Modeling Concepts for the Integrated Reasoning about Complex Systems

Michael Cebulla,
Technische Universität Berlin, Fakultät für Elektrotechnik und Informatik,
Institut für Softwaretechnik und theoretische Informatik,
Franklinstr. 28/29, 10587 Berlin
e-mail:mce@cs.tu-berlin.de

Abstract

In this paper we focus on formal concepts for an integrated modeling of complex systems. We take our examples from the field of sociotechnical systems where we state a great need for model-based reasoning regarding questions of system safety and the definition of long-term management policies. We lay our special focus on systemic capabilities related to context adaptive behavior with special consideration of cognitive parameters. We are convinced that the systemic ability of context adaptivity is insufficiently understood and is of increasing importance in sociotechnical systems as well as in advanced pervasive applications. We think that a model-based approach is well-suited for the improvement of context awareness in complex systems.

Keywords: Requirements Engineering and Specification, Frameworks, Software Architecture.

1 Introduction

Modern systems engineering is more and more confronted with a new quality of contextual embedding. For the specification of systemic behavior a growing number of environmental parameters have to be taken into account. We conceive these parameters as systemic *aspects* and use *ontologies* to achieve a modular way to manage the related knowledge.

Sociotechnical systems as well as advanced applications like ubiquitous services have to reside in complex contexts and to deal with unforeseen situations. In addition they are strongly interwoven with numerous parameters determined by the external environment. One example for these context dependencies is the new impact that human factors have in these kinds of systems. Future ubiquitous services in combination with multimodal interfaces will result in a dependency between user and service whose complexity is unknown up to now. Advanced services will have to possess the capability of *context awareness* in order to adapt their behavior to unexpected situations. Especially they will have to be able to make assumptions concerning the identity of users, their goals and their knowledge.

This prominent role of context adaptivity is a well known fact in sociotechnical systems. Hybrid teams of humans and devices have to adapt to adverse environmental conditions and have to select those strategies which are optimal in the given situation. Although these systems are of a considerable robustness there is an intensive need for a better understanding of these capabilities which are hard to grasp by formal methods. Due to the complexity of these systems their analysis and understanding is very difficult. But on the other hand for various reasons (safety, efficiency, organizational learning, change management) there is a strong need for a greater transparency of the related processes.

In our approach we suggest a visual notation for the modeling of complex sociotechnical systems as well as context-adaptive systems in general. Starting from software architecture and the experiences of *requirements engineering* [11] we provide concepts which are well suited for the description of specific context adaptive features. Especially the aspects of human cognition and organizational relations are traditionally hard to grasp by formal notations.

We claim that a visual modeling notation significantly increases system transparency, support interdisciplinary system analysis, and is well-suited to support measurements of further education. In addition we argue that domain-specific high level abstractions in general are a strong medium for the design of context aware applications. In the future we plan to provide automated tools for the support of system management.

After some general considerations (section 2) we describe the critical features of sociotechnical systems in section 3. After presenting the basics of our approach (section 4) we introduce some extensions for a cognitive process modeling which is oriented towards the agents capabilities (sections 5 and 6). After demonstrating how to define specific connectors (section 7) we discuss context-adaptive behavior and give some examples for our treatment (section 8).

Proceedings 18th European Simulation Multiconference
Graham Horton (c) SCS Europe, 2004
ISBN 3-936150-35-4 (book) / ISBN 3-936150-36-2 (CD)

2 Complexity and Safety

Complex sociotechnical systems have evolved to control high risk technologies by teams of highly qualified specialists. These systems can be defined as *complex* safety-critical systems where *teams* of human operators cooperate with *ensembles* of technical units and devices. Usually, the resulting processes are significantly more complex than in systems consisting solely of technological components because they have to be context aware to a high degree. Examples for this kind of systems are atomic power plants, medical operation theaters and air traffic control.

This new class of system complexity and its related risks have established new requirements for system design and system safety. This is documented by the sad history of catastrophes from Three Miles Island (1979) to Überlingen (2002). The analysis of such complex systems has proven too multi-faceted for the traditional single-disciplinary approach.

A model-based interdisciplinary system analysis is a promising strategy against what Leveson calls *intellectual unmanageability* of high risk systems [6]. The increasing complexity and tight coupling in contemporary high risk systems make a safe and efficient management difficult if not impossible. The main source of failure in complex systems is not human error or an erroneous component, but the *complex interactions* between components which is not understood to a sufficient degree [12]. To increase the level of understanding we choose an model-based approach which is open for results of interdisciplinary research.

In addition we argue that the results of our research and high-level concepts for the various aspects of contextual modeling can be counted as a contribution to an integrated design of very complex systems (i.e. pervasive context-aware services).

3 Sociotechnic Challenges

Of course, the methods of system modeling are not new. Especially in *software engineering* concepts and methods were synthesized to handle the challenges of complexity in development processes. We also heavily rely on the results of systems engineering [14]. The concepts and methods from requirements engineering provide the basic means to manage informal and semiformal knowledge about the target system.

