) as '*a specific way of using the system … thus a special sequence of related transactions performed by an actor and the system in a dialogue*'. The key to task analysis, and use case analysis, is to decompose high-level tasks into smaller ones in order to find the *atomic actions* constituting the tasks. One problem of task analysis is to choose the most appropriate set of 'atomic actions' (Shneiderman, 1998, p. 70). As described previously, we propose to use Graham's (1998) rule of thumb to decide the appropriate granularity of IUSs, i.e. decompose interactions until further decomposition would introduce terminology that is not part of the business' ontology. Fig 4-7 shows how this division of concerns is related to Shneiderman's model of task and interface objects and actions.

Another problem of task (and use case) analysis is to find the high-level tasks in the first place. We argue that high-level tasks ought to be found and delineated based on the business communication performed

through and by the IS. Hence, we define IUS to be a primitive sequence of business actions that consists of all interactive actions, and intermediate manual and automatic actions, that are performed adjacent in time at the same place by the same performers, cf. Holm & Ljungberg's (1996) notion of 'conversation'. This way, tasks, and hence human-computer-interactions become integrated in the business model and the business context of task analysis is emphasized. Thus, potentially, the elimination of human-computer interaction design becomes a separate activity, disconnected from the business development.

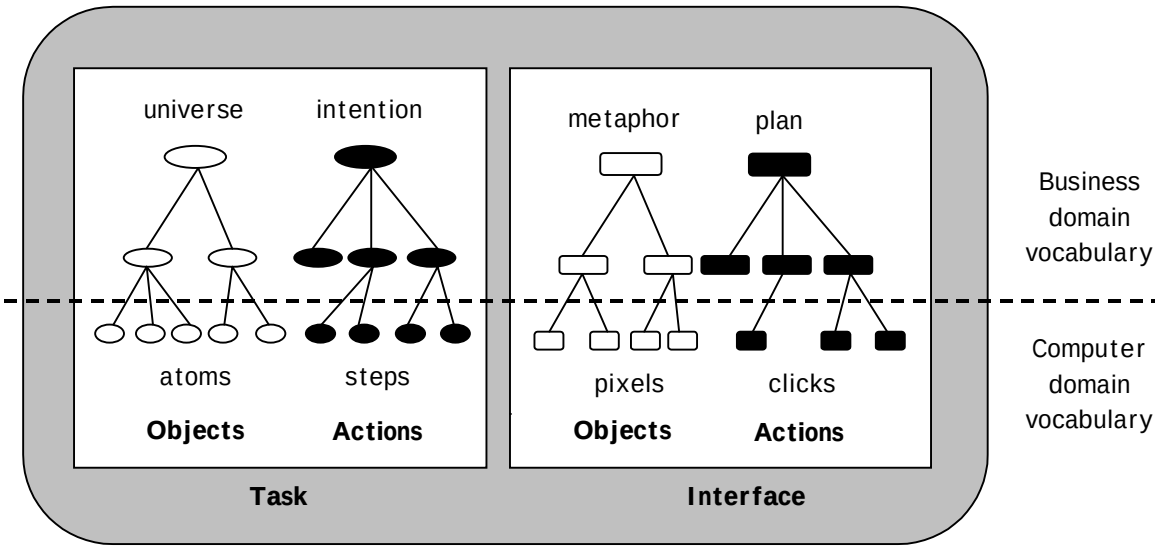



**Fig 4-7:** Task and interface concepts, separated into hierarchies of objects and actions. Adopted (and modified) from Shneiderman (1998, p. 62).

### 4.2.6 Modelling and Validation of Actability Requirements

As mentioned above, IUSs are derived explicitly from business modelling. The IUSs can then be used as a basis for a more detailed analysis concerning the interactive actions and the design of interactive documents. The 'performer part' of interactive actions is thus focused. We turn our attention to the information that flows (the messages sent) between the actor and the system, its origin and form, to analyse functional requirements and requirements concerning usability in the context of business processes, i.e. actability requirements. Analysis of human-computer interaction for actability is preferably performed with a mixed analytical and experimental approach in a dialectic manner (cf. Mathiassen *et al.*, 1995). Analytical models are continuously validated against document prototypes and prototypes are continuously verified against analytical models. This way both consistence and relevance are

achieved in a way hardly obtained by using only one of the approaches exclusively.

Both modelling and validation of requirements can be performed based on the 'three questions of actability-oriented requirements engineering' as discussed previously (Fig 4-4). The ae-messages to be sent in each IUS can be identified by looking at the results of the corresponding action(s) in the business model. These results are the means to reach desired effects of the IUS. For each ae-message there should be a propositional content and an action mode that support those effects. Furthermore, there should be possible e-interactions and initial IS actions that permit, promote and facilitate the formulation of those ae-messages. To ensure this, different usability factors, such as 'easy to learn' and 'easy to use' should be considered and, for example, interaction style guides could be used as an aid (cf. Preece *et al.*, 1994; Card & Anderson, 1987; Monk *et al.*, 1993; Nielsen, 1993; Lauesson & Younessi, 1998). One theory from the field of HCI claims that use of metaphors will make systems more learnable and easier to use. Preece *et al.* (1994) claim that the aim of using metaphors is to benefit from prior knowledge in order to develop an understanding of the new domain more readily. This is to say that metaphors can be used to represent action potential in recognizable ways and thus to promote affordance.

During this process both the contents and demarcations of the IUSs are likely to change. Therefore, as we shall see in the description of VIBA/SIMM (chapters 5 to 7), we refer to these as *Interactive usage Situation Proposals* (ISP) and *Automatic usage Situation Proposals* (ASPs) during systems engineering. Results from one of the case studies (see chapter 8) show that earlier assumptions about the business action structure might have to be reconsidered and analysis of interaction thus becomes a valuable source of information also for business process modelling. This is an example of the need to integrate business modelling and interaction design. The necessity to formalize the action structure reveals information which is hard to elicit only by interviewing and observing actors. By putting actors in front of the future system they feel the need to be explicit, and they instantly see the effects of alternative business descriptions.

## 4.3 Action Memory and Databases

What (hopefully) should be clear by now is that we propose to view information systems as part of the communication, and thus action, that take place in businesses. Suppose that an actor A does something interactively, i.e. communicates an ae-message to an actor B through an IS (A

and B might be the same actor, but do not have to be). In this communication the IS acts as mediator of the communication.

As discussed in chapter 3, an IS must hold a memory of actions already performed (action memory) as well as preconditions for future actions. This action memory is part of the *organizational memory* of the business. Ultimately these two would be equivalent. But, since there will always exist manual action as well as tacit knowledge in organisations, it is important to distinguish the two. The action memory might be implemented as a traditional database but might as well be constituted by distributed objects or some other technological solution, which might not even be invented yet.

Thus, messages might be stored in the action memory for later retrieval and use in some other context (or the same). Such a message implies an update of the action memory. A message does not, however, necessarily imply an action memory update. It is possible to imagine messages that are merely sent through the IS without making use of previously sent messages, for example e-mail. Some messages might also refer to previously sent messages without actually changing anything in the action memory. In the latter case, the message is related to previously sent messages by its propositional content. What to include in the action memory is a systems engineering issue. Note that if an ae-message that at a first glance seems to be action memory independent constitutes important business action, it shall be stored in the action memory. See Fig 4-8 for an illustration of this discussion.

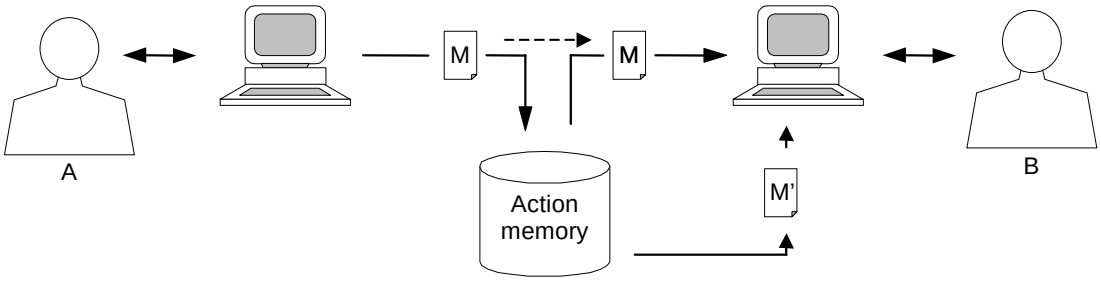



**Fig 4-8:** Information system as communication mediator.

Thus, messages communicated through an IS can be classified in two main categories: (1) messages that are independent of previously sent messages (and thereby do not depend on the action memory), and (2) messages that depend on previously sent messages (and thereby depend on the action memory).

The category of dependent messages (category 2) can be further sub-categorized into messages that (a) merely use the action memory,

and (b) affect the action memory. Since ae-messages are products of e-actions (with a one-to-one correspondence), it is possible to classify actions as:

1. Action memory independent,
2. Action memory using, or
3. Action memory affecting.

This distinction might be useful when moving to the design phase of systems engineering. It will, for example, give a hint about what kinds of components to use when constructing the user interface: static, data aware, *et cetera*.

## 4.4 Further Actability Discussion

We have argued that design of information systems should be performed with a proper understanding of the business, gained through an integrated development of business processes and their supporting information systems. When designing IS interaction, usability factors such as 'easy to learn', 'easy to use' and 'interaction styles' are important to consider. We do however believe that the common notion of 'usability' is too narrow since it is often perceived as dealing only with design of user interfaces. When designing communication through an IS, the question of how to interact is, of course, important. Equally important, though, is the question of what to communicate and that of why. Moreover, all three aspects must be considered in the business context where the communication is taking place. That is, a social context that is never static and fully predictable.

### 4.4.1 On the Definition of Actability

We use the concept 'actability', which is based on theories from the Language Action Perspective and usability, to talk about information systems use in business processes. Information systems actability was defined in chapter 3 as:

> *An information system's ability to perform actions and to permit, promote and facilitate users to perform their actions both through the system and based on messages from the system, in some business context.*

When stating that information systems are (1) '*able to perform actions*', they are thought of as agents acting on someone's behalf. IS action is always predefined and there is always a human actor responsible for it. Such responsibilities are important to identify and establish during

systems engineering. Stating that information systems should 'permit, promote and facilitate users to perform their actions' implies that systems should not only be usable and clear about their action potential but should also encourage users to take advantage of acting through the systems. User actions could be performed (2) '*through the system*'. That is when a user performs actions with an IS as a vehicle for communication. On the contrary, (3) '*based on messages from the system*' refers to the case when an IS is used to create action possibilities. These three kinds of usage situations are respectively referred to as *automatic*, *interactive*, and *consequential*. The 'degree' of actability possessed by a certain IS is always related to the particular business context. The business context includes actors' pre-knowledge and skills regarding both the IS and the business tasks performed. Therefore, actability is not a static property of an IS but depends on the social structures surrounding it.

### 4.4.2 On the use of the term 'Actability'

The term 'actability' is defined by the Oxford English Dictionary as: '*capability of being acted*'. It is the substantive form of 'actable', which is defined as: '*capable of being acted or carried out in practice*'.

The term is traditionally used in the context of theatre and drama, and Levén (1995) was probably among the first to use it when discussing information systems development. His definition refers to a slightly different concept than what we propose. According to Levén's definition, actability is what you get when raising your eyes above the computer screen and take the value creating, and customer centred, activities of a commercial firm as starting point for IS evaluation. This is in accordance with our view of actability, but a too restrictive notion to use as a definition. You do not need any customers to be able to talk about actability. The point is not whether the information system creates value for the customer, but that it creates what the actors intend to create. In a commercial firm, an actable IS would probably, directly or indirectly, create value for the customer. But in, say, a university, with a very unclear notion of 'customer', a system to keep track of whose turn it is to make coffee would still be actable if it actually eased the professors' burden of coffee making. Levén distinguishes actability from usability by stating that usability is about the relation between the user and the system whilst actability is about the relationship between the client (customer) and the user-system constellation. We would like to say that actability is about the ternary relationship between actors (i.e. users, which might be customers), information systems and business tasks (which might involve customers).

# Chapter 5

# Versatile Information and Business Requirements Analysis

This chapter introduces the VIBA/SIMM approach to information systems engineering. The design rationale and overall approach is presented as well as the main focal areas of the method. The version of VIBA that is presented in this chapter (and in chapters 6 and 7) is VIBA'99. Therefore the chapter ends with a brief comparison between VIBA'93 and VIBA'99. Note that chapters 6 and 7 go into further details of the method.

## 5.1 Design Rationale

VIBA/SIMM (Versatile Information and Business requirements Analysis according to the Situation adaptable work and Information systems Modelling Method) is a method for specifying requirements on information systems and businesses to be developed. In order to bridge the RE-gap (see chapter 4), information systems engineering is considered to be a special kind of business development and the aim is to build information systems that possess high degrees of actability, i.e. information systems that:

- are usable,
- are integrated and congruent with the business,
- are understandable and predictable,
- have visible actors and
- enable and support good working conditions.

The method builds explicitly on the Action Theory of Information Systems (ATIS), outlined in chapters 3 and 4.

One aim of the method is to facilitate a rapid development setting and to make use of rapid application development tools. Another aim is to deliver specifications that enable traceability from detailed systems specifications to business models and vice versa.

The method context is business information systems in process-oriented organizations.

## 5.2 Overall Approach

VIBA/SIMM, as a method for systems engineering, should be understood as a part of the SIMM family of methods. VIBA presupposes that some feasibility study or business change analysis has been performed. Ultimately, such a study has been carried out by use of Change analysis/SIMM (e.g. Goldkuhl & Röstlinger, 1988; 1993; Lind, 1996). We say 'ultimately' because then some of the VIBA work has already been performed. Thus, as shown in Fig 5-1, there is a smooth transition from Change analysis to VIBA.

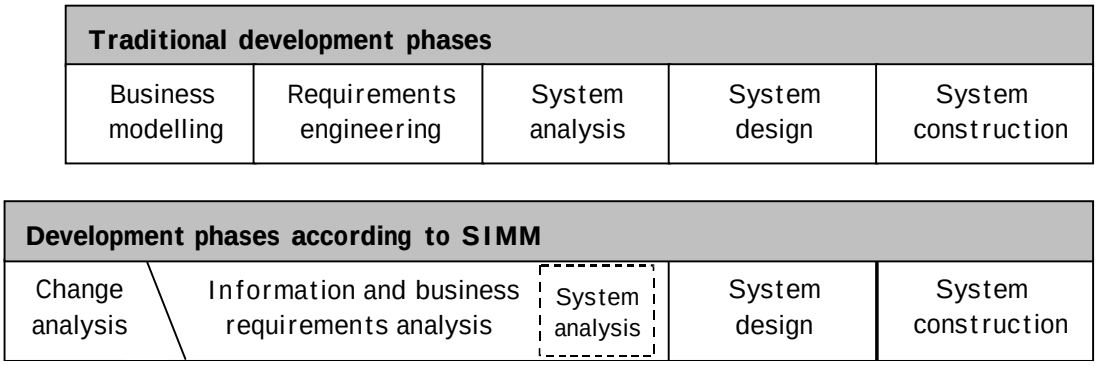



**Fig 5-1:** The systems engineering phases according to SIMM.

The 'system analysis phase' of Fig 5-1 (the dashed rectangle) illustrates that when applying VIBA, much of traditional system analysis has been performed during the requirements engineering phase. The reason that it is still included as a separate activity is that, depending on target environment and design paradigm, some complementary analysis might still be needed.

One should not be misled by Fig 5-1 to believe that development according to SIMM implies a waterfall model. The figure is merely an illustration of how SIMM relates to the traditional lifecycle model. VIBA might be used with any model, including the waterfall model, but encourages the use of a more iterative and incremental approach such as Boehm's spiral model (Boehm, 1988). Note also that, as discussed in chapter 1, we do not regard the requirements process as ending before design, but as a continuos process that spans possibly many information systems engineering projects.

VIBA/SIMM consists of three interacting parts: modelling, prototyping and evaluation (see Fig 5-2). Evaluation is based either on analytical models or developed prototypes. It is important to notice that these three parts do not constitute separate phases or focal areas (defined below) within VIBA. Instead, they should be understood as repre-

senting an attitude towards systems engineering that VIBA promotes. Specifically the evaluation part is implicit in the proposed ways of working. The idea is to promote a reflective approach that constantly evaluates models and prototypes. Evaluation means to critically examine the specification and verify its internal consistency, as well as external validity with respect to the business being designed. The key concept for evaluation is hence 'why' – why do the models and prototypes look the way they do?

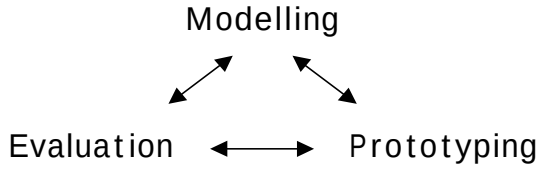



**Fig 5-2:** VIBA/SIMM as three interacting parts.

## 5.3 Focal Areas of VIBA/SIMM

Requirements engineering according to VIBA is carried out within two primary focal areas: Business Process Modelling (BPM) and Information Systems focused Modelling (ISM); see Fig 5-3.

During BPM the business' goals, problems, strengths and weaknesses, action structure, and other development constraints are analysed. As discussed in section 5.2, these activities are performed as a direct continuation of Change analysis or some other feasibility study, which is supposed to be done before system development starts. The main outcome of BPM is a business action model documented in Action diagrams. Action diagrams show the action logic of the business with the results of and prerequisites for actions. The Action diagrams are explicit as regards what actions are to be performed by or in interaction with planned information systems.

Specified interactive and automatic usage situation proposals (ISPs and ASPs), derived from the Action diagrams of BPM, are used as a basis for ISM, which consists of three main focal areas: Interaction analysis, Conceptual analysis and Document analysis. Interaction analysis is concerned with the human-computer interaction, i.e. how communication by use of the information system (IS) is to be performed, and automatic IS action. During Conceptual analysis the focus is on what to communicate, i.e. the concepts being used: their definitions, relationships and lifecycles. Document analysis is about the documents as creators, users and carriers of messages, and their relationships.

We use the term 'focal areas' rather then, for example, 'phases' or 'activities' when referring to BPM and ISM because of their tight coupling to each other in terms of modelling activities. They are not distinct activities performed in isolation but different areas to focus on during an integrated modelling effort. Even though BPM is typically initiated before ISM, it is not abandoned when ISM starts. Rather, ISM often yields insights into the business that entail further elaboration at 'BPM level'. In effect, this view of systems engineering actually prevents the use of a waterfall development model. However, if the insights gained during initial BPM work can be regarded sufficient, such a development model might still be used.

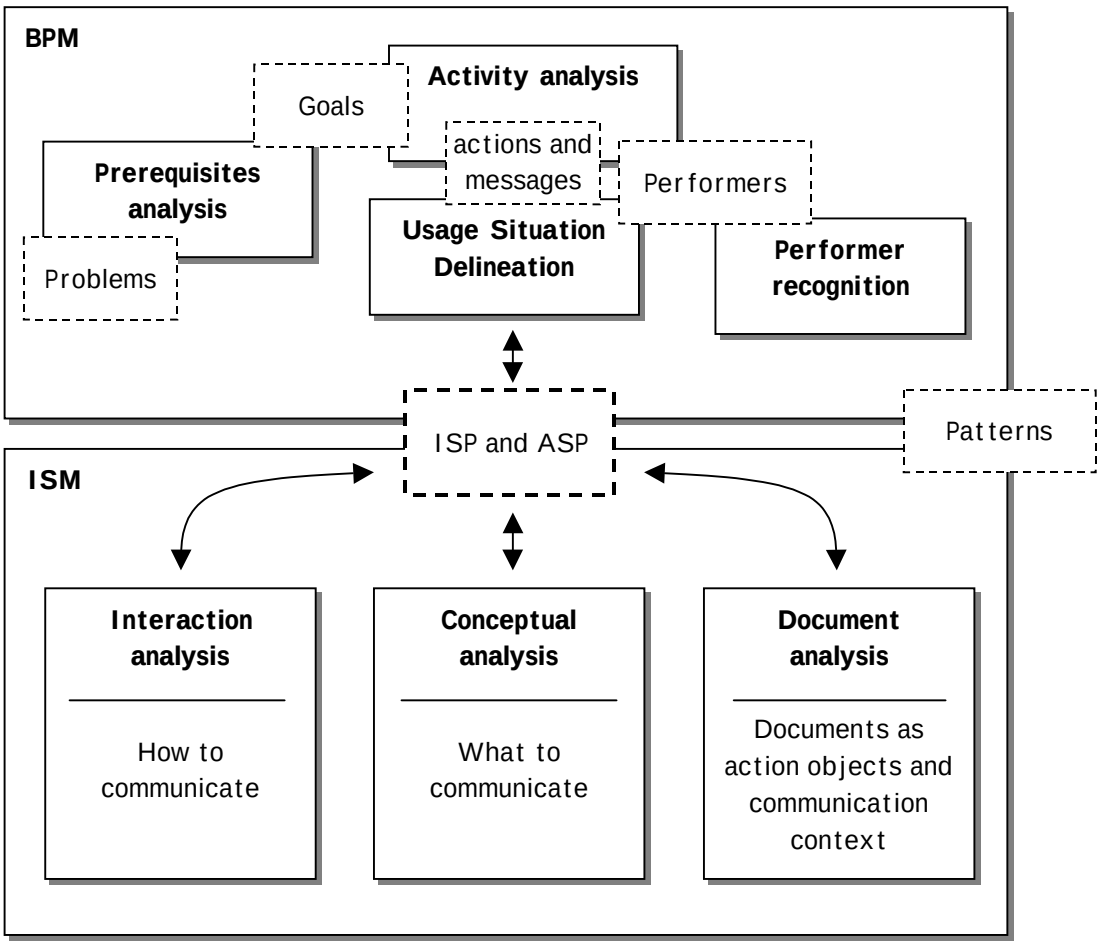



**Fig 5-3:** Main components of VIBA/SIMM

The distinction between 'focal areas' and 'phases' counts for the parts of ISM as well. In the case of BPM there is, however, a somewhat more distinct ordering of the activities – and therefore we refer to these as 'engineering activities', which does not mean that BPM can not be

used incrementally; but states that each increment involves subsequent engineering activities.

