CHALMERS

Improvement of Hazard Identification in Railway Software

Master's Thesis in Secure and Dependable Computer Systems

JENNY SCHULZE

Department of Computer Science and Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2010

MASTER'S THESIS 2010

Improvement of Hazard Identification in Railway Software

Master's Thesis in Secure and Dependable Computer Systems

JENNY SCHULZE

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2010

Improvement of Hazard Identification in Railway Software
JENNY SCHULZE

Examiner: Prof. Johan Karlsson

Master's Thesis 2010
Department of Computer Science and Engineering

Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Chalmers Reproservice
Göteborg, Sweden 2010

Improvement of Hazard Identification in Railway Software
Master's Thesis in Secure and Dependable Computer Systems
JENNY SCHULZE
Department of Computer Science and Engineering

Chalmers University of Technology

Abstract

This thesis deals with the problem of creating more complete, less ambiguous HAZOP deviations. In order to conquer this task we develop a model for actions. An action is a process that changes at least one attribute of at least one object. The actual model is split into three sub models. The first one of them describes what objects are involved in an action what role they play. The second one analyses the conditions that must be satisfied for the action to start (precondition) and to come to an end (postconditions). Lastly we present the relation between actions regarding time. So parameter such as start time, end time, number of concurrent actions and so on are defined. Since not all HAZOP guide words create reasonable deviation we also give a framework defining what guide words can be applied to which part of each model.

Our model creates a lot of HAZOP deviations, so it is impossible to take all of them into account. Therefore, we first eliminate redundancies, and also reduce the number of pairwise comparisons (comparison that deal with the interaction of two actions). This is done by using a similarity measure, that counts the number of commonly accessed objects of each two actions.

We also give a grammar of our model, and compare this to a HAZOP deviation model that is based on objects in steady state. The comparison has shown that both grammars are equal, since the object grammar allows creation of arbitrarily many attributes, and the action grammar has no structure on its definition of conditions.

Our model is evaluated on a new monitoring system for bogies in trains. The model created diminishes the time that is needed for a HAZOP, since the meeting and, therefore, the lengthy brainstorming session becomes dispensable. Furthermore, the systematic approach to create deviations is far more complete.

Keywords: HAZOP, action model, software risk assessment

Acknowledgements

I like to express my gratitude to my supervisor Dr Bernhard Hulin for his fruitful guidance and criticism.

Furthermore, I would like to thank my colleagues at the department *DB Systemtechnik – Rail Vehicle Software* for giving me such a stimulating environment for writing my thesis.

Contents

1 Introduction

Nowadays, trains make use of highly automatized computer systems. They possess software for braking, acceleration, opening/closing doors, air conditioning, monitoring systems and much more.

Increasing complexity of software results in a greater likelihood of introducing errors, which may pose a significant danger for human beings. Therefore, software needs to be evaluated according to its safety. In Germany this evaluation is supervised by the federal railway authority (EBA), that assigns an EBA accredited assessor for this task.

The evaluation is necessary if the software undergoes a major modification or if it is new software. This evaluation is a crucial part of the analysis that highly depends on the experience of the assessor and the software developer [1].

The assessor has to be independent from the software developer (EN 61508). Dependent on the Safety Integrity Level (SIL), the assessor has to be another persons than the developer or work in a different department or even organization. The company Deutsche Bahn AG (DB) has its own institution of assessors. This institution acts as independent organization and can, therefore, deal with all levels of safety according to EN 61508-1 8.2.12 (remark 2) [2].

The software assessors at DB deploy HAZOP (HAZard and OPerability Study) to identify hazards emerging from software. HAZOP is hazard identification method where certain guide words are combined with components of the system under study. These combinations are the basis for a brainstorming session (HAZOP meeting) in order to find possible deviations from the design intent.

However, DB's assessors experience significant problems, such as lengthy studies, incomplete hazard lists and the absence of any structure in their approach. DB would like to avoid the HAZOP meeting entirely, but only utilize the guide words of HAZOP. In order to substitute the meeting DB developed a framework that creates meaningful deviations. Extending this framework is the primary goal of this thesis.

A framework has the following advantages: The meeting is dispensable, so DB safes time and resources and thus becomes more efficient. Furthermore, DB will improve safety, because the automatized framework applies a structured approach and, therefore, identifies more hazards.

This thesis is structured as follows. First we describe HAZOP and its limitations. Subsequently we state the task of this thesis: to create a model for actions in order to gain a more complete HAZOP.

In the following section we present this model, which is divided into a time and a content model. Later on we produce useful HAZOP deviation based on these models. Since a lot of questions are generated we need to exclude some of them. Therefore, we first identify and eliminate redundancies. Secondly we reduce the number of pairwise comparisons using a similarity measure based on the objects involves in the action.

What follows is the evaluation of our model. We present a theoretical evaluation based on two grammars. There we compare a structured HAZOP based on objects and attributes to our model that is based on actions. Also a practical evaluation of a monitoring system is presented, here we compare our model to a standard HAZOP. The thesis finishes with recommendations for future work.

2 Background

HAZOP is a qualitative method for identifying hazards in a system or plant, developed by the chemical industries in 1965. It can be applied early in the design process since it evaluates only on design specification.

The analysis is conducted by applying guide words to components of the system (e. g. *no filter*) or interaction between components of the system (e. g. *less water in tank than expected*).

2.1 HAZOP Procedure

The following procedure is taken from the HAZOP standard BS IEC 61882 [3].

First of all the manager of a projects decides upon an evaluation team to conduct the study. The team normally consists of:

- a study leader,
- a recorder (documents the meeting),
- a designer (explains the design),
- users (e. g. train driver, train operator),
- specialists (expert of the system or the study),
- and maybe a maintainer.

Initially the team defines the scope and the objectives of the study, e. g. its boundaries, previous studies of the design, its purpose, the standards required and the human, environmental and financial risks that could occur in case of hazards.

In order to start the evaluation, the team now chooses the appropriate guide words and their interpretation for the to be undergone study. Furthermore, the design is split into subparts.

The study leader selects a part to be analyzed and the team clarifies the design intent and identifies possible subparts. The team leader then applies all guide word to the part. For each part—guide word combination the team gives all possible interpretations of the deviation. They study causes, consequences as well as prevention methods and record their findings. When there is no further interpretation or guide word left, the team continues with the next part. It is also possible to start with a guide word and then loop through all parts. This procedure is also shown in Figure 2.1.

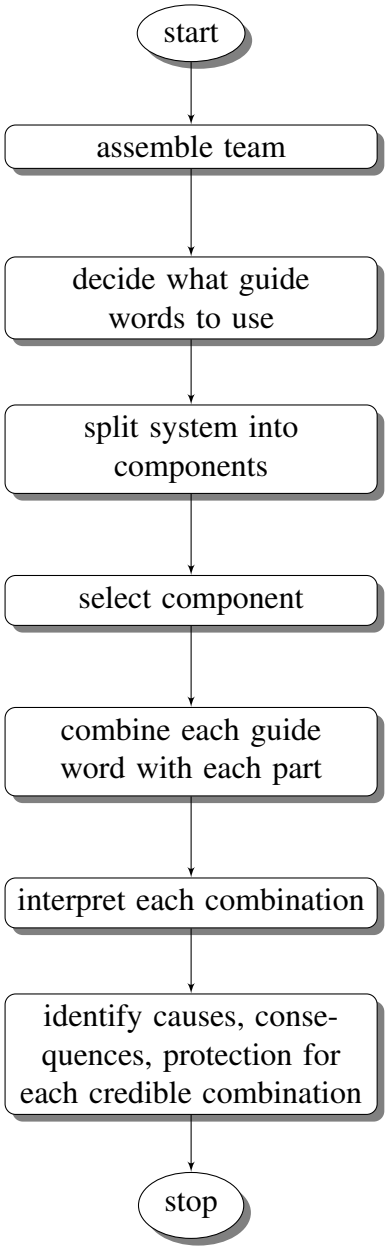

Figure 2.1: *Standard HAZOP Procedure*

Table 2.1 gives an overview about the main guide words and their meaning as defined in [3]. Different authors introduce further guide words. For instance [4] includes the guide word SOONER and LATER, to express

Table 2.1: *Basic guide words and their generic meaning*

Guide word	Meaning
NO OR NOT	Complete negation of the design intent
MORE	Quantitative increase
LESS	Quantitative decrease
AS WELL AS	Qualitative modification/increase
PART OF	Qualitative modification/decrease
REVERSE	Logical opposite of the design intent
OTHER THAN	Complete substitution
EARLY	Relative to the clock time
LATE	Relative to the clock time
BEFORE	Relating to order or sequence
AFTER	Relating to order or sequence

that a message is send earlier within a sequence of messages than intended. However, considering message timing, as done in [4], BEFORE and AFTER allow for the same interpretation. [5] introduces COMPATIBILITY (of material) and ACCESS (to equipment) as further guide words. These are specific for chemical plants and, therefore, not generally useful for software HAZOPs. Introducing further guide words is generally not considered to be needed [6] so we will mostly adhere to the standard ones.

2.2 HAZOP in Comparison to Other Fault Identification Methods

HAZOP can be seen as a compromise between FMEA and FTA [7]. All three methods try to build a tree with a hazard at the root and single node failures at the leaves. In FTA one starts with a hazard and searches for possible causes and protections. Contrary, in FMEA one begins with single node failures and tries to deduce possible consequences and protection mechanisms.

HAZOP, however, can start at any node in the tree. It subsequently searches for causes and consequences of a failure at each node and thus builds the tree in two directions: one to the root and one to the leaves. This approach allows for a much more complete tree. The use of guide words greatly assists in identifying faults since it gives some guidance on how to search for errors. This is another distinguishing feature of HAZOP.

2.3 HAZOP Derivatives

Beside excessive use in the chemical industries HAZOP is also applied to:

- mechanical plants
- electrical systems
- computer and control systems (CHAZOP)
  - alarm systems
  - military defence
  - automated control systems in general
- ...

Each derivative has its own interpretation of standard guide words. Mechanical plants behave very similar to chemical plants, so there is nearly no reinterpretation necessary. In electrical systems HAZOP focuses mostly on voltage, frequencies or current.

CHAZOP (computer HAZOP), however, is a fairly new field. The system is divided into software and hardware components. Hardware components are checked for device errors and I/O failures, but also temperature abnormalities. Software components are analysed regarding memory failures, timing failures, loops or data validation errors. After analysing all single parts the whole system is examined regarding power failures, redundancies, error diagnostics, recovery, etc. [8]

While HAZOP in the chemical industries is normally based on piping and instrumentation diagrams (P&IDs), software allows for much more aspects to consider. A lot of diagrams are usually required to illustrate most aspects. Each of these diagrams has its own set of possible guide word interpretations. The (superseded) British Defence Standard 00-58 [9] describes specialized HAZOP guide word interpretations for:

- flow control: showing how information is passed between processes
- state transition: showing what events case which actions
- object orientation: showing relationship between objects
- communication: analysis network topology, transmission rate

2.4 HAZOP Tools

Several tools to support HAZOP analysis have been developed in order to assist the evaluating team. Tools such as PhaWorks and PHA-Pro are merely content management systems specialized on HAZOP [10], therefore, they do not formulate unambiguous questions but simply list the components and a set of guide words. On the other hand HAZID tries to identify possible hazards from input models.

2.4.1 PHAPro

PHAPro is a commercial product developed by Dyadem [11]. Besides other hazard studies it supports conducting a HAZOP. The program supplies a set of templates, e. g. a spreadsheet giving columns for causes, consequences and recommendations to be filled out by the HAZOP team. Out of the spreadsheet the program is able to create a HAZOP report. The findings of the report can be imported to other hazard studies conducted at later stages.

Also an extendible library exists, containing previous data and thus saving time on further evaluations. Furthermore, PHAPro supports the distribution of work to different sites of a company. The main advantage of PHAPro is the standardization of the documentation interface for hazard studies.

2.4.2 HAZID

HAZID is a HAZOP software developed since 1999 emerging from Loughborough University [12]. It is used to perform HAZOP analysis on chemical plants. The system is deployed commercially by Hazid Technologies Ltd.

The program has a predefined, but extendible set of components used in chemical plants [13]. The components consist of subcomponents such as valves, pumps or pipelines [14]. Each component has as set of possible deviations (e. g. *low Pressure*, *reverse Flow*)

and attributes, such as height or volume, attached. Further each component has a set of possible hazards assigned, using cause and consequence relationships [12]. Also fluids are described by a set of attributes, e. g. density or viscosity.

During the analysis the user supplied plant model is transferred into a signed directed graph. HAZID traces all possible paths the fluids could take and creates a list of of possible hazards that can occur and the source of the hazard. These hazards are ranked by their severity and given to the user for further analysis.

HAZID reduces the time needed for evaluation and reduces ambiguity resulting from incompletely defined guide words. On the other hand component and fluid definitions might contain errors and thus may lead to unidentified hazards.

Currently HAZID only analysis the steady state of the plant, thus the starting phase is completely ignored. This issue is addressed by [12], but further research is needed.

HAZID uses the approach of specialization (on chemical plants), e. g. it defines every component, hazard and fluid exactly. It thus limits its applicability to other areas. PHAPro only supports the documentation of a HAZOP. However, both approaches fail to improve the conceptual weaknesses of HAZOP, that we illustrate next.

