

A Novel Approach to Minimizing the Risks of Soft Errors in Mobile and Ubiquitous Systems

Muhammad Sheikh Sadi, D. G. Myers, Cesar Ortega
Sanchez

Department of Electrical and Computer Engineering
Curtin University of Technology, Australia

Jan Jurjens
Department of Computer Science
TU Dortmund
Dortmund, Germany

Abstract—A novel approach to minimizing the risks of soft errors at modeling level of mobile and ubiquitous systems is outlined. From a pure dependability viewpoint, critical components, whose failure is likely to impact on system functionality, attract more attention of protection/prevention mechanisms (against soft errors) than others do. Tolerating soft errors can be much improved if critical components can be identified at an early design phase and measures are taken to lower their criticalities at that stage. This improvement is achieved by presenting a criticality ranking (among the components) formed by combining a prediction of soft errors, consequences of them, and a propagation of failures at system modeling phase; and pointing out the ways to apply changes in the model to minimize the risks of degradation of desired functionalities. Case study results are given to illustrate and validate the approach.

Keywords—Criticality Analysis; Soft Errors; Reliability Risks, Mobile and Ubiquitous Systems; UML Model; Metrics

I. INTRODUCTION

The demands on embedded mobile and ubiquitous systems are increasing along with more complex functionalities such as pervasive computing, mobile computing, and high-speed wireless networking [1], [2]. Continuous improvement of wireless networks has created different types of wireless systems, such as Bluetooth for personal areas, Wireless LANs (WLANs) for local areas, Universal Mobile Telecommunications System (UMTS) for wide areas, and satellite networks for global networking. These systems require to co-ordinate with each other to provide ubiquitous high-data-rate services to mobile users [3]. Hence, reliability is a high requirement in these systems. The reliability of these systems is affected by both permanent and transient faults. Permanent faults such as nodes stuck-at-1/0, transistors open, shorted transistors and so forth, arise during fabrication or result from aging, and destroy the intended function of the circuit [4]. Transient faults, in contrast, are not the result of physical damage to a chip but can be catastrophic for the desired functionalities of the system [5], [6]. These transient faults create soft errors when they are executed in the system. Soft errors are of particular concern as system complexity, reduction in operational voltages, exponential growth of the number of transistors per chip, increases in clock frequencies and device shrinking significantly increase their rate [7], [8]. Prior research to cope with soft errors mostly focuses on post-design phases such as circuit level solutions, logic level

solutions, spatial redundancy, temporal redundancy, and/or error correction codes. Early detection and correction of such problems during the design phase is much more likely to be successful than detection once the system is operational [9]. Estimating reliability (or at least identifying failure-prone components) early in the life-cycle of a design is therefore preferable [10], [11]. Ideally, this should be done at the system design level so that the designer can create required prevention or detection mechanisms in the detailed design that follows. From a pure dependability viewpoint, critical components attract more attention of protection/prevention mechanisms than others do since reliability of a system is correlated with the criticality of the system [12], [13]. Hence, an approach is needed at the design stage to highlight those components where transient faults are critical.

This paper examines the use of metrics to identify critical components of a system model. It also investigates how to encourage the designer to explore changes that could be made in the existing model. Case studies illustrate the effectiveness of this approach in determining components' criticality rankings and then lowering their criticalities. The model is expressed in Unified Modeling Language (UML) since this allows the modeler to describe different views on a system, including the physical layer [14], [15]. The paper is organized as follows. Section 2 describes related work. Section 3 outlines the methodology to measure and reduce component's criticality employed in this research. This methodology is applied to a real-life case study in Section 4. Finally, in Section 5, conclusions are drawn.