As shown in Fig 5-3 patterns play an important role during both BPM and ISM. By 'patterns' we refer to pre-existing solutions to problems similar to the ones at hand. Patterns might be requirements patterns (e.g. Maiden *et al.*, 1998), analysis patterns (e.g. Fowler, 1997), or design patterns (e.g. Gamma *et al.*, 1995), i.e. in accordance with the traditional use of the term in the fields of information systems and software engineering. Moreover, existing artefacts, such as already implemented information systems and businesses, might also constitute patterns from our point of view. The real issue of highlighting patterns as an important part of VIBA/SIMM is to encourage an attitude towards information systems engineering that is open-minded and that counteracts the so-called 'not invented here syndrome'.

## 5.4 Business Process Modelling

Business Process Modelling (BPM) constitutes analysis and design of the business studied in terms of one or more business processes. As an integrated part of BPM an analysis of the role of information systems within the business is performed. Usage situation proposals are identified and delineated, and related to goals and problems of the business. BPM yields a comprehensive requirements specification of the business, including information systems as integrated parts. It is a design of future business processes where IS functionality and IS action character are visible.

BPM consists of four primary activities:

- Prerequisites analysis,
- Activity analysis,
- Performer recognition, and
- Usage situation delineation.

Prerequisites analysis seeks to answer the questions 'why' and 'what for'. During this activity the aims and objectives of the development effort are established and the business problems to solve are identified and related to each other. Legacy demands, in terms of institutionalized 'hard to change' behaviour and legacy systems, are identified together with other constraints and prerequisites.

Activity analysis seeks to answer the questions of 'what, how, and when'. During this activity, the activities and activity structures of the business are analysed. Action objects (information and material) are identified and allocated to activities as prerequisites and/or results, and

are used a source when defining the action elementary messages (ae-messages, see chapter 4) used in the business.

Performer recognition seeks to answer the questions of 'by whom', 'for whom' and 'how many'. During this activity the business performers, i.e. human actors and information systems, and their roles, are analysed.

Finally, Usage situation delineation aims at grouping activities performed by – or in interaction with – information systems into appropriate and manageable analysis chunks, as an aid for ISM. This is where the concepts of *Interactive usage Situation Proposal* (ISP) and *Automatic usage Situation Proposal* (ASP) come into play.

Business process modelling, and its documentation forms, are described in detail in chapter 6.

## 5.5 Information Systems Focused Modelling

Information Systems focused Modelling (ISM) constitutes in-depth analysis and design of the computerized parts of the business under development, i.e. its information systems. Based on the interactive and automatic usage situation proposals (ISPs and ASPs) delineated during BPM, an analysis of the required IS support is performed. ISM yields a comprehensive requirements specification of information systems as coherent parts of the business. It is a design of future information systems based on the business actions that are to be performed with and within them.

ISM consists of three focal areas:

- Interaction analysis,
- Conceptual analysis, and
- Document analysis.

Interaction analysis constitutes an in-depth continuation of the Activity analysis of BPM. During this activity the elementary interactions and IS actions that aggregate to usage situations are in focus: which they are, how they are performed, and what sequence restrictions apply to them. Interaction analysis is thus concerned with the formulation and sending of ae-messages.

During Conceptual analysis the focus is on the propositional content of the ae-messages: their definitions and lifecycles as well as, for example, occurrence frequency. Conceptual analysis is performed at two different levels of abstraction: at *ae-message level* and at *system global level*. This is to ensure that (1) each ae-message contains the 'correct' propositional content and (2) the conceptual model of the system as a whole is consistent and unambiguous.

During Document analysis the somewhat fragmented document model of Interaction analysis is consolidated to a system global level. The mapping from ae-messages to documents is analysed for consistency and navigation between different documents and ISPs is considered.

Information Systems focused Modelling, and its documentation forms, are described in detail in chapter 7.

## 5.6 VIBA'99 Versus VIBA'93

This section explains the main differences between VIBA'93 and VIBA'99. Only high-level differences are commented upon in this section – details, for example differences in documentation forms, are treated in chapters 6 and 7.

VIBA'93 consisted of four main focal areas (Goldkuhl, 1993): Business analysis (in Swedish: Verksamhetsanalys), Message analysis (in Swedish: Meddelandeanalys), Processing analysis (in Swedish: Behandlingsanalys), and Effect analysis (in Swedish: Effektanalys); see Fig 5-4.

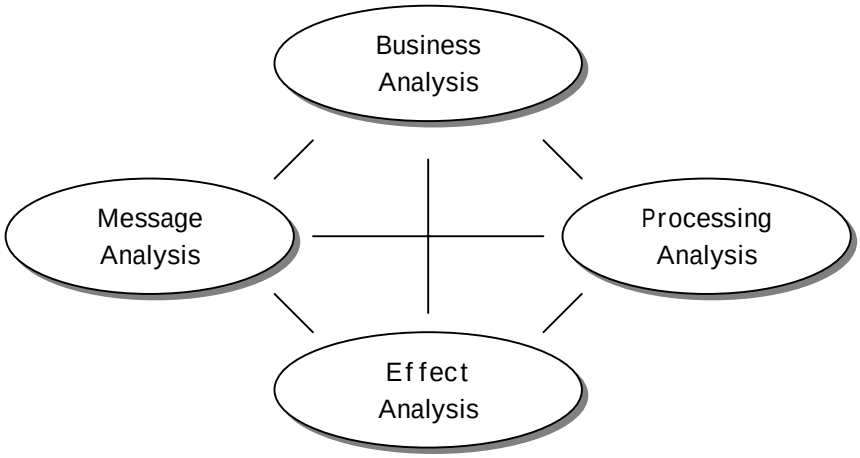



**Fig 5-4:** Main focal areas of VIBA'93 (Goldkuhl, 1993).

### 5.6.1 Business Analysis

The Business analysis of VIBA'93 corresponds roughly to the Business process modelling in VIBA'99. The aims of the two are the same, but the activities performed are grouped somewhat different. Business analysis consists of four primary activities:

- Goals inventory (in Swedish: Målaktualisering),
- Business structuring (in Swedish: Verksamhetsstrukturering),

- Computer systems overview (in Swedish: Datasystemsammanställning), and
- Work task analysis (in Swedish: Arbetsuppgiftsanalys).

Goals inventory corresponds to the BPM activity Prerequisites analysis. The name has been changed to reflect the fact that an inventory of goals is actually performed as one part amongst others during this activity.

Business structuring corresponds to the BPM activity Activity analysis. The name has been changed to emphasize that it is the structure of activities (action aggregates) that is analysed.

During Computer systems overview, the different computerized information systems found during Business structuring are described in terms of different functions (messages sent and received) and involved actors.

During Work task analysis, the different tasks to be carried out are described as well as the actors performing the tasks.

These last two activities have been regrouped in VIBA'99 to better match the concept of usage situation as proposed by ATIS (see chapter 4). In essence, a work task corresponds to a usage situation and hence the activity Usage situation delineation corresponds to a large part of Work task analysis. However, we have chosen to treat all performers (actors and information systems) together during Performer recognition in VIBA'99, instead of treating information systems during Computer systems overview and actors during Work task analysis as done in VIBA'93. This is more consistent with the concept of *performer* as suggested by ATIS (see chapter 3).

### 5.6.2 Message Analysis and Processing Analysis

In VIBA´93 Message analysis consists of four activities:

- Message definition (in Swedish: Meddelandebestämning),
- Concept analysis (in Swedish: Begreppsanalys),
- Information structuring (in Swedish: Informationsstrukturering), and
- Dimensioning (in Swedish: Dimensionering).

In addition, Processing analysis consists of four activities:

- State analysis (in Swedish: Tillståndsanalys),
- Processing design (in Swedish: Utformning av behandlingsstruktur),
- Dialogue design (in Swedish: Utformning av dialogstruktur), and
- Processing rules definition (in Swedish: Bestämning av behandlingsregler).

The aim of Message definition is to define the messages sent and used through information systems in the business. This activity is performed during the Activity analysis of BPM in VIBA'99. To treat this as part of Activity analysis seems more appropriate since messages are considered as results of actions (see chapter 4) – actions that aggregates to activities. In VIBA'99 it is also important to consider message definitions during Usage situation delineation, which is performed during BPM, as a bridge to ISM.

Concept analysis, i.e. defining the concepts used, Information structuring, Dimensioning, and State analysis, constitute together the Conceptual analysis of ISM in VIBA'99.

Processing design, Dialogue design, and Processing rules definition, have been replaced by Interaction analysis and Document analysis in VIBA'99. As shown in chapter 7, these two activities constitute a better match with our notion of formulation and sending of ae-messages, as discussed in chapter 4; they are actually directly derived from that notion.

### 5.6.3 Effect Analysis

Effect analysis consists of three activities

- Economic valuation (in Swedish: Ekonomisk värdering),
- Work situation analysis (in Swedish: Arbetssituationsanalys), and
- Design valuation (in Swedish: Designvärdering).

In VIBA'99 we have chosen not to treat this as a separate focal area with separate activities. Instead, as discussed in section 5.2, these are aspects to consider during a continuos evaluation. However, if time permits, it is possible to adopt the Effect analysis of VIBA'93, and perform a comprehensive analysis of the effects of the proposed design. This might be specifically useful when a waterfall model is used during development, since there are no ways of adjusting the requirements during later phases in such cases. However, it is beyond the scope of this thesis to go into details of Effect analysis and we therefore refer to Goldkuhl (1993).

# Chapter 6

# Business Process Modelling

In this chapter we describe the first main focal area of VIBA/SIMM: Business Process Modelling (BPM). As discussed in chapter 5, BPM is logically divided into four engineering activities: Prerequisites analysis, Activity analysis, Performer recognition, and Usage situation delineation. Each of these has a separate section of this chapter devoted to it. Throughout the chapter, excerpts from Case II (see chapter 2) will be used as examples. The chapter ends with a discussion about differences between VIBA'93 and VIBA'99, and a discussion of the grounding of BPM in the Action Theory of Information Systems.

## 6.1 Change Analysis and VIBA

As discussed in chapter 5, if VIBA is proceeded by Change analysis/SIMM (Goldkuhl & Röstlinger, 1988; 1993; Lind, 1996), some VIBA work has already been performed. Consequently, some of the documentation forms presented in this chapter are borrowed from Change analysis. However, Change analysis considers general business changes, while VIBA specifically considers information systems engineering. Hence, it is important to re-evaluate and probably refine the Change analysis documentation, from an information systems perspective. The documentation forms that are borrowed from Change analysis are:

- Problem lists and Problem Diagrams (see section 6.2),
- Goal lists and Goal diagrams (see section 6.2),
- Action diagrams (see section 6.3.2).

## 6.2 Prerequisites Analysis

The aim of Prerequisites analysis is to bring an understanding of the business under development, and to make that understanding inter-subjective among the different persons involved (engineers, users, system owners, *et cetera*). One way to start is to produce a Business definition as shown in Fig 6-1.

A Business definition defines the results (products and services) of the business, what customers and clients the result is produced for, the main tasks that have to be carried out to create the results, and the groups of persons involved in the process. In addition, it might be a good idea to include information about important raw material and suppliers as well. The main purpose of the Business definition is to ensure that developers do not 'forget' the business; its intended results used to satisfy customers, and the activities performed to create such satisfactions. Doing this early during systems engineering emphasizes that information systems are developed with the purpose of adding value to the business – and thus to the business' customers (Goldkuhl, 1993).

**Series**
The Paper Mill

**BUSINESS DEFINITION**

| **Creator** | **Date** | **Version** | **Ref. Code** |
|---|---|---|---|
| PÅ | 4.4.99 | 2 | BD1 |

**Concerns:** Material Administration System (MA)

| | |
|---|---|
| **Products:** | Tissue and Laminate |
| **Customers:** | Mostly other companies within the same combine. Occasionally to other large producers of consumer products. |
| **Tasks:** | Refine pulp and sell as tissue and laminate. This includes purchasing pulp, colouring, glue etc., planning and carrying out production, warehousing, and providing products to customers. |
| **Producers:** | Salespersons, purchasers, schedulers, workers and economy personnel. |

**Fig 6-1:** Example Business Definition.

Before development actually starts, it is important to get an understanding of the problems that the new system is supposed to solve. Thus, problems are to be identified, formulated and analysed during a *Problems inventory*.

The problems found can be documented in a Problem list as shown in Fig 6-2.

It is usually not enough to list problems, one needs to analyse how problems relate to each other as well. That is, to identify causes and effects of problems. The reason for undertaking such analysis is to structure problems in 'problem hierarchies' in order to understand which are the 'real problems', i.e. the central problems that are the source of all the trouble; these are the problems that are most important to solve.

**Series**
The Paper Mill

**PROBLEM LIST**

| Creator | Date | Version | Ref. Code |
|---|---|---|---|
| PÅ | 4.4.99 | 4 | PL1 |

**Concerns:** Material Administration System (MA)

1. Lots of information must be remembered and processed manually by the staff.
2. Communication between sales dept. and production dept. is often too ineffective. The right people are often hard to reach when needed.
3. Delivery promises are hard to live up to. Often deadlines need to be changed and re-negotiated with customers.
4. Production schedules are changed frequently. Sometimes even for production weeks that have already begun.

**Fig 6-2:** Example Problem List.

The analysis of problem causes and effects can be documented in a Problem diagram; see Fig 6-3.

**Series**
The Paper Mill

**PROBLEM DIAGRAM**

| Creator | Date | Version | Ref. Code |
|---|---|---|---|
| PÅ | 4.4.99 | 4 | PG1 |

**Concerns:** Material Administration System (MA)

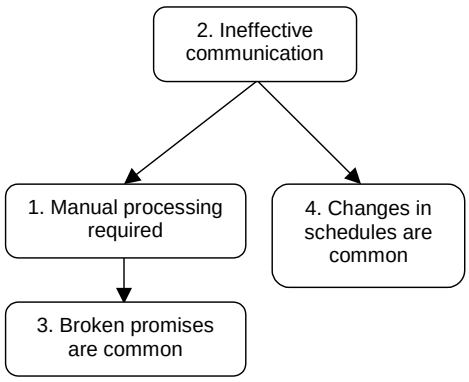



**Fig 6-3:** Example Problem diagram.

A Problem diagram consists of a directed graph (cf. e.g. Lipschutz, 1976) where the nodes represent problems and the vertices represent causal relationships between problems.

A Problem diagram graph must not necessarily be a rooted tree (as in Fig 6-3). It might in fact even be disconnected.

The most important problems to solve are those represented by nodes having indegree equal to zero (sources), referred to as *root problems*. Nodes of a Problem diagram graph having outdegree equal to zero (sinks) correspond to problems that are usually solved 'on the fly' when the harder problems (closer to root problems) are considered, given that their indegree is greater than zero.

To develop (re-engineer) a business is (or at least ought to be) to move away from the problematic and approach the desired. Therefore it is not enough to analyse the problems. One also needs to examine the business' goals during a *Goals inventory*. Goals might exist on different levels, from high-level goals such as strategies and policies to low-level goals such as norms and business rules for a particular business process or activity. During BPM it is important to concentrate on the goals that specifically relate to the IS to be developed in order to keep focus on the task at hand, i.e. systems engineering.

Goals can be documented in a Goal list, similar to problems in the Problem list (Fig 6-2).

Similar to problems, goals have to be analysed in terms of causes and effects. This is to discover which goals are the most important to achieve in order to decide where to put most engineering resources. Furthermore, it is possible that some goals contradict each other. Such 'goal conflicts' are important to identify and to solve, if possible.

The analysis of goal causes, effects and conflicts can be documented in Goal diagrams. A Goal diagram consists of a directed graph with nodes representing goals and with two different kinds of vertices: one kind representing goal/sub-goal relationships, and the other representing goal conflicts. As a notational convention Goal diagram graphs are drawn in the opposite direction of Problem diagram graphs – having the most prominent goals at the top and sub-goals beneath. Hence, nodes with outdegree equal to zero are the most prominent (important to achieve) as opposite to Problem diagrams. Fig 6-4 shows the notation used in Goal diagrams.

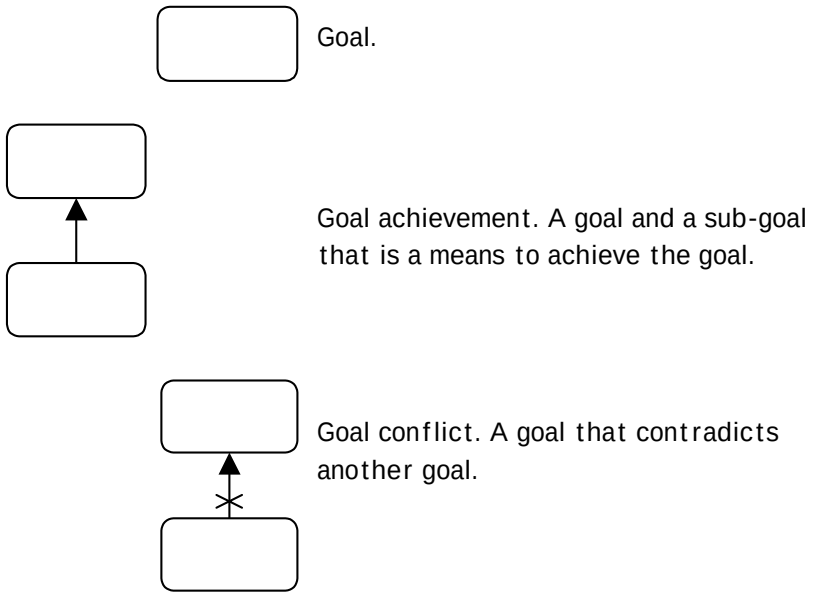



**Fig 6-4:** Notation used in Goal graphs (Goldkuhl & Röstlinger, 1988).

As discussed previously (chapter 5 and section 6.1), VIBA should optimally be preceded by Change analysis/SIMM (CA). During CA the problems and goals of (the part of) the business that the new IS supports are treated as described above. The results of CA can then be used directly after filtering goals and problems that obviously do not relate to the IS being developed. Otherwise this part of VIBA (i.e. Problems inventory and Goals inventory) must be worked through more carefully.

In addition to the business *per se* and its problems and goals, there are in general a number of different prerequisites that delimit the engineering work but also give rise to different possibilities. It is very important to make such prerequisites explicit early during systems engineering.

Prerequisites might be of different kinds. Goldkuhl (1993) lists some examples, such as:

- strategic,
- organizational,
- hardware,
- software,
- security,
- legal,
- work environment,
- integration with other (legacy or new) systems, *et cetera*.

Identified prerequisites can be documented in a Prerequisites list as shown in Fig 6-5.

**Series**
The Paper Mill

**PREREQUISITES LIST**

| Creator | Date | Version | Ref. Code |
|---|---|---|---|
| PÅ | 4.4.99 | 6 | PL1 |

**Concerns:** Material Administration System (MA)

**Hardware**

The system shall be implemented on the departments existing personal computers running Windows NT. Existing relational database management systems running on UNIX mainframes shall be used.

**Software**

The system shall be constructed using Visual Basic following Microsoft's standards as much as possible.

**System integration**

The system shall communicate with the existing sales support system (called ISP). Some data are to be imported from ISP and some (not existing) are to be owned by the new system.

**Time**

The system shall be running within 4 months.

**User experience**

Experience of computers differs widely between different groups of users. Special attention shall be paid to make interfaces easy to learn and to understand by all users.

**Fig 6-5:** Example Prerequisites list.

The Prerequisite list, together with the Problem list and the Goal list, is the most important tool in guiding the systems engineering process.

## 6.3 Activity Analysis

As stated in chapter 1 the method context of VIBA is business information systems within process-oriented organizations. A business process consists of activities ordered in a structured way with the purpose of

producing valuable results for customers. Different persons within the organization (and even outside) can perform such activities. Customers must perform certain actions, such as, for example, inquiring and ordering. Some actions of the organization can be performed by computerized information systems.

Modelling and designing business processes means describing different (types of) actions that should be performed within a business process. The process logic should capture how different actions are related to each other and different 'firing conditions' for actions.

### 6.3.1 Actions and Activities in a Business

Actions can be classified as *material actions* or *communicative actions* (see chapter 3). When discussing the concept of e-action in chapter 4, we referred implicitly to the latter, i.e. communicative actions. In a business firm many actions performed are however material actions, for example changing the location of some raw material in a warehouse. Note that in practice, an action might be both material and communicative. Moving the raw material might be done to explicitly state where raw material of a certain kind shall be kept, for example. Furthermore, when analysing business processes during a systems engineering effort it is usually of no interest to dig into subtle details of the actions performed. Therefore we use the term 'activity', which is thought of as an action aggregate of one or more actions – material or communicative. An activity might thus 'produce' one or more ae-messages as well as one or more interventions in the physical state of the business. The concept of e-action as a more fine-grained analytical tool is thus deferred till Information systems Focused Modelling (ISM).

### 6.3.2 Capturing and Describing Activity Structures

There are several techniques available to describe business processes such as, for example, sequence diagrams of the Unified Modelling Language (UML; eg. Booch *et al*, 1999) and data flow diagrams of Structured analysis (Yourdon, 1989). To describe business processes we propose to use Action diagrams of the SIMM family of methods (Goldkuhl, 1992; 1993), due to insufficient semantics in other approaches.

In Action diagrams different activities of a business process and how these activities are related to each other are explicitly described. Actions performed by human actors as well as information system actions are considered. Note that Action diagrams describe types of activities, and not the actual occurrences. Action diagrams can be used to describe material flow and information flow within a business process; see Fig 6-6.