2.5 Drawbacks of HAZOP

Although HAZOP is widely used it has certain disadvantages. First of all HAZOP is rather time consuming, since it requires a group of persons for the discussion. Secondly, the repetitive cycle of selecting a part, a guide word and finally brainstorm for interpretations makes the process very tedious. Also not all participants can take an active part in the discussion of every deviation, so they stand idle and might get bored soon. Thirdly, parts, guide words or interpretations could have been forgotten, of left out due to time constraints, so that the study becomes incomplete. Furthermore, the HAZOP discussion might digress to unimportant aspects in case of an inexperienced team leader.

In order to give an impression about the current state of actions in HAZOP we give the following example: Assume we consider *driving* as an action and we would like to apply the guide word MORE to this action. Now we get into trouble, because *MORE driving* could mean:

- driving longer distances
- more people drive the same vehicle
- more people drive different vehicles
- driving for a longer time period
- the same persons drives more vehicles
- driving faster

The possibilities are endless. In fact, if we simply take a guide word and an action we may get ambiguous output.

2.6 Limitations of HAZOP

So where do tediousness, incompleteness and digression come from? The intent, of course, is to have exactly one HAZOP expression (HE) that results in exactly one deviation. A HAZOP expression is a combination of a guide word and a part of the system. However, other relations apart from 1:1 are possible, as shown in Table 2.2. It is easy to see that there are no further possibilities for limitations.

Table 2.2: *Intent and Limitation of HAZOP, HE means HAZOP expression*

Limitation	Ratio	Explanation	Illustration HE	Illustration Deviation
incompleteness	0:1	deviation cannot be found		
redundancy	∗:1	some HEs lead to the same deviations		
ambiguity	1:∗	HE leads to more than one deviation		
absurdity	1:0	HE does not make sense		

Intent	Ratio	Explanation	Illustration HE	Illustration Deviation
intent 1	1:1	one HE leads to one deviation		
intent 2	0:0	no HE implies no deviation		

Ambiguity A HAZOP expression leads to different deviations. Assume there is a sensor that sends data. Now we want to apply the guide word MORE to this phrase and thus get: *sensor sends data MORE THAN expected*. This can be interpreted as the data is sent more frequent to one object, or it is sent to more than x objects. Meaning the time span as well as the actual receiver of the data is left out, and thus gives possibilities for interpretations. Further kinds of ambiguities are illustrated in Table 2.3.

The clarification of these ambiguities lengthens the HAZOP group discussion, but is intended in the HAZOP design. Nevertheless, in order to automatize the process ambiguities have to be reduced.

Table 2.3: *kinds of ambiguity*

Ambiguity	Explanation	Example	Interpretation
syntactical	HE can be parsed in different ways	sensor sends data MORE THAN expected	MORE sensors MORE data MORE sending
semantical	words have different meanings	bus	vehicle data bus

Incompleteness Some aspects cannot be interpreted using the standard set of guide words. For instance the statement of existence ($\exists$) is simply left out in standard guide word literature [3], one can only say that *something does not exist* using the guide word NO. Nevertheless, it plays an important role in software evaluation, since whether for some unintended reason a file is already existent, might be a serious hazard if a program plans to overwrite it.

Redundancy Some HAZOP expressions may lead to the same deviation, i. e. they cause the same hazard. E. g. *sensor sends parts of the data* and *sensor sends less data than expected* can be interpreted in the same way, e. g. the intended receiver does not get the whole data packet it expects. We give a list of kinds of redundancies in Table 2.4. Redundancies generally lengthen the HAZOP discussion and should be avoided.

Table 2.4: *kinds of redundancy, μ denotes an expected value, redundancy between two deviations a and b means $a \equiv b$*

Redundancy	Explanation	Example
commutative	the same deviation can be achieved by reversing HE	*a MORE THAN b* $\equiv$ *b LESS THAN a*
causal	one deviation directly causes another deviation	*NO current* $\rightarrow$ *NO traction*
domain	some HEs have the same domain (under certain circumstances)	*less train driver THAN* $\mu \equiv$ *NO train driver* (assuming one driver per train)
point of view	deviation regarding one aspect is repeated in another	*MORE clients write THAN* μ(content aspect) $\equiv$ *writing done MORE often THAN* μ (time aspect)

Absurdity Some HAZOP expressions simply do not make sense. Statements like *tree early* seem to be rather vaguely interpretable in a software context. The HAZOP team would waste time searching for possible meanings of this statement, and most likely would not find anything useful. Some HEs such as *no object A* might be the design intent, thus do not represent a deviation. There are three kinds of absurdity in HAZOP, namely syntactical, semantical and pragmatical absurdities. Lexical absurdity cannot occur, since the guide words are fixed and the HAZOP parameters are taken from the system under study. We give a definition and examples in Table 2.5.

Table 2.5: *kinds of absurdity*

Absurdity	Explanation	Example
syntactical	structure of HE illegal, e. g. missing operant	train AS WELL AS
semantical	syntactical correct HE, but it is impossible to deduce something	What happens if the *quality* is REVERSE?
pragmatical	semantically correct and easily interpretable HE, but totally useless regarding hazards	What happens if the *train's length* is MORE THAN the *tunnel's height*?

3 Purpose

Using the traditional HAZOP method, DB assessors identify system components while scanning through design documentations of the system they intend to evaluate. Then the assessor combine various HAZOP guide words with these components. As described above this is not the best approach possible, since it results in a huge amount of useless deviations and ambiguities. DB aims to have a more structured framework that just combines guide words and system components in case they result in a meaningful deviation.

Owing to the complexity of this task, DB wants an automated tool for question generation. An assessor supplies objects, properties and actions of the system. The tool responds with a list containing all possible deviations for this software. To simplify evaluation each deviation is supposed to have a certain priority. Also, in order to limit the length of the list some deviations should be left out, either because the assessor is not interested or they are somewhat irrelevant.

This list is sent to the participants of the HAZOP meeting, that try to answer these questions. For each question they assign a risk and list possible causes and consequences. The assessor's task is now to compare the replies and decide upon an overall risk level. Thus the HAZOP meeting becomes dispensable.

3.1 State of the Art at DB

In order to overcome the drawbacks of HAZOP DB splits the system into objects, attributes and actions, as illustrated in Figure 3.1.

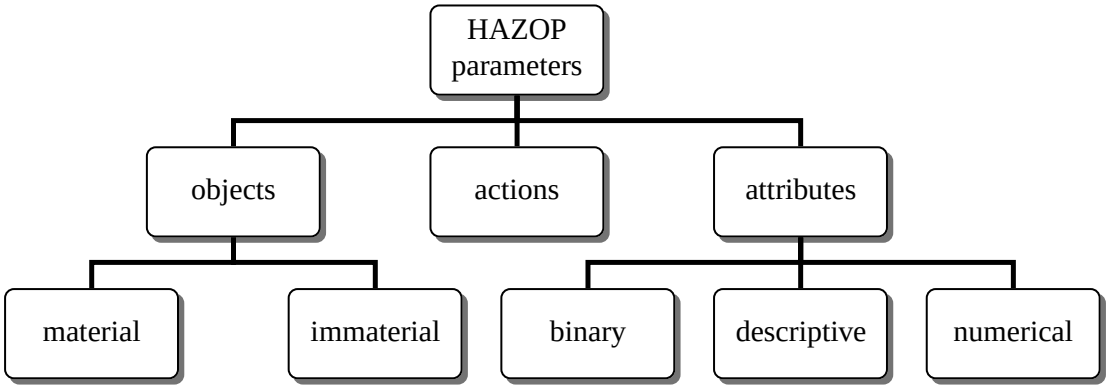

Figure 3.1: *HAZOP parameters*

The task of a previous thesis [15] was to analyse objects and attributes. The results of this analysis lead to a refinement of guide words in their relation to objects and attributes, mostly focusing on reducing ambiguity and enforcing completeness.

An object might be anything that is either immaterial, e. g. a program, or material, e. g. a train. Some objects can be divided into subobjects. For instance a train may have the subobjects locomotives and wagons and the attributes height and length. Attributes are either binary, descriptive or numerical.

Inside a program attributes are represented as variables. However, these variables do not necessarily need to have the same data type as the attribute they represent. For instance

the pressure 30 bar can be expressed as string *"thirty bar"* or as integer *30*. Therefore, variables are considered as special objects having at least a data type (e. g. integer), a name (e. g. pressure), a value (e. g. 30) and an interpretation (e. g. 30 bar).

The guide words are clustered into unary and binary ones. Binary ones are those that need either an object or an attribute, but also another value. A unary guide word stands on its own, just needing an object or an attribute. The unary ones are: NO/NOT, EXISTS and the binary ones: MORE, LESS, REVERSE, OTHER THAN, EQUALS. PART OF can be either of them. On the one hand one can say *A is PART OF B*, on the other hand *A is partially present* or *functional*.

Guide word	Attribute		
	binary	numeric	descriptive
MORE	—	$a > \mu$	—
LESS	—	$a < \mu$	—
OTHER	$a \neq \mu$	$a \neq \mu$	$a \neq \mu$
EQUALS	$a = \mu$	$a = \mu$	$a = \mu$

Table 3.1: *Binary guide words and their applicability to attributes of certain type: a is the value of an attribute of a certain object, while μ is an expected value or a value of an attribute of an object*

Unary guide words can only be applied to objects. Some binary guide words, such as MORE, LESS or OTHER THAN, can also be applied to objects . Though not all binary guide word — object combinations are useful. MORE and LESS are generally applicable to innumerable objects, such as water or current. On the other hand numerable objects would require plural notion to apply MORE and LESS, e. g. MORE trains. Not all binary guide words can be applied to all attribute types as the Table 3.1 illustrates. The guide words EXISTS and EQUALS have been introduced by DB and are not considered to be standard guide words of HAZOP. AS WELL AS has not been considered by DB so far, because it is a conjunction.

3.2 Scope

The primary goal for this master's thesis is to extend the set of questions with actions and thus timing in a systematical, nearly complete and yet still practicable way.

Since it cannot be expected that a software developer would answer several 100 questions, the set of questions will have to be reduced to an as high as reasonable practical number. Therefore, it is necessary to exclude cases that occur very infrequent or do not have a high risk.

4 Models of Actions

Previous sections have shown that HAZOP suffers from certain limitations, due to lack of formalization. A model will give rules for combining guide words and actions in a meaningful way.

DB's intention is to omit the HAZOP meeting, i. e. they will leave out the part that is of utmost priority for clarification. Therefore, the most important limitation to address is the ambiguity of HAZOP expressions, since the major point of the meeting is to discuss ambiguities.

Redundancy and absurdity result in some useless deviations, so they are only an issue of time and costs, but not of safety. Incompleteness will be reduced anyway by applying a structured approach. Regardless, before diving into different models we have to define what an action actually is, and what criteria we request our model to fulfil.

4.1 Definition of Actions

Based on excessive literature research we define an action as:

- a process that changes at least one attribute of at least one object (might be temporally), thus transfers the system into a new state [16] and
- lasts for some possibly eternal interval [17]

The state transfer is initiated by an event, i. e. some precondition comes true [18]. The next state is reached when certain postconditions are met (final event).

Actions can also be nested [19]. For instance opening a file consists of checking file permissions, allocating memory, getting the file descriptor, etc. Thus actions have different levels of detail and can be generalized and specialized to any of these.

Actions can also be called procedures, processes or tasks.

4.2 Criteria for Our Model

Giving the above definition and the limitation we presented in section 2.6, we require our model to satisfy the following criteria:

answering the five W's In order to completely grasp all aspects of the action as such we require that it can answer the following questions:

What happens in the action?
Why does the action happen?
When does the action happen?
How does the action happen?
Where does the action happen?
Who does the action?

This will enable the HAZOP team to get a clear view of what is going on. There are of course also other questions to consider, such as *how often, how long* or *which, whose, whom*. However, they either address some attribute (e. g. *how often?*), or give some kind of ownership (e. g. *whose?*), thus are not primarily interesting for our case.

Our definition requires that our model also fulfills the following criteria:

having a structure We require our model to have a clear structure, so that we can easily identify subobjects and process the system automatically. These subobjects will be used in combination with HAZOP guide words.

allowing for state transition State changing involves having requirements (conditions) for state transitions. It also enables us to express causalities.

include time constraints Since an action lasts for some interval of time our model needs to be able to express this certain interval. This aspect is partly addressed by the question *When?*

allow for nesting Our model should be able to represent the system at different levels of detail.

giving involved objects Our definition requires that an action changes some attributes of some object, so our model needs the ability to represent that object. Also this aspect is partly addressed by the question *Who?*

4.3 Models in Literature

In the following we will present some models for actions. Based on our definition and criteria we have to consider models emerging from the following areas:

- natural language
- logic
- real time systems

Since deviations are interpreted by humans using natural language, we have to take a look on how actions are described in such languages. Logical approaches give us the possibility to model state transitions, while models for real time systems will help to represent durations and other time aspects.

4.3.1 Valency Model

System requirement are mostly represented using text descriptions in natural language. Actions are, therefore, hidden in the sentences of the description. A sentence normally consists of a verb and some elements. The verb is the main part used to describe an action. Nevertheless, not all verbs describe actions, e. g. *A* **is** *of type B.* does not contain any change of attribute.

In linguistics verbs have a certain valency [20]. Valency describes what influence a verb can have on its attached objects. Verbs can have:

- a governing object — the executor of the action
- a direct object — object that is modified or transfered
- an indirect object — receiver of the direct object

For instance in the sentence *component A sends component B message M*, *component A* is the governing object, *message M* is the direct object and *component B* is the indirect object.

