APPLICATION OF SYSTEM LEVEL ANALYSIS TECHNIQUES TO ENSURE SAFETY OF EMBEDDED SOFTWARE

Janusz Górski

Summary: The paper presents research results related to the development of extensions of the conventional safety analysis techniques like FTA and FMEA to provide for their applicability to software intensive systems. An object oriented model of the system is analyzed with the help of formalized FTA and FMEA techniques, which complements the hazard reachability analysis, performed within the abstract state space of the system. In addition to this, the object model is used to synthesize a safety monitor – the device that supervises the system at run time to provide for early detection of approaching hazards. The paper refers to the case studies that have been carried out to test the applicability of the proposed techniques. Those include a gas burner system, an extra high voltage substation and a railway signaling system.

Janusz Górski, Prof., Technical University of Gdansk, 80-952 Gdansk, Poland
Tel +48 58 347 19 09, Fax +48 58 347 27 27, E-mail: jango@pg.gda.pl

1. Introduction

Computer systems are used to control industrial processes in energy production and distribution, chemical plants, transportation, medicine, aviation, and other application domains. In many cases, software embedded in such systems can cause severe damages to humans, devices, buildings, natural environment, etc. One of the main problems of industrial application of computer systems is therefore software safety, i.e. the question if software can contribute (often indirectly, through a long chain of intermediate events) to an accident.

It is required that safety analysis is conducted during the system life cycle in order to identify safety requirements and to enforce their continuous satisfaction \(1\). A number of safety analysis techniques have been successfully used for years for conventional (i.e. non-computer based) systems, e.g. Fault Tree Analysis (FTA), Failure Mode and Effect Analysis (FMEA), Hazard Operability Study (HAZOP), and many others (an overview can be found in \(2\)). However, in many cases, these techniques cannot be directly applied to software. One difficulty is that the traditional safety analysis techniques do not adequately address problems which are characteristic for software, like discrete nature of processes, complexity, absence of physical faults, or real time constraints which are often the central issue of software-controlled industrial applications.

To overcome those difficulties one can purposefully adapt traditional safety analysis techniques to software or, alternatively, look for possible ways of extending traditional methods of software analysis with the possibility of safety analysis.

The former approach can be illustrated by several examples. Fault Trees Analysis (FTA) has been adapted for analysis of safety-critical software systems in \(3\). A method of informal safety analysis of software-based systems using FTA and FMEA has been applied in nuclear industry \(4\). HAZOP is also recommended for use with respect to computer systems \(5\). The advantage of this approach is that the original techniques have already been widely used for many industrial applications and it is easier for safety engineers to adapt to their new (extended) versions. The disadvantage is that most of the traditional safety analysis techniques are rather informal. They work relatively well while applied to physical world where we can usually apply the 'continuity assumption' that small deviation of the input will result in the corresponding small deviation in the output. Consequently, a method that only approximates the reality is in many cases highly adequate and offers sufficient analytical power. With respect to software the 'continuity assumption' is not valid and consequently the precision of analysis should be much higher to offer similar level of assurance. In addition to this, the traditional safety analysis techniques do not sufficiently address issues that are often of particular importance while analyzing the industrial software, in particular they do not support time analysis in the extent which is necessary for real-time software systems.

The latter approach is based on formal and semi-formal methods and models originally developed for the software domain. The aim is to extend those methods to make them adequate for safety analysis purposes. This includes attempts to use temporal logic, Petri nets, LOTOS, event-action models, object models and others. The advantage of this approach is that those methods are better suited to describe software and provide adequate mechanisms for analysis and verification. The disadvantage is that the methods are new to safety engineers. The safety community is rather conservative and reluctant to applying new and unchecked methods for safety analysis purposes. Another problem is that many of those methods although work well for software, do not scale up to the system level and are difficult to apply to different (than computers) parts of the safety related system. This causes the problem of integration of those methods with the system-level methods, as safety of software cannot be analyzed without taking into account the system context.
The objective of this paper is to present the results of the research aiming at combining the advantages of both of the above approaches. We worked with the traditional safety analysis methods FTA and FMEA extending them with formal semantics to provide for more analytical power. In addition to this we extended the traditional object-oriented analysis in a way which enables safety analysis of the resulting models.