II. RELATED WORK

Researchers have evolved several measures to prevent soft errors. Much less attention has been dedicated, until now, to the integration of design processes with reliability verification techniques. Rather, a "fix-it-later" approach is still dominant [16]. At a system level, duplicating hardware [17], [18] and then comparing the results, and/or executing several copies of software by using the same hardware [19] to detect soft errors are the most common approaches. Then, different recovery approaches are employed to recover from the soft errors. At the circuit level, the solution is mainly to increase the critical charge of a circuit node [20]. Logic level solutions [5] mainly propose detection and recovery in combinational circuits by using redundant or self-checking circuits. Gold et al. [21] proposed distributed shared memory multi-processor features

that incorporate computation and memory storage redundancy to detect and recover from a single point of transient or permanent failure. Mohamed et al. [22] shows chip level redundant threading with recovery, where the basic idea is to run each program twice, as two identical threads, on a simultaneous multithreaded processor. These post-functional design phase approaches are costly as well as complex to implement. Moreover, they can increase time delays and power overhead without offering any performance gain. Timing and synchronizing issues are also matters of great concern in these approaches.

Few approaches [23], [24] dealt with the static complexities of the system as a risk assessment methodology to minimize the risks of faults. However, these static approaches do not deal with the matter of how a module functions in its executing environment. A fault may not manifest itself into a failure if never executed. Cortellessa et al. [9] and Yacoub et al. [13] defined dynamic metrics that include dynamic complexity and dynamic coupling metrics to measure the quality of software architecture. To assess the severity of the components they have defined only three levels of system failure. However, in real life scenarios, only three severity levels are not sufficient to represent several possible failure modes.

III. A METHODOLOGY TO MEASURE AND REDUCE COMPONENT'S CRITICALITY

Complexity is taken as a measure of the likelihood of a component to be affected by soft errors. Severity of failure of components is taken as a measure of the impact of a system's functionality being affected by a component suffering a soft error. The methodology presented here is to measure the complexity and severity of each component, plus the propagation of failure from that component, and then take the product as a measure of criticality. The model is examined by refactoring to lower component criticality by maintaining constraints. The details of these steps are outlined as follows.

A. Measuring the Complexities of the Components

There is a correlation between the likelihood of soft errors proneness and the complexity of a system [12], [25]. Complexity analysis does not measure the impact of components in system functionality, but it shows the rank of likelihood of soft error proneness among the components. Complexity is measured, in this paper, by an assessment of execution time (ET) during simulation and Message-In-and-Out frequency (MIO).

1) Execution Time during Simulation

The Failure-In-Time (FIT) of a system due to soft error is proportional to the fraction of time in which the system is susceptible to soft errors if the circuit type, transistor sizes, node capacitances, temperature etc. are kept at constant [26]. The longer duration to perform the selected operation implies that the component is being used more frequently and/or it is experiencing many state changes. A soft error occurs at any access point and/or in any behavioral change of these components can spread towards all communicating components through the large number of behavioral linkages until the soft error affected component is in execution. The

method of measuring ET during simulation (to perform an operation by a component) can be shown as follows. Component state S is a function of time: $S(t)$ where t denotes time. An external function $F()$ is required to be executed to perform the operation $F(S(t_i)) \rightarrow S(t_j)$: where $S(t_i)$ is the state of a component at t_i and $S(t_j)$ is the state of that component at t_j . Hence, ET, to execute the function $F()$ that changes the state of the p th component from $S(t_i)$ to $S(t_j)$, can be shown as (1).

$$ET_p(F(S(t_i)) \rightarrow S(t_j)) = \sum_{j=1}^n d_{p_j} \quad (1)$$

Where n is the total number of state changes in the p th component's behaviour execution and d_{p_j} is the duration in the j th slot of changing states of p th component. Since UML does not specify how to simulate the architecture models, Telelogic Rhapsody [27] is used to gain execution data via simulation. The model is executed in tracing mode. Several tracing commands are used to execute the model. The state transition times for the components are saved to a log file. At the end of the simulation, that log file is analyzed to calculate the total ET of the components to perform a selected operation.