Based on the requirements of the verbs four different valency types can be formed. These are exemplified in Figure 4.1:

Avalency Avalent means that the verb has no governing object attached. For instance *it rains* does not have an object, although one could define *it* as object. The problem that occurs with this class of verbs is that there is no possibility to assign any attributes

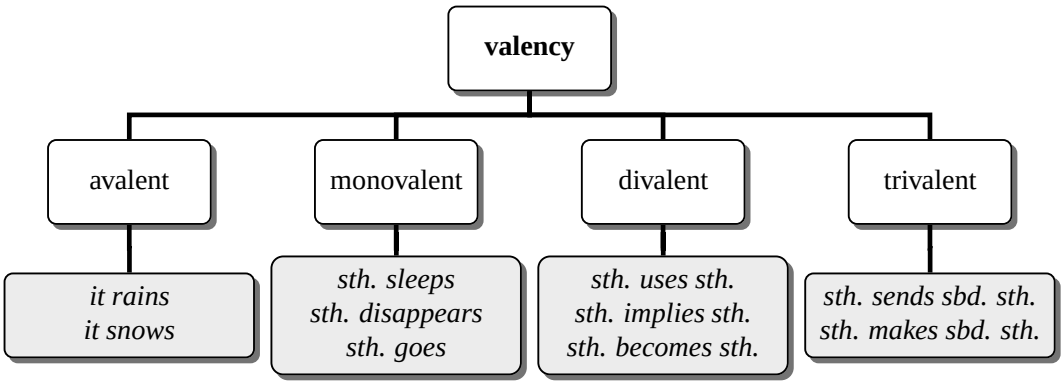

Figure 4.1: *Valencies and examples*

to the governing object, since it does not exist. However, attributes can lead to further identification of hazards.

Monovalency Monovalent verbs only use a governing object. *Process A sleeps* is an example of a monovalent verb.

Divalency A divalent verb requires a governor and a direct object. *The client reads data* is an example.

Trivalency Trivalent verbs exhibit the same components as divalent verb, but further require an indirect object, e. g. *send*.

Higher valencies Higher valencies may exist in some languages. However, this is beyond the scope of this thesis.

4.3.2 Case Structure

Borrowing ideas from linguistics, AI researchers consider the verb as the central feature of an action [21]. Consequently other elements of a sentence appear as subordinates of the verb. To allow for a structured representation each part of a sentence plays a certain role, as shown in Figure 4.2:

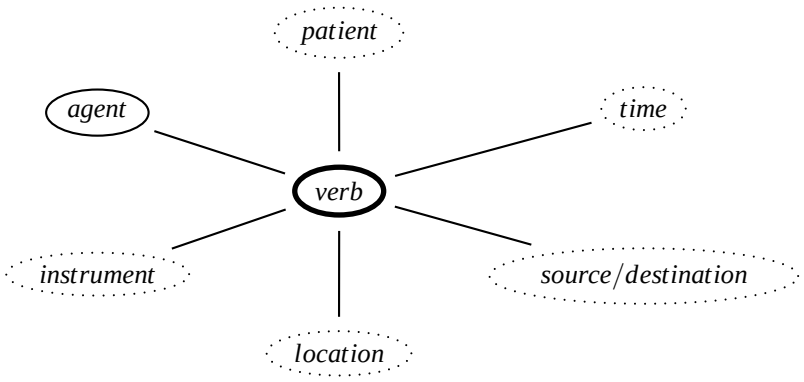

Figure 4.2: *Action model: dotted lines indicate that this element is optional*

agent The agent is the object that leads the action. As an example one could say the *process* is the agent in *The process sleeps*.

patient The object that is acted upon is called the patient. One could also say something is done to the patient. However, it can also appear in a prepositional phrase. For instance in *The passenger enters the train*, the train is the patient.

instrument The instrument is the object that is used to successfully finish the action. For example *The train is driven from A to B on the tracks* the tracks are the instrument needed to perform the action.

source/destination The destination of an action is the place where the patient is transfered to. The source is where it came from. Using the example with the train written above, A is the source and B is the destination.

time The timing aspect of the action, described in natural language. One can e. g. say *for five hours* or at *6 o'clock*, thus there is no clear structured approach.

Of course an action can have more than one agent, patient, etc.

Also source/destination are independent from locations, but can be the same. For example, sending a message takes place at the sender. The actual message is transfered to the receiver (the destination). On the other hand if a train drives from A to B, its location is not constant but gradually emerging from A to the destination B.

4.3.3 Scripts

Besides analysing what actually happens, AI researchers also considered under what circumstances actions are started and what has to be case when they finish. So they identified the need of having pre- and postconditions and thus called this model scripts [21].

The preconditions need to come true before the action can be executed. After the action has finished postconditions are met. For instance a precondition for opening a door of a train could be that the train is stopped, i. e. $Train.Speed \approx 0$. The action executes and, therefore, modifies its environment. So the door is in the open-state after it has been opened.

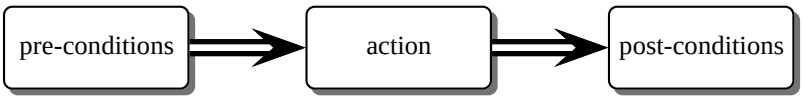

Figure 4.3: *Conditions*

4.3.4 Clock + Real Time Systems Model

Actions start and finish at some certain point in time and last for some interval. As already suggested by the standard guide words BEFORE and AFTER, wrong sequential ordering of actions can lead to hazards.

However, different clocks may come to contrasting results when comparing timing values. Assume the following setting: there are two nodes, node 1 and node 2, as well as some external observer. One event e_1 occurs in node 1 at time 3, from node 1's point of view. Another event e_2 occurs at time 2 on node 2. Both nodes immediately send a message to the other node after noticing the event. This setting is illustrated in Figure 4.4. Now one can deduce three different interpretations:

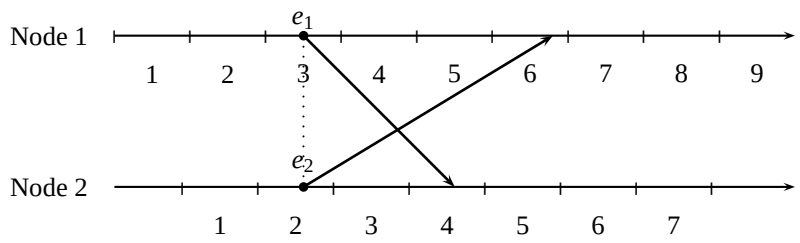

Figure 4.4: *Uncertainty in time deviations, there are two communication devices, node 1 and node 2, each of them has a clock, indicated by the horizontal lines. The scale gives the number of ticks elapsed since the start. The two clock (lines) are aligned according to an external clock. Each communication device perceives an event e, that is subsequently transmitted to the other device. The arrows show the transmission, i. e. the time when the event is sent (start of the arrow) and when it is received by the other device (the end of the arrow)*

- the two events happen at the same time: For the external observer event e_1 and e_2 happen at the same time, thus concurrently.
- e_1 before e_2, For node 1 e_1 occurs before e_2.
- e_2 before e_1, for node 2 e_1 happens after e_2.

To solve this inconsistency we introduce temporal order as defined in [22]. Thus if an event happens, the perfect clock makes a timestamp at the time the event occurs. Using timestamps two events e_1, e_2, occurring at t_1, t_2 respectively can be compared using mathematical notation ($<$, $>$). Thus if $t_1 < t_2$ then e_1 happened before e_2, if $t_1 > t_2$ then e_1 happened after e_2. In the unlikely case that $t_1 = t_2$ then the two event happen at the same time.

The Clocks As seen, we need an external observer which has a perfect clock. Furthermore, it is reasonable to equip each object with a (imaginary) clock as well. This greatly simplifies the identification of clock skew, such as the one given in Figure 4.4.

What properties does such a clock need? — Drift, accuracy and granularity. The granularity of a clock C is the time that elapses on the perfect clock between two consecutive ticks of C. The perfect clock is of very fine granularity. Two events occurring on the same node will always have a different time on the perfect clock, but it does not necessarily need to be so on an object's clock. I. e. on the object clock two consecutive events may have the same time stamp but will never have the same time stamp on the perfect clock.

Accuracy is the absolute difference of the time values between perfect clock and an object's clock. Drift (i. e. clock speed) gives the relative difference, i. e. the absolute change per time unit.

Thirdly to compare the value we need some information about the time zone, e. g. UTC or CET. This time zone information includes summer and winter time.

Time Parameters After having introduced a notion for the clock we, still, need to give timing notion for the actual action. In synchronous distributed systems, tasks (i. e. actions) have a period, a deadline, some offset and a WCET (Worst Case Execution Time) [17]. Figure 4.5 gives an example of the resulting model.

- The period describes the rate at which the task is repeated.
- The deadline is the time relative to the start of the period at which the task should have been completed.
- While the offset says how much time has to elapse before a task can start.
- WCET gives the uninterrupted worst case execution time of the task.

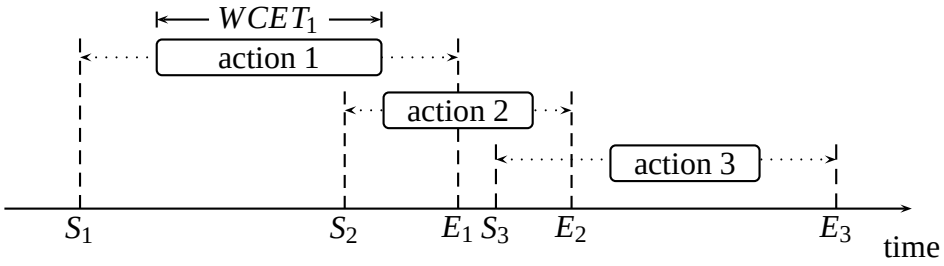

Figure 4.5: *Time model: The Figure shows three different actions — 1, 2 and 3. Each of them has a start time S_x and an end time E_x. We also illustrated the WCET of action 1. Actions whose periods overlap, such as action 1 and action 2, could be executed concurrently. Also two actions can be executed sequentially, i. e. the end of one action is before the start of another, such as action 1 and action 3. Of course also a single action can be executed more than once in its assigned interval, either sequentially or concurrently.*

4.4 Evaluation of Models

We now elaborate in what way each of the presented models fulfills our criteria. First of all we check in what way they can answer the five W's (see Table 4.1). As can be seen easily, a combination of all models can satisfy this criterion, but no model for itself.

Certain questions are only answered by a single model:

- *How?* is only sufficiently expressed in DB's current model, through use of attributes.
- Scripts is the only model that gives the possibility to answer the question *Why?* using conditions.
- *Where?* is only answered by the case structure model (taking the location).

A combination of these three models, DB's current model, scripts and case structure, is capable of answering all five W's.

Table 4.1: *applicability of our models to the five W's*

model	**five W's**					
	What?	Why?	When?	How?	Where?	Who?
DB's current model[a]				•[c]		
valency model	•					•
case structure	•		•		•	•
scripts	(•)[b]	•	(•)[b]		(•)[b]	(•)[b]
clock + RTS			•			

[a] As described in 3.1

[b] (•) indicates that the model can answer this question, but we have not elaborated this fact here

[c] through use of attributes

Table 4.2 gives an evaluation of the other criteria we specified. It is obvious that the valency model does not give us anything useful in regards to our criteria. To begin with, it has an ambiguous structure, since some verbs appear divalent in one context, but trivalent

in another. For instance *to get* can denote *to understand* (get the idea) which is divalent, but it could also mean *to bring* (get someone something) (trivalent). Furthermore, it does not have any kind of timing constraints or ability for state transition, since it only considers objects.

We, furthermore, see that state change is only represented by the script model. This is due to the use of conditions, that accurately show when a transition has to take place and what is the next state. However, one has to remark that conditions are given in natural language and thus prone to ambiguity.

Similarly time constrains can be only expressed in an acceptable manner using the clock + RTS model. Other models have some kind of timing constraints, but somehow fail to give a model that goes beyond natural language description.

All in all we cannot say that any of the presented models is able to suit our needs completely. Therefore, we need to create our own model by taking the best parts of the presented models.

Table 4.2: *comparison of the presented values, — denotes that criteria is not supported by the model, + denotes that the criteria is partly satisfied by the model, ++ denotes that the criteria is sufficiently fulfilled by the model*

model	**criteria**				
	structure	state transition	time constraints	nesting	involved objects
DB's current model	++	—	—	++	++
valency model	—	—	—	—	+
case structure	++[a]	—	+	+	++
scripts	+[a]	++	+	+	++
clock + RTS	++[b]	—	++	—	—

[a] content structure

[b] time structure

5 Our Model

We give an illustration of our model in Figure 5.1. As can be easily seen we have adopted some aspects of the case structure, the scripts and big parts of the clock + RTS model.

However, we modified the case structure model slightly, due to the fact that it highly concentrates on the verb. This approach has some drawbacks. First of all, the current DB model considers objects as centrally important. Since most verbs have an agent we find it much easier to simply assign the action to the agent and not vice versa. This makes it much easier to combine both models. Furthermore, we omitted the time constrains of this model since they are sufficiently represented in the clock + RTS model.

No adjusting has to be done for the script model, its pre- and postcondition can be used right away.