We observed some specific features of sociotechnical systems which can be conceived as challenges for traditional modeling concepts. We claim that the traditional modeling concepts and formal methods as known from software engineering have to be adapted and extended for the specific properties of sociotechnical systems. Speaking generally, just the merits of formal specification concepts as exactness and well-definedness sometimes prove as shortcomings in the context of sociotechnical systems. It is a generic feature of these processes that they have to deal with vague data, uncertainty and incomplete specifications. Modeling concepts have to adapt to these specific vagueness which can be conceived as important system quality. Paradoxically speaking, too much exactness would lead to less adequate or even wrong specifications.

3.1 Uncertainty

Sociotechnical systems tend to reduce the load of information processing by using vague concepts. So human experts normally don't use exact mathematical numbers but vague expressions from natural languages. We claim that the resulting vagueness is an important precondition for the systems' robustness and safety since it makes their specification less sensitive to contextual changes.

In addition, human actors have to deal with incomplete specifications. In many situation relevant information is not accessible for the human actors. Due to situational time pressure they have to make uncertain decisions based on incomplete information. In our approach we use fuzzy sets and fuzzy logic to specify uncertain information and vague relations [5].

3.2 Adaptive Behavior

One important feature of sociotechnical systems is their *structural dynamism.* The internal structure of systems like the medical operation theatre can be rapidly changing from one phase of the process to another according to environmental changes. For the description of this *structural dynamism* powerful concepts from dynamic architectures [10] are necessary. We handle this problem by introducing *transformation rules* (cf. section 8).

By specifying transformation rules we define the possibilities of configurations to evolve under certain context conditions. Hence, transformation rules describe the adaptive behavior of complex sociotechnical systems in describing the way these systems react to environmental changes by structural mutation.

Sociotechnical systems like the medical operation theatre have distinctive qualities regarding their adaptive capabilities. Thus, their ability to recover in the face of adverse environmental conditions or unexpected events is clearly larger compared to the behavior of traditional component-based systems. Regarding this adaptivity a deeper understanding of sociotechnical processes can contribute to the robustness and flexibility of advanced software-based systems.

3.3 Subjective Evidence

Domain experts tend to disagree about the facts in their field. Sometimes they are not completely sure about the state of affairs. We use fuzzy concepts for the representation of different degrees of subjective evidence or relevance. This enables us to distinguish between different degrees of cognitive adaptation to a given context (section 6).

4 Basic Concepts

We provide concepts for the structural description of complex systems as shown in Figure 1. We provide a visual notation for each of these concepts which allows for an expressive scenario-based modeling. In addition we define a notation for reasoning about systemic features. Our goal is to provide modeling concepts which are meaningful in terms of domain semantics and well-suited for formal reasoning.

For the sake of formal reasoning we base our approach on methods from ontological modeling [7, 3]. Thus, in this section we define a basic ontology for the structural description of complex systems. We are leaving aside all issues which are related to behavioral or temporal specification.

4.1 Agents

We conceive agents as the basic computational elements of every complex systems. Agents interact which each other in order to process requests from their environment.

Definition 1 (Agents) *An agent $\langle name, I, S\rangle$ consists of a symbolic name (denoting his identity), a set of interfaces I and a state S.*

In a description logic example for an agent we first express the hierarchical place of the concept. Then in addition we describe its internal structure as a conjunction of interfaces and features.

```
Anesthesist ⊑
  (AND Agent
       (AND IA1 IA2)
       (AND (≤ 3 acc.⊥)
            (≤ 1 vis.⊥)
            (expertise {low, medium, high})
            (motivation {low, medium, high})))
```

As an example for the visual representation of an agent cf. Figure 5.

4.2 Interfaces

Agents supply their services to their environment by publishing *interfaces.* Using interfaces agents can be assigned to certain tasks or engage in relations to each other. We describe interfaces as algebraic signatures.

Definition 2 (Agent Signature) *An agent signature is a quadruple $\langle \Sigma, I, O, A\rangle$ where Σ is a data signature $\langle S, Op\rangle$ in the algebraic sense, O is the set of output attributes (the agent variables), I is the set of external attributes that the agent has to read from it's environment and A is a set of action names.*

On the semantic level an interface is represented by a conjunction of features.

```
IA1 ⊑ (AND (acc {highPitch, lowPitch})
           (lman {take, give}))
```

In a graph based notation we denote interfaces as *little squares.*

4.3 Linguistic Variables

In this paper we use a simplified version of linguistic variables [5]. Linguistic variables are the essential carriers of data in integrated systems engineering. According to the terminology of description logic they can be conceived as *roles* which are named relations between an agent and a domain.

Thus, as we will demonstrate in examples, it is possible to model the acoustic sense of a human actor by a feature called `acc` which defines a relation between an agent and a domain. We frequently use enumeration types as domains for features.

