A Novel Approach to Derive the Average-Case Behavior of Distributed Embedded System

Rajib Ghosh[1], Arnab Mitra[2], Apurba Chakraborty[3]

[1] Dept. of CSE, Adamas Institute of Technology-Barasat, W.B. – 700126
rajibghosh01@gmail.com
[2] Dept. of CSE, Adamas Institute of Technology-Barasat, W.B. – 700126
mitra.arnab@gmail.com
[3] Dept. of MCA, Siliguri Institute of Techonology-Siliguri, W.B. – 734009
mail.apurbachakraborty@gmail.com

ABSTRACT

Monte-Carlo simulation is widely used in distributed embedded system in our present era. In this research work, we have put an emphasis on reliability assessment of any distributed embedded system through Monte-Carlo simulation. We have done this assessment on random data which represents input voltages ranging from 0 volt to 12 volt; several numbers of trials have been executed on those data to check the average case behavior of a distributed real time embedded system. From the experimental result, a saturation point has been achieved against the time behavior which shows the average case behavior of the concerned distributed embedded system.

KEYWORDS

Distributed real-time embedded system (DRE), reliability assessment, average case system behaviour testing, Monte-Carlo simulation.

1. INTRODUCTION

1.1. Distributed Real-time Embedded (DRE) system

Distributed Real-time Embedded (DRE) system's uses are includes command and control systems, and avionics mission computing, open environments such as network-centric system of systems. These open DRE systems operate in less predictable circumstances than prior generations of real-time and embedded systems (i.e. micro-controllers and PLCs).

As a result of that, planned DRE systems require a highly adaptive and flexible infrastructure which can figure out reusable resource management services from application code; thereby off-loading many monotonous and error-prone aspects of the software lifecycle from developers of DRE systems.

Promising software infrastructure technologies for DRE systems is as following: Component middleware - perform platform capabilities.

Tools - for specifying, implementing, deploying, and configuring components [1].

Services - that exchange messages between components [2].

Components are units of implementation, reuse, and composition that expose named interfaces which are connection points that components use to team up with each other. Component middleware helps simplifying the development and validation of DRE systems by providing reusable services. It optimizes that support of their functional and quality of service (QoS) which needs more effective than conventional ad-hoc software implementations.

1.2 Reliability

This idea of software reliability can be defined from the two most accepted standards: the ISO 9126 standard and the IEEE Software Engineering Standards Collection [IEEE 94]. The technical standards have translated the reliability into a verifiable definition which has lost the more emotional aspects like trust or confidence [3]. We will give some remarks to those definitions based on the problems identified previously, and the possible support of the standards to solve those problems.

In this paper our spotlight is on reliability of embedded products or systems from a user point of view. Reliability is one of the main characteristics of 'quality' according to the ISO 9126 [ISO9126] standard. The ISO 9126 standard sub-divides software reliability into three sub-parts:

Maturity: Attributes of software that accept on the occurrence of failure by faults in the software system.

Fault tolerance: Attributes of software that accept on its skill to preserve a specified level of performance in case of any software faults.

Recoverability: Recoverability: Attributes of software that accept on the ability to re-establish its level of performance and recover the data directly affected in case of any failure.

