Introduction to Software Specifications and Data Flow Diagrams

Neelam Gupta

The University of Arizona

Specification

- A broad term that means *definition*
- Used at different stages of software development for different purposes
- Generally, a statement of agreement (*contract*) between
  - producer and consumer of a service
  - implementer and user
- All desirable qualities must be specified

Uses of specification

- **Statement of user requirements**
  - **major failures occur because of misunderstandings between the producer and the user**
  - **"The hardest single part of building a softwarem system is deciding precisely what to build" (F. Brooks)**
- **Statement of requirements for implementation**

  ***requirements specification* refers to definition of external behavior**
  - **design specification must be verified against it**

  ***design specification* refers to definition of the software architecture**
  - **code must be verified against it**

Uses of specification (cont.)

- A reference point during maintenance
  - corrective maintenance only changes implementation
  - adaptive and perfective maintenance occur because of requirements changes
    - » requirements specification must change accordingly

Requirements Specification

- **Requirements Specification:** Written document, a graphical model, a formal mathematical model, a collection of usage scenarios, a prototype or any combination of these describing the external behavior of the system
- **Requirements Engineering Process**: involves all of the activities required to create an maintain the requirements specification document of a software system

Requirements Engineering

- Obtain overall requirements of product from customer including information and control needs, product function and behavior, overall product performance, design and interfacing constraints and other special needs.
- Allocate function and behavior to the four systems components
  - Software, Hardware, Data, People
- *Goal:* Specify a system that meets customers needs and expectations.

Requirements Engineering Process

- Requirements elicitation
- Requirements analysis and negotiation
- Requirements specification
- System modeling
- Requirements validation
- Requirements management

Requirements Elicitation

- Identify elicitation methods (interviews, focus groups, meetings)
- Identify people who will help specify requirements and understand their organizational bias
- Define “technical environment”
- Identify “domain constraints”
- Solicit participation from many people to get different points of view
- Create usage scenarios

Output of Requirements Elicitation

- A statement of need and feasibility
- A statement of scope for the product
- Description of technical environment of the system
- A set of usage scenarios
- Any prototypes developed
- List of people who participated in requirements elicitation

Requirements Analysis and Negotiation

- Categorize requirements and organize them into related subsets.
- Explore each requirement in relationship to others.
- Examine requirements for consistency, omissions, and ambiguity.
- Rank requirements based on the needs of customers/users.
- Examine if each requirement is achievable in the technical environment in which it will be used and if each requirement is testable, once implemented.

Requirements Analysis and Negotiation (cont.)

- Examine risks associated with each requirement.
- Rough estimates of development efforts made and used to assess the impact of each requirement on project cost and delivery time.
- Resolve conflicts in requirements by negotiating with users.

Requirements Specification

Written document, a graphical model, a formal mathematical model, a collection of usage scenarios, a prototype or any combination of these describing the external behavior of the system

Specification Qualities

(i) unambiguous

(ii) consistent

(iii) complete

- internal completeness
- external completeness

(iv) incremental

Unambiguous

- **Example: specification fragment for a word-processor**

> Selecting is the process of designating areas of the document that you want to work on. Most editing and formatting actions require two steps: first you select what you want to work on, such as text or graphics; then you initiate the appropriate action.

can an area be scattered?

Consistent

- **Example: specification fragment for a word-processor**

> The whole text should be kept in lines of equal length. The length is specified by the user. Unless the user gives an explicit hyphenation command, a carriage return should occur only at the end of a word.

What if the length of a word exceeds the length of the line?
(results in an inconsistency in the specifications)

Complete

- **Internal completeness**
  - **the specification must define any new concept or terminology that it uses**
    - » **glossary helpful for this purpose**
  - **the specification must document all the needed requirements**
    - » **difficulty: when should one stop?**

Incremental

- **Referring to the specification process**
  - **start from a sketchy specifications and progressively add details**
- **Referring to the specification document**
  - **document is structured and can be understood in increments**

Specification Styles

- Informal specifications written in a *natural language*.
- Semi-formal specifications use a notation with *precise syntax but imprecise semantics*
- Formal specifications written using a notation that has *precise syntax and semantics* (meaning).
- Operational specifications describe the *desired behavior* of the system.
- Descriptive specifications – state desired *properties* of the system.

