Fuzzy-enabled Failure Behaviour Analysis for Dependability Assessment of Networked Systems

Barbara Gallina
MRTC, Mälardalen University,
Västerås, Sweden
barbara.gallina@mdh.se

Aleksandar Dimov
University of Sofia,
Sofia, Bulgaria
aldi@fmi.uni-sofia.bg

Sasikumar Punnekkat
MRTC, Mälardalen University,
Västerås, Sweden
sasikumar.punnekkat@mdh.se

Abstract— **Dependability assessment of networked component-based systems requires fine-grained modelling of the failure behaviour and propagation aspects of individual components. We have recently introduced a formalism called FI^{4}FA, enabling the analysis of I^4 (incompletion, inconsistency, interference and impermanence) failures as well as the analysis of the corresponding mitigations.**

FI^{4}FA, like its predecessors, assumes that the failure behaviour of individual components is well defined in a deterministic way. However, in reality there exist multiple sources of uncertainty (e.g. in the estimates of individual component failure attributes, in the semantics of their composition, etc.) that require consideration.

In this paper, we propose a new technique for the assessment of the failure behaviour of networked component-based systems. The proposed technique builds on FI^{4}FA and incorporates the specificities of failure behaviours of networked systems in the I$^{4.}$ The specification of the failure behaviour of individual components is based on fuzzy sets, which have the potential to model and thus address the uncertainty aspect.

Keywords-networked systems-specific failure behaviour, dependability assessment, uncertainity, failure propagation, fuzzy-based failure behaviour analysis

I. INTRODUCTION

Dependability assessment and evaluation had been a growing area of research for several decades, with a multitude of techniques and tools being proposed by academia as well as industry. The pervasive nature of software systems and their growing importance (especially of embedded systems) in our daily lives make their dependability assessment a necessity. However, technological advances as well as changing models of interactions bring forth newer dependability threats (i.e. faults, errors and failures), thus making such assessments obsolete at a faster pace than being developed. The recent advances in component-based software development pose a great opportunity in approaching the assessment of dependability in a composable manner provided the nature of the interfaces and interactions between the components are well-understood and formally characterised.

The literature (e.g. in [12] and [2]) offers techniques that exploit these advances and analyse the system's failure behaviour in a composable way. FI^{4}FA [2], for instance, is one of them. FI^{4}FA enables the analysis of I^4 (incompletion, inconsistency, interference and impermanence) failures as well as the analysis of the corresponding mitigations. FI^{4}FA, however, makes an assumption that is rarely true: it assumes that the failure behaviour of individual components can be defined in a deterministic way. FI^{4}FA does not take into consideration the multiple sources of uncertainty that can prevent a deterministic evaluation.

In this paper, we are interested in building on FI^{4}FA and extending it to tackle the uncertainty aspect. Our aim is to offer a technique to the system designer, in the context of networked systems, to assess the failure behaviour of the composed system. The main contribution of this paper is such a technique that allows the designer to take into account the knowledge related to the failure behaviour of individual components and the system architecture.

The paper is organised as follows: Section II provides the essential background of FI^{4}FA technique as well fuzzy set theory, upon which lies the foundations of our proposal. Section III presents our proposal. Finally, Section IV concludes the paper and provides a brief road map of planned future works.

II. BACKGROUND

In this section, we briefly present the essential background on which we base our work to achieve a new technique to analyse the failure behaviour of component-based networked systems. We take into consideration the uncertainty that can be introduced in the estimations. In particular, in Section II.A we present the recently introduced technique, called FI^{4}FA, for failure behaviour analysis and, in Section II.B, we present the fuzzy sets, which permit the uncertainty to be modelled.