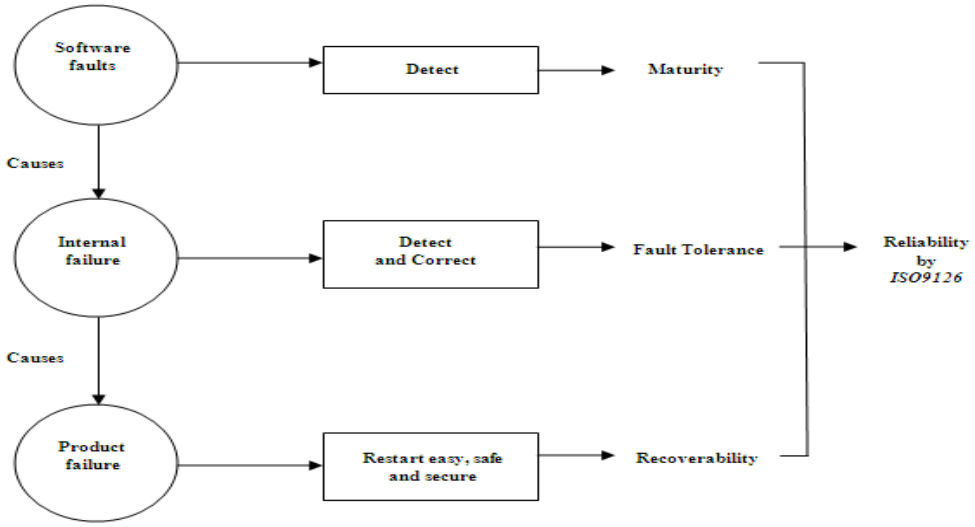


Fig. 1: Reliability according to ISO9126

According to the Fig. 1, it is easily deduced that how the reliability of a system can be measured. The main fault that can be occurred is the software faults which can cause internal as well as product failure; if the faults(software) have been detected then it proves it's maturity of the system where as in the internal failure, this facility is fault tolerance; and in product failure, the resuming part is easy, safe and secure(called recoverability). All these facilities are the reliability as a whole.

Software should be prepared to accomplish user requirements. Design faults or bugs must be taken out whose consequences are in maturity, on every occasion faults have not been detected those may cause a product to fall short. We can say a product is fault tolerant when the faults are caught in exact correct time by the product itself and failures are prevented as soon as the faults have been detected. The product should be a recoverable one.

This reliability features bearing in mind the ISO9126 standard is emphasizing operation of a product and that is an embedded product is not permitted to 'go down'. To start with the definition implies that having a product operate routinely means that all reliability requirements are satisfied, which is not obvious. Secondly, we know that embedded products have a tendency to fall short. Measures must be taken that make sure that product failure is foreseen and will not result in risky or unconfident situations. [3]

A typical sub-division of activities for the development of embedded software systems is:

- Initiation
- Specification
- Design
- Implementation

Measures are required as consequences in the essential level of reliability. Selection of measures is a tricky issue which is done in practice mostly embedded and/or based on experience. Whenever this description is available measures can be selected with a clear goal, and can be implemented.

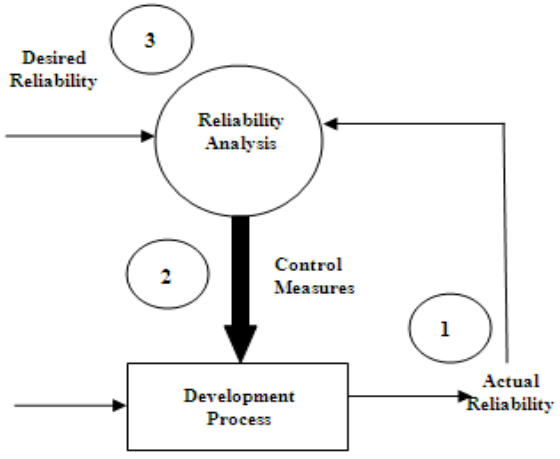


Fig. 2: Reliability control model

In the above Fig. 2, there are mainly 3 stages of the total model representation. This above chart shows us the interconnection between several sub-processes and main process of the whole system.

The objective of any development project is to create a reliable system at the end. Depending on the priority of reliability a certain no. of trials is chosen to figure out the required product reliability.

Method to assess the actual reliability: The use of the product is the synonym of ultimate reliability check but during the development phase, specific data collection appears to give a representation of product reliability. It seems logical to include data collection activities to track whether those measures give their intended effect or not.

Method to select control measures during the development processes: A supporting method is required to select the appropriate measures to fulfill reliability requirements.

1.3 Monte Carlo Simulation

Monte Carlo method is a stochastic method, meaning that it is based on using random numbers and probability to investigate the preferred problems. Monte Carlo methods are employed in a wide variety of fields (i.e., economics, finance, physics, chemistry, engineering, and even the study of traffic flows). Each field using Monte Carlo methods may relate them in different ways, but in essence they are using random numbers to inspect problems and approximate its outcome. As such Monte Carlo methods give us a way to represent model complex systems that are often extremely hard to examine with other types of techniques. Monte Carlo simulation is a technique used to understand the impact of risk and uncertainty in those complex models.

