CENELEC WGA10 Workshop, *Bridging the Gap to Railway Interoperability*,
May 11, 1999, Münich - Germany

Safety assessment of computerized railway signalling equipment

Tadeusz CICHOCKI*, Janusz GÓRSKI**

*Adtranz Zwus, ul. Modelarska 12, 40-142 Katowice, Poland, e-mail: tadeusz.cichocki@plsig.mail.abb.com
** Department of Applied Informatics, Technical University of Gdańsk,
ul. Narutowicza 11/12, 80-952 Gdańsk, Poland, e-mail: jango@pg.gda.pl

Summary. Safety analysis aims at elicitation of the knowledge about the system behaviours and about possible consequences the system depended events can have in the environment. An important source of this knowledge is the inventory of potential faults of components and the conclusions regarding their influence on other components and the overall system behaviour. The analysis helps to identify conditions triggering activities aiming at elimination of negative consequences of such faults. The paper discusses the problem of extending the applicability of the conventional safety analysis techniques like FMEA and FTA to software intensive systems. The discussion refers to the present practices and experiences in Adtranz Zwus, the company which is a major supplier of railway signalling equipment in Poland. In this context the strategy of extending the applicability of the conventional safety analysis to the computerised systems is presented.

1. INTRODUCTION

Software increases its role in industrial control systems. Rapid progress in technology development enables easier implementation of control and monitoring functions, increase of scalability and adaptability to new requirements, decrease of development and usage costs and increase of performance. At the same time in many applications dependability of software behaviour becomes a significant factor.

Safety analysis concentrates on possible negative consequences of system's behaviour in its environment and includes:

- analysis of hazards related to the system,
- analysis of risks implied by the hazards,
- identification of safety requirements and resulting design decisions in order to eliminate or control hazards (if they eventually occurred) and reduce associated risks.

As a result of safety analysis safety requirements and strategies are defined and the arguments supporting safety assurance solutions are collected. They include strategies for identification and consequences mitigation of events leading to hazards.

Safety to high degree depends on a proper co-operation of elements based on different technologies: software, computer hardware, mechanical and electrical equipment, operator control and monitoring and maintenance actions. To be successful, safety analysis should be based on the model of sufficiently high level of abstraction and broad enough scope to adequately include all the characteristics of these heterogeneous domains. Increased complexity of systems structures, implemented in hardware, software and human behaviours may lead to new hazards developed as an undesirable interactions between system components.

Software is by its nature discontinuous: a small change in code or input data may cause a significant difference in software execution and data output. Sources of changes of this kind include:

- development errors,
- compilation errors,
- input data errors,
- operating environment errors.

Due to this characteristic the problem of ensuring that software observes the basic safety rule which states that any fault (system's component failure) triggers the system's actual state to a defined safe state, is particularly difficult to implement.

2. FMEA ANALYSIS

Identification of the system architecture, modelling of functional dependencies between components, developing the inventory of potential faults of different components, and identification of error propagation scenarios initiated by those faults are the basic activities of the FMEA (*Failure Mode and Effect Analysis*) technique. However, a straightforward application of the classical FMEA (that, which is commonly used for mechanical and/or electrical equipment) to the systems which include a software component leads to difficulties and problems resulting from the specific nature of software systems.

FMEA analysis is based on systematic *identification* and evaluation of *propagation* and *effects* of single system component faults. Information collected during analysis includes *types, scenarios* and *patterns of failures,* and intended actions of error disclosure and control and state regeneration after errors.

FMEA is performed on system architecture and on components of that structure. The architecture is analysed by considering the set of anticipated component faults (deviations in required functions) with the aim to disclose the resulting failures of the whole system. This lead to the distinction of *safety related faults* (those with the hazardous effects).

FMEA starts from the lowest (base) level of the components. Those are the components which internal structure is not interesting from the point of view of the analysis. Anticipated faults of those components and the assessment of their effects are documented in the *FMEA table*. The table is a data structure which collects the results of the analysis. Scenarios of failure development in the system initiated by the anticipated components faults are identified and documented. Effects of the components faults are interpreted in terms of higher levels of the functional structure, and finally by their impact on the system external interfaces. Links between the scenarios and the proposed actions aiming at risk elimination or reduction are documented as well.

In Fig.1 a general view of the scope of FMEA analysis is illustrated. FMEA analyses system architecture by identification of error propagation scenarios leading to transitions from component faults to system failures. Efficiency of the analysis is conditioned by the following factors:

- choice of the base components and proper identification of their faults,
- adequate model of the architecture representing functional dependencies between components,
- adequate identification of error propagation scenarios initiated by faults of the base components.

