INTUITIONISTIC FUZZY METHODS IN SOFTWARE RELIABILITY MODELLING

ALBEANU G.*, POPENTIU-VLADICESCU FL.**
*Spiru Haret University, 13 Ion Ghica str., Bucharest, Romania, galbeanu@yahoo.com
**University of Oradea, 1 Universității str., Oradea, Romania, popentiu@imm.dtu.dk

Abstract: This paper considers the framework of component-based software and illustrates the usage of intuitionistic fuzzy numbers in imprecise software reliability modeling and computing. It is shown how intuitionistic triangular and trapezoidal fuzzy numbers can be used in computing the intuitionistic fuzzy reliability of serial, parallel and hybrid software architectures.

Key words: component-based software development, intuitionistic fuzzy numbers, reliability

1 INTRODUCTION

Component-based software design, implementation, testing, and reusing are the most important steps of a recent methodology facilitating agility in software project management [1, 2]. It can be described by *the usage of reusable components as the building blocks for constructing software* [4]. According to [8], "*software assets, or components, include all software products, from requirements and proposals, to specifications and design, to user manuals and test suites*". Also, the software analysis depends on three assets [11]: code, specification, and test received.

In order to establish the terminology used in this paper, by *software component* we refer to any of the following assets: *functions* (belonging to a reusable library), *modules* (or classes), *libraries* (collections of reusable functions), *packages* (collections of reusable classes), and *applications* (programs to be reused by "exec" service, including code, files, and databases). We do not refer to any specification or tests, even these are very important when analyze the software quality.

From viewpoint of software methodologies paradigms, an important step toward efficient reuse of software components consists of the evolution from *reusable function libraries* to *object-oriented class libraries* reaching a new stage called: *reusable application framework and platform* [6].

The aims of component-based methodology are to achieve multiple quality objectives related to structured-ness efficiency, usability, consistency, conciseness, completeness, portability, security, maintainability, testability, and reliability. Other quality aspects refer to the verification costs' minimization, the software reliability increase, and to the development time reduction. Recently, new quality influential factors were added: trust-ability, interoperability, transparency and extensibility [2, 13]. For the aim of this paper only the reliability characteristic will be considered.

According to [15], a software component is defined as "*a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties*". The individual components can be tested and evaluated independently [6, 7, 9]. The whole system reliability depends on the reliability of every component and the architectural model. Following the standard model, if the system consists of n components with reliabilities R_j, $j = 1, 2, \ldots, n$, and an execution path is given, for instance: $1, 4, 3, 4, 2, 1, 4, 3, n$, then the path reliability is $R_1 x R_4 x R_3 x R_4 x R_2 x R_1 x R_4 x R_3 x R_n$. The system reliability can be estimated by averaging over all path reliabilities [9]. A probabilistic approach based on the probabilities of using the components will be described in the next section. The approach is a special case of component based statistical software reliability analysis.

The most natural approach considers that the reliability characteristics are independent of the software age. However, the performance of the software would decrease in time due to inappropriate management of the data structures (files, lists, trees and graphs, dynamic arrays). Therefore, the software without periodically maintenance, or rejuvenation, would loose fractions of initial running speed. This will affect the quality of the software from the customer point of view.

In this paper the software reliability optimization problem is studied when complex architectures are used, and various constraints applied. Firstly, the classical optimization approach, described in [12], will be outlined. Then, some aspects on reliability prediction for component-based software architectures will be discussed, and new formulas and interpretations will be given in the context of intuitionistic fuzzy paradigm [3, 10].

Finally, a discussion related to the practical usage of the proposed methods and conclusions will be provided.