A. F I^{4}FA

FI^{4}FA [2] is a technique to analyze the failure behaviour of an entire system given the failure behaviour of its components. FI^{4}FA builds on the Fault Propagation Transformation Calculus (FPTC) [3] technique. A twofold motivation makes FI^{4}FA more attractive in comparison with FPTC: FI^{4}FA allows users to carry out a more fine-grained analysis of the failure behaviour and it also enables the analysis of the mitigation behaviour. FI^{4}FA makes possible a more fine-grained analysis by introducing additional failure types, which enable the analysis of Incompletion, Inconsistency, Interference and Impermanence (I^4) failures (which are typically of concern in transactional systems). Similarly, the analysis of the mitigation behaviour is made possible thanks to the introduction of mitigation types.

The failure behaviour of a component is specified as a collection of expressions and each expression is composed of two parts: the left-hand-side part specifies the behaviour received in input and the right-hand side specifies the behaviour in output. The behaviour on each input port may be normal or failure behaviour. A normal behaviour on an output port may be the consequence of the successful application of mitigation means. The failure behaviour may be due to the failure types, mentioned in Section III.A, or a combination of them.

Similar to FPTC, FI^{4}FA allows a user to evaluate from a qualitative point of view the failure behaviour of the whole system. Both techniques make a strong assumption: they consider that given a specific input the behaviour on the output is deterministic. As discussed in Subsection III.B, due to multiple sources of uncertainty, this assumption is not realistic and therefore FI^{4}FA should be further developed to allow users to take into consideration the non-determinism.

B. *Fuzzy sets and logics*

In classical set theory, it is said that a given element either belongs to a set or it does not. Fuzzy sets [5] enrich classical set theory with the ability to express that an element may belong to a set in some extent. In other words, fuzzy sets enable reasoning about the belief that an element belongs to a set with a degree in the interval between zero and one. Formally, fuzzy set is defined as a pair $F=(A, m)$, where A is a set and $m: A \rightarrow [0, 1]$ is a membership function of F. For each x in A, $m(x)$ is called the grade of membership of x in F.

Generally speaking, membership functions characterize the extent to which an element belongs to a set. As a rule, they accept non-negative values. Fuzzy sets differ from statistics and probability theory in the sense that the area under the curve of a membership function need not be equal to one (like in probability density functions) and may be any value between 0 and ∞, including them. Another distinction between the fuzzy set theory and the classical one (actually, the entire theory of classical mathematics) is that a member of a fuzzy set may assume two or more (even conflicting) membership values. For example, as shown in Fig. 1, the number *a* may be regarded both as *small* and *big* to some extent.

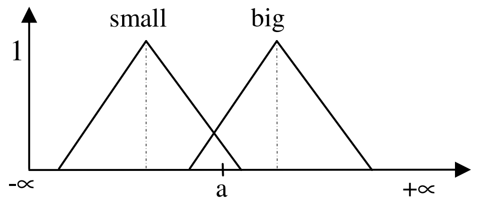

Figure 1. Fuzzy set membership functions

Fuzzy logic is based on the assumption that every variable may be *true to some extent*. In fuzzy logic, the traditional Boolean values of *true* and *false* are enriched. More specifically, a fuzzy variable is associated with a tuple of values (μ, ν) in the interval $[0, 1]$. μ is the degree of belief that the variable is true and ν is the degree of belief that the variable is not true. Since in classical fuzzy sets $\mu+\nu=1$, the ν information is usually omitted.

Fuzzy logic allows to model uncertainty in establishing the ownership to a set.

Let $x=(\mu_x, \nu_x)$, $y=(\mu_y, \nu_y)$ be two fuzzy variables. The NOT, OR and AND operators of Boolean logic, in fuzzy logic, are usually defined by the following equations:

$$\neg x=(\nu_x, \mu_x) \tag{1}$$

$$\begin{gathered}\mu_{x\cup y}=max(\mu_x, \mu_y)\\ \nu_{x\cup y}=1-\mu_{x\cup y}=min(\nu_x, \nu_y)\end{gathered} \tag{2}$$

$$\begin{gathered}\mu_{x\cap y}=min(\mu_x, \mu_y)\\ \nu_{x\cap y}=1-\mu_{x\cap y}=max(\nu_x, \nu_y)\end{gathered} \tag{3}$$

III. PROPOSED TECHNIQUE

In this section, we present a new technique to evaluate in a fine-grained way the failure behaviour of component-based, networked systems on the basis of the failure behaviour of single components. Our technique permits users to take into consideration uncertainty by combining fuzzy sets with failure behaviour analysis techniques. In the following subsections, we present: a) the failure types addressed; b) the types of uncertainty's sources; c) the integration of fuzzy knowledge in the FI^{4}FA technique and d) the process to be followed to apply our new technique.