An example

Operational Specifications: “Let a be an array of n elements. The result of its sorting is an array b of n elements such that the first element of b is the minimum of a (if several elements of a have the same value, any one of them is acceptable); the second element of b is the minimum of the array of n-1 elements obtained from a by removing its minimum element; and so on until all n elements of a have been removed.”

Descriptive Specifications: “The result of sorting array a is an array b which is a permutation of a and is sorted.”

- **Operational Specification**
  1. Data Flow Diagrams
  2. Finite State Machines
  3. Petri nets

- **Descriptive Specification**
  1. Entity-Relationship Diagrams
  2. Logic Specifications
  3. Algebraic Specifications

Data Flow Diagrams (DFD)

- A semi-formal operational specification
- System: collections of data that are manipulated by functions.
- Data: (i) can be *persistent* i.e., stored in *data repositories*; (ii) can flow i.e., represented by *data flows*
- DFD Notation


Bubbles used to represent functions

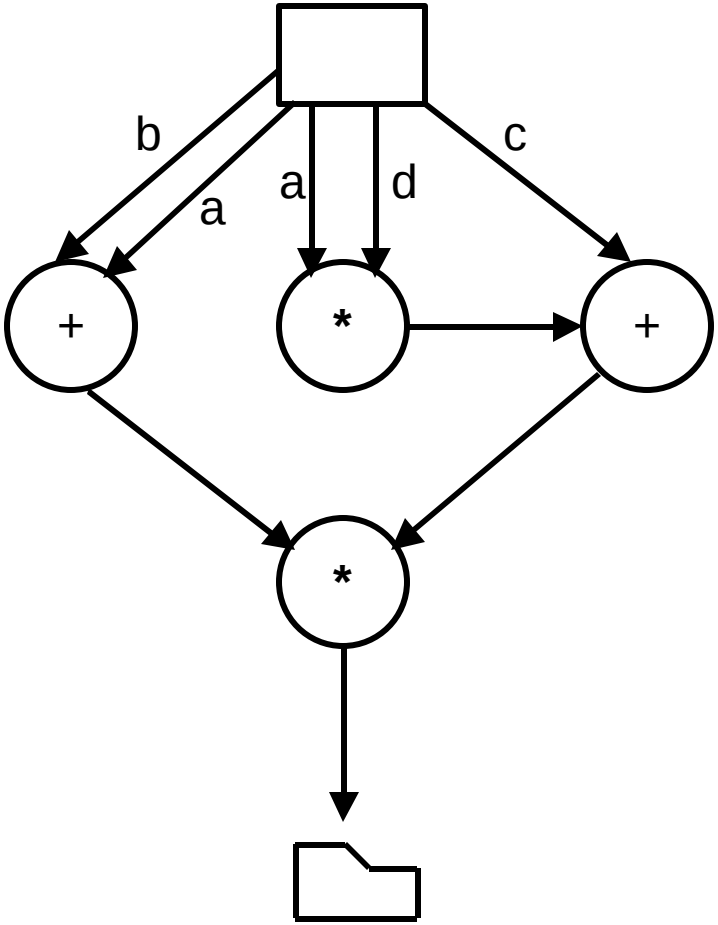

Arrows used to represent data flow


Open Boxes represent persistent data storage


I/O Boxes represent data acquisition and production during human computer interaction

