An Approach for the Reliability Analysis of Automotive Control Systems

Mikhail Glukhikh, Mikhail Moiseev
St. Petersburg State Polytechnical University
St. Petersburg, Russia
glukhikh@mail.ru, mikhail.moiseev@gmail.com

Harald Richter
Clausthal University of Technology
Clausthal, Germany
hri@tu-clausthal.de

***Abstract*—In this paper, we present an approach and a tool that automates and thereby accelerates the most time-consuming phases of reliability engineering. In this approach, an operational function is computed automatically from a high-level system description by using of system components' properties, fault types propagation rules and other auxiliary information. The tool allows arbitrary component types, any component couplings and failure types and covers thus mandatory features for a profound reliability analysis. It calculates the mean time to failure, the mean fault number and the components' influence on the overall reliability as system reliability characteristics. This tool was tested by a major car manufacturer in an embedded electronic system of a car. The main advantage of the developed tool is that it simplifies reliability analysis of complex-structured systems using a novel method for system operational state description.**

Keywords-reliability analysis; automotive control system.

I. INTRODUCTION

In reliability engineering, there are two different tasks to accomplish: the analysis of the reliability of a given system, and the synthesis of a successor system that is more reliable than the first one, under the boundary conditions of the costs needed to produce it. Reliability analysis is used iteratively many times in the process of system design. It is thus important to evaluate the reliability parameters quickly and accurately with minimal effort [1][2].

This paper presents a new approach and a tool for automating the reliability analysis of complex-structured control systems. Our approach is based on a system meta-model, which allows to represent many classes of control systems. As an example, the application of this approach for automotive control systems is given in this paper.

Analyzing the reliability of car electronics becomes more and more important because of two reasons. First, the number of Electronic Controller Units (ECUs) that are built into contemporary cars has already reached the amount of 100, thus reducing the mean time to failure (MTTF) by the pure quantity of components. Second, the passengers' safety depends more and more exclusively on the reliability of ECUs' hardware and software, together with other components such as sensors, actuators, cable trees and connectors, power supplies, generator and battery [3][4].

Because of the used general methodology our approach is not restricted to cars only but can also be applied to other technical systems, such as in medicine, aerospace and nuclear power plants, where harsh environment conditions are prevailing or where system breakdown is unacceptable.

The rest of the paper is organized as follows: Section 2 shows state-of-the-art, Section 3 summarizes related work in this field, Section 4 describes the main idea of our approach, Section 5 presents its specialization for automotive applications, Section 6 describes the used reliability analisys methods, the tool itself is explained in Section 7, the paper ends in Section 8 with a conclusion and an outlook to future work.

II. STATE-OF-THE-ART

The goal of reliability engineering is to achieve either a prescribed reliability for a planned system or a higher reliability at lower costs. State-of-the-art to achieve this is to change the system structure, to add more reliable components, or to add redundancy such as standby reserve, hot reserve, standby containers and load sharing containers.

Reliability analysis of a technical system normally needs many iteration cycles in which the so-called survival or reliability function is calculated multiple times, together with other important parameters such as the MTTF, the mean fault number and the components' influence on the overall reliability. These parameters allow to identify weaknesses in the system design. The reliability function is a probabilistic function over time that is based on the individual failure rates of the system components. For the calculation of the reliability function, the system has to be modelled first. Standard models are reliability block diagrams (RBDs), as well as fault trees (with or without Markov chains) and event trees as alternatives [5]. Another common model is the operational function [6] which is used by our approach.

In practice, automotive control systems are complex with respect to their interconnect topology between components and because of the sheer amount of parts. High-end cars have already several thousands of cables and connectors. Furthermore, there are different types of data paths and power lines that couple the components together, and many component types that have to be differentiated as well.

Typical automotive control systems can not be decomposed into a set of elementary serial or parallel circuits of components. Instead, loops and crossings of links exist in

 ISBN: 978-1-61208-149-6

the interconnect topology. The system topology is thereby considered to be a general graph.

State-of-the art in current tools is that the constructing of reliability models is hardly automated yet. However, the manual constructing of the models mentioned is too time-consuming in practice because of the large size and the high complexity of real-world system topology. We propose an automated approach based on a configurable high-level system description and component parametrization.

III. Related Work

Beside research projects, there are several commercially avaliable tools for the reliability analysis of complex systems such as [11][12][13][14][15].

