Chapter-2

Real Time Specification & Design Techniques

classmate Page

2.0. Requirement Engineering Process:

Internal

- It is the process of establishing what services are required and constraints on the system operation and development.
- It vary widely depending on the application domain, the people involved and the organization developing the requirements.
- It encompasses the set of task that lead to an understanding of what business impact of the software will be, what the customer wants and how end-users will interact with the software.

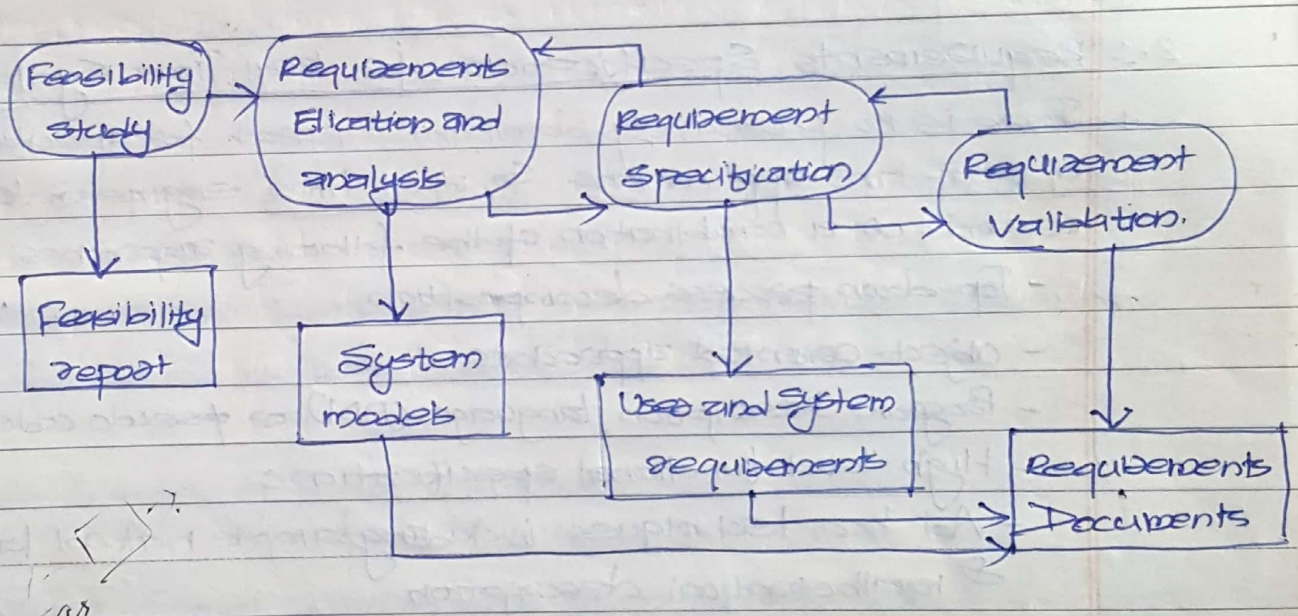


fig: Requirement Engineering Process.

SRS (System Requirement Specification)

- It is a requirement specification for a software system.
- It is a complete description of behaviour of a software system to be developed.
- It consists of ① Use case to describe all the interations the user will have with the software.
  ② Non-functional req^ms (performance, quality, ity)

* 2) Non-functional requirements such as performance, quality constraints, design constraints, reliability, availability, security, portability, maintainability } — Software system attributes.

Types of Requirements:-
1) Functional
2) Non-functional *

2.1. Requirements Specifications for Real Time Systems:

→ There is no particularly dominant approach for specification of real-time applications. So, real-time engineers tend to use one or a combination of the following approaches:-
- Top-down process decomposition
- Object-Oriented approaches
- Program Description language (PDL) or pseudo code.
- High-level functional specifications
- Ad hoc techniques including simple natural language & mathematical description

Formal Methods in Software Specification:-

→ Formal method attempts to improve requirements formulation and expression by the use and extension of existing mathematical approaches such as propositional logic, predicate calculus and set theory.