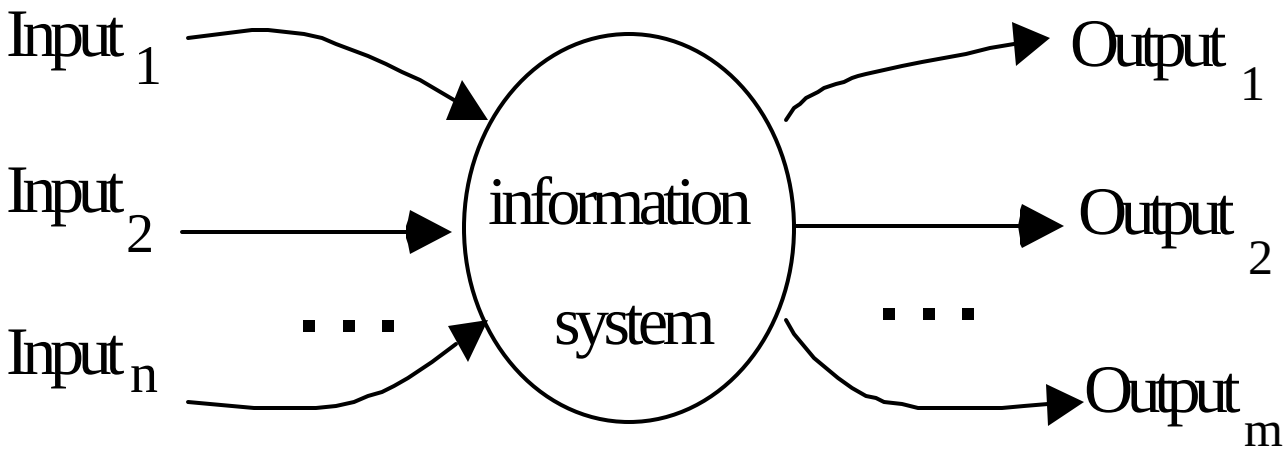
An Example of DFD

The example below specifies evaluation of expression (a+b)*(c+a*d)