2. Formalized Fault Tree Analysis

Fault Tree Analysis (FTA) is a well understood technique, widely used during analysis of safety related applications. Extending FTA with formal semantics and equipping it with the possibility to explicitly address timing adds a new value to the traditional FTA by strengthening its expressive and analytical power. Consequently, a safety engineer can continue to use the technique he/she is already familiar with and in addition may use some new features that provide for more precise and more powerful analysis. Nowadays FTA is widely applied in many industries. The main objective of the method is to analyze possible causes of a hazard (not to identify hazards). Hazard is understood as a state or a set of conditions that together with other conditions in the environment of the system, will inevitably lead to an accident /2/. Fault trees are analyzed in qualitative and quantitative ways. The objective of the qualitative analysis is to determine causes of the hazard and relations between them, while the quantitative analysis aims at evaluating the probability of the hazard. For more details one can refer to /6, 7/. FTA has been used for software systems analysis on different stages of the system life-cycle in both, qualitative and quantitative ways.

Our focus was on the qualitative system safety analysis stage which output is used to define the system safety requirements. The work presented in this paper can be related to the work of other authors researching qualitative analysis of fault trees. In /4/ a conventional FTA is used in System Safety Analysis of a robot control system to derive system safety requirements. In /8/ fault trees are specified in Temporal Logic and the model is used for system verification. This approach covers causal relations only and timing relations are ignored. In /9/ causal relations in system level fault trees are modeled with Petri Nets. If the basic fault tree events describe the behavior of a computer, one can then investigate their causes using Software Fault Tree Analysis (SFTA) /2/. SFTA is a source code analysis method. It uses backward reasoning (weakest precondition) to identify any legal path through the code that produce the output. The approach presented in /10/ assume that the system model is available and fault trees are generated from that model. This approach can be adapted for most state-machine modeling languages. The basic idea differs from the one presented in this paper as our assumption is that we do not change the conventional way of fault tree development and extend it with additional benefits resulting from formalization.

2.1 Overview of the method

The method consists of four steps.

STEP 1: Conventional Fault Tree Analysis

This is the conventional system-level Fault Tree Analysis conducted by a safety expert together with experts in the application domain. The result of this step is a set of fault trees, each representing a possible hazardous event together with its scenario expressed in terms of the contributing events and their causal relationships.

STEP 2: Fault Tree Formalization

The objective of this step is to define a fault tree in a formal way. This removes ambiguities and provides for formal analysis of the tree. The formalization is based on the model given in /11, 12/. The model supports specification of time properties of the participating events. The formalization task comprises few steps as described in /13/.

STEP 3: Minimal Cut Sets with Time Analysis

This step consists of the minimal cut set analysis. For each cut set, a time relation between the events belonging to the set is derived from the tree in the form of a formal expression, called the enabling condition of the cut set. The enabling condition describes the necessary time relationships for the events of the cut set to contribute to the hazard. Consequently, if this condition is being maintained False during the system operation, we exclude the possibility that the cut set can contribute to the hazard occurrence.

STEP 4: Software Safety Requirements Definition

In case the enabling condition can become True, we look for possible design changes which would guarantee that the condition is maintained False. In particular, if the enabling condition refers to the software initiated events then the hazard can be prevented if, throughout the system operation, the software maintains the negation of the whole condition. This observation provides for derivation of safety requirements to be fulfilled by software /14/.
3. Object-Oriented Safety Analysis

Object-Oriented Modelling (OOM) is widely used in industry, have a broad scope in a sense that it assumes the system-wide view, and have a considerable tool support. OOM takes a holistic approach were the whole system is the subject of modelling. Our work on OOM focuses on development of object-oriented models of a safety related application, from both the mission point of view and the safety point of view. The models are then integrated which enables to analyse if the mission and the safety requirements are in conflict.

Object models provide a good communication platform between the designers and the application domain specialists and are convenient means for expressing ideas of developers. Object oriented approach is often used for system mission modelling.

3.1 Overview of the method

We have developed three steps of object oriented safety analysis. Each of them addresses a separate safety aspect. The steps are not independent. They co-operate by mutual use of their results. The starting point is the mission oriented model resulting from the conventional OO analysis stage, see e.g. /15/. This model is validated and verified in order to check whether it adequately represents the application and fulfils its requirements. It answers the question “what is the system intended to do?”. From the safety standpoint, however, the relevant question is “what is the system prohibited do?”. To answer this question the model has to be modified and extended.