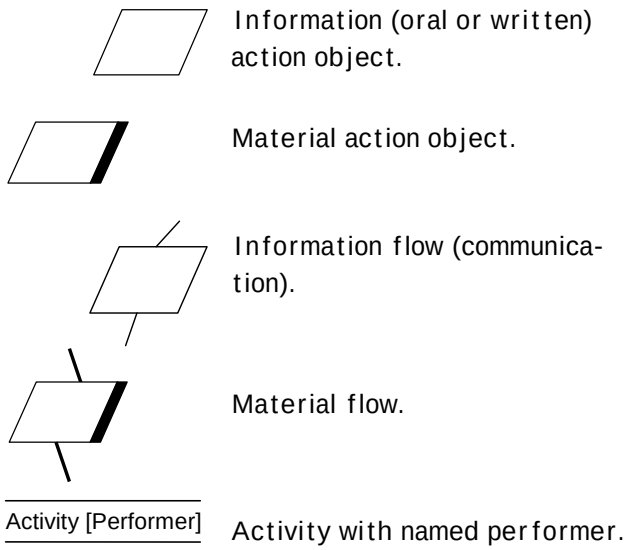


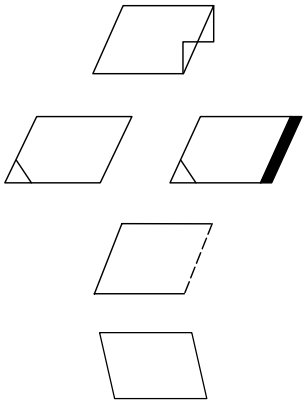

**Fig 6-6:** Basic notation used for activities, action objects and flows in Action diagrams.

Material and information (as action objects) are described and related to activities as prerequisites (input) or results (output). One important notational feature is that Action diagrams describe the performer of each activity; i.e. what actor/actor group (actor role) or which IS that is supposed to perform the particular activity.

In addition to the basic notation for action objects shown in Fig 6-6, it is possible to, as shown in Fig 6-7, distinguish between:

- Encapsulated information (i.e. not directly readable by humans),
- Stores of action objects (used when activities add action objects to already existing quantities), and
- Knowledge (i.e. not externalized information that exists within some human actors mind)

Some activities (in Action diagrams) are delineated to be interactive with several performers, e.g. [Customer → Salesman ↔ System]. Interactive activities might be one-way [A → B] (from one performer to another) or two-way [A ↔ B] (a dialogue between performers), see Fig 6-8. The two-way interaction is the most common one but one-way interaction is typically the case with, for example, reports and orders.



Encapsulated information.

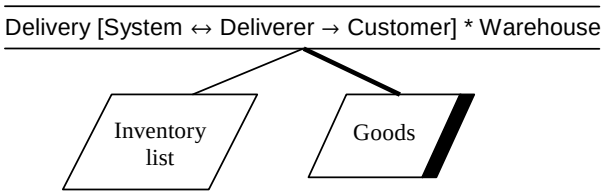
Store of action objects. Applicable to both information and material.

Knowledge, which has not been externalised.

Non-action. An action object that is the result of an omitted action (i.e. an omission).

**Fig 6-7:** Special notation used for encapsulated information, stored action objects, knowledge, and non-action in Action diagrams.

In addition to performers, the physical location where an activity is to be performed can be shown in Action diagrams by adding an asterisk and the name of the location after the performer list (see Fig 6-8).




Activity with several performers, named location and resulting action objects.

**Fig 6-8:** Notation used to show several performers and activity location in Action diagrams.

One important aspect of Action diagrams is their semantic power to describe action logic (see Fig 6-9). It is possible to describe sequential ordering of activities (i.e. the flow aspect), alternative activities (decision points), conjunctive activities (combinations), triggering (initiation) of activities (by time or communication), interruption of activities (by time or communication), contingent activities (i.e. activities occurring only sometimes), condition for activities, and parallel activities.

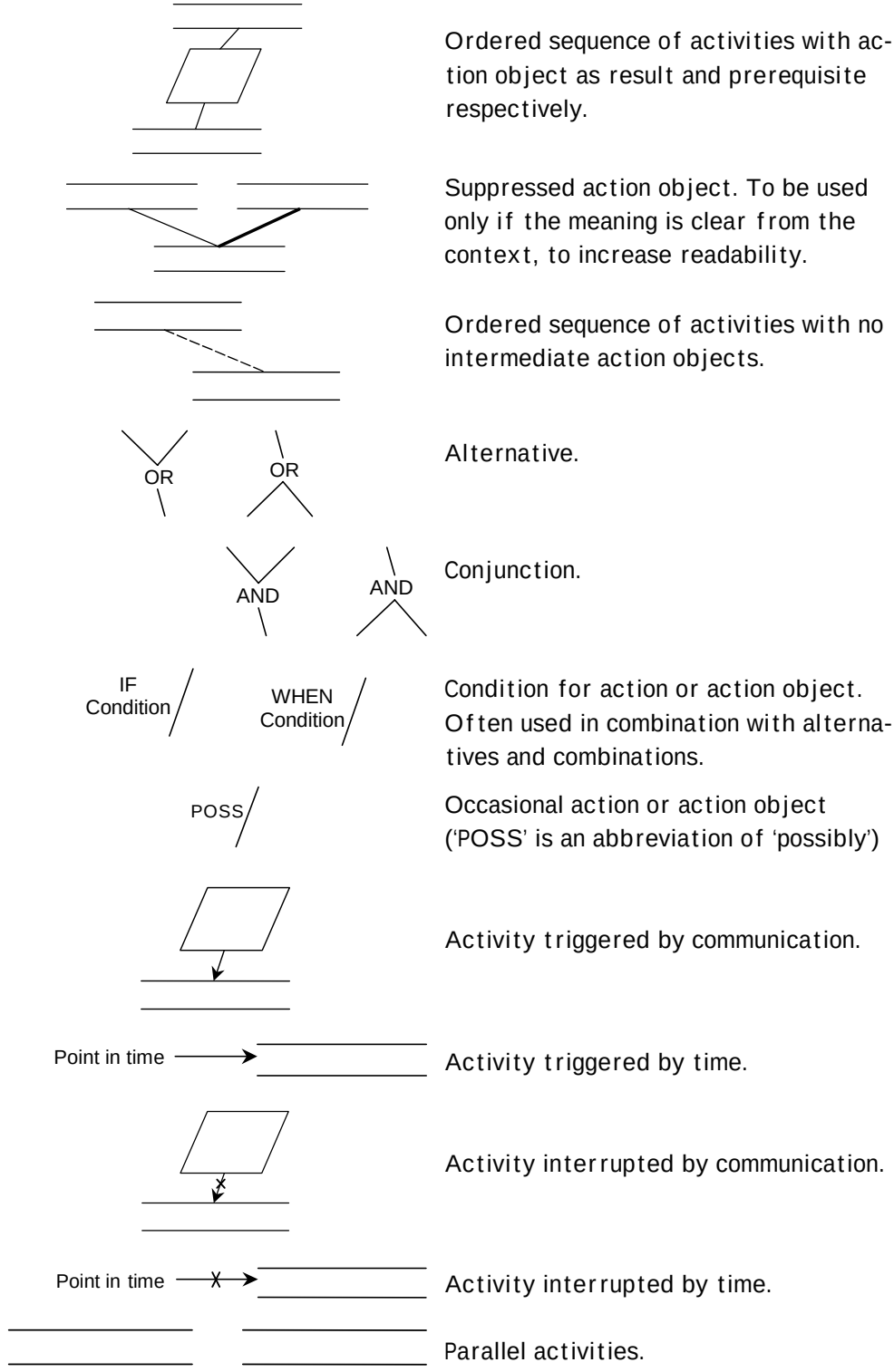
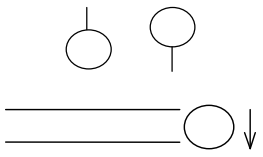



**Fig 6-9:** Basic notation to describe action logic in Action Diagrams.

A contextual descriptive approach is to be preferred when working with Action diagrams. Each Action diagram describes a business context within a business process. Different Action diagrams are related to each other through descriptive connectors (i.e. links to other Action diagrams,

see Fig 6-10). The demarcations of each Action diagram (i.e. business context) are arbitrary; thus, an analyst is free to choose appropriate boundaries for each described context.



Sequential connector to/from another Action diagram. The Ref. Code of the connected diagram is shown in the circle.

Hierarchical connector to another Action diagram that shows a decomposition of the connected activity.

**Fig 6-10:** Connectors used in Action Diagrams.

Action diagrams make it possible to model and design the business and its information systems as an integrated whole. The actions of different performers – human actors and/or information systems (internal and/or external to the organization) – are described as a whole. Action diagrams can thus be used to identify and delimit IS action. Such actions are described as integral parts of the business process. To say that IS actions are derived from the business process design is one way to put it. That business process design includes design of IS actions is equally correct. This counts for both interactive IS action (performed together with a user) and automatic IS action (performed by the computerized IS itself). The Action diagrams also show activities performed as a consequence of IS action. These IS-related actions form what we call *usage situations*, which can be classified as (1) *interactive*, (2) *automatic*, or (3) *consequential* (cf. chapter 3), and are used as a basis for delineating Interactive and Automatic usage Situation Proposals (see section 6.5).

Fig 6-11 and Fig 6-12 show examples of Action diagrams describing two connected business contexts. The first diagram (PurInq, see Fig 6-11) describes how feasibility checking of raw material is performed at The Paper Mill. Feasibility checking is performed to ensure that enough raw material is either in stock or ordered before an enquiry or order from a customer is acknowledged. The second diagram (PurAck, see Fig 6-12) describes how purchase order acknowledgements from suppliers, and deliveries, are received and registered. In addition, the second diagram also shows an activity 'Move raw material' that might occur when receiving raw material, but which might well occur in isolation. This exemplifies the freedom to choose arbitrary boundaries for business contexts during analysis.

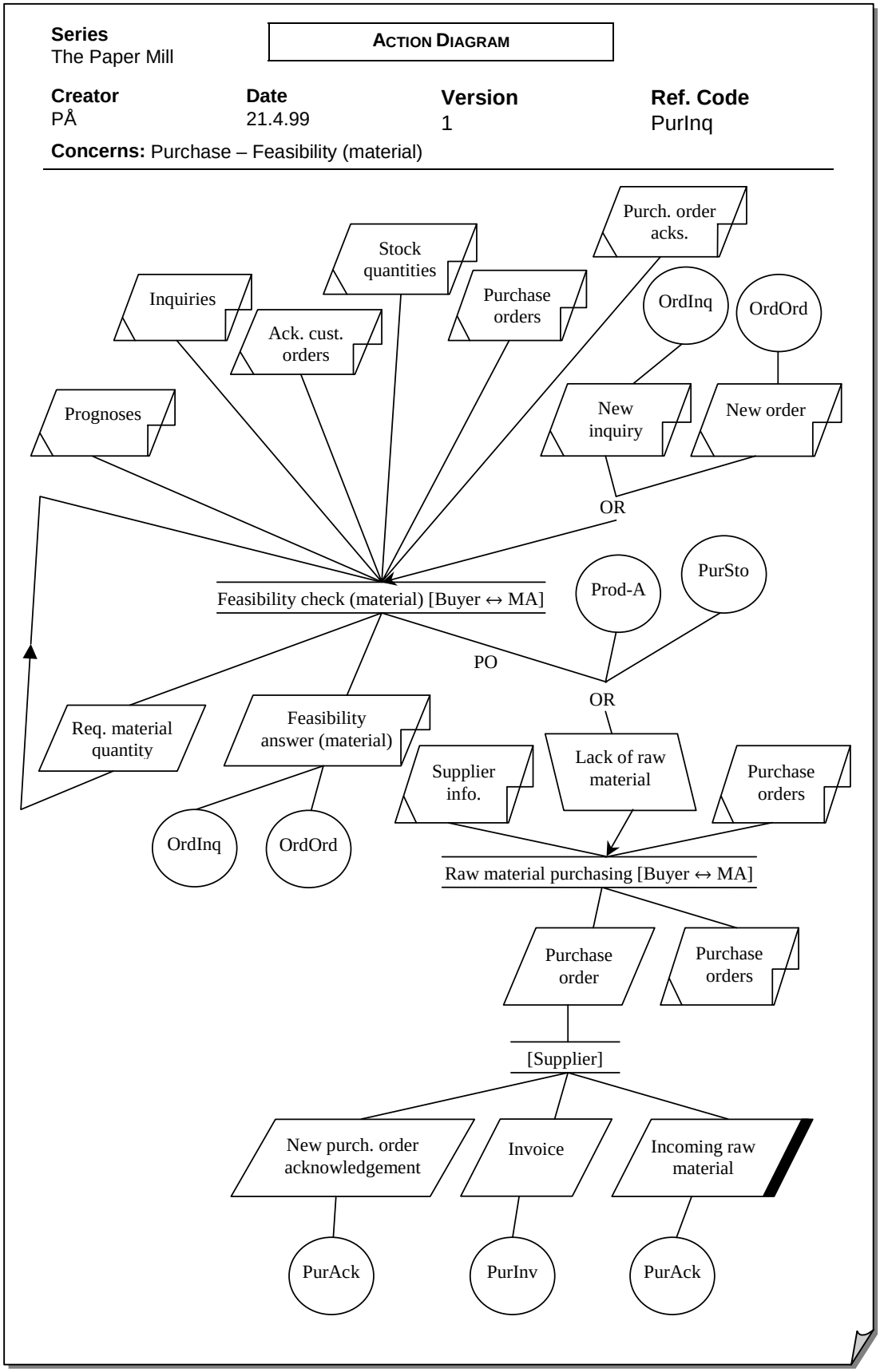



**Fig 6-11:** Example Action Diagram I.

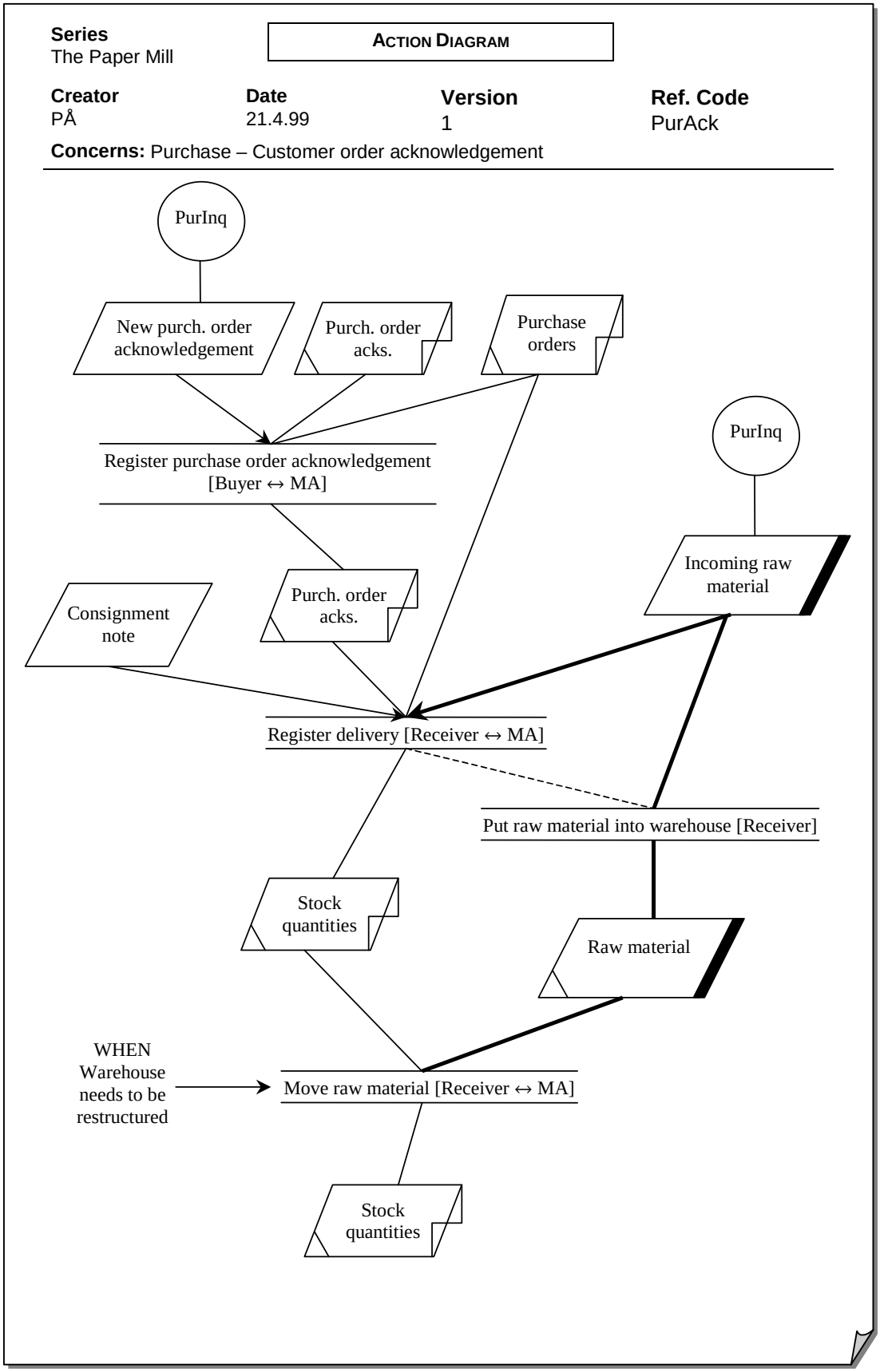



**Fig 6-12:** Example Action Diagram II.

### 6.3.3 Messages and Documents

When Activity analysis has proceeded so far that the Action diagrams seem reasonably stable it is time to take a closer look at the messages being communicated in the business process. Messages (or ae-messages to be correct) are identified in the Action diagrams as information action objects flowing between activities. Information action objects that correspond to ae-messages must be identified and explicitly defined in Message definitions (see Fig 6-13).

**Series**
The Paper Mill

**MESSAGE DEFINITION**

| Creator | Date | Version | Ref. Code |
|---|---|---|---|
| PÅ | 29.4.99 | 3 | MD2 |

**Concerns:** Message 2 **-** Purchase order

**Description:** A purchase order is sent from The Paper Mill to internal or external suppliers to order raw material for use in production.

**E-action:** 2 **–** Purchase order

**E-action character:** Interactive

**E-action dependence:** Action memory affecting

**Communicator:** The Paper Mill – Purchase dept.

**Performer role:** Buyer

**Communication function**
Primary: Order raw material
Secondary: Maintain information about forthcoming raw material deliveries

**Communication effect**
Primary: Delivery of raw material
Secondary: -

**Creating document:** DD2 – «ISD» Purchase order

**Content**

| | | |
|---|---|---|
| Heading: | Purchase order ID | aPurchase order.ID |
| | Date | aPurchase order.Date |
| | Our reference | aPurchase order.OurRef.Name |
| | Terms of delivery | aPurchase order.TermsOfDel |
| | Terms of payment | aPurchase order.TermsOfPay |
| Supplier: | Supplier ID | aPurchase order.Supplier.ID |
| | Name | aPurchase order.Supplier.Name |
| | Address | aPurchase order.Supplier.Address |
| | Zip code | aPurchase order.Supplier.Zip |
| | Phone | aPurchase order.Supplier.Phone |
| | Fax | aPurchase order.Supplier.Fax |
| | Reference | aPurchase order.Supplier.Reference |
| Detail: | Art. No. | aPurchase order.PO Row.Raw material.ID |
| | Name | aPurchase order.PO Row.Raw material.Name |
| | CC | aPurchase order.PO Row.Raw material.CC.ID |
| | Quantity | aPurchase order.PO Row.Quantity |
| | Price | aPurchase order.PO Row.Price |
| | Unit | aPurchase order.PO Row.Raw material.Unit |
| | Del. Date. | aPurchase order.PO Row.DelDate |
| | Req. spec | aPurchase order.PO Row.ReqSpec |

**Fig 6-13:** Example Message Definition.

Message definitions are important documents since much of the work during ISM centres around them. A Message definition states what e-action the ae-message is a result of, the character of that e-action, how the e-action depends on the action memory, the communicator of the message, the performer of the e-action, communication function and effect, and what document creates it. Furthermore it states the propositional content of the message. This can be done in different ways, and typically the amount of detail increases as the analysis proceeds. At first, there might be textual references to concepts and properties. However, when, during ISM, the propositional content is analysed in terms of classes, references can be made more specific. In Fig 6-13, the propositional content is given with a label and an expression in Object Navigation Notation (ONN), as described by Blaha & Premerlani (1998). By using ONN, and explicitly traversing the classes of the Class diagram (see section 7.3) for the message, a verification of the information content is performed. This is however considered to be a part of Conceptual analysis during ISM.

A message definition also contains a brief description of the message and how it is used in the business.

As stated in chapter 4, ae-messages are created, carried and visualized by documents. At this point it is not possible to specify all documents that are going to be used in the IS. Especially not interactive screen documents (ISDs), which are solely treated during ISM. Some documents might however be identified during BPM – typically paper documents, for example reports and other 'output documents'. Documents are explicitly defined in Document Definitions (see Fig 6-14).

**Series**
The Paper Mill

**DOCUMENT DEFINITION**

| **Creator** | **Date** | **Version** | **Ref. Code** |
|---|---|---|---|
| PÅ | 29.4.99 | 3 | DD2P |

**Concerns:** «PD» Purchase order

**Description:** A Purchase order paper document is what is physically sent from The Paper Mill to internal or external suppliers to order raw material for use in production.

**Message(s) carried:** MD2 – Purchase order

**Creating document:** DD2 – «ISD» Purchase order

**Used in ISP(s):** ISP 3 – Raw material purchasing

**Prototype:** DP2P

**Fig 6-14:** Example Document Definition for a paper document.

A Document definition for a paper document states which messages are being carried by the document, who is the creator of the document, in which ISP(s) it is being used, and a reference to a prototype of the document. Furthermore, as message definitions, it contains a brief description of the document and its intended use.

The creator of a paper document is either an interactive screen document or an automatic usage situation. If the creator of a paper document is an ISD the creator information probably has to be left blank until later in the engineering process.

