Real time System:

Anuja Anjal
Roll no- 171704

Assignment 1:

Q1. Explain different type of Real time Operating Systems and the common misconceptions about Real time systems.

Ans: A real time system is a computer system that must satisfy bounded response-time constraints or risk severe consequences, including failure, i.e. a system whose logical correctness is based on both the correctness of the outputs and their timeliness.

Real time Operating System is an OS intented to serve real time application that process data as it comes in, mostly without buffer delay. Here, processing time requirements are calculated in tenths of seconds increments of time. It is time-bound system that can be defined as fixed time constraints. In this type of system, processing must be done inside the specified constraints. Otherwise, the system will fail.

Type of Real time system are,

i) Hard Real-Time System

A hard real-time system is the one in which failure to meet even a single deadline may lead to complete or catastrophic system failure. Here, the handled task must be executed on specified scheduled time.

Example, Aircraft systems, Medical critical care systems, etc.

ii) Soft-real time System

A soft-real time system is one in which performance is degraded but not destroyed by failure to meet response-time constraints. So the deadlines are handled softly.

Example, Online Transaction system, livestock price quotation

system.

iii) Firm Real-time system:
A firm real-time system is one in which a few missed deadlines will not lead to total failure, but missing more than a few may lead to complete or catastrophic system failure.
E.g. Various types of multimedia applications.

As a part of truly understanding the nature of real-time systems, it is important to address a number of frequently cited misconceptions.

i) Real time systems are synonymous with "fast" systems.
ii) Rate-monotonic analysis has solved the real-time problem
iii) There are universal, widely accepted methodologies for real-time systems specification and design.
iv) There is no more a need to build a real-time operating system, because many commercial product exists.
v) The study of real-time system is mostly about scheduling theory.

Q. 2. Differentiate between Real time and non-real time system with examples.
Ans:- Real time tasks are the tasks associated with the quantitative expression of time, while non-real time tasks are the tasks which is not associated with the timing constraint. The non-real time task are not described by timing expressions.

Real Time Systems	Non-real time systems
i) Here, the task associated with time bound.	i) Task associated is not time bound.
ii) Task here can be expressed as quantitative expression of time.	ii) It can't be expressed as function of time.
iii) It can be hard, soft & firm.	iii) It is not further classified.
iv) Deadline of real-time tasks are in the order of seconds.	iv) Deadline of non-real time task can be minutes, hours or even days.
v) It includes interactive task.	v) It can include some old jobs used some decades ago.
vi) It is more common in computer system.	vi) It is less common these days.
vii) Here, OS guarantee that the task will run at a given time for a given time. It will be deterministic in nature. Eg. Airline services, Satellite tracking, etc.	vii) Non-real time OS make no such guarantees and critical task can fail under them. Eg. accounting systems, workstations, etc.

Q.3. ~~Examp~~ Explain the working mechanism of a Petri net.

Ans: Petri nets are formal methods used to specify and analyse concurrent operations in real-time systems. It can produce executable specifications and are particularly suitable for modeling synchronisations among asynchronous tasks.

A set of circles called "places" is used to represent data stores or conditions. Rectangular boxes, on the other hand, represent transitions or events. Each place (P) and transition (T) is labeled with a data count and transition function, respectively, and are connected by unidirectional arrows. The initial petri net graph is labeled with a marking given by m_0, which represents the initial data count in all the places. Subsequent markings, m_j, $i \in \{1, 2, 3, \ldots\}$, are results of the firing of transitions, where each firing is an atomic operation by its nature.

A transition fires if it has as many input data as required for producing an associated output. In petri nets, the graph topology does not change over time, only the marking (data count) of process do.

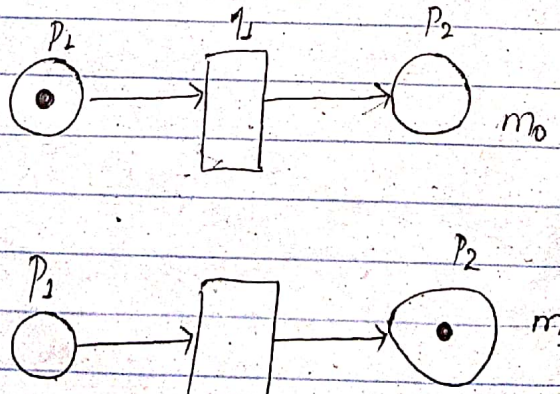

fig:- Simple petri net before (m_0) and after (m_j) firing

	P_1	P_2
m_0	1	0
m_1	0	1

firing table for given petri net

Q.5. Explain Yourdon, Ward and Miller's contribution in extending SASD for gaining requirements specification for Real time system.