A. *Failure types and mitigations in networked systems*

A packet transmission in networked systems can be seen as a transaction in transactional systems. Similar to a transaction, a packet transmission is also supposed to make the system transit from a consistent state to another consistent state. A consistent and successful state-change takes place when from the initial state (in which the packet to be transmitted is located at the source) the system transits to a state in which the packet is located at the destination. Ideally, the packet moves from source to destination through the transmission channel as if it were an instantaneous transition and as if it were the only user of the channel.

In reality, since the transition cannot be instantaneous and since the channel is a shared resource, inconsistent state-changes may take place. A packet can be subjected to multiple threats while it is being transmitted. More specifically, in networked systems, global consistency can be hindered by the following types of threats [1]:

- *loss*: the packet disappears from the channel/destination point.
- *cut*: some adjacent bits of the transmitted packet disappear from the channel.
- *duplicate*: the transmitted packet is sent twice.
- *bit corruption*: some bits of the transmitted packet are flipped.
- *collision*: the channel is used by more than one source at the time.
- *late*: the packet reaches the destination late.
- *early*: the packet reaches the destination in advance.

Taking into consideration the causality chain that inter-relates the dependability threats in complex systems (e,g.

hierarchical component based-systems) [13], these threats in some circumstances can be classified as failures and in some others as faults. It is well known that a failure in one component can represent a failure in another. Thus, as explained in [2], we only use the term failure.

The transaction-oriented point of view can also be used to interpret the failure types that may threat the correct behaviour of the transaction (packet transmission). In particular, we propose the following interpretation:

- a loss can be seen as an impermanence failure;
- a cut can be seen as an incompletion failure;
- a duplicate can be seen as an incompletion failure;
- a collision can be seen as an interference failure;
- a bit corruption can be seen as an inconsistency failure;

This interpretation permits techniques conceived for transactional systems to be reused in the context of network-systems.

To be able to avoid or mitigate these failures, it is necessary to detect them. In case of availability of means to carry out a fine-grained detection, corresponding fine-grained counter-measures need to be proposed. The above-mentioned failures can be addressed through the following mitigation strategies:

- *collision avoidance*, which conceptually is equivalent to serializability, allows the system to face interference failures.
- *(partial) retransmission*, which conceptually is equivalent to (compensation) all or nothing semantics, allows the system to face incompletion as well as impermanence.
- *(partial) packet reconstruction* which conceptually is equivalent to (partial) full consistency, allows the system to face inconsistency.

Since, conceptually, these types of failures and their counter-measures present interesting analogies with the failure types and counter-measures in the framework of transactional systems, techniques that might be used to evaluate the failure behaviour of transactional systems might be used in the context of networked systems as well.

B. Uncertainty in failure information collection and modelling

Zhang and Pham [15] state that the factors of uncertainty in software systems include software characteristics such as program complexity, test coverage, development environment and many others, appearing during the development lifecycle. In the specific context of software reliability, [14] defines uncertainty as a deviation of the reliability estimate given by the model, from the 'true' reliability of the system.

