11

Modification of Safety Critical Systems: An Assessment of three Approaches

Tor Stålhane, Ph.D and Kari Juul Wedde, Research scientist
SINTEF Telecom and Informatics, Norway
Tel. +47 - 73593014 E-mail: Tor.Stalhane@informatics.sintef.no

Abstract

This paper sums up the experience at SINTEF Telecom and Informatics on analysis of a safety critical systems for traffic control. After a short description of the system under consideration, the paper naturally falls into two parts. The first one is a description of two modifications, how they were implemented and how they were analysed for safety. The second one contains a discussion of the three methods used - FTA, FMECA and Code analysis. We here concentrate on how these methods differ in focus, the knowledge and information needed, and the types of problems they can handle.

The paper's conclusion is that all three methods are needed in order to analyse modifications of a safety critical system. The knowledge needed and the problem focus will, however, differ.

Keywords

safety critical systems, fault tree analysis, failure mode effect analysis, code analysis

1 INTRODUCTION

This paper describes the methodical part of our work on analysis of a safety critical system. The goal of the paper is to discuss the pros and cons of the three methods: fault tree analysis (FTA), failure mode, effect and criticallity analysis (FMECA) and code analysis.

As it turned out, these three ways of analysing a safety critical system provided the analysts with different foci and strongly influenced the failure modes that were identified. The paper is discussing the differences between analysing a complete system and analysing the effect of modifications. In addition, we will discuss how we - in the future - can combine all three methods and apply each in a way that supports the current part of the analysis, and enables the analyst to let his expertise have a maximum impact on the quality of the result.

2 STATE OF THE ART

Both FTA and FMEA has a long tradition of use in the analysis of safety critical systems. Both methods have been standardized by several bodies - both national and international. See for instance IEC 812 (IEC, 1985) and IEC 1025 (IEC, 1990). Problems related to the analysis of safety critical software systems has been standardized for instance by the MoD in their DEFence Standards 00-55 and 00-56 (MoD, 1991). A thorough treatment of the area is presented in (Redmill, 1993). See also (Bloomfield, 1989).

The use of FTA on software intensive systems started with Peter R. Harvey's Ph.D. thesis in 1983 (Harvey, 1983). The thesis contained a FTA of part of the control software for a solar satellite, and the author was able to discover a set of events that would lead to an uncontrolled spin of the satellite. This work was followed by several others, for instance Harvey and Leveson (Leveson, 1983) and Leveson and Stolzy (Leveson, 1984). The application of FTA for software safety analysis has, however, been slow to reach take-off speed.

The use of FMECA for safety analysis of software intensive systems has seen several attempts over the years. One of the oldest attempts is D. Reifer's paper (Reifer, 1979). In 1992, ESA published a guide for conducting FMECA on software intensive systems, (ESTEC, 1992). It remains to be seen how much of an impact this guide will have on software safety evaluation. One of the latest published papers in this field was written by T. Maier (Maier, 1995). This paper discusses both FTA and FMECA.

Several companies that develop embedded systems have been using FTA and FMECA for some years. See for instance (Rydholm, 1995) and (Stålhane, 1990). Both SINTEF and several other companies will keep on using FTA and FMECA in the future, and will work to extend the methods in order to improve their applicability for software.

3 THE SYSTEM

3.1 Background

The system under consideration was developed for traffic control. Some of the system functions could have an impact on traffic safety. It was therefore necessary to analyse the system design and the software with respect to safety. The result of this analysis was that the system, as delivered, was considered safe. The system was put into operation and after some time two non-safety problems appeared, causing the system to be modified. Before the modified system could be put into operation, SINTEF was engaged to analyse the changes with respect to system safety. This last analyses is the basis for this paper.

3.2 System design and needed functionality

The purpose of the system is to transfer messages between a Mobile Unit and a Central Unit, placed in a control room. The system consists of:

- Several hundred signal transmitters which have fixed positions.
- Mobile Units that receive signals from the transmitters and transfer these signals to the nearest Base Station located along the route of the Mobile Units.
- Base Stations that transfer the signals to a Front End Unit by cable.

- Front End Units that transfer the signals to a Central Unit for handling and to be displayed on a VDU, where the operator uses the information for a set of decisions.

The transferred information includes the identification of the Mobile Unit´s position. A correct identification of this position is required due to its use in a safety critical task carried out by the control room personnel.

