Elicitation and Analysis of Actability Requirements[!]

Pär J. Ågerfalk[1, 2] and **Göran Goldkuhl**[3, 2]

[1] Dept. of Informatics, Örebro University, SE-701 82 Örebro, Sweden
E-mail: pak@esa.oru.se

[2] Centre for studies on humans, technology and organization (CMTO)
Linköping University, SE-581 83 Linköping, Sweden

[3] Dept. of Informatics, Jönköping International Business School
P.O. 1026, SE-551 11 Jönköping, Sweden
E-mail: ggo@ida.liu.se

Key words: Language Action Perspective; Actability; Usability; Requirements Engineering; Information Systems Development;

Abstract

This paper presents an approach to requirements elicitation and analysis based on an action view of information systems. The approach aims at helping developers to elicit requirements based on the business context in which the planed system is to be used and on the business actions that is to be performed with and within it. It is shown how the approach supports traceability of requirements, from business model to system design, in order to ease the burden of system development, maintenance and evolution.

[!] Formal reference to this paper is: Ågerfalk P J, Goldkuhl G (1998). Elicitation and Analysis of Actability Requirements. In *Proceedings of the 3rd Australian Conference on Requirements Engineering (ACRE'98)*, pp. 14-28. Fowler, D and Dawson, L (Eds.). School of Management Information Systems, Deakin University, Geelong, Vic, Australia.

1. Introduction

It is a well-known fact that elicitation and management of requirements is a difficult task in all software and system engineering efforts. Poor understanding and handling of requirements often lead to systems that do not meet user expectations and are hard to maintain and further enhance. This paper presents an approach to requirements elicitation and representation founded on an action view of information systems. Such a view emphasises what people do while communicating by means of computerised information systems (Goldkuhl & Ågerfalk, 1998)

The presented techniques and modelling notations is part of the phase "interaction analysis" within the requirements engineering method VIBA/SIMM that is developed at the Centre for studies on humans, technology and organization (CMTO) at Linköping University, Sweden. VIBA[1] is an acronym for Versatile Information and Business requirements Analysis and SIMM should be interpreted as Situation adaptable work and Information systems Modelling Method.

Interaction analysis focuses on the interaction between an IS and its users. It is therefore mainly functional requirements and requirements regarding interface layout to support user actions, i.e. what we call the actability of the IS, that are addressed. Other requirements are handled within other parts of VIBA and it is outside the scope of this paper to go into detail of those parts.

The Requirements Engineering Process

Requirements engineering (RE) is really a kind of "never-ending story". The story begins with some fuzzy ideas about how a computerised information system (IS) might support the way business is done (or perhaps ought to be done). The process of eliciting and formalising such ideas into a requirements document 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 (McConnel, 1996). Thus requirements change management becomes an important part of the entire RE process (c.f. Lam, 1998; Lam et al, 1998). Even when an IS is to be phased out the RE process continues. By preserving requirements documents (including the changes made to them with the rationale behind those changes) developers can learn from previous experiences when building new information systems, c.f. the use of patterns in software engineering (Gamma et al, 1995; Maiden et al, 1998). Thus, in order to support the RE process as a whole, a method for elicitation and analysis of requirements must support traceability of requirements from the IS to the business of which the system is a crucial part. That is, the produced requirements specification must support traceability from detailed system design all the way back to business model.

Information Systems as Action

Many approaches and theories within the field of information systems are based on a strict representational view of information. One main purpose of requirements engineering is, with that view, to get an accurate "image" of reality in order to have the analysed piece of reality

[1] VIBA is used as replacement for the now obsolete term BIA from previous work, e.g. Goldkuhl & Ågerfalk (1998).

properly represented in the (database of the) IS. This strict representational view can be challenged in several ways, which a language action perspective certainly does (Goldkuhl & Lyytinen, 1982; Winograd & Flores, 1986). Information systems are here not considered as ”containers of facts” or ”instruments for information transmission”. Information systems are instead seen as vehicles for organisational communicative action (Goldkuhl & Ågerfalk, 1998). In the language action perspective communication and language use are conceived as action. *To speak is to act* following speech act theory (Austin, 1962; Searle, 1969). In this theory utterances and messages are described as consisting of two parts: A propositional content (”what is talked about”) and an action mode (”what speaker does in relation to listener”). We do not only use utterances and messages to describe the world around us. There are also performative utterances as e.g. requests, questions, proposals, orders, promises and claims.

Information and processing in a computerised IS is reinterpreted following a language action view. Information is not restricted to a propositional content as in a strict representational view. In many information systems the action character of information (stored in a database or represented on a user interface) is implicit or obscure. The processing of information means in this view performing organisational action. One important feature of the proposed RE approach is to design information systems with more visible and transparent action characters. VIBA is based on such a language action perspective (Goldkuhl & Ågerfalk, 1998) and we define information system as consisting of:

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

One of the most important features of an information system is its *actability,* i.e. its capability to perform actions and to permit and facilitate users to perform their actions both through the system and based on documents (messages) from the system. Requirements engineering will in this context be a key to successful design of actability.