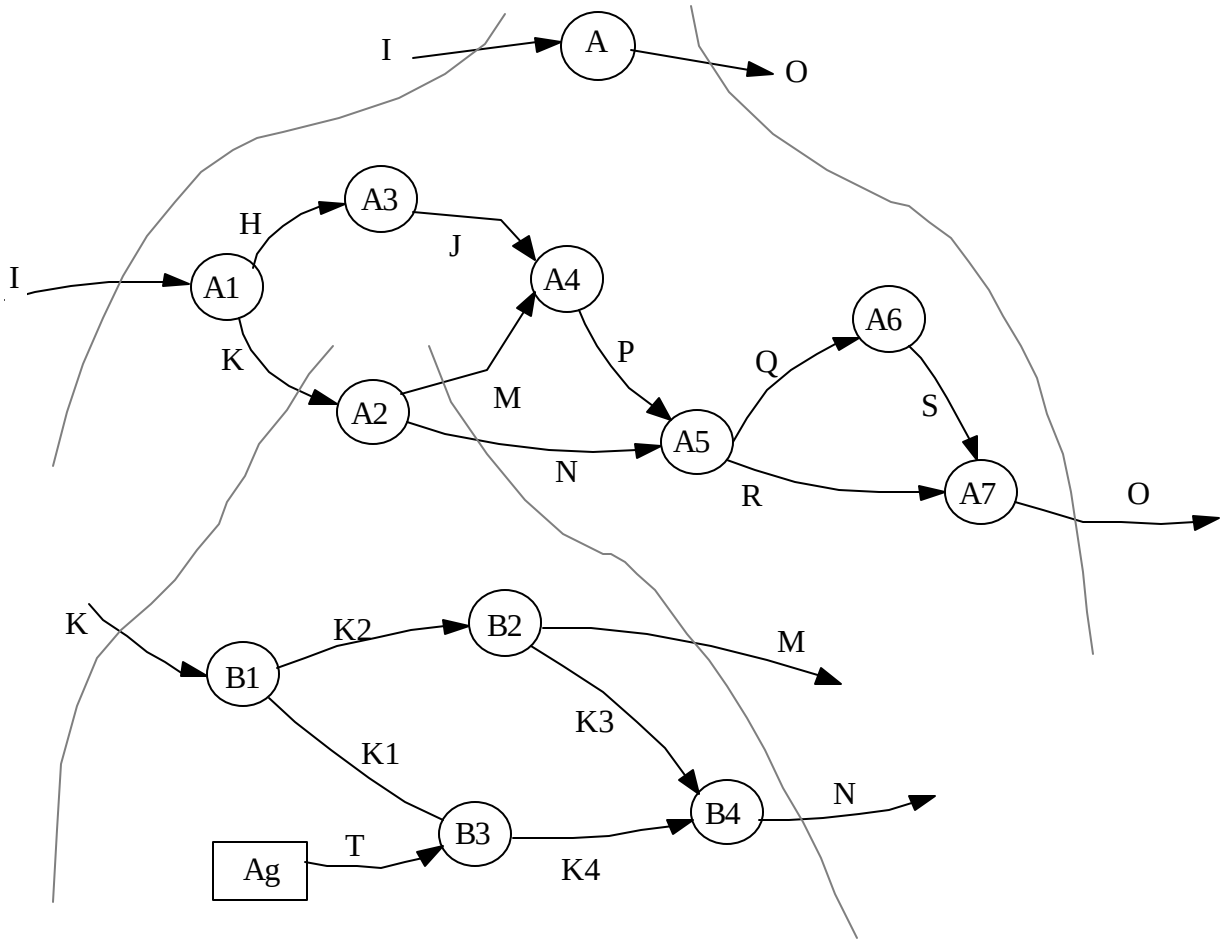
Construction of DFDs

1. Start from the “context” diagram