The position information is saved in a table that contains all information about each Mobile Unit. This information is also displayed on a VDU when requested by the personnel or in connection with the display of a message from a Mobile Unit to the Central Unit.

3.3 Fault tree (FT)

Figure 1 shows the first four levels of the delivered system's FT, extended by the events "Case 1" and "Case 2". These two events represent the modifications, and are included in order to see the later analyses in the context of the first analysis. The analysis pertaining to these two events are discussed in chapters 4 and 5.

In FTs, square boxes denote events that are further analysed. These events can lead to the top event and are in the rest of the paper called unsafe events. Basic events are denoted by ellipses and are not going to be analysed any further. Events are connected trough gates. AND-gates are denoted by semi-circles, OR-gates by triangles and conditions by octagons.

3.4 Problems

The analysed problems were related to two problem areas and were analysed by two different persons:

Case 1: Hang or loss of the communication between the Central Unit and the VDU. The development company and the customer agreed that the reason for this was electromagnetic noise from the system's environment - EMI.

Case 2: In areas with disperse radio coverage, the Mobile Unit's position could be set to zero - unknown position - for shorter or longer periods. This is caused by a timeout and happens both in the position table and on the monitor in the control room. Unknown position is an event that can be handled by manual operator routines and is thus not a safety critical problem.

3.5 Basic assumptions

Only those parts of the system that contain changes were analysed. The reasons for this are:

1. Other parts were previously analysed and found safe. This includes the compiler, the operating system, all built-in procedures, the mechanisms used for passing parameters and all communication.
2. The system has been tested and in use by the customer. Up till now the system has accumulated approximately 36 unit years of operation without safety critical failures.
3. The unchanged parts have the same reliability and safety as before the modifications. We therefore assume that data changed or maintained by unchanged parts of the system have correct values.

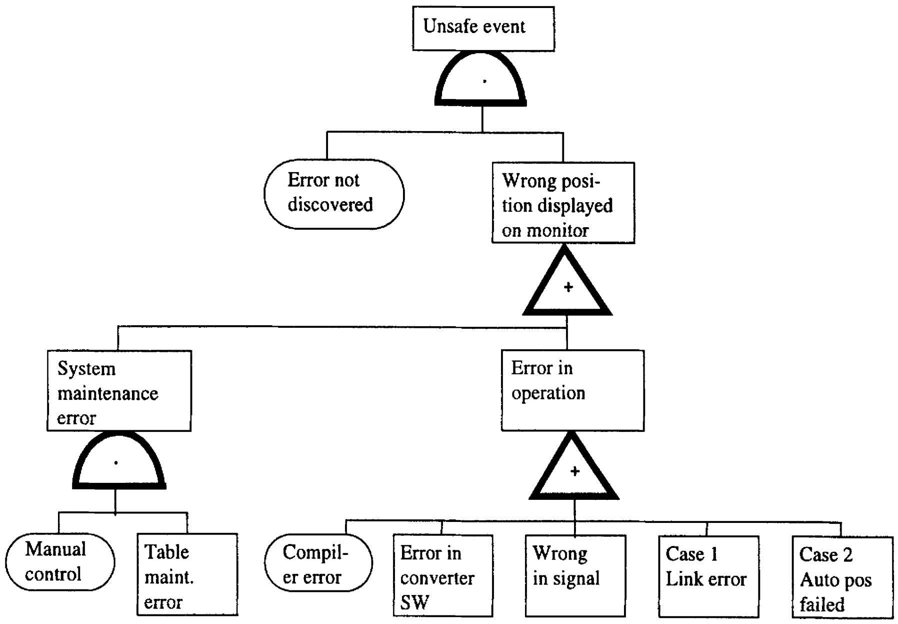


Figure 1 Delivered system's FT.

4 CASE 1

4.1 Problem and Solution

It was decided by the customer and the development company that the EMI problem should be solved as follows:

1. A timer watchdog was installed in the VDU link controller. If the time between two consecutive reset commands was too long, this timer was released and the communication software was restarted.
2. The timer was reset at regular intervals by a special message from the Central Unit. The choice of interval for resetting the timer watchdog defined the longest time the link could be down and thus, the availability of the VDU subsystem.

The solution and the required changes gave rise to two types of software modifications. The first one was that some new modules had to be written in order to implement the restarting of the communication link software and to refresh the information displayed on the screen. The other one was that some parts of the code had to be modified in order to adapt the system to the changes in the communication software.

