Function-Oriented Software Design (continued): Lecture 6

Dr. R. Mall

Organization of this Lecture

- Brief review of previous lectures
- A larger example of Structured Analysis
- Structured Design
  - A major objective of this lecture is that you should be able to develop structured design from any DFD model.
- Examples
- Summary

Review of Last Lecture

- Last lecture we started discussion on **Structured Analysis/ Structured Design (SA/SD) technique:**
  - incorporates features from some important design methodologies.
- SA/SD consists of two important parts:
  - structured analysis
  - structured design.

Review of Last Lecture

- The goal of structured analysis:
  - perform functional decomposition.
  - represent using Data Flow Diagrams (DFDs).
- DFDs are a hierarchical model:
  - We examined why any hierarchical model is easy to understand
  - number 7 is called the magic number.

Review of Last Lecture

- During structured analysis:
  - Functional decomposition takes place
  - in addition, data decomposition takes place.
- At the most abstract level:
  - context diagram
  - refined to more detailed levels.
- We discussed two small examples:
  - RMS calculating software
  - tic-tac-toe computer game software

Review of Last Lecture

- Several CASE tools are available
  - help in design activities:
  - help maintain the data dictionary,
  - check whether DFDs are balanced, etc.
- DFD model:
  - difficult to implement using a programming language:
  - needs to be transformed to structured design.

Observation

- From the examples,
  - observe that DFDs help create:
    - data model
    - function model

Observation

- As a DFD is refined into greater levels of detail:
  - the analyst performs an implicit functional decomposition.
  - At the same time, refinements of data takes place.

Structured Design

- The aim of structured design
  - transform the results of structured analysis (i.e., a DFD representation) into a structure chart.
- A structure chart represents the software architecture:
  - various modules making up the system,
  - module dependency (i.e. which module calls which other modules),
  - parameters passed among different modules.

Structure Chart

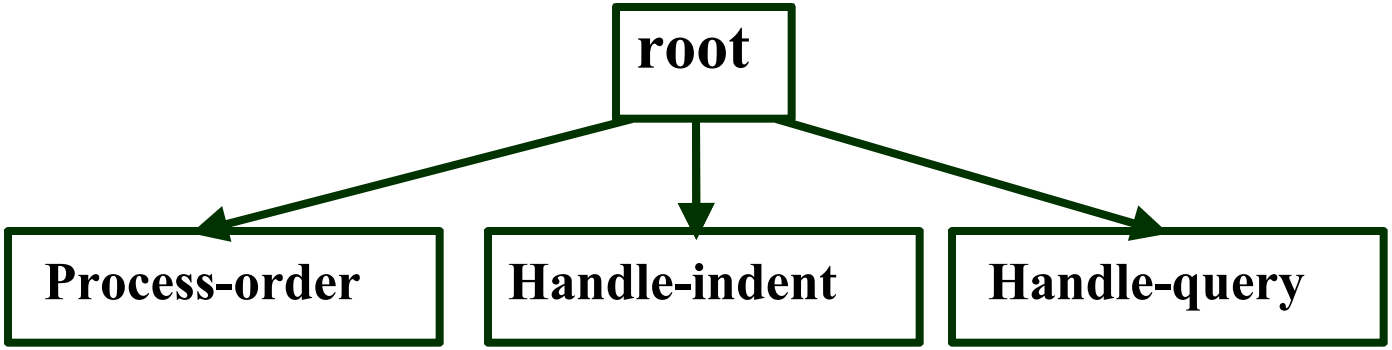
- Structure chart representation
  - easily implementable using programming languages.
- Main focus of a structure chart:
  - define the module structure of a software,
  - interaction among different modules,
  - procedural aspects (e.g, how a particular functionality is achieved) are not represented.

Basic building blocks of structure chart

- Rectangular box:
  - A rectangular box represents a module.
  - annotated with the name of the module it represents.