Usually, these tools support a comprehensive range of analysis methods. According to [5], the most common methods are fault tree analysis (FTA), which may be preceded by an RBD system model, and event tree analysis (ETA). FTA can be combined with Markov chains (MC) too. The most important features of these commercial tools are depicted in Table 1.

Table I
Reliability analysis tools

Tool	RBD	ETA	FTA	MC
ITEM ToolKit	+	+	+	+
RAM Commander	+	+	+	+
Isograph FaultTree+	–	+	+	+
PTC Relex	–	+	+	+
ReliaSoft BlockSim 7	+	–	+	–

All tools suffer from the manual definition of the fault tree for FTA. Already from 10 components on, the effort for fault tree construction by hand is high because its time-complexity grows exponentially. In some cases known from practice, a preceded RBD with subsequent conversion into a fault tree may simplify this task [9]. However, in complex cases, the RBD construction complexity is already comparable with fault tree construction complexity.

The other model used for fault tree synthesis automation is Fault Tolerant Data Flow (FTDF) [10]. Fault tree synthesis algorithm traverses FTDF graph finding all event combinations which lead to system failure. However, this algorithm does not support cyclic dependencies between elements and uses exhaustive search for synthesis.

Reliability parameters are usually calculated by using the system reliability function. There are several possibilites to obtain the reliability function out of a previously established fault tree or out of an operational function. These possibilites are:

1) Selection of a minimal-cut set in the disjunctive normal form of the boolean description of the fault tree and subsequent use of the inclusion/exclusion principle as described by [5][7].
2) Establishing a binary decision tree as an intermediate data structure of the fault tree as proposed by [7][8].
3) Using a substitution form of the operational function, together with boolean-probabilistic transformation rules that are described in [6][1].

All three ways exhibit in the general case an exponential complexity as soon as the system size increases. However, we found out that in practice the boolean-probabilistic method has good scalability which is why we used it in our tool.

IV. Semi-Automated Analysis

We suggest a new approach for the semi-automatic reliability analysis that consists of three major steps:

1) An abstract meta-model.
2) An application-specific configuration of the meta-model by a so-called pattern.
3) A set of boolean-probabilistic transformations.

Our first step employs an abstract meta-model instead of a concrete RBD, fault tree or structure graph. This meta-model is defined by the tuple $M = \langle T, G, I, F, O \rangle$, with T is the set of component types, G is the graph of the system topology, I is the set of failure types each component type can have, F is the set of failure propagation rules for every failure type, and O is a set of rules for constructing the operational function.

Step 2 in our approach yields a concrete description of a given system from the meta-model, after having configured the model with an application-specific pattern. The pattern has to be created by hand, together with the system's topology. The pattern we used for automotive control systems is given in detail in Section 5. Other control systems need different patterns but the meta-model can remain the same.

Because of the time-consuming manual definition of a fault tree its generation should be automated. We decided to use an operational function for the system operability description (see Section 6). From the operational function, a reliability function can be derived automatically. To achieve this, boolean-probabilistic transformations are employed.

The approach requires in detail the following 6 phases:

1) Definition of all component types, failure types, failure propagation rules and operational function constructing rules as the application-specific pattern.
2) Construction of the graph of the system topology.
3) Definition of all failure rates.
4) Derivation of the operational function.
5) Conversion of the operational function into the reliability function.
6) Computation of the MTTF, the mean fault number and the components influence on the overall reliability.

The first phase must be performed once by hand for each application-specific pattern. Furtherfore, for any given system, phases 2-3 also have to be performed manually once.

 ISBN: 978-1-61208-149-6

Phases 4-6 are computed automatically. These are the steps that are repeated multiple times for reliability engineering which is why we automated their execution.

V. Pattern for Automotive Applications

Modern cars may contain the following assistance systems for driver and infotainment: motor management, electronic stabilization (ESP) with or without active steering, adaptive cruise control (ACC), speed control, distance control, rear vision, night vision, lane keeping, lane changing, parking, navigation etc. These systems can be analysed by the subsequently described pattern.

A. System elements

There were the following element types defined by us:

1) Electronic Controller Units (ECUs).
2) Active and passive gateways and connection lines that provide for data propagation.
3) Power supply generators, batteries, power lines, connectors and fuses that provide for power propagation.
4) Sensors and actuators.

B. System Topology

In this pattern, the system topology is defined via two subgraphs, one for the data paths and one for the power lines of the system. Both graphs have the same instances of component types.