4.2 Analysis and discussion

We decided to use two different approaches for the two types of software changes for "Case 1":

- **New modules**: Here we would use FTA, both for the new modules and for their interaction with the old, unmodified code. The analysis of the new interactions was done by augmenting the FTAs from the first analysis of the system.
- **Changes**: Here we decided to start with an FMECA. The failure modes were related to the influence of the changes. If the FMECA identified a change as having a possible effect on the overall safety-related top event of the FT of the delivered system, this event was added as an unsafe event in the appropriate place in the FT and the analysis was repeated.

The FMECA showed that potentially, any change involving an assignment or a procedure call can change one or more system parameters in a way that could compromise the system's safety. The job was to single out those changes that could either change a variable unintentionally or introduce an event or activity that could compromise safety in other ways. An example of an FMECA is shown in Figure 2 and the FT for "Case 1" - Link error - is shown in Figure 3.

The two main safety-related events identified in the new FT, namely "Link error not detected" and "System cannot repair link error" were developed further into new FTs. The FT for the first of these events - "System cannot repair link error" - is shown in Figure 4. The analysis that followed concluded that all changes were done in such a way that no new risks to the system's safety were introduced.

Module Description		*Failure Description*			*Failure Mode Effect*			
Id.	*Function*	*Mode*	*Cause*	*Detection*	*Local*	*System*	*Cnsq*	*Rate*
M1	Initialize link check count	LnkWD is not reset	X_2 error	Spurious restarts	Unnecessary BG restarts	Short periods of unavailability	Low	Low
M2	Loop until restart	LnkWD is destroyed	HW error	No restart when needed OR Spurious restarts	No restart of BG when needed OR Un-necessary BG restarts	No recovery when link error OR Short periods of unavailability	Med Low	Low

Figure 2 Example FMECA

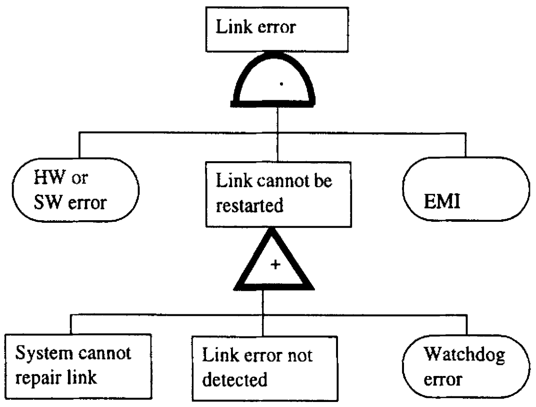


Figure 3 FT for "Case 1"

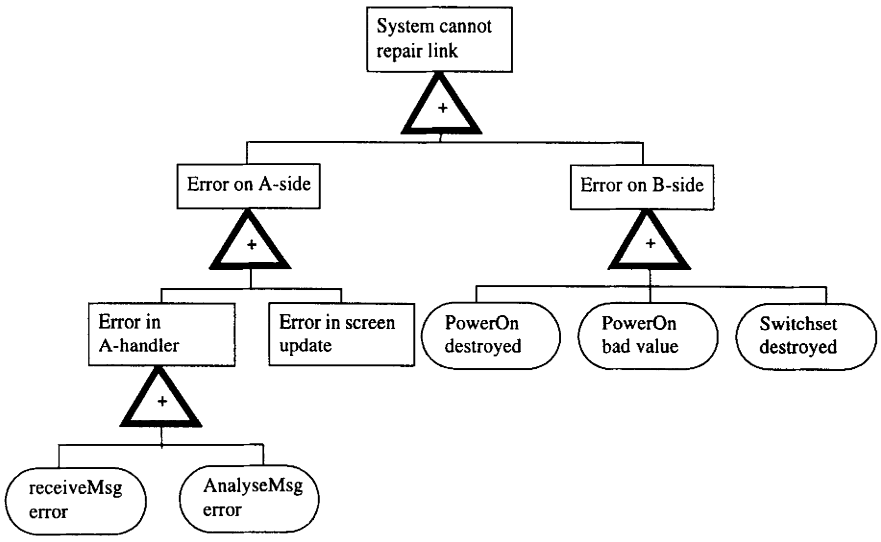


Figure 4 FT for the event "System cannot repair link error"

5 CASE 2

5.1 Problem and Solution

The problem concerning loss of position was solved by letting the Central Unit automatically send a request to the Mobile Unit and ask for the position. The request is sent when a Mobile Unit logs on the radio for the first time after leaving a radio shadow area. The main reason for choosing this solution was that it only required changes to the Central Unit. The other units remained unchanged. In addition, a position request is a message that has been in use for a long time by the control room personnel, for manually sending a request for position to the Mobile Unit. It was therefore seen as a well tested and thus a safe solution. The solution required changes to several code modules in the Central Unit.