In the context of this paper, uncertainty is meant as the impossibility to establish exactly the ownership to a class. Uncertainty inherently characterizes the failure modelling task and therefore, while modelling the failure behaviour, only a belief can be given. In this section, we discuss possible types of uncertainty sources in failure modelling. These sources require to be taken into consideration to provide more realistic measures:

- *Uncertainty due to ambiguity in judging the component behaviour (failure vs. normal behaviour)* - The oracle, in charge of establishing whether a component behaves as expected or not, may fail or may be imprecise.
- *Uncertainty due to ambiguity in failure classification* – Different users may classify one failure as belonging to different failure types. For example, a cut during a packet transmission, for instance, can be perceived in various ways according to the user. A user for whom the lost information is the only one that matters classifies the cut as an omission, whereas a user for whom the lost information is not essential classifies the cut as a normal behaviour. In some cases, a user for whom the lost information is partially useful may classify the cut as an incompletion, Due to this diverse possible user viewpoints, a failure cannot be classified with certitude. In component-based systems, a component may be used to compose different types of systems. Its behavioural specification should thus reflect this source of uncertainty, which cannot be simply neglected by considering a single-user-point-of-view.
- *Uncertainty due to ambiguity in the component behaviour modelling (transformation) rules* - Here bias in the individual judgments of the designer could be an issue. This is due to the fact, that there does not exist an exhaustive and well-defined theory behind transformation rules, which leaves room for different interpretations by the system designers.
- *Uncertainty due to ambiguity in the estimation of component failure properties* - The individual component failure behaviours are estimated using diverse sources of data. The way in which we collect these data may be one of the following: (1) based on expert decisions during design; (2) based on test data; (3) based on real-world (field) operation of the software. Combining all such multiple sources of inputs and synthesizing them to get reliable estimates are often time consuming or even infeasible, which leads designers to use a single source of information adding to uncertainty in the estimation.

All of the above-listed uncertainty sources may be experienced into real practice either in the phase of collection of failure data (for instance by testing) or in the phase of failure modelling.

C. Integration of fuzzy knowledge in FI^{4}FA

In this subsection, we motivate the need for the incorporation of the fuzzy method within FI^{4}FA. As seen in the background, FI^{4}FA represents an appealing technique to be able to carry out fine-grained evaluation of the failure behaviour as well as evaluation of the mitigation behaviour. Moreover, in the context of networked systems, the I^4 failures, as discussed previously, might have crucial interpretations. FI^{4}FA, however, being based on standard FPTC does not allow users to take into consideration the uncertainty discussed in Section III.B. Fuzzy sets, as recalled in the background, allow users to represent uncertainty. Thus, we integrate fuzzy knowledge within FI^{4}FA. The result of the integration is called F^{2}I^{4}FA.
The syntactical rules of F^{2}I^{4}FA are given in the textbox below. The main addition in comparison with FI^{4}FA is the non-terminal 'tb' which permits qualifying the behaviour with

additional information concerning the belief. More precisely, the three rules that are affected by the addition are listed at the end of the textbox and they are emphasized in *italics*.

behaviour = expression | expression (';' expression)+
expression = LHS '→' RHS
LHS= token | '(' token (',' token)+ ')'
token = no-failure | alphachar | failure |
'{'failure (',' failure)+ '}'
failure = basic-standard | combined
basic-standard = timing | value | sequence
timing ='early' | 'late'
value ='coarse' | 'subtle'
sequence = 'omission' | 'commission'
combined = basic-standard'.'basic-standard |
basic-standard'.'A-avoidable'.'C-avoidable'.'I-avoidable'.'D-avoidable
A-avoidable = 'incompletion' | no-failure
C-avoidable = 'inconsistency' | no-failure
I-avoidable = 'ww-cycle-based-interference' |
'all-cycle-based-interference' | no-failure
D-avoidable = 'impermanence' | no-failure
no-failure = basic-star | detailed-star
basic-star = '*'
detailed-star = basic-star'.'A-mitigation'.'C-mitigation'.'I-mitigation'.'D-mitigation
A-mitigation = 'all-or-nothing' | 'all-or-compensation' | 'none'
C-mitigation = 'full-consistency' | 'partial/consistency' | 'none'
I-mitigation = 'serializable' | 'none'
D-mitigation = 'no-loss' | 'partial-loss' | 'none'

***RHS** = tb | '{' tb (',' tb)+ '}'*
***tb** = '{' token ';' belief'}'*
***belief** = real*

These syntactical rules allow a user to specify the failure behaviour of individual components. The behaviour consists of a collection of expressions and each expression, in turn, indicates a propagation/transformation rule. More specifically, an expression indicates how the component behaves in response to a specific input (token), specified on the left-hand side of the arrow. The component may propagate the token as it is or it may transform it. The response to the token received in input is specified on the right-hand side of the arrow and a belief is associated to it. The response of the component is specified on the right-hand side of the arrow.

