ENGINEERING REAL TIME SYSTEMS pdf

1: Real-time computing - Wikipedia

3 Hard Real-Time Systems wSystem where a failure to respond within a given time causes a critical failure. wExample: Nuclear Power Plant n If the system doesn't notice a problem with the.

You should also have a basic working knowledge of UNIX like operating systems and commands, as well as the ability to program and debug within an integrated development environment IDE such as Eclipse or Visual Studio. Course Description This course covers the design and implementation of multi-threaded and real-time systems, with particular emphasis on real-time systems for control of industrial processes and for embedded systems based on ARMv7 and x86 architectures. Contents of the course include: Features and characteristics of real-time systems. Concurrent processes and mutual exclusion operations. Inter process communication and message passing between programs running on the same system or another system on the network. Analysis and design of real- time systems. Real- time scheduling principles. You will be required to demonstrate your understanding by applying your gained knowledge to a week project using a commercial grade industrial real-time operating system and development environment. Please note that if you take this course for a bachelor honours program, your overall mark in this course will be one of the course marks that will be used to calculate the weighted average mark WAM that will determine your award level. This applies to students who commence enrolment in a bachelor honours program from 1 January onward. On completion of this course you should be able to: Characterise real time systems and describe their functions. Analyse, design and implement a real-time system. Apply formal methods to the analysis and design of real-time systems. Apply formal methods for scheduling real-time systems. Characterise and debug a real-time system. Overview of Learning Activities Student learning occurs through the following experiences and evaluation processes: A series of 10 2 hour lectures will guide you to important concepts and give you many practical hints for the design of real-time systems. The laboratory work will help you to connect theory with practice. The project is a problem based learning activity that will require you to exercise many of the skills required for real-time design and implementation. It will also help develop your team skills and give you experience with an industry leading real-time operating system and associated development tools. The course resources accessible from the Web have links to on-line resources for you to access and expand your knowledge of the topics. Overview of Learning Resources The learning resources for this course include: Lecture notes prepared by the Teaching staff. Electronic media Linux based live-DVD image containing a real time development environment will be made available to assist with independent study. See the course guide Part B available at the start of classes for the list of recommended references. Supplementary course content eg. There are three key components: The formalised laboratory tasks will help you gain competence in solving problems in real time systems as well as teach you the fundamental concepts that are required to solve the group project task. This will help develop teamwork and leadership skills as well as provide the potential to use software version-control and collaboration tools. The final exam will test your individual capabilities. This includes your ability to describe the principles of real time systems, and apply techniques covered in lectures to solve various real time systems problems. Written feedback will be provided on all written reports. Face-to-face feedback will be provided for anything marked during the laboratory sessions and final group project demonstration session. Laboratory Tasks Five laboratory tasks will be held in the first 6 weeks of the course.

2: NPTEL :: Computer Science and Engineering - Real Time Systems

Real-time systems. The design of real-time systems is becoming increasingly important. Computers have been incorporated into cars, aircraft, manufacturing assembly lines, and other applications to control processes as they occurâ€"known as "in real time.".

Describing the architectural design i. While notations have been defined to describe abstract models and implementations, little work has been done in order to define a notation While notations have been defined to describe abstract models and implementations, little work has been done in order to define a notation for architectural design. Solutions are proposed in UML for describing the system software structure and the deployment of software components on a hardware platform. We believe that these UML solutions are not yet mature and that they are incomplete. In our work, we have clarified and extended them in order to model the architectural design. Our work has thus focused on the transition from an SDL specification to a concrete system. However, the solutions that we propose are general and may be applied when using other notations for modelling the abstract system, e. In this document, we first introduce the architectural design concepts and present the SOON notation. Then we describe the implementation concepts defined in UML and present how these concepts can, together with the UML extension mechanisms, be used for architectural design. Generating code automatically from the design level increases product quality and productivity but also facilitates maintenance and evolution by limiting changes to the design level. Flexibility is a basic requirement that should be fulfilled by automatic code generators: We present our approach to flexible code generation, in the frame of our SDL methodology, and the code generator tool ProgGen. The methodology focuses on the design phase and distinguishes between functional design i. The implementation design description plays a central role in the code generation process. ProgGen is a generic tool which can be used to produce SDL translators; the output is controlled by a set of code skeletons. The skeletons can easily be tailored to support For the rst time, the textual description of a standard has no priority over the corresponding SDL speci- cation, which in the case of Core INAP CS-2 is published as a normative annex to the protocol standard. The power of a standard SDL specication has been shown by the successful application of computer aided test generation methods for the production of the necessary standard conformance test suites. Show Context Citation Context The SDL denition includes many additional language constructs which cannot be introduced here. Figure 4 shows an example of an MSC.