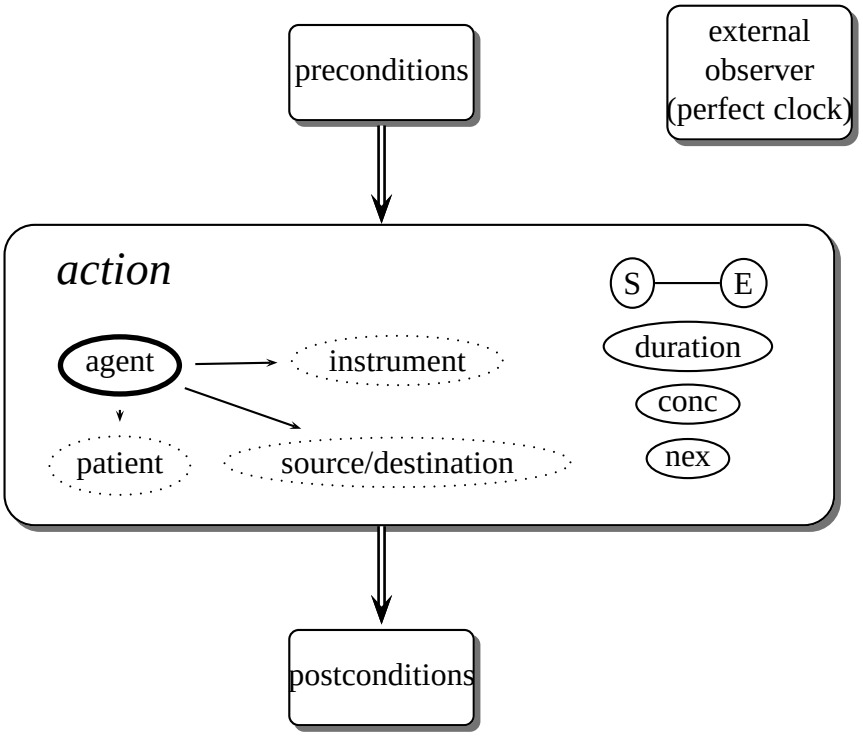

Figure 5.1: *Content model: the agent is the center of an action, dotted arrows indicate that the target is optional*

However, the clock + RTS model needs some simplification, since its time parameters are far too detailed for a requirement specification. First of all the WCET is impossible to get at that stage of the design. We, therefore, decide to use duration instead. This is not an absolute duration, it merely describes that duration as such has to be considered.

We combine the deadline and offset measure to give a start time S and an end time E of the task, as well as an execution interval I. In the same way as the duration the start and end time of an action are purely symbolic, even though they can be used to give some causal ordering between actions. We further on have to include the number of concurrent instances of an action and the number of sequential instances.

Thus an action has the following time parameters:

- start
- end
- duration
- number of concurrent instances (conc)
- total number of executed instances (nex)

Each object also has its private clock and there is an omnipotent external observer, which has a perfect clock.

To summarize, an action needs an agent object that executes the action. Based on the desired outcome further objects, such as patients, instruments, and sources or destinations might be needed. Moreover, the action has some time parameters as well as a set of pre and postconditions that need to be fullfilled at the start or end of the action, respectively.

Each object has its own imaginary private clock. In order to allwo for comparison between these clocks, our model facilitates an external observer.

6 Deviations

Deviations are created according to the traditional HAZOP way in applying all guide words to each subelement of an action. These subelements are shown in Figure 5.1. However, not all guide words can be applied in all situations as we show in the following. Our basis for choosing appropriate guide word is DB's current model, described in [15]. For convenience these guide words are also shown in Table 6.1.

Table 6.1: *Guide words in their relation to objects and attributes, as described in [15]*

Guide word		attribute	
	Object	**binary/descriptive**	**numerical**
NO	•		
EXISTS	•		
MORE (OF)	•		•
LESS (OF)	•		•
OTHER THAN	•	•	•
PART OF	•		
EQUALS		•	•

6.1 Timing Related Deviations

Starting off with our timing submodel we take all timing parameters (start, end, duration, conc, nex) and apply all guide words that seem reasonable. All time values are numerical. Thus, according to Table 6.1 we can apply the guide words MORE, LESS, OTHER THAN and EQUALS. The standard guide words EARLY and LATE are similar to MORE and LESS, where EARLY means the parameter is less than expected and LATE that it is more than expected.

The guide word OTHER THAN is redundant since it would either mean MORE or LESS. Also usage of EQUALS is pointless, because we expect the value of the attribute to be in the expected range.

In fact, we only apply the guide words MORE and LESS to all timing parameters we find in Figure 4.5. Table 6.2 shows the complete list of deviations applied to all time parameters.

Besides checking for an early or late start one could also check for an early or late end of the action. However, the HAZOP expressions MORE/LESS time account for this in most of the cases.

Table 6.2: *Complete list of guide word interpretations for timing*

HAZOP expression	**Interpretation**
MORE/LESS start	action starts earlier/later than expected
MORE/LESS end	action ends earlier/later than expected
MORE/LESS duration	action lasts longer/shorter than expected
MORE/LESS conc	more/less instances of the action execute concurrently
MORE/LESS nex	action is more/less often executed than expected

6.2 Condition Related Deviations

As already stated an action has some preconditions and some postconditions. A condition is per se an object. So according to Table 3.1 we can make use of the guide word NO.

- EXISTS is not applicable since a condition is a defined object and is, therefore, intended to exist.
- Also PART OF is not applicable since a condition is either true or false, thus does not embody any kind of intermediate state.
- MORE and LESS also do not adapt to a true and false state.
- OTHER THAN is similarto NO in this case.

Table 6.3 lists the discussed guide word and their interpretation.

Table 6.3: *Guide word interpretation for conditions of actions*

HAZOP expression	Interpretation
NO precondition	action executed, but precondition not met
NO postcondition	action executed, but postcondition not met

6.3 Content Object Related Deviations

Similar to conditions we can apply guide words to the agent, patient, source/destination and instrument of an action. Likewise we consider these elements as objects. So applicable guide words, according to Table 6.1, are MORE, LESS, OTHER THAN, NO/NOT, PART OF and EXISTS.

- MORE/LESS tells that the action requests more or less objects of these kind than expected.
- OTHER THAN gives rise to a substitution of any element by something else.
- NO simply means that the required object is not used, while PART OF denotes it is partly used.
- EXISTS is excluded, since agents, instruments and sources/destinations, are generally already existent before the action takes place, so EXISTS is intended. On the other hand considering patients EXIST does make sense, since an object might be created during execution of an action. In that case it is probably not desired that this object existed beforehand, e. g. creating a file may have the side effect of deleting another one.

Table 6.4 gives possible deviations gained from applying the basic guide words.

For illustrational purposes we give some examples. Assume a node has the ability to send data on a redundant bus. So node is the agent of the action, the message acts as patient and the bus represents the destination.

- *MORE messages* means that the node sends more data than expected.
- *PART OF bus* denotes that the data does not reach all participants on the bus.
- *OTHER THAN node* describes that this node does not send but some other node performs this action.

Table 6.4: *Guide word interpretation for content of actions*

HAZOP expression	Interpretation
agent A	
NO A	object O does not exist
OTHER THAN A	another object than intended executes the action
MORE A	more agents than expected execute the action
LESS A	less agents than expected execute the action
PART OF A	only parts of the agent are fully functional and thus able to execute the task
object O, can be patient, instrument or source/destination	
NO O	object O is ignored during execution of the action
OTHER THAN O	object O is substituted by something else during execution of the action
MORE Os	the action makes use of more objects of O's kind
LESS Os	the action makes use of less objects of O's kind
PART OF Os	only parts of O are accessed by the action
AS WELL AS Os	O executes action A on the specified objects, but also on some other ones
patient O	
EXITS O	object existed before execution of the action

6.4 Timing and Content Related Deviations

Timing and content related deviations are those that have wrong time values as well as wrong content. They only deal with the action object itself, not its attributes. As already described the guide word PART OF belongs to this section, but also NO.

NO means the action is not executed at all. PART OF means that the action has started but does not finish successfully. This behaviour can be achieved in two ways, either the action begins to loop, or is aborted [23]. Therefore we introduce two new guide words — LOOPS and ABORTS — for either case. Table 6.5 gives the list of deviations.

Table 6.5: *Guide word interpretation for content and timing of actions*

HAZOP expression	Interpretation
action a LOOPS	action a loops forever
action a ABORTS	action a is aborted
NO action a	action a does not execute

6.5 Relations to Other Actions

Beside considering just one action, we can also compare values of two actions. An action consists of a set of time parameters and some content objects.

As a matter of fact, only time parameters of the same kind should be compared. So we compare the start value of an action a to the start value of an action b. Comparing the start value of one action to the end value of another, is also possible. However in this particular case these two action happen concurrently, so we do not need to compare start and end values.

The deviations are similar to those in section 6.1, but also include EQUALS. In order to avoid confusion, the time used for comparison is always the time measured by the perfect clock.

Furthermore, content objects (agents, patients, instruments, sources, destinations) can also be compared with each other. Meaning if node a makes use of an instrument it seems wise to check what role this instrument plays in another action. Therefore, we use AS WELL AS an indication of possible concurrent access to an object in the system. Table 6.6 gives an overview about interaction deviations.

Table 6.6: *Guide word interpretation for timing in relations: the reference clock is always the perfect clock*

HAZOP expression	**interpretation**
LESS start	action a starts before action b
EQUALS start	action a starts at the same time as action b
MORE start	action a finishes after action b
LESS end	action a finishes before action b
EQUALS end	action a finishes at the same time as action b
MORE end	action a finishes after action b
LESS duration	action a takes less time than action b
MORE duration	action a takes more time than action b
EQUALS duration	action a takes the same time as action b
LESS conc	the number of concurrent executions of action a is less than the number of concurrent execution of action b
MORE conc	the number of concurrent executions of action a is more than the number of concurrent execution of action b
EQUALS conc	the number of concurrent executions of action a is the same as the number of concurrent execution of action b
LESS nex	action a is less often executed than action b
EQUALS nex	action a and action b are equivalently often executed
MORE nex	action a is more often executed than action b
AS WELL AS	action a and action b require access to the same object

7 Number of Deviations

In order to analyse the usability of the above mentioned framework we have to find an upper and a lower bound for the number of questions, i. e. HAZOP deviations.

Therefore, we classify our HAZOP expressions into four groups, as shown in Figure 7.1. Firstly we can check timing attributes, such as what happens if the start time deviates from its expected value μ. However, actions also have content objects, where we define deviations such as OTHER THAN patient. This gives the second group. The third groups contains comparison between time parameters of two actions, while the last one takes care of possibly concurrent access to some object.

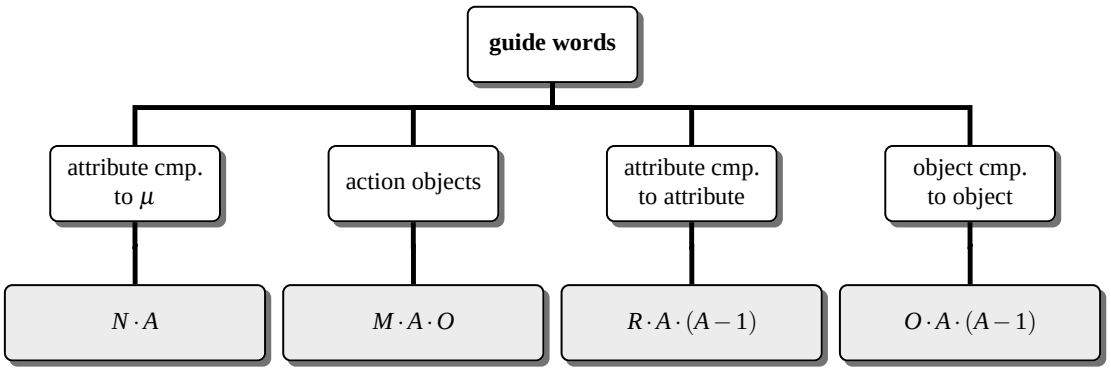

Figure 7.1: *Classification of guide words and the number of deviations created at maximum (see section 7.1 for an explanation of the formulas)*

7.1 A Formula

Assume that there are N HAZOP expressions that compare some attribute of an action to some expected value (the first group in Figure 7.1), i. e. not to another action. Then we would need to apply every guide word to every action thus accounting for

$$N \cdot A \tag{7.1}$$

questions, where A is the number of actions.

Also assume there are M HAZOP expressions that concern deviations dealing with objects (the second group), e. g. omission of patients, agents, conditions etc. Since each action could, at maximum, make use of all objects, we thus need:

$$M \cdot A \cdot O \tag{7.2}$$

questions at maximum, where O is the total number of objects.

Furthermore, we need to consider those HAZOP expressions that compare two attributes of two actions, namely the third group in Figure 7.1. Let us call the total number for these expression R. We need to compare every actions to every other action and to every HAZOP expression, so there are in total:

$$R \cdot A \cdot (A-1) \tag{7.3}$$

The last group in Figure 7.1 only contains the guide word AS WELL AS. In the worst case each action deals with every object, so we would need:

$$(O-C) \cdot A \cdot (A-1) \tag{7.4}$$

questions for this case. Here we have to subtract the number of conditions (C), since they cannot be access by two objects at the same time.

Thus the obvious formula for the total number of questions becomes:

$$(N+MO) \cdot A + (R+O-C) \cdot A \cdot (A-1) \tag{7.5}$$

We illustrate these equations in Figure 7.2.