C. Failure Types

The automotive pattern defined by us contains the following failure types:

1) Internal Error. This type is valid for ECUs, active and passive gateways and data paths. In case of an internal error, the component does not perform its function. It outputs therefore no data or even wrong data.
2) Silence Error. This type is again valid for ECUs, active and passive gateways and data paths. In case of silence error, the component outputs no data although it should. However, it does not output wrong data.
3) Babbling Error. This type is valid for ECUs and active gateways. In case of babbling error, the component continuously outputs data although it should not. Most or all data are wrong.
4) Short Circuit Error. This type is valid for power supplies, power lines, generator, battery, fuses and data paths.

D. Failure Propagation Rules

Some failure types can propagate through the system. Their propagation characteristics are predefined in the automotive pattern as:

1) The babbling error propagates through data paths and passive gateways. It can not propagate through ECUs and active gateways.
2) The short circuit error propagates through power lines and data paths. It does not propagate through fuses.
3) The internal error and the silence error can not propagate.

E. System Operational Function

In the following, it is assumed that all tasks of the control system can represented by a set of functions. Then, the system operational function is the boolean AND of all these functions. This means that the system function F_{sys} is defined by the functions of its components, according to $F_{sys} = \bigwedge F_i$. Every function F_i in turn relies on the well-functioning of one or several ECUs. Finally, each ECU needs a power supply and may need input data from other ECUs.

VI. Reliability analysis methods

A. Operational Function

The system operational function is a boolean expression, that represents the operational states of its individual components. In this function a "1" denotes an operable state. For the calculation of the operational function, let us consider a system comprising of n elements – $C_j, j \in 1, 2, \ldots, n$.

Let X_j be the boolean variable that indicates whether the element C_j of the system is operational. Then, the system operational function is constructed by the means of the basic rules O, the component types T ($\forall C_j : \exists! T_i \in T : C_j \in T_i$ – C_j of type T_i) and the component connection graphs G for power and data lines. Also the operational functions F_i of the individual components are constructed by G, as well as by the failure types I and the failure propagation rules F. Additionally, it has to be taken into account that there are dependencies between components, and as a consequence, a linear system of equation has to be set-up and solved (the unknown variables in this system are the X_j). Every F_i in turn is determined by its corresponding ECU type T_i. F_i is "1" if there is at least one operable component C_j of type T_i: $F_i = \bigvee_{\forall j: C_j \in T_i} X_j$, $X_j = x_j X_j^{power} X_j^{data}$. In this equation, the used variables have the following meaning:

- x_j – is operable state of element C_j.
- X_j^{power} depends on the power supplies that are connected to C_j by power lines and fuses. The analysis of the power supplies must take into account that short circuits can propagate.
- X_j^{data} is "1" if input is available from all necessary providers. Let C_k be a data provider for component C_j then X_j^{data} contains the component $X_k P_{k,j}$, where $P_{k,j}$ is the operable function of the data paths. In addition, $P_{k,j}$ must consider babbling errors that propagate through data lines and passive gateways.

Subsequently, the equation system of all X_j has to be set-up. In the left part of each equation is a X_j according to $X_j = \mathcal{F}(X_1, ..., X_n)$. However, X_j can also be an independent variable of $\mathcal{F}$ on the right side of the equation.

The system of equations is therefore solved by exclusion-of-unknowns method which means that X_j in $\mathcal{F}$ is replaced by "1" (correctness of this action is proven). An example of this process is given in Section 7.

B. Reliability Function

The reliability function has independent stochastic variables that denote probabilistic failure events of components. For the computation of the reliability function out of the operational function, a boolean-probabilistic method [6][1] is used. This method includes the following two steps:

1) Conversion of the operational function into an equivalent form called "substitution form".
2) Conversion of the substitution form to the reliability function by applying boolean to probabilistic transformation rules according to [6][1].

1) Substitution Form: The substitution form of the operational function F_{sys} is the boolean sum of the orthogonal and repetition-free summands that are concatenated with AND/NOT only. To explain that, let us consider the conjunctive form (which is not normalized) of the operational function $F_{sys} = \bigwedge F_i$, where $F_i = \bigvee_{\forall j:C_j \in T_i} X_j$. This form is converted into the disjunctive normal form by applying the distributivity law. After that, each summand is made orthogonal to every other such that the boolean multiplication of any summand pair yields "0". After that, the orthogonal disjunctive normal form is augmented by the property that every x_k occurs only once in a summand. This is achieved by successively factoring out all x_k in each summand that occurs more than once. Finally, all x_k have to be concatenated with AND and NOT operators only. This is accomplished by applying de Morgan rules. For example, $F_{sys} = \overline{\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3}}$ is a repetition-free form with AND/NOT. This example is also considered to have orthogonal summands because it consists only of one summand.