2 THE BASIC SOFTWARE RELIABILITY MODEL

Following [12], let n be the number of software components, σ the reliability system target ($0 < \sigma < 1$) and T be the length of the mission interval ($T > 0$). Let us consider the exponential model, which is a good assumption when software rejuvenation is not used. For all $j = 1, 2, \ldots, n$, let λ_j be the failure intensity of the j^{th}

component. Hence, the probability density function of the random variable giving the time to failure of the j^{th} component is:

$$f_j(t) = \lambda_j exp(-\lambda_j t),$$

with the corresponding reliability function:

$$R(t) = exp(-\lambda_j t).$$

Let $R(\lambda_j;\ t)$ be the reliability of the j^{th} component: $R(\lambda_j;\ t) = exp(-\lambda_j t)$. For a serial composition, the reliability of the system can be computed as the product:

$$R(\lambda_1, \lambda_2, ..., \lambda_n;\ t) = \prod_{j=1}^{n} R(\lambda_j; t).$$

If we consider information concerning the operational profile, T - the time mission, p_i - the probability of executing the operation i, and φ_{ij} - the time allocated to the component j, then the expected proportion of the total mission time that the software spends executing in component j, denoted by τ_j, is given by

$$\tau_j = \sum_{i=1}^{m} p_i \varphi_{ij}$$

Therefore, the reliability of the j^{th} component with respect to the proportion of time it runs is

$$R(\lambda_j; T) = \exp(-\sum_{i=1}^{m} p_i \varphi_{ij} \lambda_j T) = \exp(-\tau_j \lambda_j T),$$

and the reliability of the integrated system with respect to time mission T is given by:

$$R(\lambda_1, \lambda_2, \ldots, \lambda_n; T) = \exp(-\sum_{j=1}^{n} \sum_{i=1}^{m} p_i \varphi_{ij} \lambda_j T).$$

An important objective when we model the software reliability is linked to behavior prediction. The difficulty appears when consider the deployment context [14]. As Reussner et al. (2003) shown, "design and implementation faults of software have a different impact on the reliability of the software, depending on how frequently the faulty code is executed". If such probabilities can be estimated, the methodology described above can be used. Another aspect in component-based software engineering is related to component dependencies. The exact properties of some components are not known until deployment. The class of unknown components includes: operating system(s) modules, middleware, and network and transport services. The usage profile can be modelled with Markov chains, but this approach is not considered here.

In the following sections we consider the intuitionistic fuzzy environment. Triangular and trapezoidal intuitionistic fuzzy numbers are used for reliability computing.

3 INTUITIONISTIC FUZZY MODELLING

3.1 From GFNs to IFNs

The fuzzy sets were introduced in 1965 by Zadeh [17], while the intuitionistic fuzzy sets were considered, in 1983, by Atanassov [3].

Chen [5], in 1985, represented a generalized fuzzy numbers (GFN) by a 5-tuple (a, b, c, d; w) of five real numbers, such that $0<w\leq 1$ and $a < b < c < d$. The GFN denoted by A is a fuzzy subset of the real line R, whose membership function μ_A satisfies the following conditions:

1) $\mu_A : R \rightarrow [0,1]$;
2) $\mu_A(x)= 0$, for $-\infty < x \leq a$;
3) μ_A is strictly increasing on $[a, b]$;
4) $\mu_A(x) = w$, for $x \in [b, c]$;
5) μ_A is strictly decreasing on $[c, d]$;
6) $\mu_A(x)= 0$, for $d \leq x < \infty$.

According to [10], the membership model of a Generalized Triangular Fuzzy Number (GTFN), given by (a, b, c; w), can be written as:

$$\mu_A^w(x) = \begin{cases} w\dfrac{x-a}{b-a}, \text{for } a \leq x \leq b \\ w\dfrac{c-x}{c-b}, \text{for } b \leq x \leq c \\ 0, \text{otherwise.} \end{cases}$$

For an universe of discourse defined by X, the intuitionistic fuzzy set (IFS) A in X is characterized by a membership function $\mu_A(.)$ and a non-membership function $\nu_A(.)$, where $\mu_A : X \rightarrow [0,1]$, and $\nu_A : X \rightarrow [0,1]$. For each point x in X, $\mu_A(x)$ (resp. $\nu_A(x)$) is the degree of membership (resp. non-membership) of x in A, with $0\leq\mu_A(x)+\nu_A(x)\leq 1$. An intuitionistic fuzzy set becomes a fuzzy set if $\nu_A(x) = 0$ for all x in A.

