Using Qualitative Models for Safety Analysis of Industrial Automation Systems

Uwe Biegert

University of Stuttgart
Institute of Industrial Automation and Software Engineering
Pfaffenwaldring 47, D-70550 Stuttgart, Germany
phone: +49-711-6857293, fax: +49-711-6857302
biegert@ias.uni-stuttgart.de

Abstract

Nowadays software enables to control more complex processes, but at the same time it is also responsible for the welfare of humans and environment. A failure in a software program can influence the technical process with unforeseeable consequences. Generally the safety of a computer-controlled system depends on a complex interaction between technical process, controller software and human task. Classic methods for safety analysis mostly are specialized to consider one part of the system and the analysis is a brainstorming procedure. In this paper a model-based approach for safety analysis is discussed. All parts of the computer controlled systems are first described with the help of a qualitative modeling. Then the different qualitative models are combined to a unique model of a computer-controlled system. Based on this model a computer supported safety analysis can be realized. The model enables the analysis of the interaction between the system parts even by considering any multiple failure.

Key Words

safety analysis, computer-controlled system, automation, qualitative modeling, SQMA, UML-RT

1 Introduction

The safety requirements for computer-controlled systems are defined more and more strictly, particularly then, if the health of human beings is immediately concerned. Such systems can be divided into three different parts: the technical process (which must be controlled), the control software (which contains the automation concept and functions to prevent serious accidents) and the human as operator, who normally supervises the whole system, see figure 1, [1]. In dependence of the degree of automation the role of these parts can differ, nevertheless the interaction between them is complex. Failure of the control software can influence the technical process with unforeseeable consequences. Defects of elements of the technical process can impair the regular behavior control software in similar way. Wrong human interventions affect the technical process directly or indirectly by the control software. Furthermore, the increasing complexity of computer-controlled systems makes it harder to overview the whole interaction.

This fact can be explained with an example of a modern airplane. While at the beginning of aeronautics the pilot was depending only on his instinctive perception, nowadays dozens of controllers prepare the necessary flight information for the pilot, show him the optimal route or warn of dangerous situations. The yaw damper steered by the computer evaluates the information a thousand times faster and therefore reacts upon unexpected events better than human beings. The interactions between airplane, control software and pilot must work perfect. Quite a number of airplane accidents already have shown, what happens if a sensor fails: the computer misinterprets the information and the pilot is no longer in a position to handle the situation correctly.

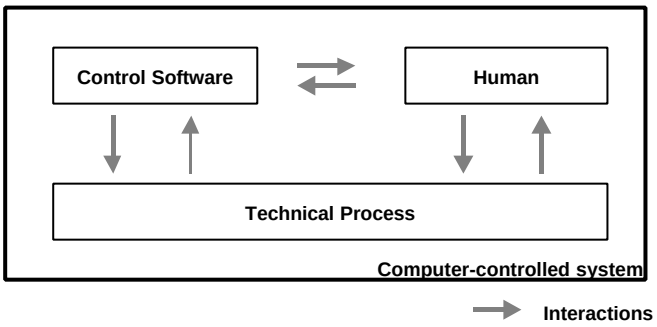

Figure 1: General structure of a computer-controlled system

2 Classic Safety Analysis

Safety analysis offers the potential to determine whether the risk associated with a computer-controlled system is acceptable in relation to the overall system risk, in context of defined failure assumptions. In general, classic methods for safety analysis are specialized to analyze one of the three mentioned system parts. Methods which are developed for considering the technical process, like FMEA (Failure Mode Effect Analysis), HAZOP (Hazard and Operability Analysis) or FTA (Fault Tree Analysis), are less effective for software models [2].

An other disadvantage of most classic methods is, that the principal step is a brainstorming of all kinds of possible events, actions, system situations and possible consequences of failures [6]. The analyst must be an expert and must know all the details of the system. Therefore the quality of the result depends on his knowledge and normally is hard to reproduce by an other person. Other problems of classic methods are proofing the completeness of the analysis and the evaluation of multiple failures.