2. Brief overview of VIBA/SIMM

When using VIBA/SIMM the traditional borderline between business modelling and systems development is more or less obsolete. The reason is that VIBA treats analysis and design of business processes and information systems as an integrated whole. The rationale for such an approach is that we believe that information systems are such a vital part of today’s business’ that it’s impossible to view system development as an isolated activity. When changing an IS the business is changed as well. Thus by viewing system development as business development the change of the business becomes conscious and well managed.

Development according to VIBA is divided into two main components; Business Process Modelling (BPM) and Information System focused Modelling (ISM), see figure 1. During BPM the business goals, problems, strengths and weaknesses, action structure and other development constraints are analysed. These activities are often done as a direct continuation of some feasibility study or change analysis, e.g. Change analysis (CA) according to SIMM (Goldkuhl & Röstlinger, 1993), that is supposed to be done before system development starts. The main outcome of BPM is a business action model documented in action diagrams[2]. Such diagrams show the action logic of the business with results of and prerequisites for actions.

[2] An example of action diagrams used in the modelling of a system to support a travel agency can be found in appendix.

The action diagrams are explicit regarding what actions are to be performed by or in interaction with planned information systems.

The action diagrams of BPM are used as a basis for ISM, which consists of three main areas; Interaction analysis, Conceptual analysis and Document analysis. Interaction analysis (which is described in more detail in the next section) is concerned with the user interaction, i.e. how the communication by use of the IS is to be performed. During Conceptual analysis the focus is on what to communicate, i.e. the concepts being used, their relationships and lifecycles. Document analysis is about the documents being used as carriers of messages and their relationships (both conceptual and based on actions).

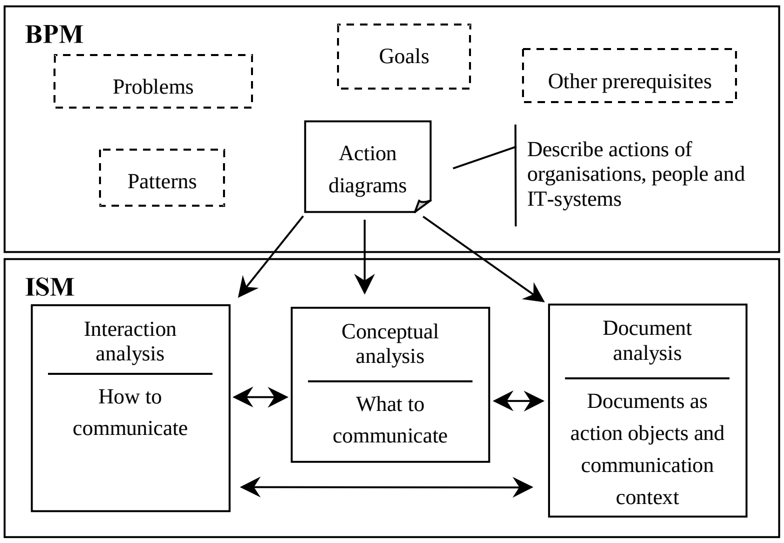


Fig. 1. Main components of VIBA/SIMM

As mentioned above this paper's focus is on Interaction analysis and it is therefore out of the scope to go into detail of the other parts of VIBA. See (Goldkuhl & Ågerfalk, 1998) for a somewhat more exhaustive treatment.

3. Interaction Analysis

Interaction analysis (IA) is concerned with the interaction between users and information systems. Its main purpose is to bring an understanding of how the communication through the IS is to be performed. IA thus means requirements elicitation and analysis of the roles of screen documents, relations between different interactive situations, interactive actions (of both user and IS) and maintenance of action memory (e.g. the database).

Interaction analysis follows an iterative process that uses a combination of analytical modelling and prototyping. Interactive situation proposals are derived from the interactive actions of action diagrams of BPM. These are analysed and validated by use of models and prototypes. Requirements elicitation, analysis and validation are then iterated until the system meets the required actability (Goldkuhl & Ågerfalk, 1998).

This section presents the part of interaction analysis that is concerned with the analytical modelling of actability requirements. We start by presenting a state model of user interaction, which is used as theoretical foundation for the analysis of interactive situations presented subsequently.

A State Model of User Interaction

In order to (analytically) analyse user interaction there is a need for a concept of smallest possible interactive communicative user action, i.e. actions that cannot reasonably be divided into sub-actions. Such actions are referred to as *atomic actions.* Thus an interactive situation is constituted by a non-empty set of atomic actions with associated sequence restrictions.

When analysing user interaction we adhere to the notion of a discrete system as always being in some well-defined state. When dealing with user interaction there is a need to distinguish between two different systems (of which one is a part of the other). The IS, i.e. the computer artefact, is by definition a system. Since the analysis also considers the interacting user there is another system constituted by the communication context, i.e. the IS and the interacting user. In this section the computerised artefact is thus referred to as IS and the communication context is referred to as CC.

When a human actor (i.e. the user) performs an atomic action the state of CC is changed. As a response to the actor's action, the IS probably performs another action in turn, i.e. the functionality requested by the user is executed. That IS action then again changes the state of CC. This would yield a model of user interaction with three states and two actions in the prescribed order. However, in the context of social interaction (of which IS usage is an example) an action always reflects on the actor (e.g. Giddens, 1984). In order to perceive that reflection the actor always performs an interpretation act in response to his/her previous intentional action.