2) Message-In-and-Out Frequency

In a model-based system, components are often interdependent. They communicate with each other by message passing among them. Number of messages from and to a component shows the measure of dependence with other components. Components with more dependence could easily manifest themselves into a failure of the system because services of these components are frequently accessed by other components [14]. To figure out this fault proneness, a component's MIO, which is the ratio of number of messages from and to a component in a scenario and total number of messages in that scenario is calculated. Define MIO_{i_k} as the MIO for i th component in k th scenario. $M_{(i,j)}$ is the message between component i and component j (where $j=1, \dots, m$, $i \neq j$, and m is the number of messages from i th component to other components) in k th scenario, and n_k is the total number of messages, communicating among all the components, in that scenario. Then, MIO_{i_k} can be derived as shown in (2).

$$MIO_{i_k} = \frac{|\sum_{j=1}^m M_{(i,j)}| \quad |i \neq j|}{n_k} \quad (2)$$

For each component, Total MIO (TMIO) in all possible different scenarios can be calculated using (3). TMIO for i th component is:

$$TMIO_i = \sum_{k=1}^{n'} P(Sc_k) MIO_{i_k} \quad (3)$$

where n' is the total number of scenarios in the system, $P(Sc_k)$ is the probability of k th Scenario in that system, and MIO_{i_k} is the MIO for i th component in k th scenario.

3) Overall Complexity

The Overall Complexity of the i th Component (OCC_i) is the summation of different complexity factors for that component. The equation to derive OCC_i is shown in (4).

$$OCC_i = ET_i + TMIO_i \quad (4)$$

where ET_i and $TMIO_i$ are Execution Time, and Message-In-and-Out frequency for the i th component. Since, ET_i and $TMIO_i$ are independent on each other, OCC_i is calculated using the summation of these two factors. For simplicity, the weights of ET and TMIO in measuring total value of complexities are assumed as equal.

B. Measuring the Severity of the Failure of the Components

A single soft error in a particular component could have a greater effect than multiple soft errors in another or a set of components. For this reason, the effects of soft errors in the whole system need to be analyzed by injecting transient faults (which will create soft errors if activated) into each component. These results are merged with the component's complexities to obtain a better measure of their impact on system if they are affected by soft errors. The severity of failures of components is determined by the Failure Mode and Effects Analysis (FMEA) method [28]. FMEA is a procedure for the analysis of potential failure modes within a system by classifying severity or determination of the failure's effect upon the system. Hosseini et al. [28] suggested evaluation criteria and a ranking system for the severity of effects for a design FMEA as shown in TABLE I. Transient faults are injected at each component, into one bit at a time. The reason is that transient faults change the value of one bit at a time and the probability of changing two bits and/or two transient faults are almost zero.

TABLE I. EVALUATION CRITERIA AND RANKING SYSTEM OF FMEA

Linguistic terms for severity of a failure mode	Rank
Hazardous	10
Serious	9
Extreme	8
Major	7
Significant	6
Moderate	5
Low	4
Minor	3
Very minor	2
No effect	1

C. Measuring Propagation of Failure from the Components

Before measuring the component's propagation of failure, its complexity and severity are multiplied together to measure their combined impact (if there is any soft error) on the whole system. Measuring the propagation of failure refines this impact to obtain a clearer picture of the impact or criticality of each component. The method of measuring the propagation of

failure is shown in Fig. 1, which is a scenario of a system model showing three components: C_1 , C_2 , and C_3 .

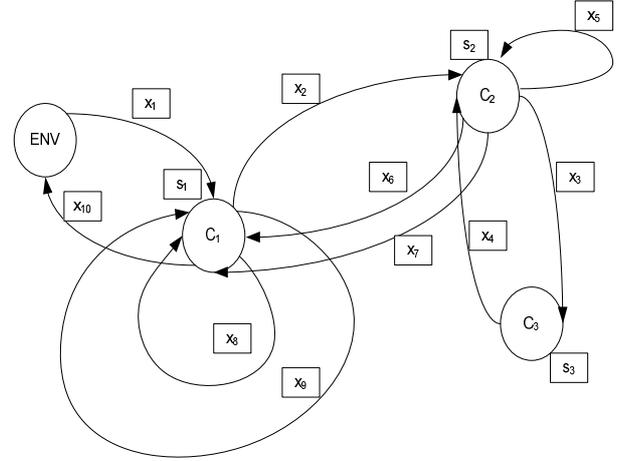


Fig. 1. An Example Scenario of a System Model to Measure the Propagation of failure