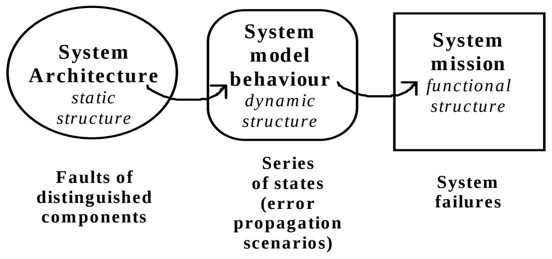

Fig.1. Scope of FMEA

FMEA analysis may be performed early in the system life-cycle, even then, when the project development process is not advanced enough to define a complete model of the architecture. In this case the analysis is performed on the „idealised architecture", reflecting early intentions and design assumptions and a general outline of system structure. Descriptions of this kind are generally not sufficient for complete understanding of the nature and consequences of the faults of the low level components. Nevertheless, the early analysis provides for design decisions being an adequate response to identified hazards. While the project progresses the analysis is extended to include more detail and to increase precision in the most critical areas of the system structure. For each subsequent life-cycle phase FMEA helps in discovering potential hazards resulting from the undertaken design decisions. Therefore FMEA can be considered as an iterative tool, supporting decision making from the earliest stages of the system development.

The role of FMEA in safety analysis may be summarised as follows:

- supports identification and evaluation of hazards arising from faults in the system,
- helps in identification of potential, possibly not recognised before, states of the system,
- provides for identification of modules, procedures, processes or functions which are safety-critical,
- supports the review of events in the system associated with signals passing from sensors to their final receivers (which leads to better

understanding of the system),

- supports identification and verification of requirements for safety-critical functions,
- supports design of safety-critical functions and verification of their correctness,
- supports verification of independence of safety-critical functions and other system functions (dependency may result from a common resources usage, for instance),
- supports identification of those areas of the system through which the consequences of component faults are transferred to the higher level of system functions,
- supports the design of system test cases verifying error scenarios resulting in hazards,
- supports the definition of diagnostic activities and the requirements for a safe system service and operation.

The activities of FMEA correspond to the first three of six functions defined in the SEI *Continuous Risk Management* model [8]:

- *Identification* (search for risk before it becomes a real problem),
- *Analysis* (transformation of risk data into information which allows decision making),
- *Planning* (use of the information in decision making in order to transform the information into system implementation).

3. FMEA FOR SOFTWARE

The question arises if the FMEA analysis may be performed directly on the software implemented parts of the system. According to IEC 812 [9], confirmed by the experiences of NASA, Boeing Corporation and other companies, FMEA may be used to identify hardware system states influencing requirements specifications for software controlling the system. But the analysis extended into the internal structure of software encounters substantial difficulties. Three basic sources of the difficulties are:

- lack of commonly accepted and univocal model of software architecture,
- low level of software code quality,
- dependency of software on hardware and the development and operation environments.

More detailed discussion of those problems is presented below.

Software Architecture

There is an open problem of choosing low level software components which form the basic level of structure during FMEA analysis. Another problem is the classification of fault types for those components. Currently there is no common opinion on what the components are and how they and their interactions should be specified in order to understand their impact on the whole system.

Syntactic constructors present in programming languages, like modules or classes are used to group definitions of data and related actions but their semantics is not fully defined and univocal. Description of module interactions in classical languages is not explicit. Examples of mutual dependencies and influences of programming modules are:

- *procedural and functional dependencies* – a procedure or function may refer to elements (global type declarations, global variables and constants, other procedures and functions) defined outside the body of this procedure or function,
- *type and structure data dependencies* - basic data types defined in different modules may be used to construct more complex data types; variables and constants may represent a specific type defined by an explicit declaration in the program or by (implicit) result of an operation,
- *data interference* - unintended modifications of common data caused by design errors, external data errors, hardware faults,
- *system services interference* - caused, for instance, by mutual blocking of processor usage,
- *program control interference* - in implementation of exception handlers, concurrent processes triggered by external signals, and as a result of program control errors resulting in the execution of wrong procedures,
- *common cause faults* - as software modules may use common resources (what is not explicit in the program source code) some of their faults are in strong correlation: a common resource error or fault may lead to an error of all software modules using the resource.