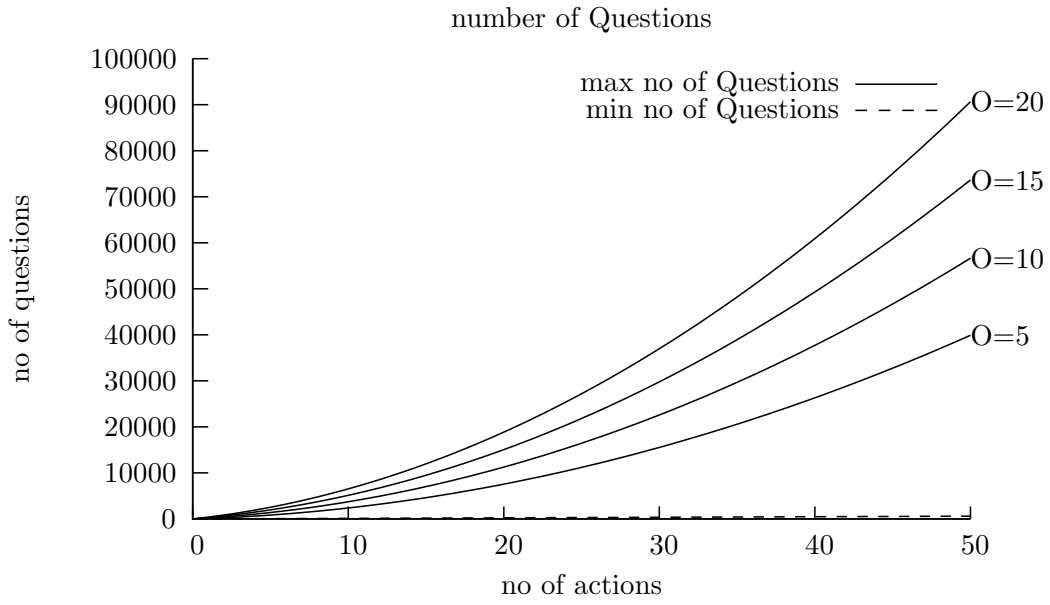

Figure 7.2: *Number of questions (omitting the condition term)*

7.2 Reducing the Number of Questions

It is easy to see that simply creating HAZOP expressions results in a vast number of questions. Evaluating all these questions will result in enormous costs. However, only a small fraction of these questions will lead to hazards. The question is how to find this small fraction.

One way of reducing the number of questions is to identify redundancies, thus we do not loose any information. Another way is to skip comparisons between actions that do not seem to be relevant.

7.2.1 Lossless Reduction by Eliminating Redundancies

Our presented model contains redundancies:

MORE [time parameter] — LESS [time parameter] While comparing time parameters actions to each other, MORE [time parameter] is always redundant to LESS [time parameter]. For instance if action A takes LESS TIME than some action B, it is obvious that action B takes MORE TIME than action A. Therefore, *MORE time, MORE conc* and *MORE often* are redundant to their *LESS time/conc/often* counterpart, respectively.

action ABORTS — NO postcondition There is a redundancy between the aborting of an action and the postconditions, since if the action is aborted at least one of its postconditions will not be met. On the contrary, the HAZOP expression *action LOOPS* is not redundant, the action does not finish and might still influence other objects.

NO agent — NO precondition — NO action When the action is executed it is implicitly assumed (in the precondition) that its agent exists. If it would not exist it is hardly imaginable how that action could even start its execution. Therefore, we exclude *NO agent* and *NO action*.

NO patient, NO instrument, NO source/destination — NO postcondition In case the patient is missing, e. g. does not exist or is not accessed, the whole action fails, i. e. its postcondition(s) are not met or it is aborted. The same applies to *NO instrument*. If e. g. a data bus is not existent, or the action does not make any use of it, although it is supposed to do so, the action fails to satisfy all of its postconditions. Similar for *NO source/destination*. If the destination does not exist, it cannot receive or send anything from or to the agent of the action so not all postconditions are achieved either. Thus we exclude *NO patient/instrument/source/destination*.

NO, LESS/MORE nex — causal actions Assume, there is an action A that needs to execute successfully so that action B can take place. If A fails then B will fail as well. There is no need to ask what happen if A is not executed or A is less often executed, since it results in the same deviation as asking for B. Thus for causal relations one only needs to evaluate those actions at the end of the causal chain, namely B in our example.

LESS/MORE/EQUAL start/end — infinitively repeated action In case an action is regularly executed, such as polling a bus, there is no need to ask what could happen if an action occurs before or after another regularly executed task. Therefore, comparing two regularly repeated actions regarding their start and end time does not result in deviations.

MORE/LESS agents — MORE/LESS often If an action is more or less often executed than expected, it will in most cases have more or less agents than expected. This content deviation (MORE/LESS agents) is, therefore, redundant to the timing relation (MOE/LESS often).

MORE/LESS start LESS start (=action executed to early) can be interpreted as to early in relation to a specific fixed point in time, earlier than some precondition is satisfied or even earlier than another action. The two later ones are dealt with by other guide words. The fixed point in time can be expressed in a precondition. So there is no need for LESS start and thus similar for its counterpart MORE start.

NO content object — LESS/MORE content object Some content objects only exists in one instance, either because it is just one single piece of hardware or it is imaginary such as a port number. In that case it is not necessary to question the quantity of such an object. A deeper analysis of this problem is given in Section 7.2.2.

Table 7.1 list the guide words that remain after eliminating all redundancies. Still, a huge number of HAZOP expressions persists, especially those that compare attributes of two actions.

Table 7.1: *List of HAZOP expressions*

Type	HAZOP expressions
attributes cmp. to μ	LOOPS action, NO action, MORE/LESS time, MORE/LESS conc, MORE/LESS nex
attributes cmp. to attribute	LESS start, EQUALS start, LESS end, EQUALS end, LESS duration, EQUALS duration, LESS conc, EQUALS conc, LESS nex, EQUALS nex
action objects	NO preconditions, NO postconditions, PART OF agent, OTHER THAN agent, PART OF patient, AS WELL AS patient, OTHER THAN patient, MORE patients, LESS patients, EXISTS patient, PART OF source/destination, AS WELL AS source/destination, OTHER THAN source/destination, MORE sources/destinations, LESS sources/destinations, OTHER THAN instrument, MORE instruments, LESS instruments, PART OF instruments, AS WELL AS instruments
object cmp. to obj.	AS WELL AS

7.2.2 Lossless Reduction by Objects in Actions

Above we illustrated what role objects can play in actions. As said they can appear in pre- and postconditions, act as agent, patient, instrument or source/destination. However, there are some other aspects of objects that are desirable to look upon when modelling actions. These are shown in Figure 7.3.

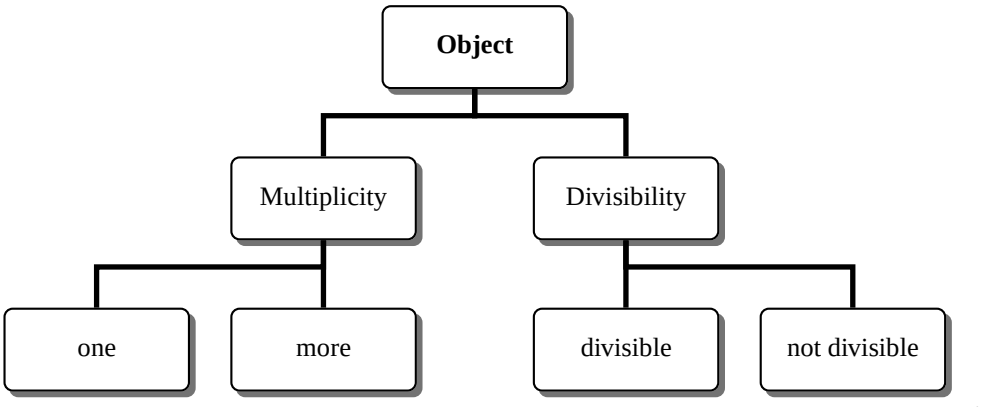

Figure 7.3: *An objects and its useful properties for modelling actions*

Multiplicity We have objects that are only in singularity represented in the system. If there is only one object of a certain kind in the system, its multiplicity is one. A train normally has just one driver or a specific port in a communication protocol does not have any further representations. Thus deviations like *MORE/LESS often* can be neglected for object that exist in singularity, which is the advantage of this classification.

Divisibility An object is divisible when some of its parts can fail and the object is still able to accomplish at least parts of its functionality or it is possible to use only parts of this object. That is the object has a degraded mode. Objects that do not have any kind of degraded mode are indivisible. For such objects some deviations can be omitted. For instance a train waggon that is sliced in half, is certainly not operable anymore. However, if one part of a redundant unit fails, it is still functioning.

7.2.3 Lossy Reduction by Similarity Measure

As presented above, most deviations originate from comparisons of time parameters of actions among each other. Thus we have to find a method enabling us to reduce this number.

We assume that actions that are alike, interfere more with each other than actions having completely different content. How can we define this similarity? Considering the previous analysis we decomposed actions into conditions, agents, patient and the like. The obvious idea is to use these elements to define similarity.

Thus we create a boolean matrix in the following manner: one row for each action and one column for each object. The element $M_{i,j}$ is 1 if any of the elements in action i (conditions, content elements, agent) contains object j, otherwise $M_{i,j}$ is 0. Thus we can compare each action to any other actions using this binary vector, i. e. giving a similarity between two actions.

The literature [24] gives numerous methods to define similarity.

For simplicity we only consider the following three measures that define similarity between two actions i and j:

$$S_{i,j} = a \tag{7.6}$$

$$S_{i,j} = \frac{a}{Min(a+b, a+c)} \tag{7.7}$$

$$S_{i,j} = \frac{a}{Max(a+b, a+c)} \tag{7.8}$$

where

a ... number of objects used by both actions
b ... number of objects only used by i
c ... number of objects only used by j

Equation 7.6 has the disadvantage that it needs to be normalized, in order to define levels of similarity. Nevertheless this can be easily achieved. Thus all three measures will have the same range [0,1] and thus the assessor can define levels of similarity.

To evaluate which of these mentioned formulas is most appropriate for our problem we give this example:

action	$O1$	$O2$	$O3$	$O4$	$O5$	$O6$	$O7$
a	1	1	1	1	1	1	1
b	1	1	0	0	0	0	0
c	0	0	1	1	0	0	0
d	0	1	0	1	0	1	0
e	0	1	0	1	0	0	0

Applying our formulas we gain the results illustrated in Table 7.2.

Table 7.2: *Level of similarity between all action unsing the formulars 7.6, 7.7, 7.8*

	Equation 7.6				Equation 7.7				Equation 7.8			
	b	c	d	e	b	c	d	e	b	c	d	e
a	1	1	1	1	2	2	3	3	2/7	2/7	3/7	2/7
b		0	1/5	1/2		0	1	1		0	1/3	1/2
c			1/5	1/2			1	1			1/3	1/2
d				1				2				2/3

Based on these values we can rank the similarity and thus the importance of a comparison between two actions i and j. Table 7.3 list the resulting ranking.

It can be clearly seen that equations 7.6 and 7.7 favour comparing to actions that deal with many objects (e. g. broadcasting), while equation 7.8 focusses more on those that use roughly the same set of objects. It also gives a more fine granular distribution. Because of these properties we prefer equation 7.8 to the other ones.

Table 7.3: *Comparison of similarity measures*

Similarity	7.6	7.7	7.8
most alike	a-b, a-c, a-d, a-e, d-e	a-d, a-e	d-e
	b-e, c-e	a-b, a-c, d-e	b-e, c-e
	b-d, c-d	c-e, b-e, c-d, b-d	a-d
↓	b-c	b-c	a-b, a-c, a-e
			b-d, c-d
least alike			b-c

8 A Grammar for HAZOP

Now the question arises if our HAZOP deviation creation model is more powerful than the one previously develop by [15]. In order to facilitate this comparison we develop a grammar for both systems.

This is done in the following manner: We define HAZOP as being a list of deviations. Deviations are defined as guide words and the parameters applicable to it.

The grammar is object oriented, meaning symbols like *Object.Action.Content. Instrument* denote the instrument of action *Action* of object *Object*. If we would just write *GUIDE WORD Instrument*, thus skipping the object orientation, it is impossible to know to what action and object this deviation belongs to, so something meaningless would be created.

8.1 Grammar for Objects and Attributes

The attributes are grouped into numerical, descriptive and binary ones, since the guide words that create meaningful deviations differ based on the attribute's type. On the one hand all of these attributes can be compared to μ (some expected value), but on the other hand also to another attribute having the same type. In this way binary attributes are compared to binary attributes, but not to numerical ones.

The grammar for objects and attributes is given in Table 8.1, while the grammar for actions is given in Table 8.2.

8.2 Comparison

The relation between two grammars G and F will have at least one of the following properties:

- $G \cap F = G = F \Rightarrow G = F \Rightarrow G \subseteq F, F \subseteq G$
- $G \cap F = G \Rightarrow G \subseteq F$
- $G \cap F = F \Rightarrow F \subseteq G$
- $G \cap F = \emptyset$
- $G \cap F \neq \emptyset$

So in order to compare the object grammar in Table 8.1 and the action grammar in Table 8.2 we have to check these five cases. The approach taken is to show how each rule of a grammar can be expressed using rules of the other grammar.

8.2.1 Action Grammar $\subseteq$ Object Grammar?

Rule A1 It is obvious that rule A1 of the action grammar in equal to rule O1 of the object grammar.

Rule A2 We define condition as being child objects of actions. So *Object. Action. Condition. Pre* is certainly a term that is expressible by *object.[path to child object]*. Therefore, *NO Object.Action.Condition.Pre* is a HAZOP expression created by rule O2. The same applied to *Object.Action.Condition.Post*.

Rule A3 Similar to rule A2.