ENV denotes the environment communicating with the system. The product of complexity and severity of these three components are s_1 , s_2 , and s_3 respectively. In Fig. 1, x_1, \dots, x_{10} indicate the severity in corresponding messages where indexing is made according to their time of occurrences in the whole scenario. Failures due to soft errors may be propagated via message communication. The propagation of failure from or in the environment is not considered. To measure propagation of failure through message passing involves finding the increase in level of consequences of each message. A soft error in C_1 (before it passes a second message) sees an increase in level of consequences in C_2 to s_1x_2 since soft errors may propagate from C_1 to C_2 through the passed message.

After passing the 2nd message, there is an increase in level of consequences in C_2 : s_1x_2 and after passing the 3rd message, there is an increase in level of consequences in C_3 : $s_1x_2s_2x_3$.

Similarly, after passing the 9th message, there is an increase in the level of consequences in C_1 : $s_1x_2s_2x_3s_3x_4s_2x_5s_2(x_6+x_7)s_1x_8s_1x_9$

The total consequences in the system can be defined as $CONC_{C_1} (= s_1x_2s_2x_3s_3x_4s_2x_5s_2(x_6+x_7)s_1x_8s_1x_9)$

If the soft error occurs in C_1 within the 2nd and 8th messages then the consequences ($CONC_{C_2} = s_1x_8s_1x_9$) can be propagated in the system after passing the 8th message.

If the soft error occurs in C_1 after passing the 8th message then the consequences ($CONC_{C_3} = s_1x_9$) can be propagated in the system with the 9th message. In the same way, if soft errors occur in C_2 , and/or in C_3 then the increase in level of consequences can be checked at different stages of message passing. The consequences in the system can be measured as follows.

$$CONC_{C_2} = s_2x_3s_3x_4s_2x_5s_2(x_6+x_7)s_1x_8s_1x_9$$

$$CONC_{C_3} = s_2x_5s_2(x_6+x_7)s_1x_8s_1x_9$$

$$CONC_{2_3} = s_2(x_6+x_7)s_1x_8s_1x_9$$

$$CONC_{2_4} = s_2x_7s_1x_8s_1x_9$$

$$CONC_{3_1} = s_3x_4s_2x_5s_2(x_6+x_7)s_1x_8s_1x_9$$

The total propagation of failure from each component (due to a soft error in that component) can be derived as follows.

$$CONC_{1_T} = \sum_{i=1}^3 CONC_{1i}$$

$$CONC_{2_T} = \sum_{i=1}^4 CONC_{2i}$$

$$CONC_{3_T} = CONC_{3_1}$$

If the values of s_1 , s_2 , and s_3 ; and x_1, \dots, x_{10} are known then the above propagations can be derived. The total propagation of failure in the whole system (due to a soft error in any component) can be shown as follows:

$$CON(C_i) = \sum_{k=1}^{n'} P(Sc_k) \times CON(C_{i_k}) \quad (5)$$

where n is the total number of scenarios in the system, $P(Sc_k)$ is the probability of the k th scenario, and $CONC_{i_k}$ is the propagation of failure from the i th component in the k th scenario.

D. Measuring Criticalities of the Components

For each component, criticality is the product of complexity, severity, and the propagation of failure. The combined impact of complexity and severity is used to calculate the propagation of failure. Criticality is calculated by taking the product of complexity, severity, and the propagation of failure. If the criticality of the i th component is Cr_i then the equation to derive it can be shown as:

$$Cr_i = \prod (OCC_i, CON(C_i), Se(C_i)) \quad (6)$$

where, OCC_i is the overall complexity of the component, $CON(C_i)$ is the propagation of failure from the component, and $Se(C_i)$ is the severity of the component. $OCC_i, CON(C_i), Se(C_i)$ are dependent on each other; i.e. for any increase in complexity there is a high probability that the severity will increase, and if the product of complexity and severity increases then the probability of propagation of failure will increase too. Hence, criticality is taken as the product of overall complexity, severity, and propagation of failure.