The semantics of a F^2I^4FA specification is given by using the classical fuzzy logic to calculate the belief and by using a similar algorithm as done in [11] to calculate the failure information as a fixed-point calculation.

To give an informal intuition of the semantics, we show how to reason about failure propagation of two components C_A and C_B, connected in series and resulting in a composition. Let us have the following two rules, each of which defines the failure behaviour of a single component:

$token^A \rightarrow \{\{token_1;\ \mu_{1A}\},\{token_2;\ \mu_{2A}\},\dots,\{token_n;\ \mu_{nA}\}\}$
$token^B = token^n \rightarrow \{\{token_1;\ \mu_{1B}\},\{token_2;\ \mu_{2B}\},\dots,\{token_m;\ \mu_{mB}\}\}$

Let us suppose that the following conditions hold:

$i=\{1..n\}$,
$j=\{1..m\}$

By applying equation (3), given in Section II.B, we obtain:

$$token^A \rightarrow \begin{cases} token_1, \mu_{1A} \\ token_2, \mu_{2A} \\ \vdots \\ token_n \rightarrow \begin{cases} token_1, \mu_{nA} \cap \mu_{1B} \\ token_2, \mu_{nA} \cap \mu_{2B} \\ \vdots \\ token_n, \mu_{nA} \cap \mu_{nB} \end{cases} \end{cases} = \begin{cases} token_1, \mu_{1A} \\ token_2, \mu_{2A} \\ \vdots \\ token_n \rightarrow \begin{cases} token_1, \min(\mu_{nA}, \mu_{1B}) \\ token_2, \min(\mu_{nA}, \mu_{2B}) \\ \vdots \\ token_n, \min(\mu_{nA}, \mu_{nB}) \end{cases} \end{cases} \quad (4)$$

Further, by applying equation (2), we calculate the union of all equal tokens that appear. Finally, we get the following:

$$token^A \rightarrow \begin{cases} token_1, \max(\mu_{1A}, \min(\mu_{nA}, \mu_{1B})) \\ token_2, \max(\mu_{2A}, \min(\mu_{nA}, \mu_{2B})) \\ \vdots \\ token_{n-1}, \max(\mu_{n-1A}, \min(\mu_{nA}, \mu_{n-1B})) \\ token_n, \min(\mu_{nA}, \mu_{nB}) \end{cases} \quad (5)$$

Similarly, we may obtain the system-level rules for the case when m<n, n<m or when we have more different expressions for the LHS of F^2I^4FA specification.

To illustrate the approach more concretely we consider the following simple example.

Let us suppose that the component C_A represents a sender and C_B – a receiver. If chained, these two components form a composed system (C_A+C_B) that may represent a transmission system. Let us suppose that the behaviour of each single component (specified according to the F^2I^4FA rules) is:

C_A behaviour:
**→{{*; 0.2},{late;0.5},{coarse.incompletion; 0.65}}*

The above behaviour consists of a single rule which specifies that in response to a normal behaviour, the sender is expected: to keep on behaving normally with a belief of 0.2; to generate a late failure with a belief of 0.5; to generate a coarse incompletion failure with a belief of 0.65.

C_B behaviour:
coarse.incompletion→{{;0.1},*
{late;0.75},{coarse.incompletion; 0.6}}

The above rule specifies that in response to a coarse incompletion failure, the receiver is expected to behave normally with a belief of 0.1; to transform the coarse incompletion failure into a late failure with a belief of 0.75; to propagate the coarse incompletion failure with a belief of 0.6.

Given the behaviour of the single components, the behaviour of the composed system (C_A+C_B) is calculated as the result of the union of the intersection of the behaviour of the single components.

