Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2017, 4 (3): 201-208

Research Article ISSN: 2394 - 658X

Study on Enhancement on Software Quality by Scheduling Techniques of Real Time Systems

Sehrish Aqeel

Department of Computer Science, King Khalid University, Abha, Saudi Arabia
sageel@kku.edu.sa

ABSTRACT

Real time systems are type of operating systems that respond with respect to time. Quality of real time systems is of at most importance as failure of system can lead to severe results. Quality assurance procedures, methods and standards must be implemented to ensure quality of the system. Different scheduling techniques are adopted as per requirement of the system. Systems are designed according to user/organization needs and accordingly scheduling technique is followed. Whatever technique is adopted; it must follow quality attributes to make it work successfully. Paper has described dependency of scheduling on the type of the system being chosen along with task/processes working in it. Comparative analysis is performed to evaluate quality assurance of each technique discussed with respect to quality attributes. Quality can be different from organization to organization but all systems expect degree of excellence to be implemented without fail. Fault tolerance and reliability are main features of real time computing too. So Quality assurance can only guarantee effective scheduling approach to make real time system a successful sample for other systems to survive in race of modernized quality assured systems.

Keywords: Real time systems, quality, quality assurance, scheduling, comparative analysis

INTRODUCTION

In the field of computer vision, facial feature detection is an essential step to implement face recognition system. Facial features detection methods are classified as geometric based, appearance based, statistic based, colour segmentation based and template match based methods. Most previous studies related to facial feature detection focus on eye detection as the first phase of extracting the detailed information that is needed [1-2]. The main advantages of templates match based methods over others are; it does not require negative training examples or facial feature points. Creating eye template from face images requires only cropping eye images and then calculating a correlation value between the eye template and all parts of face images. Templates match based methods are also well suited for both high and low-resolution face images [3].

What is Quality?

Quality is defined differently by different scholars, Crsby has stated as ‘conformance to requirements’ [1]. Juran has defined it as ‘Fitness for use’ [2]. Edwards Deming defined quality as ‘predictable degree of uniformity and dependability with a quality standard suited to the customer’. Another definition can be ‘performance meets expectations’. Product quality is specified by quality of fit, finish, appearance, function, and performance [3]. American Society for Quality (ASQ) defines it as ‘an excellence in goods and services, particularly conformance to the requirements and customer satisfaction. This definition combines the previous ones. So if we want specific meaning of quality, it is nothing but ‘the degree of excellence’ [3]. Usually quality and reliability goes together. The customer expects good quality product that is reliable too.

Difference between Quality Assurance and Quality Control

Our research emphasizes on quality assurance. General meaning can be measures to assure quality. Here it is important to know the difference between quality assurance and quality control as both seems similar concepts. They relate with each other but at the same time differ too. Following table describes the difference between QA and QC.

Table -1 Difference Between Quality Assurance and Quality Control Adopted from [4]

Quality Assurance	Quality Control
It includes processes, procedures and standards to verify developed software and requirements.	It includes activities to verify developed software and requirements.
Emphasizes on processes and procedures.	Focuses on actual testing of the developed software to identify bug/defect, with the help of process and procedures.
It has process oriented activities.	It has product oriented activities.
It involves preventive activities.	It is a corrective process.
It is part of software test life cycle (STLC)	It is part of QA.

Let's have introduction of real time systems now.

REAL TIME SYSTEMS

A real-time system is defined by Laplante [5] as one whose correctness is based on both the correctness of the outputs and their timeliness [6]. According to Jane, real–time systems are those systems in which the correctness of the system does not depend only on the logical results of computations but also on the time at which the results are produced [7]. Barralso defined a real time system as computer system that has timing constraints and is partly specified in terms of its ability to make certain calculations or decisions in a timely manner [8].

Time is really essential bound in real time systems. System must respond within time frame in efficient way without fail. If system is unable to fulfil assign tasks within the time limit, it can have various effects depending upon the nature of real time systems [9]. May it will not have any effect, or it can lead to affect some particular function related to that system, or it can be complete disaster of the system. As per dependency to time factor, systems are divvied into two categories. First one is hard real time systems [10]. In hard real time system, task must complete within time frame without fail. Examples are traffic control system, auto pilot etc [11]. Second category is Soft real time systems. Soft real time systems consider time limit but not as tough as hard real times systems follow [12]. Examples are online attendance system, online transaction etc. Also it has been noticed that mostly systems are soft real time or weak hard real time [13].