4.4 Connectors

In complex systems the understanding of agent interactions is a central issue. For the articulation of a system's interaction structure we adopt the concept of *connector* from software architecture [16, 9].

In the context of an ontological approach a connector has to be conceived as a special concept related to interaction.

Definition 3 (Connector) *A connector $\langle name, R\rangle$ consists of a name and a set of roles.*

Similar to agents connectors publish interfaces. They do so to express their requirements concerning services which have to be supplied by the specified agents. This is why we sometimes speak of *roles* when referring to a connectors interfaces.

In our visual notation we use ovals to represent connectors (cf. Figure 8). Concerning a notation which provides better reasoning support connectors are very similar to agents.

4.5 Configurations

Agents constitute *configurations* by interacting using connectors. Connectors express their requirements by publishing interfaces while agents describe their capabilities by publishing interfaces. When a specific selection of an agent's capabilities satisfies a part of the connectors requirements we say that the concerned interfaces are *matching.*

The typical usage of this concept of matching interfaces can be illustrated by the following example. Since we use connectors for the representation of systemic tasks, interfaces are frequently used for the specification of requirements for the fulfillment of a given task. For each task interface an agent has to be found which supports this interface. The process of this finding usually is called *negotiation* [2]. For the sake of reasoning about matching interfaces we conceive interfaces as concepts in the sense of ontological modeling. We claim that two interfaces match if the concept representing the required interface *subsumes* the provided interface. In terms of model-theoretic semantics we claim that each instance of the provided interface has to be an instance of the required interface. To reason about subsumption *tableau algorithms* can be used which are common in description logics [3].

In our graphical notation we connect matching interfaces using a *link,* which is expressed by a solid line (cf. Figure 8).

4.6 Conceptual Framework

In Figure 1 we show our conceptual framework in a UML-like visual notation. We use the term *conceptual framework* in order to avoid the term *meta-meta-model.*

As we show in Figure 1 we conceive every concept as a subconcept of *Entity* which we express by hollow arrows. Some of the concepts are related by *has*-relations (aggregations) which are expressed by solid arrows. Aggregations may be annotated by cardinality constraints.

5 Availability

Frequently, it is not enough to know that an agent can provide a certain service (as represented by an interface). Moreover, it is necessary to check if the required services are available in the given situation. Consequently we need concepts to reason about an agent's temporary workload. In order to do this we have to reason about the *actual state* of an agent.

Conceiving the current state of a given agent as a conceptual model we state that the required services which are represented as an additional model are available iff the combination of that model with the new concept is *consistent.* Of course, with respect to the availability of services the consistency of cardinality constraints (as supported by SHIQ-DL) are of special importance. Again, tableau algorithms can be used to check consistency.

6 Mental Models

In our approach we provide a *cognitive perspective* which takes into account the actors' subjective motivations and their influence on the global system's behavior. As we already argued we conceive human factors and cognitive representations as essential contextual features which are crucial for the behavior of sociotechnic systems as well as for advanced pervasive applications. For this sake we use the concept of *mental models,* which describes an actor's internal representation of his environment.

In our approach a mental model consists of three fuzzy sets (adopting ideas from agent-oriented modeling [17]):

- The agent's intentions ($\mathcal{I}$): usually a decision aims at a certain goal. That means that the agent tries to achieve a certain system state by selecting between alternative behavioral options.
- The agent's beliefs ($\mathcal{B}$): a decision is highly influenced by the agent's belief concerning the system's actual state and its further behavior.
- The agent's desires (preferences) ($\mathcal{D}$): usually the agent has a subjective preference for a certain behavioral option which may or may not interfere with the real situation.

We claim that for a proper understanding of the system's behavior it is necessary to take these factors into account. These factors may influence human behavior to a high degree and thus determine global system behavior. Moreover subjective factors are subject to a large scale of organizational measures like further education and simulator training.

We represent this internal information using fuzzy sets. This gives us the possibility to represent the *subjective relevance* of a proposition by the membership relation μ.

We notate fuzzy sets by using *calligraphic font.* The function of the three fuzzy sets in figure 2 consists in the mental representation of the relevant contextual features [4].

We use a mental model to reason about the adaptation of an actor in a given situation. We define *adaptation* as a relation between a *Context* and an actor. As relevant features of a context we conceive the situational goal (which has to be a strong member of the actors intentions) , the behavioral alternatives (which have to be reflected in the actor's belief),