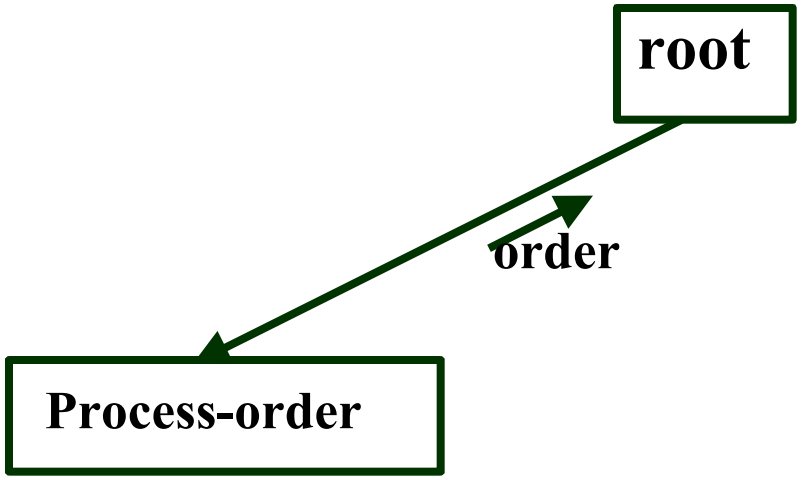
Arrows

- An arrow between two modules implies:
  - during execution control is passed from one module to the other in the direction of the arrow.



Data flow Arrows

- Data flow arrows represent:
  - data passing from one module to another in the direction of the arrow.

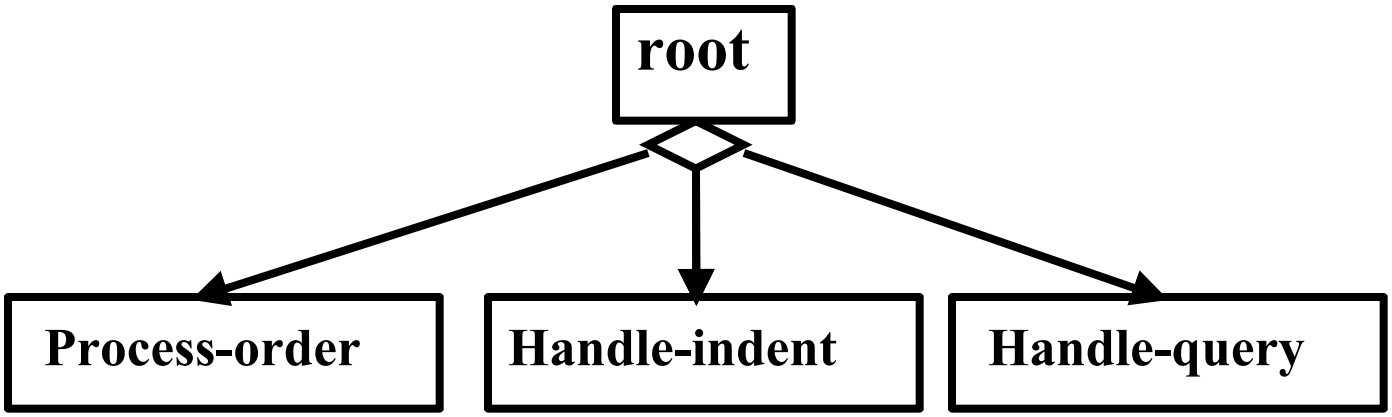


Library modules

- Library modules represent frequently called modules:
  - a rectangle with double side edges.
  - Simplifies drawing when a module is called by several modules.