In a Monte Carlo simulation, a random value is selected for each of the tasks, based on the series of estimates. The result of the model is recorded, and the process transformed into a repetitive system. A typical Monte Carlo simulation calculates the model hundreds or thousands of times; each time this module use different values those were selected randomly. When the simulation is complete, we have a large amount of results based on random input values. These results are used to describe the probability of reaching various results in the model.

The simulation will only be as good as the estimates the developer or user makes. It's important to remember that the simulation only represents probabilities and not certainty. Nevertheless, Monte Carlo simulation can be a valuable tool when forecasting an unknown future in respect to any system.

The Monte-Carlo method is based on the generation of multiple trials to determine the expected value of a random value.

There are a number of commercial packages that run Monte-Carlo simulation; however a basic spreadsheet program can be used to run a simulation. In this case the generation of multiple trials is implemented by propagating a basic formula as many times as the number of iterations required by the model.

The total cost of the project is a random variable with a value between the minimum and maximum. This variable will be normally distributed since it is the sum of a number of random variables. This is also the reason why the individual distribution of each variable is not important.

The general scheme of Monte-Carlo method is as follows,

- Generate random values for each of the activity costs.
- Add each series of random values to arrive at a total project cost.
- The expected project cost is the average of these values.

The first step is to generate random values for each of the activity costs. Assuming a uniform distribution, we can use the RAND () function to generate random numbers in the interval (0, 1) and multiply these by the range of each variable.

2. Previous Work & Motivation

Evaluating the time-behavior of distributed embedded real-time systems can be performed in several ways. Mainly three different approaches can be used as shown in Fig. 3. In the graph the lines indicate a different area like the first one is to test the implemented system, the second one is to perform a simulation and the third one is to conduct a real-time analysis.

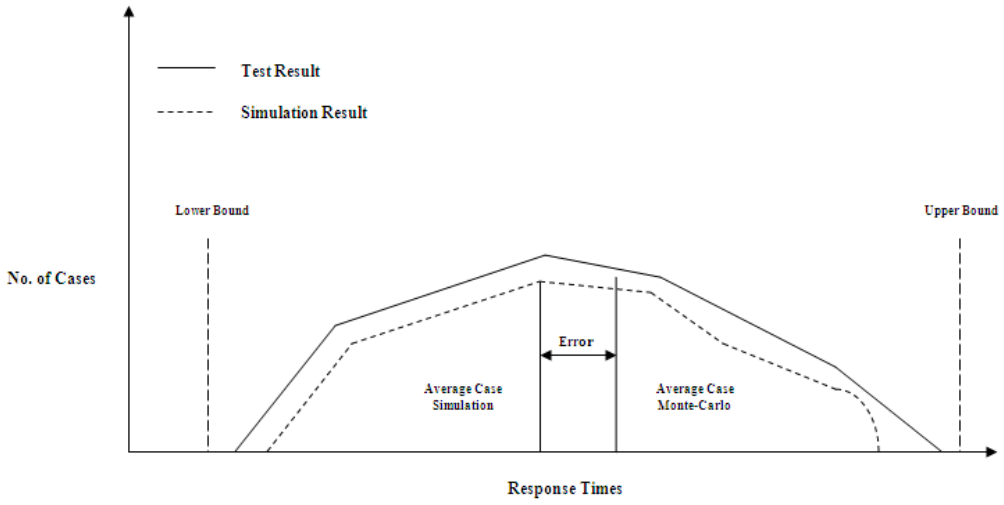


Fig. 3: Different evaluation techniques

Testing the implemented system is certainly the most realistic approach concerning the results, because the final system is used. But correcting any failures discovered at this late design stage can lead to enormous costs. To prevent this situation other approaches have to be used that are suitable for early design stages where mistakes are less expensive to fix. Such methods are the simulation or the analysis of a system. These can also be used in late design phases. For example a tool like chronSIM [4], which allows simulating a system, can provide information on what a system is actually doing especially in the average case. With a simulation however the border cases like the best-case and worst-case time behavior cannot be determined. Due to the coverage problem that simulations have, it is unknown whether such a border case has been simulated or not. Analytical approaches like those based on Tindell and Clark [5] construct the border cases and calculate guaranteed bounds for the time behavior. In [6] an automotive case study was conducted where both approaches: simulation and analysis are compared. The conclusion of the paper is that both approaches are needed for the design process of embedded hard real-time