→ This practice is becoming more and more feasible with the increasing availability of supporting tools.

→ It is generally considered to be difficult to use and error-prone if used without computer based tools.

- It increases early life cycle cost and delay projects.

Uses of Formal Methods:-

1. Consistency Checking:-
- Consistency of the system can be checked by using mathematical notation.

2. Model Checking
- Whether a given property is satisfied or not under all conditions can be verified using finite state Machine or their extensions.

3. Theorem Proving:-
- A system will behave in a given way or not can be verified by using axioms of system behaviour.

Advantages of Formal Methods:-

- Writing formal requirements often lead to early discovery in the earliest phases of the software life cycle, where they can be corrected quickly and at low cost.

Disadvantages of Formal Methods:-
- cannot guarantee 100% correctness or safety.
- don't offer efficient way to reason about alternative architecture.

▷ It increases early life cycle cost and delay projects.

Uses of Formal Methods:-

1. Consistency Checking:-
▷ Consistency of the system can be checked by using mathematical notation.

2. Model Checking
▷ Whether a given property is satisfied or not under all conditions can be verified using finite state Machine or their extensions.

3. Theorem Proving:-
▷ A system will behave in a given way or not can be verified by using axioms of system behaviour.

Advantages of Formal Methods:-

▷ Writing formal requirements often lead to error discovery in the earliest phases of the software life cycle, where they can be corrected quickly and at low cost.

Disadvantages of Formal Methods:-
- cannot guarantee 100% correctness or safety.
- donot offer efficient way to reason about alternative architecture.

imp # Different Formal Methods:-

2.1.1. Z, Finite State Machines, State Charts, Petri Nets.

Finite State Machines.

- a.k.a Finite State Atomata or State Transition Diagram.
- It is the formal state based technique used in the specification and design of real-time software.
- It rely on the fact that to any systems can be represented by fixed number of states & certain transitions between them.
- The system may change its state depending on time (real-time clock) or the occurance of specific events.
- Finite State Machine can be represented by 5 tuple.

  $M = \{ S, I, T, \Sigma, \delta \}$

where,

  S is the finite, non-empty set of states
  I is the initial state ($I \in S$)
  T is the finite set of terminal states ($T \subseteq S$)
  Σ is the finite alphabet used to mark transition between states
  δ is the transition function that describes the next state of FSM given the current state.

- There are 2 types of Finite State Automata:-
  1) Moore Implementation
  2) Mealy Implementation.

1) Moore Implementation:-

→ Here, system outputs are defined in terms of state variables

2) Mealy Implementation:-

→ Here, system outputs are defined in terms of input conditions and state variables.

Example: of Moore Implementation

Let us consider a simple task to recognize the word CAT from a string of letters. The input is the next letter of the string, the output is a single bit which is 1 when CAT is recognized.