Ans: Structured Analysis and structured design (SASD) have widely evolved over 30 years and are used in diverse real-time applications, throughout the world. Several extensions to the original structured analysis have emerged already, for instance, system dynamics and usage of SA for the specification of embedded systems.

Ward and Miller extended data-flow diagrams by adding a way to model control-flows (interrupts), as well as finite state machines (or state-transition diagram), for defining control processes.

Example, Teamwork and Software through Pictures.

Structured analysis for real-time system is based on fundamental notion of the flow of data between successive data transformations and provides little support for identifying concurrency. Yourdon's Modern SA uses three viewpoint to describe a system: an environmental model, a behavioral model, and an implementation model.

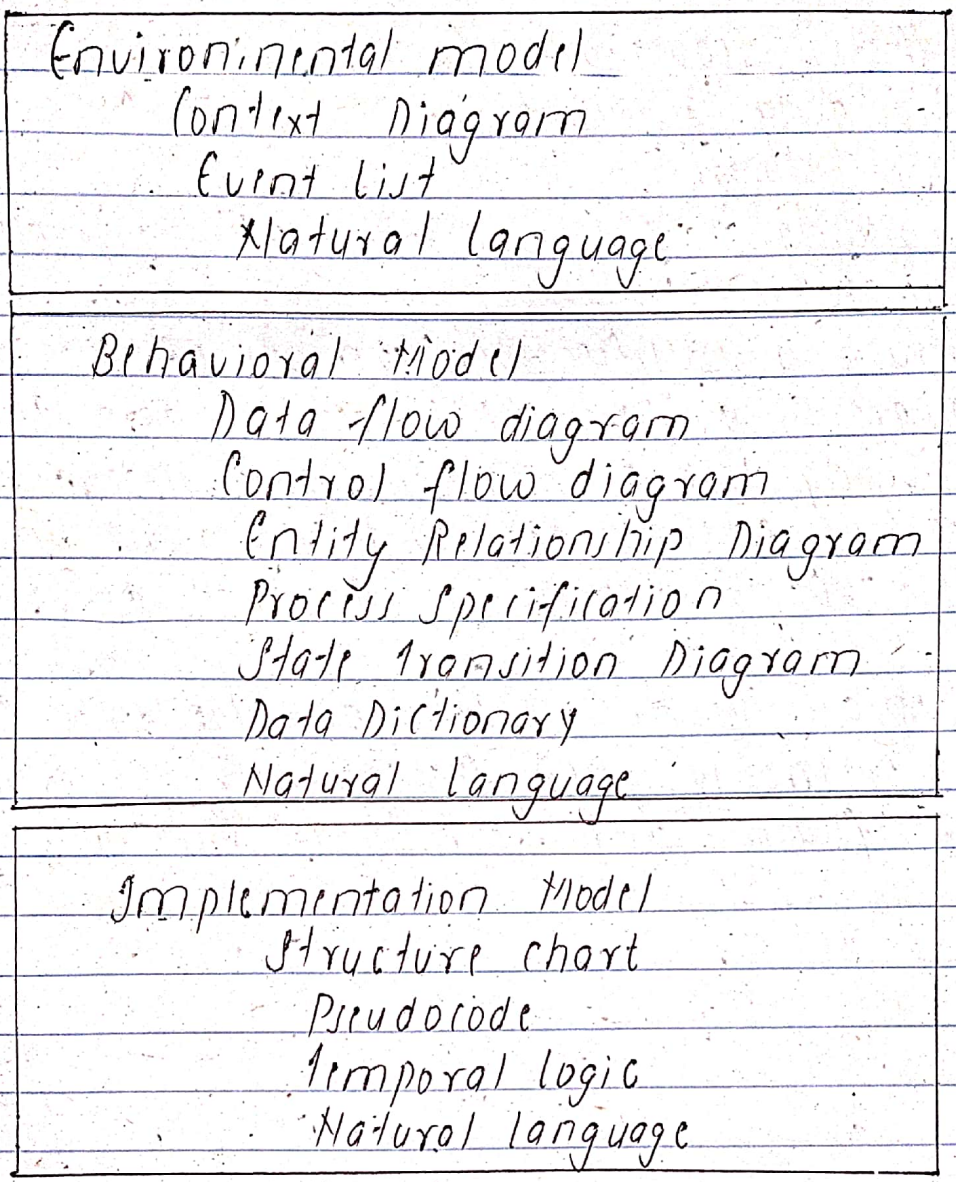

fig:- Models and elements of Structural analysis and design.

- It is highly potential way to overcome the problems of classic analysis using graphical tools and a top-down, functional decomposition method to define system requirements, and models system's context, process and content. Also, the model provides,
- Easy maintainability of the target document.
- Use of illustrative graphics.

- Use of an effective method for partitioning
- building a logical model of the system for the user before implementation.
- Reduction of ambiguity and redundancy.

The target document for SA is called the structured specification that consists of a system context diagram, an integrated set of data-flow diagrams showing the decomposition and interconnectivity of components, and an event list to represent the set of events that drive the system.

Q.6. What is the role of kernel in an Operating system? Explain different types of kernels.

Ans: Real time Operating Systems provide three essential functions with respect to software tasks,

- Scheduling
- dispatching
- intertask communication and synchronisation

The kernel of an operating system is the smallest entity that provides all these functions. Scheduler determines which task will run next in a multitasking system, while a dispatcher performs the necessary bookkeeping to start that particular task. Moreover, intertask communication and synchronisation assures that parallel task may cooperate effectively.
The layer of OS functionality and an associated taxonomy are,

Operating Systems	Users User Interface, Security features and file manggiment
Executive	Privatized memory blocks, I/O Services and Supporting Features
Kernel	Intertask communication and synchronisation
MicroKernel	Task scheduling and Dispatching

fig:- role of kernel in OS

Objectives of kernel,
- To establish communication between user level application and hardware.