For the computation of the reliability function, that substitution form of F_{sys} is used because it allows boolean to probabilistic transformations. However, the described procedure to obtain the substitution form is too time-consuming in practice which is why we use a shortcut to obtain it. This shortcut is called Cutting Algorithm [6].

2) Cutting Algorithm: The goal of the Cutting Algorithm is to reduce in every step the number of arguments x_k of the operational function F_{sys} by one. The general form of the cutting algorithm is:

$$\begin{aligned} f(x_1, x_2, ..., x_i, ..., x_n) = \\ = \overline{x_i} f(x_1, x_2, ..., x_{i-1}, 0, x_{i+1} ..., x_n) + \\ + x_i f(x_1, x_2, ..., x_{i-1}, 1, x_{i+1} ..., x_n), \end{aligned} \quad (1)$$

Every reduction step is applied to that argument which occurs most often in F_{sys}. This argument is factored out according to (1).

For example, $F_{sys} = x_1x_4 + x_2x_5 + x_1x_3x_5$ is a boolean function of 5 independent variables x_1 to x_5 comprising of 3 summands from which x_1 and x_5 are the most frequent ones. The algorithm factors out x_1 in the first step according to: $F_{sys} = \overline{x_1}x_2x_5 + x_1(x_4 + x_3x_5 + x_2x_5)$. Now, x_5 is the most frequent variable, etc. The algorithm stops after one more step in which x_5 have also been factored out, yielding the result: $F_{sys} = \overline{x_1}x_2x_5 + x_1\overline{x_5}x_4 + x_1x_5(x_2 + x_3 + x_4)$.

Finally, AND/OR is replaced by AND and NOT giving: $F_{sys} = \overline{x_1}x_2x_5 + x_1\overline{x_5}x_4 + x_1x_5\overline{\overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4}}$. This is the desired substitution form of F_{sys}, i.e. of the operational function. Each summand is orthogonal to every other, and each x_k occurs only once in every summand.

As one can see, factoring out is achieved at the expense of doubling the number of summands. If N is the number of arguments in F_{sys}, then the algorithm terminates at most after N steps. Therefore, after N steps, the number of summands can be grown by a factor of 2^N. In order to make the cutting algorithm usable, some optimizations have to be introduced additionally.

3) Optimization: Two kinds of optimizations are used to reduce the execution time and memory requirements of the cutting algorithm. These two optimization types are:

1) Simplification of the operational function by boolean algebra rules.
2) Idenfifying independent segments in the system model.

The boolean simplifications have to be employed in every step of the algorithm, while the identifying of independent segments happens only once.

Independent segments are parts of the system which have an own power supply or parts of the system who have regions with local data traffic only. Each segment corresponds to independent part of the operational function. Independent parts can be obtained formally by the following procedure:

1) Find in F_{sys} boolean expressions that occur more than once.
2) Exclude from them those expressions who have the same arguments x_i also in other expressions of F_{sys}.
3) Replace the remaining expressions by substituting them with new functions y_i.
4) Treat y_i as new independent argument in F_{sys}.

For example: $f = (x_1 + x_2x_3 + x_4x_5x_6)(x_1x_2 + x_3 + x_4x_5x_6)$ can be converted into $f = (x_1 + x_2x_3 + y)(x_1x_2 + x_3 + y)$, with $y = x_4x_5x_6$ as new function.

The result of the optimization is that in a system of 100 components, for example, segment splitting reduces time and space complexity by a factor of 5-10, and function simplifying accelerates by an other factor of 3-5, resulting in a cycle time that is 15-50 times quicker than before.

4) Boolean to Probabilistic Transformation: For the subsequent computation of the reliability function, two more phases have to be accomplished. These phases are:

1) Replacement of all variables x_k and boolean operators by stochastic variables p_k and probabilistic operators.
2) Replacement of all p_k with $e^{-\lambda_k t}$, where λ_k is the failure rate of component C_k and t is the independent variable, i.e. argument of the reliability function.

For the replacement of boolean variables and boolean operators, all ANDs in the substitution form of the operational function are replaced by the multiplication of probabilities, all boolean ORs are replaced by the addition of probabilities, and all boolean NOTs are replaced by the $1 - P$ operator, where P is the probability that an event occurs.