Time factor is decided by following factors: [14,15]

Minimum Delay
Minimum time that task must wait before starting. In other words, it is waiting time of the task.

Maximum Delay
Maximum time that task can wait for starting. It is actually the upper bound so task can't wait more than this bound, it must get started at this amount of time.

Ready Time
This is the time at which scheduled task becomes ready for execution.

Run Time
Time consumed for completing the task.

Deadline
It is the end time of any executed task; task must finish execution at this allotted time limit.

Priority
Priority is an important factor which can alter the regular scheduling of tasks.

BACKGROUND

As our focus is one scheduling techniques of real time systems, we must discuss the factors that are necessary to determine scheduling technique. Mohammadi and Akl in [9] have analysed factors as:

Job
Job can be identified by time at which job arrives, waiting time for the job, execution time and deadline. This is discussed in section 1 of the paper already along with Priority factor. Once a schedule has been started, even then it can be interrupted in between by various events. Job/task as per scheduling can be categorized as: [16]

So job can be static or dynamic. In static, scheduling techniques are all set before and system will operate it in systematic way as scheduled in the system. More need arises if technique is dynamic. Let's take example of priority scheduling as dynamic, it means system can change the job dynamically by receiving any higher priority job. Dynamic scheduling can be classified as [16]:

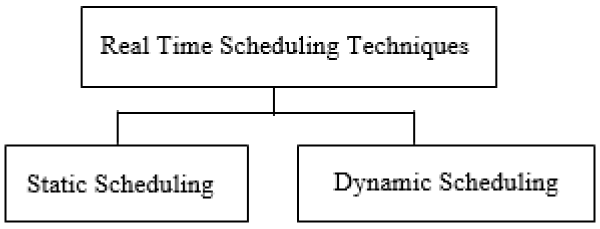

Fig. 1 Real time scheduling techniques

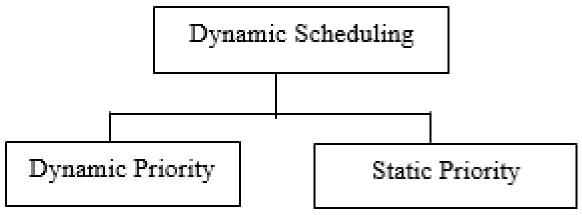

Fig. 2 Classification of dynamic scheduling in real time systems

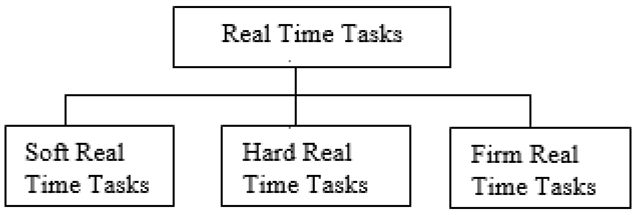

Fig. 3 Real time task classification with respect to deadline

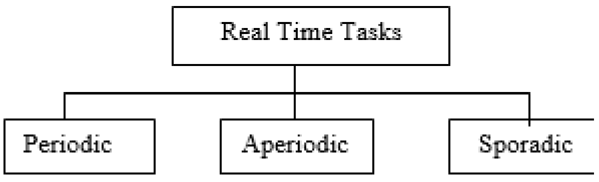

Fig. 4 Real time task classification in terms of execution

Any job cannot start unless all pre jobs of it complete execution. [17] Job execution depends upon resources associated with it too [18].

Task Categories
Task can be categorized in variety of ways.

Soft, Hard, Firm Real Time Tasks
If task has to complete within deadline, they are classified as hard real ti8me tasks. In soft real time, task can complete after deadline too. In firm real time, if task complete before deadline, they will get award. Different awarding functions can link with ending time to check the completion time with respect to deadline [19].

Periodic, A-Periodic and Sporadic Tasks
Periodic tasks repeat after a set period. Period can be deadline itself or any other scheduled period. A-periodic tasks can start irregularly without following any schedule or period. Sporadic tasks can execute irregularly but within some particular limit of time [20].

Pre-emptive and Non Pre-emptive
If one task or any event can pre-empt the other task during its execution, that will be pre-emptive task. If task can't be pre-empted by any other task or event, then that are non-pre-emptive task. As per nature as pre-emptive and non-pre-emptive, scheduling mechanism also do differ. Pre-emptive scheduling follows different techniques and algorithms as compared to the other one [21]. Normally Pre-emptive scheduling provides better scheduling than non-pre-emptive approach.