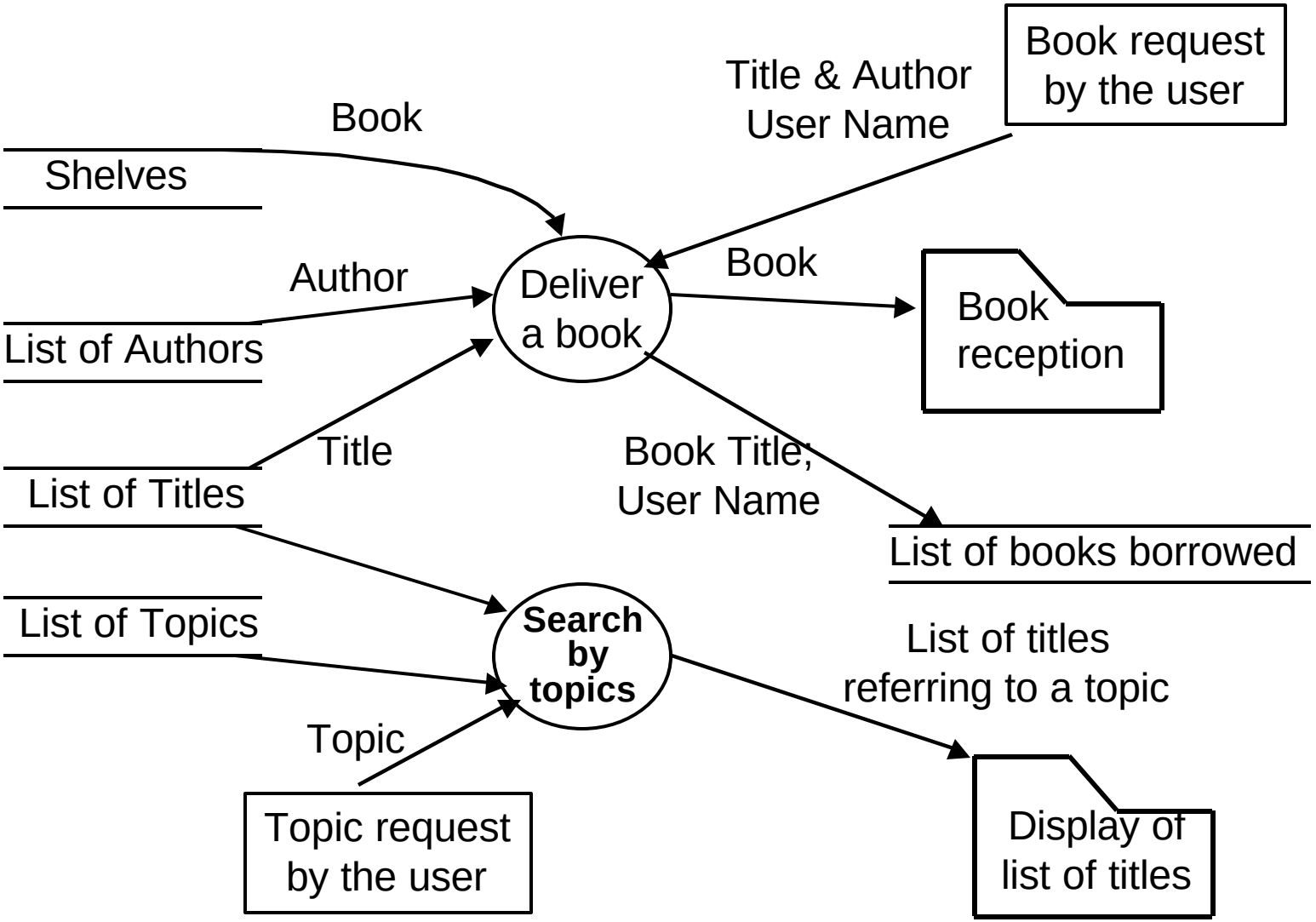


Construction of DFD's (cont.)