The operations on IFS can be introduced according to the generalized fuzzy set theory. Some examples follow:

$A \cap B = \{(x, \min(\mu_A(x),\mu_B(x)), \max(\nu_A(x), \nu_B(x))), x\in X\}$;
$A \cup B = \{(x, \max(\mu_A(x),\mu_B(x)), \min(\nu_A(x), \nu_B(x))), x\in X\}$;
$A + B = \{(x, \mu_A(x)+\mu_B(x)-\mu_A(x)\mu_B(x), \nu_A(x)\nu_B(x)), x\in X\}$;
$AB = \{(x, \mu_A(x)\mu_B(x), \nu_A(x)+\nu_B(x)- \nu_A(x)\nu_B(x)), x\in X\}$;

An intuitionistic fuzzy number (IFN) can also be defined using the template of GTFNs. The most used IFNs are the Triangular Intuitionistic Fuzzy Numbers (TIFNs). The TIFN A is described by five real numbers $(a_1,\ a_2,\ a_3;\ a',\ a'')$, $a' \leq a_1 \leq a_2 \leq a_3 \leq a''$, and two triangular functions

$$\mu_A(x) = \begin{cases} \dfrac{x-a_1}{a_2-a_1}, \text{for } a_1 \leq x \leq a_2 \\ \dfrac{a_3-x}{a_3-a_2}, \text{for } a_2 \leq x \leq a_3, \\ 0, \text{otherwise} \end{cases}$$

and

$$\nu_A(x)=\begin{cases}\dfrac{a_2-x}{a_2-a'}, \text{for } a'\le x\le a_2\\ \dfrac{x-a_2}{a''-a_2}, \text{for } a_2\le x\le a''\\ 1, \text{otherwise}\end{cases}.$$

Another important intuitionistic numbers have a trapezoidal shape. Let $a_1^{'}\le a_1\le a_2\le a_3\le a_4\le a_4^{'}$. A Trapezoidal Intuitionistic Fuzzy Number A in ***R*** (TrIFN), written as *(a_1, a_2, a_3, a_4; $a_1^{'}$, a_2, a_3, $a_4^{'}$)*, has the membership function

$$\mu_A(x)=\begin{cases}\dfrac{x-a_1}{a_2-a_1}, \text{for } a_1\le x\le a_2\\ 1, \text{for } a_2\le x\le a_3\\ \dfrac{a_4-x}{a_4-a_3}, \text{for } a_3\le x\le a_4\\ 0, \text{otherwise}\end{cases}$$

and the non-membership function

$$\nu_A(x)=\begin{cases}\dfrac{a_2-x}{a_2-a_1^{'}}, \text{for } a_1^{'}\le x\le a_2\\ 0, \text{for } a_2\le x\le a_3\\ \dfrac{x-a_3}{a_4^{'}-a_3}, \text{for } a_3\le x\le a_4^{'}\\ 1, \text{otherwise.}\end{cases}$$

The arithmetic operation, denoted generically by *, of two IFNs, is a mapping of an input subset of ***RxR*** (with elements $x = (x_1, x_2)$) onto an output subset of ***R*** (with elements denoted by y). Let A_1 and A_2 be two IFN, and (A_1*A_2) the resultant of the operation *. Then:

$$(A_1*A_2)(y)=\left\{\begin{pmatrix} y, \\ \vee_{y=x_1*x_2}[A_1(x_1)\wedge A_2(x_2)], \\ \wedge_{y=x_1*x_2}[A_1(x_1)\vee A_2(x_2)]\end{pmatrix}^T \forall x_1,x_2,y\in R\right\},$$

with

$$\mu_{(A_1*A_2)}(y)=\vee_{y=x_1*x_2}[A_1(x_1)\wedge A_2(x_2)],$$

and

$$\nu_{(A_1*A_2)}(y)=\wedge_{y=x_1*x_2}[A_1(x_1)\vee A_2(x_2)].$$

The arithmetic operations on IFNs can be defined using the (α, β) – cuts method. Let $\alpha, \beta \in [0, 1]$ be fixed numbers such that $\alpha + \beta \le 1$. A set of (α, β) - cut generated by an IFS A is defined by:

$$A_{\alpha,\beta} = \{(x, \mu_A(x), \nu_A(x)), x \in X, \mu_A(x) \ge \alpha, \nu_A(x) \le \beta\}.$$