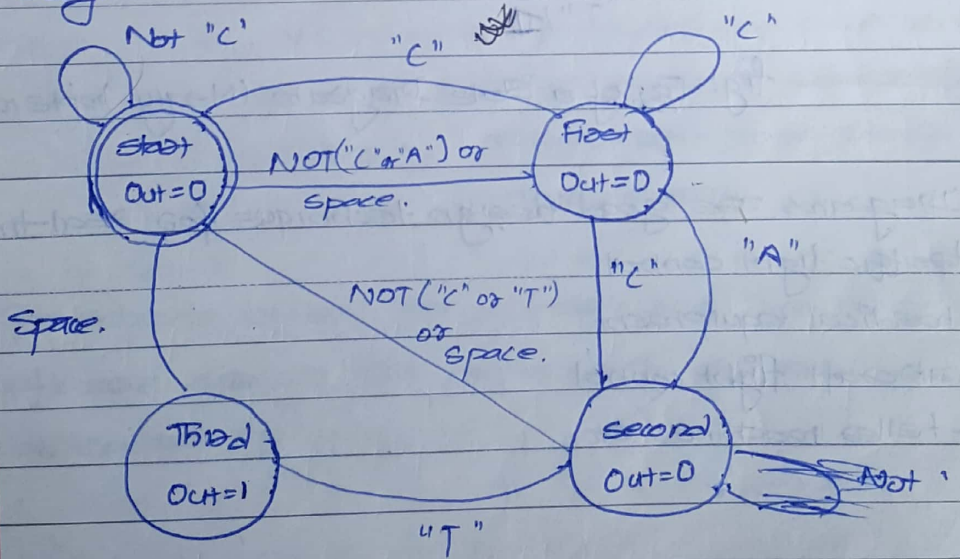


fig: Example of State Diagram (Moore Implementation)

In Mealy transition, outputs are shown on the transitions along with input causing the transition, separated by a /.

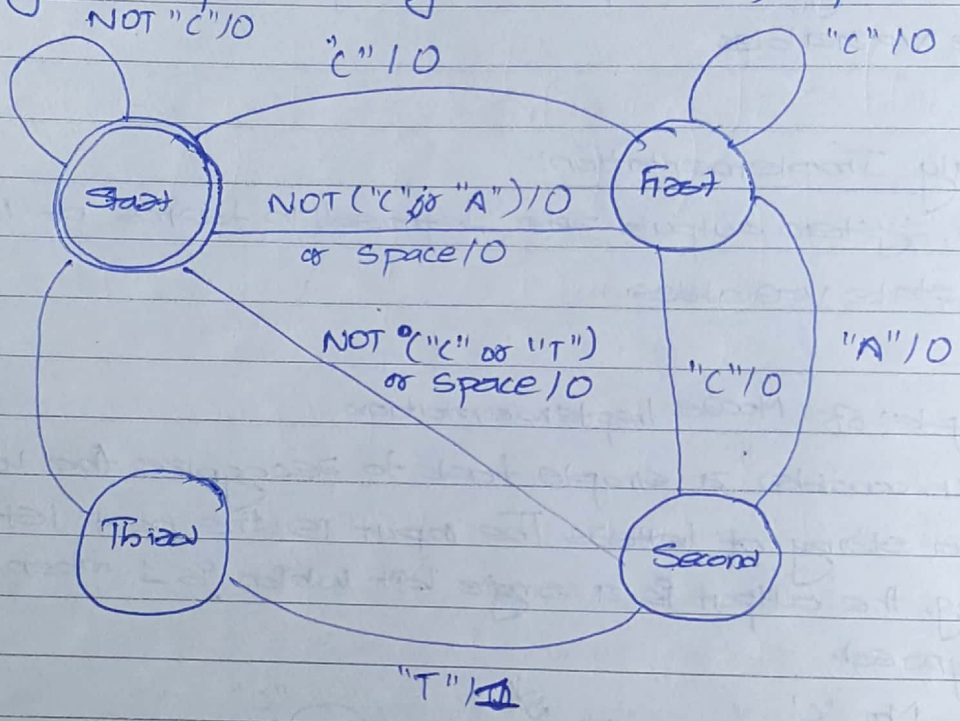


fig: Eg of a State Diagram (Mealy implementation)

- State Diagrams are good design technique for real-time systems like - traffic light control
  - medical equipment
  - aircraft flight control
  - teller machines etc.

Statecharts:-

→ Statecharts are the combination of Finite State Machines and Data Flow Diagrams

Statecharts:-

→ It is an extension of a finite state machine, where each state can contain its own FSM that further describe its behaviour.

→ State chart can be defined as:

State~~chart~~ = FSM + Depth + Orthogonality + Broadcast Communication

where,

FSM = Finite State Machine.