Dependent and Independent Tasks
Lastly another classification can be based on dependency on any other task. So tasks if they are dependent on any other task are classified as dependent tasks. Inversely task that doesn't depend on any other task is independent task.

Single or Multiprocessor Real Time System
Real time system can be of uni-processor or more than one processor connected via any set topology. According to number of processors, scheduling scheme is being implemented. Here we will focus on uni-processor scheduling techniques only.

Dynamic scheduling is basically scheduling based on priority. Now priority can be fixed priority working as Static priority in dynamic scheduling, or it can be dynamic priority. Examples of static priority driven scheduling are rate monolithic (RM) and deadline monolithic (DM) [24]. Similarly, earliest deadline first (EDF) and Least slack time first (LSTF) are dynamic priority scheduling examples in real time scheduling [25].

Let's have a little introduction of these scheduling techniques.

Rate Monolithic
It is a category of dynamic scheduling working with priority assignment. Priority is statistically given to system as per the cycle of the job, so cycle length decides priority of the job. Short cycle will have more priority and vice versa. Scheduling is usually pre-emptive in nature [26].

Deadline Monolithic

DM works under set deadline and must accomplish job within that deadline. If any task takes more time than deadline, Audsley’s algorithm works to find optimal priority and then assign it to jobs to ensure that tasks will meet deadline within the period of time. So basically priorities are static but work under fixed periods [27].

Earliest Deadline First

Another name of it is least time to go. It is dynamic scheduling technique working under dynamic priorities. Here priority queue can change regularly based on new job or any job ending etc. Based on priority, task which is having shortest deadline will be given higher priority. So basically tasks are weighted according to their finishing time. As this priority can change too, that’s why categorized under dynamic priority scheduling.

Least Slack Time First

Another name is Least Laxity First. Here dynamic priorities are assigned according to slack time. Slack time is the remaining time of a job, once being started. In other words, it is the difference between deadline, ready time and run time. It is most commonly used in real time and embedded systems especially for multiprocessor designs [23].

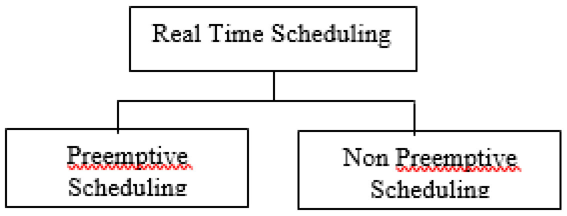

Fig. 5 Real time scheduling techniques

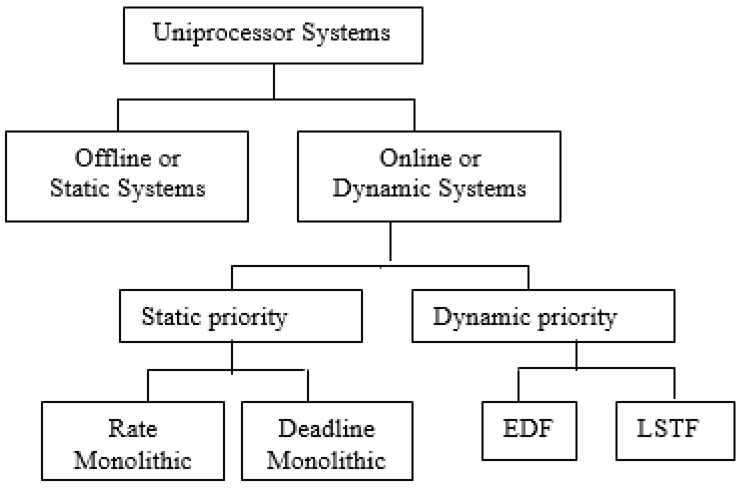

Fig .6 Uniprocessor scheduling [16, 22]

Improvement in Fixed Priority Scheduling

As fixed priority generally doesn’t provide good scheduling as compared to dynamic one, different researchers have tried to find solution to solve this problem. Priority inversion is also used to make better scheduling in fixed priority. Deferred pre-emption [28], Pre-emption threshold scheduling [29] and Quantum based scheduling [30] are examples of different techniques proposed to enhance scheduling of fixed priority u7sung priority inversion.