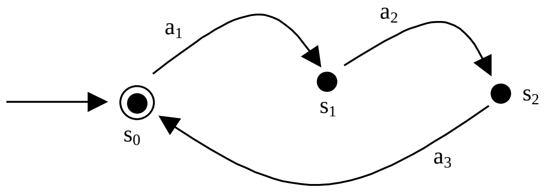

$$\begin{tikzcd} \arrow[r] & s_0 \arrow[r, bend left, "a_1"] & s_1 \arrow[r, bend left, "a_2"] & s_2 \arrow[ll, bend left, "a_3"] \end{tikzcd}$$

Fig. 2. The state model of interaction as a finite automaton

By applying this theoretical reasoning to user interaction we need to extend the model to consist of three states and three actions. The first state of CC, s_0, is the initial state that constitutes prerequisites for the user action a_1, i.e. s_0 is a precondition for a_1. The second state s_1 is then reached by performing a_1. As a response to a_1 the IS automatically performs a_2 which puts CC in state s_2. Note that a_2 might be constituted by a series of IS actions. Finally the user performs the interpretation action a_3 to perceive what the IS accomplished.

Figure 2 shows the proposed state model of interaction as a finite automaton. In the model state s_0 is both initial state and the only accepting state since once an action a_1 is performed both a_2 and a_3 must be performed in order to keep the systems consistent. It is important to notice that the interpretation act a_3 does not change the state of the computerised part of the system, i.e. the IS, but only the system constituted by the IS together with the interacting user, i.e. CC.

A model of user interaction usually contains many atomic actions that are performed in some sequence, and probably grouped into interactive situations. The presented state model is applicable to every such atomic action since s_2 in atomic action n usually becomes s_0 during atomic action n+1 – from the IS point of view. The IS is thus ready for another user action a_1 (i.e. a new atomic action) as soon as it completes a_2. Note that the same atomic action (at the type level) can be part of several interactive situations and the ordering is thus dependent on the actual interactive situation being performed.

Modelling Actability Requirements

The main input to interaction analysis is the action diagrams created during BPM (see example in appendix). From the action diagrams interactive situation proposals (ISP) are derived. The ISPs' are then analysed at two different levels of abstraction. First, the atomic actions of each ISP are identified and analysed according to the state model presented above and documented with interaction tables (described below). Second, possible sequence restrictions among the different atomic actions are analysed and documented with the UML (1997) version of Harel statecharts (Harel, 1987).

Interaction tables

The primary notation used to model actability requirements is the interaction table (ITable), which builds explicitly on the above presented state model of user interaction. An ITable is a table with three rows and three columns. The leftmost column is used for the user actions, the middle column for the state of the current interactive document(s)[3], which represents the state of the IS, and the rightmost column is used for the IS actions. Since we are not dealing with sequence restrictions at the moment one ITable is used for each atomic action.

3	User action	Document	IS action
S_0		1. Order form 2. Customer details	
S_1	Associate customer to order {By drag 'n drop $2\rightarrow 1$}	Order form Order with associated customer.	1. Associate customer to current order. 2. Transfer information about current customer to current order (from customer file).
S_2	Acknowledge customer information: 1. Correct customer information. 2. Incorrect customer information. (Must be changed ($\rightarrow$ 2))	Order form Order with associated customer information visible.	

Fig. 3. Example ITable for an IS to support a travel agency

Figure 3 shows the ITable for atomic action 3 within the interactive situation "Create customer order" from the same IS as in the example action diagrams in appendix. From the example in figure 3 we see how s_0 specifies what documents are to be visible in order to perform atomic action 3. Cell (2,1) specifies what action the ITable is concerned with, i.e. associating a customer to an order by the use of "drag 'n drop" from the document "Customer details" to the document "Order form". Cell (2,2) specifies the state of the system after atomic action 3 has been performed, i.e. s_1, the customer is now associated to the 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 (in this case two consecutive IS actions). Cell (3,2) shows the

[3] Interactive documents, i.e. the windows and dialogs of the GUI, are thought of as *action medium* in the communication between user and IS. One can think of them as the *arena* for an interactive language game.

response from the IS, i.e. s_2. Finally, cell (3,1) shows the interpretation act to be performed by the user. In this case the interpretation should yield either that the customer information is correct or that the information needs to bee updated, which leads to the performance of atomic action 2, i.e. updating of customer information. Note that the restrictions concerning the interpretation action a_3 states that the result of the system action(s) should only be possible to interpret as one of the stated alternatives. This is admittedly a quite hard requirement imposed on the IS designers but highly relevant to actability.

In most cases the number of IS actions that corresponds to an atomic action is one or perhaps two in strict sequence. Sometimes it might be several IS actions involved and a need to describe how these relates to each other. In such cases statecharts are used to model the sequence restrictions and thereby sub-states of s_1.