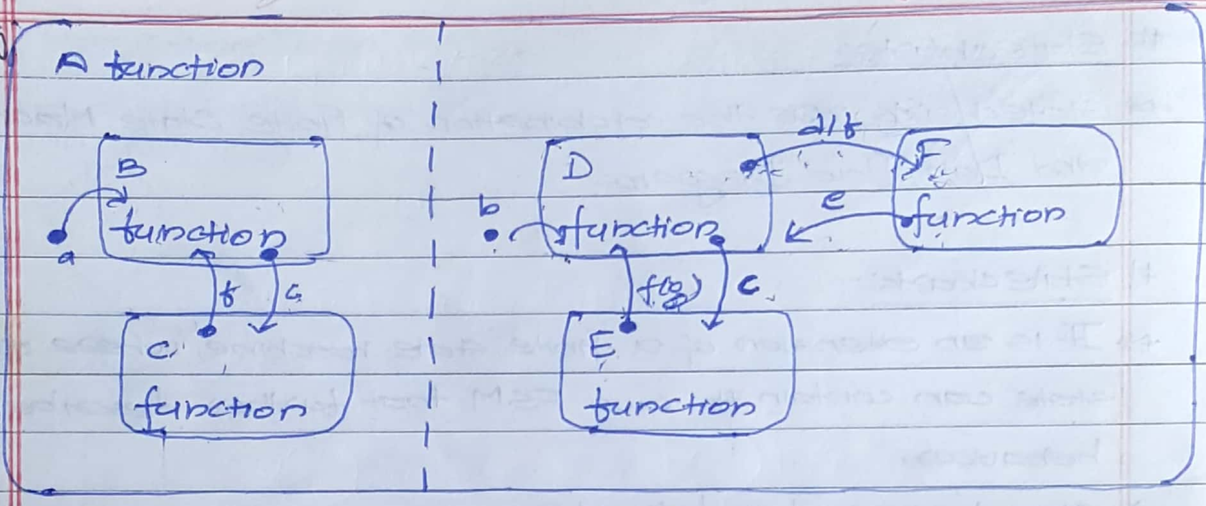
Depth represents hierarchical levels of detail

Orthogonality represents existance of parallel states

Broadcast Communication represents ~~a method for allowing~~ multiple orthogonal states that react to the same event. transitions on

→ Statecharts offer many features required for modern software design, like concurrency, modularity and inter-task communication.

Small letters a, b... z represent events that trigger transitions



Example of State Chart.

The above statechart is comprised of six states A to F with two orthogonal (concurrent) process.

- one containing states B and C
- other states D, E and F

- States B to F are said to be nested states of state A,
- The presence of the dashed line indicates two processes may be run concurrently, in this case triggered by separate events a and b.
- Each process can run independently but some transitions can be synchronized by common events called broadcast communications like c and f.

Transitions to state B is triggered by event a.
" " D " " " b
" " C & E are simultaneously " " c
Control returns to states B & D respectively, triggered by event f, with the transition to D with condition g.
Control returns to state A from state F on event E.

Explanation Note 27 & Page 15.

P.M.B

Imp # Petri Nets :-

- Petri nets are both a graphical and mathematical modeling tools.
- It is capable of modeling concurrency, nondeterminism, and asynchronous systems.
- It is used to specify and analyze concurrent operations in real-time systems
- It consists of 2 entities
  - i) places
  - ii) transition
  - ~~iii) tokens~~.

i) places: (P)

- It is a set of circles used to represent data stores and conditions
- A place may have zero or more tokens denoted by dot (•).

ii) Transitions (T)

- It is a set of rectangular boxes to represent transitions or events.

- An arc exists only from a place to a transition or a transition to a place and is denoted by arrow (→).

- A transition is enabled when no of tokens in each of its input places is at least equal to the arc weight going from place to the transition.
- An enabled transition may fire at any time.
- When fired, the tokens in the input places are moved to output places according to arc weights & place capacity.

	Input place (P_1)	Output place (P_2)
fig (1)	1	0
fig (2)	0	1

Table: firing Table.

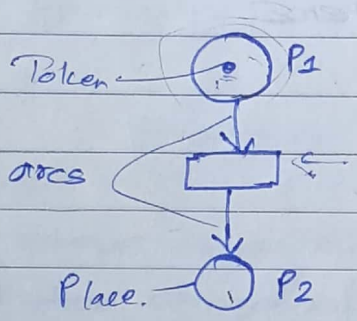


fig (1)

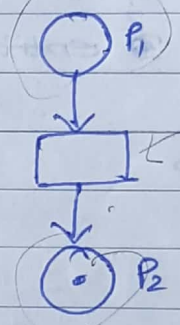


fig (2).