Controlling Task Release

CTR is a fixed priority scheduling technique that improves schedulability by controlling released tasks. It blocks the release of tasks for some period of time so that already executed task can complete efficiently. At run time, tasks will be blocked state and released at their release time only. Block time is assigned to each task and they can block till that time only. Blocking doesn’t affect deadline of the processes but it really works for lower priority executing processes to complete their execution with extra space. CTR provides better schedulability as compared to RM Preemptive scheduling and other fixed priority techniques [31].

EXAMPLES OF SUCCESSFUL REAL TIME SYSTEMS

Airborne Safety-Critical Software

Safety critical software proposed by DTB is used to simulate the FMS with a real aircraft environment and conditions. FMS which once installed on an airplane might fail and result in terrible consequences. FTA [32], FMEA [33], HAZOP [34] are examples of those techniques.

FTA represents errors or faults graphically. FMEA is a quality planning tool that checks' Cause of Failure, Effect of Failure, Frequency of Occurrence, Degree of Severity, Chance of Detection, Risk Priority, Design Action, and Design Validation. HAZOP assumes deviation of design has caused risk. In safety critical software, different standards are followed by different software companies but three standards are quite common, that are standards followed by aircraft industry, nuclear energy and railway transport. Generally, the main tasks are collecting safety requirements and accessing the system and software [35].

Real Time Attendance Logging with Multi Node Embedded System Connected Via Wi-Fi

Real time attendance logging system with more than one node is designed that works with Wi-Fi. Wi-Fi is most demanding internet technology and its presence is ensured in proposed system that not only combines LAN embedded system features but also Wi-Fi technology to work widely other than LAN too. System guarantees accuracy in data collection, recognition, processing, integrity and security too [36].

Performance Evaluation of Intelligent Adaptive Traffic Control Systems

SCATS's effectiveness in controlling traffic with intelligent management of traffic routes is discussed in [37]. Working as effective traffic control, system should provide less delay and less travel time to users for different routes. Here need of enabling more parameters in design phase is also elaborated. Overall proposed system provides successful traffic control over heavy traffic especially at intersection nodes.

ANALYSIS

Section 2 discussed various scheduling techniques working in real time systems. Main purpose of any scheduling algorithm is to provide better scheduling by making reducing delay time of jobs, applying fairness and ensuring maximum utilization of CPU. Let's discuss scheduling techniques in terms of quality assurance principles.

Pre-emptive Scheduling and Quality Assurance

As pre-emptive scheduling allows jobs to be pre-empted in case of higher priority job, it means quality attributes of performance is achieved. But at the same time availability of jobs that have low priority can suffer. It will provide good performance for higher priority jobs or jobs that have short life cycle but other jobs can have starvation issue.

Non Pre-emptive Scheduling and Quality Assurance

Another scheduling approach is as non-pre-emptive which means job will not pre-empt unless it finishes. Best example of it is FCFS. This approach of scheduling ensures availability, reliability, scalability and maintainability but performance which is the main quality attribute can suffer. For example, first job itself can have maximum length as compared to other jobs in the queue, so it must finish then other jobs can get their turn. Delay time will increase, CPU utilization will suffer and it will not be fair for short jobs in the queue too.

Scheduling in Offline Systems and Quality Assurance

An Offline system doesn't have that much challenging issue as online systems have. They will be working on static scheduling algorithm under the requirement of system itself. Here system quality is more important. Supportability and testability are better served here. Also systems will possess integrity, security and availability attributes to serve jobs in pre-defined scheduling mechanism. Scheduling approach can be pre-emptive or non-pre-emptive depending upon kind of system implemented as per user requirements.

Static priority Scheduling and Quality Assurance

Static priority scheduling is not considered as good scheduling approach because of unfair CPU utilization and jobs latency. Lots of approaches have been adopted to make it work in good way like priority inversion as discussed in section 2.3. Again it depends on the nature of the system. If system requires static priority to be implemented as per user requirement, all other attributes become side line as usability is achieved along with maintainability, availability and supportability. As per OS, may be any scheduling algorithm is not providing good scheduling scheme but approach of using any scheduling algorithm depends on many factors. Top of it is user requirements for whom system is designed. If system fulfils user requirements as per QA, it would be good quality software.