5.2 Analysis and discussions

The analyses approach applied was a combination of two approaches:

FTA: FTA was used to analyse the modifications in the context of the delivered system, to see if the changes could lead to any of the unsafe events identified in the original FT. It was also used to structure and organize the information obtained during the code analysis.

Code analysis: In order to see which unsafe events the changes could lead to, detailed knowledge of the modified parts of the system was needed. These parts were therefore studied in detail, partly by manually executing message sequences for the changed parts of the communication.

Seen in the context of the delivered system, the changes were related to the event "Wrong position displayed on monitor". Related to the "Case-2" changes, this event may occur in two situations. Either the automatic position request is not sent when the Mobile Unit logs on after leaving a radio shadow area or the automatic position request is sent, but fails.

For the first situation, the system will behave as before the modifications and no new risks are added. We therefore only have to analyse the second situation, which adds a new unsafe event to the FT of the delivered system. This new event was put in the "Error in operation" branch, and is denoted "Case 2" in the FT in Figure 1 and is the starting point of our analysis of the modified system. The top event for "Case 2" is named "Automatic position test failed".

The code analyses was used for familiarisation and with respect to what could lead to a wrong position - we found that:

1. Automatically sending "Position request" requires that the response is converted in order to be handled automatically. If such a conversion is performed on response to any other message than a position request, it will lead to a safety critical event, namely that the data part of another message is wrongly interpreted as position information.
2. The request number is not returned as part of the answer. The conditions used to single out the response to a position request are based on the values of the source and destination variables of the message.

The FTA was used for information structuring. The FT for "Case 2" is shown in Figure 5. Two events were expanded in lower level FTs. Only the "Wrong message contents" event is described here. The FT for this event is shown in Figure 6. In this FT the "Converting response to request >< position request" event is the only concern. How can we be sure that the conditions used to single out responses to a "Position request" does not let other messages through?

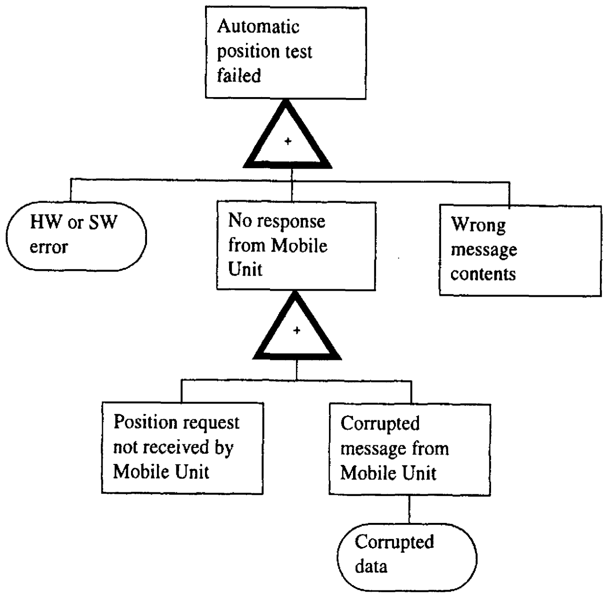

Automatic position test failed
+
HW or SW error
No response from Mobile Unit
Wrong message contents
+
Position request not received by Mobile Unit
Corrupted message from Mobile Unit
Corrupted data

Figure 5 FT for "Case 2"