Table 8.1: *Grammar for objects and attributes: [path to child object] gives the path to the leaf attribute, this structure cannot be known beforehand, can be empty*

O1	HAZOP ::=	[Deviation];
O2	Deviation ::=	(**NO** | **PART OF** | **EXISTS** | **MORE THAN** | **LESS THAN** | **OTHER THAN**) object.[path to child object];
O3	Deviation ::=	object.[path to child object].NumericAttibute (**MORE THAN** | **LESS THAN**) μ;
O4	Deviation ::=	object.[path to child object].DescriptiveAttribute **OTHER THAN** μ;
O5	Deviation ::=	object.[path to child object].BinaryAttribute **OTHER THAN** μ;
O6	Deviation ::=	object.[path to child object].NumericAttribute (**LESS THAN** | **EQUALS**) object.[path to child object]. NumericAttribute;
O7	Deviation ::=	object.[path to child object].DescriptiveAttribute (**OTHER THAN** | **EQUALS**) object.[path to child object]. DescriptiveAttribute;
O8	Deviation ::=	object.[path to child object].BinaryAttribute (**OTHER THAN** | **EQUALS**) object.[path to child object].BinaryAttribute;

Rule A4 Similar to rule A2 for guide words OTHER THAN, MORE THAN, LESS THAN. However, the guide word AS WELL AS requires further attention. It tells that the action does what it should do but also makes use of another object. For this case we introduce additional binary attributes for actions called *usesAdditionalPatient, usesAdditionalInstrument, usesAdditionalDestination, usesAdditionalSource* with the expected value $\mu = false$. Then one can easily apply rule O5, e. g. *Object.Action.usesAdditionalPatient OTHER THAN* μ to express rule A4.

Rule A5 Since EXISTS is a guide word given in Rule O2, and a patient is an object, we can safely express rule A5 using rule O2.

Rule A6 Object.Action.Timing.parameter is a numeric attribute, so we can easily apply rule O3 to express rule A5.

Rule A7 If we consider an action as an object, *NO action* can be easily expressed using rule O2. The HAZOP expression *LOOPS* can be derived by creating a new binary attribute *loops*, having the expected value $\mu = false$. Then we can apply rule O5: *Object.Action.loops OTHER THAN expected*.

Rule A8 Rule A8 tells us that two actions access the same object at the same time. This is a bit cumbersome to express in the object grammar. The objects that an action accesses are those given in the conditions as well as the agents, patients, instruments, sources and destinations. If we give each object a unique number *ObjectNumber*, we can check if these numbers are equal. So one can say e. g. *Object1.Action1.Content.Patient.ObjectNumber*

Table 8.2: *Grammar for actions: Timing.param are all those parameter related to timing as illustrated in Figure 5.1, μ denotes the expected value, t is a time parameter*

A1	HAZOP ::=	[Deviation];
A2	Deviation ::=	**NO** (Object.Action.Condition.Pre | Object.Action.Condition.Post);
A3	Deviation ::=	(**OTHER THAN** μ | **PART OF**) Object.Action.Content.Agent;
A4	Deviation ::=	(**OTHER THAN** μ | **MORE THAN** μ | **LESS THAN** μ | **AS WELL AS** | **PART OF**) (Object.Action.Content.Patient | Object.Action.Content.Instrument | Object.Action.Content.Source | Object.Action.Content.Destination);
A5	Deviation ::=	**EXITS** Object.Action.Content.Patient;
A6	Deviation ::=	(**MORE THAN** μ | **LESS THAN** μ) Object.Action.Timing.parameter;
A7	Deviation ::=	(**LOOPS** | **NO**) Object.Action;
A8	Deviation ::=	Object.Action **AS WELL AS** Object.Action;
A9	Deviation ::=	Object.Action ((**EQUALS** t) | (**LESS THAN** t)) Object.Action;

EQUALS *Object2. Action1. Content. Instrument. ObjectNumber* (rule O6). If one would derive that these two actions access the same object is of course questionable.

Rule A9 All time parameters are numeric attributes, so rule A9 is easily expressible using rule O6.

Conclusion In general we can say that the action grammar is a subset of the object grammar. Sometimes it is necessary to abuse the attribute creation approach in order to deduce the same deviation.

8.2.2 Object Grammar $\subseteq$ Action Grammar?

Rule O1 Compare to rule A1.

Rule O2 For each object we create an action that starts at the start of the systems and finishes at the end of the system. Thus this action always runs while the system runs. The object is the patient of the system. So using rule A4, we can express the HAZOP deviations resulting from MORE THAN, LESS THAN and OTHER THAN and PART OF. We can express *objects EXISTS* using rule A5.

Rule O3 As long as the numeric attribute of rule O3 is a timing attribute, rule O3 can be expressed using rule A5. In any other case we can invent a new object for the numeric attribute and create an infinity action as in rule O2. Than we make use of rule A3.

Rule O4 Similar to above we invent a new object for the descriptive attribute with an infinity action and apply rule A4.

Rule O5 Similar to rule O4.

Rule O6 As long as the numeric attribute is a timing attribute we use rule A8 to express rule O6. For any other case we define a new object with an infinity action for the numeric attribute, but store the numeric value in one of the timing parameters and use rule A8.

Rule O7 Here we use the same approach as taken is rule O4. Additional we create a list of preconditions, i. e. two preconditions for each object. One of these preconditions is that the object is EQUAL to the descriptive attribute of another object and the other one that it is different (i. e. OTHER THAN) the descriptive attribute of another object. Finally we can express rule O7 using rule A2.

Rule O8 Similar to rule O7.

Conclusion We can also say that the object grammar is a subset of the action grammar. Since we previously stated that the action grammar is a subset of the object grammar, it means that the cardinality of these sets is the same, therefore, action grammar = object grammar.

8.3 Weaknesses of Both Grammars

The weakness of the object grammar is its indefiniteness regarding the use of attributes. One can certainly define each and everything as an attribute or even an object. Consider, for example, a seat in a train. Without considering the actual application of this seat, one can identify a lot of attributes for different lengths, colors and materials. Furthermore one can identify subobjects at different levels starting from the arm rest, going over to single screws and even to single atoms. One does not generally know what level of detail should be applied. We can easily define some non relevant attributes for safety such as *number of elephants that can sit on the chair* or *number of squares on covering*. Thus the model is far to general to be used for a HAZOP.

On the other hand the conditions of the action grammar can contain everything one could ever want. However, it is rather impossible to limit this possibility. Furthermore, also this model permits the creation of all possible objects one could think of. Also meaningless actions can be created.

Nevertheless for both grammars it is sometimes hard to tell whether an object/attribute/action is really necessary or not.

9 Changes in the HAZOP Procedure

The structured, automated question generation results in some changes to the standard HAZOP method. The most important one being that the actual HAZOP meeting becomes dispensable, since brainstorming for possible deviations and their interpretation is already done automatically.

We illustrate the new approach in Figure 9.1. The assessor, or HAZOP team leader, generates deviations. These deviations are subsequently handed over to the HAZOP team members, that analyse them independent of each other. The assessor task is to evaluate the results, i. e. compare the risk level assigned to each and every deviation by each and every participant. Finally the assessor calculates the overall risk level of the assessed system.

Of course the question arises how the assessor creates deviations?

The assessor first identifies objects and actions in the specification, as usual. However, it is done in a more structural approach. Besides just identifying the actions as such, the assessor also has to give their internal and external structure. It is wise to start with the conditions, since they also determine what objects are needed internally.

Afterwards the assessor assigns the necessary objects (patients, instruments, sources/ destination) to the identified actions. At this state it can be helpful to make use of a table, structured like the one shown in Table 9.1. This table makes it easier to check whether one has identified all the objects needed for a particular action.

The next step is to define the time parameters. They can be absolute as well as relative. Most important are the start parameter S and the end parameter E, because they deepen the understanding of the system's flow. For instance if one action a_1 is supposed to happen before another action a_2, there is no need to ask what happens if a_1 happens BEFORE a_2, because this is the design intent.

In order to check the plausibility of the start and end parameters we suggest to make use of another table such as the one given in Table 9.2. This table can be very useful to identify causalities between actions, which is the next step.

After thus having defined all parameters of all actions the assessor runs a program that calculates the level of similarity between all actions, using the formula given in Equation 7.8. Based on the results of this calculation the assessor has to choose a minimum similarity level for the pairwise comparison of actions.

Table 9.1: *object - action matrix: the dots are used to illustrate that this action has something to do with this object*

	object 1	object 2	object 3	...	object n
action 1	•		•		•
action 2		•	•		•
action 3			•		
action 4	•	•			
action 5		•			
...					
action n			•		•

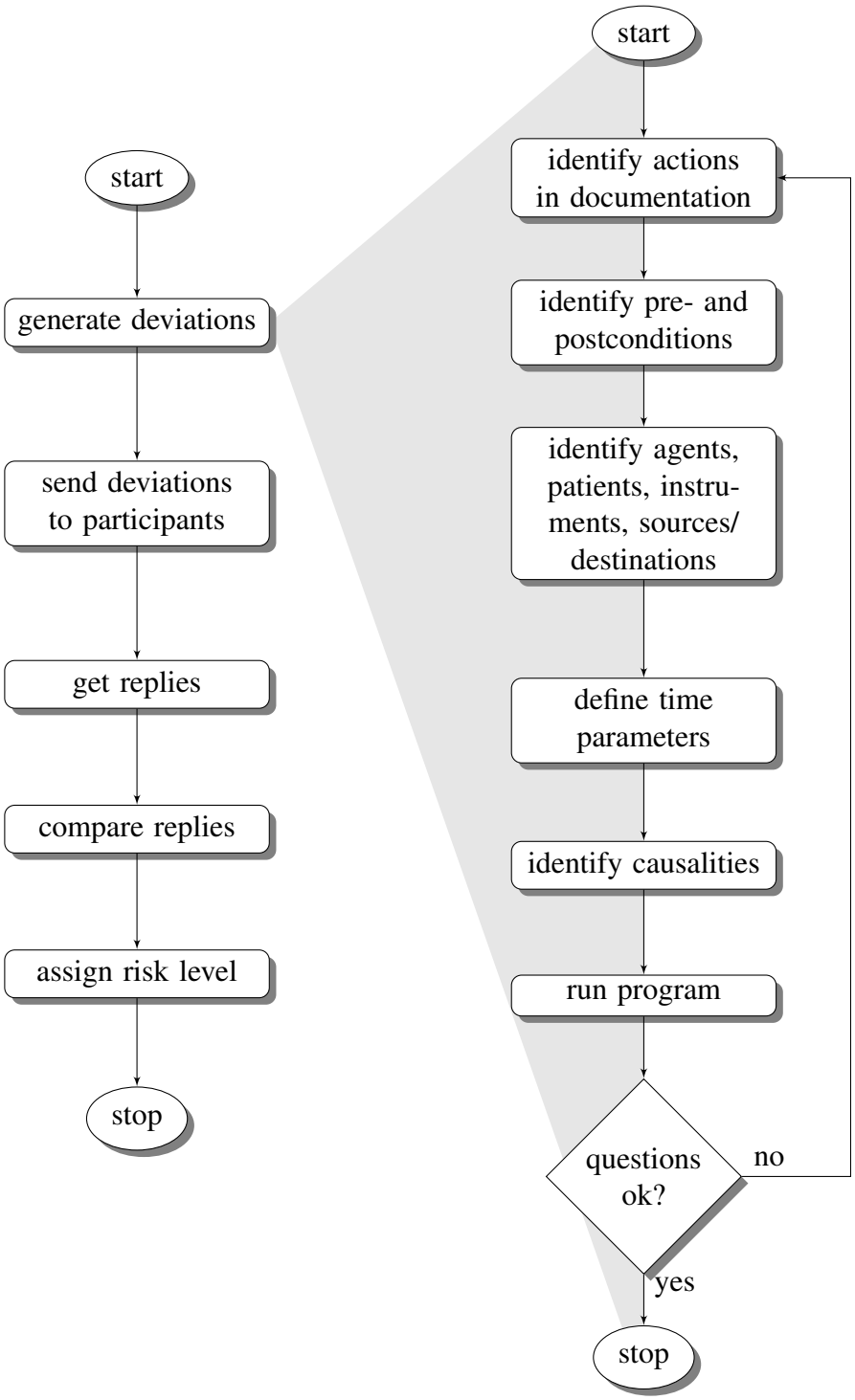

Figure 9.1: *Overview of the new HAZOP procedure. The main difference is the omission of the HAZOP meeting*

Finally the assessor can check the questions created by our automated system. In case some of the questions seem to be ill-conditioned or missing the assessor might need to redefine some parts on the actions.

Now after reading through all questions and maybe eliminating some obvious ones, the assessor sends these questions to the HAZOP team and evaluates the replies.

	S	E
action 1	1	5
action 2	7	10
action 3	12	13
action 4	12	15
action 5	18	20
...		
action n	19	30

Table 9.2: *time - action matrix: the table lists the start and end time of each action, to assist the assessor in identifying causalities*

9.1 Some Common Actions to Consider

What actions should an railway assessor consider when scanning the documentation? Empirical tests have shown that the following actions normally occur when dealing with software:

- When there is any kind of message transfer, there will be at least one sending action and at least one receiving action (send/receive in Table 9.3).
- In case some devices are in standby redundancy an action to awaken them is needed (change role).
- Also devices generally can be deactivated/destroyed or activated/created.
- Furthermore, devices could load software or other configurations (load).
- Of course also storing is needed (store).
- Additionally displaying information to the user is also an often occurring task (inform).