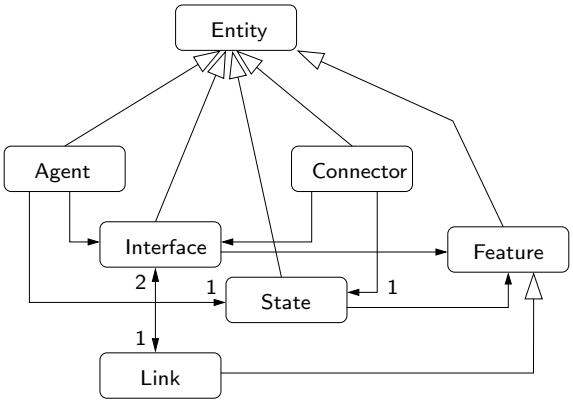

Figure 1: Our Conceptual Framework

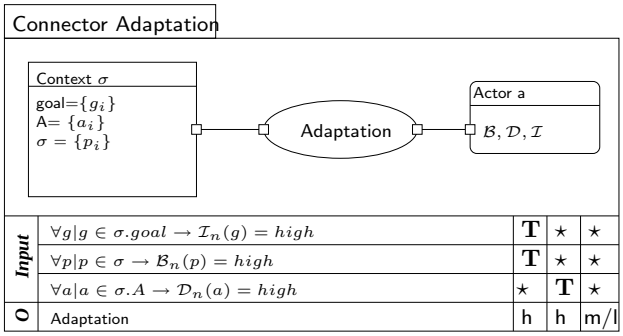

Figure 2: The Concept of Adaptation

and the relevant context conditions (which the actor has to know).

We use fuzzy rules to define a relation between these situational features and actor's mental representations. In general, we claim that an actor's contextual adaptation is good if he has an intensive representation of the relevant features. We use an AND/OR-table to reason about different configurations and their consequences for the quality of adaptation. Thus in the first column of Figure 2 we describe a situation where the actor's adaptation is high because he has strong internal representations of the situational goal and the contextual conditions. In the second row we describe the case that (occasionally) the actor has the adequate preferences. In the rightmost column however we claim that in all other cases the actors adaptation is medium or low (using the $\star$ as wildcard).

For organizational purposes there are some interesting features of contexts. Thus, the cardinality of set A describes the free room of an agent's decision. If it is small this has inhibiting effects on his motivation. If it is too great the agent may be overstrained. Moreover, another important feature of a given context consist in the possibility of differentiation of behavioral alternatives.

7 Sociotechnical Connectors

Agents are well-furnished with capabilities to process specific tasks on their own. In complex distributed systems however they have to coordinate their behavior with other agents. Human agents (and in our approach other agents, too) coordinate their activities with each other by the mutual manipulation of their cognitive states. For this sake they rely on interaction and organizational mechanisms of coordination.

We conceive connectors as behavioral specification about the behavior of agents. An important relation between agents is *Communication* (cf. Figure 3). For the definition of this connector again we have to use formulas about the mental models of agents.

We integrate in our specification some claims about the mental models of the actors (as known from speech act theory [15]).

- Perlocution: a result of a communication process is that the message's recipient believes in the content of the message.
- Illocution: for a communication act it is a necessary condition that it belongs to the actor's goals to induce the receiver's conviction that the proposition m holds.

Other parts of the specification we describe the relation between certain parameters of communication (the expertise of the actors) and some global system properties (safety, performance). For example we claim that only highly trained experts can communicate using signals and reach good safety and performance values (cf. column 1 of figure 3).

We conceive organizational relationships are mechanisms that help to compensate weak situational adaptation. While communication is an obvious way

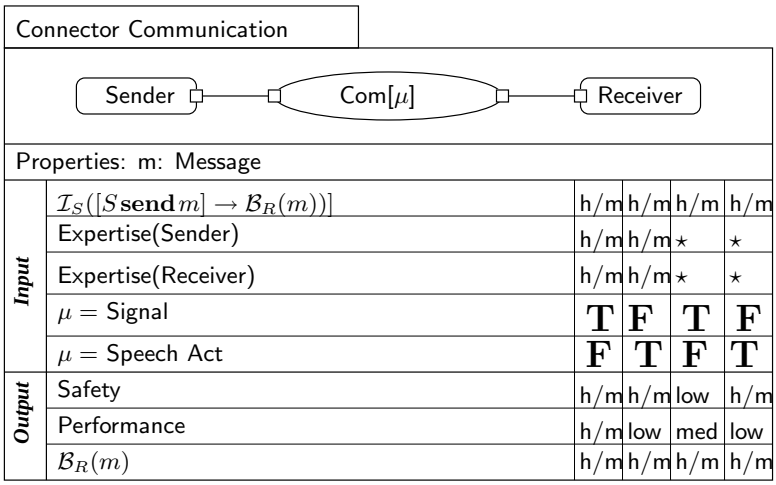

Figure 3: Specification of Connector Communication

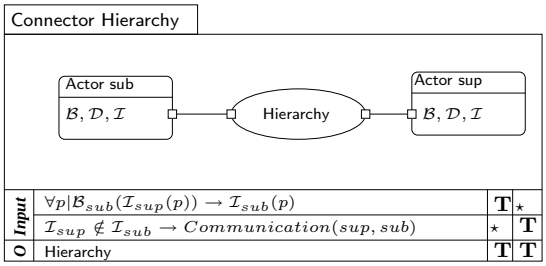

Figure 4: Coordination by hierarchy

for agents to manipulate their mental states there are more subtle mechanisms that have their effect in sociotechnical systems. One of them is hierarchy [8].