The granularity by which the documents are represented in ITables is dependent of how far the analysis has proceeded. At first, the documents are referred to by textual references (as in figure 3). When the analysis proceeds the layouts of the documents are getting more and more explicit. This evolution can (and should) be shown in the ITables. This is done as in figure 4, which shows the atomic action number 2 from the same IS as in figure 3.

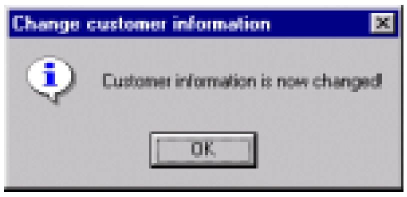
2	User action	Document	IS action
S_1	Correct invalid customer information	Change customer information Old information New information [Register change]	Update customer information.
S_2	State fact: Correct customer information is registered.	Change customer information (acknowledged)  Customer information is now changed! 	{RT ≤ 1s}

Fig. 4. ITable with detailed document layout

It is sometimes unnecessary to show the initial state s_0 within an ITable, and hence the first row can be omitted. E.g. when there is no doubt in which contexts the document is used, such as in figure 4 where the “Change customer information”-document only can be shown when a customer is already selected. Such restrictions are modelled with *navigation charts* as discussed in a subsequent section.

The language used within ITables is natural language such as English or Swedish. We do however use some conventions that semi-formalise the language use. In the “user action”-column the intended action is underlined. Underlining is also used for the documents in the “document”-column in order to distinguish between the document and its informational (propositional) content. We use square brackets to note that something should be a clickable[4] item in a document, e.g. a button with the caption “Register change” will probably be used to represent the “[Register change]” of figure 4. Further, curly brackets are used to state preconditions to distinguish between actions, e.g. “{By drag ‘n drop 2→1}” in figure 3. Curly brack-

[4] By “clickable item” we refer to the item used in order to cause the state change, i.e. to perform the action from the IS point of view.

ets are also used to specify restrictions on IS actions such as response time (RT), e.g. “{RT ≤ 1s}” in figure 4.

Sequence restrictions

Sequence restrictions among atomic actions are usually well documented in the action diagrams. As a matter of fact, such restrictions should be avoided as far as possible in order to keep user interfaces flexible. Sometimes there however is a need to explicitly model sequence restrictions in order to reach higher precision in the interaction model. For that purpose the UML (1997) version of Harel statecharts (Harel, 1987) is used. One reason for using statecharts is that a method design rationale for VIBA is to make use of existing notations and formalisms that are possible to integrate with the proposed action view of information systems. Another reason for statecharts is that they, in a natural way, models the state of the interactive situation which in some sense is equivalent to the state of the current documents being used.

Even in cases where statecharts are not suitable it might still be important to document some restrictions regarding the sequence of atomic actions. If so, one can apply information about atomic actions that are needed and those not permitted as pre- and post-conditions to each ITable.

Navigational actions and relationships between document

As seen in the examples of figures 3 and 4 there exists a relation between the documents “Order form” and “Change customer information”. The actions performed to navigate between documents are generally not considered to be communicative actions but *navigational actions*. Such actions are directed towards how to communicate rather than communication per se. It would be possible to model such kind of actions with ITables but the number of ITables would increase combinatorically due to the many possible navigation paths between documents. Because of this, a special kind of statecharts, called *navigation charts*[5], is used to model navigational actions and navigational relations between documents. Each state in a navigation chart represents a document and each transition represents a navigational action.

Within a state symbol of a statechart it is possible to associate actions that is to be performed upon entry and/or exit of the state (Booch, 1994). We use this property of statecharts to describe IS actions that are performed automatically when a document is entered or exited. It might for example be the case that some validation of the document content must be performed by the IS before the user can switch to another document or that some initialisation is to be performed on entrance.

During interaction analysis the navigation charts are used to model each interactive situation in turn. The revealed information about document relations is then used during document analysis where the whole IS is considered and the fragmented document model of interaction analysis is consolidated to system global level.

The requirements document

There are primarily two kinds of requirements, with corresponding requirements specifications, that has to be taken into account when building an IS. These are usually referred to as *initial requirements* and *detailed requirements* (e.g. Sommerville, 1996). The initial requirements deals with the fuzzy ideas in the beginning of IS development. Their specification is

5 An interpretation of the Swedish term “navigeringsdiagram” used by Mathiassen et al (1998).

thus positioned in time somewhere between CA and VIBA in SIMM terminology. The initial requirements are important to capture since it is often based on them the contract between the software organisation and the customer is committed. This paper deals primarily with the *detailed requirements* that are elicited and analysed during IS development and enhancement. Our discussion about the requirements document, which we refer to as 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 etc. (Sommerville, 1996). As mentioned above, actability requirements concern both functional and non-functional requirements. We therefore chose to divide actability requirements into three categories; 1) functional requirements, 2) usability requirements and 3) other non-functional requirements. The usability category includes requirements regarding document layouts, navigation and interpretability. The others category includes all other non-functional requirements that relates to actability. It is important to note that there exists other usability requirements that does fall outside the scope of analytical analysis of actability (and thereby this paper) but which are handled within other parts of VIBA. There are also other non-functional requirements, e.g. regarding information content, as well as functional, e.g. as the result of operating environment constraints, which are added to the SRS at other points in time. But once again, that is done within other VIBA-areas.