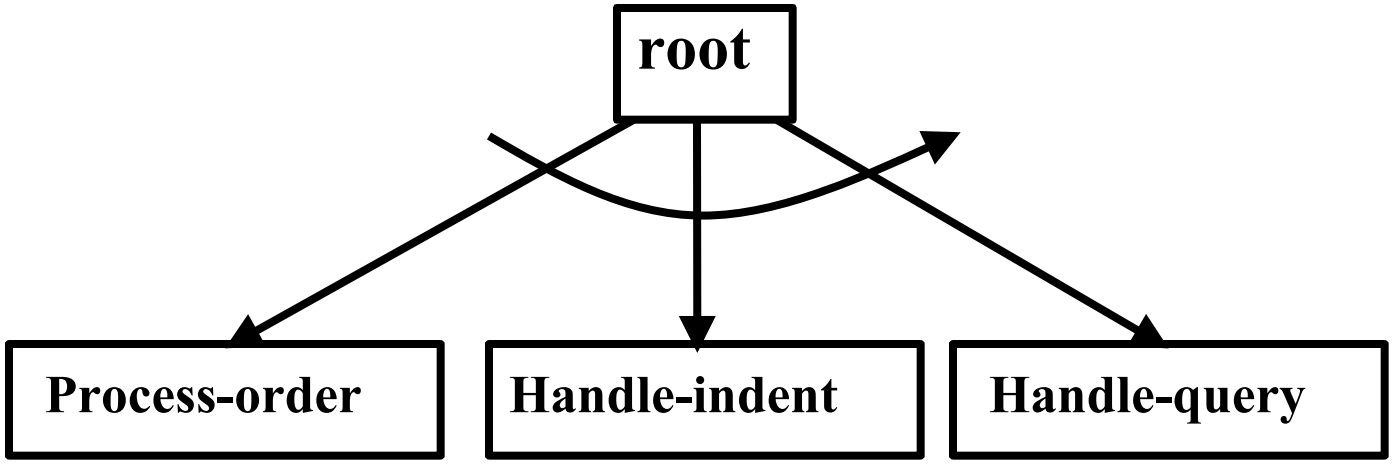
Selection

- The diamond symbol represents:
  - one module of several modules connected to the diamond symbol is invoked depending on some condition.



Repetition

- A loop around control flow arrows denotes that the concerned modules are invoked repeatedly.



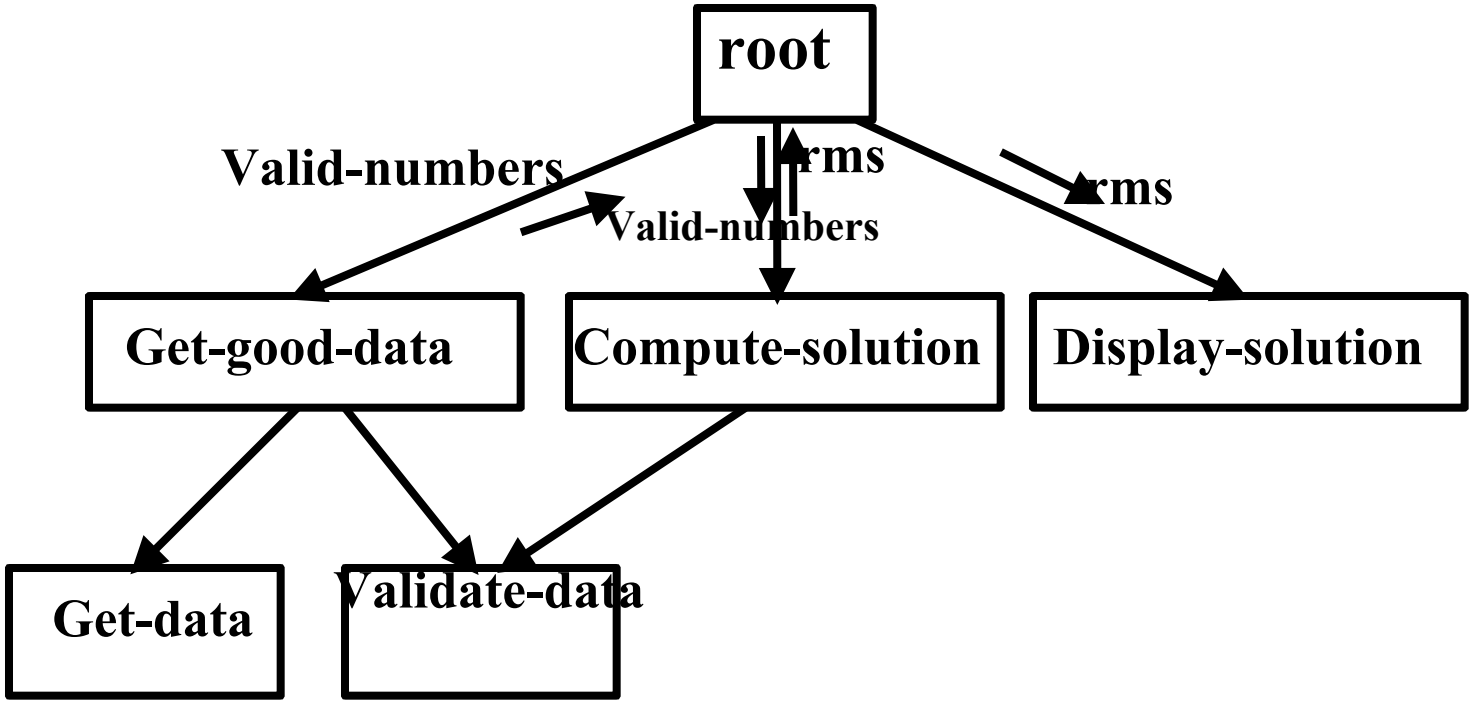
Structure Chart

- There is only one module at the top:
  - the root module.
- There is at most one control relationship between any two modules:
  - if module A invokes module B,
  - module B cannot invoke module A.
- The main reason behind this restriction:
  - consider modules in a structure chart to be arranged in layers or levels.