For a new safety analysis approach it is important to consider the whole computer-controlled system, not only its parts separately. Furthermore in the age of computer science the brainstorming procedure of the classic methods should be supported by a software tool. Then most of the problems coming up with brainstorming can be solved. To make the system analyzable by a computer, an appropriate model of a computer-controlled system must be used. Because of the complex structure of such systems, the requirements for an appropriate model are very demanding and difficult to realize. This model should be able to describe the behavior of the technical process, the control software and the human task. It must deal with the complexity of the systems and it must be developed especially for the mean of safety analysis. This includes that possible states or situations of the computer-controlled system must be evaluated in respect to their potential risk and probability.

3 Model Based Safety Analysis

A research field at the Institute of Industrial Automation and Software Engineering (IAS) deals with model based safety analysis [3]. The computer shall support the user at the execution of the safety analysis. Based on the available system documents the technical process, the control software and the human task are first modeled separately, then joined together to a uniform model.

With the help of the uniform qualitative model the computer can investigate, whether dangerous situations can appear in the real system or not by considering the interactions of the three system parts. The effects of multiple failures to the complete system can be analyzed, too. The computer relieves the experts by calculating all possible situations of the computer-controlled system by combining the three different qualitative models.

The user must examine these situations, especially the critical ones. It's upon his knowledge to decide if these critical situations are probable or not. If so, additional safety functions (or a redesign) of the system are necessary. The situations also help to locate design failures of the control software, figure 2. In the following, each step of the model-based safety analysis will be discussed.

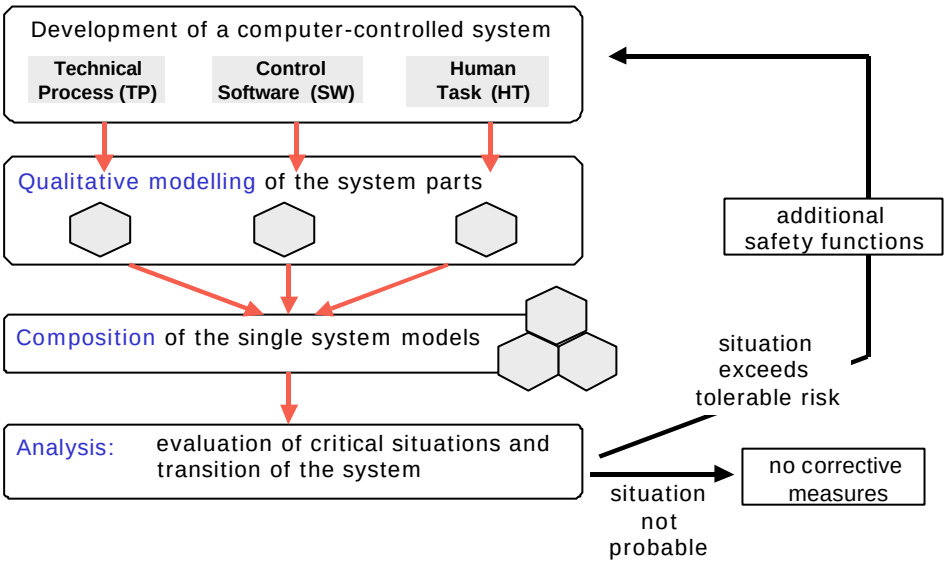

Figure 2: Model-based approach for safety analysis

4 Qualitative Modeling

By using qualitative models it is possible to describe complex systems even with uncompleted knowledge [4]. As modeling language we use the SQMA-approach (Situation-based Qualitative Modeling and Analysis). SQMA is a component-oriented modeling method [5]. First, all components of a system are described qualitative independent of their function in the system. The structure of the system represents the rules of the interactions between the components. Within this assumption a model of the whole computer-controlled system can be generated by the computer.

For modelling a system has to be divided into subsystems or at least into components. These components can be real (such as technical components) or fictive (such as software component). The first step is a black-box consideration of the defined component. All important terminals of the component are considered. The quantities on the terminals are expressed by using interval variables. Therefor the value of these quantities are spited up into ranges. Figure 3 shows a short example of a fluid level quantity. The chosen intervals are [0,0], (0, 100) and [100, ∞). SQMA offers the possibility to give the intervals a comment (or interpretation) e.g. the intervall [0,0] stands for *empty*.