3: Realtime Utility Engineers

The only book you will find about software engineering/design for real-time systems that covers as many topics and with the depth (in some areas) that I haven't found anywhere else. It covers methods/design based on Structured, data flow and Object Oriented.

Introduction Real-Time systems span several domains of computer science. They are defense and space systems, networked multimedia systems, embedded automative electronics etc. In a real-time system the correctness of the system behavior depends not only the logical results of the computations, but also on the physical instant at which these results are produced. A real-time system changes its state as a function of physical time, e. Based on this a real-time system can be decomposed into a set of subsystems i. A real-time computer system must react to stimuli from the controlled object or the operator within time intervals dictated by its environment. The instant at which a result is produced is called a deadline. If the result has utility even after the deadline has passed, the deadline is classified as soft, otherwise it is firm. If a catastrophe could result if a firm deadline is missed, the deadline is hard. Commands and Control systems, Air traffic control systems are examples for hard real-time systems. On-line transaction systems, airline reservation systems are soft real-time systems. The first two classifications, hard real-time versus soft real-time, and fail-safe versus fail-operational, depend on the characteristics of the application, i. The second three classifications, guaranteed-timeliness versus best-effort, resource-adequate versus resource-inadequate, and event-triggered versus time-triggered, depend on the design and implementation, i. However this paper focuses on the differences between hard and soft real-time classification. The response time requirements of hard real-time systems are in the order of milliseconds or less and can result in a catastrophe if not met. In contrast, the response time requirements of soft real-time systems are higher and not very stringent. In a hard real-time system, the peak-load performance must be predictable and should not violate the predefined deadlines. In a soft real-time system, a degraded operation in a rarely occurring peak load can be tolerated. A hard real-time system must remain synchronous with the state of the environment in all cases. On the otherhand soft real-time systems will slow down their response time if the load is very high. Hard real-time systems are often safety critical. Hard real-time systems have small data files and real-time databases. Temporal accuracy is often the concern here. Soft real-time systems for example, on-line reservation systems have larger databases and require long-term integrity of real-time systems. If an error occurs in a soft real-time system, the computation is rolled back to a previously established checkpoint to initiate a recovery action. Real-Time Scheduling A hard real-time system must execute a set of concurrent real-time tasks in a such a way that all time-critical tasks meet their specified deadlines. Every task needs computational and data resources to complete the job. The scheduling problem is concerned with the allocation of the resources to satisfy the timing constraints. Figure 2 given below represents a taxonomy of real-time scheduling algorithms. Real-Time scheduling can be categorized into hard vs soft. Hard real-time scheduling can be used for soft real-time scheduling. Some of the research on QoS [Klara95] addresses this problem in detail and is not covered here. The present paper focuses on scheduling algorithms for hard real-time. Hard real-time scheduling can be broadly classifies into two types: In static scheduling, the scheduling decisions are made at compile time. A run-time schedule is generated off-line based on the prior knowledge of task-set parameters, e. So run-time overhead is small. More details on static scheduling can be found in [Xu90]. On the otherhand, dynamic scheduling makes its scheduling decisions at run time, selecting one out of the current set of ready tasks. Dynamic schedulers are flexible and adaptive. But they can incur significant overheads because of run-time processing. Preemptive or nonpreemptive scheduling of tasks is possible with static and dynamic scheduling. In preemptive scheduling, the currently executing task will be preempted upon arrival of a higher priority task. In nonpreemptive scheduling, the currently executing task will not be preempted until completion. Dynamic Scheduling Algorithms Schedulability test often used by dynamic schedulers to determine whether a given set of ready tasks can be scheduled to meet their deadlines. Different scheduling algorithms and their schedulability criteria is explained below. The rate monotonic algorithm assigns static priorities based on task periods. Here task period is the time after which the tasks repeats and inverse of period is task arrival rate. For example, a task with a period of 10ms repeats itself after every 10ms. The task with the shortest period gets the highest priority, and the task with the longest period gets the lowest static priority. At run time, the dispatcher selects the task with the highest priority for execution. According to RMA a set of periodic, independent task can be scheduled to meet their deadlines, if the sum of their utilization factors of the n tasks is given as below. EDF algorithm is an optimal dynamic preemptive algorithm based on dynamic priorities. In this after any significant event, the task with the earliest deadline is assigned the highest dynamic priority. A significant event in a system can be blocking of a task, invocation of a task, completion of a task etc. The dispatcher operates in the same way as the dispatcher for the rate monotonic algorithm. The Priority Ceiling Protocol: The priority ceiling protocol [Lui90] is used to schedule a set dependant periodic tasks that share resources protected by semaphores. The shared resources, e. The sharing of resources can lead to unbounded priority inversion. The priority ceiling protocols were developed to minimize the priority inversion and blocking time. Static Scheduling Algorithms In static scheduling, scheduling decisions are made during compile time. This assumes parameters of all the tasks is known a priori and builds a schedule based on this. Once a schedule is made, it cannot be modified online. Static scheduling is generally not recommended for dynamic systems. Applications like process control can benefit from this scheduling, where sensor data rates of all tasks are known before hand. There are no explicit static scheduling techniques except that a schedule is made to meet the deadline of the given application under known system configuration. Most often there is no notion of priority in static scheduling. Based on task arriaval pattern a time line is built and embedded into the program and no change in schedules are possible during execution.