Dynamic Priority Scheduling and Quality Assurance

Dynamic priority serves as good scheduling technique in terms of changing priority dynamically based on best suitable approach. At the same time performance is also increased as per methodology adopted of good scheduler. System must be reliable to ensure smooth running of scheduling scheme to make it robust and fault tolerant. Scalability and usability are also achieved in dynamic priority scheduling approach.

Planning based Scheduling and Quality Assurance

Planning based scheduling works on the principle that if any task is accepted, it must complete its execution. Execution is based on dynamic priority that can be assigned based on many criteria like resource consumption or length of job etc. Planning based scheduling ensures quality in terms of availability, performance and reliability.

Best Effort Scheduling and Quality Assurance

DASA (Dependent Activity Scheduling Algorithm) and LBESA (Lock Best Effort Scheduling Algorithm) are popular examples of best effort scheduling algorithms. In best effort, scheduler finds the best solution under provided condition. In other words, it provides accurate befitted solution in given scenario. It is the most intelligent form of scheduling performed in real time systems. It fulfils majority of quality attributes and system makes check by QA that quality of the system is perfectly achieved with the implementation of scheduling algorithm chosen. Reliability, Performance, Security, Usability are main features of best effort scheduling approach of real time systems.

Uniprocessor Scheduling and Quality Assurance

Quality is better practiced and evaluated also in single processor real time systems as compare to multiprocessor real time systems. As system is based on single processor, implementing availability, security, integrity, usability and ensuring performance will be manageably. Also if any fault happens or system fails to observe quality behaviour, it can be quickly repaired to ensure quality making system reliable and persistent.

Multiprocessor Scheduling and Quality Assurance

Already in this section scheduling techniques are being discussed. This section will just put focus QA implementation in more than one processor. Obviously it is difficult as compared to single processors as one processor failure can disturb the complete structure. For successful structures, different designs and architectures are already working. But here mainly fault tolerance is the main issue. In multiprocessors, need of QA is more and testability is implemented in advanced way to ensure that system will handle different kinds of behaviour in real time computing. Also this design has alternate approaches too that can be used in case of any node failure. As quality is important here too, availability, scalability, usability, testability, maintainability and security are ensured in effective way to make system work without fail.

Table -2 Good Scheduler Attributes with Discussed Scheduling Approaches

Good scheduler attributes	Pre-emptive	Non- Pre-emptive	Offline	Static priority	Dynamic priority
Minimum delay of jobs	✓				✓
CPU utilization	✓				✓
Fairness		✓	✓	✓	Some times

Table -3 Scheduling Schemes with Quality Attributes

QA achieved by	Pre-emptive	Non-pre-emptive	Offline	Static priority	Dynamic priority
Integrity	Can effect	✓	✓	✓	Can effect
Availability	Lower priority can suffer	Some extent	✓	✓	✓
Maintainability	Can effect	✓	✓	✓	✓
Manageability	✓	Can effect	✓	✓	Can effect
Performance	✓	Can effect	Can effect	Can effect	✓
Reliability	✓	Can effect	✓	✓	✓
Reusability	✓	✓	Can effect	can effect	✓
Security	✓	✓	✓	✓	✓
Scalability	✓	Can suffer	Can suffer	Can suffer	✓
Testability	Can suffer	✓	✓	✓	Can suffer
User experience	✓	Can suffer	Depends upon system	Depends on system	✓
Supportability	✓	✓	✓	✓	Can suffer
interoperability	✓	Can suffer	Not supported	Not supported	✓

Comparison of Scheduling Approaches as Good Scheduler

Table -1 describes scheduling approaches used in real time system in term of good scheduler. We have discussed three attributes of good scheduler here. Apparently Pre-emptive approach and dynamic scheduling with dynamic priority scheduling best serve as good scheduler as compared to other. But we can’t say that other scheduling techniques are useless. It depends upon the requirement of the system and users. Most probably for certain requirement offline system serves best as compared to others, so it is purely requirement based solution that which scheduling approach should be selected in given requirements.

Comparison of Scheduling Approaches with Quality Assurance

Table -2 has summarized implementation of processes, procedures and standards to ensure mentioned quality attributes in scheduling approaches used in real time systems. Almost all discussed approaches are achieving quality by one or another way as without enforcing quality assurance, none of the scheme can work in object oriented way to achieve objectives or goals of manufactured system.

