SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Information Technology

Course Name – Software Engineering

II Year / III Semester

DESIGN CONCEPTS AND PRINCIPLES

Real Time Software Design

Designing Real time software involves several steps. The basic steps are listed below:

- Software Architecture Definition
- Co-Design
- Defining Software Subsystems
- Feature Design
- Task Design

Software Architecture Definition

- This is the first stage of Real time Software design. Here the software team understands the system that is being designed.
- The team also reviews at the proposed hardware architecture and develops a very basic software architecture.
- This architecture definition will be further refined in Co-Design.
- Use Cases are also used in this stage to analyze the system.
- Use cases are used to understand the interactions between the system and its users.
- For example, use cases for a telephone exchange would specify the interactions between the telephone exchange, its subscribers and the operators which maintain the exchange.

Co-Design

- The software functionality should be partitioned in such a fashion that processors and links in the system do not get overloaded when the system is operating at peak capacity. This involves simulating the system with the proposed software and hardware architecture.
- The system should be designed for future growth by considering a scalable architecture, i.e. system capacity can be increased by adding new hardware modules. The system will not scale very well if some hardware or software module becomes a bottleneck in increasing system capacity.
- Software modules that interact very closely with each other should be placed on the same processor, this will reduce delays in the system. Higher system performance can be achieved by this approach as inter-processor message communication taxes the CPU as well as link resources.

Defining Software Subsystems

Determine all the features that the system needs to support.

1. Group the various features based on the type of work they perform. Identify various sub-systems by assigning one subsystem for one type of features
2. Identify the tasks that will implement the software features. Clearly define the role of each task in its subsystem.
3. Within each subsystem, classify and group the features appropriately and associate the various tasks constituting the subsystem. For example, the Call Handling subsystem in a switching system would support features like:

- V5.2 Originating to ISUP Outgoing Call
- V5.2 Originating to V5.2 Terminating Call
- Conference Call
- Toll free call

Feature Design

1. Specify the message interactions between different tasks in the system
2. Identify the tasks that would be controlling the feature. The controlling tasks would be keeping track of progress of feature. Generally this is achieved by running timers.
3. The message interfaces are defined in detail. All the fields and their possible values are identified.

Task Design

Designing a task requires that all the interfaces that the task needs to support should be very well defined. Make sure all the message parameters and timer values have been finalized.

Selecting the Task Type

Once the external interfaces are frozen, select the type of task/tasks that would be most appropriate to handle the interfaces:

Single State Machine:

The tasks functionality can be implemented in a single state machine.

Multiple State Machines:

The task manages multiple state machines. Such tasks would typically include a dispatcher to distribute the received messages to an appropriate state machine. Such tasks would create and delete state machine objects as and when required.

Multiple Tasks:

This type of tasks are similar to the multiple state machine tasks discussed above. The main difference is that the task now manages multiple tasks. Each of the managed tasks implements a single state machine. The manager task is also responsible for creating and deleting the single state machine tasks.

Complex Task:

This type of task would be required in really complex scenarios. Here the manager task manages other tasks which might be managing multiple state machines.