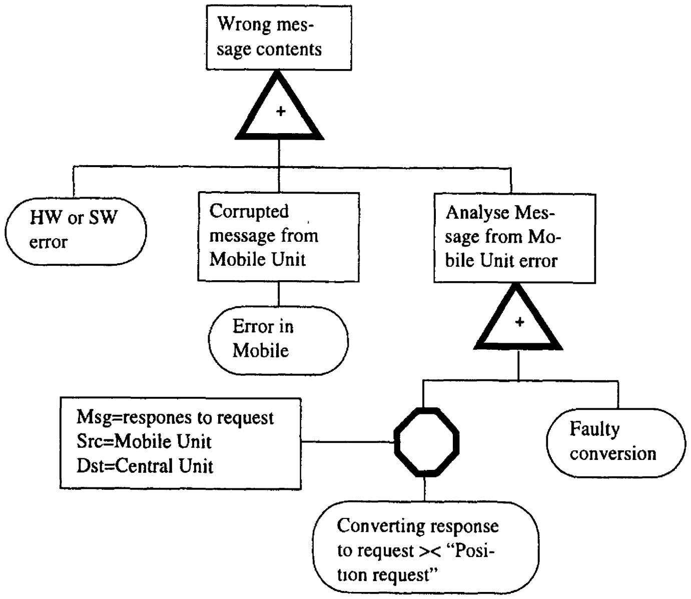

Wrong message contents
+
HW or SW error
Corrupted message from Mobile Unit
Analyse Message from Mobile Unit error
Error in Mobile
+
Msg=respones to request
Src=Mobile Unit
Dst=Central Unit
Faulty conversion
Converting response to request >< "Position request"

Figure 6 FT for the event "Wrong message contents"

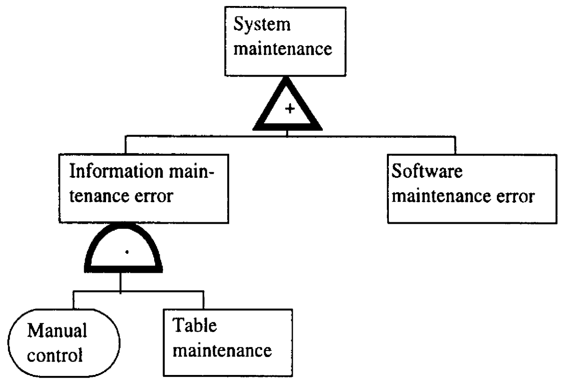


Figure 7 Augmented system FT

According to the system developer, "Position request" is the only message receiving an answer with "Source = Mobile Unit". Discussions with the developer has convinced us that this statement is valid and that the modified system is safe. The rather complicated predicate gave us, however, some concerns regarding system maintenance. The statement may, however, be violated by future changes, for instance by extensions to the message set. The restrictions imposed by the changes must therefore be thoroughly documented in order to avoid problems later on. This is especially important since the conditions used to single out the response to a position request were not designed to be used for such a purpose.

The objective of the analyses was to see if the modified system was safe and could be put into operation. Therefore, only the "Error in operation" branch of the original FT was analysed. What we learned from this case, however, was that we also have to take future changes into consideration and that the "System maintenance error" branch has to be analysed as well. As a result of this we will extend the "System maintenance error" part of the FT in Figure 1 with a "Code maintenance" branch through an OR gate as shown in Figure 7.

6 DISCUSSION OF THE THREE METHODS

6.1 The goal of the discussion

The two modifications -"Case 1" and "Case 2" - were analysed in two different ways - partly due to the different nature of the changes and partly due to the differences in background and experience for the two persons who performed the analysis. Even though all three methods have the same focus - to check if the system is still safe after the changes - the approaches differ in how they achieved their goals.

All methods for safety analysis - or any other analysis for that matter - are mainly used to organize the analyst's ideas, experience and knowledge. It is thus clear that people with different mind sets will need different methods in order to apply their experience in the most efficient way for a particular problem.

Our goal in the following discussion is thus not to look for a best method for safety analysis, but to study the types of problems that can be identified by applying each analysis method to the problem at hand.

6.2 FTA

An FTA includes all kinds of system components, such as hardware, software, operators and environment and organizes all events that are identified as being safety critical, irrespective of how they are implemented. In addition, it focuses on one single, critical event and then follow this down through the system levels to a predetermined level of details.