systems. The focus in the paper is to determine the worst-case time behavior. But to consider only the border cases during a system design is often not sufficient to construct a reliable system. An example for this can be found in the domain of control systems. To design a reliable controller on the one hand the border cases are required to verify the stability of the control loop. On the other hand the most important part in the design is the knowledge of the average-case time behavior, because a controller should perform best in the average case. Therefore both values are needed to design a reliable controller. In [7] the impact of the time behavior on a control system is discussed more in detail.

As mentioned with a simulation the average case can be obtained, because it will most likely describe the actual system behavior. However the more complex a system is, the longer will be the runtime of the simulation. A good impression about the costs required to perform a simulation can be obtained by considering the results in [6]. The runtime can easily be in the magnitude of days to get significant data regarding the average case due to it depending on the history of the simulation. Therefore such an approach cannot be parallelized easily. It is possible to start various simulation runs simultaneously, but it is not clear how long each run has to be executed in order to obtain the desired data. So the question arises whether other methods can be used to determine the average case while needing less time than a simulation.

Analytical approaches currently only construct the border cases and calculate guaranteed bounds for them. Naturally the average case must be between the calculated best case and worst case. The question is if an analytical approach like the SymTA/S approach [8] or the real-time calculus [9] can be modified or extended in such a way that the average case or at least an approximation of the average case can be obtained more quickly than with a simulation. In [10] an extension of the real-time calculus is proposed where probabilistic arrival and service curves are considered. But it is an open question if the approach is able to construct the average-case behavior, because with the presented model it seems improbable that the average case of the system can be found.

We now sketch a proposal on how an analytical approach can be modified to possibly obtain the average case. The idea is based on a Monte-Carlo simulation [11] and to the best of our knowledge there is no successful work which describes how schedulability analysis techniques can be used with a Monte-Carlo simulation to compute the average-case behavior of a system.

In [12] a technique is presented which is based on a similar idea. A Monte-Carlo simulation is used to determine the time-behavior of a system, but it is not clear which method is used for the Monte-Carlo simulation. It is stated that the network-calculus is not used, because the system behavior cannot be modeled adequately.

3. Proposed Approach

Now, we are proposing an approach to deduce the average case behavior of a distributed real time embedded system. The benefit of such a method is that the different correlations between the tasks are covered intrinsically; but there are some issues in respect to analytical approach which are the dependencies within a system. Many approaches have been developed to consider data dependencies [9] or task dependencies [13]. However, it is uncertain whether the existing job regarding the dependencies is adequate enough to be able to analyze the average-case behavior. We propose a technique to use a Monte-Carlo simulation based on a schedulability analysis by varying parameters like stimulation, execution time, etc. The stimulation may only

be varied within its bound as well as the execution time which may be varied between its best-case and worst-case execution time. To alter the stimulation of the periodic model with jitter is insufficient; instead more significant models have to be used like the event streams [14] or the arrival curves used by the real-time systems [9].

Generally, Monte Carlo simulation is based upon the schedulability analysis which leads us to an analytical approach in constructing the corner cases; but here we can obtain the average case time behavior using the Monte Carlo simulator. The randomized data that has been used in the approach is a must one for this analysis as these data signifies the randomization of the input voltages that has been supplied to the DRE system. We have to make an approximation behavior of the simulator during analysis, so some dependencies should be handled properly, i.e, task dependencies. In this approach, we can set the threshold values and get acceptable quality results.

4. Results & Analysis

Using Monte-Carlo simulator, a set of data (range for embedded system is 0 volt to 12volt) has been sent as the input voltage to the real time distributed embedded system to check the reliability of the whole system. The data that is used as the input should be chosen as random basis.