▷ ~~The primitive sto~~ In RTS, Petri nets are used to detect race condition, Deadlock, ~~loop~~ ~~poo~~ & conflict condition.

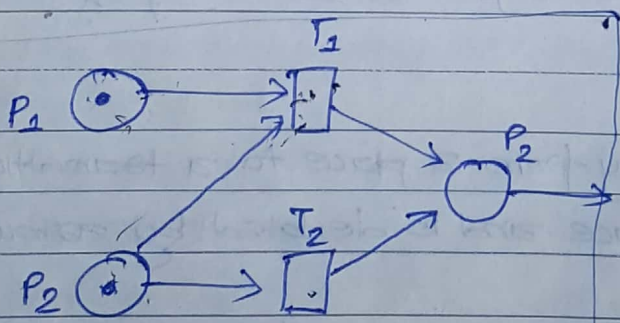


fig: Race condition

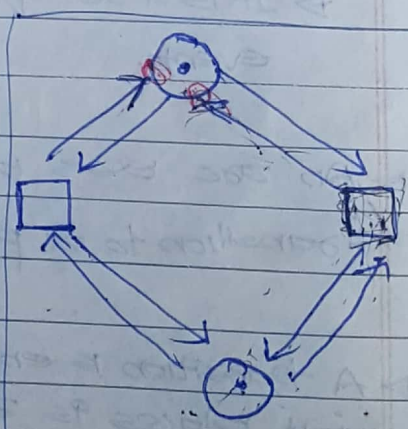


fig: Deadlock condition.

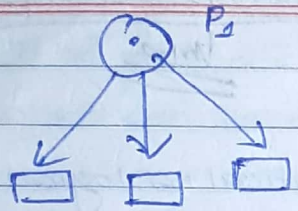


fig: Conflict Condition.

Z (pronounced zed)

- It is a formal specification language where the final specification in Z is reached by a refinement process starting from the most abstract aspects of systems.
- In z, there is a mechanism for system decomposition known as the Schema calculus. Using this calculus, the system specification is decomposed in smaller pieces called schemas where both static and dynamic aspects of system behaviour are described.
- Therefore, several extensions of time management have emerged.
- The other extension of Z which is used to accomodate the object-oriented approach provides modularity and specification reuse.
- These extensions provide information hiding, inheritance, polymorphism and similar other services like
  - Designing algorithms
  - Designing error process handling.
  - Designing test software.
  - Creating Documentation.
  - Conducting internal reviews.
  - Specifying physical location of components and data.

12

FSM
State Chart
Petrinets, Z

3 General Classification of Specification Techniques

Imp

1) Formal

strict

- This method have a rigorous mathematical or logical basis
- Example: State chart, finite State Machine, Z etc.

2) Informal

- In this method, the requirement specifications can't be completely translated into a rigorous mathematical notation. for eg; flowchart,

3) Semiformal:-

- ~~In~~ this method, ~~requirement~~ of requirement specification. ~~data~~ classification is either formal or informal defines.
  eg:- UML Statechart is formal and other modeling techines have a pseudomathematical basis.

V. Imp

Warnier-Orr Notation:-

- It is a semi-descriptive, semi-structural system of notation which can represent program structure, data and their conditional relationships.
- It uses set-theoretic notation to carefully indicate the conditions under which various processes are executed.
- It's notation is written top-to-bottom in order of execution and formed from a combination of other sets and steps.
- Each set can be made of a combination of other sets & steps.
- Each step can include a conditional decision which indicates alternative sets for execution.
- As the notation is written left-to-right increasing levels of detail are introduced.
- Logical operators of exclusive or (Ex-or) and or (+) can used to indicate mutually exclusive cases and combinational cases respectively