The (α, β) – cut of a TIFN is given by

$$A_{\alpha,\beta}=\left\{[A_1(\alpha),A_2(\alpha)],[A_1^{'}(\beta),A_2^{'}(\beta)]\right\},$$

where:

i) $A_1(\alpha)$, and $A_2^{'}(\beta)$ are continuous, monotonic increasing functions of α, respective β;

ii) $A_2(\alpha)$, and $A_1^{'}(\beta)$ are continuous, monotonic decreasing functions of α, respective β;

iii) $A_1(1)=A_2(1)$, and $A_1^{'}(0)=A_2^{'}(0)$.

When using the 5-tuple notation, we obtain:

$$A_1(\alpha)=a_1+\alpha(a_2-a_1),$$
$$A_2(\alpha)=a_3-\alpha(a_3-a_2),$$
$$A_1^{'}(\beta)=a_2-\beta(a_2-a'),$$

and

$$A_2^{'}(\beta)=a_2+\beta(a''-a_2).$$

To fulfill the aim of this paper we need the following properties [10]:

1. If TIFN $A = (a_1, a_2, a_3; a', a'')$, and $k > 0$, then the TIFN kA is given by $(ka_1, ka_2, ka_3; ka', ka'')$.
2. If TIFN $A = (a_1, a_2, a_3; a', a'')$, and $k < 0$, then the TIFN kA is given by $(ka_3, ka_2, ka_1; ka'', ka')$.
3. If $A = (a_1, a_2, a_3; a', a'')$ and $B = (b_1, b_2, b_3; b', b'')$ are TIFNs, then the TIFN $A\oplus B$ is defined by $(a_1+b_1, a_2+b_2, a_3+b_3; a'+ b', a''+b'')$;
4. If $A = (a_1, a_2, a_3; a', a'')$ and $B = (b_1, b_2, b_3; b', b'')$ are TIFNs, then the TIFN $A\otimes B$ is defined by $(a_1b_1, a_2b_2, a_3b_3; a'b', a''b'')$.

The above results can be proved using the (α, β) – cuts method.

The (α, β) – cut of a TrIFN is defined as usually, by

$$A_{\alpha,\beta}=\left\{[A_1(\alpha),A_2(\alpha)],[A_1^{'}(\beta),A_2^{'}(\beta)]\right\},$$
$$\alpha+\beta\le 1, \alpha,\beta\in[0,1],$$

where

$$A_1(\alpha)=a_1+\alpha(a_2-a_1),$$
$$A_2(\alpha)=a_4-\alpha(a_4-a_3),$$
$$A_1^{'}(\beta)=a_2-\beta(a_2-a_1^{'}),$$

and

$$A_2^{'}(\beta)=a_3+\beta(a_4^{'}-a_3).$$

Also, the above operations defined for TrIFNs have similar properties:

5. If TrIFN $A = (a_1, a_2, a_3, a_4; a_1^{'}, a_2, a_3, a_4^{'})$, and $k > 0$, then the TrIFN kA is given by $(ka_1, ka_2, ka_3, ka_4; ka_1^{'}, ka_2, ka_3, ka_4^{'})$.
6. If TrIFN $A = (a_1, a_2, a_3, a_4; a_1^{'}, a_2, a_3, a_4^{'})$, and $k < 0$, then the TrIFN kA is given by $(ka_4, ka_3, ka_2, ka_1; ka_4^{'}, ka_3, ka_2, ka_1^{'})$.
7. If $A = (a_1, a_2, a_3, a_4; a_1^{'}, a_2, a_3, a_4^{'})$ and $B = (b_1, b_2, b_3, b_4; b_1^{'}, b_2, b_3, b_4^{'})$ are TrIFNs, then the TrIFN $A\oplus B$ is defined by $(a_1+b_1, a_2+b_2, a_3+b_3, a_4+b_4; a_1^{'}+b_1^{'}, a_2+b_2, a_3+b_3, a_4^{'}+b_4^{'})$;