These rules are based on the addition and multiplication rules for independent events. Finally, every stochastic variable p_k is replaced by $e^{-\lambda_k t}$ thus obtaining the reliability function $R(t)$. For example, $f = \overline{x_1}x_2 + x_1$ is transformed first into $P = (1 - p_1)p_2 + p_1$, and then into $R(t) = (1 - e^{-\lambda_1 t})e^{-\lambda_2 t} + e^{-\lambda_1 t}$.

Out of the reliability function, the MTTF, the mean fault number MFN in the interval $[T_{min}, T_{max}]$ and the components influence CI_k on the system reliability are automatically computed by the following formulas:

$$MTTF = \int_0^\infty tR(t)dt \tag{2}$$

$$MFN = \frac{(R(T_{min}) - R(T_{max}))(T_{max} - T_{min})}{\int_{T_{min}}^{T_{max}} R(t)dt} \tag{3}$$

$$CI_k(t) = R_k^*(t) - R(t) \tag{4}$$

$R_k^*(t)$ is the reliability function of the system under the assumption that C_k is 100% reliable, i.e. $x_k = 1$. So, for the computation of $CI_k(t)$, $R(t)$ has to be computed twice, one time with C_k in the system, the other time without. Furthermore, all $CI_k(t)$ are calculated at the time point T of MTTF and are subsequently normalized to $[0, 1]$.

VII. Reliability Analyzer Tool

The reliability analyzer tool implements the meta-model with the automotive pattern. All pattern data can be conveniently entered, extended or modified. This is accomplished by means of several menus based on tabs for defining pattern values, as well as by a graph editor, by a component type editor and by other features. All patterns are stored in XML format. Analysis results can be displayed by a chart viewer. The components influence CI_k on the system reliability is color-coded in the viewer in order to inform the users eyes quickly.

Let us consider as an example Figure 1 which contains 4 ECUs (E_1 - E_4), 2 batteries (A and B), one active gateway (G), one passive gateway (P) and 2 fuses (U_1 and U_2).

The elements E_1 and E_2 are of type T_1, E_3 and E_4 are of type T_2. The system operational function is $F_{sys} = F_1$, $F_1 = X_1 + X_2$, where X_i is the operational function of element E_i. The elements E_1 and E_3 have no input data from other

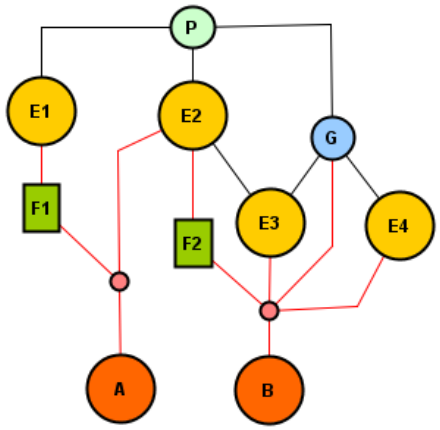

Figure 1. Example of an automotive control system

elements. Element E_2, however, uses data from elements of type T_2, Element E_4 uses data from other elements of type T_1. Under the assumption that all data paths are absolutely reliable, the operational functions of these elements are:

$$X_1 = E_1 \cdot (A \cdot U_1),$$

$$X_2 = E_2 \cdot (A + B \cdot U_2)(X_3 + X_4 \cdot G \cdot P),$$

$$X_3 = E_3 \cdot B,$$

$$X_4 = E_4 \cdot B \cdot (X_1 + X_2) \cdot G \cdot P.$$

This equation system is solved first for the unknowns X_2 and X_3 by applying the mentioned exclusion-of-unknowns method:

$$X_2 = E_2 \cdot (A + U_2) \cdot B \cdot (E_3 + E_4 \cdot P \cdot G),$$

$$X_4 = E_4 \cdot B(E_1 \cdot A \cdot U_1 + E_2(A + U_2)B(E_3 + E_4 \cdot P \cdot G)).$$

From that, the system operational function is achieved which is afterwards converted into substitution form, and then into the reliability function. All other reliability parameters are computed out of the latter. The result of the example is shown in Figure 2. The chosen example was simple but it showed the main points. The analysis of larger systems is accomplished accordingly. Our tool was tested with real-world automotive control systems, that were given to us by one of the leading car manufacturers. It has helped to compare various alternative control system architectures and to select the one with the needed reliability.