E. Lowering the Criticalities of Components

Component criticality suggests to the designers where in the system design changes are necessary or helpful to minimize soft errors risk. These changes can be done by applying a suitable approach where he/she may change the architecture or behavioral model of the component to lower its complexity, and/or severity, and/or propagation of failure. Refactoring is a

good candidate for this type of approach. UML model refactoring re-structures the model, at the conceptual level, to improve quality factors such as maintainability, efficiency and fault tolerance without introducing any new behaviour [29]. Once the criticality ranking is returned, a model can be refactored with the goal of reducing the criticalities of the components. Lowering the criticalities can be achieved by reducing any of the multiplying factors: complexity, severity or propagation of failure. Fig. 2 details the methodology of lowering component criticality by refactoring.

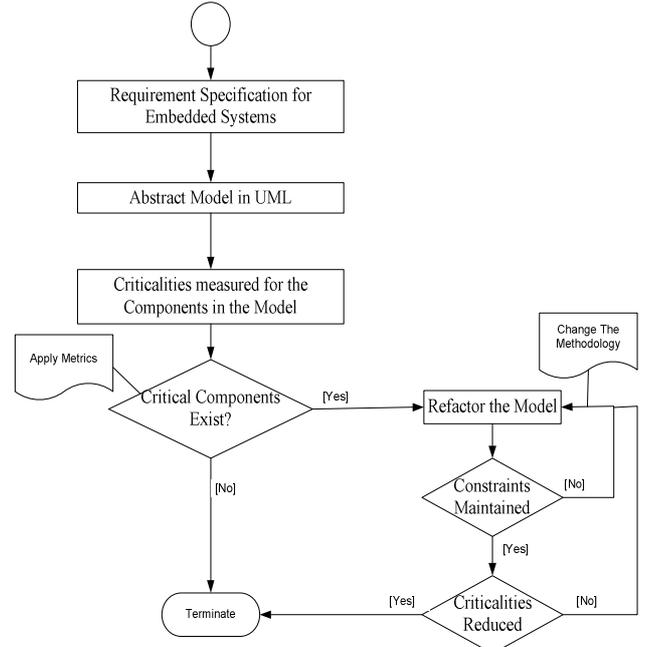


Fig. 2. Methodology to lower the criticalities of the Components by Refactoring

IV. CASE STUDIES

Real-life case study: A wireless telephony Handset System illustrates the application of the metrics. It is chosen, as it is illustrative of a broad class of systems that must have high reliability. Handset System has three sub-systems (for this example, sub-systems represent components): (i) a Connection Management (CM) system to handle the reception, setup, and transmission of incoming and outgoing call requests, (ii) a Data Link (DL) system to handle the registration and location of users, and (iii) a Mobility Management (MM) system to monitor registration.

A. ET Analysis of the Handset System

The 'Call Control' Statechart diagram of CM in the Handset system is used for ET analysis and is shown in Fig. 3. If it receives a confirmation, the call connects, and remains connected until it receives a message to disconnect. When operation succeeds, the time of executing the 'Place Call' event at 'Idle' state, and the time when the system reached at 'Connected' state of 'Call Control' statechart were recorded to calculate the ET of these sub-systems. The second column of TABLE II shows the normalized values of ET of the sub-systems.