8. If $A = (a_1, a_2, a_3, a_4; a_1', a_2, a_3, a_4')$ and $B = (b_1, b_2, b_3, b_4; b_1', b_2, b_3, b_4')$ are TrIFNs, then the TrIFN $A \otimes B$ is defined by $(a_1b_1, a_2b_2, a_3b_3, a_4b_4; a_1'b_1', a_2b_2, a_3b_3, a_4'b_4')$.

3.2 Intuitionistic fuzzy software reliability

For simplicity reason we assume that every component is called only once, but this is not a serious constraint.

a) Using TIFNs

Let S be a software system integrating a number of components according to a serial architecture,

S = **SEQ** C_1; C_2; …, C_n; **END**.

This is the case of single processor computer systems running the software during a period of time T. If R_j is the triangular intuitionistic fuzzy reliability of the j^{th} component C_j, and R_S is the triangular intuitionisc fuzzy reliability of the entire system (with n items), and $R_j = (r_{j1}, r_{j2}, r_{j3}; r_j', r_j'')$, then

$$R_S = R_1 \otimes R_2 \otimes \cdots \otimes R_n,$$

defined by $(r_1, r_2, r_3; r', r'')$, with:

$$r_i = \prod_{j=1}^{n} r_{ji},\ i = 1, 2, 3,\ r' = \prod_{j=1}^{n} r_j',\ \text{and}\ r'' = \prod_{j=1}^{n} r_j''.$$

If S is a software system composed by n items running in parallel:

S = **PAR** C_1; C_2; … C_n; **END**,

using the above notations, we evaluate the triangular intuitionistic fuzzy reliability of S by:

$$R_S = 1 \Theta \prod_{j=1}^{n} (1 \Theta R_j),$$

defined by $(r_1, r_2, r_3; r', r'')$, with:

$$r_i = 1 - \prod_{j=1}^{n} (1 - r_{ji}),\ i = 1, 2, 3,$$

$$r' = 1 - \prod_{j=1}^{n} (1 - r_j'),$$

and

$$r'' = 1 - \prod_{j=1}^{n} (1 - r_j'').$$

b) Using TrIFNs

If R_j is the trapezoidal intuitionistic fuzzy reliability of the j^{th} component, and R_S is the trapezoidal intuitionisc fuzzy reliability of the entire serial system (with n items), and $R_j = (r_{j1}, r_{j2}, r_{j3}, r_{j4}; r_{j1}', r_{j2}, r_{j3}, r_{j4}')$, then

$$R_S = R_1 \otimes R_2 \otimes \cdots \otimes R_n,$$

defined by $(r_1, r_2, r_3, r_4; r_1', r_2, r_3, r_4')$, with:

$$r_i = \prod_{j=1}^{n} r_{ji},\ i = 1, 2, 3, 4,\ r_1' = \prod_{j=1}^{n} r_j',\ \text{and}\ r_4' = \prod_{j=1}^{n} r_j''.$$

If S is a parallel software system composed by n items, using the above notations, we evaluate the trapezoidal intuitionistic fuzzy reliability of S by:

$$R_S = 1 \Theta \prod_{j=1}^{n} (1 \Theta R_j),$$

defined by $(r_1, r_2, r_3, r_4; r_1', r_2, r_3, r_4')$, with:

$$r_i = 1 - \prod_{j=1}^{n} (1 - r_{ji}),\ i = 1, 2, 3, 4,$$

$$r_1' = 1 - \prod_{j=1}^{n} (1 - r_{j1}'),$$

and

$$r_4' = 1 - \prod_{j=1}^{n} (1 - r_{j4}'').$$

Using the above methodology intuitionistic fuzzy reliability formulas can be derived for hybrid software architectures.