In some cases the produced models will serve as sufficient specification of requirements, but there are good reasons for extracting requirements from the models and put 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 interaction analysis relates to the SRS, where the requirements are to be found and how to achieve requirements traceability.

Functional requirements

There is a direct correspondence between the IS actions found in the ITables and navigation charts 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 covers all IS actions. Some constraints on functional requirements can also be found as restrictions in the IS column of the Itables, but most of such restrictions have to do with usability requirements.

Usability requirements

Requirements regarding user interface design are found in the ITables (layout) and the navigation charts (navigation). One aim of ITables is to capture requirements regarding understandability. The restrictions on interpretation (a_3) are sources for that kind of requirements. Another source is the stated captions on clickable items. These requirements aim at making the IS more action transparent and intuitive, i.e. the IS should behave just as the user assume. Other usability restrictions imposed on the IS (curly brackets in the third column) are another source for usability requirements, e.g. response times.

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 other parts of VIBA, e.g.

restrictions of data representation, which ought to be found during conceptual modelling. Since VIBA is used with a bottom up approach the actability oriented analysis often serves as rationale for such restrictions. It is therefore important to crosscheck the different models in order to deduce requirements from their interconnections and to avoid ambiguity and inconsistency.

Traceability of requirements

Requirements traceability (RT) is important for system maintenance and evolution. 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 e.g. predict software change and software change costs when re-designing or evolving a business. The other viewpoint concerns interrelationships between individual software requirements. The second view of traceability is important to understand how changes propagate through a software system. The views are not orthogonal and both should be taken into account during RE. The first kind of RT is achieved within VIBA by use of the fact that requirements are derived from the ITables which in turn are derived from action diagrams that models the business. To make use of this fact the requirements of the SRS must contain references to the model elements to which they relate. There are existing approaches to the other kind 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 further in order to make explicit use of the business oriented action approach of VIBA.

4. Related Work

Our approach to IS development and RE is based on what we call a language action perspective on information systems. That perspective is part of the so-called language action perspective on communication modelling (LAP), which has gained more and more interest during the last years (Dignum et al, 1996; Dignum & Dietz, 1997; Goldkuhl et al, 1998). Our approach thus tries to reconcile LAP and 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.

The Language Action Perspective

The best known approaches within LAP are probably Action Workflow (Denning & Medina-Mora, 1995), DEMO (Dietz, 1994) and SAMPO (Auramäki et al, 1998). In these approaches business processes (with support of information systems) are described as communicative action of various kinds. The Action Workflow and DEMO approaches use a predefined set of communicative actions structured in specified way to describe business processes. The action workflow loop is rather well known with its four predefined action stages (preparation, negotiation, performance, and acceptance). The VIBA approach uses a six-stage model for business interaction – the BAT model (Goldkuhl, 1996; Goldkuhl, 1998), that has some resemblances with Action Workflow. The description of business processes (in the SIMM approach) follows this six-stage model, but uses it in a much more free 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. Goldkuhl (1996) makes a comparison between Action Workflow and SIMM describing these issues more fully and Reijswoud & Lind (1998) make a comparison between SIMM and DEMO.

These other L/A approaches do not (according to our knowledge) go into detail of requirements definition. One important feature of VIBA is the bridging of the gap between 1) business process modelling and 2) detailed modelling of documents and interaction within the information system. We do not recognise this ambition in these other approaches. Their scope thus seems to be narrower. The important notion of IS actability does not seem to be used in these other approaches.

Requirements Engineering

Our notion of the RE process as a continuing process that span the lifecycles of possibly many single information systems is a notion much like 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 e.g. business action theory (Goldkuhl, 1998) in order to perceive the business and social impacts on RE and change management.

Our notion of interactive situations is quite similar to that of use cases (Jacobson et al, 1992). The main difference is that we take communicative action as a starting point rather than the behavioural view of event state systems that is used as the generic abstraction in scenario based approaches.

We use the term "usability" to refer to a subset of actability. This is not to be mistaken for the use of the term within e.g. usability engineering (Nielsen, 1993) where it is also used for empirical testing of different (usability) metrics, such as number of accepted mistakes by a user and the learnability of the IS. Such measurements are however important for systems development and usability testing might be used in parallel with VIBA.

Goal driven RE (e.g. Yu & Mylopoulos, 1998) is another area that relates much to our work. We believe that communicative action is always based on some intentions and intention can be viewed as a kind of goal (Ågerfalk & Åhlgren, 1998). Goals are e.g. used in VIBA when analysing the existing business. The goals of current actions and action structure can be used to transfer tacit knowledge and good practice when designing new businesses.

5. Conclusions and future work

In this paper we have presented intermediate results of the ongoing research work of creating a requirements engineering method, named VIBA/SIMM, based on a language action view of information systems. We argue that information systems are not to be regarded as containers of fact, but rather as vehicles for communication among people and organisations. The approach thus takes communicative action as the main starting point for analysis of business and supporting information systems as integrated parts. To "perceive information systems as action" have impacts both on how to perform the development process and on the designed products, i.e. the information systems. One aim of our research is to build on existing RE knowledge, modelling formalisms and good practice. Such existing knowledge and practices are integrated in our method when possible; i.e. when they are in alignment with or can be reinterpreted according to this new way of understanding information systems.