Power is the most important side of all embedded systems. Without electric power, nothing works. More than a few options for power are available for different brand of application. AC adaptors are used to power a lot of electronic gadgets at home, like radios, answering machines, wireless routers, etc. AC adaptors used as mobile phones chargers. AC adaptors convert the high voltage AC in the wall socket to low voltage DC suitable to run the appliances. They usually provide the output voltage somewhere in the range of +3.3V to +12V DC, and supply current up to few amperes.

An embedded system consists of a lot of different machinery that can operate from a large range of power supply; but some components, such as Analog-to-Digital Converters (ADCs), require a constant voltage supply to provide an precise output because they need a reference voltage for converting the analog signal to digital count. The device, voltage regulator, is used for this purpose. Its job is to convert a range of input DC voltages to a constant output voltage.

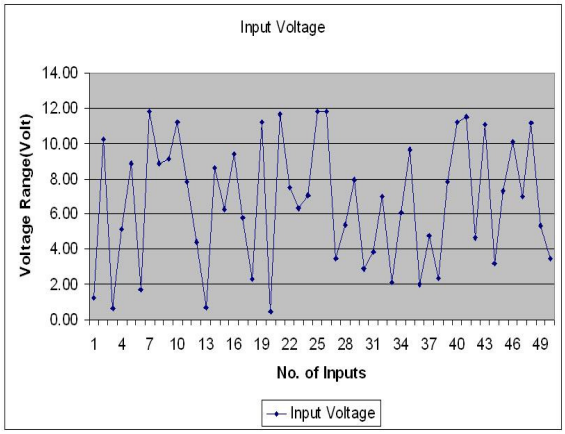
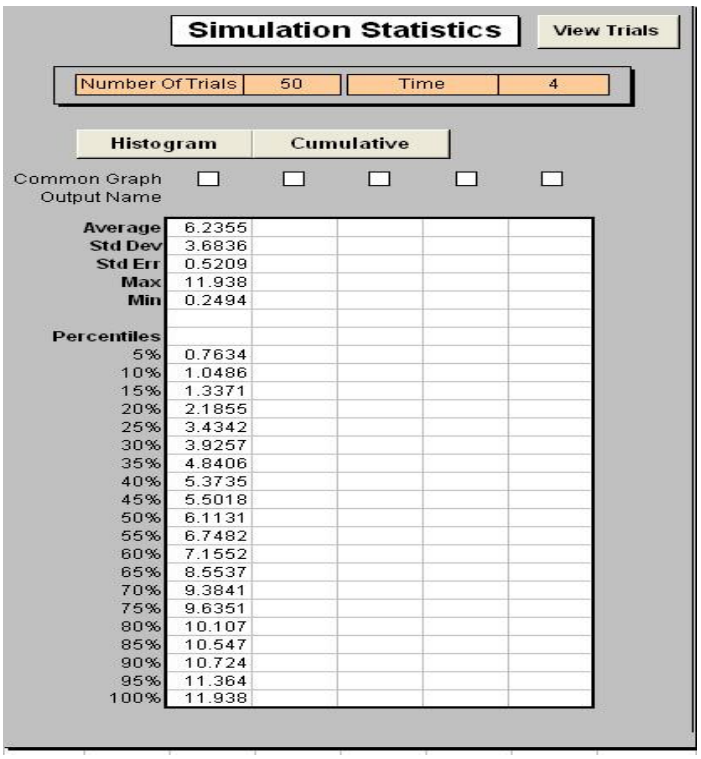


Fig. 4: Input voltage to the system

Besides that job, a voltage regulator also minimizes the power supply noise and provides a sort of protection for the embedded system from any possible damages due to fluctuating input voltages. The bottom line is that including a voltage regulator in your design is always good.

According to the Fig. 4, in this experiment the no of input voltage supplied is 50(the input voltage value used in this experiment has been chosen randomly). 50 trials have been made upon that input data to deduce the reliability of a real time distributed embedded system.