Note that not all the information showed in the example documents of Fig 6-13 and Fig 6-14 can be explicitly stated during BPM (for example the references to classes in the content-part of Message definitions). Instead these documents are carried on to ISM and continuously refined.

## 6.4 Performer Recognition

Performers, communicators, interpreters, and the roles they play, are identified during Activity analysis. The purpose of Performer recognition is to explicitly define these, and to analyse relationships between them.

In Action diagrams, the performer part of activities are either information systems[1] or roles played by the different business actors.
Important relationships to discover between roles are IS-A (i.e. inheritance in object-oriented terminology). For example a buyer might be able to act as receiver of goods but not the other way around. Hence 'Buyer' can be regarded as a special case of 'Receiver', i.e. a 'Buyer' IS-A 'Receiver'. These relationships are documented in Role diagrams and the roles are defined in Role definitions. These two documentation forms are similar to the Class diagrams and the Class definitions used during ISM (see chapter 7). Actually, following UML, a role might be regarded as a special kind of class (at the meta-level); in UML's terminology 'role' is referred to as 'actor', however. When it comes to design, the roles identified and structured during Performer recognition can be used as a source for maintaining access privileges to different parts of the constructed system(s).

[1] Actually it should be thought of as a role played by an information system. For example, MA (which stand for 'the material administration system') of Fig 6-11 can be thought of as a role played by an IS 'the administrative system'. However the distinction is not important until the design-phase of systems engineering when functionality is to be mapped onto actual systems.

During Performer recognition it is also important to investigate the cardinality of different roles, and, if possible, enumerate the actual performers cast to play each role.

## 6.5 Usage Situation Delineation

The purpose of Usage situation delineation is to group together activities from the Action diagrams into (1) Interactive usage Situation Proposals (ISPs) during ISP Delineation, and (2) Automatic usage Situation Proposals (ASPs) during ASP Delineation. That is, to group activities into chunks that are manageable from an analysis and design perspective.

### 6.5.1 ISP Delineation

In chapter 4, we defined ISU to be a primitive sequence of business actions that consists of all interactive actions, and intermediate manual and automatic actions, that are performed adjacent in time at the same place by the same performers. This definition also counts for ISPs. ISPs are thus found by selecting arbitrary interactive activities in the Action diagrams and traversing the flow of activities, in both directions (preceding and succeeding), until all adjacent activities, in each direction, not in accordance with the definition are found.

The identified ISPs are given consecutive identifying numbers and are documented in an ISP List, as shown in Fig 6-15.

As shown in Fig 6-15, an ISP List consists of one table per ISP stating:

- A descriptive name of the ISP,
- the performers involved in the ISP,
- the messages needed to perform the ISP (input messages) and the messages that are results of the ISP (output messages),
- the documents used and/or produced during the ISP, and finally
- a set of references to the Action diagrams in which the ISP is described/used.

When viewing an ISP as a whole, there might be messages that do not seem to be outputs but are used ‘ISP internally’. Such messages can be treated as both outputs and inputs since they are outputs of some activity, or activities, within the ISP and inputs to some other.

The ISP List is an important document since it constitutes a cross-reference between the Action diagrams and the more detailed documentation produced during Information Systems focused Modelling (ISM).

If the current engineering effort embraces several information systems it is a good idea to give ISPs numbers such that the ISP List is

grouped by IS. This is however not always possible due to the fact that a single ISP might involve several information systems.

**Series**
The Paper Mill

**ISP LIST**

| Creator | Date | Version | Ref. Code |
|---|---|---|---|
| PÅ | 20.4.99 | 2 | ISPL |

**Concerns:** Material Administration System (MA)

| ISP 1 | Feasibility check (time) | | |
|---|---|---|---|
| **Performers** | **Input msg.** | **Output msg.** | **Document(s) used** |
| Planner ↔ MA | Coarse plan,<br>Inquiries,<br>Prognoses,<br>Ack. cust. orders,<br>Mtrl. structures, | Feasibility answer (time) | «ISD» Feasibility,<br>«ISD» Planning,<br>«ISD» Details planning,<br>«SSD» Quality spec,<br>«SSD» New inquiry,<br>«SSD» RE: New inquiry |
| **Action Diagram Ref. :** OrdInq, OrdOrd, PlaFea | | | |

| ISP 2 | Feasibility check (material) | | |
|---|---|---|---|
| **Performers** | **Input msg.** | **Output msg.** | **Document(s) used** |
| Buyer ↔ MA | Prognoses,<br>Inquiries,<br>Ack. cust. orders,<br>Stock quantities,<br>Purchase orders,<br>Purch. order acks. | Feasibility answer (material) | «ISD» Material needs calculation |
| **Action Diagram Ref. :** PurFea | | | |

| ISP 3 | Raw material purchasing | | |
|---|---|---|---|
| **Performers** | **Input msg.** | **Output msg.** | **Document(s) used** |
| Buyer ↔ MA | Purchase orders,<br>Material need | Purchase order | «ISD» Purchase order,<br>«PD» Purchase order |
| **Action Diagram Ref. :** PurFea | | | |

**...**

**Fig 6-15:** Example ISP List.

### 6.5.2 ASP Delineation

When the ISPs are identified and documented, it is time to check the Action diagrams for automatic activities that are not included in any ISP.

This activity is referred to as ASP Delineation and is documented in an ASP List, similar to the ISP List.

## 6.6 BPM Versus VIBA'93

As pointed out in chapter 5, BPM bears considerable resemblance to Business analysis of VIBA'93. In chapter 5 we concluded that the activities are grouped differently, and that Message definition is now considered as part of Activity analysis and hence separated from Conceptual analysis. In addition to this, there have been several changes in the documentation forms, but several document types have also been retained. The document types that are unchanged (or at least very similar) are: Business definitions, Problem lists and diagrams, Goal lists and diagrams, Prerequisites lists, and Action diagrams.

In VIBA'93 there is no notion of 'document' in the way suggested by ATIS and VIBA'99. Consequently, there are no Document definitions either. Instead the *media* is declared together with the definition of each particular message. This is done in so-called Message summaries (in Swedish: Meddelandeöversikt). The Role diagrams and definitions, and the ISP and ASP lists, have no directly corresponding document types in VIBA'93. Instead there are IS summaries, Unit summaries, and Work task descriptions.

An IS summary describes what functions a particular IS is supposed to have; what messages it uses and produces, and for each message: its sender and receiver. In VIBA'99 this information is to be found in the ISP list, grouped by ISP instead of IS. Of course, the information can easily be extracted from the ISP list and grouped by performer if needed.

A Unit summary corresponds to a Role definition for organizational units. In VIBA'99 Role definitions are used for all performers (not just organizations), communicators, and interpreters. This is a consequence of the distinction between different kinds of performers, and the separation between communicator and performer, in ATIS. The relations between these are not clearly articulated in VIBA'93.

A Work task description contains a description of a particular activity. In essence, this is a more verbal description of a particular ISP, even though the notion of ISP is not present in VIBA'93.

## 6.7 Notes on The Relation Between ATIS and BPM

In this section we will discuss some important relationships between Business Process Modelling (BPM) and the Action Theory of Information Systems (ATIS).

### 6.7.1 Action as the Starting Point

The most obvious connection between BPM and ATIS is perhaps the central roles that e-actions and ae-messages play. During BPM the action structure of the analysed business processes is documented in Action diagrams. The Action diagrams then serve as a basis for delineation of usage situations (which is another central ATIS concept). The dual character of information (propositional content and action mode) as proposed by speech act theory, and consequently by ATIS, is fully considered when activities and messages are analysed in BPM.

Another central issue in ATIS is the distinction between performer and communicator. Communicative action is intentional and it is hence important to discuss and define all possible performers of actions during systems engineering. Since ATIS regards information systems as performers, it is important to distinguish between who performs the action and who is actually doing something, i.e. who is the performer and who is the communicator. This is regognized to be one of the most important parts of Activity analysis as discussed in section 6.3.3). This distinction is also elaborated during Performer recognition.

Another implication of the fact that action is considered intentional is the analysis of goals during Prerequisites analysis. Identified goals can serve as an analytical tool to verify that the business' goals are made operational by corresponding actions (cf. Ågerfalk & Åhlgren, 1999).

### 6.7.2 Business Action Theory and BPM

Business Action Theory (BAT) and the theory driven analysis of business processes promoted by it (see chapter 3) is incorporated implicitly in BPM. There are no specific engineering activities that relate explicitly to BAT. However, the idea is to use BAT, and its six-phased model, as an analytical tool when creating and evaluating the BPM models. By cross-checking the documentation with the BAT model, it is possible to ensure that no important business actions are missing from the design.

# Chapter 7

# Information Systems Focused Modelling

In this chapter we describe the second main focal area of VIBA/SIMM: Information Systems focused Modelling (ISM). As described in chapter 5, ISM is logically divided into three focal areas: Interaction analysis, Conceptual analysis, and Document analysis. Throughout the chapter, excerpts from Case II (see chapter 2) will be used as examples. The chapter ends with a discussion about differences between VIBA'93 and VIBA'99, and a discussion of the grounding of ISM in the Action Theory of Information Systems.

## 7.1 Introduction to ISM

The aim of Information Systems focused Modelling (ISM) is to produce a clear and unambiguous specification of information systems that satisfies the required actability. ISM is performed on a usage situation basis, i.e. the engineering activities are based on the identified ISPs and ASPs of Business Process Modelling (BPM); cf. section 6.5.

As discussed in chapter 4, analysis of user interaction is preferably performed as a mixed analytical and experimental process. Since human-computer interaction is performed in order to formulate and send action elementary messages (ae-messages), and ae-messages are the result of three simultaneous acts (utterance, propositional and illocutionary), ISM as a whole ought to be performed with an integrated approach. During ISM Usage situations are analysed with respect to e-actions (formulation and execution), concepts used in ae-messages (propositional content), and documents (carriers and creators of ae-messages). These three focal areas are analysed as an integrated iterative process. The process continues until the required actability is satisfied.

Furthermore, the fifth sub-act of speech acts, the perlocutionary act, must always be taken as the point of departure, as well as the destination. Thus, the reasons for performing the different e-actions shall be kept in mind and used as a tool for evaluation of different design solutions.

Fig 7-1 illustrates this approach.

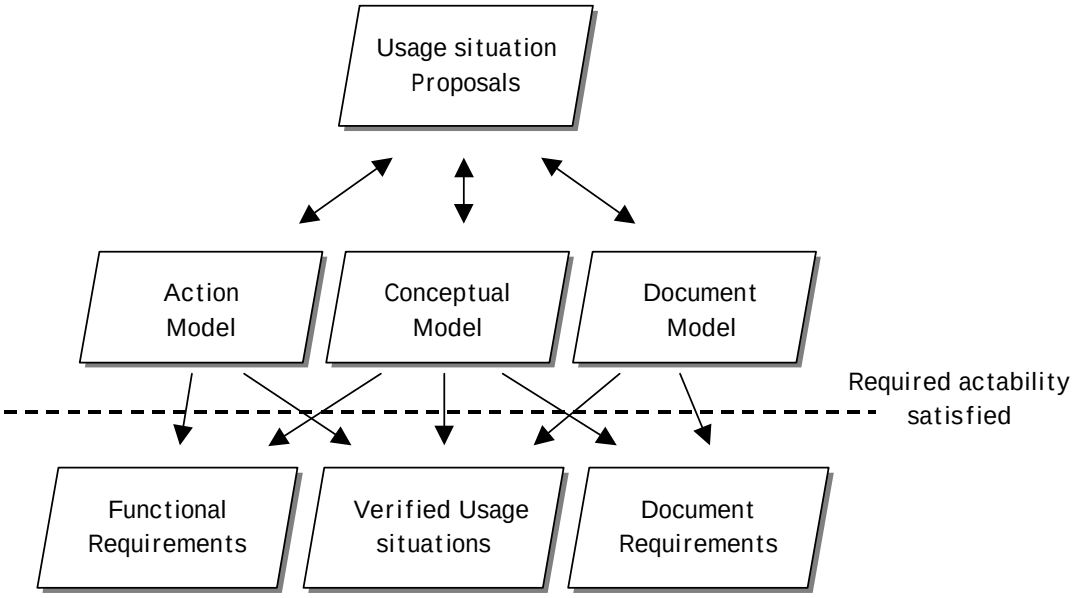



**Fig 7-1:** ISM as a mixed analytical and experimental process.

## 7.2 Interaction Analysis

To describe the e-interactions needed to formulate and send the messages of any given ISP, we propose to use Interaction Tables (I-Tables), which build explicitly on the EIAL (see section 4.2.3). An I-Table is a table with three rows and three columns. The left-hand column is used for the user actions, the middle column for the state of the current interactive document(s) and the right-hand column is used for the IS actions. One I-Table is used for each e-interaction (EIAL).

Fig 7-2 shows an E-Interaction list, which is the document type used to describe e-interactions. One E-Interaction list is produced for each ISP. An E-Interaction List consists of three parts. First there is a list of initial IS actions that are to be performed upon entry to the ISP. Second, there is a listing of the e-interactions to be performed in the ISP. Finally, there is one I-Table for each e-interaction that specifies the interaction in detail.

From the I-Table describing e-interaction 3.3 in Fig 7-2 we see how $s_0$, cell (1,2), specifies that the 'Purchase order' document, with information about the ID of the current order, the order date and the buyer's reference person, is to be visible in order to perform e-interaction 3.3, i.e. a precondition for the e-interaction. Cell (2,1) specifies what e-interaction the I-Table is concerned with, i.e. associating a supplier to a purchase order.

**Series**
The Paper Mill

**E-INTERACTION LIST**

| Creator | Date | Version | Ref. Code |
|---|---|---|---|
| PÅ | 26.4.99 | 4 | EI3 |

**Concerns:** ISP 3 – Purchasing

Initial system actions
S3.1 Retrieve purchase order with highest Order ID.
S3.2 Show info from order in all applicable fields.
S3.3 Fill cbPurOrderId with all available purchase order IDs in descending order.
S3.4 Fill cbSupId with all supplier IDs in ascending order.

E-interactions
3.1 Create new purchase order.
3.2 Retrieve earlier registered purchase orders.
3.3 Associate supplier with purchase order.
3.4 Print purchase order.
3.5 Register purchase order.
3.6 Change {Date, Our reference, PurOrderLine (dbgOrderLines)}.
3.7 Change {Your reference, Delivery details, Payment details}.
3.8 Cancel new purchase order.

| 3.1 | User action | Document | IS action |
|---|---|---|---|
| $S_1$ | **Create new purchase order** | **Purchase order**<br>"New order" (btnNewOrder) | 1. Clear all fields.<br>2. Generate new purchase order ID (next consecutive) and show in cbPurOrderID.<br>3. Show today's date in txtDate. |
| $S_2$ | **State fact:**<br>New purchase order created. | **Purchase order**<br>Order ID (cbPurOrderID)<br>Date (txtDate) | |

| 3.2 | User action | Document | IS action |
|---|---|---|---|
| $S_1$ | **Retrieve earlier registered purchase orders** | **Purchase order**<br>"Order ID" (cbPurOrderID) | 1. Retrieve order with selected ID.<br>2. S 2. |
| $S_2$ | **State fact:**<br>Selected purchase order retrieved and showed. | **Purchase order**<br>All order information visible | |

| 3.3 | User action | Document | IS action |
|---|---|---|---|
| $S_0$ | | **Purchase order**<br>Order ID, Date, Our reference | |
| $S_1$ | **Associate supplier with purchase order** | **Purchase order**<br>"Sup. ID" (cbSupID) | Retrieve information about selected supplier and show in all supplier-related fields. |
| $S_2$ | **Interpretation:**<br>1. Correct supplier associated.<br>2. Wrong supplier associated. (→ 3) | **Purchase order**<br>All supplier information visible. | |

**Fig 7-2:** Example E-Interaction List (Page 1 of 3 in a set of documentation).

Cell (2,2) specifies the state of the e-interaction after the user action and the specific GUI components used to perform the e-interaction. In this case a combo-box labelled 'Sup.ID'. The supplier is now associated to the purchase order (from the user's point of view). Column 3 shows the actions that are to be performed by the IS in response to the user's action. Cell (3,2) shows the response from the IS, i.e. the purchase order document with supplier information shown. Finally, cell (3,1) shows the interpretation act to be performed by the user. In this case the interpretation should yield either that the correct supplier has been associated with the purchase order or that another supplier must be chosen, which, in the latter case, leads to the re-performance of the same e-interaction.

The language used within I-Tables is natural language such as English or Swedish. We do however use some conventions that semi-formalize the language use, as described in Table 7-1.

| **Convention** | **Example** |
|---|---|
| Constraints regarding the performance of user actions are written surrounded by curly brackets in cell (2,1). | {By drag 'n drop 2→1} means that the user action is performed by use of drag 'n drop from document 2 to document 1 as numbered in cell (1,2). |
| Constraints regarding the performance of IS actions are written surrounded by curly brackets in column 3. | {RT ≤ 3s} means a response time equal to or less than 3 seconds. |
| Clickable items are GUI components used to execute the user action. These are surrounded by quotation marks in cell (2,2). | "New order" is a clickable item labelled 'New order' used to create a new order. |
| References to names given to GUI components in a prototyping tool might be shown after their 'visible' caption as shown in the GUI. | "Order ID" (cbPurOrderID) means that the clickable item 'Order ID' is named cbPurOrderID in the prototyping tool. This also indicates that it is a combo-box due to the 'cb'-prefix (using Hungarian notation). |
| To reference a sequence of IS actions the function sys can be used. | Sys(3.1) means that the IS actions stipulated in I-Table 3.1 shall be performed also here. |

**Table 7-1:** Notational conventions used in I-Tables.

In most cases the number of IS actions that correspond to a user action are one or perhaps two in strict sequence. Sometimes there might however be several IS actions involved and thus a need to describe how these relate to each other. In such cases, for example, a regular expression specifying the sequence, can be augmented to column 3. Sometimes it is preferable to instead use an Object Navigation Notation (ONN) expression (Blaha & Premerlani, 1998) that dictates what information to show, and thereby omitting to enforce any sequence restrictions. If se-

quence restrictions are important from a business perspective they shall be explicitly shown.

The granularity of which the documents are represented in I-Tables depends on how far the analysis has proceeded. At first, the documents are referred to by textual references (as in Fig 7-2). When the analysis proceeds the layouts of the documents get more and more explicit. This evolution can (and should) be shown in the I-Tables by inserting thumbnails of documents or document parts in column 2.

It is sometimes unnecessary to show the initial state $s_0$ within an I-Table, and hence the first row can be omitted, as in I-Tables 3.1 and 3.2 of Fig 7-2.

Each I-Table describes one e-interaction so there is a need to describe how these are related to each other. As described in chapter 4, the action potential of any given interactive screen document might vary over time. Since action sequence restrictions constitute the state of the document, and thereby restrict action potential, we use the formalism of Statecharts (Harel, 1987) to model them (see Fig 7-3 and Fig 7-4).

A related issue is the modelling of navigation paths between documents, which could be done with I-Tables but such an attempt would not be tractable due to the many possible navigation paths. Instead, this is also done with Statecharts where each high-level state represents a document and each navigational action is represented by a transition. The resulting state model can be viewed as consisting of documents as high-level states (Fig 7-3) where the action sequence restrictions of each document (Fig 7-4) correspond to sub-states of those.

Note that the Statechart of Fig 7-3 is somewhat simplified. Actually it should, of course, be possible to 'navigate' back from the different ISDs to the main form, as well as switch between different ISDs. Usually, it is not necessary to show such details in the system global Statechart, which purpose is to outline the overall navigation possibilities, at least not initially.

This approach to model action sequence restrictions is an extended version of Horrocks' (1999) Statechart approach to user interface construction. With Horrocks' approach there is a clear path from interface design to implementation in, preferably, event driven rapid development tools (e.g. MS Visual Basic and Borland Delphi). However, his approach lacks a clear connection to business actions, which is provided by the use of e-interactions as a source for transitions rather than Horrocks' vaguely defined notion of event.

Since our approach is an extension of Horrocks' (*ibid.*) the techniques and help-documentation suggested, and thoroughly discussed, by Horrocks might help when performing Interaction analysis.