According to equation (3), the intersection is calculates as follows:

$$* \rightarrow \begin{cases} *;0.2 \\ \text{late};0.5 \\ \text{coarse.incompletion} \rightarrow \begin{cases} *;\min(0.65;0.1) \\ \text{late};\min(0.65;0.75) \\ \text{coarse.incompletion};\min(0.65;0.6) \end{cases} \end{cases} \quad (6)$$

Then, according to (2), we calculate the union and we obtain the system's behaviour, which is:

- Normal behaviour with a belief of 0.2
- Late with a belief of 0.65
- Coarse incompletion with a belief of 0.6

D. Generic application process for fuzzy FI4FA

To apply the F^2I^4FA technique, the following three steps have to be performed:

1) *Analysis of the components' behaviour.* The response of a component to its input is analysed in isolation from the rest of the system. From this analysis it is possible to establish the belief with which a component: a) propagates a failure received in input as it is; b) transforms a failure; c) generates a failure (source behaviour); d) stops a failure (sink behaviour).
2) *Specification of the component's behaviour.* The component's behaviour analysed at step 1 must be specified as a collection of propagation or transformation expressions, using F^2I^4FA syntax.
3) *Calculus of the whole system's behaviour.* The inter-connected components are considered as a token (failure and believe)-passing network and a fixed-point calculation is performed. In particular, a similar algorithm to that one proposed in [3] is used to calculate the failure behaviour at system level and the fuzzy operators, as explained in Section III.C, are used to calculate the belief associated to that failure behaviour.

Once the result of the analysis is available, it can be used to select a specific counter-measure to react to the system's behaviour in case of failure (as explained in [16] as well as in [2]). It is crucial to observe, however, that only if the belief associated to the failure behaviour is reasonably high, a specific counter-measure should be selected. In case of a low belief, the level of uncertainty would be too high and in that case it would be better to treat the failure behaviour as an arbitrary one. As a reaction to an arbitrary failure, a general instead of a specific counter-measure should be selected. The user can re-engineer the architecture by introducing a component capable of mitigating the arbitrary behaviour.

IV. Related work

To assess dependability, it is crucial to have: the system's threats (fault, error and failure) model and techniques to perform analysis of the system's behaviour with respect to the threats model. In [1], authors propose a network-based fault model and they use finite-state automata to describe the system's behaviour. Their work contributes in putting in evidence the importance of having a fine-grained classification of failures. Their work, however, does not provide any contribution in failure propagation analysis techniques.

In [11], authors propose an extension of FPTC to consider the probability of the failure propagation. One drawback of probabilistic approach is that it usually requires a lot of input data in order to make statistically correct estimations about given probabilistic value. Moreover, as mentioned in the background, a probabilistic approach does not allow users to model the degree with which an element belongs to a set.

Another direction of work, related to ours is concerned with application of fuzzy based approach to software reliability and error propagation modelling. In [6], an architecture based fuzzy reliability model is proposed, which is able to take into account uncertainty in reliability estimates. A profound survey of application of fuzzy techniques in lot of fields of systems engineering, including software engineering is presented in [7]. Other approaches [8], [9] aim towards estimation of reliability, based on fuzzy reasoning on system testing data. A model for description of operational profile of software systems is proposed in [10] and is based on input data from experts.

These last efforts, however, do not consider either analysis of failure behaviour of components within software systems, or mitigation behaviour.

V. Conclusions and future work

In this paper we have proposed a new technique for the analysis of networked and component-based systems. A twofold motivation makes the proposed technique advantageous with respect to its predecessors: it allows users to take into consideration the uncertainty that typically affects the estimation of the failure behaviour (thus a more realistic measure can be obtained); it allows reasoning on the specificity of failures that threaten networked systems (fine-grained analysis).