- To decide state of incoming processes.
- To control disk managemint, mimory managemint and task managemint.

Types of Kernel
1) Monolithic kernel
It is type of kernel where all operating system services operate in kernel space. It has dependencies between systems components and huge lines of complex code.

ii) Micro Kernel
It is type of kernel which has minimalist approach. It has virtual memory and thread scheduling. It is more stable with less services in Kernel space, and puts rest in user space.
However, there are lots of system calls and context switches.

iii) Hybrid Kernel
It is combination of both monolithic kernel and micro kernel.

iv) Exo Kernel
It is the type of kernel which follows end-to-end principle. It has fewest hardware abstractions as possible. It allocates physical resources to applications.

v) Nano Kernel
It is the type of kernel that offers hardware abstraction but without system services.
Micro Kernel also does not have system services therefore the micro kernel and nano kernel have become analogous.

Kernel is thus central component of the operating system that manages operations of computer and hardware. It acts as bridge between applications and data processing performed at hardware level using interprocess communication and system calls.

Q.7. Explain different types of Real time programming languages with examples.

Ans: A programming language represents the nexus of design and structure. So, the actual building of any software depends on tools to compile, generate binary code, link and create binary objects. The main tool in the code generation process is the language compiler. Real time systems are currently being built with a variety of programming language including various dialects of C, C++, C#, Java, Ada, assembly language, and even Fortran or Visual basic.

Real time programming languages are chosen as per need and compiling capacity of various language. The commonly used one includes, Assembly languages, Procedural languages, and object-oriented languages.

Assembly Language

Assembly language have limited role in real-time programming. Although it lacks user-friendliness and productivity features of high-level languages, assembly language does have a particular advantage for use in real-time programming; it provides the most direct control of the computer hardware over high-level languages.

However, coding in assembly language is time consuming to learn, tedious and error prone since the language is unstructured and has very limited abstraction properties.

Assembly languages are used in special circumstances

When the compiler does not support certain machine language instructions, or when the timing constraints are so high (tight) that manual tuning is the only way to produce code that fulfills the extreme response-time requirements.