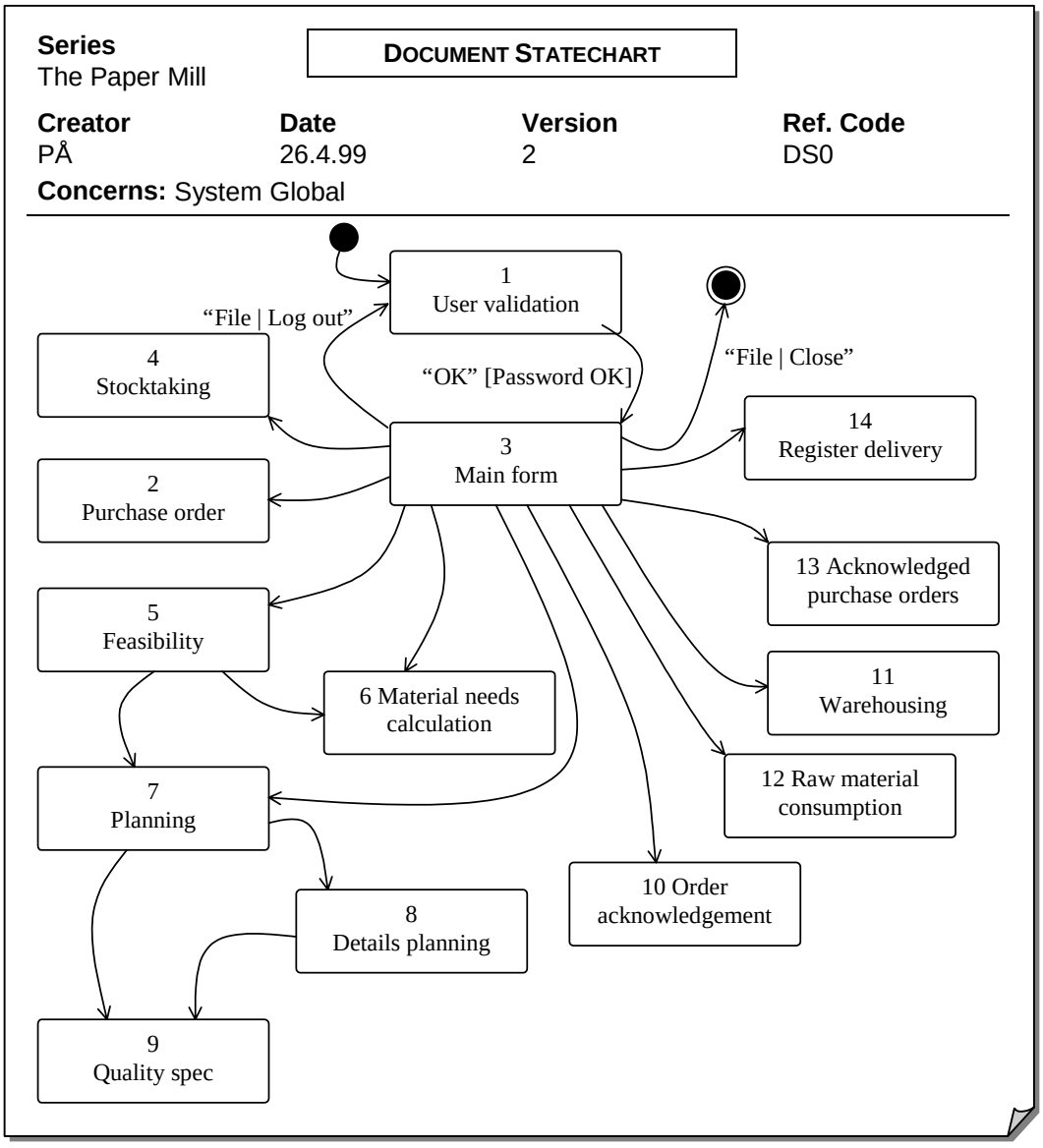



**Fig 7-3:** Example Document Statechart I.

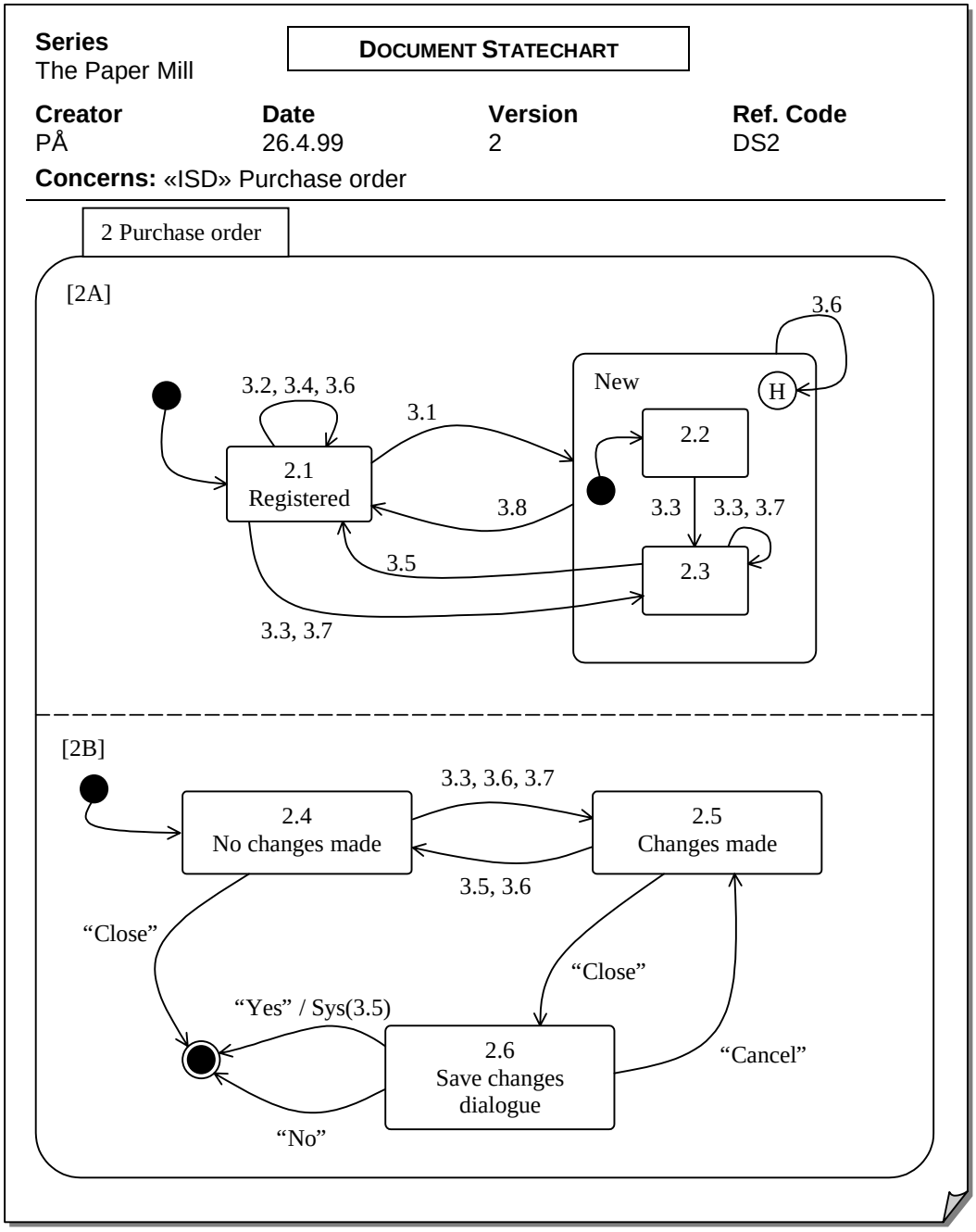



**Fig 7-4:** Example Document Statechart II.

## 7.3 Conceptual Analysis

In order to design interactions and interactive documents as well as automatic actions, there is a need to consider the propositional content of messages. This is preferably done during a Conceptual analysis based on identified ae-messages. Conceptual analysis and Interaction analysis (and Document analysis, see below) are thus performed as an integrated

process. During Interaction analysis the professional language used by the actors is captured and used in interactive documents. This language is then analysed using 'traditional' conceptual modelling in terms of concepts (classes) and properties; cf. object-oriented analysis (e.g. Graham, 1998; Booch, 1994; Blaha & Premerlani, 1998). To document conceptual relationships between concepts (i.e. properties of concepts) we propose to use UML class diagrams[1], where one class diagram is produced for each ae-message (see Fig 7-5).

Each discovered class (concept) is described in a class definition, which, following UML, consists of a brief description; the cardinality of the concept, i.e. how many instances are allowed and/or supposed; the class' position in a possible inheritance hierarchy by stating superclasses, attributes, and associations. Fig 7-6 shows the Class definition of one of the classes in the Class diagram of Fig 7-5. Note that the same class might be used in different ae-messages. The class definition, however, is a 'central' description of the properties of a class that holds for all ae-messages. Hence, not all attributes and associations defined in the Class definition are shown in all Class diagrams. This is, for example, the case with the association 'Purchase order acknowledgement' in Fig 7-6, which is not part of the Class diagram of Fig 7-5.

Following the UML, it is possible to also attach a state machine to a class in order to describe its dynamic behaviour. This is not always necessary, but if it is done a reference to the Class Statechart showing the state machine should be included in the Class Definition. This is also a way of describing what e-actions create, modify, and destroy, instances of a particular class. These are important properties when designing the action memory (during, for example, database design). Note that this is also a way of pragmatizing dynamic conceptual modelling. That is, there is an explicit connection between business actions and the dynamic model of the system, something that is often unclear in contemporary approaches to conceptual modelling (e.g. Sundgren, 1992; Snoeck & Dedene, 1998).

[1] One can, following Graham (1998), argue that UML is not fully suitable for this kind of modelling due to its bias towards relational concepts, such as allowing for constraints to be attached to relationships (which hence are not encapsulated by any class and so break the principle of encapsulation). We do however argue that UML can be used, with caution, and that it is the best choice available due to its widespread use and the fact that it is a *de facto* standard in many software organizations.

**Series**
The Paper Mill

**CLASS DIAGRAM**

| Creator | Date | Version | Ref. Code |
|---|---|---|---|
| PÅ | 26.4.99 | 2 | CD2 |

**Concerns:** Purchase order

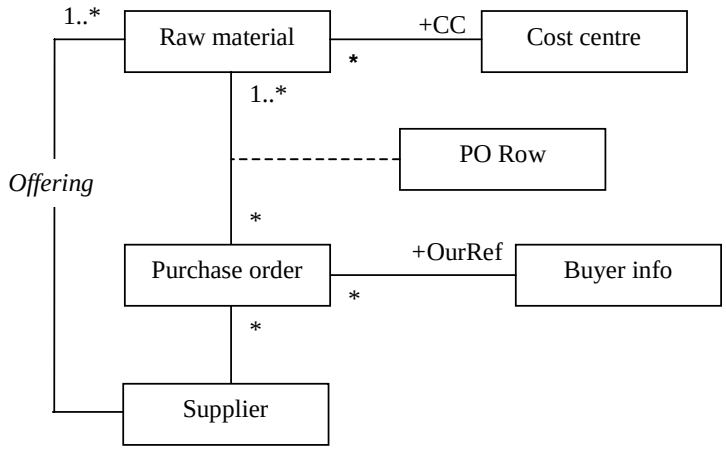



**Fig 7-5:** Example Class Diagram.

**Series**
The Paper Mill

**CLASS DEFINITION**

| Creator | Date | Version | Ref. Code |
|---|---|---|---|
| PÅ | 28.4.99 | 1 | CL4 |

**Concerns:** Purchase order

**Description**
Raw material order – internal or from external supplier

**Cardinality**
n

**Super-classes**
None

**Attributes**
PurOrderID : Integer
Date : Date
YourRef : String[40]
TermsOfDel : String
TermsOfPay : String

**Associations**
<no rolename> : Raw material in association <unnamed>
OurRef : Purchaser info in association <unnamed>
<no rolename> : Purchase order acknowledgement in association <unnamed>
<no rolename> : Receiver in association <unnamed>

**Fig 7-6:** Example Class Definition.

## 7.4 Document Analysis

When designing documents – interactive and others – during Interaction analysis, the focus is on the particular ISP under investigation. Some documents, however, are used in many ISPs and so there is a need to carefully verify that no inconsistencies arise. One way to assure this is to create a Document definition for each document, as shown in Fig 7-7.

**Series**
The Paper Mill

**DOCUMENT DEFINITION**

| Creator | Date | Version | Ref. Code |
|---|---|---|---|
| PÅ | 29.4.99 | 4 | DD2 |

**Concerns:** «ISD» Purchase order

**Description:** A Purchase order interactive screen document is used to formulate and send purchase orders (MD2). Physically, purchase order paper documents (DD2P) are printed by use of this document and subsequently sent to suppliers.

**Message(s) carried:** MD2 – Purchase order

**Message(s) created:** MD2 – Purchase order

**Document(s) created:** DD2P – «PD» Purchase order

**Used in ISP(s):** ISP 3 – Raw material purchasing

**Prototype:** DP2

**Fig 7-7:** Example Document Definition for an interactive screen document.

The heading of a Document definition contains the 'usual' meta-information, including the name of the document defined. The name is prefixed by a stereotype stating what type of document we are dealing with; for example: a paper document, or an interactive screen document. The tentative classification we have used so far is showed in Table 7-2.

| Stereotype | Meaning |
|---|---|
| <<PD>> | Paper Document |
| <<ISD>> | Interactive Screen Document |
| <<SSD>> | Static Screen Document (non interactive) |

**Table 7-2:** Stereotypes used for different types of documents.

This naming convention is used throughout the documentation whenever a document is referenced (for example in Fig 7-4).

A Document definition consists of six sections. First, there is a verbal description of the document briefly describing its intended use. Subsequently, there are references to message(s) carried by the document, message(s) created by the document, document(s) created by the docu-

ment, which are usually paper documents, the ISPs in which the document is used, and to a prototype of the document.

A prototype is a visualization of a document, usually created during Interaction analysis in parallel with I-Tables and sequence restrictions. Fig 7-8 shows the Document prototype of the document defined in the Document definition of Fig 7-7.

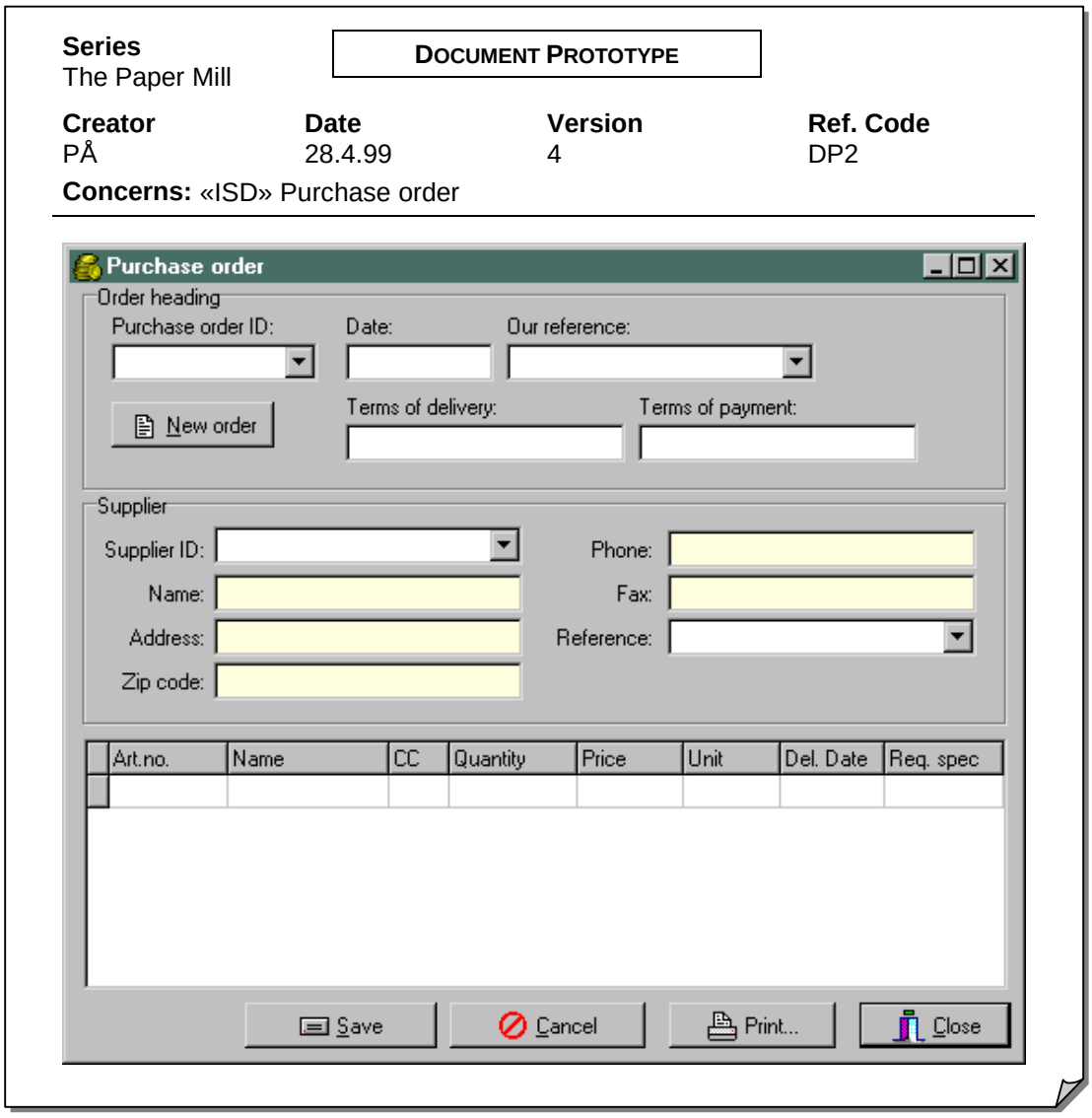



**Fig 7-8:** Example Document Prototype.

Note that prototypes of paper documents might be created during BPM in conjunction with their document definitions.

As we can see, most of the activities performed during Document analysis is actually performed intertwined with Interaction analysis. The reason for treating it as a separate focal area is that it promotes a

holistic view of the documents used whereas Interaction analysis is focused on one ISP at a time.

## 7.5 Additional Considerations

Interaction analysis concerns interactive usage situations and hence interactive documents have been in focus so far. The documents used in consequential and automatic usage situations are often based on ae-messages from interactive usage situations and they are therefore normally treated during Interaction analysis. However, there is a need to 'walk through' the business model and make sure that all the documents have been taken into account. There might also be a need to consider all ae-messages, documents and propositional contents as a whole and the, to some extent, fragmented document model and conceptual model of interaction analysis and conceptual analysis are consolidated to system global level. Note that through the use of a method tool (CASE), such as Rational Rose, it is possible to do conceptual analysis based on 'parts' of an object model, for example messages, and still maintain global consistency. This is important in order to verify that the same class (concept) is not used with different definitions in different contexts, for example.

## 7.6 Actability Software Requirements Specifications

There are primarily two kinds of requirements, with corresponding requirements specifications, that have to be taken into account when building information systems. These are usually referred to as initial requirements and detailed requirements (e.g. Sommerville, 1996). The initial requirements deal with the fuzzy ideas in the beginning of IS development. Their specification is thus positioned in time somewhere between Change analysis and VIBA using SIMM terminology. It is important to capture the initial requirements since they often form the basis of the contract between the software organization and its customer. This thesis deals primarily with the detailed requirements that are elicited and analysed during IS development and enhancement. Our discussion about the software requirements specification (SRS), therefore presupposes the analyses described in earlier sections.

The specification of requirements within the SRS is often divided into functional and non-functional requirements. Non-functional requirements can be further divided into categories such as usability requirements, implementation requirements, *et cetera* (Sommerville, 1996). Actability requirements concern both functional and non-functional requirements. We therefore chose to divide actability requirements into three categories; (1) functional requirements, (2) us-

ability requirements and (3) other non-functional requirements. The 'usability category' includes requirements regarding document layouts, navigation and interpretability. The 'other non-functional category' includes all other non-functional requirements that relate to actability.

In some cases the models produced will serve as sufficient specification of requirements, but there might be good reasons for extracting requirements from the models and putting them together in dedicated sections of the SRS (one for each category). The models and specifications then serve as background and rationale for the requirements and should also be part of the SRS. This approach is similar to the traditional notion of the requirements document (Sommerville, 1996).

In the remainder of this section we discuss how the VIBA documents relates to the SRS and hence where the requirements are to be found.

### 7.6.1 Functional Requirements

There is a direct correspondence between the IS actions found in the ITables and Document Statecharts, and the functional requirements of the SRS. When producing the SRS we thus have to go through these models and check that there are requirements that cover all IS actions. Some constraints on functional requirements can also be found as restrictions in the 'IS column' of the I-Tables, but most of such restrictions have to do with usability requirements.

### 7.6.2 Usability Requirements

Requirements regarding user interface design are found in the I-Tables, the Statecharts, and the Document prototypes. One aim of I-Tables is to capture requirements regarding understandability. The restrictions on interpretation are sources for requirements of that kind. Another source is the stated captions on clickable items. These requirements aim at making the IS more action transparent and intuitive; the IS should behave just as the user assumes (i.e. affordance). Other usability restrictions imposed on the IS (curly brackets in the third column) are another source for usability requirements, such as, for example, response times.

### 7.6.3 Other Non-Functional Requirements

This section of the SRS is concerned with non-functional requirements and restrictions on the software such as required operating platforms and specific standards the system has to conform to. These kinds of requirements are most often found within different parts of VIBA, for example restrictions of data representation, which ought to be found dur-

ing conceptual modelling; and other requirements found in the Prerequisite list of BPM.

Since VIBA is used with a bottom-up approach the actability oriented analyses often serve as a rationale for these kinds of requirements. It is therefore important to crosscheck the different models in order to deduce requirements from their interconnections and to avoid ambiguity and inconsistency.

## 7.7 ISM Versus VIBA'93

As pointed out in chapter 5, ISM corresponds approximately to Message analysis and Processing analysis in VIBA'93 – even though the task of identifying and defining ae-messages is regarded as belonging to BPM instead of ISM. Hence, what is left of Message analysis corresponds to the Conceptual analysis of ISM. One important difference between VIBA'93 and VIBA'99 is that in VIBA'99, the notation for Conceptual analysis is borrowed from UML whilst VIBA'93 proposes a method specific variant of traditional entity/attribute/relationship notation. One high-level design goal when re-designing VIBA was to make use of existing notations and modelling formalisms. Therefore, it seems natural to use UML for this purpose.

VIBA'93 do not explicitly promote the use of prototypes, even though this is mentioned as a possible complementary approach. Instead human-computer interaction is modelled with Interaction diagrams (in Swedish: Interaktionsgrafer) in VIBA'93. Interaction diagrams are, in essence, fine grained Action diagrams that describe human-computer interaction in a procedural way. Such an approach does not lend itself very well to development with event-driven rapid application development tools, which VIBA'99 aims to support. A state-based approach is more appropriate to model event driven software (Horrocks, 1999), which is why Statechart modelling is proposed by VIBA'99.

In VIBA'93, IS actions (which where referred to as functions) are described by Processing diagrams and Function descriptions (in Swedish: Behandlingsgrafer och Funktionsbeskrivningar). A Processing diagram is, similar to Interaction diagrams, a fine grained Action diagram describing the processing logic procedurally. A Function description describes how different business rules apply to each function of the IS. Following the discussion about formulating and sending ae-messages in ATIS (chapter 4), the aim was to make the connection between business context, human-computer interaction, and IS functionality more explicit than in VIBA'93. Therefore, I-Tables are used to describe IS functionality (IS action) integrated with the description of formulation and sending

of ae-messages. In VIBA'99 it is also possible to describe IS-action by use of ONN expressions (see section 7.2). Hence, one is not restricted to procedural descriptions but can use declarative statements when appropriate.