Our technique, currently, is in its initial stages of development. In the future, to advance its development, first of all we plan to improve its formalization. Moreover, to allow users to perform the analysis automatically, we intend to provide an adequate tool-support. Finally, to validate the usefulness of our approach, we intend to carry out an experimental evaluation, potentially in safety/mission critical industrial settings, which, we believe may benefit from this analysis to provide dependability arguments in the context of system certification.

Acknowledgment

This work has been partially supported by the European Project ARTEMIS-JU100022 CHESS [4], the National Science Fund of Bulgaria under the PD01-0106 ARECS project (contract № Д002-298/2008) and Knowledge Foundation, Sweden and RISE Holding AB through the ARIES project. We would also like to thank the anonymous reviewers for their valuable remarks and comments on the submitted version of this paper.

REFERENCES

[1] F. Fummi, D. Quaglia, F. Stefanni. Network Fault Model for Dependability Assessment of Networked Embedded Systems. IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems, pp.54-62, 2008.

[2] B. Gallina and S. Punnekkat. FI^4FA: A Formalism for Incompletion, Inconsistency, Interference and Impermanence Failures Analysis. In Proceedings of the IEEE International workshop on Distributed Architecture modeling for Novel Component based Embedded systems (DANCE), September, 2011.

[3] M. Wallace. Modular architectural representation and analysis of fault propagation and transformation. Electronic Notes in Theoretical Computer Science (ENTCS), volume 141 n.3, pp.53-71, December, 2005.

[4] ARTEMIS-JU-100022 CHESS- Composition with guarantees for High-integrity Embedded Software components aSsembly.

[5] G . Chen, and T. Pham. Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems. CRC Press LLC, 2001.

[6] A. Dimov and S. Punnekkat. Fuzzy reliability model for component-based software systems. In Proceedings of the 36th EUROMICRO SEAA Conference, pp. 39-46, France, September 2010.

[7] C. Cai. System failure engineering and fuzzy methodology – an introductory overview, in Fuzzy Sets and Systems. Volume 83, Issue 2, Fuzzy methodology in system failure engineering, pp 113-133, October 1996.

[8] G. Junhong, Y. Xiaozong, and L. Hongwei. Software Reliability Nonliner Modeling and its Fuzzy Evaluation. In Proceedings of the 4th WSEAS International Conference, Sofia, Bulgaria, Oct. 27-29, pp.49-54, 2005.

[9] O. Georgieva, and A. Dimov. Software Reliability Assessment via Fuzzy Logic Model. In Proceedings of the International Conference on Computer Systems and Technologies, CompSysTech 2011, June 2011, Vienna, Austria, in press.

[10] K. Kumar, R. Misra and N. Goyal, Development of Fuzzy Software Operational Profile. In Proceedings of the 2^{nd} International Conference on Secure System Integration and Reliability Improvement (SSIRI), pp. 195-196, 2008.

[11] X. Ge, R. F. Paige, and J. A. Mcdermid. Probabilistic failure propagation and transformation analysis. In Proceedings of the 28th International Conference on Computer Safety, Reliability, and Security (SAFECOMP), Bettina Buth, Gerd Rabe, and Till Seyfarth (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 215-228, 2009.

[12] L. Grunske, J. Han. A comparative study into architecture-based safety evaluation methodologies using AADL's Error Annex and failure propagation models. 11th IEEE High Assurance Systems Engineering Symposium (HASE), pp.283-292, 2008.

[13] A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr, Basic concepts and taxonomy of dependable and secure computing. In: IEEE Trans. Dependable Sec. Comput. 1(1): 11-33. 2004.

[14] S. Chandran, A. Dimov and S. Punnekkat, Modeling uncertainties in the estimation of software reliability - a pragmatic approach. In Proceedings of the Fourth IEEE International Conference on Secure Software Integration and Reliability Improvement (SSIRI), Singapore, pp. 227-236, June 2010.

[15] X. Zhang and H. Pham. An analysis of factors affecting software reliability. In Journal of Systems and Software, 50(1), pp. 43-56, 2000.

[16] F. Ye and T. Kelly. Component failure mitigation according to failure type. Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC), pp.258-264 vol.1, 28-30 Sept. 2004.