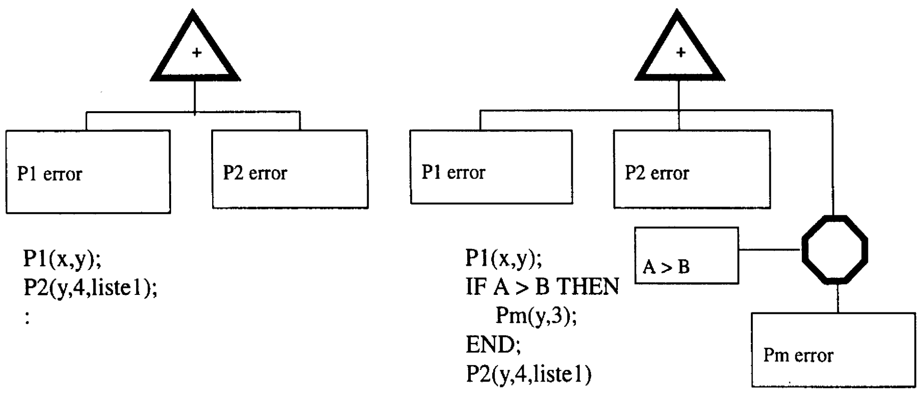
It is relatively straight forward to go from software code to a corresponding FT by means of an automatic tool - see Leveson (1984). However, since this generated FT is just a new representation of the code, no new insight is gained in the process. In addition, the purely automatic approach leads to an FT that lacks focus and ignores system and software knowledge that can surface during a more goal oriented FT building process (Stålhane, 1990).

For small and dispersed changes, the FTA will consist of going through all existing FTs related to the changed parts of the software and see if we need to change the FT because of code modifications. Parts that were not safety critical in the delivered system, could be made safety critical by a modification. This approach supposes that we already have a system FT.

Some of the parts that were modified were not originally considered safety critical and they were thus not part of any of the FTs made for the delivered system. The approach described above could not be used for the small, dispersed code modifications except if the influenced areas already were covered by a FT.

For the new subroutines and additions of large code segments - more than say, 50 lines of code each - the application of FTA was straight forward. Figure 8 shows a simple example.

The FTA is in our experience a convenient way to combine system knowledge, application knowledge and knowledge of software implementation. All three types of knowledge are important, but the focus of the analysis will decide their relative contribution in the analysis. An FTA will only be efficient in two situations. One is for large modifications, where a new FT or substantial additions to an existing FT are needed. The other one is in cases where an FT already exists for the parts of the system that are modified, and the new FTA just consists of checking how the components are influenced by the modifications.



```
P1(x,y);
P2(y,4,liste1);
:
```

```
P1(x,y);
IF A > B THEN
   Pm(y,3);
END;
P2(y,4,liste1)
```

Figure 8 Example FT for a code modification

6.3 FMECA

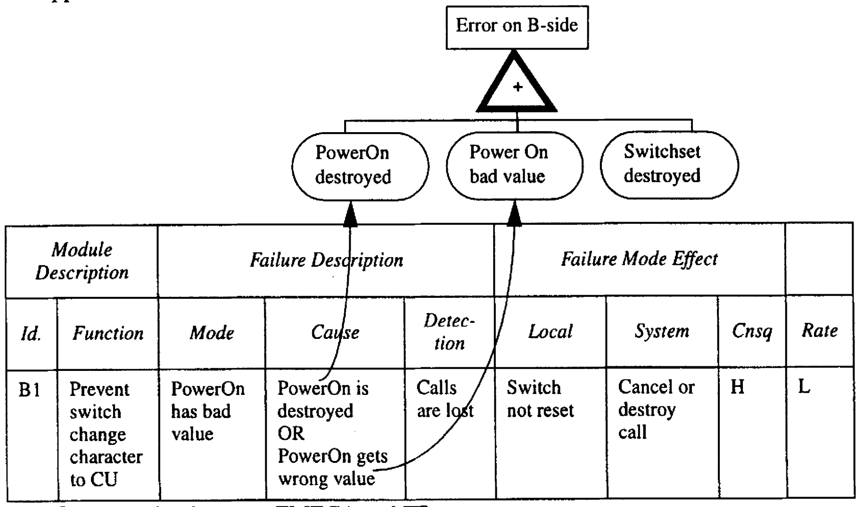
In principle, an FMECA could be used for a safety analysis of the delivered system. However, in order to trace each failure mode from its local source to the top level in the system, one must perform an analysis that logically is close to an FTA or to an extended structure diagram - ESD (Stålhane, 1990). Even though this is usually not the case for simple systems, it will almost always be the case for software. In our opinion, a FMECA should not be used alone on a complete software system. When it comes to modifications, however, and the FT is already available, an FMECA is an important method. We can approach the safety analysis through the following steps:

1. Identify all changes that may have an impact of the system's safety. Such changes are changes related to the logical structure of the system - decisions - or to the data managed by the system - the system's state. This follows from the fact that a software system only can fail if it receives new input in any state or old input in a new state.
2. Follow the identified changes up through the system's levels and identify which low level events in the FT that will be influenced by the modification.
3. Re-evaluate the FT with the changes caused by the software modifications.