4: CiteSeerX â€” Citation Query Engineering Real Time Systems

Real-time systems are expanding to several other domains such as automative industry and embedded real-time systems. Especially the marriage of the Internet with multimedia applications has opened several new volume applications.

History[edit] The term real-time derives from its use in early simulation , in which a real-world process is simulated at a rate that matched that of the real process now called real-time simulation to avoid ambiguity. Analog computers , most often, were capable of simulating at a much faster pace than real-time, a situation that could be just as dangerous as a slow simulation if it were not also recognized and accounted for. Real-time operating systems would also be used for time-sharing multiuser duties. For example, Data General Business Basic could run in the foreground or background of RDOG and would introduce additional elements to the scheduling algorithm to make it more appropriate for people interacting via dumb terminals. The possibility to deactivate other interrupts allowed for hard-coded loops with defined timing, and the low interrupt latency allowed the implementation of a real-time operating system, giving the user interface and the disk drives lower priority than the real-time thread. Compared to these the programmable interrupt controller of the Intel CPUs However, several coding libraries exist which offer real time capabilities in a high level language on a variety of operating systems, for example Java Real Time. The Motorola and subsequent family members , etc. This application area is one in which real-time control offers genuine advantages in terms of process performance and safety. The usefulness of a result is zero after its deadline. Thus, the goal of a hard real-time system is to ensure that all deadlines are met, but for soft real-time systems the goal becomes meeting a certain subset of deadlines in order to optimize some application-specific criteria. The particular criteria optimized depend on the application, but some typical examples include maximizing the number of deadlines met, minimizing the lateness of tasks and maximizing the number of high priority tasks meeting their deadlines. Hard real-time systems are used when it is imperative that an event be reacted to within a strict deadline. Such strong guarantees are required of systems for which not reacting in a certain interval of time would cause great loss in some manner, especially damaging the surroundings physically or threatening human lives although the strict definition is simply that missing the deadline constitutes failure of the system. For example, a car engine control system is a hard real-time system because a delayed signal may cause engine failure or damage. Other examples of hard real-time embedded systems include medical systems such as heart pacemakers and industrial process controllers. Hard real-time systems are typically found interacting at a low level with physical hardware, in embedded systems. Early video game systems such as the Atari and Cinematronics vector graphics had hard real-time requirements because of the nature of the graphics and timing hardware. In the context of multitasking systems the scheduling policy is normally priority driven pre-emptive schedulers. Soft real-time systems are typically used to solve issues of concurrent access and the need to keep a number of connected systems up-to-date through changing situations. An example can be software that maintains and updates the flight plans for commercial airliners: Live audio-video systems are also usually soft real-time; violation of constraints results in degraded quality, but the system can continue to operate and also recover in the future using workload prediction and reconfiguration methodologies. That means that the mean processing time per sample, including overhead , is no greater than the sampling period, which is the reciprocal of the sampling rate. This is the criterion whether the samples are grouped together in large segments and processed as blocks or are processed individually and whether there are long, short, or non-existent input and output buffers. Consider an audio DSP example; if a process requires 2. However, if it takes 1. A common life analog is standing in a line or queue waiting for the checkout in a grocery store. If the line asymptotically grows longer and longer without bound, the checkout process is not real-time. If the length of the line is bounded, customers are being "processed" and output as rapidly, on average, as they are being inputted and that process is real-time. The grocer might go out of business or must at least lose business if they cannot make their checkout process real-time; thus, it is fundamentally important that this process is real-time. A signal processing algorithm that cannot keep up with the flow of input data with output falling farther and farther behind the input is not real-time. But if the delay of the output relative to the input is bounded regarding a process that operates over an unlimited time, then that signal processing algorithm is real-time, even if the throughput delay may be very long. Live audio digital signal processing requires both real-time operation and a sufficient limit to throughput delay so as to be tolerable to performers using stage monitors or in-ear monitors and not noticeable as lip sync error by the audience also directly watching the performers. Tolerable limits to latency for live, real-time processing is a subject of investigation and debate but is estimated to be between 6 and 20 milliseconds. Real-time and high-performance[edit] Real-time computing is sometimes misunderstood to be high-performance computing , but this is not an accurate classification. Conversely, once the hardware and software for an anti-lock braking system have been designed to meet its required deadlines, no further performance gains are obligatory or even useful. Furthermore, if a network server is highly loaded with network traffic, its response time may be slower but will in most cases still succeed before it times out hits its deadline. Hence, such a network server would not be considered a real-time system: In a real-time system, such as the FTSE Index , a slow-down beyond limits would often be considered catastrophic in its application context. Therefore, the most important requirement of a real-time system is predictability and not performance. Some kinds of software, such as many chess-playing programs , can fall into either category. For instance, a chess program designed to play in a tournament with a clock will need to decide on a move before a certain deadline or lose the game, and is therefore a real-time computation, but a chess program that is allowed to run indefinitely before moving is not. In both of these cases, however, high performance is desirable: This example also illustrates the essential difference between real-time computations and other computations: High-performance is indicative of the amount of processing that is performed in a given amount of time, whereas real-time is the ability to get done with the processing to yield a useful output in the available time. Near real-time[edit] The term "near real-time" or "nearly real-time" NRT , in telecommunications and computing , refers to the time delay introduced, by automated data processing or network transmission, between the occurrence of an event and the use of the processed data, such as for display or feedback and control purposes. For example, a near-real-time display depicts an event or situation as it existed at the current time minus the processing time, as nearly the time of the live event. The term implies that there are no significant delays. Near real-time also refers to delayed real-time transmission of voice and video. It allows playing video images, in approximately real-time, without having to wait for an entire large video file to download. The distinction between "near real-time" and "real-time" varies, and the delay is dependent on the type and speed of the transmission. The delay in near real-time is typically of the order of several seconds to several minutes.