Assembly language is used in,

- used in codes with interrupts handler and device drivers
- in situations where predictable performance for code is difficult / impossible.
- for using all architectural feature of CPU (parallel adders and multipliers)
- Writing code with minimum execution time for time-critical applications.
- debugging hard problems below the level of high-level language code

Example:

Ada

- Ada was intended to be used specifically for programming real-time systems
- Three particularly useful constructs were introduced in Ada 95,
  - a pragma that controls how tasks are dispatched.
  - a pragma that controls the interaction between task scheduling.
  - a pragma that controls the queuing policy of task- and resource-entry queues.

Procedural languages

Procedural languages are those in which the action of the program is defined by a set of operations executed in sequence. These language are characterized by facilities that allow for instructions to be grouped together into procedures or module. Appropriate structuring of the procedures allows for achievement of desirable properties of the software, for example, modularity, reliability and reusability.

There are several programming-language features standing out in procedural languages that are of interest in real-time systems, particularly,

- modularity
- strong tying
- Abstract data typing
- Versatile parameter passing mechanisms
- Dynamic memory allocation facilities
- Exception handling.

Example, Ada, C, Fortran and Visual Basics

Object Oriented languages

The benefit of object-oriented languages, such as improved programmers productivity, increased software reliability, and higher potential for code reuse are well known and appreciated.

Object are an effective way to manage the increasing complexity of real-time systems, as they provide

a natural environment for information hiding, or protected variation and encapsulation.
In encapsulation, a class of objects and methods associated with them are enclosed or encapsulated in class definitions.
Object Oriented language provide a fruitful example for information hiding, for instance, in image-processing systems, it might be useful to define a class of type pixel, with attributes describing its position, color, and brightness, and an operation that can be applied to a pixel, such as add, activate, and deactivate.
Some OO languages are not sufficiently modular and require recompilation of superclasses when compiling subclasses. Hence, the time spent in compilation may grow disproportionately with size of the system. So, the language is mostly recommended for soft and firm real-time systems.
It includes programming language as Ada, C++, C# and Java, these language support data abstraction, inheritance, polymorphism and message passing.

Special Real-time language includes,

- PEARL
- Real-time Euclid
- Occam 2
- Real-Time C
- Neuron C
- Real-Time C++

Q. 8. How can requirements for a real time system be captured using SASD and OOA/D approaches? Explain by mentioning the extensions incorporated.

Ans:- SASD methods are used widely in diverse real-time applications. These techniques are closely associated with the procedural programming language with which they co-evolved and in which count-less real time system are written. Also, the SASD tools are globally available, thus are used to capture requirements for a real time system.

Structured analysis for real-time systems is based on fundamental notion of the flow of data between successive data transformations, and it provides very little support for identifying concurrency. Hence, depending upon the detail of the analysis phase, there is usually something arbitrary in identifying the appropriate set of processes.
Yourdon's Modern-Structural Analysis has three complementary models for describing a real time system,

i) Environmental model
ii) Behavioral model
iii) Implementation model

Environmental model embodies the analysis aspect of SASD and consists of a context diagram and an associated event list. The behavioral model embodies design aspect as a series of data flow and control flow diagrams, entity-relationship diagrams, process and control specifications, state transition diagrams, and a data dictionary.

However, the data-flow and control-flow analysis should be performed separately.
In implementation model, the developer uses a selection of structure charts, pseudo code, and temporal logic to describe the system to a level that can be readily translated to some procedural programming language. SA is a highly potential way to overcome the problems of classic analysis using graphical tools and a top-down, functional decomposition method to define system requirements.

Also, in structured Analysis, it is used to model a system's context i.e. where input come from and where outputs go, processes (what functions the system performs, how the functions interact, and how inputs are transformed to outputs) and context (the data the system needs to perform its functions). The target document for SA is called the structured specification. It consists of a system context diagram, a hierarchical set of data flow diagrams showing the decomposition and interconnection of various components, and an event list to represent the set of external events that drives the system.

Object Oriented approach is considered a viable alternative to the structured-analysis approach to developing software requirements specifications. OO uses structure of collaborating objects, in which each part performs its specified processing by reacting to inputs from its immediate neighbors. OOA also uses UML diagrams for graphical representation of requirements.