4 CONCLUSIONS

In this paper we consider subjects like software reliability estimation based on user profile. Firstly we introduce the component-based software engineering paradigm as a new methodology for development and deployment of reliable, secure and high quality software. Next, based on probabilistic thinking and statistical testing we describe a reliability model for software systems when considering the operational profile. The imtuitionistic fuzzy paradigm is considered in the third section. Intuitionistic fuzzy sets, operations on IFS, triangular and trapezoidal intuitionistic fuzzy numbers are described and used to model the reliability of serial, parallel, and hybrid systems.

The study presented here is relevant for architectures based on components.

REFERENCES

1. G. Albeanu, H. Madsen, and A. Averian, On the influence of software vulnerabilities on software reliability. The case of open source component based software. In R. Bris, C. Guedes Soares, S. Martorell (eds.), Reliability, Risk and Safety: Theory and Applications, Proceedings of ESREL Conference, Prague, 7-10 September 2009 , Vol. **2**, 1341-1346. CRC Press/Balkema, Taylor and Francis Group, London, UK, 2009.
2. G. Albeanu. Agile CMMI for e-learning software development. In Roceanu I (coord.), Jugureanu R., Stefan V., Popescu V. & Radu C. (eds.), Proceedings of the 5th International Scientific Conference eLSE 2009 (Bucharest, April. 09-10, 2009), 135-142. " Carol I" National Defence University P. H. , 2009.
3. K.T. Atanassov, "Intuitionistic Fuzzy Sets", Physica-Verlag, Heidelberg, New York, 1999.
4. A. Averian, G. Duda, and G. Albeanu, Quality Assurance for Agile Component-Based Software Development, In H Pham, T Nakagawa (eds.), Proceedings of the 15th ISSAT International Conference on Reliability and Quality in Design, San Francisco, California, USA, August 6-8, 2009, 100-104.
5. S.H. Chen, Operations on fuzzy numbers with function principal, Tamkang J. Management Science, **6**(1), 1985, 13-25.
6. J.Z. Gao, H.-S. J. Tsao, and Y. Wu, "Testing and Quality Assurance for Component-Based Software", Artech House, Boston, London, 2003.
7. H.-G. Gross, "Component-Based Software Testing with UML", Springer-Verlag, Berlin, Heidelberg, 2005.
8. K.S. Jasmine, R. Vasantha, DRE – A Quality Metric for Component based oftware Products, World Academy of Science and Technology, **34**, 2007, 48-51.
9. J.-H. Lo, S.-Y. Kuo, M.R. Lyu, and C.-Y. Huang, Optimal Resource Allocation and Reliability Analysis for Component-Based Software Applications, Proceedings of the 26th Annual International Computer Software and Applications Confereence, IEEE Computer Society, 2002.
10. G.S. Mahapatra, Reliability Optimization in Fuzzy and Intuitionistic Fuzzy Environment, PhD Thesis, Bengal Engineering and Science University, Shibpur, India, 2009.
11. J. May, "Component-Based Software Reliability Analysis", Department of Computer Science, University of Bristol, CSTR-02-2002, 2002.
12. Fl. Popentiu-Vladicescu, G. Albeanu, Optimal Reliability Allocation for Large Software Projects. In Proceedings of the 8th Int. Conf. Opt. Electrical & Electronic Equipment, **II**, 601-606, The Transilvania University of Brasov, 2002.
13. F. Popentiu, G. Albeanu, On the Software Trustability Assessment, Proc. Conf. KONBIN'01 Safety&Reliability, Szczyrk-Poland, vol. **1**, 2001, 5-8.
14. R.H. Reussner, H.W. Schmidt, I.H. Poernomo, Reliability prediction for component-based software architectures, The Journal of Systems and Software, **66**, 2003, 241-252.
15. C. Szyperski, "Component Software - Beyond Object Oriented Programming", ACM Press, 1998.
16. I. Vaduva, G. Albeanu, "Introduction to fuzzy modelling" (in romanian), Bucharest Publishing Press, 2004.
17. L.A. Zadeh, Fuzzy sets, Information and Control, **8**, 1965, 338-353.