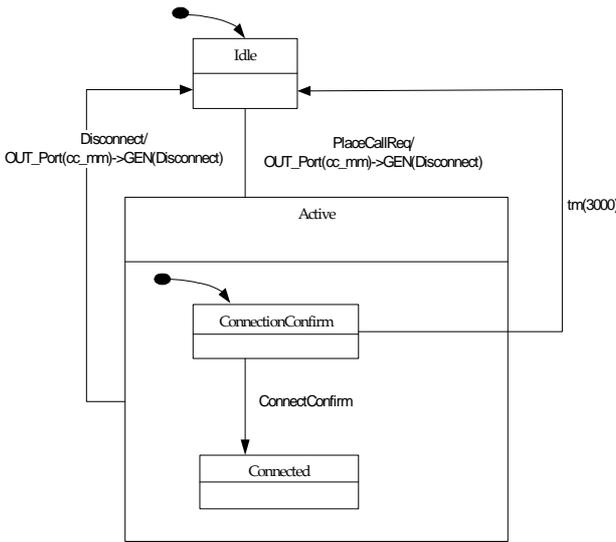


Fig. 3. Call Control statechart diagram at the beginning of execution

B. MIO and TMIO of the Sub-systems in the Handset System

There are three scenarios in the Handset system: i) Place Call Request Successful, ii) Network Connect, and iii) Connection Management Place Call Request Success. The probabilities of the occurrences of three scenarios are assumed as 0.45, 0.30, and 0.25 respectively. The assumptions are made with respect to their usage in real life scenarios. MIO and TMIO for three different sub-systems are calculated for three different sequence diagrams using (2) and (3). All values of TMIO for the three sub-systems are shown in the third column of TABLE II.

C. Overall Complexities of the Sub-systems in the Handset System

Overall complexities of three sub-systems are calculated using (4), and the last column in TABLE II shows their overall complexities. Overall complexity is the summation of ET during simulation, and Message-in-and-out-frequencies of the sub-systems. Though MM has the highest value of ET, and DL has the same for TMIO, considering both of the complexities CM is the most complex sub-system in Handset system. Overall complexities of DL and MM are almost equal.

TABLE II. THE COMPLEXITY OF THE SUB-SYSTEMS IN HANDSET SYSTEM

Sub-Systems	Normalized values of ET	TMIO	Overall Complexities
CM	0.29	0.62	0.91
DL	0.05	0.74	0.79
MM	0.67	0.13	0.80

D. Measurement of the Severity of the Failure Sub-systems in the Handset System

The severity of sub-systems are determined by the FMEA where the effects of soft errors in each sub-system are analyzed by injecting transient faults, and checking their effects. TABLE III shows the results. To simplify the column name in the TABLE III, Severity of Failure, and Severity Rank are abbreviated as SOF, and SR respectively.

TABLE III. THE SEVERITY OF THE SUB-SYSTEMS IN HANDSET SYSTEM

Sub-Systems	Failure Mode	Effect of Failure	SOF	SR
CM	Failed to trigger the Connection Call event	Could not complete the connection and Connection went back to idle stage	Serious	9
DL	Failed to respond to Registration request	Could not initiate the registration and Connection went back to idle stage	Extreme	8
MM	Failed to update Location	Connection was held in Location Update state and could not confirm the connection	Major	7

E. Measuring Propagation of Failures from the Sub-systems in Handset System

At first, the failure propagation is calculated for each sub-system and for the three different scenarios. Then the total failure propagation for each sub-system is calculated using equation (5). The second column in TABLE IV shows the calculated failure propagation (normalized) due to soft errors in these three scenarios. Fault propagations are calculated using only participating components.

TABLE IV. THE PROPAGATION OF FAILURE AND CRITICALITIES OF THE SUB-SYSTEMS IN THE HANDSET SYSTEM

Sub-Systems	Propagation of failure	Criticality of the Components
CM	10	81.9
DL	0.00027	0.0017064
MM	0.123	0.6888

F. Measuring Criticalities of the Sub-systems in Handset System

The criticalities of the sub-systems are shown in the last column of TABLE IV. The results show that CM is the most critical sub-system in the Handset system and is followed by MM, and DL.

G. Lowering the Criticalities of the Sub-Systems of Handset System

As shown in TABLE IV, CM has large criticality differences with the other sub-systems. This paper then targets to reduce the criticality of the sub-systems according to their criticality order. The behaviour models of all three sub-systems are carefully examined to be refactored. The functionality is being affected for any change made in the behavioural diagrams of CM, and DL sub-systems. The MMCallControl activity diagram of MM sub-system is able to bring under refactoring by maintaining the constraints. The calculated normalized ET of the sub-systems of refactored model and existing model (to establish a handset connection) is shown in TABLE V.

