Building the Analysis Model 2

Suradet Jitprapaikulsarn

Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Data Flow Diagram (DFD)

Represents how data objects are transformed as they move through the system

Input-Process-Output (I-P-O) view of software

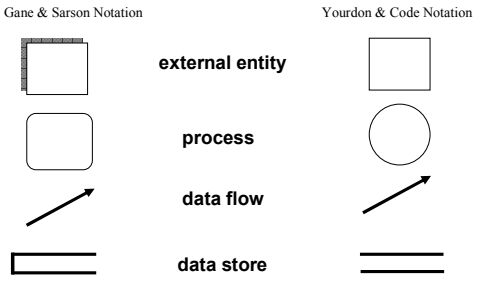
Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Flow model

Every computer-based system is an information transform …

Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

DFD Notation



Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

External Entity

A producer or consumer of data

Examples: a person, a device, a sensor

Another example: computer-based system

Data must always originate somewhere and must always be sent to something

Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Process

A data transformer (changes input to output)

Examples: compute taxes, determine area, format report, display graph

Data must always be processed in some way to achieve system function

Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

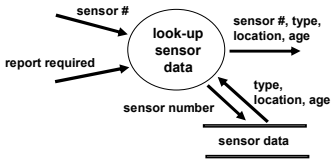
Flow

Data flows through a system, beginning as input and be transformed into output.

Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Data Store

Data is often stored for later use.



Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Guideline for DFD

- all icons must be labeled with meaningful names
- the DFD evolves through a number of levels of detail
- always begin with a context level diagram (also called level 0)
- always show external entities at level 0
- always label data flow arrows
- do not represent procedural logic

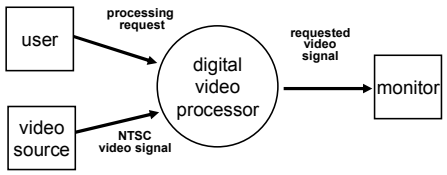
Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Constructing DFD—1

- review the data model to isolate data objects and use a grammatical parse to determine "operations"
- determine external entities (producers and consumers of data)
- create a level 0 DFD

Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

DFD Level 0



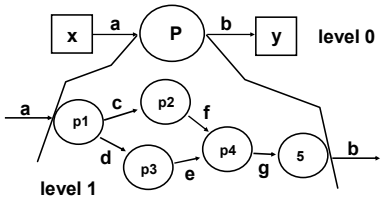
Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Constructing DFD—2

- write a narrative describing the transform
- parse to determine next level transforms
- "balance" the flow to maintain data flow continuity
- develop a level 1 DFD
- use a 1:5 (approx.) expansion ratio

Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

DFD Hierarchy



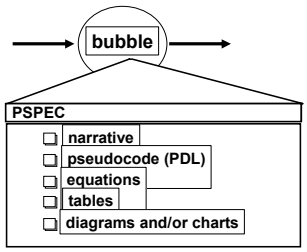
Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

DFD Notes:

- each bubble is refined until it does just one thing
- the expansion ratio decreases as the number of levels increase
- most systems require between 3 and 7 levels for an adequate flow model
- a single data flow item (arrow) may be expanded as levels increase (data dictionary provides information)

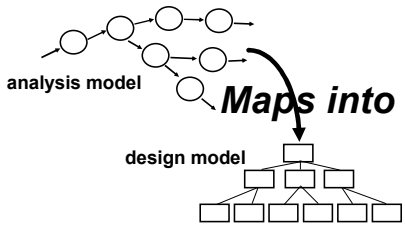
Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Process Specification (PSPEC)



Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

DFD: What's next?



Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Control Flow Diagram

- Represents "events" and the processes that manage events
- An "event" is a Boolean condition that can be ascertained by:
  - listing all sensors that are "read" by the software.
  - listing all interrupt conditions.
  - listing all "switches" that are actuated by an operator.
  - listing all data conditions.
  - recalling the noun/verb parse that was applied to the processing narrative, review all "control items" as possible CSPEC inputs/outputs.

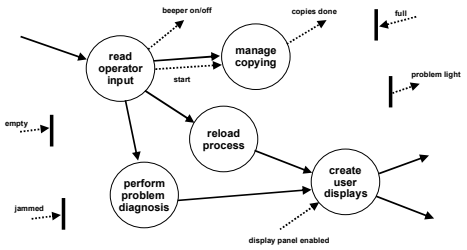
Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

The Control Model

- the control flow diagram is "superimposed" on the DFD and shows events that control the processes noted in the DFD
- control flows—events and control items—are noted by dash arrows
- a vertical bar implies an input to or output from a control spec (CSPEC) — a separate specification that describes how control is handled
- a dashed arrow entering a vertical bar is an input to the CSPEC
- a dashed arrow leaving a process implies a data condition
- a dashed arrow entering a process implies a control input read directly by the process
- control flows do not physically activate / deactivate the processes—this is done via the CSPEC

Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Data Flow with Control



Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Control Specification (CSPEC)

The CSPEC can be:

- ❑ state diagram (sequential spec)
- ❑ state transition table
- ❑ decision tables
- ❑ activation tables

} combinatorial spec

Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Guidelines for Building a CSPEC

- list all sensors that are "read" by the software
- list all interrupt conditions
- list all "switches" that are actuated by the operator
- list all data conditions
- recalling the noun-verb parse that was applied to the software statement of scope, review all "control items" as possible CSPEC input/outputs
- describe the behavior of a system by identifying its states; identify how each state is reach and defines the transitions between states
- focus on possible omissions … a very common error in specifying control, e.g., ask: "Is there any other way I can get to this state of exit from it?"

Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005