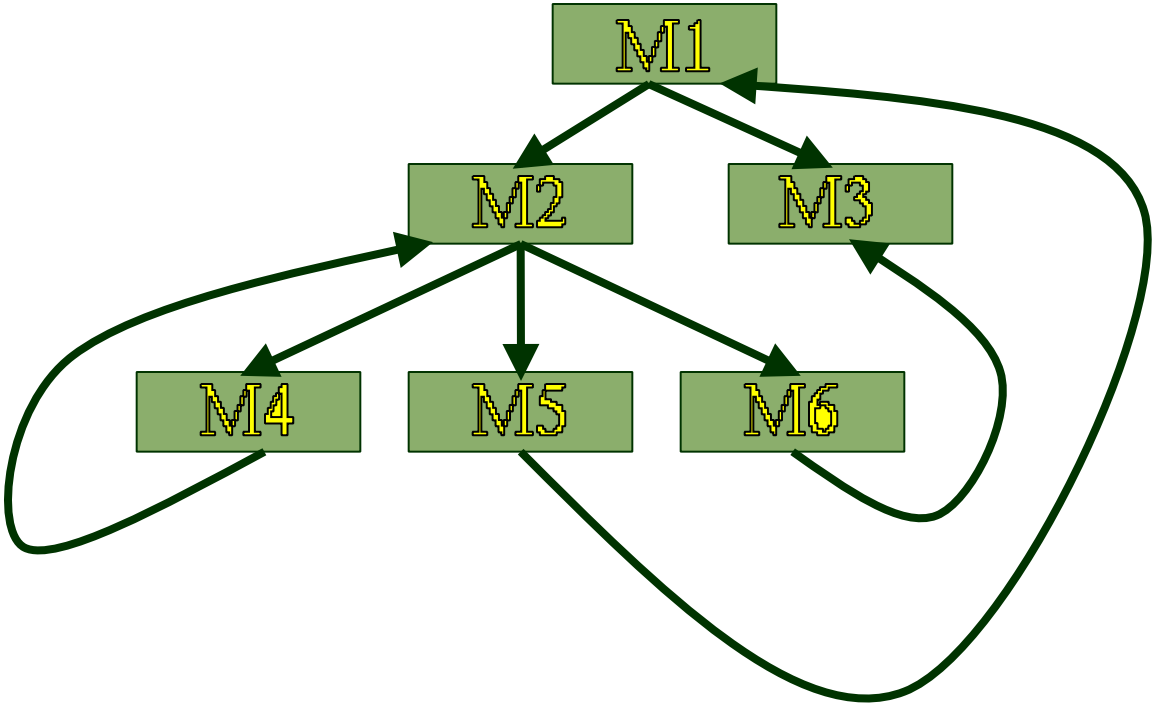
Structure Chart

- The principle of abstraction:
  - does not allow lower-level modules to invoke higher-level modules:
  - But, two higher-level modules can invoke the same lower-level module.

Example


root
Valid-numbers
rms
Valid-numbers
rms
Get-good-data
Compute-solution
Display-solution
Get-data
Validate-data