On the other hand, an analysis only based on software, leaving the mechanical interactions aside would be rather incomplete. Compared to software these systems are also able to move, to apply force or to recognize something. We identified these basic actions:

Table 9.3: *Standard actions to consider in software, the bullets indicate that it is usually necessary to specify this content object*

Action	Agent	Patient	Instrument	Source/Destination
send	•	•	•	•
receive	•	•	•	•
change role	•	•		
activate	•	•		
deactivate	•	•		
load	•	•		
store	•	•		•
inform	•	•		

transfer means to change the location of the object itself
push requires to apply force to some object
pull opposite of push
move describes moving of a part of the object itself
grasp means getting hold of some other object, which has to move with the grasping object in consequence
release opposite of grasp
consume represents the transfer of something from outside of the object to inside
emit represents the transfer of something from inside of the object to outside
sense denotes to identify something with some kind of sense organ

This basic actions for a train are exemplified in Table 9.4.

Of course not all of these actions are needed in general. For instance a door does not change its physical location, it just moves some parts of it, the frame remains in place.

All of these actions have their own interpretation. One could e. g. split *transfer* into acceleration, braking, or uniform/circular/chaotic motion. However, the applicability of these forms of transfer depends on the subject that transfers itself. For instance, a train normally jerks, accelerates, drives and brakes.

Our analysis has been greatly influenced by [25], who defined a set of primitive actions for animals.

9.2 Actions Lacking Agents

There are certain actions that do not possess any agent, such as *it rains*. At least *it* can hardly be seen as a serious object.

How should the assessor deal with these? There are two possible ways. First one could search for synonyms, e. g. *rain falls* is equivalent to it rains. We now certainly have an agent called *rain*, that has a transfer action *falling*. Secondly, one could ask *Who or what does the action[rain/snow] most likely ?* Thus creating some kind of dummy agent. As mentioned earlier rain falls out of cloud and, therefore, cloud is a likely agent.

Table 9.4: *Standard actions to consider in mechatronical systems, the bullets indicate that it is usually necessary to specify this content object, those actions defined in 9.3 are needed as well*

Action	Agent	Patient	Instrument	Source/ Destination	Example
transfer	•			•	train drives from A to B
push	•	•			train pushes the bumper
pull	•	•			locomotive pulls waggons
move	•	•			train turns wheels
grasp	•	•			train grasps overhead line
release	•	•			train releases overhead line
consume	•	•			train consumes power
emit	•	•			train emits passengers
sense	•	•	•		train senses animal using a camera

Actions that lack agents only seem to appear in relation to weather. However, weather plays an important role in the railway industries:

- Heavy snowfalls cases heavy delays or even cancellation of trains [1].
- Heat caused air conditioning systems to malfunction [2].
- Wind may lead to derailment [3].

Identifying the action *it rains* does not necessarily imply *it snows*. However, defining the agent cloud or rain instead of *it*, allows us to assigning attributes like temperature, speed or density. These attributes make it easier to identify further weather conditions apart from rain. E. g. *low temperature* may lead to ice on tracks, *high density* could case fog.

[1] Snow and ice to slow train travel in Germany during peak travel weekend: http://www.dw-world.de/dw/article/0 5053647,00.html

[2] Deutsche Bahn blames heat nightmare on climate change: http://www.dw-world.de/dw/article/0 5810356,00.html

[3] Wind derails freight waggon: http://www.mainpost.de/lokales/franken/Windboee-hebt-Gueterwagen-aus-dem-Gleis;art1727,5705404

10 Evaluation and Discussion

We now want to evaluate our previously designed model on a real system — the surveillance and diagnosis of suspensions system (SDS). This evaluation is made by comparing the deviations found by humans applying HAZOP to those found by the here described model.

10.1 The Surveillance and Diagnosis of Suspensions System

Normally the running gear goes to routine maintenance after a specific amount of kilometers driven. However, DB plans to send it to maintenance on demand, that is in case some component needs special treatment because of malfunction.

Therefore, the company plans to introduce a system for surveillance and diagnosis of suspensions. This system is supposed to monitor all parts of the running gear of a train, e. g. the wheelsets and bogies and to check whether these components are still fully functional. In case the system detects any deviations it generates an error message.

To facilitate this, sensors are mounted on the running gear to measure acceleration and temperature. These sensors forward their measurements to the surveillance device, that compares these to some safe values. In case an error is detected this error is forwarded to the train driver console, and saved in a database for the maintenance staff. The driver might get a notice telling him to drive at a certain speed or to stop the train, while the maintenance staff receives more detailed information about what needs to be repaired. This is sent during certain intervals to the maintenance staff via remote transmission.

10.2 The Bus System

The train ICE 3 uses TCN (Train Communication Network) as its communication network. Its predecessor ICE 1 and ICE 2 use a slightly different communication network. TCN was introduced to standardize the communication interface for the various electronic devices connected in a train, as well as to simplify coupling of international trains. [26]

The TCN consist of two buses, the MVB (Multifunction Vehicle Bus) and the WTB (Wire Train Bus). The MVB is used to interconnect the electronic devices in the not divisible parts of the train. In case of the ICE 3 the non-divisible part consists of the motor coach and three waggons.

The WTB on the other hand connects all vehicles of the train, which enables forwarding commands from or to the driver.

Both buses use the master - slave architecture, to limit access to the network. It works like this: The master polls its assigned slaves sporadically by sending a *master frame*. The addressed slave can now respond with a *slave frame* within a specified time period.[27] All other slaves on the same bus can of course also read these data transmissions. Both buses are redundant in ICE 3. The WTB is guarded by a gateway, to limit communication between the WTB and the MVBs. This gateway has a storage for messages from both buses it connects (MVB and WTB).

MVB Each MVB in the ICE 3 train runs through four different coaches, which represent the indivisible part. One waggon hosts the central units — the master of the MVB bus. The central units are in standby redundancy.

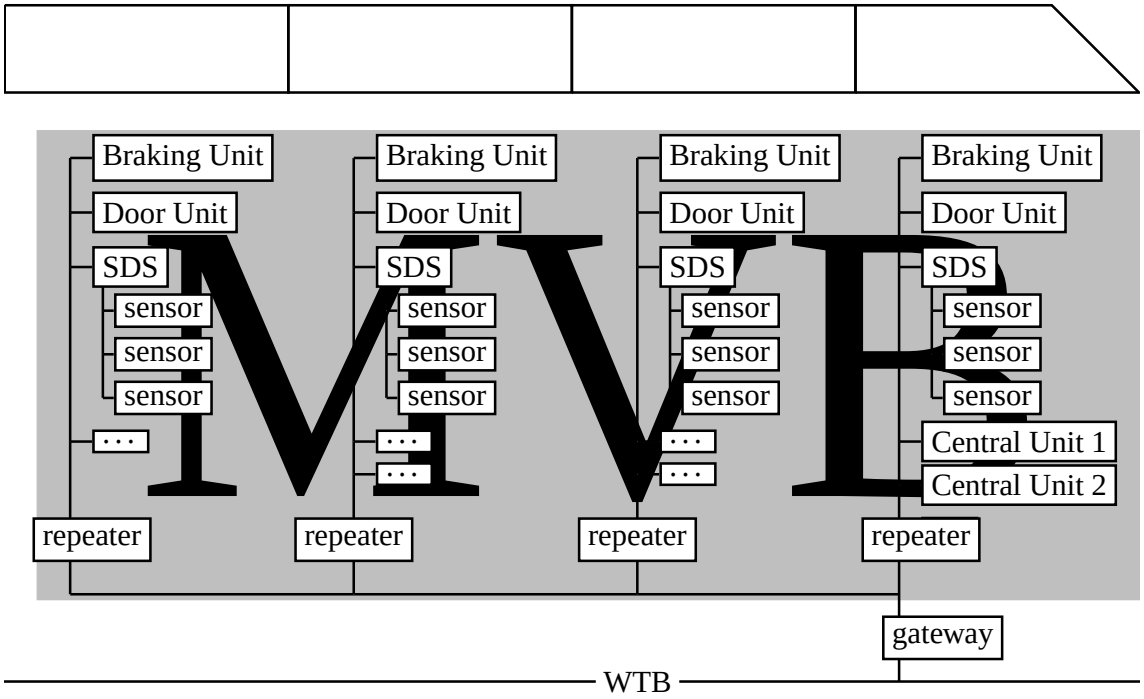

Figure 10.1: *Multifunction Vehicle Bus*

All of these waggons contain:

- a door unit
- a braking unit
- a traction unit
- a climate unit

Some other units for power supply, man-machine interface etc. also exist, but not in all waggons. All these components share a single dual-redundant bus.

The ICE 3 MVB architecture utilizes repeaters, to overcome the 200 m maximum transmission distance of the MVB. Figure 10.1 gives a schematic representation of the MVB.

WTB Figure 10.2 shows the WTB and its assigned MVBs. The WTB dynamically reconfigures itself when a new coach is coupled, while the MVB has a static configuration. This dynamic reconfiguration allows for faster coupling since no further information has to be given to the bus administrator. Its primary use is to forward messages from and to the train driver, as well as to the passengers.

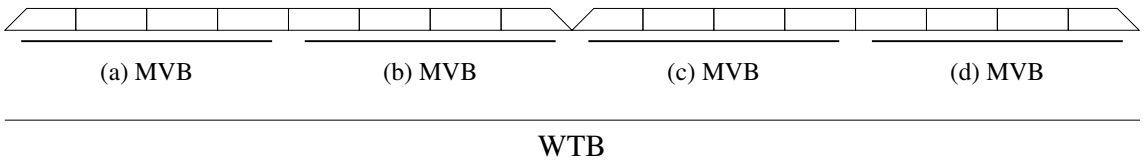

Figure 10.2: *Wire Train Bus*

SDS on the buses As illustrated in Figure 10.1 SDS (Surveillance and Diagnosis of Suspensions System) sits in each of the four waggons of an MVB. Each SDS is responsible for its own waggon. Each SDS system receives data from its temperature and acceleration sensors and checks whether these values are in the allowed range. When polled by the central unit of the MVB the addressed SDS replies with the found deviations. In case there is no deviation it still replies with a slave frame, thus to show that it is still functioning. The information display collects all data send by the SDS systems, and forwards them to the WTB when polled.

Based on the severity on the fault the driver might get the recommendation to drive at a certain speed, to stop the train, but also he might not be informed at all. In any case the fault is stored in a database for maintenance and transmitted remotely at certain intervals. SDS also reads data from the bus, thus in order to get information about speed, time or the train number. It is possible to disable SDS in a waggon. [28]

SDS is put to service in three integration phases:

phase 1:
- sensors and SDS integrated in the waggons
- SDS is fully functional
- deviations sent not forwarded to the driver
- not used for maintenance
- stored in some database for testing purpose

phase 2:
- gives driving recommendations for driver
- not primarily used for maintenance, but available

phase 3:
- fully used for maintenance
- substitution for regular (driven kilometers based) maintenance system

10.3 What is to be assessed?

The goal of the assessment is to evaluate whether there might be any safety issues in integrating the SDS system into the train ICE 3. Since the SDS is attached to the MVB it has primarily to be checked whether this might interfere with any other devices on that bus.

The system's risk has been assessed in a HAZOP previously.[29] They limited their evaluation mostly to the interaction of SDS and the MVB. However, also some deviations that occur in the driver's cabin or on the WTB are listed in their risk assessment. So we decided to do a HAZOP on all objects from the SDS device to the driver and maintenance staff. Thus only excluding the communication between SDS and its assigned sensors.

10.4 Deviations

The human assessor based their entire analysis on the same action: *SDS transmits data to MVB*. Nevertheless, they did not consider:

- SDS is also able to read from the bus (e. g. speed)
- the central unit needs to load software to support SDS
- SDS can be deactivated by the central unit

In connection with these missing actions they also omitted certain objects, namely:

- central unit
- the button to deactivate an SDS
- the WTB

In total the standard method found ≈ 40 deviation, while our system could identify approximately 300 deviations. The additional deviation found by us where:

SDS has problems reading from the MVB The SDS reads for instance the current speed from the MVB bus. If this speed is not interpreted correctly, SDS could send wrong information to the train driver. One possibility is that SDS creates wrong error messages, because of wrong speed information. From phase 2 on it might recommend to unnecessarily drive at a lower speed, thus prolonging the journey. This is, however, not a hazard. On the other hand it might not announce an error although there is one, thus leaving the passengers in a significant danger.

SDS is not deactivated properly The train driver can deactivate each SDS in the train, if it is assumed to malfunction. However, if this deactivation fails, the driver will continuously be disturbed by this SDS device. So the driver could miss some important other messages.

SDS is deactivated unintentionally In case a SDS unit is deactivated unintentionally, it will not send any error messages any more. Thus the driver does not get any recommendation about the speed from this SDS unit.

SDS sends wrong error messages SDS might send error messages although there is no error present. Although there is no risk for human lives, it is a not desired behaviour.

Software button not set In case of a software update of the central unit, the software has to be informed that SDS is present. If this is not done SDS is not polled. The train driver might believe SDS is running, and thus not perform some regular checks as done before SDS was available. So in case some error occurs it will neither be detected by the driver, nor by SDS.

Besides giving additional deviations, our system somehow failed to identify one deviation that the human assessors identified:

inconsistent messages SDS might advice the driver to stop the train, but in a later message advice to drive at 30 km/h. First of all the driver could miss the prior message to stop the train and thus expose passengers to a significant danger. Secondly the driver might deliberately choose to drive at 30 km/h, which leads to the same hazard.

Neither our model nor DB's current model is able to identify this deviation, since it deals with the guide word REVERSE. As of now, this guide word has not been included into the analysis.

10.5 Performance

We now need to evaluate the two HAZOP approaches, namely the automated, structured one presented in this paper and the standard one, where human define the deviations.

The most important aspects that need to be considered regarding performance are time and completeness. Saving time directly results in saving costs. Comparing the completeness of both approaches is necessary to check that the automated approach has at least the same level of completeness as the standard approach.