An example is shown in Figure 9. Here, the FMECA identified the variable PowerOn as safety critical and defined two events where this variable could cause safety-related problems. These two events where added to the appropriate FT. See also the FT in Figure 4.

An alternative approach could be to make an FMECA for the next level up - in our case the procedure level - and let the identified possible critical events be failure modes for the next level FMECA ans so on. See (Rydholm,1995). Since we already had the system FTs available, this approach was not tried in our case.

For an FMECA of code modifications, the implementation and programming language knowledge is of major importance. Application and system knowledge will be important only at the upper level.



Module Description		*Failure Description*			*Failure Mode Effect*			
Id.	*Function*	*Mode*	*Cause*	*Detection*	*Local*	*System*	*Cnsq*	*Rate*
B1	Prevent switch change character to CU	PowerOn has bad value	PowerOn is destroyed OR PowerOn gets wrong value	Calls are lost	Switch not reset	Cancel or destroy call	H	L

Figure 9 Connection between FMECA and FT

6.4 Code Analysis

Code Analyses was used to study the code in order to check if the changes could lead to a specific unsafe event - identified by a previously performed FTA. In order to do this we needed broad and long experience with software development. This is of major importance in order to pinpoint problem areas. In this context, side effects of the modification are of special interest since they can impose new risks. Changes that does not fully solve the problem are not critical since the system remains as before with respect to safety.

In addition, system knowledge is needed in order to have a context for judging the problem areas found. It is of great importance that changes do not impose unwanted interactions between changed and unchanged functions. It is also important to check that the changes fit into the original design of the system.

Except for the relationships to the unsafe events, it is not easy to define a general approach to code analyses. The reason for this is that a major part of the analyses always will be the combination of system understanding and general development experience. A possible approach may be to develop a checklist for safety critical code. SINTEF will later look into this possibility. In our opinion code analyses is most efficient for system modifications. The amount of states and decisions that need to be considered will be too large for a complete system.

Below is a small piece of code that was of special interest in our case. It contains the changes to a procedure in the Central Unit that analyses messages from the Mobile Units.

```
(* --- check if answer to an automatic position request--- *)
IF(Class = Answer) &
(Code = Request) &
(Source = MobilUnit) &
(Destination = CentralUnit) THEN
(*--- Convert data part of the message to position information --- *)
END;
```

The code is new and entered in order to handle response to an automatic position request. The purpose of the code is to convert the data part of this response to position information. Four predicates are needed in order to single out the messages for which this conversion has to be done. None of the predicates pertain directly to a position request. Our general experience tells us that changes combining several - originally unrelated - status variables frequently lead to problems. Thus, this code is considered a problem area. The conversion of the data part of a "wrong" message would be a safety critical side effect.

In order to verify whether this change is safe or not, we need to know more about the system and especially about the communication protocols. The protocols has to be studied in order to find out if there exist other messages satisfying the given predicates. If such messages exist, we would get the side effect mentioned above. In addition, the conversion would disturb the function that the message is a part of. We will thus get unintended interactions between changed and unchanged functions.

By studying the message protocols and discussing with the developer, we concluded that source and destination - in this case - could be used in order to single out the response to a position request. The solution could thus be considered safe. Such solutions were, however, not planned for and not covered by the design. It therefore imposes future risks. Our recommendation was therefore that the modification could be put into operation if comments were added to the Mobile Unit code where it responds to requests the following actions were performed. The protocol specifications also had to be updated according to the changes performed, especially with respect to the restrictions imposed by the conditions set forth in the added comments.

7 SUMMARY AND CONCLUSIONS

We have seen in the discussion above how the three methods FTA, FMECA and code analysis can be brought to bear on safety evaluation for modifications to a software intensive system. By starting with a system that has already been analysed for safety through a FTA, we could trust those parts of the system that were not effected by the modifications. In addition, the FT - possibly augmented - could be used to study the impact of the local modifications on the system's safety.