TABLE V. COMPARISON OF ET OF THE COMPONENTS BETWEEN REFACTORED MODEL AND EXISTING HANDSET MODEL

Sub-systems	Normalized ET of Refactored Model	Normalized ET of Existing Model
CM	0.29	0.228
MM	0.05	0.0393
DL	0.67	0.67

Lower ET will result to lower complexity as well as lower criticality of the sub-systems. TABLE V shows that refactoring the model is able to lower the ET of the CM, and MM sub-systems to a mentionable extent. The ET for DL is constant. DL is the least critical sub-system in the Handset system and its criticality is so low that it does not create any matter of concern.

V. CONCLUSIONS

This paper develops metrics for complexity analysis that could be analyzed in the early system design phase based on UML artifacts, develops a severity assessment methodology by analyzing UML model simulation results, and develops the methodology of measuring the propagation of failures from the components. This paper then integrates the three different methods to rank the component's criticality that highlight the variations of the impact of soft errors among the components. It then shows how possible changes can be made in the existing design to lower the criticalities of the components to minimize the risks of soft errors. In summary, the approach presented in this paper is effective in measuring the soft errors risks of the components in a system and in lowering the criticalities of components to minimize the risks of functional degradation.

REFERENCES

- [1] H. Pao-Ann, L. Shang-Wei, H. Chin-Chieh, F. Jih-Ming, L. Chao-Sheng, C. Cheng-Chi, C. Kuo-Cheng, L. Chun-Hsien, and L. Pin-Hsien, "Real-time embedded software design for mobile and ubiquitous systems," in International Conference on Embedded and Ubiquitous Computing, EUC 2007, (Lecture Notes in Computer Science vol. 4808), pp. 718-729.
- [2] S. Manzoni, F. Nunnari, and G. Vizzari, "Towards a model for ubiquitous and mobile computing," in Thirteenth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Los Alamitos, CA, USA, 2004, pp. 423-428.
- [3] I. F. Akyildiz, S. Mohanty, and J. Xie, "A ubiquitous mobile communication architecture for next-generation heterogeneous wireless systems," IEEE Communications Magazine, vol. 43, pp. 29-36, 2005.
- [4] A. Timor, A. Mendelson, Y. Birk, and Suri, N, "Using Under Utilized CPU Resources to Enhance Its Reliability," Dependable and Secure Computing, IEEE Transactions on, vol. 5, no. 4, 2008.
- [5] Zhang, M., Mitra, S., Mak, T.M., Seifert, N., Wang, N.J., Shi, Q., Kim, K.S., Shanbhag, N.R., Patel, S.J., "Sequential Element Design With Built-In Soft Error Resilience," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 14, pp. 1368-1378, 2006.
- [6] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. S. Kim, "Robust system design with built-in soft-error resilience," Computer, vol. 38, pp. 43-52, 2005.
- [7] G. P. Saggese, N. J. Wang, Z. T. Kalbarczyk, S. J. Patel, and R. K. Iyer, "An experimental study of soft errors in microprocessors," Micro, IEEE, vol. 25, pp. 30-39, 2005.
- [8] Y. Crouzet, J. Collet, and J. Arlat, "Mitigating soft errors to prevent a hard threat to dependable computing," presented at 11th IEEE International On-Line Testing Symposium, IOLTS, pp. 295-298, 2005.
- [9] Cortellessa, V., Goseva-Popstojanova, K., Appukkutty, K., Guedem, A.R., Hassan, A., Elnaggar, R., Abdelmoez, W., Ammar, H.H., "Model-based performance risk analysis," IEEE Transactions on Software Engineering, Vol. 31, pp. 3-20, 2005.