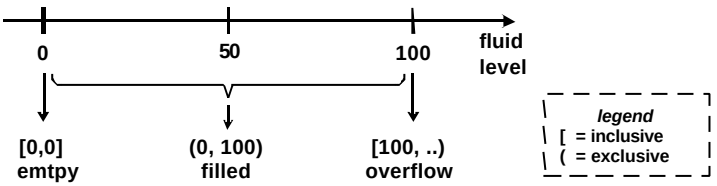

Figure 3: Qualitative variable based on intervals

After describing all quantities a white box consideration takes place. The interrelation of the quantities is a rule-based description (e.g. if there is a inflow into a tank, the fluid level must rise). An other very important use of a rule-based description is the classification of not-intended and dangerous situations (e.g. if the door of a tank is open and the fluid level is greater than 50 gallons then there is a outflow through the door). After the black and white-box consideration the computer is able to build up a so-called situation-table.

After having modeled each component, the connections between them have to be described by using a netlist. Finally an algorithm combines all situations of the component model. The algorithm considers questions like the following: Is it possible that, component “A” can be in situation “n” and at the same time component “B” is in situation “m”. For each combination, the quantities of a component have to be checked with the system equations. If the result of all system equations for a special combination of situations (system situation) is true, then this system situation can happen in reality. Then an even more important question must be answered: Does the system contain dangerous system situations? – When modeling the components we classified different kinds of situations with an attribute. With these attributes a computer can quickly separate the system situations into dangerous, not-intended or regular. Now an analyst doesn’t need to think about possible combinations. He only considers the result of the computer analysis. The analyst has to interpret the result himself by considering the likelihood of these system situations and whether prevention is necessary or not.

In the following section we illustrate our approach by an example.

5 Qualitative Modelling of the System Parts

Figure 4 shows a technical system, in which a homogenizing process is to be run. An inlet valve supplies a liquid into a tank with an integrated mixer. The liquid is to be homogenized and to be removed afterwards by the drain valve. The tank has a door, which has to be locked during the process cycle.

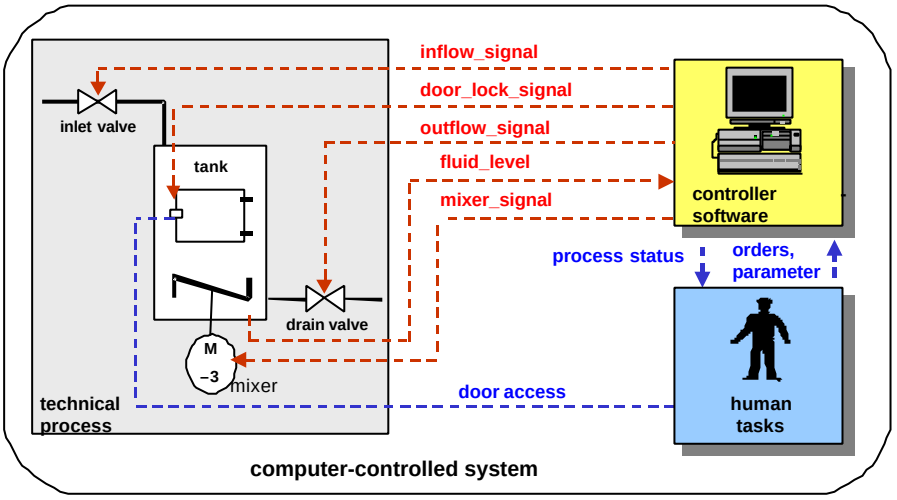

Figure 4: A computer-controlled system "homogenising plant"

The system was automated with an open-loop structure, therefore an operator has control functions like starting, terminating and interrupting the process. Furthermore he can modify the value for the level of the tank (desired value specification). The controller is receiving the requests from the operator, checking them and thereupon sending the suitable messages. Also, the controller must prevent safety critical states of the technical process. The controller was implemented in C++ .