As we show in Figure 4 we conceive hierarchy as a mechanisms which asserts that human operators follow the goals of their superiors.

8 Adaptive Behavior

Processes in general as well as sociotechnical processes are structured in *tasks.* Task are logical units of systemic processes. In sociotechnical systems however (as well as in the case of pervasive services) the way a given task is processed may vary in dependence to context conditions. This is a strong adaptive feature of sociotechnical systems: they choose the behavioral alternative which is best regarding the conditions of the actual situations.

For the description of these adaptive features we introduce two levels of abstraction. On the top level we define *abstract tasks* which specify the goals which have to be reached without consideration of context conditions. On the bottom level we define context specific specializations of tasks. In this respect we obviously use ideas taken from *model-driven architecture*[18].

We use configurations to specify the initial state which is necessary for the processing of a task. Configurations describe the agents which are necessary to process a given task and the way these agents have to interact. We use configuration for the description of abstract as well as of context specific tasks.

To describe the adaptivity of complex behavior we specify *transformation rules* which describe which task specializations are selected in which situations. Triggered by context conditions a certain rule is selected and transforms an abstract configuration into a context specific one. The foundations of this approach are described in [10].

8.1 Example: Coordination

The operators in complex sociotechnical system use technical devices or units to fulfill complex tasks or control complex processes. In order to manage this they have to coordinate their activities. From this point of view coordination is conceived as a compensatory system reaction given adverse environmental conditions.

We separate the aspect of coordination from what we call the *abstract task,* which is defined in terms of the system's primary goal. We discriminate two kinds of coordinative tasks:

- *Transfer tasks* define the need to transfer resources or devices to the place where they are needed. In the operational theatre for example the nurses are heavily involved in passing medical instruments.
- *Communication tasks* define the need for some agents to coordinate their activities by using signs or speech. For instance sometimes the anesthesist has to tell the nurse to pass him an instrument.

In our approach we provide a mechanism to map abstract tasks to context specific tasks. In the example of laryngoscopy implicit coordinative tasks are woven into the specification of the abstract task by a specific transformation rule.

Using this kind of rule-based specification allows us to catch the context-sensitivity of systemic procedures. Which type of procedure is used by the actors in a specific situation is highly dependent on the parameters of the context. One example for such parameters is the *expertise* of the actors: highly trained experts tend to coordinate their activities using signals, while novices normally use convention-based speech acts [15].

In figure 5 we show an example for the description of the coordination aspects of a task. The trigger for the firing of this rule is the precondition that the nurse is the owner of the laryngoscope. This is an adverse context condition since in the specification of the task (not shown in the pictures) it is prescribed that the anesthesist should be the owner of the device.

A special rule describes the selection of interaction type according to the expertise of the actors. The type variable μ is instantiated with type *Signal* if the actors' expertise are high. This rule is very similar to configuration rules in feature modeling [1].

8.2 Example: Observing

In the medical operation theatre during an operation the anesthesist has to control the patient's state and the course of anesthesia. Consequently we define as an abstract task that the anesthesist has to observe the patient. For this sake both, the patient and the anesthesist have to be members of the initial configuration. And they have to be connected by an observation connector.

In Figure 6 for example we use this configuration as a the start configuration of a transformation rule. (For the sake of this presentation we are leaving aside the role of interfaces.) In our first rule we state that the abstract task of observation may be mapped to the context-specific configuration on the right hand side of the rule. In some contexts the anesthesist may delegate the responsibility for the patient's observation to the nurse and communicate with her about the patient's actual state. A precondition for the application of this rule is that the nurse has an adequate qualification.

Another typical way to process the abstract task of patient observation consists in the usage of a monitoring device which controls the characteristic parameters of the patient's state. The corresponding rule is frequently applicated in situations where the patient's state is already stabilized. In many cases anesthesists with low or medium expertise level tend to rely on the monitoring display rather than on their own observations.

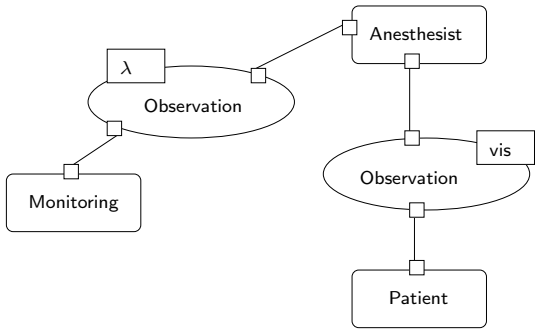

Figure 8: Parametric Configuration

By using simple relations for the sensual capabilities of human agent's we are able to reason about the consequences of task assignments. We are specifying some sensual constraints for human agents.

```
HumanActor ⊑ Agent
HumanActor ⊑ (AND (acc ⊥)
                  (lman ⊥)
                  (vis  ⊥))

Assignment(x) =
   #(Interfaces.linked ∩ Interface ∋ x)

Assignment(acc)≤ 3
Assignment(vis)≤ 1
Assignment(lman)≤ 1
```

We represent senso-motoric capabilities as feature variables of agents. The assignment of such a capability is defined by the cardinality of the set of an agent's linked interfaces where the corresponding feature is contained. We claim that for the visual capability of an agent only unary assignments are possible, while the acoustic sense may be used in more than one assignments.