- [10] Jurjens, J., Wagner, S., "Component-based development of dependable systems with UML," Lecture Notes in Computer Science, vol. 3778, pp. 320-344, 2005.
- [11] A. Bondavalli, M.D.C., D. Latella, I. Majzik, A. Pataricza, and G. Savoia: Dependability Analysis in the Early Phases of UML Based System Design. Journal of Computer Systems Science and Engineering 16 (2001) 265—275
- [12] Khoshgoftaar, J.M.a.T., "Software Metrics for Reliability Assessment," Handbook of Software Reliability Eng., M. Lyu ed., Chapter 12, pp. 493-529, 1996.
- [13] Yacoub, S.M., Ammar, H.H., "A methodology for architecture-level reliability risk analysis," IEEE Transactions on Software Engineering, Vol. 28, pp. 529-547, 2002.
- [14] S. K. Wood, D. H. Akehurst, O. Uzenkov, W. G. J. Howells, and K. D. McDonald-Maier, "A model-driven development approach to mapping UML state diagrams to synthesizable VHDL," IEEE Transactions on Computers, vol. 57, pp. 1357-1371, 2008.
- [15] Linzhang, W., Wong, E., Dianxiang, X.: A threat model driven approach for security testing. IEEE, Minneapolis, MN, USA (2007) 64-70
- [16] Hiller, M., Jhumka, A., Suri, N., "EPIC: profiling the propagation and effect of data errors in software," IEEE Transactions on Computers, Vol. 53, 2004.
- [17] Meaney, P.J., Swaney, S.B., Sanda, P.N., Spainhower, L., "IBM z990 soft error detection and recovery.," IEEE Transactions on Device and Materials Reliability, Vol. 5, 2005.
- [18] Austin, T.M., "DIVA: a reliable substrate for deep submicron microarchitecture design," 32nd Annual International Symposium on Microarchitecture, pp. 196 – 207, 1999.
- [19] Xinpeng, Z., Wei, Q., "Prototyping a fault-tolerant multiprocessor SoC with run-time fault recovery," 43rd ACM/IEEE Design Automation Conference, pp. 53 – 56, 2006.
- [20] Cazeaux, J.M., Rossi, D., Omana, M., Metra, C., Chatterjee, A., "On transistor level gate sizing for increased robustness to transient faults," 11th IEEE International On-Line Testing Symposium, pp. 23 – 28, 2005.
- [21] B. T. Gold, J. Kim, J. C. Smolens, E. S. Chung, V. Liaskovitis, E. Nurvitadhi, B. Falsafi, J. C. Hoe, and A. G. Nowatzky, "TRUSS: a reliable, scalable server architecture," Micro, IEEE, vol. 25, pp. 51-59, 2005.
- [22] A. G. Mohamed, S. Chad, T. N. Vijaykumar, and P. Irith, "Transient-fault recovery for chip multiprocessors," IEEE Micro, vol. 23, pp. 76, 2003.
- [23] T. J. McCabe, "A Complexity Measure," Software Engineering, IEEE Transactions on, vol. SE-2, pp. 308-320, 1976.
- [24] Chidamber, S.R., Kemerer, C.F., "A metrics suite for object oriented design," IEEE Transactions on Software Engineering, Vol. 20, pp. 476-493, 1994.
- [25] Harrison, R., Counsell, S., Nithi, R., "Coupling metrics for object-oriented design," IEEE Comput. Soc, Bethesda, MD, USA, pp. 150-157, 1998.
- [26] H. T. Nguyen, Y. Yagil, N. Seifert, and M. Reitsma, "Chip-level soft error estimation method," Device and Materials Reliability, IEEE Transactions on, vol. 5, pp. 365-381, 2005.
- [27] <http://www.telelogic.com/Products/rhapsody/index.cfm>
- [28] S. M. Seyed-Hosseini, N. Safaei, and M. J. Asgharpour, "Reprioritization of failures in a system failure mode and effects analysis by decision making trial and evaluation laboratory technique," Reliability Engineering & System Safety, vol. 91, pp. 872-81, 2006.
- [29] S. Gerson, P. Damien, T. Yves Le, J. Jean-Marc, z, and quel, "Refactoring UML Models," in Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools: Springer-Verlag, 2001.