Now, with the help of the technique of SQMA this system is modeled. It will demonstrate how complex the interaction between technical process, automation software and human control interventions is even for such a small system. First, all parts of the homogenizing plant are regarded and modeled individually. Subsequently, the three resulting models are joined to the unique model of the automation system. With the analysis of the unique model, predictions about the behavior and about the safety of the system can be made.

5.1 The Technical Process

The technical process represents the physical behavior of the system. The process is a typical continuous flow process. The technical system consists of two identically constructed electromagnetic valves and a tank with an integrated mixer. The technical system is usually described with PI[1]- and process flow diagrams, which serve as a base for the qualitative modeling of the technical process.

The qualitative model of the technical process contains 50 different situations, which can be understood as scenarios for different operations modes. The situations are classified according to their meaning for safety. Table 1 shows as an example for different situation of the technical process. The header of the table shows the single components of the technical process. The dangerous situation can be read as following: There is an inflow into the filled tank, because the inlet valve is open. The drain valve is close, so there is no outflow from the tank. The situation is classified as dangerous, because the mixer is running while the door of the tank is open.

inlet valve	tank	drain valve	status
..	..	..	..
closed	empty, mixer off, door closed	open	intended
open	contains fluid , inflow, mixer on, door open	closed	dangerous
..	..	..	..

Table 1. An extract of the situation table of the technical process.

5.2 The Control Software

For this system the controller software was developed with the method UML-RT. UML-RT stands for **U**nified **M**odeling **L**anguage - **R**eal **T**ime and supports the development process of software projects [8]. With UML-RT functions of the software are divided into components, the so-called capsules. These capsules can only communicate by messages on defined channels. Messages have a signal part and optionally a data part. For further information about UML-RT, see “Modeling Language Guide" [8].

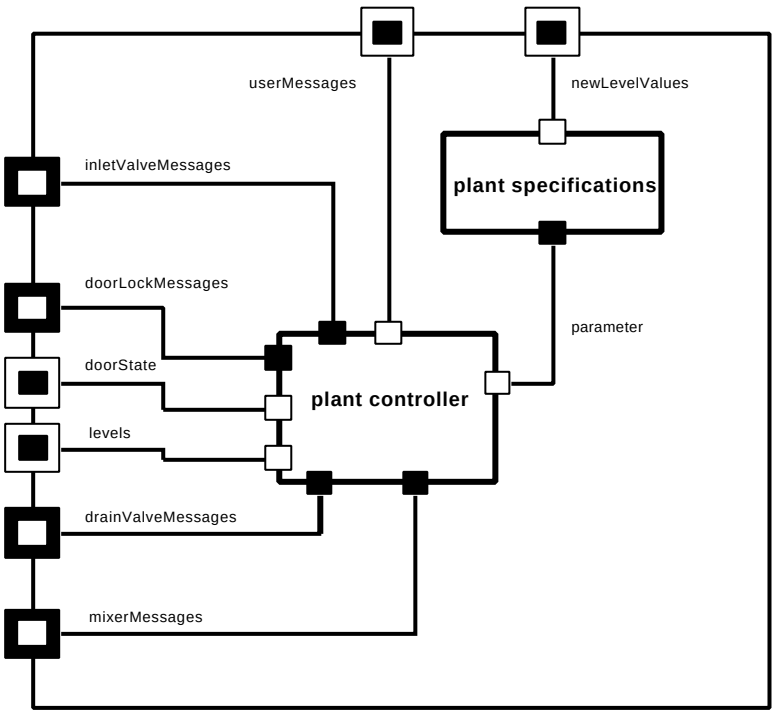

Figure 5: Structure model of the controller software

[1] PI: Piping and Instrument Diagram

Figure 5 shows the structure model of the controller software in the UML-RT notation. It contains the capsule *plant specifications* and the capsule *plant controller*. The capsule *plant controller* controls and monitors the desired process sequences, while the capsule *plant specifications* contains the data of the technical process e.g. the tank volume. It also realizes the dialog with the user. Furthermore, Figure 5 shows which process information and manipulated variables were selected for controlling the homogenizing process. The interface to the user (*userMessages*, *newLevels*) and to the technical process (*inletValveMessage*, *mixerMessage*,...) are represented in the structure model as communications ports (black-and-white rectangles)[2].