In VIBA'93 there is no specific Document analysis, instead so-called Layouts (sketches of documents) are created during Message analysis in conjunction with the creation of Message summaries.

## 7.8 Notes on The Relation Between ATIS and ISM

In this section we will discuss some important relationships between Information Systems focused Modelling (ISM) and the Action Theory of Information Systems (ATIS).

The baseline for ISM is the ATIS division of speech acts into five sub-acts (see section 4.2.1). According to ATIS, a usage situation (referred to as *usage situation proposal* during ISM, i.e. ISP and ASP) consists of formulation and execution of one or many e-actions. Each performance of an e-action can be viewed as consisting of five different acts (following the discussions in chapter 4):

a) Performing a *formulation act*.

b) Performing an *utterance act*.

c) Performing a *propositional act*.

d) Performing one or several *illocutionary acts*.

e) Performing one or several *perlocutionary acts*.

The first and second act (a and b), can and should be supported by the information system – this is what Interaction analysis is about. According to ATIS (section 4.2.3), each interaction with an IS, performed to formulate and execute e-actions, follows a structure called the Elementary InterAction Loop (EIAL). The I-Table notation builds explicitly on that structure. However, since the formulation act consists of a sequence of interactions, we need a way to describe sequence restrictions – this is done with Document Statecharts.

The third act (c), concerns the conceptual structure of the resulting ae-message – this is what Conceptual analysis is about. During Conceptual analysis, concepts are explicitly defined in Class definitions, and their relationships to each other are described in Class diagrams. If some important properties of a class (concept) change over time, it is possible to describe the 'life cycle' of a class in a Class Statechart.

The fourth and fifth acts (d and e), form the point of departure for the analyses performed during ISM – this is the input to ISM from BPM.

To get a holistic view of the use of different documents, a Document analysis is performed. Document analysis thus concerns the different media and contexts of different e-actions, and how they relate.

# Chapter 8

# Lessons Learned from the Case Studies

*Small is the number of them that see with their own eyes*
*and feel with their own hearts.*

Albert Einstein

As explained in chapter 2, the empirical work has resulted in *ideas* and *data*, respectively. When interpreting the data, and drawing conclusions from it, it is thus necessary to be aware of that ideas have changed the design, to which data refers, both during and after data collection and interpretation. The data collected during the realization phase of Case II therefore were interpreted in the light of what the specification looked like when handed over to the consultants (see section 2.5). We will therefore introduce this chapter by examining the differences between what the specification looked like at that time (referred to as VIBA'98), in contrast to what has been described in chapters 5 to 7 in this thesis (referred to as VIBA'99). We then continue by concluding the lessons learned from the empirical work. Throughout the chapter we will use the labels given to the different sources of data as described in Table 2-2 (the Source and What columns) on page 40 (section 2.5.2.2).

## 8.1 VIBA'98 Versus VIBA'99

The VIBA'98 specification consisted of the following documents:

- Action diagrams,
- Document definitions,
- ISP List,
- I-Tables,
- Document prototypes,
- Statecharts,
- Class diagrams, and
- Class definitions.

The Action diagrams correspond to the ones in VIBA'99. The Document definitions, however, have undergone important changes.

When working with the document metaphor as a generic concept, including screen documents, we had, as we come to understand during Case II, mixed up the concept of ae-message with that of document. Under the false assumption that one document type carried one, and only one, ae-message type, we had put the descriptions of propositional content, action mode, *et cetera*, in the document definitions. Hence, the concept of message had been somewhat underrated in VIBA'98, compared to the original VIBA'93. This, of course, also affected the externalization of ATIS. Even though the concept of ae-message was included in ATIS, its relationships to documents were not made explicit.

As a consequence of the 're-introduction' of the message concept, the ISP List came to look a bit different also. In VIBA'98 the 'input message' and 'output message' was missing. Hence, the ae-message concept as a bridge from BPM to ISM was not as emphasized, and thus not as utilized as an analytical tool, as proposed in VIBA'99.

The I-Tables and Document prototypes looks similar in VIBA'98 and VIBA'99. The Statecharts, however, have changed somewhat. Of course, the formalism as such has not changed, but the way in which we use it. In VIBA'98 the ISP was in focus when modelling sequence restrictions; in fact 'Sequence restrictions' was the phrase used for the Statecharts documents. We also used the distinction between *user actions* and *navigational actions* to treat these in two separate models, both based on Statecharts. However, influenced by Horrocks (1998), we have decided to treat them together in VIBA'99 as argued in chapter 4. The Class definitions and Class diagrams still follow UML and hence look the same. However, in VIBA'98 we proposed to create one Class diagram per ISP, instead of one per message, as in VIBA'99. The VIBA'98 approach seems valid from a 'task' or 'use case' perspective, but when following a language action perspective, the VIBA'99 approach seems more intuitive. In many cases these two approaches yield the same result, i.e. is when one (type of) ISP 'produces' one message type. However, if the same message type is used in several ISPs, the VIBA'99 approach reduces redundancy in the documentation.

## 8.2 Analysis Results

This section presents the result of the analysis of recorded data as explained in chapter 2. From the data we have identified two major categories, *process* and *product*. These two have been used as a means to structure the presentation of the result. The process category contains issues relating to the systems engineering process, and the product category contains issues relating to the specification (i.e. the documents produced

during the process). The process category has been sub-categorized into issues regarding the overall development model and those regarding specific modelling activities. The product category has, similarly, been sub-categorized into the different types of documents used in the VIBA'98 specification. The section ends with some overall concluding remarks. Please note that the quotations are the author's translation from Swedish.

### 8.2.1 Process

#### *8.2.1.1 Development Model*

VIBA does not presuppose any particular development model. Instead analysts are supposed to choose a development model suitable for the particular engineering effort at hand. However, during Case II we were forced to rely partly on a waterfall model. During requirements elicitation and analysis we worked iteratively with analytical models and prototypes. But since the specification was handed over to the IT consultants, the specification constituted a milestone and implied a waterfall. This resulted in several criticisms from the IT consultants. The most important being that they felt that the specification was fixed and that they had difficulties knowing when and what to change. And changes were necessary due to both technical restrictions and to budget reasons. What should have done was probably to overlap the project team doing analysis with that doing realization. As it turned out, the specification had seriously reduced the available design space, which led to frustration. Actually, problems of this sort were expected with the chosen approach (cf. chapter 2), and lead over to another observation made during Case I, and utilized during the analysis phase of Case II.

The observation concerns the integrated approach to business modelling and human-computer interaction modelling proposed by VIBA. What we observed was that assumptions about the business process activity structure, manifested in Action diagrams, came to change during Interaction analysis. When digging deeper into the action structures and letting users work with prototypes, the business model changed. One conclusion drawn from this is that an integrated approach, such as the one we propose, is necessary to develop an accurate business model.

#### *8.2.1.2 Process Specifics*

One important lesson to learn from Case I is related to the prototyping approach. Remember that the heart of the IS was production planning, including optimization of production schedules. To be able to build a pro-

totype that showed functionality related to this, we had to design the optimizing algorithm. Unfortunately, that was easier said than done. Since the algorithm, in turn, relied on the underlying data structure, we had, in practice, to come up with the complete system to be able to visualize that part of the IS. A part of the IS that, from a user interface perspective, seemed quite simple[1]. Hence, it is best to be careful not to overwork document prototypes. In Case I this led to a delay of several months[2].

### 8.2.2 Product

#### *8.2.2.1 Overall Opinions*

Both C1 and C2 were of the opinion that the specifications were a bit too formalized. This was most obvious when working with I-Tables and Statecharts. They proposed to supplement each document with a verbal description of the main ideas in order to increase assimilation. However, they agreed that, in general, formality and unambiguity in a specification is something positive. What they wanted was a smoother path into the documentation. One suggestion in this direction was to give a brief statement about how each interactive screen document was supposed to function, in addition to Statecharts and I-Tables.

#### *8.2.2.2 Action Diagrams*

Both C1 and C2 showed a positive attitude towards the Action diagrams. C2 emphasized the need to understand business routines and rules, and acknowledged that Action diagrams help to give such understanding. One problem, noted by both C1 and C2, lies in the limitations of the paper format. Lots of connectors are needed to relate different Action diagrams showing different business contexts (or parts of the same context). Of course, this would be helped by appropriate tool support.

During the early phases of Case II Action diagrams were used to communicate with the users, who were non-software professionals. That approach was very successful and the users had no difficulties in following the discussions. The Action diagrams were subsequently used as road maps while Document prototypes were discussed and refined together with the users. This helped the users to (1) put the prototypes in familiar contexts, and (2) focus discussions on the particular interactive situation under analysis. When not relating the prototypes to the Action diagrams, the users had difficulty in focusing on the particular task that we were interested in at the moment. The reason was that the same documents were used in more than one ISP.

---

[1] The optimization was performed just by clicking a button.

[2] Calendar months that is, not man months.

One problem experienced by R1 during Case I, however, was that iterative tasks were somewhat too complex to visualize efficiently with Action diagrams, which are most appropriate for describing sequential flows of activities. This was also a problem in Case II, according to both R1 and R3.

#### 8.2.2.3 ISP Lists

The ISP List was 'invented' during the analysis phase of Case II as a result of problems with navigation in the documentation. The ISP List was also appreciated by C1 and C2 as giving a good overview. However, one suggestion (by C2) was to include references to Statecharts and Class diagrams. This can easily be augmented (spelled out) to the corresponding messages in the ISP List. Another suggestion was to mark each e-interaction with the number of the ISP to which it belonged. This has been incorporated in VIBA'99 as shown in chapter 7.

#### 8.2.2.4 I-Tables

One major criticism regarding I-Tables was brought up by C1. The fact that all e-interactions are treated equally in the I-Tables means that trivial e-interactions are described as fairly complex. Examples of such e-interactions are simple GUI-updates such as changing one or several data fields. Such e-interactions often lead to the phrase 'Update GUI but not action memory'. This was also a reflection made during specification by R1 and R3. Perhaps it might be possible to distinguish simple e-actions, and treat them differently than more complex ones.

One suggestion was that the I-Tables should be presented in an order that reflects the main course of e-interactions. Both C1 and C2 had initial difficulties when assuming that such ordering was present, which it was not. However, one must remember that e-interactions constitute a set. Hence, it is not always possible to present a 'logical' order among them. The suggestion is, nevertheless, worth considering.

#### 8.2.2.5 Document prototypes

Phrases used to describe the Document prototypes were: '*Absolutely necessary, has been used as foundation to at least 80%*' (WC2) and '*Very valuable to tie together the other documentation*' (WC1). However, the prototypes were also judged as devastating for creativity during system construction, if they are rigidly followed. An interesting comment was that prototypes '*preclude prototyping*'. This should probably be interpreted as a need to allow for prototypes to change all the way through the construction phase.

During the specification (both Case I and Case II) we used the Document prototypes as the primary 'language' when talking to the users. However, one experience is to use 'paper prototypes' initially. This relates to the previously discussed problem of prototypes becoming too complex. By using paper prototypes, different alternatives can be sketched, and easily thrown away if unsuitable. This approach is also in line with what Graham (1998) and Cooper (1994) propose, for example.

#### *8.2.2.6 Document Definitions*

The Document definitions, which, as explained previously, looked radically different in VIBA'98, were of no help at all to C1 and C2. They both felt that the information was '*formal and useless*', i.e. information for the sake of information. They got the relevant understanding from other parts of the documentation. R1 and R3 felt somewhat similar about them. However, the Document definitions typically play a more vital role during early ISM when straightening out the purpose of each document. In VIBA'99 it is also supposed to be used actively in conjunction with the Message definitions, and as a source for connecting different pieces of the documentation in conjunction with the ISP List.

#### *8.2.2.7 Statecharts*

C2, who was the one developing the interface code during Case II, admitted that the Sequence restrictions (Statecharts) gave good support for event-driven programming[1]. During this work he did not, however, utilize the support given by the approach as described in detail by Horrocks (1999). This was because Horrocks' approach was unfamiliar to him, and required too much learning time to be worthwhile during this project. Nevertheless, the Statechart approach felt a bit complex and even very simple interactive documents tended to look complicated.

We believe that the obstacles indicated by C2 had to do with experience of the approach. At first both R1 and R3 felt that the approach was complicated. However, while working with it, it became more and more straightforward. Also, as Horrocks (1999) points out, the Statechart approach requires a shift in perspective from a functional view of user interfaces to a state-oriented one. Horrocks (*ibid.*) admits that this shift can be cumbersome, which our experience thus confirms.

One aspect of Statecharts, not commented upon by Horrocks (1999), was utilized during Case I. Interactive documents can be regarded as objects, and the Statecharts thus represent the dynamics of such objects. We tried to make use of this when implementing inheri-

[1] The system was constructed using MS Viual Basic, which rely heavily on event-driven programming.

tance hierarchies of documents; a facility supported by the development tool used. However, we had problems describing such inherited behaviour. This is a common problem in object-orientation, as discussed by Snoeck & Dedene (1998), which resulted in that we had to create, quite unnecessarily, a separate Statechart for each sub-classed document. Even though a promising approach, we have chosen not to include document inheritance in VIBA until this has been investigated more thoroughly.

#### *8.2.2.8 Class Diagrams*

As indicated previously, the Class diagrams served as a source for understanding of the business and of the system. It also served as the primary foundation for creating the relational database schema. However, using Class diagrams, and thereby an object-oriented perspective during analysis, leads to some inherent problems when translating analysis models to a relational schema (cf. Ågerfalk, 1999).

The structuring of Class diagrams in one diagram per ISP, caused some initial confusion during the design phase of Case II. However, once straightened out, this was no longer a problem. Nevertheless, according to C2, the system global Class diagram was more useful than the 'partial models'. Without the global Class diagram '*it would have felt very divided…Whether it is an ISP concerning ordering or purchase doesn't matter, we look at –what relationships are there between these classes*' (I1). Probably, the partitioning of the class model into parts is of more value during analysis and specification than during design and construction. R1 and R3 felt that the approach helped when creating the model. When a global class model is created from scratch, it is hard to know where to begin. It is interesting to notice also that the global Class diagram was created automatically from the partial models by the CASE-tool used.

#### *8.2.2.9 Class Definitions*

The Class definitions were, not surprisingly, used as the primary basis for the design of relational tables, and specifically when allocating attributes to different tables. According to C2 they were used extensively.

#### *8.2.2.10 Missing Documentation*

One interesting suggestion, made by both C1 and C2 during Case II, was that a data model (for example E/R-diagrams) ought to have been created during analysis. The reason for this was that some information was incorrect or missing from the Class definitions and the Class diagrams. The interesting thing about this is that we believed, and still believe,

that a class model contains the same information as a 'traditional' data model. Of course, foreign keys and candidate keys, for example, were not explicitly stated in the specification. However, the problems do not seem to derive from the formalism *per se*, but from the use of the formalism and from the consultants lack of UML experience. Nevertheless, some information was missing from the specification. Information that, according to C2, probably would have been identified if a deeper analysis of existing data structures had been made. The problem was thus that our conceptual model was an idealized one that did not reflect the format of data that the IS was supposed to import from other systems. This led to an analysis model with assumptions about, for example, attributes of classes, that was not suitable for storing in a 'one class, one table' fashion. Hence, the conclusion to be drawn is that substantial attention should be paid to existing data structures during analysis in order to ease the burden of translating the analysis results to a working design.

As a consequence of the lack of class attributes in the specification, the idea of explicitly referring to attributes by use of ONN in message definitions came up. This approach has thus not been empirically tried, but, according to Blaha & Premerlani (1998), it should be a practicable way to continuously verify the class model's completeness and consistency.

### 8.2.3 Concluding Remarks

One thing that has become obvious, if it was not so already, is the need of formal training of method users in order for them to appreciate and accept a method. Our aim of incorporating 'standard' modelling techniques and notations thus seems to be a promising approach. Nevertheless, problems related to method knowledge and systems engineering skills occurred during Case II. Of course, abstract system models are not optimal for communication with non-software professional users. This is also one reason why we have adopted a prototyping approach with the new VIBA. However, we experienced problems that probably could have been avoided through formal training of the IT consultants and the systems co-ordinator. Both C1 and C2 have had formal training at University level in an early version of VIBA, called IBA (pre-VIBA'93), and were familiar with the principles of working with Action diagrams. Nevertheless, they both experienced difficulties with some of the newer additions to the technique[1]. They were not fully acquainted with UML either, unfortunately. However, since they both had a general understanding of object orientation they managed, after surmounting some initial obsta-

[1] The semantic richness of Action diagrams has continuously evolved over the years.

cles, to interpret the Class diagrams, and produce a relational database schema based on those. SC, on the other hand, had no formal training in systems engineering at all, which we (R1 and R2) realized a little late. SC was, however, used to working with Action diagrams from earlier Change analysis projects. He was also familiar with relational databases, but not in terms of entity/relationship diagramming (or object orientation for that matter). As a consequence, errors which we thought had been straightened out, eventually ended up in the specification. And, since C1 and C2 took the specification as the 'ultimate truth', time consuming restructuring of the developed database was the result.[1]

What, then, can we learn from all this? Well, first of all, formal training of all actors participating is imperative. How, and to what extent, remains to be elaborated though. We can also conclude that the need for training, and the appreciation of different modelling techniques, depend to a great extent on the actors involved. For example, C1 gave a more positive image of Action diagrams as a source of business understanding than C2. C2 on the other hand made it clear that he was more used to thinking in terms of entity-relationship modelling, and consequently rated the Class diagrams as the best source for business understanding.

Furthermore, one of SC's intentions, when engaging in Case II, was to evaluate whether VIBA could be practicable within his organization. Practicable in the sense that non-software experts with some modelling experience (such as SC), should be able to create requirements specifications of high quality. The conclusion is that the proposed techniques do not seem to lend themselves very well to such 'end-user-specification'.

## 8.3 Overall Reflections on the Empirical Results

In this section we will reflect upon the empirical results in relation to the high-level re-design goals of VIBA, and to the five knowledge domains that we wanted to influence the re-design (see chapter 1). Note that a general discussion of these topics, not focused on the empirical work, is given in chapter 9.

The high-level design goals for the re-design of VIBA/SIMM were to (1) make theoretical foundations (ATIS) explicit and to take full advantage of their consequences, (2) design a method that produces documentation that is communicable between developers and other

[1] We would like to emphasize that we do not blame anyone other than ourselves for this. It was, of course, our responsibility to verify and validate the specification.

stakeholders, and (3) make use of as much existing knowledge and good practice as possible.

The five recent knowledge domains that we wanted to influence the re-design of VIBA were business process orientation, rapid development, object modelling, usability, and software requirements management.

### 8.3.1 High-Level Re-Design Goals

As discussed in chapters 6 and 7, ATIS has been made explicit in VIBA. The question remains, though, whether it has been shown to be explicit in (1) information systems built according to VIBA, and (2) information systems specifications produced according to VIBA.

Unfortunatley, due to reasons discussed in chapter 2, the first aspect has not been possible to investigate, and remains an important research topic for the future.

In the specifications produced, the action character of the specified information systems had become visible. As one example, according to C2 (if interpreting his words from an action perspective) the produced Document prototypes and Document Statecharts show explicitly what e-actions and e-interactions that are possible to perform, and in what sequence. What was interesting, though, was that the Document definitions, in which the separation between propositional content and action mode was made explicit, did not attract C1 and C2. According to them, that information was visible already in the Action diagrams. Probably, this was, at least partly, a result of insufficient internalization of ATIS. The separation, and hence the Document definitions (Message definitions in VIBA'99), should be highly important during, for example, database design.

As discussed previously, the documentation was communicable, even though some formal training would have increased the understanding.

The most visible reuse of existing knowledge and good practice (in the specification) was probably the use of UML, which  Both C1, C2 and SC were positive to. In fact, C1, expressed that the use of method specific notation for conceptual modelling in VIBA'93 was both unnecessary and hard to learn.

### 8.3.2 Influential Knowledge Domains

The business process orientation is most visible in the Action diagrams, which span several functional units of the business in Case II. The biggest problem related to this was that the business was not really process oriented. Thus, R1 and R3 spent much time trying to contact the right

people in different departments. Nevertheless, the approach was useful in two respects. First, the specification was created based on the work tasks of the users, which actually had an 'informal' process structure. Second, it highlighted some communication problems between different functions, that were important to solve, but were beyond the scope of this thesis. Hence, a process view helps reveal problems which are difficult to discover using a traditional approach.

During the analysis phase of Case II, rapid development principles were applied successfully (prototyping, user involvement, *et cetera*). Unfortunately, the research design made it impossible to continue with these principles throughout the realization (see chapter 2). One big problem was that C1 and C2 considered the specification to be fixed, which considerably reduced the available design space. There is, however, nothing inherent in VIBA that prevents the use of rapid development practices throughout an entire project.

Object modelling, in the form proposed by Conceptual modelling of VIBA, was experienced as both positive and negative. C1 and C2 had difficulties in understanding the Class diagrams, due to insufficient UML experience. Another problem, expressed by C2 was that it caused confusion as to whether to build the system truly object-oriented or not. This should not be regarded as a weakness of the specification, but as a strength, though. A specification that does not make unnecessary presumptions about the design is of greater value than one that does. In this respect, the design space was increased.

Usability requirements can, and have, become visible in the documentation proposed by VIBA. However, to what extent this is achieved probably depends on the people involved in each development project. One problem during Case II was that a prerequisite of conforming to Microsoft's style guides was brought up when the analysis phase was almost ended. A consequence was that changes were made to the Document prototypes without user involvement during the design phase. These changes were not reflected in the specifications, a fact that has led to some reflections made about the last influential knowledge domain.

Software requirements management has not been explicitly dealt with during the empirical work. The foremost reason for that is the lack of sufficient tools for managing the different models proposed by VIBA. Because of this, changes made during the realization were not reflected in the specification. Of course, that is not good requirements management.