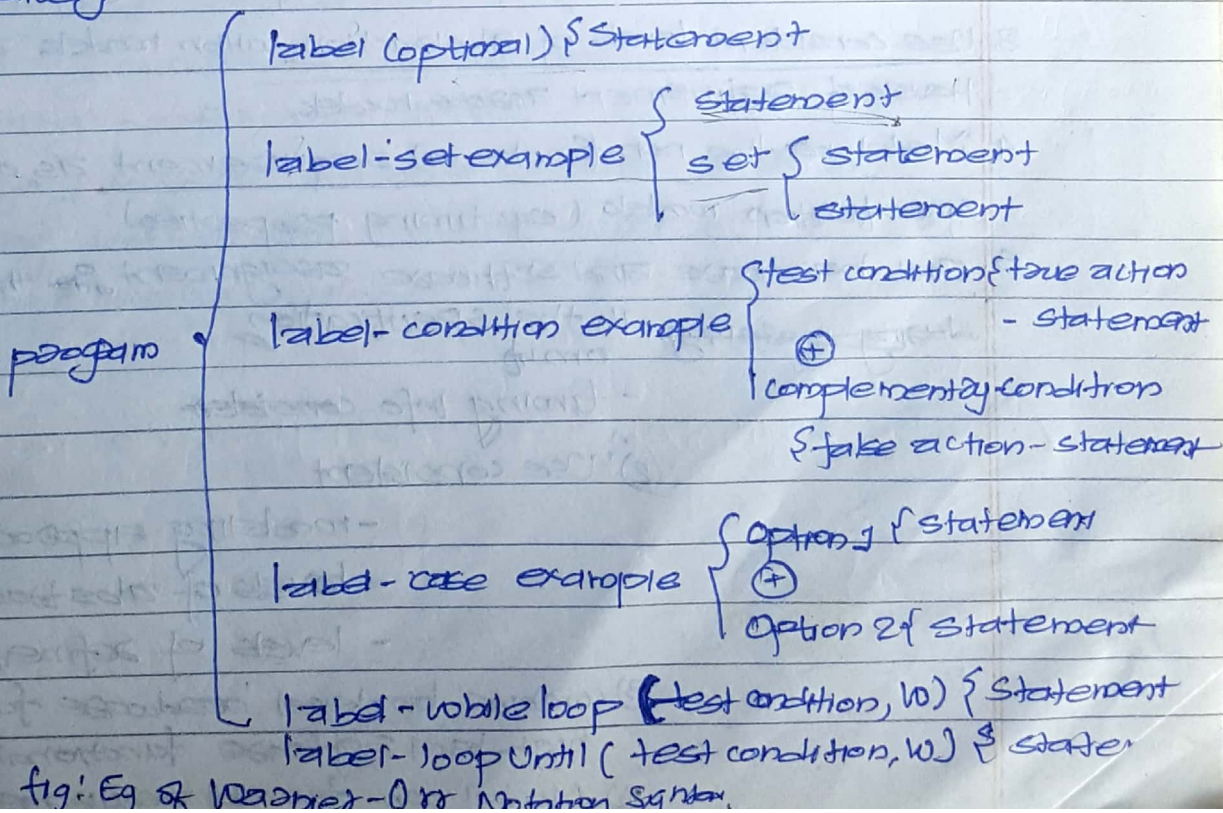


fig: Eg of Warnier-Orr Notation System

classmate
Date
Page

→ It provides the features of clear sequencing, looping, counters, modularity and decision capability.

2.2 Recommendations on Specification Approch for Real-Time Systems:-

Challanges in specifying real-time system

1. Mixing of operational and descriptive specifications.
2. Combining low-level hardware functionality and high-level systems and software functionality in the same functional level.
3. Omission of Timing Information.