The behavior of the capsules is described with Statecharts. They are extended state machines, based on Harel's statecharts [9]. A transition is triggered by a certain event (signal). Optionally, further conditions for a transition can be specified, e.g. the data part of a messages can be checked for certain values. The statechart of the capsule *plant controller* is shown in Figure 6.

The definition of the states is oriented on the desired process states. The state *initialization*, like the name implies, is called within the initialization of the system. At this point the valves are closed, user inputs are enabled and the door to the tank is unlocked. If the user starts the system, the *plant controller* changes into the state *filling/mixing*. This state opens the inlet valve, starts the mixer, refuses further user inputs and locks the door of the tank. When reaching the desired level, the inlet valve is closed and the capsule *plant controller* changes into the status *mixing*.

The user can terminate the process at any time. In this case the state *draining* is reached, the mixer motor is stopped and the drain valve is opened. Finally when the tank is empty, the *plant controller* switches into the state *ready* and the process can be started again. The state *stopped* is reached by the controller when irregular events occur (e.g. abort by the user or the door is open when the tank is being filled). This safe state will warn the user and stop the process.

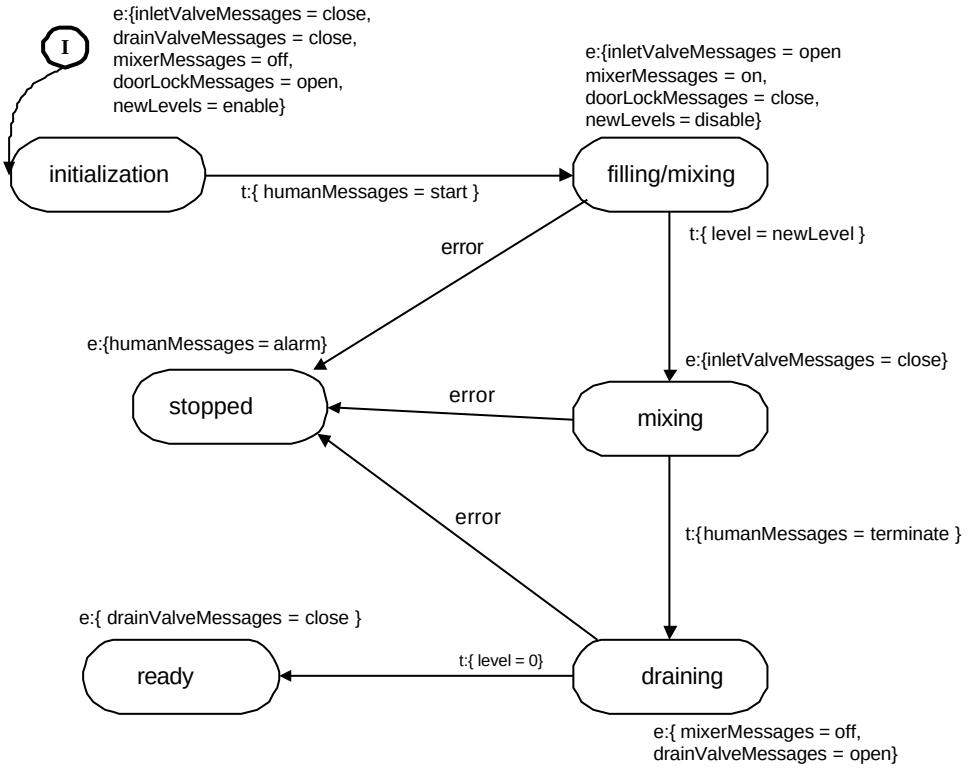

Figure 6. Statechart of the capsule *plant controller*

[2] The different patterns of the ports depends on the definition of the respective protocol and is insignificant for the understanding of this report

The function of the capsule *plant specifications* is to check the users inputs for their validity and to output suitable messages to the user. Its behavior is also described with a statechart. Its state are *ready, inputs_blocked, inputs_ok* and *inputs_illegal.*

After discussing the design of the controller software, we are going to translate the UML-RT-design into a qualitative model. The goal is to generate scenarios in form of situations for the behavior of the control software.