The specific part of VIBA that has been focused in this paper is interaction analysis and thus IS actability. We have introduced a notion of (interactive communicative) *atomic action* and a *state model* of user interaction used to analyse *interactive situations*, in the context of the business of which the planed information system is a crucial part. We have also shown how the software requirements specification can be constructed based on the analysis in a bottom up fashion.

The proposed approach seems to lend itself well for managing of requirements by use of a systematic handling of requirements traceability. Change management of requirements is, as mentioned above, an area which we plan to study further. We believe that our action approach and the use of business action theory (Goldkuhl, 1996) ought to be appropriate to understand such activities.

We have not yet done any comprehensive empirical testing of VIBA. What we have done so far is the specification of a production planing system in collaboration with industry. From that work we have learned a lot that is now incorporated in the method. Though, more empirical work is still needed.

6. Acknowledgements

We would like to thank Stefan Cronholm (Linköping University) and Owen Eriksson (Dalarna University) who has contributed extensively to the evolution of VIBA. We would also like to mention Örebro Mekaniska Verkstad, the mechanical engineering firm that we use for empirical testing, we really appreciate your patience.

References

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.

Booch G (1994). *Object-oriented analysis and design with applications*. 2nd ed. The Benjamin/Cummings Publishing Company, Inc. Redwood City, California.

McConnel S (1996). *Rapid Development – Taming Wild Software Schedules*. Microsoft Press.

Denning P J, Medina-Mora R (1995). *Completing the loops*. Interfaces.

Dietz J (1994). *Business modelling for business redesign.* In Proc. 27th Annual Hawaii Intl. Conference on Systems Science, IEEE.

Dignum F, Dietz J (1997). *Communication modelling - the Language Action Perspective*. Proc. 2nd International workshop on Communication modelling, Computer science Reports, Eindhoven University of Tecnology.

Dignum F, Dietz J, Verharen E, Weigand H (1996). *Communication modelling – the Language Action Perspective*. Proc. 1st International workshop on Communication modelling, Electronic Workshops in Computing, Springer Verlag.

Dubois E, Pohl K, Opdahl A L (1998). *Proc. 4th Intl. Workshop on Requirements Engineering: Foundations of Software Quality*. Presses universitaires de Namur, Belgium.

Gamma E, Helm R, Johnson R, Vlissides J (1995). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley Publishing Company Inc.

Giddens A (1984). *The constitution of society. Outline of the theory of structuration.* Polity Press, Cambridge.

Goldkuhl G, Lind M, Seigerroth U (1998). *Proc. 3rd Intl. Workshop on the Language Action Perspective on Communication Modelling*. Jönköping Intl. Business School.

Goldkuhl G, Lyytinen K (1982). *A language action view of information systems*. In Ginzberg, Ross (Eds.) Proc. 3rd Intl. Conference on informations systems, Ann Arbor.

Goldkuhl G, Röstlinger A (1993). *Joint elicitation of problems: An important aspect of change analysis*. In Avison D et al (1993): Proc. Human, organizational and social dimensions of Information systems development, North-Holland/IFIP w.g. 8.2.

Goldkuhl G, Ågerfalk P J (1998). *Action Within Information Systems: Outline of a Requirements Engineering Method.* In Dubois et al (1998).

Goldkuhl G (1996). *Generic business frameworks and action modelling*. In Dignum et al (1996).

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 Intl. Business School.

Jacobson I, Cristerson M, Jonsson P, Övergaard G (1992). *Object-oriented software engineering – A use case driven approach*. ACM-press.

Harel D (1987). *Statecharts: A Visual Formalism for Complex Systems*. In Science of Computer Programming 8 (1987) pages 231-274.

Lam W, Shankararaman V, Jones S (1998). *Managing Requirements Change: A set of Good Practices*. In Dubois et al (1998).

Lam W (1998). *Change Analysis and Management in a Reuse-Oriented Software Development Setting*. In Pernici B, Thanos C. Proc. 10th Intl. Conference on Advanced Information Systems Engineering (CaiSE'98), pages 219-236. Springer-Verlag, Berlin, Heidelberg, 1998.

Maiden N A M, Cisse M, Manuel D (1998). *CREWS Validation Frames: Patterns for Validating Systems Requirements*. In Dubois et al (1998).

Mathiassen L, Munk-Madsen A, Nielsen P A, Stage J (1998). *Objektorienterad analys och design*. (In Swedish, Object-oriented analysis and design). Studentlitteratur, Lund.

Nielsen J (1993). *Usability engineering*. Academic press.

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 965-979.

Reijswoud V E van, Lind M (1998). *Comparing two business modelling approaches in the language action perspective*. In Goldkuhl et al (1998).

Sawyer P, Sommerville I, Viller S (1998). *Improving the Requirements Process*. In Dubois et al (1998).

Searle J R (1969). *Speech acts. An essay in the philosophy of language*. Cambridge University Press, London.