# Chapter 9

# Summing up and Planning for the Future

*Sure, it works in practice, but will it work in theory?*

Erik K. Clemons

In this chapter, we discuss how the re-designed VIBA meets the design goals, and how it makes use of the knowledge domains discussed in chapter 1. We also present some reflections on the contributions made. Finally, some related work within the Language Action Perspective and requirements engineering are commented upon. Throughout the chapter alternatives and possible future directions are mentioned where appropriate.

## 9.1 Conclusions and Reflections

The aim of this section is to conclude the work by discussing how the re-designed VIBA/SIMM meets the re-design goals, and makes use of the influential knowledge domains; and to briefly reflect upon and highlight some of our most important contributions. That is, we are trying to show to what extent have we accomplished what we intended.

### 9.1.1 High-Level Re-Design Goals

As discussed in chapter 1, we had three high-level re-design goals for the re-design of VIBA/SIMM: (1) to make theoretical foundations (ATIS) explicit and to take full advantage of their consequences, (2) to design a method that produces documentation that is communicable between developers and other stakeholders, and (3) to make use of as much existing knowledge and good practice as possible.

#### *9.1.1.1 Theoretical Foundations*

We believe that the theoretical foundations have been made explicit (chapters 3 and 4). The re-designed VIBA takes communicative action as a starting point and all analyses are based on the dual character of information: the propositional content and the action mode. This characterization of doing when speaking is utilized both to (1) gain under-

standing of the business and the use of information systems within it and (2) as an analytical tool during specification.

For the reasons explained in chapter 1, we never had the opportunity to actually investigate how, and to what extent, ATIS can be made operationalized, and hence visible, in produced information systems. This is thus one of our most important future research topics. However, we have shown that by applying VIBA, documentation that emphasizes the action character of information systems is produced.

#### *9.1.1.2 Communicable Documentation*

The results of Case II show that the documentation proposed by VIBA is communicable, with some restrictions. We concluded that to make use of VIBA, some sort of formal training is probably needed. That is, however, the case with any systems engineering method; systems engineering is a difficult task. However, both Action diagrams and Document prototypes were well suited for communication with the users. The suggestion to include more 'prosaic' descriptions in addition to the more formalized notations used so far seems like a good idea. To what extent this is done ought to be decided during each particular development situation, and depends on the developers experience of (1) the method, and (2) the business under development. It is also dependent on (3) the development model used (waterfall, iterative, *et cetera*) and the project setting.

#### *9.1.1.3 Reuse of Knowledge and Good Practice*

The reuse of existing knowledge and good practice is most evident in the incorporation of large parts of UML into VIBA. The specific UML parts used are class models and Statecharts. These have both proven useful and possible to integrate with the proposed action-oriented approach. The use of prototyping is an example of good-practise that we have adhered to, as are the influences from the human-computer interaction field (style guides, metaphors, *et cetera*). The documentation forms that have not been 'borrowed' are those that result from the operationalization of ATIS and can, to a large extent, be viewed as extensions to existing formalisms. One example is the I-Tables that clearly extend Horrocks' (1999) approach. Another example is the Message definitions. A Message definition can be regarded as defining a 'sub-universe of discourse' for object modelling.

### 9.1.2 Influential Knowledge Domains

As discussed in chapter 1, there were five knowledge domains that we wanted to influence the re-design of VIBA: business process orientation,

rapid development, object modelling, usability, and software requirements management.

#### 9.1.2.1 Business Process Orientation

Business process orientation constitutes an underlying perspective for ATIS and VIBA. A business process can be regarded as an institutionalized pattern of activities, aiming to create value for customers. The business modelling part of VIBA (i.e. BPM) builds explicitly on this view of businesses. The purpose of BPM is to describe the action patterns, and the goals and problems inherent in those. A crucial part of business activities is the use of information systems, which is focused during ISM. However, one criticism that can be directed towards VIBA is that the business process perspective is too implicit. The next step in the development of VIBA would probably be to explicitly incorporate modelling techniques that make this perspective more explicit. Within the SIMM family of methods, such techniques exist (e.g. Lind, 1996). These techniques are focused on capturing and describing business processes at a higher level of granularity than Action diagrams, and on relating business processes to each other. Such modelling is important when building systems that involve more than one business process, which has been the focus of this work.

#### 9.1.2.2 Rapid Development

The work reported in this thesis has focused more on different modelling techniques than on overall process models (framework). Much of the work on rapid development, on the other hand, has to do with such overall models. Hence, we have not emphasized rapid development as much as we intended to from start. In fact, there was a contradiction in focusing on both rapid development, and expressiveness and communicable documentation, from a research perspective. The design of the research process of Case II (see chapter 1) led to a waterfall model, even though it was perhaps not the ultimate choice if following the guidelines of McConnel (1996). This, together with circumstances outside of our control, led to problems that ended up with yet another software project failure.

The conclusion is that so far we have not verified that VIBA is actually suited for rapid development. However, it is our belief that it is. Further research, especially empirical research, is thus needed. One possible future direction would be to try to use, for example, DSDM (1999) as a process model and apply VIBA within the DSDM framework. To date, we have not made any attempt to incorporate guidelines for requirements selection (or reduction), nor have we investigated the im-

pacts of such an approach. This is certainly an important research topic if VIBA is to become a true rapid development method.

One aspect of rapid development that showed itself to be very useful during the empirical work, was the prototyping. The mixed analytical and experimental approach worked well both in terms of documentation and in terms of user participation.

#### *9.1.2.3 Object Modelling*

Object modelling and object orientation have substantially influenced the re-design of VIBA, even though VIBA is certainly not a true object-oriented method. One reason for this is that we do not commit to object orientation during business modelling, due to object orientation's mechanistic view of human activity, which would contradict the underlying human infological tradition. However, the work of Graham (1998) introduces fuzzy logic into descriptions of objects, which is an interesting approach, worth looking into more thoroughly. We have chosen to use object orientation during conceptual modelling. In this respect, what we propose is an object-oriented version of 'traditional' conceptual modelling. However, the reason for including object orientation is not pure populism. Object orientation seems to better match a language action approach than traditional 'static' E/R-modelling.

Since we have not dealt with design, and object-oriented analysis, in this work, the full potential of using object orientation has not been explored. This is an area that remains to be studied. One example of extending the 'degree of object orientation' is to view documents as pure objects, as explored to some extent during Case I. This seems like a 'natural' approach since we consider documents as dynamic entities that contain knowledge (ae-messages) and exhibit behaviour (action potential). However, this remains to be studied further.

#### *9.1.2.4 Usability*

The influences of usability are most noticeable in the concept of actability, which has become to play a central role within ATIS and VIBA (see chapter 4). Following the IS use-situation model[1] by Shackel (1984), the main contributions of the Language Action Perspective, and traditional information systems engineering methods, have been to the relation between the user and the task, and to the relation between these two and the environment. Usability, on the other hand, has emphasized the relation between the user and the tool. Our aim has been to combine these two traditions in order to achieve a better coverage of all four compo-

[1] An IS use-situation consists, in this model, of four components: user, task, tool, and environment. See Fig 1-4 on page 11.

nents. In essence, this will probably imply a move away from Shackel's model, which focuses on the binary relationships between the different use-situation components, into a model with a ternary relationship between user, task, and tool (cf. section 4.4.2). By treating human-computer interaction as a crucial part of doing business, such coverage has been achieved. Further research is still needed, though. One important issue is, for example, to incorporate usability (or rather actability) testing techniques (cf. e.g. Nielsen, 1993) into VIBA in a more structured way. As of now, such techniques are possible to integrate, but no 'formal' procedures for how to do that has been proposed.

As pointed out by Gulliksen (1999), different parts of systems engineering require different skills. To ensure that usability aspects are emphasized, he argues for adding a specific 'Usability designer' to systems engineering projects. This seems like a good idea. Even though an integrated effort is needed, different areas of expertise must probably be combined in order to achieve high quality actability.

We use the term 'usability' to refer to a subset of actability. Please note, however, that the issue is not whether usability should be considered part of actability, and actability an extension of usability, or vice versa. The issue is to make information systems more actable and thus more usable.

#### *9.1.2.5 Software Requirements Management*

As discussed in chapter 1, requirements traceability (RT) is important for system maintenance and evolution, and hence for requirements management. RT can be viewed from two different angles. One viewpoint is traceability from business model through systems models to software system. We believe that such traceability is important in order to, for example, predict software change and software change costs when re-designing or evolving a business. The other viewpoint concerns inter-relationships between individual software requirements. The second view of traceability is important for understanding how changes propagate through a software system. The views are not orthogonal and both should be taken into account during RE. The first view on RT is adressed by VIBA in that requirements are derived from the I-Tables, which in turn are derived from Action diagrams that model the business. To make use of this traceability, the requirements contained in the software requirements specification must have references to the model elements to which they relate. There are approaches to the other view of RT (at least at the implementation level), such as program slicing (Weiser, 1984) and program dependence graphs (Podgurski & Clarke, 1990), which might be

useful. This is an area that we have just begun to explore and that remains to be studied in order to make explicit use of the business oriented action approach of ATIS and VIBA.

As of now, VIBA does not explicitly deal with requirements management, but builds a solid foundation for its success.

### 9.1.3 Pragmatization of Information Systems

One basic assumption within the field of, for example, entity-relationship modelling (Chen, 1976), and object orientation (e.g. Booch, 1994; Rumbaugh *et al.*, 1991) is that the information system should serve as an image, or simulation, of reality. The information system could then be used to inform its users about the world, so that the users do not have to observe the world directly. We do however believe that there exist dimensions in our minds beyond the strictly conceptual one. People do not just look at the world and talk about it. They do things, act and communicate in the world. Utterances carry more than mere facts that tell something about something. They also carry the actor's intention. Utterances *per se* can therefore by viewed as actions. These properties of information as action are seldom discussed or taken into account in traditional approaches to requirements engineering. A fact that often leads to systems that do not fulfil the needs of their users. We could talk about 'unactability' as a situation occurring when actors are not able to fulfil their intentions.

The key concept in traditional dynamic modelling is the event (e.g. Jackson, 1983; Cameron, 1989; Booch, 1994; Coad & Yourdon, 1991; Rumbaugh *et al.*, 1991). Booch (1994) defines an event as '*... some occurrence that may cause the state of a system to change*'. However, in a social context things most often do not just happen. An event, from the information system's point of view, is almost always the result of some action performed by some actor. Since an action is always based upon some intention we could say that an action is similar to an event but with additional attributes and constraints. Our approach thus gives an opportunity to identify the 'events' needed for system design based on the actions performed in the business. In that way the requirements specification does not just state what the system must be capable of doing but also why. Note that the same business action model is also used to identify interactive situations. Traceability from computerized system to business is thereby straightforward as regards both functional requirements and sequence restrictions. Snoeck & Dedene (1998) mention that '*modelling interaction by means of common event types requires the identification of relevant business event types*' but they do not mention how to do that. We argue that such relevant business event types are to be

found by identifying business actions. It is interesting to note that Blaha & Premerlani (1998) suggest that dynamic modelling most often is not necessary when developing database applications. They claim that objects in such a system do not have interesting dynamic behaviour. That is a claim in contradiction to Snoeck & Dedene (1998). The latter's method M.E.R.O.DE. is mainly focused on dynamic modelling and the concept of shared events – similar to JSD (Jackson, 1983). We believe that all actions, with corresponding events, that use or affect (create, modify or delete) the action memory are important for information systems engineering.

We propose information systems to be viewed as vehicles for communication among people and organizations. Such vehicles constitute a combination of:

- an action potential (a repertoire of actions and a vocabulary),
- a memory of earlier actions and action prerequisites, and
- actions performed interactively by the user and the system and/or automatically by the system.

We also propose that the most important quality of any IS is its actability. That is, its ability to perform actions and to permit, promote and facilitate users to perform their actions both through the system and based on messages from the system, in some business context.

We refer to this action enhancement as *pragmatization of information systems*.

### 9.1.4 Bridging the Requirements Engineering Gap

By perceiving business and IS usage as action it is possible (and necessary) to design business processes and information systems as an integrated whole. We propose the use of Action diagrams to describe action structures and to delimit interactive situation proposals (ISPs). The ISPs are then used as a basis for a deeper analysis of requirements concerning the design of interactive documents and action structures at a 'lower' level. Such analysis leads to deeper understanding of the business and often reveals inconsistencies in previous assumptions.

This way IS interaction design and implementation of different usability factors is both the result of business process modelling and an invaluable tool for its success, i.e. actability becomes a bridge over the requirements engineering gap (see Fig 9-1).

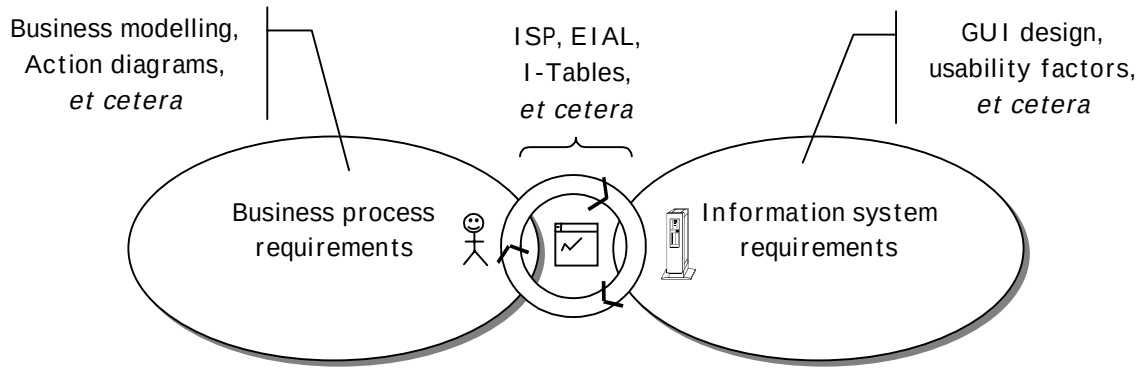



**Fig 9-1:** Actability–bridging the requirements engineering gap.

## 9.2 Related Work

Our approach to information systems and requirements engineering is based on the Language Action Perspective (LAP), which has gained more and more interest during the last few years (Dignum *et al.*, 1996; Dignum & Dietz, 1997; Goldkuhl *et al.*, 1998; 1999). Our approach thus tries to reconcile LAP and requirements engineering (RE) in order to make use of the best of both. In this section our work is related to some current (and important) trends within LAP and RE.

### 9.2.1 The Language Action Perspective

The best known approach within LAP is probably Action Workflow (Denning & Medina-Mora, 1995); cf. chapter 3. Other important approaches within LAP are DEMO (Reijswoud & Dietz, 1999), SAMPO (Auramäki *et al.*, 1998), and Commodious (Holm, 1996). In these approaches business processes (with support of information systems) are described as communicative action of various kinds. The DEMO approach uses a predefined set of communicative action types, structured in specified way, to describe business processes; similar to Action Workflow. The VIBA approach uses a six-stage model for business interaction – the BAT model – that bears some resemblance to Action Workflow, cf. chapter 3. The description of business processes (in the SIMM approach) follows this six-stage model, but uses it in a much more liberal way than Action Workflow and DEMO use their respective stage models. The predefined structure of actions is not imposed on the business process descriptions (in SIMM) in the same strict way as in these other approaches.

These other LAP approaches do not (as far as we know) go into the detail of requirements definition, even though the Commodious method (Holm, 1996) shows ambitions in that direction. This is also pointed out

by Dietz *et al.*, (1998) as being an urgent future research topic within LAP. One important feature of VIBA is the bridging of the RE gap between (1) business process modelling and (2) detailed modelling of documents and interaction within information systems. We do not recognize this ambition in these other approaches. Their scope thus seems to be narrower. The important concept of IS actability does not seem to be used in these other approaches.

### 9.2.2 Requirements Engineering

Our concept of interactive situation is quite similar to the use case concept (Jacobson *et al.*, 1992; 1995). Also our way of deriving interactive usage situations is quite similar to the way Jacobson *et al.* (1995) use the business object model to define use cases for the supporting information system. That similarity can be useful during systems design and implementation due to reuse of knowledge from the area of scenario-based RE. One major difference, however, lies in the fact that Jacobson's *et al.* (*ibid.*) definition of use case comprises '*...flow of events*' as a central notion. In contrast, our interactive situations are constituted by 'flow of actions'. Therefore the previous criticism (section 9.1.3) of the concept of event is also applicable to use cases. We thus disregard use cases and objects as modelling primitives during business modelling. Instead we use Action diagrams to capture and describe interactive situations. We believe that the formal systemic approach that is implicit in object orientation is not sufficient to understand a social system constituted by actors acting and interacting – a system such as a business. Therefore we have chosen not to talk about objects but rather about the terminology and concepts used in the business context. Since concepts' life cycles and propositional contents are analysed, we are, however, quite close to object orientation. The important point is that we do enforce object orientation during analysis. In fact, we believe that a requirements engineering method should deliver models that do not depend on any particular implementation paradigm. We do however believe that object orientation is a powerful and valuable approach when it comes to the computerized part of the information system. We also believe that if we chose an object-oriented implementation strategy, the documentation from VIBA is a good starting point for object-oriented analysis and design (i.e. much of the work has already been done).

A problem regarding use cases is on which level they should be defined (e.g. Vemulapalli, 1995). We aggregate activities based on the actors' intentions to delineate interactive usage situations. That approach should also be applicable to use cases if a move from the concept of event to the concept of action was performed. Hence, one line of future work is

to investigate similarities and differences between VIBA and different scenario based approaches more systematically, and possibly use ATIS as a theoretical foundation for scenario-based RE.

Our understanding of the RE process as a continuing process that spans the lifecycles of possibly many single information systems is a notion similar to those proposed by recent RE work (e.g. Sawyer *et al.* 1998; Lam *et al.*, 1998) in order to cope with requirements change. We do believe that more insight can be gained by use of, for example, Business Action Theory (chapter 3; Goldkuhl, 1998) in order to understand the business and social impacts of RE and change management.

Goal driven RE (e.g. Yu & Mylopoulos, 1998) is another area that is relevant to our work. We believe that communicative action is always based on some intentions and intention can be viewed as a kind of goal. Goals are, for example, used in VIBA when analysing the existing business. The goals of current actions and action structure can be used to transfer seemingly tacit knowledge and good practice when designing new businesses.

When applying a rapid development approach, as suggested by DSDM (1999), selection and reduction of requirements, to keep time and resources fixed, (see chapter 1), are important topics. Some work has been performed in this area by, for example, Karlsson (1995). This is an important future research topic: how to select and reduce requirements and still maintain a high degree of actability.

## 9.3 Some Final Words

This thesis is concerned with theory and method for information systems engineering. The reported work has been performed as a combination of theoretical work and action research. '*Sure it works in practise, but will it work in theory?*'. The question was brought up by Erik Clemons during his key note speech at the 1999 IRMA conference in Hershey Pennsylvania. A question with humorous undertones but with serious implications. One aim of our research is to create (or reveal, depending on metaphysical position) knowledge that 'works' both in theory and practice. To theorize about practise and to pragmatize theory is, according to our belief, the task with a capital T of academic IS research. Unfortunately it sometimes seem easier to try to change good practice than to change bad theory.

# References


Action Technologies (1993). *Action Workflow Analysis Users Guide*. Action Technologies.

Auramäki E, Lehtinen E, Lyytinen K (1988). A speech-act-based office modeling approach. *ACM Trans of OIS*, Vol. 6, No. 2, pages 126-152.

Austin J L (1962). *How to do things with words*. Oxford University press.

Avdic A (1999). *Användare och utvecklare – om anveckling med kalkylprogram*. In Swedish. Doctoral dissertation, Deptartment of Computer and Information Science, Linköping University.

Berger P, Luckmann T (1989). *The social construction of reality*. Anchor Books, New York.

Blaha M, Premerlani W (1998). *Object-oriented modeling and design for database applications*. Prentice hall, Inc. Englewood Cliffs, New Yersey.

Boehm B W (1988). A spiral model of software development and enhancement. *IEEE Computer*, 21 (5), 61–72.

Booch G (1994). *Object-oriented analysis and design with applications*. 2nd ed. Benjamin/Cummings.

Booch G, Rumbaugh J, Jacobson I (1999). *The Unified Modeling Language User Guide*. Addison Wesley Longman, Inc.

Brinkkemper S, Saeki M, Harmsen F (1998). Assembly Techniques for Method Engineering. In Pernici B, Thanos C (eds.), *Proceedings of 10th International Conference on Advanced Information Systems Engineering*. Pisa, Italy, June 8–12, 1998.

Brinkkemper S. (1996). Method Engineering: Engineering of Information Systems Development Methods and Tools. In Wrycza-Zupancic (ed.), *Proceedings of the Fifth International Conference on Information Systems Development* (ISD'96). Gdansk, Poland.

Bubenko J A jr, Kirikova M (1999). Improving the Quality of Requirements Specifications by Enterprise Modelling. In Nilsson A G, Tolis C, Nellborn C (eds.), *Perspectives on Business Modelling – Understanding and Changing Organisations*. Springer Verlag, Heidelberg.

Bubenko J, Källhammar O (1969). Computer Aided Design of Information Systems. CADIS Working Paper – 1, sept 1969. Objectives for Research in Information System Design. In Bubenko J jr, Källhammar O, Langefors B, Lundeberg M, Sölvberg A (Eds.), *Systemering 70*, pp. 395–403. Studentlitteratur, Lund, Sweden, 1970.

Budde R, Kautz K, Kuhlenkamp K, Züllighoven H (1992). *Prototyping: An approach to evolutionary system development*. Springer-Verlag.

Cameron J (1989). *JSP & JSD: The Jackson approach to software development*. IEEE computer society press.

Card S K, Anderson A (1987). A multiple virtual-workspace interface to support task swithing. In *Proceedings of the Human Factors in Computing Systems CHI'87*, pp 53–9. New York, ACM Press.

Checkland P (1981). *Systems Thinking, Systems Practice*. Chichester, UK, Wiley.

Chen P (1976). The entity-relationship model – toward a unified view of data. *ACM TODS* 1, No. 1 mars 1976.