First the capsules *plant controller* and *plant specification* are described separately. Therefore the signal and data part of a message must be expressed by using interval variables. The values of the qualitative variables stand for the different contents of a messages. During the black box modeling, the complete situation space is regarded. The situations describe all possible combinations of the messages from a capsule. In contrast to the modeling of the technical process, there is no physical law for the behavior of the software which can be consulted for the formulation of the situation rules. With UML-RT the behavior of the capsules were specified by statecharts, which permit only predetermined combinations of the messages. A statecharts diagram can be expressed with the help of interval arithmetic, if the following points are observed:

- *the trigger condition* for a transition, i.e. the event which caused the state
- *the modification (actions),* which are caused by a state
- *the history* of the preceding states.

For this paper more detailed information of the transformation is of no importance. The white-box modeling step consists of the transformation of the state-transition diagram into situation rules. After considering these situation rules, only the situations which describe the behavior of the capsule remains. Due to its specification, 4096 different situations were combinatorial possible for the capsule *plant controller*. With the situations rules for this capsule only 108 situations remained. Similarly to the modeling of technical components, groups of situations can be commentated. Figure 7 shows the number of situations which are assigned to the individual states of the capsule *plant controller*. For example there are 24 situations which can be assigned to the state *filling/mixing*. We can interpret these situations as scenarios for the suitable states.

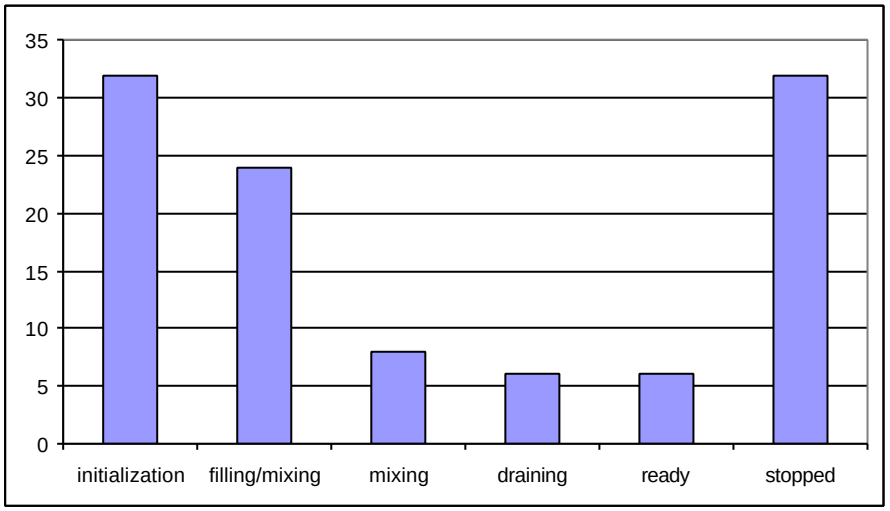

Figure 7. Amount of situations according to the states of the capsule *plant controller*

The design of the qualitative software model is oriented on the UML-RT structure model. Due to the determined message channels, only certain messages can be exchanged between the two capsules and their environment. This fact is described with system equations. Subsequently, the calculation of the qualitative software model takes place with the help of a program. Only 50 situations of the 108 situations of the capsule *plant controller* are suitable for the desired communication protocol with the capsule *plant specifications.*

5.3 Human Task

How do we model human behavior with SQMA? We consider the operator as a black box and are only interested on his possibilities to interact with the system. His access to the system is also represented with terminals of the component *human*. Generally, the human task can be divided into concurrent tasks or single tasks and tasks as answer of messages (reactive tasks) or tasks which do not rely on messages (active tasks). In the presented example an operator has different possibilities to interact with the process: He can open the door of the tank, start (active task), terminate or abort the process and set new values for the filling level. Because of the system structure, he can't open the door of the tank and change current process values at the same time, so these interventions are single tasks. Additionally he receives information from the software about the process status. Therefore in this example the component "human" has three terminals (Figure 8). The quantities of the terminals are also described with interval variables.