→ Real-time system modeling should incorporate the following best practices

1. Use consistent modeling approaches and techniques throughout the specification. (top-down decomp or obj. oriented).
2. Separate operational specification from descriptive behaviour.
3. Use consistent levels of abstraction within models and consistent levels of refinement across models
4. Model the non-functional requirement as a part of the specification models (eg: timing properties)
5. Omit hardware and software assignment ~~in the specification during modeling~~ in the specification
   - omit
   - timing info ~~consider~~
   2) Use consistent
      - modeling approach for specification
      - levels of abstraction
      - levels of refinement
      } across models
   3) combine low-level hardware functionality & high-level software functionality in same functio
   4) Model non-functional requirement as part of

2.3 Real Time System Design Activity:-

- The design activity involves identifying the components of the software design and their interfaces from the SRS.
- The principal artifact of this activity is Software Design Description (SDD).

- During the RTS design, following tasks involves.

1. Performing hardware/software trade off analysis.
2. Designing interfaces to external components
3. Designing interfaces between component
4. Designing control strategies
5. Designing database structures
6. Designing the start-up and shutdown processing
7. Designing algorithms.
8. Designing error processing & message handling
9. Designing test software.
10. Determining concurrency of execution.
11. Conducting performance analysis
12. Specifying physical location of components & data.
13. Creating documentation for the system.
14. Conducting internal reviews.

Java,
specification model.

To perform the design activities from SRS, we have two methodologies:-

- Procedural-oriented design
- Object-Oriented Design.

Problems with Structured Analysis & structured Design in Real Time Applications

* There are several problems in using structured analysis and structured design to model the RTS. Some of them are:
  - It is data oriented but not control oriented approach.
  - difficulty in modeling time and events.
  - concurrency is not easily depicted (एकाई; युगपत्)
  - Control flows are not easily translated directly into code. because they are hardware dependent.
  - Control flow doesn't really make sense since there is no connectivity between portions of it.

Real Time Extensions of structured Analysis & Structured Design:-

* The SASD methodology is not well equipped for dealing with time, as it is data oriented and not a control-oriented approach.
* So, to make SASD, more control-oriented; new control flows and control stores are added to ~~Control Flow Diagram~~.
* The addition of these control flows & stores allows. ~~the~~ ~~for~~ the creation of diagram containing only these elements called a Control Flow Diagram (CFD) & P-SPECs. The
* These CFDs can be decomposed into C-SPECs (Control Specifications), ~~which then~~ can then be described by a finite state machine.