Coad P, Yourdon E (1991). *Object-oriented analysis*. Object international, Inc. Published by Prentice Hall, Inc. Englewood Cliffs, New Yersey.

Computer Sweden (1999). Computer Sweden. (Swedish IT newspaper). Year 17, No. 24, Mars 8.

McConnel S (1996). *Rapid development*. Microsoft press.

Cooper A (1994). The Perils of Prototyping. *Visual Basic Programmers Journal*, August September 1994.

Cronholm S, Ågerfalk P J (1999). On the Concept of Method in Information Systems Development. In Käkölä T (ed.), *Proceedings of 22nd Information Systems Research Seminar in Scandinavia* (IRIS 22), Keuruu, Finland, August 7–10, 1999.

Cronholm S, Ågerfalk P J, Goldkuhl G (1999). From Usability to Actability. In Bullinger, Ziegler (eds.), *Proceedings of 8th International Conference on Human-Computer Interaction* (HCI International'99), Munich, August 22–27, 1999.

Davenport T H (1993). *Process Innovation – Reengineering Work through Information Technology*. Boston, MA: Harvard Business School Press.

Denning P J, Medina-Mora R (1995). Completing the loops. *Interfaces*, Vol. 25, No 3, pp. 42–57.

Dietz J L G, Goldkuhl G, Lind M, Reijswoud V E van (1998). The Communicative Action Paradigm for Business Modelling – A Research Agenda. In Goldkuhl G *et al.* (eds.), *Proceedings of 3rd International Workshop on the Language Action Perspective on Communication Modelling* (LAP'99). Stockholm, Sweden, June 25–26, 1998.

Dignum F, Dietz J (eds. 1997). Communication modelling – the Language Action Perspective. Proceedings of the 2nd International workshop on Communication modelling, Computer science Reports, Eindhoven.

Dignum F, Dietz J, Verharen E, Weigand H (eds. 1996). Communication modelling – the Language Action Perspective. Proceedings of the 1st International workshop on Communication modelling, Electronic Workshops in Computing, Springer Verlag.

DSDM (1999). *DSDM Method Overview*. DSDM Consortium, WWW document: http://www.dsdm.org/oview.htm, 14/4/1999.

DuPont (1989). Information from W. Robert White, Information Engineering Associates, E. I. DuPont de Nemours and Company, Inc., Wilmington, Delaware. Cited by Martin (1991).

Falkenberg E D, Hesse W, Lindgreen P, Nilsson B E, Oei J L H, Rolland C, Stamper R K, Van Assche F J M, Verrijn-Stuart A A, Voss K (1998). *A Framework of Information System Concepts: The FRISCO Report (Web edition)*. IFIP. Available from ftp://ftp.leidenuniv.nl/pub/rul/fri-full.zip, 28/6/1999.

Flores F, Ludlow J (1980). Doing and speaking in the office. In Fick, Sprague (eds.), *Decision support systems: Issues and challenges*. Pergamon Press.

Fowler S (1997). *Analysis Patterns*. Addison-Wesley Publishing Company, Inc.

Fåhraeus E (1986). *Metodkedjornas och verktygens roll vid systemutveckling*. In Swedish. Rapport nr 24 Riksdataförbundet, Stockholm.

Gamma E, Helm R, Johnson R, Vlissides J (1995). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley Publishing Company Inc.

Goldkuhl G (1980). *Framställning och användning av informationsmodeller*. In Swedish. Doctoral dissertation. Dept. of Information Processing, University of Stockholm, Sweden.

Goldkuhl G (1991). *Stöd och struktur i systemutvecklingsprocessen*. In Swedish. Deptartment of Computer and Information Science, Linköping University.

Goldkuhl G (1992). Contextual Activity Modelling of Information Systems. In *Proceedings of the 3rd International Working Conference on Dynamic Modelling of Information Systems*. Noordwijkerhout.

Goldkuhl G (1993). *Verksamhetsutveckla Datasystem*. In Swedish. Intention, Linköping.

Goldkuhl G (1994). *Välgrundad Metodutveckling*. In Swedish. Department of Computer and Information Science, Linköping University.

Goldkuhl G (1995). *Information as Action and Communication*. In Dahlbom B (ed.), *The Infological Equation: Essays in the Honor of Börje Langefors*, Gothenburg Studies in Information Systems.

Goldkuhl G (1998). *The Six Phases of Business Processes – Business Communication and the Exchange of Value.* Presented at Beyond Convergence: The 12th Biennial ITS Conference (ITS'98) in Stockholm. Jönköping International Business School.

Goldkuhl G, Lind M, Seigerrot U, Ågerfalk P J (eds., 1999). *Proceedings of 4th International Workshop on the Language Action Perspective on Communication Modelling* (LAP'99). Copenhagen, Denmark, September 12–13, 1999. Jönköping International Business School.

Goldkuhl G, Lind M, Seigerroth U (1997). Method integration: The need for a learning perspective. In Magee J N *et al.* (eds.) *IEE Proceedings – Software,* Vol. 145, No. 4, August 1998, pp. 113–8.

Goldkuhl G, Lind M, Seigerroth U (eds., 1998). *Proceedings of 3rd International Workshop on the Language Action Perspective on Communication Modelling* (LAP'99). Stockholm, Sweden, June 25–26, 1998. Jönköping International Business School.

Goldkuhl G, Lyytinen K (1982). *A Language Action View of Information Systems*. In Ginzberg, Ross (eds.), *Proceedings of the 3rd International Conference on Information Systems*. Ann Arbor, December 13–15, 1982.

Goldkuhl G, Röstlinger A (1988). *Förändringsanalys: Arbetsmetodik och förhållningssätt för goda förändringsbeslut*. In Swedish. Lund: Studentlitteratur.

Goldkuhl G, Röstlinger A (1993). Joint elicitation of problems: An important aspect of change analysis. In Avison D *et al.* (eds.), *Proceedings, Human, organizational and social dimensions of Information systems development.* North-Holland/IFIP WG 8.2.

Goldkuhl G, Ågerfalk P J (1998). Action Within Information Systems: Outline of a Requirements Engineering Method. In Dubois E, Opdahl A, Pohl K (eds.), *Proceedings of 4th International Workshop on Requirements Engineering: Foundation for Software Quality*. Pisa, Italy, June 8–9, 1998.

Graham I (1998). *Requirements Engineering and Rapid Development: An Object-oriented Approach*. ACM Press, Addison-Wesley.

Greithuysen J J van (ed. 1982). *Concepts and Terminology for the Conceptual Schema and the Information Base.* International Organization for Standardization, ISO/TC97/SC5 – N 695.

Gulliksen J (1999). Bringing in the Social Perspective: User Centred Design. In Bullinger, Ziegler (eds.), *Proceedings of 8th International Conference on Human-Computer Interaction* (HCI International'99), Munich, August 22–27, 1999.

Habermas J (1984). *The theory of communicative action 1. Reason and the rationalization of society*. Beacon Press.

Hammer M, Champy J (1993). *Reengineering the corporation. A manifesto for business revolution*. Nicholas Brealey, London.

Harel D (1987). Statecharts: A Visual Formalism for Complex Systems. In *Science of Computer Programming*, 8 (1987) pp. 231–74.

Harmsen A F (1997). *Situational Method Engineering*. Doctoral disertation, Moret Ernst & Young Management Consultants, Utrecht, The Netherlands.

Holm P (1996). *On the Design and Usage of Information Technology and the Structuring of Communication and Work*. Doctoral dissertation, Stockholm University, Sweden.

Holm P, Ljungberg J (1996). Multi-Discourse Conversations. In *Proceedings of 4th European Conference on Information Systems (ECIS'96)*, Lisbon.

Horrocks I (1999). *Constructing the user interface with statecharts*. Addison-Wesley.

Hägerfors A (1994). *Co-learning in participative systems design, Enhancements of genuine participation by consideration of communication and group dynamics*. Lund University.

Iivari J, Lyytinen K (1998). Research on Information Systems Development in Scandinavia – Unity in Plurality. *Scandinavian Journal of Information Systems*, Vol. 10, No. 1&2, pp. 135–86.

Jackson M A (1983). *System development*. Prentice Hall, Inc. Englewood Cliffs, New Yersey.

Jackson M, Twaddle G (1997). *Business Process Implementation: Building Workflow Systems*. ACM Press. Addison Wesley.

Jacobson I, Christerson M, Jonsson P, Övergaard G (1992). *Object-oriented software engineering: a use case driven approach*. ACM-press.

Jacobson I, Ericsson M, Jacobson A (1995). *The object advantage: business process reengineering with object technology*. ACM-press.

Jayaratna N (1994). *Understanding and Evaluating methodologies*. McGraw-Hill Book Company, London.

Karlsson J (1995). *Towards a Strategy for Software Requirements Selection*. Licentiate Thesis. Dept. of Computer and Information Science, Linköping University, Sweden.

Keen P G W (1997). *The Process Edge: Creating Value Where it Counts*. Boston, MA: Harvard Business School Press.

Kuhn T C (1970). *The structure of scientific revolution*. University of Chicago Press, USA.

Lam W (1998). Change Analysis and Management in a Reuse-oriented Software Development Setting. In Pernici B, Thanos C (eds.), *Proceedings of 10th International Conference on Advanced Information Systems Engineering*. Pisa, Italy, June 8–12, 1998.

Lam W, Shankararaman V, Jones S (1998). Managing Requirements Change: A set of Good Practices. In Dubois E, Opdahl A, Pohl K (eds.), *Proceedings of 4th International Workshop on Requirements Engineering: Foundation for Software Quality*. Pisa, Italy, June 8–9, 1998.

Langefors B (1966). *Theoretical Analysis of Information Systems*. 4th ed., Studentlitteratur, Lund, Sweden, 1974.

Langefors B (1995). *Essays on Infology*. Dahlbom B (Ed.), Studentlitteratur, Lund, Sweden.

Lauesen S, Younessi H. (1998). Six Styles for Usability Requirements. In Dubois E *et al.* (eds.), *Proceedings of 4th International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'98)*. Pisa, Italy, June 8–9, 1998.

Levén P (1995). *Från användning till handling*. In Swedish. Licentiate Thesis, Umeå University.

Lif M (1998). *Adding Usability. Methods for Modelling, User Interface Design and Evaluation*. Doctoral dissertation, Uppsala University.

Lind M (1996). *Affärsprocessinriktad Förändringsanalys – utveckling och tillämpning av synsätt och metod*. In Swedish. Licentiate Thesis, Dept. of Computer and Information Science, Linköping University.

Lipschutz S (1976). *Schaum's Outline of Theory and Problems of Discrete Mathematics*. McGraw-Hill, Inc.

Ljungberg J, Holm P (1996). Speech Acts on Trial. *Scandinavian Journal of Information Systems*. Vol 8, Nr 1.

Lyytinen K (1983). *Reality Mapping or Language Development – A Tentative Analysis of Alternative Paradigms for Information Modeling*. Syslab Wp No 27, April 1983, Stockholm University.

Löwgren J (1993). *Human-computer interaction – What every system developer should know*. Lund: Studentlitteratur.

Löwgren J (1995). *Perspectives on Usability*. Department of Computer and Information Science, Linköping University.

Magoulas T, Pessi K (1991). *En studie om informationssystemarkitekturer*. In Swedish. Department of Computer Sciences, Chalmers University of Technology and the University of Göteborg.

Maiden N A M, Cisse M, Manuel D (1998). CREWS Validation Frames: Patterns for Validating Systems Requirements. In Dubois E *et al.* (eds.), *Proceedings of 4th International Workshop on Requirements Engineering: Foundation for Software Quality*. Pisa, Italy, June 8–9, 1998.

Martin J (1989). *Information engineering – Introduction*, Englewood Cliffs NJ: Prentice Hall.

Martin J (1991). *Rapid Application Development*. Macmillan Publishing Company, New York.

Mathiassen L (1982). *Systemudvikling og systemudviklingsmetode.* In Danish. Doctoral disertation. Datalogisk afdelning, matematisk institut, Aarhus universitet.

Mathiassen L, Seewaldt T, Stage J (1995). Prototyping and Specifying: Principles and Practices of a Mixed Approach. *Scandinavian Journal of Information Systems*, Vol. 7, Nr. 1.

Monk A, Wright P, Haber J, Davenport L (1993). *Improving your Human-Computer Interface*. New York, Prentice Hall.

Nielsen J (1993). *Usability Engineering*. Academic Press, San Diego, CA.

Nilsson A (1991). *Anskaffning av standardsystem för att utveckla verksamheter*. In Swedish. Doctoral dissertation. Ekonomiska forskningsinstitutet (EFI), Handelshögskolan i Stockholm

Nilsson A G (1999). The Business Developer's Toolbox: Chains and Alliances between Established Methods. In Nilsson A G, Tolis C, Nellborn C (eds.), *Perspectives on Business Modelling – Understanding and Changing Organisations*. Springer Verlag, Heidelberg.

Nilsson I, Raunio L-L, Cronholm S, Öberg F (1997). *Systemutvecklingsprojekt*. In Swedish. Lecture notes (kurskompendium). Department of Computer and Information Science, Linköping University.

Norman D A (1988). *The Psychology of Everyday Things*. Basic Books. New York.

Nurminen M (1988). *People or Computers: Three Ways of looking at Information Systems*. Lund: Studentlitteratur.

Näslund T (1996). Seven traditions for information systems development. In Dahlbom B *et al.* (eds.), *Proceedings of 19th Information Systems Research Seminar in Scandinavia* (IRIS 19). Gothenburg Studies of Informatics, Report 8, June 1996.

Opdahl A L (1998). Multi-perspective modelling of requirements: a case study using facet models. In Fowler D, Dawson L (eds.), *Proceedings of 3rd Australian Conference on Requirements Engineering*. Geelong, Vic, Australia, Oct 26–27, 1998. Deakin university.

Parnas D L (1971). On the Criteria to be Used in Decomposing Systems into Modules. *Communications of the ACM*, Vol. 5, No. 12, December 1972.

Patton M Q (1990). *Qualitative Evaluation and Research Methods*. 2nd ed. Newbury Park, California: SAGE Publications, Inc.

Podgurski A, Clarke L (1990). *A formal model of program dependencies and its implications for software testing, debugging and maintenance.* IEEE Transactions on Software Engineering, 16(9), pages 965-79.

Porter ME (1985). *Competitive advantage. Creating and sustaining superior performance.* Free Press, New York.

Preece J, Rogers Y, Sharp H, Benyon D (1994). *Human Computer Interaction*. Addison-Wesley.

Reijswoud V E van, Dietz J L G (1999). Business Process Re-Engineering With DEMO. In Goldkuhl *et al.* (eds.), *Proceedings of 4th International Workshop on the Language Action Perspective on Communication Modelling* (LAP'99). Copenhagen, Denmark, September 12–13, 1999.

Reijswoud V E van, Lind M (1998). Comparing Two Business Modelling Approaches in the Language Action Perspective. In Goldkuhl G *et al.* (eds.), *Proceedings of 3rd International Workshop on the Language Action Perspective on Communication Modelling* (LAP'99). Stockholm, Sweden, June 25–26, 1998.

Rumbaugh J (1995). *What is a method?* Technical Paper, Rational software, Inc.

Rumbaugh J, Blaha, M, Premerlani W, Eddy F, Lorensen W (1991). *Object-oriented modeling and design*. Prentice hall.

Röstlinger A, Goldkuhl G (1994). *Generisk flexibilitet – På väg mot en komponentbaserad metodsyn*. In Swedish. Dept. of Computer and Information Science, Linköping University.

Sawyer P, Sommerville I, Viller S (1998). Improving the Requirements Process. In Dubois E *et al.* (eds.), *Proceedings of 4th International Workshop on Requirements Engineering: Foundation for Software Quality*. Pisa, Italy, June 8-9, 1998.

Searle J R (1969). *Speech Acts. An essay in the philosophy of language.* London: Cambridge University Press.

Searle J R (1979). *Expression and Meaning. Studies in the Theory of Speech Acts*. London: Cambridge University Press.

Shackel B. (1984). The Concept of Usability. In Bennet J, Case D, Sandelin J, Smith M (eds.), *Visual Display Terminals: Usability Issues and Health Concerns*. Englewood Cliffs NJ: Prentice Hall.

Shneiderman B (1998). *Designing the user interface: strategies for effective human-computer-interaction*. Addison Wesley.

Sims O (1994). *Business Objects – delivering co-operative objects for client-server*. McGraw-Hill, Berkshire, England.

Smith M F (1991). *Software prototyping: adoption, practice and management*. McGraw-Hill.

Snoeck M, Dedene G (1998). *Existence dependency: The key to semantic integrity between static and behavioural aspects of object types*. Research report. Katholieke universiteit Leuven, Belgium. To appear in IEEE software.

Sommerville I (1996). *Software engineering*. 5th ed. Addison-Wesley.

Stamper R (1988). Analysing the Cultural Impact of a System. *International Journal of Information Management*, vol 8, no 3.

Strauss A, Corbin J (1998). *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. 2nd ed. Thousand Oaks, California: SAGE Publishing, Inc.

Sundgren B (1973). *An Infological Approach to Databases*. Doctoral dissertation. SCB, Sweden.

Sundgren B (1992). *Databasorienterad systemutveckling*. In Swedish. Studentlitteratur, Lund, Sweden.

Törnebohm H (1976). En systematik över paradigm. In Swedish. Rapport nr 85. Institutionen för vetenskapsteori, Göteborgs universitet, Göteborg.

Vallgren A (1992). *Verksamhets- & Informationsbehovsanalys/SIMMetoden i 4GL-miljö – erfarenheter från utveckling av händelseregister vid Bravikens pappersbruk*. In Swedish. Bachelor's thesis. Department of Computer and Information Science, Linköping University.

Weiser, M (1984). *Program slicing*. IEEE Transactions on Software Engineering, 10(4), pages 352–7.

Vemulapalli C (1995 ). *A Use Case FAQ*. Advanced Software Technologies Group, WorldCom Inc. Contribution to the first Workshop on Use Cases/OOPSLA '95

Verharen E (1997). *A language-action perspective on the design of cooperative information agents*. Doctoral dissertation, KUB, Tilburg.

Whitaker (1992). *Venues for Contexture*. Doctoral dissertation. Umeå University.

Winograd T, Flores F (1986). *Understanding computers and cognition: A new foundation for design*. Ablex, Norwood.

Wittgenstein L (1958). *Philosophical Investigations*. Oxford: Basil Blackwell & Mott Ltd.

Yourdon E (1989). *Modern structured analysis*. Prentice Hall, Inc. Englewood Cliffs, New Yersey.

Yu E, Mylopoulos J (1998). Why Goal-Oriented Requirements Engineering. In Dubois E *et al.* (eds.), *Proceedings of 4th International Workshop on Requirements Engineering: Foundation for Software Quality.* Pisa, Italy, June 8–9, 1998.

Ågerfalk P J (1999). Encapsulation or Availability: On the Combination of Objects and Relations in Systems Development. In Käkölä T (ed.), *Proceedings of 22nd Information Systems Research Seminar in Scandinavia* (IRIS 22), Keuruu, Finland, August 7–10, 1999.

Ågerfalk P J, Goldkuhl G (1998). Elicitation and Analysis of Actability Requirements. In Fowler D, Dawson L (eds.), *Proceedings of 3rd Australian Conference on Requirements Engineering.* Geelong, Vic, Australia, October 26–27, 1998. Deakin university, School of management information systems.

Ågerfalk P J, Goldkuhl G, Cronholm S (1999). Information Systems Actability Engineering: Integrating Analysis of Business Processes and Usability Requirements. In Goldkuhl G *et al.* (eds.), *Proceedings of 4th International Workshop on the Language Action Perspective on Communication Modelling* (LAP'99). Copenhagen, Denmark, September 12–13, 1999.

Ågerfalk P J, Åhlgren K (1999). Modelling the Rationale of Methods. In Khosrowpour M (ed.), *Managing Information Technology Resources in the Next Millennium,* proceedings of the 1999 Information Resources Management Association International Conference. Hersey, PA, USA, May 16–19, 1999.

Öberg F (1998). *Object-Oriented Frameworks – A New Strategy for CASE Tool Development.* Licentiate Thesis. Dept. of Computer and Information Science. Linköping University.

Överström P (1994). *Övergång från Verksamhets- och Informationsbehovs-analys enligt SIMMetoden till implementering i fjärde generationens utvecklingsverktyg.* In Swedish. Bachelor's Thesis, Department of Computer and Information Science, Linköping University.

LINKÖPINGS UNIVERSITET

**Avdelning, institution**
Division, department

Institutionen för datavetenskap

Department of Computer and Information Science

**Datum**
Date

1999-09-30

**Språk**
Language

☐ Svenska/Swedish
☑ Engelska/English

☐ _____

**Rapporttyp**
Report category

☑ Licentiatavhandling
☐ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport

☐ _____

**ISBN** 91-7219-610-6

**ISRN**

**Serietitel och serienummer** **ISSN**
Title of series, numbering 1401-4637

Faculty of Arts and Science thesis FiF-a 30

**URL för elektronisk version**

**Titel**
Title

# Pragmatization of Information Systems – A Theoretical and Methodological Outline

**Författare**
Author

Pär J. Ågerfalk

**Sammanfattning**
Abstract

Pragmatization of information systems means to climb the semiotic ladder from the syntactic and semantic levels, and take the pragmatic use of language as a starting point for understanding information and information systems as social phenomena. This thesis concerns theory and method for conceiving and developing information systems based on such an understanding. We propose an action theory of information systems, which is founded in the information system methods tradition, based on a language action perspective, and influenced by current and important trends within the fields of information systems, software engineering, and human-computer interaction. As a methodological consequence of the theory, a re-design of an existing systems engineering method has been performed as a combination of theoretical work and action research. Both the theory and the method, as well as the empirical work, are presented and elaborated upon in this thesis.

**Nyckelord**
Keywords

Actability, Systems Engineering, Requirements Engineering, Information Systems Theory, Language Action Perspective, Speech Act Theory, Usability