STEP1: Safety analysis of system mission

The main objective is to reveal design faults of a control system. On this stage the random failures of plant objects are not taken into account. To enrich the model with the safety aspect we use what we call the hazard model. It explicitly distinguishes between the safe and hazardous states. In order to apply this generic model to a specific application we identify critical objects (objects the hazard refers to) in the model of the system mission. For a given hazard, the aggregation of critical objects forms the critical subsystem. The dynamic model of the critical subsystem and the hazard model are merged to one model which explicitly distinguishes between safe and unsafe states. This model is used to analyse the consistency between the system safety and its mission. Reachability analysis is performed to check whether the hazardous state is reachable within the mission constraints. The detected hazard scenarios provide hints on how to reconstruct the system to ensure safety. More details on the method can be found in /16/.

STEP2: Analysis of impact of errors

Step 1 assumes that the objects of the plant and the environment are reliable in a sense that they behave as specified. This assumption is not always valid as the plant objects are exposed to random failures and the objects of the environment can violate assumptions concerning their behaviour. Therefore, the system should be systematically examined with the aim of identification of possible faults and checking their impact on safety. The method provides a set of templates of faulty behaviours that are defined as deviations from the normal behaviour. Within the context of a specific application, those templates are validated against possibility of their instantiation in the actual system. Those which are considered significant are then introduced to the model as alternative behaviours. Then we again use the reachability analysis to check whether these faulty behaviours have any impact on safety. If the faults are dangerous the method gives some ideas on how to restore safety either by modifications of the control system or by reconstruction of the plant. More details on this method can be found in /17/.

STEP3: Safety monitoring

Steps 1 and 2 concentrate on verifying that the system achieves its mission goals within the safety constraints. To strengthen safety guarantees the system can be additionally enriched with a device called safety monitor. Its only concern is safety. We have developed a method of systematic synthesis of such safety monitor for a given application. The idea of monitoring /18/ is simple: if a potentially hazardous sequence of events is developing in the system, inform the operator or another control system by raising an alarm. A corrective action aiming at safety restoration can be then executed before the hazardous behaviour converts into an accident. The practical approach /19/ to safety monitoring is based on the following observation: if the critical subsystem model were stimulated with exactly the same events as the actual system then reaching the hazardous state by the model would mean the hazard in the actual system. Consequently, the model of the critical subsystem constitutes the first version of the safety monitor. The method transforms simplifies this first version in the following steps:

- **reduction**: removing all safety irrelevant details from the monitor,
- **implantation**: ensuring that the monitor is stimulated by only those events that are measurable i.e. there are technical means to detect their occurrence,
• tuning: achieving appropriate sensitivity of the monitor: no overlooked hazards, false alarms avoidance and early warning.

4. Formalized Failure Mode and Effect 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). 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 leads 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.

The question arises if the FMEA analysis may be performed directly on the software implemented parts of the system. According to IEC 812 /20/, confirmed by the experience, 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 still encounters substantial difficulties.

4.1 Overview of the method

To provide for precision and unambiguity we extended the object-oriented model with its formal specification. We have represented the object structure, object states, object operations and system state invariants in terms of Z schemas /27/ and the dynamic aspects of object co-operation by using the CSP /26/ notation. Within this framework we model faults and then analyse the consequences of those faults. By a fault we mean any deviation from desired properties of objects or their interaction, including disturbances of event timing and precedence. Faults may manifest themselves as failures of components, represented as a negation of the component function’s desired result (its post-condition) or negation of any other assumption made while specifying the desired behaviour of the component (e.g. the pre-condition of the function or the state invariant of the component). In the CSP-based specification we represent them as additional fault events. Consequently the specification is extended by the cases handling the fault events occurrences.

Like in the standard FMEA, we develop complete lists of component failures. Two domains of failure are analysed and, respectively, two tables are constructed, consistent with current classification of defects in software /29, 28/: – a Data Table, involving communication failures and input deviations, used to analyse data dependencies and software interface errors, and
– an Event Table, involving process failures and constraints on software states, used to analyse the effects of failures and deviations in output, possibly caused by software that fails to function correctly.

The method is still under development. It assumes hierarchical specification of properties of a system architecture and verification of consistency between subsequent layers of the hierarchy. The method comprises the following activities /25/:

STEP 1: Specification.

This includes the following steps:

• application of object-oriented modeling to find representation of the system that includes objects (classes), attributes and associations of interest;
• building a formal specification of the object model.

STEP 2: Failure identification

• referring to the formal specification, develop an inventory of possible component failure modes,
• argue on the completeness and the validity of the list of failures.