The relationship between the control and process models is shown as

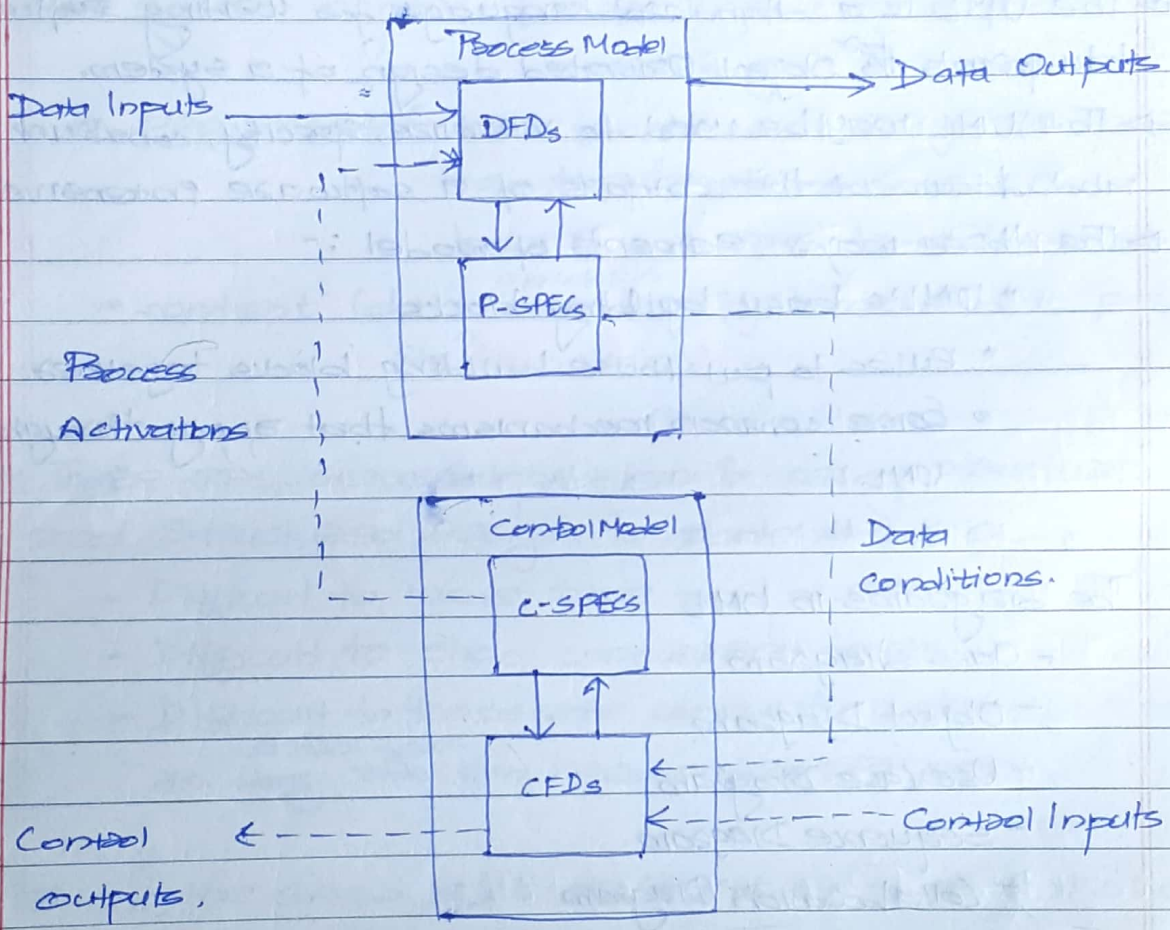


fig: The relationship between the control and process model.

CSPECs.

Real-time Extensions of Unified Modeling language:

- Object oriented programming languages are those which have the features like data abstraction, inheritance, polymorphism and message passing.
- The UML is a standard language for writing software blueprints for Object Oriented design of a system.
- The UML may be used to visualize, specify, construct and document the artifacts of a software intensive system
- The three major elements of model :-
  - UML's basic building blocks
  - Rules to put those building blocks together.
  - Some common mechanisms that apply throughout the UML.

The diagrams in UML
- Class diagram
- Object Diagram
- Use case Diagram
- Sequence Diagram
- Collaboration Diagram
- Activity Diagram
- Component Diagram
- Deployment Diagram.

Real-Time Extensions of SASD:-

- Structured Analysis is used to model ~~a system~~ consists
  - system context (where input come from and where outputs go).
  - processes (what functions the system performs, how the functions interact, how inputs are transformed to outputs
  - content (data ~~that~~ on which the system need to perform its functions).

- There are several problems in using structured Analysis and Structured Design to model the RTS.
  - Difficult to model time and event
  - Difficult to show concurrent task.
  - Difficult to translate control flow directly into code as ~~they~~ control flows are ~~can~~ hardware dependent.

Since, the SASD is data oriented not control oriented, ~~for~~ the ~~use~~ to model RTS, SASD should have Real-time Extensions. To make Real-time Extensions of SASD, control flows and control states are added in SASD.

The ~~elements~~ models of SASD are:

1. Environmental Model:-
   - It consists of analysis aspect of SASD.
   - It model the system at high level of abstraction
   - It consist of - Context diagram
     - Event list.

2. Behavioural Model:-

- It consists of design aspect of SASD ~~as a series of data flow and flow~~
- It consists of:
  i) Data Flow Diagram
  ii) Control Flow Diagram
  iii) Entity Relationship Diagram
  iv) Process Specification
  v) State Transition Diagram
  vi) Data Dictionary.

3) Implementation Model,

- It describe the system to a level that can be readily translated to code.
- It includes:-
  - structure charts
  - natural language
  - pseudocode.

~~# Real time Extensions of UML:-~~

- UML is.

fig.