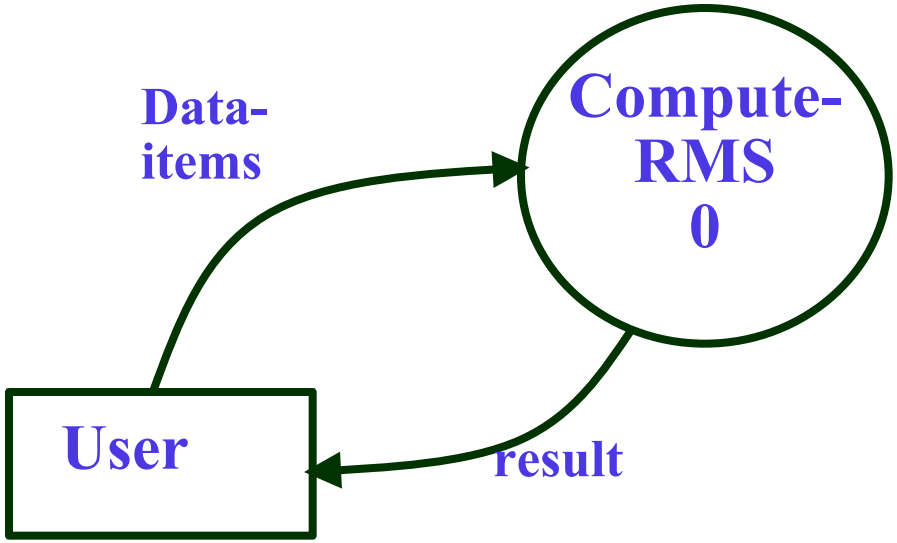
Bad Design



Shortcomings of Structure Chart

- By looking at a structure chart:
  - we can not say whether a module calls another module just once or many times.
- Also, by looking at a structure chart:
  - we can not tell the order in which the different modules are invoked.