For our example we can conclude that an acoustic observation leaves more flexibility for an actor than a visual engagement. Consequently the configuration in Figure 8 is only legal with λ = `acc`. The binding of variable λ with `vis` would violate the assignment constraints. Thus an anesthesist is not able to check another patient when he is occupied with the visual observation of the monitoring device. On the other hand when it is sufficient to control the acoustic alarm tones he can move to another room to assist his colleagues.

9 Human Error

Since we have a special interest for human error we reason about errors using fuzzy fault trees. Using fault trees we can reason about different types of human errors using propositions about mental models.

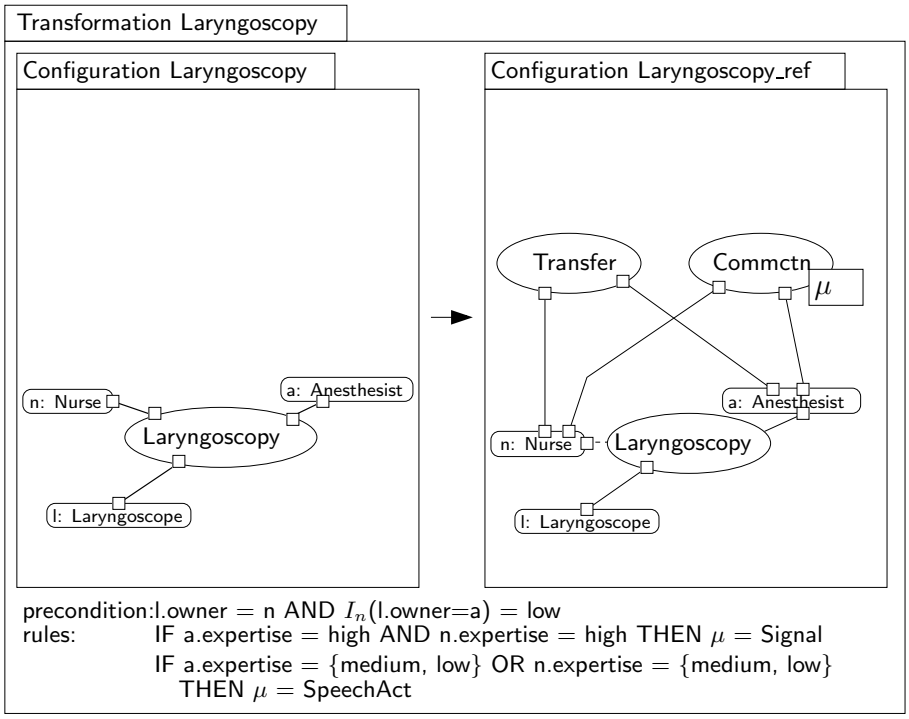

Figure 5: Coordination aspects of Laryngoscopy

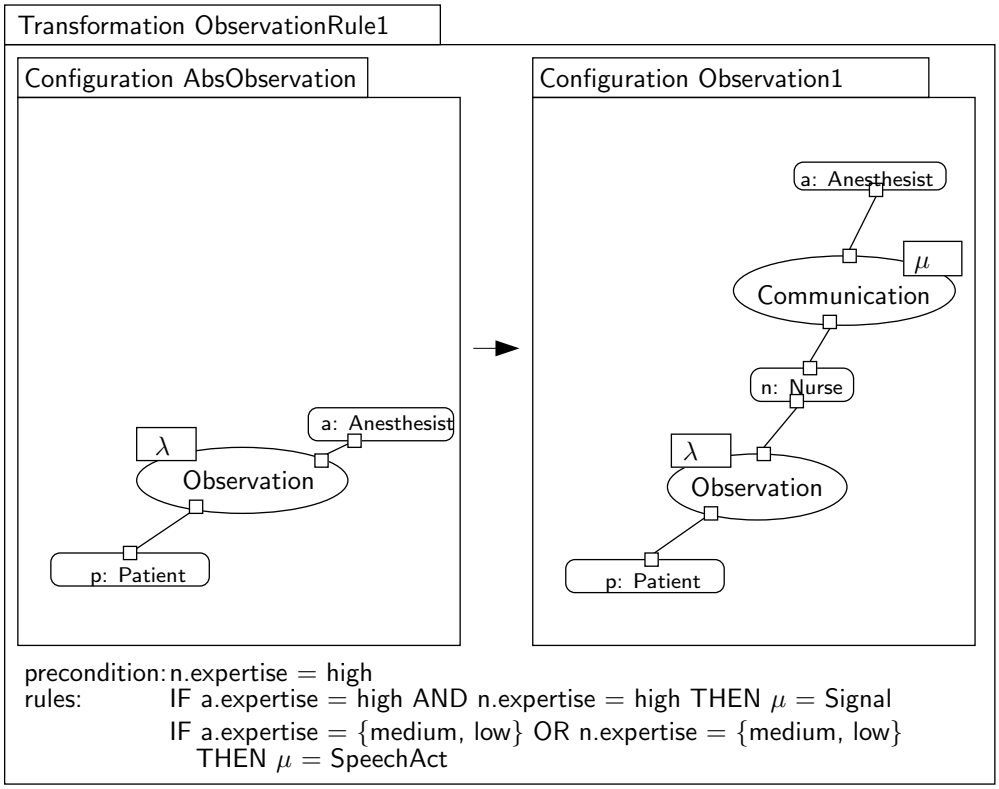

Figure 6: Transformation Rule 1

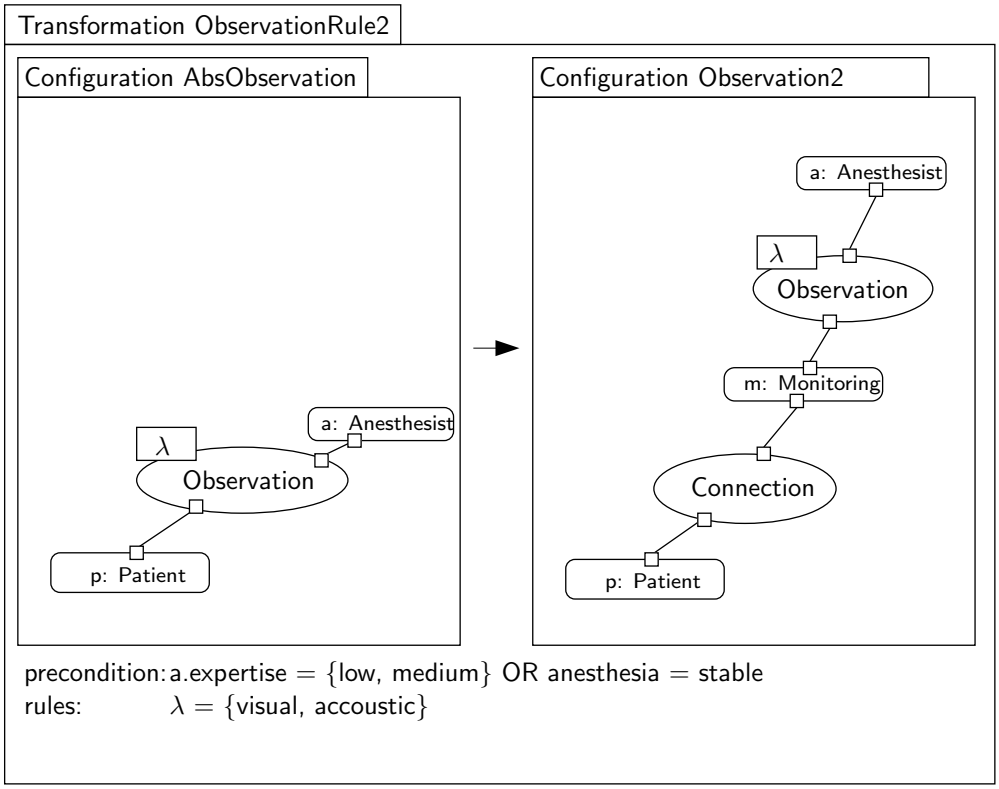

Figure 7: Transformation Rule 2

As shown in figure 9 we distinguish between three cases of human error which are grouped in two sections (following [13]):

- Execution-based errors (slips) occur when the subjective preferences of an actor are not adapted to the context (goals and beliefs may be fitting).
- Knowledge-based errors (mistakes) may occur when the beliefs of an actor are not adapted to the given context. For example a nurse may have the right goals but may not act right because she doesn't see the need to act. Another error-case occurs when an actor's goals do not fit well into the given context.

10 Conclusion

In this paper we provided a visual notation for the context adaptive behavior of complex systems. We started with structural description, dynamic architectures, and rules for reasoning about adaptive system properties. We then took a *cognitive perspective* by introducing the concepts of mental model to reflect cognitive parameters which are highly relevant for systemic behavior.

Using our transformation based approach we employ our notation for the analysis of sociotechnic adaptivity with special consideration of cognitive context conditions. We claim that the results of this analysis are useful for the design and management of context aware systems in general.

References

[1] Krzysztof Czarnecki and Ulrich W. Eisenecker. *Generative Programming. Methods, Tools, and Applications*. Addison Wesley, Reading, Massachussets, 2000.

[2] R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. In A. H. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 333–356. Kaufmann, San Mateo, CA, 1988.

[3] Ian Horrocks, Peter F.Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL, the making of a web ontology language. *Journal of Web Semantics*, 1, 2003.