Possibility to perform an effective FMEA analysis for low level software components (programming language constructs visible to designers) is dependent on a proper semantics defined for the language. For the purposes of FMEA, a semantics needed is the semantics mapping programs into their properties which explain the impact of the program on its environment, instead of how the program works.

Quality of object code

Conclusions from FMEA analysis of hardware systems are in most cases the descriptions of real world system states and their dependencies. The analysis covers component failures caused by random faults during operation (resulting from ageing and wearing of the components, electromagnetic interference, vibrations, dust, and other factors). The faults are usually well known and recognised (in terms of their probabilities) for a particular type of component in relation to the conditions of its operation.

In the case of software, the situation is quite different. There are no random defects of software components (software is not subjected to ageing and wearing). The basic software fault is a design fault. Present technology is not able to guarantee that software is free of faults (unintentionally introduced by designers, programmers or tools). These faults can have a substantial influence on software module behaviour and consequently on the whole system. Therefore the analysis of possible faults arising from defects in physical resources „carrying" the software program (processors, memory, communication channels etc.) and the analysis of the system architecture should be extended to include the software development process which is "responsible" for the faults introduced to software. In this extension the techniques and methods used during software development (the „components" of the development process) could be analysed similarly to the components of the system architecture. Necessity to perform such an analysis is postulated by standards, DS 00-55 [2] standard for instance. Like in the case of the classical FMEA, the development process analysis should be performed from the earliest phases of the software life-cycle. The conclusions from the analysis may be used to improve this process in order to increase its overall effectiveness.

Environmental dependencies

Some software defects lead to serious consequences (accidents) only in coincidence with additional hardware faults, human errors (user o operator) or some specific influences from other components or from the environment. It is estimated [1] that 35 % of all software errors are the consequence of software-hardware dependencies. Other experiences show that large amount of defects is transient in nature (for instance: a transient fault of a hardware component destroys data). In these situations normal operation can be restored, when the program restarts from a previously stored correct state. For instance, EN 50128, DS 00-55 and IEC 812 standards [4, 2, 9] postulate to perform the analysis of the system operation deviations (and FMEA) for the whole system with its software - hardware - environment interactions.

4. SAFETY ANALYSIS IN ADTRANZ ZWUS

The analysis of the previous experiences and the recapitulation of the recent training and research project formed the foundation for a new systems safety assurance program in Adtranz Zwus. Adtranz Zwus is the major supplier of railway signalling equipment in Poland. In order to satisfy the demanding safety requirements for developed systems Adtranz Zwus continuously searches for more effective safety and reliability assurance technologies. The objective is to meet the requirements of the CENELEC EN 50126/-8/-9 and Polish Railways (PKP/CNTK) [3, 4, 5, 12] norms.

One of the pilot projects implementing some elements of the program was *Computerised Line Block System SHL-1* project.

Initial assumptions

System safety analysis was performed according to the scheme in Fig. 2.

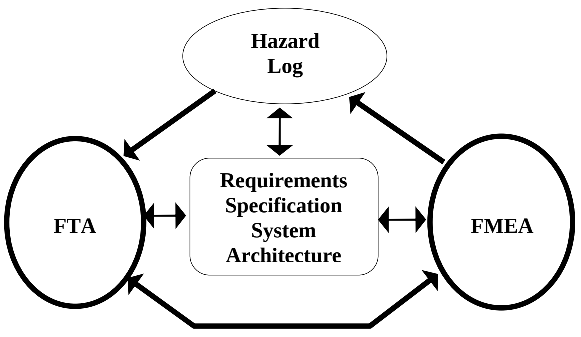

Fig. 2. Structure of safety analysis for SHL-1.

The starting point of the analysis were the

Hazard Log, the Requirements Specification and the Architecture Specification of the system. The basic techniques used in the analysis were *Fault Tree Analysis* - FTA and *Failure Mode and Effect Analysis* - FMEA. The complementary nature of those techniques was exploited during the analysis: a mutual verification of the delivered results could be performed. In the sequel we present the scope of the analysis with some additional details.

Hazard Analysis

After considering the hazard characteristics of the Line Block System in its domain, the following classification of hazards was defined:

- lack of „Stop" signal, if the protected line block is in use (resulting from the red light bulb fault or a brake in its circuits),
- disconnection of train cars,
- disconnection of the rail track,
- brake in transmission from train detection systems or semaphores,
- decay of power supply for train detection systems or semaphores,
- disturbances in train detection systems (wrong wheels number count).