Example 1: RMS Calculating Software

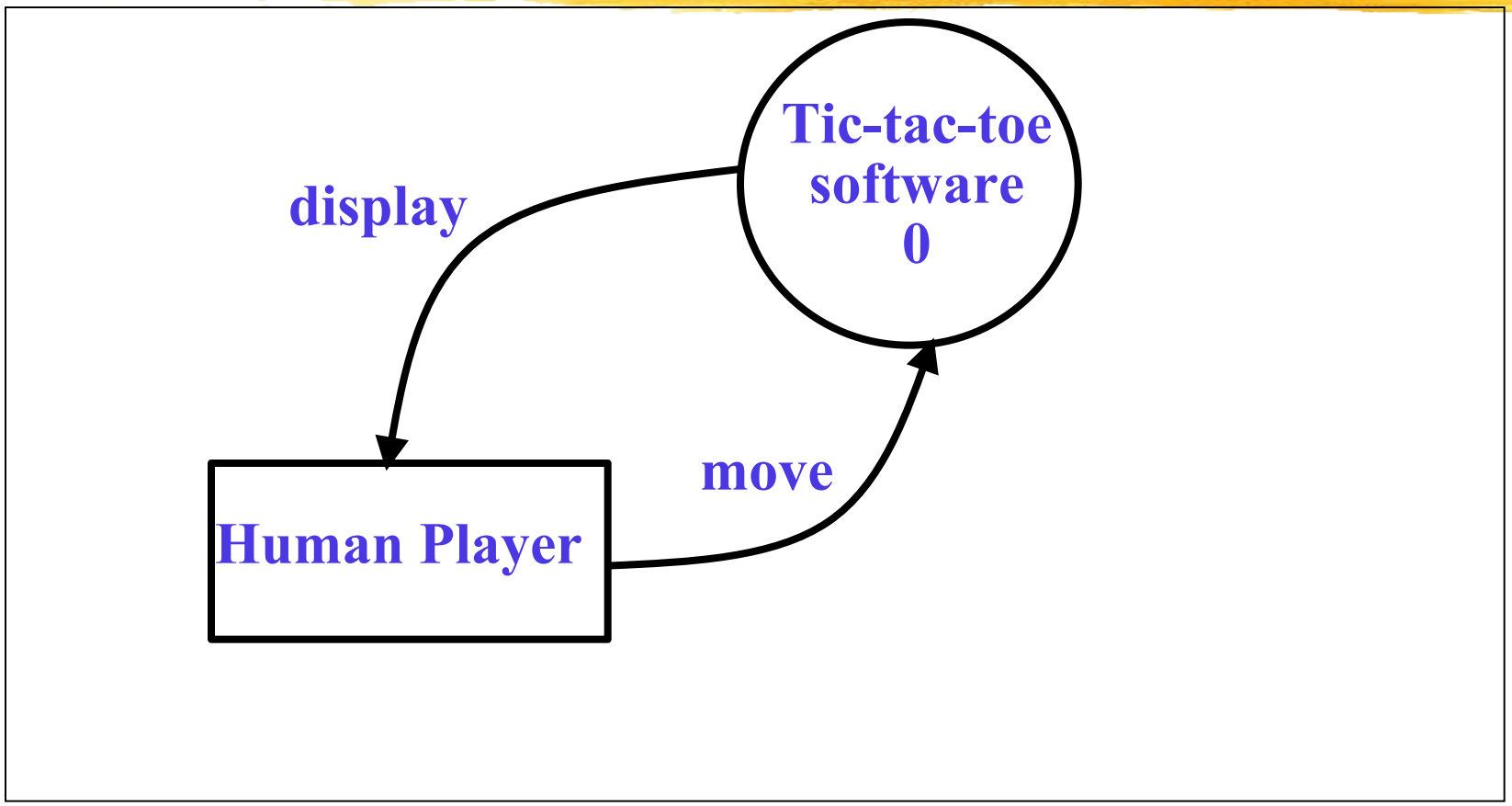


Context Diagram

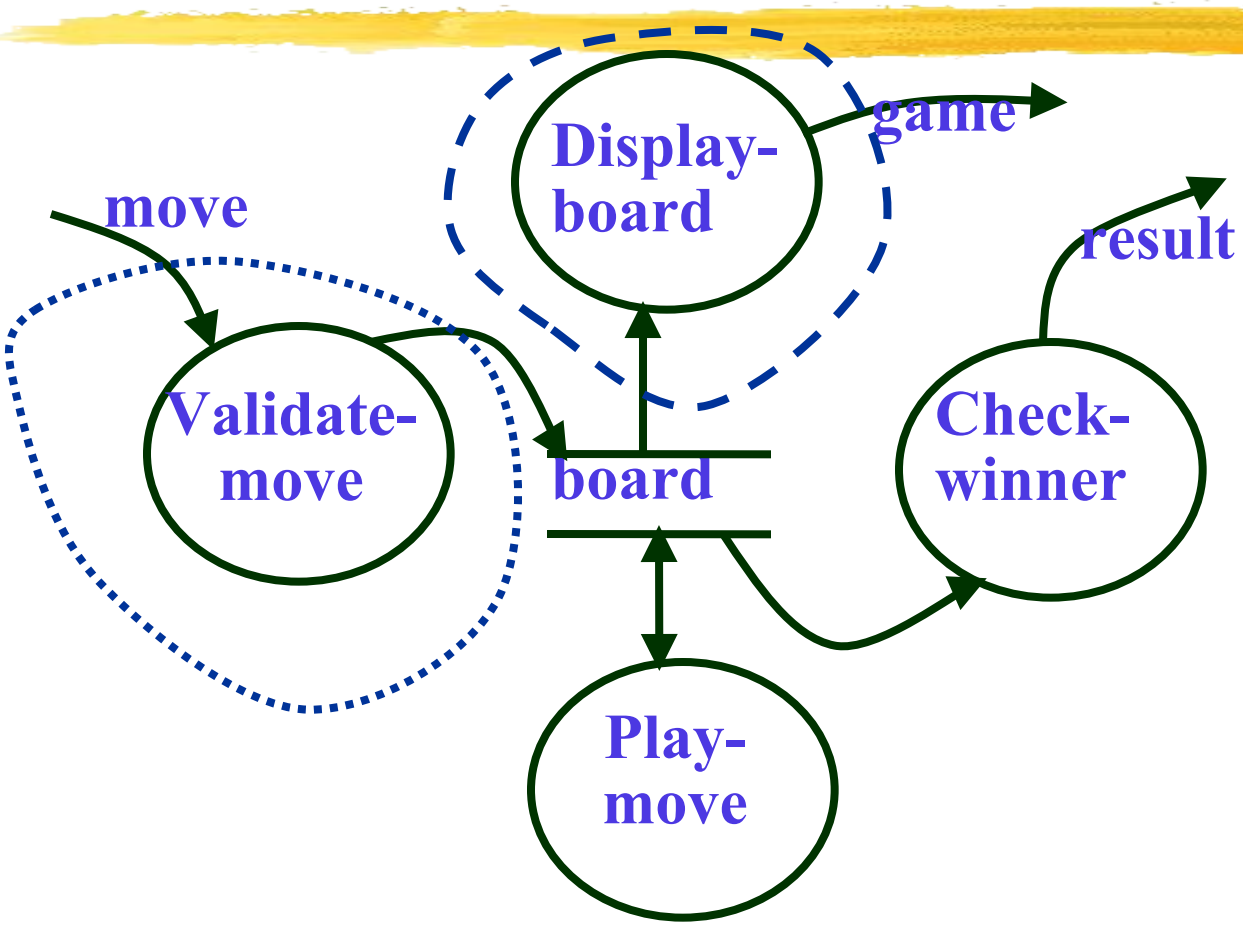
Example 2: Tic-Tac-Toe Computer Game

- As soon as either of the human player or the computer wins,
  - a message congratulating the winner should be displayed.
- If neither player manages to get three consecutive marks along a straight line,
  - and all the squares on the board are filled up,
  - then the game is drawn.
- The computer always tries to win a game.