CONCLUSION

Quality differs from system to system, product to product and organization to organization. Way to describe quality also changes as per system requirements but one thing is common that is degree of excellence. For real time systems, quality needs reliability, fault tolerance and completion of tasks on time. As real time systems are recognized for functionality as per time constraints. There are different scheduling approaches based on uni-processor and multi-processor. Each approach has various task categories like dependent and independent, periodic/a-periodic or Sporadic and hard/soft or firm real time task. Scheduling approach depends upon type of the task and usability requirement (for whom we are designing the system). Organization specifies requirements to Project lead and designers designs the system that best fits the requirements. Designers decide which kind of task, job, processes, scheduling technique and other essential of software can best match the required demands. Most popular scheduling approaches that are discussed are pre-emptive and non-pre-emptive scheduling approaches. Similarly, offline systems follow different strategy of scheduling as compared to dynamic systems. Dynamic scheduling is further sub divided as static priority driven systems and dynamic priority driven systems. Each scheduling approach must adopt quality assurance and TABLE 3 has drawn to shown comparative analysis of scheduling techniques with quality attributes that must be followed to ensure quality in the system. Results state that Dynamic priority has achieved maximum quality by fulfilling availability, manageability, performance, reliability, usability, reusability, security, stability and interoperability. Other scheduling approaches also do follow various quality attributes. It depends upon system requirements that which technique will be preferred as sometimes offline systems can best suit the user expectations as compared to dynamic systems. So ball is in organization hand how to implement quality and which scheduling approach should be selected for real time system. Whatever technique will be; real time systems expect quality assurance from organization to maintain quality for its successful implementation in real time computing. How these attributes are adopted by scheduling techniques is an open area for future work.

List of Abbreviations

EDF: Earliest Deadline First	**SCATS:** Sydney Coordinated Adaptive Traffic System
LSTF: and Least slack time first	**FMEA:** Failure Mode and Effect Analysis
RM: Rate monolithic	**HAZOP:** Hazard Operability Analysis
DM: Deadline monolithic	**DASA:** Dependent Activity Scheduling Algorithm
CTR: Controlling task release	**LBESA:** Lock Best Effort Scheduling Algorithm
DTB: Dynamic Test Bed	**FCFS:** First come first served
OS: Operating system	**FMS:** Flight Management System
QA: Quality assurance	**FTA:** Fault Tree Analysis

REFERENCES

[1] Collins, *Dictionary of the English Languages*, Collins, Australia ,**1979**.
[2] JM Juran, *Juran on Quality by Design*, Macmillan, USA, **1992**.
[3] Tirupathi R Chandrupatla, *Quality and Reliability in Engineering*, Cambridge University Press, **2009**.
[4] *https://www.tutorialspoint.com/software_testing/software_testing_qa_qc_testing.htm*
[5] PA Laplante, *Real-Time Systems Design and Analysis: An Engineers Handbook*, Second Edition, IEEE Press, **1997**.
[6] F Schlindwein, EG7017 – Real-time DSP, *http://www.le.ac.uk/eg/fss1/real%20time.htm*, **2002**.
[7] JWS Liu, *Real-Time Systems*, Pearson Education, India, **2001**.
[8] M Barr, *Programming Embedded Systems in C and C++* , O Reilly and Associates, Inc. USA, **1999**.
[9] Arezou Mohammadi and Selim G Akl, *Scheduling Algorithms for Real-Time Systems*, Technical Report No. 2005-499, School of Computing, Queen's University Kingston, Ontario, Canada K7L 3N6, **2005**.
[10] A Burns, Scheduling Hard Real-Time Systems: A Review, *Software Engineering Journal*, **1991**, 3, 116-128.
[11] Chandra Mouli and Smriti Agrawal, An (M, K) Model Based Real-Time Scheduling Technique for Security Enhancement, *International Journal of Computer Science and Information Security*, **2015**, 13 (10), 10-18.