During analysis the potential malfunctions and environmental influences on system operation were considered. The following failure states that must be communicated to the operators were identified:

- fault of a semaphore bulbs or their circuits,
- opening of the container of the safety related equipment
- fire in container.

The overall system safe state was defined and safe states of the geographically distributed components were defined as well.

Requirements specification

A fundamental step towards a more detailed analysis was the requirements specification of the system. Defining the requirements required additional interpretations and elicitation of more knowledge from the Polish regulation authorities concerning the Line Block System (PKP WTB E-10 and E-1 [1, 4]). This work resulted in a development of the System Requirements Specification document. This document was structured in accordance with a previously defined standard. Operational modes of the system were defined, as well as the system boundary (logical and physical). All relevant notions and concepts of the application domain were explicitly defined in the system vocabulary. Functions of the system responsible for the system control and functional safety were specified. A system safe state was distinguished to be assumed in response to a detected fault, with the following characteristics:

- no train departure from a station is allowed,
- no incoming control signals to the system are processed,
- the rail track segment (the line block) related to the fault is protected by the proper image of lights shown on the corresponding semaphores.

The safe state of each distributed system component was defined as follows: it is a state of the component which initiates the transition of the Line Block System to its safe state.

FMEA analysis

Identification of the error paths caused by potential faults of distinguished components was used to verify the mechanisms of system reaction to the fault events. This helped in reviewing and completion of the component exception responsibilities (rules and procedures). In particular, it was checked if the system safety reaction (that is the system state transition to a pre-defined safe state) is taken in an automatic fashion in response to each system component fault. The level of faults identification in the system structure and of safety reactions was checked for being complete. This reasoning was based on distinguishing the critical components (through analysing their mission in the system), the critical components local FMEA analysis and the global FTA analysis.

The basic level of system components was chosen to be the level in the structure where the system reactions to hazards are initiated (structural safety). The list of those components was extended by including main objects of the environment (the modules being the sources of data or the modules controlled by the system).

In the cases where the safety reaction to a component fault did not result in a safe state, further identification and assessment of the resulting hazards was performed. Those behaviours which could not be accepted (in the light of the associated hazards) were collected in order to look for the adequate design modifications. After the design has been changed the analysis was repeated to see if the change provided for hazard elimination. The documentation of the last cycle of the analysis was used to demonstrate the completeness of the analysis and to support the claim that the architecture was safe. In addition, the results of the analysis were used as the basis for test cases specification.

The final form of the FMEA table was defined included the following items: type of fault, effect of fault (local, secondary, final), cause of fault, effect assessment (risk connected to the fault), solutions proposed (improvements or prevention), method and time of fault detection, safety reaction.

A fault of a component was defined as the negation of its function/service provided to other components. The (positive) effect of a function/service was understood as the delivery of proper operation (data and/or signals) at proper time. The data on faults were collected in the following table:

Tab.1. Documentation of component faults.

No.	Compo-nent	Component faults list	Justification of the list

It was assumed that faults in communication required for a particular service, not explicitly described in the analysis, are limited to a lack of the communication and are attributed to the component delivering the service.

Criticality assessment was based on the level of function degradation after a particular fault:

- extent of the lose and damage caused by the fault,
- duration of the disturbance or system function degradation,
- availability of the alternative measures for safe train operation with a degraded functionality of the system,
- user/operator assessment and diagnostic possibilities.

The following criticality scale was assumed:

Tab.2. Fault criticality levels.

Critica-lity level	Assessment of the event	Assessment from a user/operator point of view.
5	Not accepted	User considers the situation to be correct.
4	Not accepted	User does not know, that the situation is not accepted.
3	Not accepted	User does know, that the situation is not accepted.
2	Not accepted with moderation	User does know, that the situation is not accepted (*disturbance*).
1	Slightly not accepted	Not important.

The following strategy was adopted:

- even if the analysis of a high level of the system structure confirms its safe behaviour, it is required to consider the possibility of interactions of lower level components leading to hazards [10],
- every fault considered in the analysis should be discussed as a symptom of some underlying process.

Local FMEA analysis performed for critical components (selected from the list of the base components) was used to support the above classification of faults.

FTA analysis

Combinations of faults which could result in hazards and related design decisions aiming at prevention of this situation were considered during the FTA analysis.

The list of faults potentially leading to hazards is contained in the set of faults postulated and considered in FMEA. On the basis of fault trees it was argued that no single fault can depart the system from its safe state. It was verified that no fault in the system is able to trigger a system state change while the system is in a safe state. It was also demonstrated that moving the system from the safe state requires an external operator intervention. Those analyses were supported by a test program defined for the laboratory and domain environments.