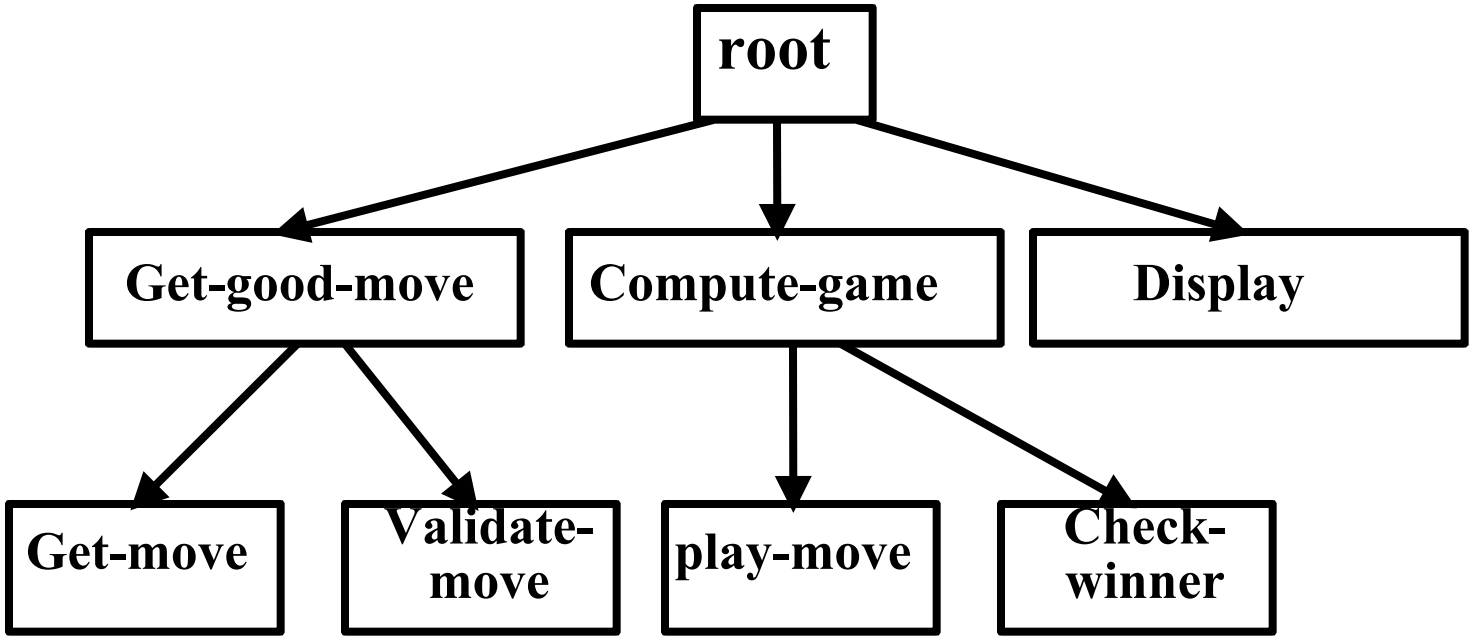
Context Diagram for Example 2



Level 1 DFD


Display-
board
game
move
result
Validate-
move
board
Check-
winner
Play-
move

Structure Chart



Summary

- We first discussed structured analysis of a larger problem.
- We defined some general guidelines
  - for constructing a satisfactory DFD model.
- The DFD model though simple and useful
  - does have several short comings.
- We then started discussing structured design.

Summary

- Aim of structured design:
  - transform a DFD representation into a structure chart.
- Structure chart represents:
  - module structure
  - interaction among different modules,
  - procedural aspects are not represented.

Summary

- Structured design provides two strategies to transform a DFD into a structure chart:
  - Transform Analysis
  - Transaction Analysis

Summary

- We Discussed three examples of structured design.
- It takes a lot of practice to become a good software designer:
  - Please try to solve all the problems listed in your assignment sheet,
  - not only the ones you are expected to submit.