STEP 3: Analysis.
• effects of the anticipated lower level failures are related to the higher level assumptions and failure lists. This stage concentrates on the consequences of the specification. It may involve a range of approaches, from informal reading and interpretation of (formal) specifications to proving theorems from the axioms included in the specification.

5. Case Studies
The research is based on the assumption that the methods should be demonstrated to work on realistic examples. Therefore each of the methods is tried on a realistic case study performed together with some industrial partners.

Formalized Fault Tree Analysis
The method has been applied to a common gas burner case study and to the ESFAS (Engineered Safety Features Actuation System) of the protection system of the nuclear power plant in Krško (Slovenia) /22/.

Object-Oriented Safety Analysis
The method has been applied to the gas burner case study and to the line bay of the Mościska 400/110 kV extra high voltage substation. The substation is situated near Warsaw and is an important node supplying energy to the capital city. Its main objective is reconfiguration of the power grid. The substation consists of two busbar systems and eight bays. Our case study focused on the bay number 1 through which the substation is connected to the Milosna substation.

The case studies were supported by the Statemate tool /24/ which was used for reachability analysis of the developed models. The results of the case studies can be found in /23/.

Formalized Failure Mode and Effect Analysis
The method is presently under development. It is being applied to the computerized Line Block System (LBS) which is supposed to control railway traffic on a rail-track between two stations. It uses light signals shown on semaphores, each protecting a line segment (block) of the rail-track. Each segment is tested in order to determine its occupation or its availability to move a train toward the next segment. The main goal is then to achieve separation of trains on the line and to provide for smooth passing of trains in the required direction /21, 25/.

6. Conclusion
Safety case is a documented body of evidence giving a convincing and valid argument that a system is adequately safe for a given application in a given environment. In the present practice, safety case includes a combination of arguments of different nature as no single model and the way of reasoning about its properties is commonly accepted as convincing enough to form the sole base to develop safety cases.

The structure of safety arguments is such that they accept a body of evidence and use some inference mechanism to form a “bridge” between the evidence and the safety claims about the system under consideration. Presently used safety arguments can be classified as follows:
• deterministic - the evidence is given in a form of axioms, the inference mechanism is based on logic and the argument takes a form of proof that the safety claim is the consequence of the axioms,
• probabilistic - the evidence asserts about the failure rates and the independence of failures and the argument takes a form of statistical reasoning,
• qualitative - the evidence is the demonstration of adherence to standards, design rules, good practices, or guidelines. Then the argument takes a form of some acceptance criterion based on this evidence.

The paper presented three approaches that are tested against some industrial case studies. The motivation behind those approaches stemmed from the observation that the safety industry is relatively conservative and reluctant to applying completely new and unchecked technologies. Therefore, instead of proposing methods which are completely new (with an implicit suggestion “forget everything what you were using so far, we are now giving you something which is much better to cope with your problems”), we have decided to concentrate on selection of widely used technologies which already have proven their applicability and then to invest our effort into extending and strengthening those technologies so as to equip them with additional power. This is particularly dedicated to facilitate safety analysis in presence of some problems which are characteristic for software such as: complexity, timing constraints, design faults.

Our choice was Fault Tree Analysis (FTA), Object-Oriented Modelling (OOM) and Failure Mode and Effect Analysis (FMEA). Those technologies are widely used in industry, have broad scope in a sense that they assume the system-wide view, and have a considerable tool support. They are very different in a sense that the FTA’s concentrates on possible scenarios of hazard occurrences, FMEA concentrates on failure modes of components and their consequences and OOM takes a holistic approach were the whole system is the subject of
modelling. We believe that this diversity may result in a complementary evidence which can be used to strengthen a safety case for software intensive industrial systems.

Acknowledgement

It is to acknowledge the contribution to the described research of my doctoral students Tadeusz Cichocki, Dr Bartosz Nowicki and Dr Andrzej Wardziński. The work on the Formalised FTA and OO Safety Analysis has been supported by the European Union Copernicus grant CP 94 1594.

References

10. V. Ratan, K. Partridge, N. Leveson, Generating Fault Trees from RSML Specifications
20. IEC 812 (1985): Procedure for failure mode and effects analysis (FMEA), TC56
22. M. Cepin, A. Wardziński, On Integration of Probabilistic and Deterministic Safety Analysis, 3rd Meeting Nuclear Energy in Central Europe, Portoroz (Slovenia), September 16-19, 1996