Unit-2
Software Requirement

Chamundeswari Arumugam
Professor
SSN College of Engineering, Chennai

February 2018

Outline

- Structured analysis
- Flow Oriented Modeling
  - Data flow modeling
  - Control flow modeling
- Behavioral Model
- Petrinets

Structured Analysis

Why do we need requirement model or structured analysis ?

- First technical representation of the system
- Models help Us to see things that might be missing.
- Depict system specification with a combination of diagrams and structure
- Models create a visual representation of whats expected.
- Requirements models allow us to organize our data in multiple ways.

Models developed in structured analysis

- Data model eg. ER diagram, DFD.
- Function model eg. Use case diagram
- Behavior model eg. STD

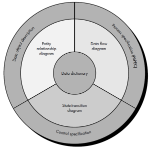
Data dictionary

- A repository that contains descriptions of all data objects consumed or produced by the software.

Requirement model

- **ER diagram** - depicts relationship between data objects. Data object description contain the description of each data.
- **DFD** - represents how data are transformed as they move through the system. Depicts the functions(and sub functions) that transform the data flow. Process specification contain the description of each function.
- **STD** - basis for system behavior model. Represents the behavior (called state) of the system and transistions(actions) made from one state to another state. Control specification description about the control aspects of the software.

Fig : structure of the analysis model



Flow oriented modeling

Creating a Data Flow Model

- DFD is widely used in requirement analysis.
- It takes input, process output view of the system.
- Data objects flow into the software are transfered by processing elements and resultant data objects flow out of the software.
- Notation : Arrow represent data object, circle represent transformation.
- Level 0 DFD or context diagram - system as a whole.
- Level 1 and level 2 DFD is the refinement of the system.

Guidelines of DFD

- Level 0 data flow diagram depicts software system
- Processing input and output should be carefully noted.
- Refinement should begin at next level.
- Arrows and bubbles should be labeled with meaningful names.
- Information flow continuity must be maintained from level to level.

Sample DFD

- Sample DFD diagram for SafeHome software.

Fig : Level 0 DFD /Context diagram

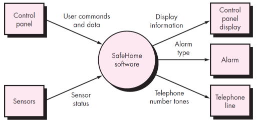


Fig:Level 1 DFD-SafeHome security function

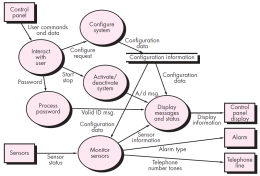
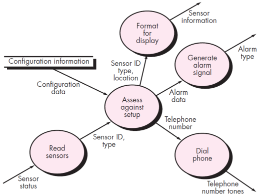


Fig:Level 1 DFD-SafeHome security function



Flow oriented modeling

Creating a Control Flow Model

- Control flow can be used by applications that are driven by events rather than data and process information with heavy concern for time and performance.
- Such applications require the use of control flow modeling in addition to data flow modeling.

CFM-Control Specification

- A control specification (CSPEC) represents the behavior of the system. A state diagram indicates how the system responds to events.
- Fig A represents the state diagram and the transitions from one state to another.Eg.Operating system
- CSPEC doesn't provide information about the inner working of the processes that are activated as a result of the behavior.

CFM-Control Specification

- The process specification (PSPEC) is used to describe all flow model processes that appear at the final level of refinement
- The content of the process specification can include narrative text, a program design language (PDL) description of the process algorithm, mathematical equations, tables, or UML activity diagrams. Fig B represents the activity diagram.

Fig A : OS : state diagram

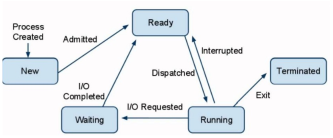
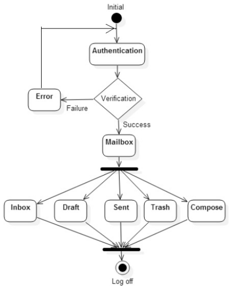


Fig:B Activity diagram for Email



Behavior Modeling

Behavior modeling

- Transition to dynamic behavior of the system or product
- Behavior of the system as a function of specific events and time.
- It indicates how software will respond to external events or stimuli.

Steps to create dynamic behavior model

- Evaluate all use cases to fully understand the sequence of interaction within the system.
- Identify events that drive the interaction sequence and understand how these events relate to specific objects.
- Create a sequence for each use case.
- Build a state diagram for the system.
- Review the behavioral model to verify accuracy and consistency.

Identifying Events with the Use Case

- Two different characterizations of states must be considered: (1) the state of each class as the system performs its function (2) the state of the system as observed from the outside as the system performs its function.

Fig : State diagram

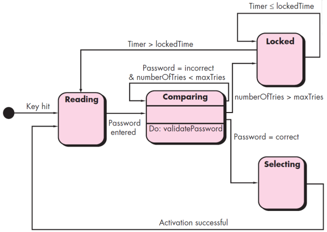
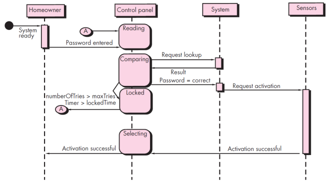


Fig: Sequence diagram

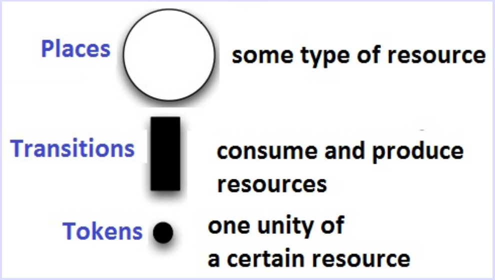


Petrinet

Introduction

- Petri nets are a tool to model concurrent systems.
- Petri net is primarily used for studying the dynamic concurrent behavior of network-based systems where there is a discrete flow.
- First introduced by Carl Adam Petri in 1962.
- Based on strong mathematical foundation.

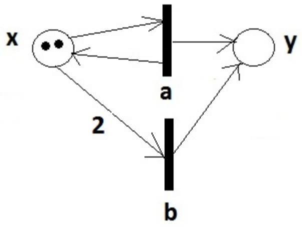
Notations


Places
some type of resource
Transitions
consume and produce resources
Tokens
one unity of a certain resource

Mathematical equation

- P={x,y} , T={a,b}
- I(a)={x}, I(b)={x,x}
- O(a)={x,y}, O(b)={y}
- μ ={x,x}

Fig: Example - Petrinet



References I

[1] RogerS. Pressman.
"Software Engineering a Practitiner's Approach"".
Seventh Edition, McGraw Hill Higher Education, 2010.