Average	6.2355
Std Dev	3.6836
Std Err	0.5209
Max	11.938
Min	0.2494
Percentiles	
5%	0.7634
10%	1.0486
15%	1.3371
20%	2.1855
25%	3.4342
30%	3.9257
35%	4.8406
40%	5.3735
45%	5.5018
50%	6.1131
55%	6.7482
60%	7.1552
65%	8.5537
70%	9.3841
75%	9.6351
80%	10.107
85%	10.547
90%	10.724
95%	11.364
100%	11.938

Fig. 5: Analysis of input voltages to the system

In the above chart, the input data that has been supplied to the simulator to analyze the system (Fig. 5) plotted. With those data the average data, standard deviation along with standard error, the maximum and minimum values of the data also reflected in the chart. As per the chart, the 'Average' is the average value of the input data. The 'Percentiles' is the portions of input data percentage used in the experiment by the simulator.

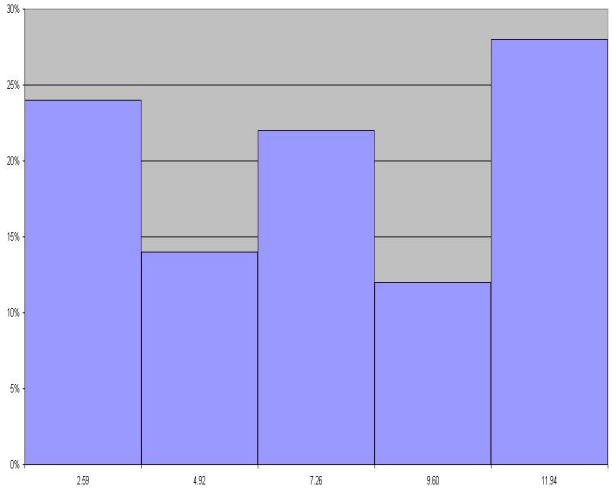


Fig. 6: Histogram representation of reliability upon the supplied input data

In the histogram representation, the range of data (0volt-12volt) has been equally divided into 5 portions. The input data of all the trials should be placed inside that exact range (horizontal axis of the Fig. 6). The vertical axis shows the reliability percentage of the real time distributed embedded system.

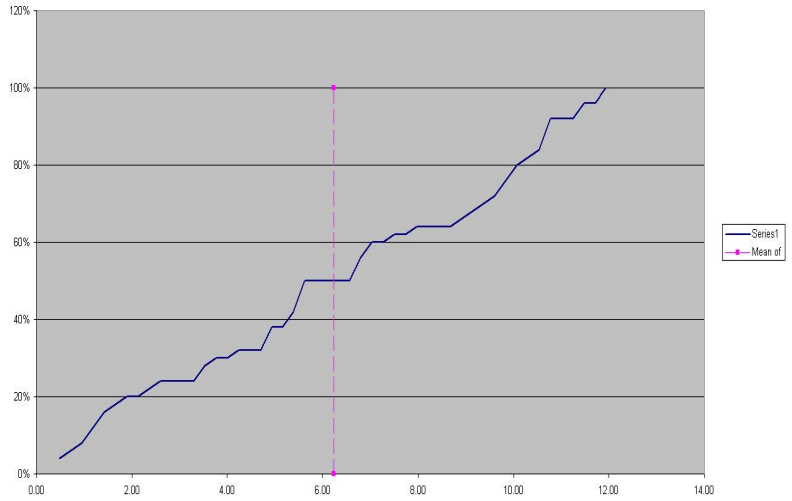


Fig. 7: Cumulative representation of the input data

If we go through the figure 7, then we can have a clear view about the reliability saturation point against the input data (voltage) of the system. We can see that from the point 5.63 to 6.56 the line became a straight one and the mean value is 6.24 which is an input value supplied to the simulator. This is the average case of the reliability calculation. The prior section of 5.63 (from the point 0.48 to 5.63) and the subsequent section of 6.56 (from the point 6.56 to 11.94) are in an unstable situation according to the graphical representation.

5. Conclusion

From the Fig. 4 and 5, we can get to know about the input data, which were taken randomly to the system alias simulator. The Fig. 6 and 7 tells us the impact of the data to the system like the saturation point, the mean of point of the whole graphical representation, the range of data used in the simulator. As the previous points, we also get to the region of input data where the system becomes unstable (Fig. 7). In this above case, we have to increase the input value of the system at the very first portion of the graph and we have to decrease the value of the input voltages at the very last portion of the graph; by this concept, we can try to make the system more and more reliability prone.