5: Computer science - Real-time systems | www.enganchecubano.com

Real-Time Task Scheduling on Multiprocessors and Distributed Systems (Contd.) Clock Synchronization in Distributed Real-Time Systems Internal Clock Synchronization in Presence of Byzantine Clocks.

6: ECE Real-Time Systems | ECE | Virginia Tech

Real-time systems find application in command and control systems, process control, flight control, avionics, defense systems, vision and robotics, pervasive and ubiquitous computing, and an abundance of embedded systems.

7: RTS - Institute of Systems Engineering - Real Time Systems Group

It is also an excellent textbook for graduate courses in computer engineering, computer science, information technology, and software engineering on embedded and real-time software systems, and for undergraduate computer and software engineering courses.

8: Real Time Systems Engineering - RMIT University

ENGINEERING REAL TIME SYSTEMS pdf

This course covers the design and implementation of multi-threaded and real-time systems, with particular emphasis on real-time systems for control of industrial processes and for embedded systems based on ARMv7 and x86 architectures.

9: Real-Time Systems

for Real-Time Systems and associated relevant core utility systems. Works in conjunction with business operations, technology, and software vendors to support the design, build, and testing phases of new implementations, enhancements, or maintenance of Real-Time system projects such as EMS, SCADA, DMS/ADMS, DA, OMS, TOA, PI Historian, GIS.

Grange of Illinois. The pagans and the cross The Saints Everlasting Rest Filetype lightfoot the development of children 7e Israel : a community with prophets and visionaries Frommers Washington State (Frommers Complete) Asimov foundation and empire THE CAVE DWELLERS A PLAY IN TWO ACTS How to Sketch Animals Rules for using historical records at the National Archives and Records Administration The norton anthology of american literature shorter 8th ed Create Your Own Joy Ritual for small churches The Early American Chroniclers Metallurgy fundamentals Equestrian Vaulting Hip Hop in American Cinema Spanish for nurses New travels into the interior parts of Africa General epistles (James; I II Peter; I, II, III John; Jude) Techno-rusticity Martyn Wiltshire Administrative law notebook Pt. 1. Enduring issues Junior high math facts for summer Driving Guides to America Analysis and synthesis of single-input single-output control systems The Beginnings Of Faith 74 Truth matters walter veith Dariens Angelwalk for Children All of me trumpet sheet music Contemporary project management 2nd edition Annual Review of Medicine, 2002 Dawn And the Darkest Hour Kudankulam nuclear power plant design in The heart of the monarchy Adam Zertal Introduction to Management of Reverse Logistics and Closed Loop Supply Chain Processes Grace for the widow An American slave society Westin, A. F. Also on the bench: / Leadership in the Church