It uses both structural and Behavioral diagrams which include, Class diagram, Component diagram, Composite structure diagram, Deployment diagram, Object diagram, Package diagram, Activity diagram, State-machine diagram, Use-case diagram, etc.
Use cases are an essential artifacts in OOAD and are described graphically using any of several techniques. In the specification of an embedded system, this is also where overall timing constraints, sampling rates, and deadlines are often specified.
The use of OO approaches in real-time system modeling provides numerous desired characteristics,

- Distributivity and concurrency
- Effective management of complexity
- Enhanced reusability
- Excellent traceability
- Improved understandability and maintainability
- Increased extensibility
- Modularity of design.

Q.9. "Real time computing is all about fast computing." Do you agree? Give your justification.

Ans:- A real time system is a computer system that must satisfy bounded response-time constraints or risk severe consequences, including failure, also it is system whose logical correctness is based on both the correctness of the outputs and their timeliness.

Real time systems are often reactive or embedded systems. Reactive systems are those in which task scheduling is driven by ongoing interaction with their environment, for example, a fire-control system reacts to certain buttons pressed by a pilot.
Embedded system is a system containing one or more computers having a central role in the functionality of the system, but the system is not explicitly called a computer.
As a part of truly understanding the nature of the real-time systems, it is important to address a number of frequently cited misconceptions. Among which, real-time system as fast-computing system is one. It arises with the fact that many hard-real time systems indeed deal with deadlines in the ten of milliseconds, such as the aircraft navigation system. In a typical food industry application, however, pasta-sauce jars can move along the conveyor belt past a filling point at a rate of every five seconds. Furthermore, the airline reservation system could have a deadline of 15 seconds. These latter deadlines are not particularly fast, but satisfying them determine the success or failure of the system.
Thus, fast computing does not makes a system real-time, the system must have logical correctness along with faster computation.

However, the deadlines of the real time system must meet before catastrophic failure.

Q. 10. Differentiate between soft, hard and firm real-time systems.

Ans:- A soft real time system in one in which performance is degraded but not destroyed by failure to meet response-time constraints.

A hard real time system is one in which failure to meet even a single dead-line may lead to complete or catastrophic system failure.

A firm real-time system is one in which a few missed deadlines will not lead to total failure, but missing more than a few may lead to complete or catastrophic system failure.

In automatic teller machine as soft-real time system, missing even many deadlines will not lead to catastrophic failure, but leads to significant customer dissatisfaction and potential threat to loss of business.

In embedded navigation controller for an autonomous weed killer robot, as firm real-time system, missing a few navigation deadlines causes the robot to veer out from a planned path and damage healthy crops.

In Avionics weapons delivery system in which pressing a button launches an air-to-air missile as hard real time system, missing the deadline to launch the missile within a specified time after pressing the button may cause the target to be missed, which will result in catastrophe.

Q. 11. Why these are considered real time programming languages?
i) Ada ii) C# iii) Java iv) C.

Ada
- Ada was intended to be used specifically for programming real-time systems
- The thoroughly revised Ada 95 is considered the first internationally standardized object-oriented programming language, and referred as first-real time language.
- The useful constructs introduced in revised version of Ada to resolve scheduling, resource contention and synchronisation are,
  - a pragma that controls how task are dispatched.
  - a pragma that controls the interaction between task scheduling.
  - a pragma that controls the queuing policy of task- and resource-entry queues.

- The further revised Ada 2005 also supports additional dispatching policies, support for timing events, and support for control of CPU-time utilisation. It compri Object oriented model to provide multiple inheritance. Thus, is considered real time programming language.

C#
- C# is a C++ like language that, along with its operating environment, has similarities to Java and Java virtual machine, respectively.

- The minimum kernel configuration provides basic networking support, thread management, dynamic link library support, and virtual memory management.
- C# supports "unsafe code" allowing pointers to refer to specific memory locations.
- Object referenced by pointers must be explicitly "pinned" disallowing the garbage collector from altering their location in memory. The garbage collector collects pinned objects. This increase schedulability, allowing DMA to write to specific memory locations; a necessary capability in real-time embedded systems.
- C# supports an array of thread synchronisation constructs: lock, monitor, mutex and interlock.
- Timers in C# are configured how long to wait in milliseconds before their first invocations, and are also supplied an interval, again in milliseconds specifying the period between subsequent invocations.
- C# interact effectively with operating systems, also it's good floating-point performance make it a programming language that is highly potential for soft and even firm real-time applications.