The FT of the delivered system - augmented because of later changes - was the basis for both types of analysis. This FT, or the corresponding ESD, is needed in order to follow the local effect up to the system's effect. This may not be necessary for an FMECA for another type of system but is in our opinion important for a software intensive system due to the complexity of such systems. Thus, the FT is needed both for FMECA and code analysis.

The FMECA and the code analysis did, however, have quite different foci for their analysis. This difference influences what we find. The FMECA focuses on what happens if a statement goes wrong or does not have the intended effect, while the code analysis have three areas of concern: The unsafe events, good development practice and the combination of development and system knowledge.

The FMECA in all cases pointed out code segments or procedures that needed to be further investigated by the FTA. As such, the FMECA should be considered a supplementary method for FTA of changes to an already existing system.

The code analysis pointed out cases where the code, as it was after the modifications, was correct but was written in such a way that later changes could jeopardize the system's safety. This experience is consistent with our experience from the safety analysis of an ILS system (Stålhane, 1995).

Our experiences can be summed up in Table 1.

Table 1 Experience summary

Method	*Performed by*	*Input*	*Results*
FTA	Systems engineers Software engineers Safety engineers	System description Software code	FTs Dangerous events organized in cut sets
FMECA	Systems engineers Software engineers Safety engineers	Software code FTs	Dangerous events for FT augmentation
Code analysis	Software engineers	System documentation Software code FTs	Dangerous events for FT augmentation Dangerous design side effects for later maintenance

8 REFERENCES

Robin E. Bloomfield et al. (1989), Requirements for the Analysis of Safety Critical Hazard. Adelard report.

CEI/IEC (1990), Fault tree analysis (FTA), CEI/IEC standard 1025.

ESTEC, (1992) Guidelines for considering a software intensive system with FMECA studies. QS/91/247/082/RA

IEC (1985), Analysis techniques for system reliability. Procedures for failure mode and effect analysis (FMEA). IEC Standard publication 812.

Peter R. Harvey, (1982) Fault Tree Analysis of Software, Ph.D. Thesis, University of California, Irvine.

Ministry of Defence (1991), Hazard Analysis and Safety. Classification of the Computer and Programmable Electronic System Elements of defence Equipment. Standard 00-56 / Issue 1.

Nancy Leveson and Peter R. Harvey, (1983) Software Fault Tree Analysis, Journal of Systems and Software, no. **3**, 173-181.

Nancy Leveson and Janice L. Stolzy, (1984) Software Fault Tree Analysis Applied to Ada, COMPSAC, November 7-9, Chicago, USA, 458-466.

Thomas Maier, (1995) FMEA and FTA to support safe design of embedded software in safety-critical systems, First annual ENCRESS Conference, Bruges, Belgium, 12 - 15 September, section 20.

Donald J. Reifer, (1979) Software Failure Modes and Effect Analysis, IEEE Transactions on Reliability, vol. **R-28, no. 3**, August, 247-249.

Felix Redmill, (1993) Safety-critical Systems - Current issues, techniques and standards. Chapman & Hall, London.

Kjell Rydholm, (1995) FTA and FMECA for Software, ENCRESS seminar, Borås, Sweden, 29 November.

Tor Stålhane, (1990) Fault Tree Analysis as Tool for Safety and Reliability, Second European Conference on Software Quality Assurance, May 30 - June 1, Oslo, Norway.

Tor Stålhane and Joe Gorman, (1995) Review of DSP Software, SINTEF memo 400407.37, SINTEF, Trondheim.

9 BIOGRAPHY

Tor Stålhane was born in 1944 in Skien, Norway. He studied electrical engineering at the Technical University of Norway from 1964 to 1969. After this he worked with compiler development and maintenance until 1985. He then had a four years leave to complete a Ph.D. in statistics, which was finished in 1988. After returning to SINTEF, he has been working on software reliability and safety plus software process improvement. He is associate professor in computer science at the polytechnic in Stavanger, Norway.

Kari Juul Wedde was born in Steinkjer, Norway. She studied computer science at the Trondheim Engineering high school from 1970 to 1972. After that she has worked for SINTEF. During this period she has combined practical work with theoretical studies in computer science at the Technical University of Norway. Past experience include compiler development and maintenance, software engineering environments and formal verification and validation of communication systems. Current work includes software process improvement and distributed software architectures.