[4] Janusz Kacprzyk. *Multistage fuzzy control*. Wiley, Chichester, 1997.

[5] George J. Klir and Bo Yuan. *Fuzzy Sets and Fuzzy Logic. Theory and Applications*. Prentice Hall, Upper Saddle River, N.J., 1995.

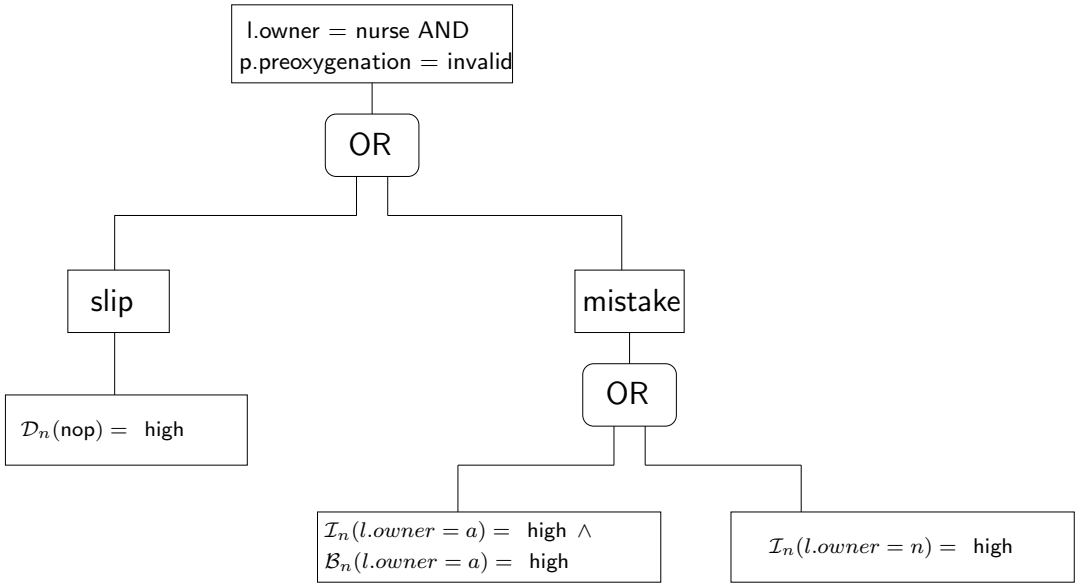

Figure 9: Fault tree for human error

[6] Nancy Leveson. *Safeware. System safety and computers.* Addison Wesley, Reading, Mass., 1995.

[7] Deborah L. McGuiness and Frank van Harmelen. OWL Web Ontology Language Overview. http://www.w3.org/TR/2004/REC-owl-features-20040210/, February 2004.

[8] Henry Mintzberg. *Structures in fives: designing effective organizations.* Prentice Hall, Englewood Cliffs, 1983.

[9] Peter Pepper, Michael Cebulla, Klaus Didrich, and Wolfgang Grieskamp. From program languages to software languages. *The Journal of Systems and Software*, 60, 2002.

[10] Peter Pepper, Ch. Frank, W. Holfelder, D. Jiang, and G. Matylis. Dynamic software architectures for a "sometimes somewhere" telematics concept. Technical report, Technische Universität Berlin, 2003.

[11] Peter Pepper and Martin Wirsing. A method for the development of correct software. In Manfred Broy and Stefan Jähnichen, editors, *KORSO: Methods, Languages, and Tools for the Construction of Correct Software*, pages 27–57. Springer, 1995.

[12] Charles Perrow. *Normal Accidents. Living with High-Risk Technologies.* Basic Books, New York, 1984.

[13] James Reason. *Human Error.* Cambridge Univ. Pr., 1990.

[14] Andrew P. Sage and William B. Rouse, editors. *Handbook of Systems Engineering and Management.* Wiley, New York, 1999.

[15] John R. Searle. *Speech Acts.* Cambridge Univ. Pr., 1969.

[16] Mary Shaw and David Garlan. *Software Architecture. Perspectives on an emerging discipline.* Prentice Hall, Upper Saddle River, N.J., 1996.

[17] Munindar P. Singh. *Multiagent Systems. A Theoretical Framework for Intentions, Know-how, and Communications.* Springer, Berlin, 1994.

[18] R. Soley. Model-driven architecture. white paper. ftp://ftp.omg.org/pub/docs/omg/00-11-05.pdf, November 2000.

AUTHOR BIOGRAPHIES

MICHAEL CEBULLA was born in Frankfurt/M, Germany and obtained his diploma in Computer Science as well as his Ph. D. in philosophy in Berlin. He worked in industry as a system designer and software developer. His research areas are: systems architecture, modeling concepts for the design and management of complex systems and specification techniques for safety-critical systems. For further information: e-Mail: `mce@cs.tu-berlin.de` or visit his homepage `http://uebb.cs.tu-berlin.de/~mce`.