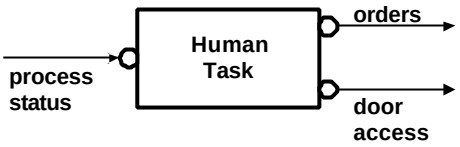

Figure 8. Operator as a black box

For example the qualitative variable for the new filling level contains all possible values from negative to positive infinite. This means that the operator can set any value. Actions of a human can easily be expressed in a qualitative way (e.g. open the tank door, shut it, or do nothing).

5.4 Composition of the Single Models

To connect the models of the technical process and the software, additionally, the necessary actuators and sensors can be modelled. Software information is converted into physical quantities, or vice versa. A qualitative modeling of the field peripherals additionally enables the possibility of analyzing the system behavior with defective actuators or sensors.

Before the situations of the whole system can be calculated, the three model must be joined together. The basic idea is to combine all possible situations of the three models. Therefore the interactions between these parts are important. The structure of the computer-controlled system represents the possible interactions and serves as the base for the combination. The situations of the three different models must fit together. For example a situation of the technical process presents a scenario for an open inlet valve. This situation only can be connected with an adequate situation from the qualitative model of the controller software: For example the capsule *plant controller* is in the state *filling/mixing* and therefore the manipulated variable for inlet valve has the value *open*. This means no conflict to the mentioned situation of the technical process; these two situations fit together.

A valid situation triple of the different models form a so-called system situation, which represent one special scenario for the whole system. For the homogenising plant, combinatorial 58210 situations are to be expected. The specified system structure reduced the number of possible situations to 49.

5.5 The Safety Analysis

Now, the analyst must examine the system situations with respect of their meaning for safety. The status of the situations helps the analyst to examine the system safety. In the situation table, Table 2, an extract of the result is shown. The first row contains the components of a special system part. For example the computer-supported analysis has discovered, that a situation can happen in which the inlet valve is open, the tank is filled and the mixer is running, the drain valve is closed. An adequate state of the controller software is that the capsule *plant controller* is in the state *filling/mixing* and the capsule *plant specification* is in the state *inputs blocked*. By evaluating the attributes of the situations of the system parts, the computer declares this situation as intended.

We can compare the system situations to a snapshot of the controlled system. In our example we have 36 situations, which are scenarios of the normal operation mode of the homogenizing plant. A program sorts the system situations into intended (normal), not-intended and dangerous operation. The safety analysis consist in examine the not-intended and particularly the dangerous situations.

technical process			controller software		human	status
inlet valve	*tank*	*drain valve*	*plant controller*	*plant specification*	*human task*	
opened	contains fluid mixer on door closed	closed	filling/mixing	inputs blocked	nothing	intended
closed	full mixer off door opened	closed	initialization	ready	open door	dangerous
...	...	...	...	...	...	...

Table 2: An extract from the situation table of the homogenizing plant.

For the homogenizing plant 6 not-intended operation modes of the systems are detected:

- *Inlet valve is triggered, but there is no flow through the valve (4 situations)*

  These situations show the analyst, that the cause of this fact is a blocked inlet valve or a run dry of the source. The analyst can get further information from the model. The above-mentioned situations only occur when the capsule *plant controller* is in the state *filling/mixing*. The evaluation of the dynamical behavior (the transitions) shows that this state could not be left. In reality, this means a deadlock of the system. Now the analyst has to interpret this result. In this case, the developed controller software is not able to detect whether water actually flows into the tank or not. This means that the current software design can't act upon this event.

  The next step is to evaluate if this event is probable. And if so the analyst has to think about a solution. For example, this deadlock problem can be solved with an extension of the software design. We can specify a new capsule *monitor*. The function of this new capsule is a temporal monitoring of the level of the tank. The user will get a message, if the current level of tank and the expected one are inconsistent.