Comparing the here presented system to the previous one done by Tschachtli[15] is not necessary since we formally proved that they are equivalent.

10.5.1 Time Aspect

Normally, in the standard HAZOP approach, an assessor (or team leader) needs two person days for scanning the systems documentation and thinking about possible HAZOP deviations [30]. This time will be roughly the same for an automated approach, after some time of training to get used to the model.

The actual HAZOP meeting usually takes one person day for each participant, plus additional travelling time. Since this meeting does not take place in the automated HAZOP this time is saved entirely.

However, the additional factor of answering the automatically created questions is needed. We assume that this only takes two person hours per participant, since the questions are rather clear and the assessor has pre scanned the questions. Moreover, in the automated approach the assessor needs to compare the answers, which will take an additional person-day. On the other hand the standard approach may need some clarification after the meeting. The time needed to prepare the report is the same for both approaches.

All in all the standard HAZOP takes about 180 person-hours in total considering each participant, while the automated approach takes only 140 person-hours. Thus the automated approach saves a person-week for an average risk assessment. Table 10.1 summarizes these findings.

Table 10.1: *Comparing Performance: unit is person-hours (ph), there are 6 participants in the HAZOP team, the assessor is the leader of this team, h denotes hours, p denotes persons*

		person-hours required for:	
		Standard HAZOP	Automated HAZOP
preparation	assessor	16 h · 1 p	16 h · 1 p
	other participants	8 h · 5 p	8 h · 5 p
meeting	HAZOP meeting	10 h · 6 p	—
follow-ups	clarification	4 h · 6 p	—
	answering questions	—	2 h · 6 p
	comparing answers	—	8 h · 1 p
report		8 h · 6 p	8 h · 6 p
total		180ph	142ph

10.5.2 Completeness Aspect

The automated system gives a more complete list of possible deviations, since it takes an analytic approach, rather than just combining components with guide words as done traditionally. However, some important actions could have been forgotten, so possible risks cannot be identified. Also only single elements of an action could be missing, so also risks resulting from this cannot be found. Moreover, all these definitions depend

on a single person, namely the assessor, who is the only one responsible for creating deviations.

The group study has the advantage that more than one person has influence on the deviations. So aspects that have not been considered by the group leader could be identified by the other participants. Nevertheless, also here deviations can be overlooked. Even worse, there are far less possibilities to check completeness, than with a more structured approach.

So neither the automated approach nor the standard approach give a guarantee for completeness of the risk assessment. However, the assessment is only needed to assign a preliminary risk level. Based on this risk level further risk analysis methods will be needed. So actually there is no need of completeness, since finding one deviation that leads to a higher risk level is enough to require further more detailed analysis.

In anyway the risk analysis depends on documentation that does not necessarily need to be complete or fully trustworthy. However, having more deviations gives input for further analysis, so one might save time later on if the preliminary risk assessment is more complete.

11 Conclusion

In this thesis we presented a model for actions. This model gives a structured approach to conquer the problem of finding deviations in a HAZOP.

The tedious task of interpreting deviations has been simplified, because the systematically created deviations are less bound to interpretations. So HAZOP participant spend far less time in interpreting guide word – component combinations, since they are less ambiguous. Therefore, the team members can concentrate all their efforts on the identification of risks.

Most of the limitations of HAZOP have been dealt with. Our model enforces completeness, since conditions, involved objects as well as causal relations have to be specified. We eliminated redundancies to a vast extent, on the one hand by finding similarities in deviations, but also by requiring further parameters for objects. Our model reduces ambiguities since, in the need to obey the structure, very clear deviations are created, that give no rise to any interpretation. By analysing each and every guide word in its applicability to each property of an action, we diminished the amount of absurd HAZOP expressions. In order to limit the number of generated pairwise deviations, we give a similarity measure based on the number of similar objects.

Practical evaluation has shown that the structured automated approach is able to find more deviations than the standard approach. However, it did not led to a higher risk level, since it was already the second highest possible.

All in all it can be said that an automated, structured approach is more efficient than a standard HAZOP, although it requires some training on the participants site.

11.1 Future Work

Although much has been done there is, still, the need for some additional work. One the one hand this includes further testing and on the other hand improving the deviation creation program to simplify usage. Furthermore it might be wise to standardise the requirement specification.

11.1.1 Real World Test

The system has not been tested in a production environment. That means the generated question list has not been send out to any HAZOP participants, and thus no replies could be evaluated and checked whether it really saves time. This test is, however, crucial in order to check whether the standard method and the automated one gain the same results.

This could be achieved in the following manner: One assembles one team to do a standard HAZOP on some system and another independent team to do an automated HAZOP of the same system. Finally we need to check whether they identified the same deviations and how much time it took.

11.1.2 Program

The current program used to create the deviations is a prototype. The assessor has to define objects and actions in XML and run the program that calculates the levels on the command line.

The only output the assessor gets is a list of questions in HTML format. This format is not suitable for conducting a HAZOP since the assessor needs to copy it to a spreadsheet and add some more columns that are needed for the risk assessment.

A more user friendly program would support the assessor with a graphical interface to define the actions as such and then create a spreadsheet document to be filled out. It could of course also directly send the list of deviations to the participants. Furthermore, it is desirable if the program would also collect the replies scan through them and perform some kind of pre-sorting.

11.1.3 Standardise Requirement Specification

Nowadays, each company applies its own methods for documentation of requirements and further documents in the system's life cycle. They may even look different for two different systems of the same company.

This creates a problem for the assessor, since for each systems he has to get used to a different style of documentation. First of all, this takes time, since the assessor may need to scan through several documents to find the information wanted. Secondly, it is easy to forget some specific point, since it is hidden in the text. It is, therefore, worth considering to standardise the entire documentation since this would diminish the workload on the assessors side and shorten the time for development.

The developers could e. g. already fill the XML file with the required entries. That means to give a list of actions and their pre and postcondition as well as the objects they interact with and their causal and timing relations. For each action the developers could, furthermore, give a description in a specific section in the documentation. The assessors task is then simply to check this list and the documentation for correctness and completeness and to create the deviations and continue as usual. In that way he does not need to create the list of actions in the first place.

References

[1] Hulin B, Schulze T. Failure analysis of software for displaying safety relevant information. In: Brís R, Soares CG, Martorell S, editors. Reliability, Risk and Safety: Theory and Applications. vol. 2. London: Taylor & Francis Group; 2010. p. 1327 – 1331.

[2] IEC EN 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems; 1999.

[3] BS IEC 61882:2001: Hazard and operability studies (HAZOP studies) - Application guide. London, UK; 2001.

[4] Lano K, Clark D, Androutsopoulos K. Safety and Security Analysis of Object-Oriented Models. In: SAFECOMP '02: Proceedings of the 21st International Conference on Computer Safety, Reliability and Security. London, UK: Springer-Verlag; 2002. p. 82–93. Available from: http://www.springerlink.com/content/xme0kun7kpkfd7ar/fulltext.pdf.

[5] Khan FI. Knowledge-based expert system framework to conduct offshore process HAZOP study. In: IEEE International Conference on Systems, Man and Cybernetics. vol. 3; 2005. p. 2274 – 2280. Available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=\&arnumber=1571487.

[6] Redmill F, Chudleigh M, Catmur J. System Safety: HAZOP and Software HAZOP. John Wiley & Sons Ltd; 1999.

[7] Hoepffner L. Analysis of the HAZOP study and comparison with similar safety analysis systems. Gas Separation & Purification. 1989;3(3):148 – 151. Available from: http://www.sciencedirect.com/science/article/B6TG4-43RTHXB-85/2/c9755d5f06329c5d0e3880913cd12e96.

[8] Macdonald D, Mackay S. Hazop method. In: Macdonald D, Mackay S, editors. Practical Hazops, Trips and Alarms. Oxford: Newnes; 2004. p. 97 – 129. Available from: http://www.sciencedirect.com/science/article/B875S-4P9TJ05-6/2/5a92c234b6f8eb317403ff73c06b77d5.

[9] Defence Standard 00-58; 2000. Cancelled, Superseded by Defence Standard 00-56.

[10] Zhao C, Bhushan M, Venkatasubramanian V. PHASuite: An Automated HAZOP Analysis Tool for Chemical Processes: Part I: Knowledge Engineering Framework. Process Safety and Environmental Protection. 2005;83(6):509 – 532. Available from: http://www.sciencedirect.com/science/article/B8JGG-4RSJN4T-5/2/67a0262e6648fed1cdaab2b781d2e133.

[11] Dyadem. PHA Software / HAZOP Software; 2009 [Retrieved: 2010-04-09]. Available from: http://www.dyadem.com/products/phapro/.

[12] Palmer C, Chung PWH. An automated system for batch hazard and operability studies. Reliability Engineering & System Safety. 2009;94(6):1095 –

1106. Available from: http://www.sciencedirect.com/science/article/B6V4T-4VBMNF1-2/2/19d3031e87d6c7d657fc25ccd26a94e4.

[13] McCoy SA, Wakeman SJ, Larkin FD, Jefferson ML, Chung PWH, Rushton AG, et al. HAZID, A Computer Aid for Hazard Identification: 1. The Stophaz Package and the Hazid Code: An Overview, the Issues and the Structure. Process Safety and Environmental Protection. 1999;77(6):317 – 327. Available from: http://www.sciencedirect.com/science/article/B8JGG-4RSJN4J-2/2/ba86e1f38ec9032d4d8c8265fc98b5f9.

[14] McCoy SA, Wakeman SJ, Larkin FD, Chung PWH, Rushton AG, Lees FP. Hazid, A Computer Aid for Hazard Identification: 4. Learning Set, Main Study System, Output Quality and Validation Trials. Process Safety and Environmental Protection. 2000;78(2):91 – 119. Available from: http://www.sciencedirect.com/science/article/B8JGG-4RSJN4K-3/2/5430fcbe515fd1bf24336d3aa0024edb.

[15] Tschachtli R. Entwurf einer Methode zur Identifikation von Fehlermöglichkeiten bei der Entwicklung von Softwarekomponenten. Fachhochschule Bingen; 2009.

[16] Zhang Y. Specifying causality in action theories: a default logic approach. Theoretical Computer Science. 1999;220(2):489 – 513. Available from: http://www.sciencedirect.com/science/article/B6V1G-3X52GNT-8/2/5d47c55ab58141ab3fc8f4c51dcf8044.

[17] Burns A, Wellings A. Real-Time Systems and Programming Languages. Addison-Wesley; 2009.

[18] Denger C, Trapp M, Liggesmeyer P. SafeSpection — A Systematic Customization Approach for Software Hazard Identification. In: SAFECOMP '08: Proceedings of the 27th international conference on Computer Safety, Reliability, and Security. Berlin, Heidelberg: Springer-Verlag; 2008. p. 44–57. Available from: http://dx.doi.org/10.1007/978-3-540-87698-4_7.

[19] Palmer C, Chung PWH. An automated system for batch hazard and operability studies. Reliability Engineering & System Safety. 2009;94(6):1095 – 1106. Available from: http://www.sciencedirect.com/science/article/B6V4T-4VBMNF1-2/2/19d3031e87d6c7d657fc25ccd26a94e4.

[20] Rickheit G, Sichelschmidt L. Valency and cognition - a notion in transition. In: Herbst T, Götz-Votteler K, editors. Trends in Linguistics, Studies and Monographs: Valency: Theoretical, Descriptive and Cognitive Issues. Walter de Gruyter GmbH & Co. KG; 2008. Available from: http://site.ebrary.com/lib/chalmers/Doc?id=10256673\&ppg=175.

[21] Luger GF. Artificial Intelligence: Structures and Strategies for Complex Problem Solving. Addison-Wesley; 2005.

[22] Zuberi KM, Shin KG. A causal message ordering scheme for distributed embedded real-time systems. In: SRDS '96: Proceedings of the 15th Symposium on Reliable Distributed Systems. Washington, DC, USA: IEEE Computer Society; 1996. p. 210. Available from: http://portal.acm.org/citation.cfm?id=830880#.

[23] Avizienis A, Laprie JC, Randell B, Landwehr C. Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing. 2004 jan-march;1(1):11 – 33. Available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=\&arnumber=1335465.

[24] Yin Y, Yasuda K. Similarity coefficient methods applied to the cell formation problem: A taxonomy and review. International Journal of Production Economics. 2006;101(2):329 – 352. Available from: http://www.sciencedirect.com/science/article/B6VF8-4FSK7S3-2/2/a84370d7988fb41297291f6c8c77c377.

[25] Schank RC, Rieger CJ. Inference and the computer understanding of natural language. Artificial Intelligence. 1974;5(4):373 – 412. Available from: http://www.sciencedirect.com/science/article/B6TYF-4808W0K-2S/2/2ee1f29018786eccd746c84561c7908b.

[26] Iturbe X, Jimenez J, Zuloaga A, Lazaro J, Martin JL. The Train Communication Network: Standardization goes aboard. In: 2010 IEEE International Conference on Industrial Technology (ICIT); 2010. p. 1667 –1672. Available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=\&arnumber=5472587.

[27] Jimenez J, Martin JL, Zuloaga A, Bidarte U, Arias J. Comparison of two designs for the multifunction vehicle bus. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on. 2006 may;25(5):797 – 805. Available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=\&arnumber=1624514.

[28] Knitter. ÜDF Einbindung in die Leittechnik: Anforderungsspezifikation. Siemens; 2010.

[29] Ruf N. Risikobetrachtung ÜDF. Basler & Hofmann - Ingenieure und Planer AG; 2010.

[30] Hulin B. Length of a HAZOP for Software; 2010. private conversation.