Reliability of the transmission subsystem was not analysed on the level of frames generation. It was assumed that implementation of information coding is comparable to the one used in Ebilock 850 system (a product of Adtranz Signal, Sweden), certified and used in multiple European installations. Nevertheless a possibility of errors in data management between control points of the system (according to the classification of EN 50156 [6]) was considered: transmitter identifier error, data type error, data value error, out-dated data or data not received in due time, the loss of communication after a pre-defined delay, the functional independence of the safety-related transmission functions and the used layers of the non-trusted transmission system.

The SHL-1 system model in its environment, expressed in object-oriented notation [13], is shown on Fig. 3.

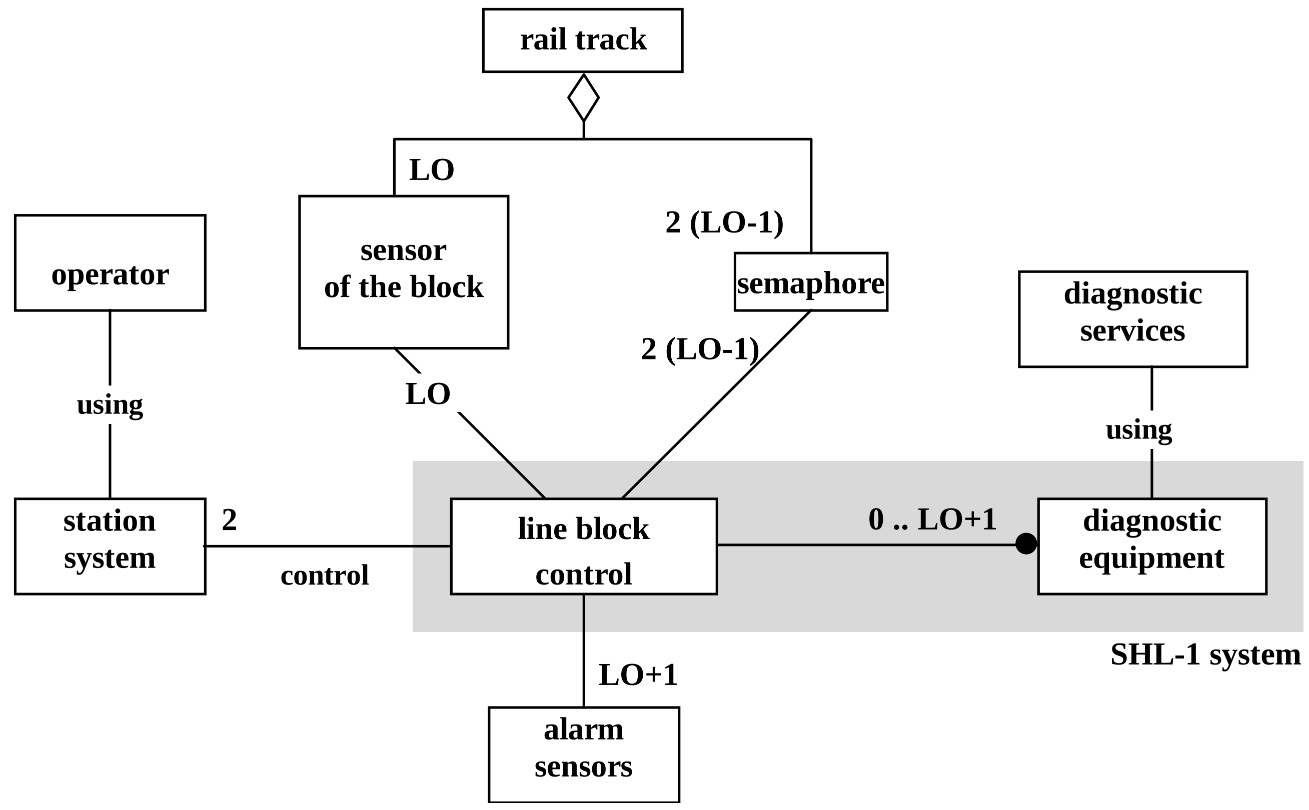

Fig. 3. Object model of SHL-1 system in its environment.
LO - number of line blocks on the line controlled by SHL-1.