[12] R Al-Omari, AK Somani and G Manimaran, An Adaptive Scheme for Fault-Tolerant Scheduling of Soft Real-Time Tasks in Multiprocessor Systems, *Journal of Parallel and Distributed Computing*, **2005**, 65 (5), 595-608.
[13] G Bernat, A Burns and A Llamosi, Weakly Hard Real-time Systems, *IEEE Transactions on Computers*, **2001**, 50 (4), 308 – 321.
[14] PA Laplante, Real-time Systems Design and Analysis, An Engineer Handbook, *IEEE Computer Society*, *IEEE Press*, **1993**.
[15] JA Stankovic and K Ramamritham, Tutorial Hard Real-Time Systems, *IEEE Computer Society Press*, **1988**.
[16] M Kaladevi and S Sathiyabama, A Comparative Study of Scheduling Algorithms for Real Time Task, *International Journal of Advances in Science and Technology*, **2010**, 1(4), 8-14.
[17] JW de Bakker, C Huizing, WP de Roever and G Rozenberg, Real-Time: Theory in Practice, *Proceedings of REX Workshop*, Mook, The Netherlands, Springer-Verlag Company, **1991**.
[18] JA Stankovic and K Ramamritham, Tutorial Hard Real-Time Systems, *IEEE Computer Society Press*, **1988**.
[19] M Joseph, *Real-Time Systems: Specification, Verification and Analysis*, Prentice Hall, **1996**.
[20] D Isovic and G Fohler, Efficient Scheduling of Sporadic, Aperiodic and Periodic Tasks with Complex Constraints, *Proceedings of the 21st IEEE RTSS*, Florida, USA, **2000**.
[21] CL Liu and JW Layland, Scheduling Algorithms for Multiprogramming in a Hard-Real- Time Environment, *Journal of the ACM*, **1973**, 20 (1), 46-61.
[22] JA Ayeni, AE Odion and IF Ogbormor-Odikayor, An Analytical Survey of Real Time System Scheduling Techniques, *International Journal of Science and Research*, **2014**, 3(10), 1774- 1778.
[23] G Saini, Application of Fuzzy Logic to Real-Time Scheduling, *14th IEEE-NPSS Real Time Conference*, Stockholm, Sweden, **2005**.
[24] J Liu, *Real-Time Systems*, Pearson Education, **2000**.
[25] K Kotecha and A Shah, ACO based Dynamic Scheduling Algorithm for Real-Time Multiprocessor Systems, *International Journal of Grid and High Performance Computing*, **2011**, 3 (3), 20-30.
[26] J Layland, Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment, *Journal of the ACM*, **1973**, 20 (1), 46–61.
[27] JY Leung and J Whitehead, On the Complexity of Fixed-Priority Scheduling of Periodic, Real-Time Tasks, *Performance Evaluation*, **1982**, 2 (4), 237–250.
[28] A Thekkilakattil, R Dobrin and S Punnekkat, The Limited-Pre-Emptive Feasibility of Real-Time Tasks on Uniprocessors, *Real-Time System*, **2015**, 51(3), 247-273.
[29] Y Wang and M Saksena, Scheduling Fixed Priority Tasks with Pre-Emption Threshold, Proceedings of the *6th International Conference on Real Time Computing Systems and Applications*, Hong Kong, **1999**, 328-335.
[30] M Park, HJ Yoo and J Chae, Analysis on Quantum-Based Fixed Priority Scheduling of Real-Time Tasks, *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication*, Korea, **2009**, 627-634.
[31] B Mahmood, SR Malik, N Ahmad and A Anjum, Enhancement in System Schedulability by Controlling Task Releases, *International Journal of Advanced Computer Science and Applications*, **2016**, 7(7), 436-445.
[32] NG Leveson, SS Cha and TJ Shimeall, Safety Verification of Ada Programs using Software Fault Trees, IEEE Software, **1991**, 8 (4), 48-59.
[33] T Maier, FMEA and FTA to Support Safety Design of Embedded Software in Safety-Critical Systems. *Proceedings of the ENCRESS Conference on Safety and Reliability of Software Based Systems*, Bruges, **1995**.
[34] F Redmill, M Chudleigh and J Catmur, *System Safety: HAZOP and Software HAZOP*, John Wiley & Sons, New York, **1999**.
[35] Nadia Bhuiyan and Habib A El Sabbagh, A Quality Assurance Model for Airborne Safety-Critical Software, *Journal of Software Engineering and Applications*, **2014**, 7, 162-176.
[36] Mohammed Bilal, Real Time Attendance Logging with Multi Node Embedded System Connected Via Wi-Fi, *International Journal of Reconfigurable and Embedded Systems*, **2012**, 1(3), 103-107.
[37] S Samadi, AP Rad, FM Kazemi and H Jafarian, Performance Evaluation of Intelligent Adaptive Traffic Control Systems: A Case Study, *Journal of Transportation Technologies*, **2012**, 2, 248-259.