References

1. Object Management Group, "*CORBA Component Model*" 2002. [Online Available: htt//www.omg.org]
2. T. H. Harrison, D. L. Levine, and D. C. Schmidt, "*The Design and Performance of a Real-Time CORBA Event Service*" in Proceedings of the 12th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications. ACM Press, 1997, pp. 184–200.
3. K. Rob, V.S. Rini, T. Jos, W. Hans," *User-perceptions Of Embedded Software Reliability*".
4. chronSIM, http://www.inchron.com.
5. K. Tindell and J. Clark, "*Holistic schedulability analysis for distributed hard real-time systems*" Microprocessing and Microprogramming, vol. 40, pp. 117–134, April 1994.
6. S. Kollmann, V. Pollex, K. Kempf, F. Slomka, M. Traub, T. Bone, and J. Becker, "*Comparative application of realtime verification methods to an automotive architecture*" in Proceedings of the 18th International Conference on Real- Time and Network Systems, 2010.
7. T. Bund, S. Moser, S. Kollmann, and F. Slomka, "*Guaranteed bounds for the control performance evaluation in distributed system architectures*" in Proceedings of the International Conference on Real-Time and Embedded Systems (RTES 2010), Singapore, Sep. 2010.
8. K. Richter, "*Compositional scheduling analysis using standard event models - the symta/s approach*" Ph.D. dissertation, University of Braunschweig, 2005.
9. E. Wandeler, "*Modular performance analysis and interfacebased design for embedded real-time systems*" Ph.D. dissertation, ETH Zurich, September 2006.
10. L. Santinelli and L. Cucu-Grosjean, "*Toward probabilistic real-time calculus*" SIGBED Rev., vol. 8, pp. 54–61, March, 2011.
11. D. Kroese, T. Taimre, and Z. Botev, Handbook of MonteCarlo Methods. Wiley, 2011, vol. 706.
12. C. Mauclair, "*Analysis of Real-Time Networks with MonteCarloMethods*" http://sites.onera.fr/journeesdesthesestis/sites/sites.onera.fr.journeesdesthesestis/files/articles/JDT-TIS-2011Mauclairedric.pdf,2011.
13. J. C. Palencia and M. G. Harbour, "*Schedulability analysis for tasks with static and dynamic offsets*" in RTSS, 1998, p. 26 ff.
14. K. Gresser, "*An event model for deadline verification of hard real-time systems*" in Proceedings of the 5th Euromicro Workshop on Real-Time Systems, 1993, pp. 118–123.

Authors:

Mr. Rajib Ghosh Mr. Rajib Ghosh is presently working as an Assistant Professor in the Department of CSE with Adamas Institute of Technology-Barasat,WB (India). He obtained his B.Tech. (IT) in 2006 followed by M.Tech (CSE) from West Bengal University of Technology (India) in 2009. He has about one year of industry experience and two years of academic experiences in reputed engineering Institutions. His broad research interests are Artificial Intelligence, Cryptography and Embedded Systems.

Mr. Arnab Mitra Mr. Arnab Mitra is presently working as an Assistant Professor in the Department of CSE with Adamas Institute of Technology-Barasat,WB (India). He obtained his B.E. (IT) from University of North Bengal (India) in 2003 followed by M.Tech (CSE) from West Bengal University of Technology (India) in 2010. He has about two years of research experiences and five years of academic experiences in reputed engineering Institutions. His broad research interests are Artificial Intelligence, Theory of Computation and Embedded Systems.

Mr. Apurba Chakraborty Mr. Apurba Chakraborty is an Assistant Professor of Department of MCA, Siliguri Institute of Technology (India). He obtained his BCA, MCA and M. Tech (CSE) degree from West Bengal University of Technology (India) in 2004, 2007 and 2010 respectively. He has about Three years of academic experience. The broad area of his research interests are in Cryptography & Network Security, Embedded Systems and Artificial Intelligence.