Sommerville I (1996). *Software engineering*. 5th ed. Addison-Wesley Publishing Company Inc.

UML (1997). *UML Notation Guide v 1.1*. Rational Software, Inc.

Weiser, M (1984). *Program slicing*. IEEE Transactions on Software Engineering, 10(4), pages 352-357.

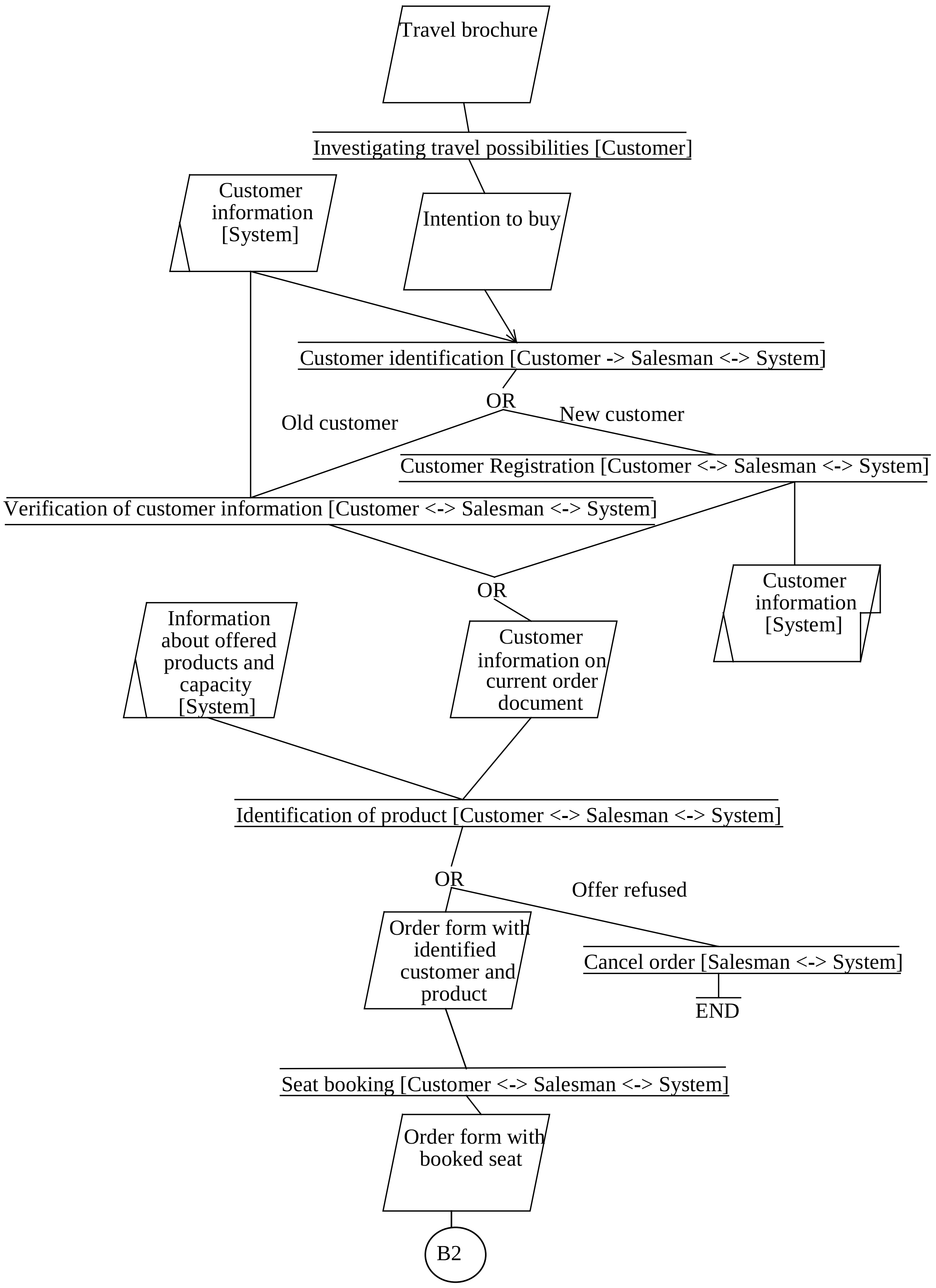
Winograd T, Flores F (1986). *Understanding computers and cognition: A new foundation for design*. Ablex, Norwood.

Yu E, Mylopoulos J (1998). *Why Goal-Oriented Requirements Engineering*. In Dubois et al (1998).

Ågerfalk P J, Åhlgren K (1999). *Modelling the Rationale of Methods*. In Proc. 1999 IRMA Intl. Conference: Managing Information Technology Resources in Organizations in the Next Millennium. Idea Group Publishing, Hershey, PA, USA.