- *Drain valve is triggered, but there is no flow through the valve (3 situations)*

  Something similar occurs with the state *draining* of capsule *plant controller*. It is also not detected by the software whether water actually flows out of the tank or not. This is obvious in the situations when the drain valve is blocked. This problem can be solved with the solution described above.

The discussed not-intended situations don't mean a risk for humans and environment, they only are important to the correct functioning of the plant. But the analysis has shown 6 dangerous situations of the homogenizing plant.

- *Liquid flows out through the door of the tank (6 situations)*

  With the developed software it can happen that liquid flows out from the open door of the tank. Depending upon the type and amount of the liquid the consequences can be differently serious. Nevertheless these system situations should never occur during the normal operation mode. Here, the qualitative model helps to detect a design error of the controller software. These dangerous situations can occur only if the capsule *plant controller* is in the state *initialization*. If an analyst considers these situations, he will soon notice the causes for this failure: the process conditions at the beginning of the process is not checked by the software during its initialization. If e.g. in a previous operation of the system the process was aborted by a user, liquid can still be in the tank. When restarting the system the door is unlocked again and the users has the possibility to open it or not.

  Therefore the developers of the software are forced to change the software design. The state transition diagram of the capsule *plant controller* is not sufficiently specified. A possible solution is to extend the statechart by a state *checking process status*. This new state has the function to lock the door first and only change into the state *initialization* if the tank does not contain any liquid. In the other case, the user is informed immediately and the state is changed into *draining*.

When the discussed modifications of the software design are made, the new design can be modeled once more and integrated into the model for the system. After the renewed analysis of the model, the dangerous situations shouldn't exist. But if so there must be new failures in the design and the analyst has to think about it again.

6 Summary and Outlook

The presented example shows, how computer aided safety analyses can be realized with the help of the qualitative modeling. Instead of brainstorming, the computer combines all possible situation of the technical process, the controller software and human task. The intended operation modes are checked under any conditions, moreover the analysis distinguishes between intended, not-intended and dangerous situations of the system. This helps the analyst to judge the system safety. A further advantage of the model-based approach is that the result is reproducible at any time. Thereby, the quality of the safety analysis can rise noticeably.

The example demonstrates how complex the interaction between the system parts can be, even for a small example. For larger systems it is almost impossible to discover or analyze several failures at the same time by conventional methods.

Like other modeling techniques the quality of the result depends strongly on the underlying model. The component-oriented view of the presented qualitative model creates outstanding prerequisites for support by a library containing checked model-components. Similarly to conventional CAD-Systems, models could be created on a graphic level comfortably.

7 Literature

[1] R. Lauber, P. Göhner, *Prozessautomatisierung*, Band 1 und 2, 3.Auflage, Springer-Verlag Berlin Heidelberg New York 1999

[2] Leveson, Nancy G.: *SAFEWARE - System Safety and Computers*, Addison-Wesley 1995

[3] Biegert, U.: *Computer-aided Safety Analysis of Computer-controlled Systems: a case example*, Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, Poland, 28.-31.08.2000

[4] Kuipers, Benjamin; *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*, Automatica, Vol. 25-4, 571ff, 1989.

[5] Xaver Laufenberg, *Ein modellbasiertes qualitatives Verfahren für die Gefahrenanalyse*, Dissertation, Institut für Automatisierungs- und Softwaretechnik, IAS, Universität Stuttgart 1997

[6] S.Mohr & S.Montenegro, *Analysen der Anlagen-Sicherheit und –Gefahren*, GMD-First, http://www.first-gmd.de/~sergio

[7] E Huber, G. Burgbacher, U. Biegert, W. Billmann, *Qualitative Systemanalyse und Computerunterstützte Gefahrenidentifikation*, Wiley-VCH, Chemie-Ingenieur-Technik, 7 / 97

[8] *UML-RT - Modeling Language Guide, Rational Rose*, Manual, www.rational.com , 2000

[9] Harel, D.: *Statecharts: A Visual Formalism for Complex System, Science of Computer Programming*, Elsevier Science publischers (North Holland) 1987., S 231-274.