5. CONCLUSIONS

The article presented the context of the FMEA and FTA techniques used in the safety assurance framework for a software intensive railway traffic control system. The next step requires strengthening the FMEA and FTA techniques by [14]:

- improving the quality of data sources applied to the process of analysis, and improving the specifications and their corresponding models,
- integration of FMEA and FTA processes with other activities (performed in parallel) of the development process,
- improving the effectiveness of the process of quality assurance of software under development.

One of possible directions is an application of object-oriented technology to the process of software design and development. The advantages of the object orientation in the context of FMEA analysis include:

- stress on univocal system decomposition and component independence (precisely written ‘contracts’ for object interfaces),
- covering multiple aspects of system modelling (e.g. the OMT methodology [13] covers the static, dynamic and functional aspects of the system),
- flexibility and stability: the models relatively easily can be modified, and

modifications are localised in a limited number of objects,

- possibility of „smooth" transition from analysis to design and code; objects identified during the analysis stage have their direct representation in implementation.

Our experience strongly emphasises the question of the analysis process effectiveness (particularly in the case of project schedule pressure). To be practical the analytical framework should be customisable to project constraints expressed mainly in terms of the project budget and schedule.

The acceptance of the results of the safety analysis of SHL-1 (it was one of the first analysis and design processes in Poland in relation to CENELEC norms) by the national certification institution was conditioned by a postulate to consider an additional fault of lower level component. Independently designed functional and safety tests disclosed one fault and the associated behaviour not covered in the analysis. In response an additional verification of the system specifications and the proper modifications of the system were performed. Successful environmental tests together with the additional explanations by the design team completed the required safety case: the SHL-1 system received its safety certificate.

Brainstorming was a technique extensively used during the analysis in this project. It was used by the in relation to the distinguished components of the architecture. In many cases the analyses were not adequately supported by the system documentation. The knowledge of the system was in many cases elicited directly from a team member who actively participated in the design or coding activities.

An important indirect effect of the safety analysis was better understanding by the engineers of the goals and requirements of the process aiming at ensuring the required level of system safety.

REFERENCES

[1] BOWEN J., STAVRIDOU V.: Safety-Critical Systems, Formal Methods and Standards. *Oxford University Computing Laboratory Technical Report*, PRG-TR-5-92, 1992.

[2] Defence Standard 00-55, Requirements for Safety Related Software in Defence Equipment (Part 1&2), Issue 1, UK Ministry of Defence, 1997.

[3] EN 50126: Railway applications. The Specification and Demonstration of Dependability, Reliability, Availability, Maintainability and Safety (RAMS), CENELEC, Final Draft version, June 1997.

[4] EN 50128: Railway applications. Software for railway control and protection systems, CENELEC, Final Draft version, June 1997.

[5] EN 50129: Railway applications. Safety Related Electronic Systems for Signalling, CENELEC, ver. 1.0, January 1997.

[6] prEN 50159-1: Railway applications - Communication, signalling and processing systems. Part 1: *Safety-related communication in closed transmission systems*, CENELEC, Final Draft, June 1998.

[7] GÓRSKI J.: Extending Safety Analysis Techniques with Formal Semantics, in: *Technology and Assessment of Safety Critical Systems (F. J. REDMILL, Ed.).* Springer-Verlag, 1994.

[8] HIGUERA R. P., HAIMES Y. Y.: Software Risk Management, *CMU, Software Engineering Institute Technical Report*, CMU/SEI-96-TR-012, June 1996.

[9] IEC 812 (1985): Procedure for failure mode and effects analysis (FMEA), TC56.

[10] LEVESON N. G., Safeware: System Safety and Computers, Addison-Wesley Publishing Co., 1995, ISBN 0-201-11972-2.

[11] OWRE S., RUSHBY J., SHANKAR N., von HENKE F.: Formal Verification for Fault-Tolerant Architectures: Prolegomena to the Design of PVS, *IEEE Trans. on Software Eng.*, vol. 21, no. 2, February 1995, ss. 107-125.

[12] PKP / CNTK, Zadanie nr 1060/23: Wymagania bezpieczeñstwa dla urzadzeñ sterowania ruchem kolejowym. Warszawa, wrzesieñ 1997.

[13] RAMBAUGH J., BLAHA M., PREMERLANI W., EDDY F., LORENSEN W.: *Object-Oriented Modelling and Design*, Prentice-Hall Int., 1991.

[14] CICHOCKI T., GÓRSKI J., Application of FMEA to Safety Analysis of Industrial Computer Applications, (*in Polish*) III Konferencja Naukowo-Techniczna „Diagnostyka Procesów Przemyslowych", Jurata k/Gdańska, 7-10 of September, 1998.