Series	**ACTION DIAGRAM**		**Appendix A**
ACRE'98			
Creator	**Date**	**Version**	**Ref. code**
Ågerfalk, Goldkuhl	24.09.98	2	B1

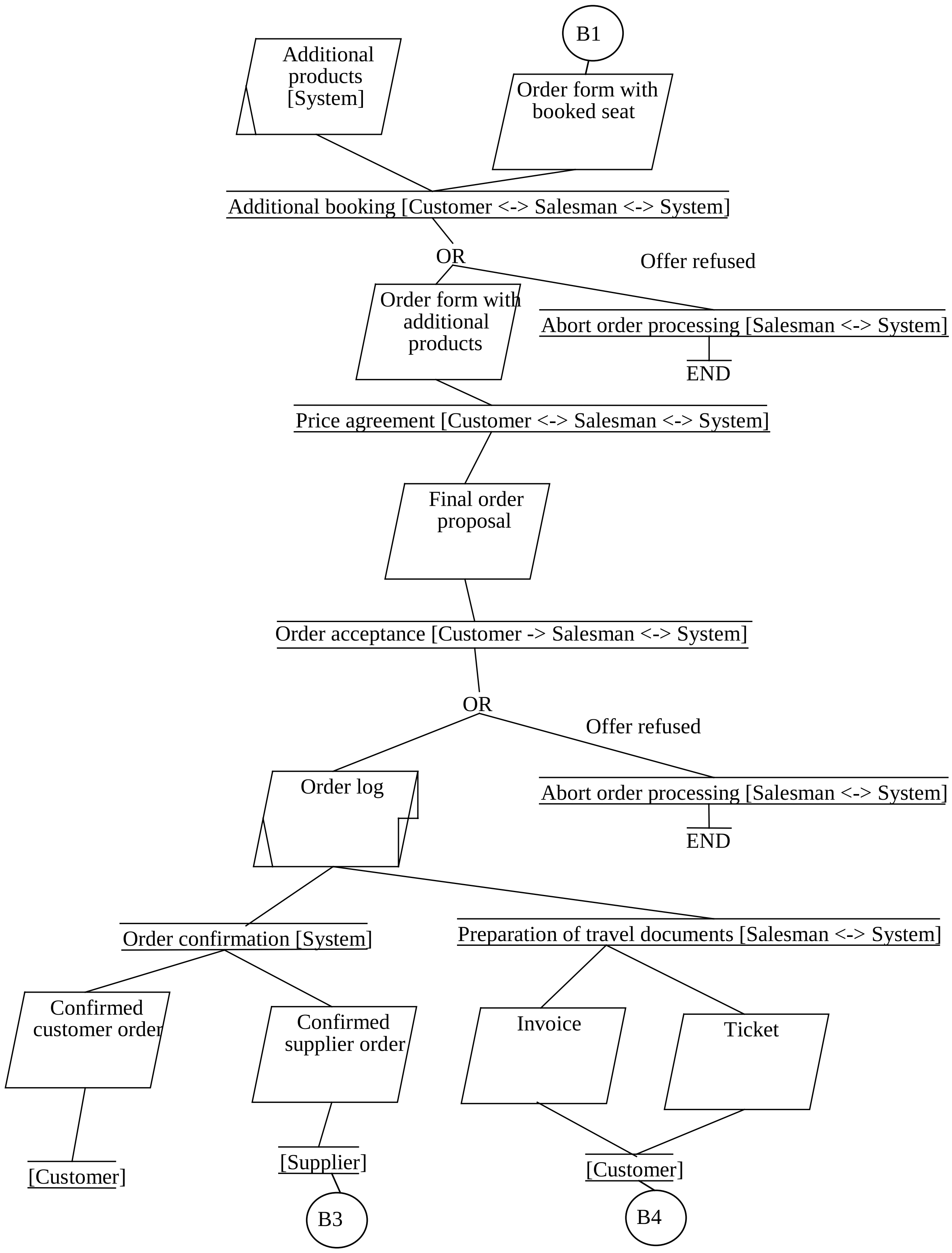
Concerns: Order phase 1



Travel brochure
Investigating travel possibilities [Customer]
Customer information [System]
Intention to buy
Customer identification [Customer -> Salesman <-> System]
OR
Old customer
New customer
Customer Registration [Customer <-> Salesman <-> System]
Verification of customer information [Customer <-> Salesman <-> System]
OR
Customer information [System]
Information about offered products and capacity [System]
Customer information on current order document
Identification of product [Customer <-> Salesman <-> System]
OR
Offer refused
Order form with identified customer and product
Cancel order [Salesman <-> System]
END
Seat booking [Customer <-> Salesman <-> System]
Order form with booked seat
B2

Series	ACTION DIAGRAM		**Appendix A**
ACRE'98			
Creator	**Date**	**Version**	**Ref. code**
Ågerfalk, Goldkuhl	24.09.98	2	B2

Concerns: Order phase 1



B1
Additional products [System]
Order form with booked seat
Additional booking [Customer <-> Salesman <-> System]
OR
Offer refused
Order form with additional products
Abort order processing [Salesman <-> System]
END
Price agreement [Customer <-> Salesman <-> System]
Final order proposal
Order acceptance [Customer -> Salesman <-> System]
OR
Offer refused
Order log
Abort order processing [Salesman <-> System]
END
Order confirmation [System]
Preparation of travel documents [Salesman <-> System]
Confirmed customer order
Confirmed supplier order
Invoice
Ticket
[Customer]
[Supplier]
[Customer]
B3
B4