The current implementation of the tool can analyze systems of up to several hundreds of components on a PC with AMD Athlon CPU with 2,4 GHz and 256 MB of RAM. We measured the following computation times for systems of different sizes (Table II).

The tool has some overhead compared to commercial tools because of the automated construction of the system

 ISBN: 978-1-61208-149-6

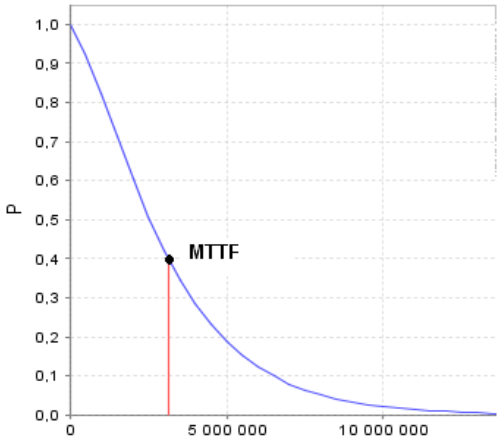

Figure 2. MTTF and the reliability function

Table II
ANALYSIS TIME

Number of components	10	20	30	50	100
Analysis time, sec	1	3	11	57	292

operational function and the automatic conversion of the operational function into the reliability function. However, this overhead can be neglected.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we proposed a new approach and tool for reliability engineering. Our approach is based on a meta-model that allows to many classes class of systems by configuring the model with so-called patterns. The pattern for automotive applications are given by us. It was implemented in our reliability analyzer tool. The tool was used by a large car manufacturer to improve the control system of a automobile. The approach constructs semi-automatically the operational function of the system under test by using information about the system's structure. Our tool computes the operational function, performs an automatic conversion to the reliability function and determines the mean time to failure, the mean fault number and the components' influence on the overall reliability as reliability parameters.

In the future, we will extend the tool to further accelerate the design cycle for reliability engineering by automating the synthesis of redundant systems. Furthermore, we will extend our approach and tool to other application areas outside of automotive control systems.

REFERENCES

[1] G.N. Tcherkesov, *Hardware-software systems reliability.* St. Petersburg: Piter, 2005.

[2] M. Rausand and A. Hoyland, *System Reliability Theory. Models, Statistical Methods and Applications.* Hoboken, NJ: John Wiley & Sons, Inc., 2004.

[3] R. Bosch, *Automotive Electrics and Automotive Electronics, Completely Revised and Extended.* Hoboken, NJ: John Wiley & Sons, Inc., 2007.

[4] T. Denton, *Automobile Electrical and Electronic Systems.* Burlington, MA: Elsevier, 2004.

[5] C. Ericson, *Fault Tree Analysis – a History.* Proceedings of the 17th International Systems Safety Conferencem, 1999, pp. 1-9. http://www.fault-tree.net/papers/ericson-fta-history.pdf. – Retrieved 2011-06-04.

[6] I.F. Ryabinin, *Reliability and safety of structural-complex systems.* St. Petersburg: Polytechnika, 2000.

[7] J.D. Andrews, *An Analysis Strategy for Large Fault Trees.* Proceedings of the 21st International System Safety Conference, August 2003, pp. 375-386.

[8] L.M. Bartlett, *Progression of the binary decision diagram conversion methods.* Proceedings of 21st International System Safety Conference, August 2003, pp. 116-125.

[9] ReliaSoft Corporation. *Comparison of RBD and Fault Tree Simulation.* HotWire Issue 44 (October 2004). http://www.weibull.com/hotwire/issue44/ relbasics44.html. – Retrieved 2011-06-04.

[10] M. McKelvin, G. Eirea and C. Pinello et al., *A Formal Approach to Fault Tree Synthesis for the Analysis of Distributed Fault Tolerant Systems.* Proceedings of the 5th ACM international conference on Embedded software, ACM Press., 2005, pp. 237-246.

[11] Isograph. *Isograph Reliability Analysis Software.* http://www.isograph-software.com. – Retrieved 2011-06-04.

[12] ITEM. *Reliability Software from ITEM.* http://www.itemsoftware.com. – Retrieved 2011-06-04.

[13] ALD. *Advanced Logistics Development.* http://www.aldservice.com. – Retrieved 2011-06-04.

[14] PTC. *The Product Development Company.* http://www.ptc.com. – Retrieved 2011-06-04.

[15] Reliasoft. *Reliability Software.* http://www.reliasoft.com. – Retrieved 2011-06-04.

 ISBN: 978-1-61208-149-6