2. Proceed by refinements until you reach "elementary" functions (preserve balancing)



A DFD describing a simplified library information system

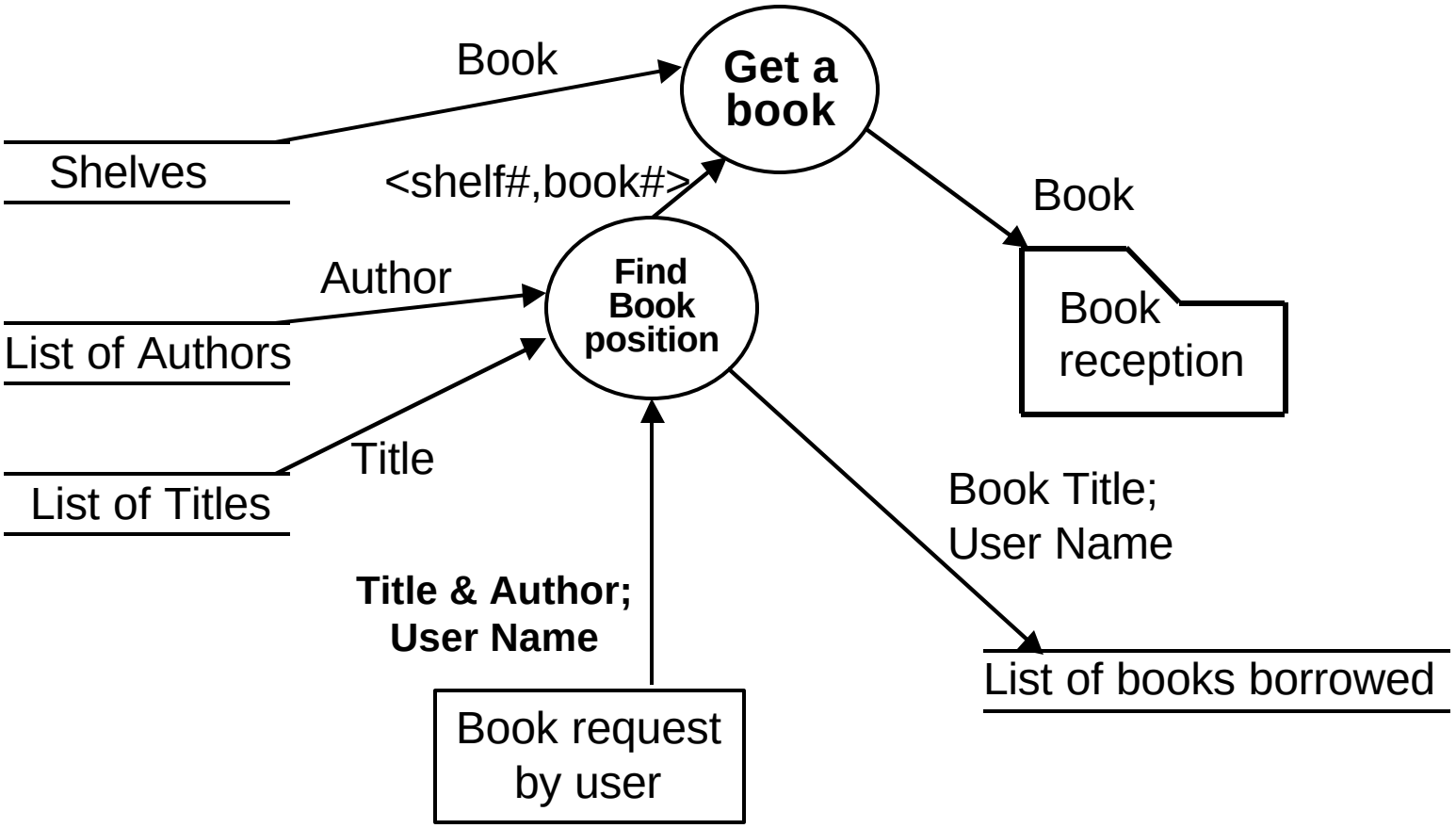


In order to obtain a book, the following are necessary.

- User request
- Access to bookshelves
- List of authors
- List of titles

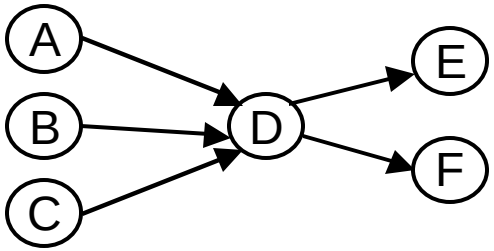
The way the book is actually obtained is not mentioned.

Refinement of "Deliver a book" in DFD for Library System



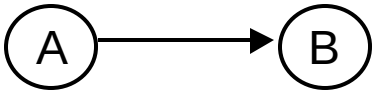
Limitations of DFD

1. Semantics of the symbols used is specified only by the identifiers chosen by the user. Easy to read, but …Informal semantics
2. Control aspects are not defined by the model



The above DFD does not specify how inputs are used and how outputs are produced by the function D.

- D needs all or only one of of A, B and C to execute?
- D outputs to just one or both of E and F?
- D outputs same or different data to E and F?



The above DFD does not specify synchronization between modules (absence of control information)

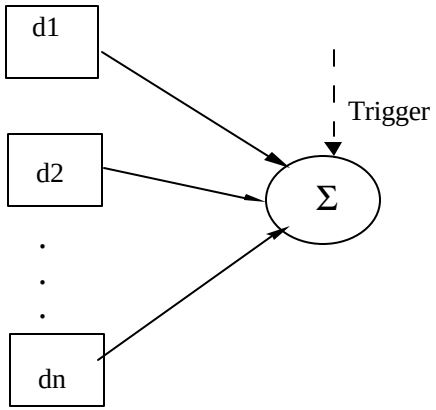
- **Does A produce a datum and waits until B has consumed it?**
- **Are A and B asynchronous activities with different speeds with a buffering mechanism between them to prevent data loss?**

Conclusions

- **DFD provide a graphical notation for capturing the flow of data and operations involved in an information system. However, they lack precise semantics**
- **A prototype to test whether specifications reflect the user's expectations cannot be derived directly from a DFD since no machine execution is possible without precise semantics for the notation.**
- **The syntax, i.e., way of composing bubbles, arrows, and boxed is defined precisely, but the semantics of DFDs is not specified precisely. (therefore DFD's provide a semiformal notation for specifying systems)**

Remedies

- **Use a complementary notation to describe aspects not captured adequately by DFDs.**
- **Augment DFD model by introducing new features.**



- **Revise DFDs to make them fully formal.**