Java

- Java is an object oriented language with code similar to C++. It supports call by value, and is pure object-oriented language.
- Real-time java is modification of the standard java language, and is implemented in soft, firm and even hard real-time systems.
- In addition to the unpredictable performance of garbage

collection, the java specification provides only broad guidance for scheduling.

- Different requirements for real-time specification of Java has been defined including mutual extensions, APIs (for communication and synchronisation), etc.
- Real time Java defines the real-time thread class to create threads, which help in executing resident scheduler.

C

- C is programming language that supports machine-related items like address, bits, bytes and characters.
- These entities can be used to control CPU's work registers, peripheral interface units, and other memory-mapped hardware needed in real-time systems.
- It provides special variable types (register, volatile, static and constant), which allow for effective control of code generation at the procedural language level.
- C is particularly good for embedded programming, because it provides for structure and flexibility without complex language restrictions.

Q.32. Distinguish between periodic, aperiodic and Sporadic tasks with examples.

Ans: The real-time task that is repeated after a certain time interval is periodic-real time task. They are controlled by clock interrupts. So, they are also called clock-driven task.

The dynamic tasks that occur at any random time and have soft deadlines are known as aperiodic-real time task.

The real-time task that occur at any random instant and have hard deadline are known as sporadic real-time tasks.

Periodic task	Sporadic task
i) It occurs after a certain period of time.	i) It occurs at random instant.
ii) It is controlled by clock interrupts.	ii) It is not controlled by clock interrupts.
iii) Time of occurrence of periodic task can be predicted.	iii) Time of occurance cannot be predicted.
iv) It includes moderate critical or low critical tasks.	iv) It includes highly critical tasks.
v) It includes normal system tasks.	v) It includes task that may lead to system failure.
vi) It can be easily scheduled by cyclic scheduler.	vi) It is complex to schedule sporadic task by cyclic scheduler.
vii) There are allotted time frome in generalized task scheduler.	vii) There are allotted slack time in generalized task scheduler.

Sporadic task (hard RT)	Aperiodic task (soft)
i) It has hard deadline.	i) It has soft or no deadline.
ii) highly critical	ii) low / moderate critical
iii) min. seperation between 2 consecutive instance cannot be zero.	iii) min. seperation can be zero.
iv) Deadlines can be met easily.	iv) difficult to meet deadlines
v) Executed when sufficient slack time is available.	v) execution does not depend on slack time
vi) Rejected when less slack time.	vi) Never rejected by scheduler.

Q-17. It is good idea to allow preemption a hard real time system? Discuss your view.

Ans:- A higher priority task is said to preempt a lower-priority one if it interrupts the executing lower-priority task. Different hard real time system also allow preemption as per need of the resource sharing and priority of different task.

During preemptive priority system, priority assigned to each interrupt are based on the importance and urgency of the task associated with the interrupt. For example, the nuclear power plant supervision system is best designed as a preemptive priority system, even when system is hard-real time. While appropriate handling of intruder events, for example, is critical, nothing is more important than processing the core overtemperature alert.

Priority interrupts can be either of fixed priority or dynamic priority type. Fixed priority systems are less flexible, since the task priorities cannot be changed after system initialization. Dynamic priority systems, on the other hand, allow the priority of tasks to be adjusted at runtime to meet changing real-time demands.

Preemptive priority schemes can suffer from resource hogging by higher-priority tasks. This may lead to a lack of available resources for lower-priority tasks. In such case, the lower-priority tasks are said to be facing a serious problem called starvation. The potential hogging/starvation problem must be carefully addressed when designing preemptive priority systems.

However, a special case of fixed-rate, preemptive priority, interrupt-driven systems, called rate-monotonic systems, comprises those real-time systems where the priorities are assigned so that the higher the execution rate, the higher the priority. This scheme is common in embedded applications like avionics systems, and has been studied extensively.
So, preemption is allowed in hard real time systems with proper exception.