Session 2b: structured specifications

Purpose and criteria
Structured specification components
Introduction to dataflow diagrams

Criteria for the ESD
(from session 1)
1. Must be understandable in complete detail by
   a. sponsoring user representatives
   b. programmers and other developers
2. Must be
   a. unambiguous
   b. complete
   c. consistent
   How do we know?
3. Must provide a reasonable basis for estimating.
   We may argue about how well a particular technique supports the criteria, but not about the criteria themselves.

Origin of structured system specification
- ~1978: Structured programming (SP) was recent and viewed as highly successful
  - Systems analysis was still a major bottleneck in application development
  - Can't we do something for requirements specification similar to SP for programming?
- Several experts worked to devise structured analysis (SA).
- Several published books and presented courses, notably:
  - Gane & Sarson
  - DeMarco

Characteristics of SA
- Satisfies the criteria for the ESD
  - In particular, understandable by non-technical people without special training.
- Every component has a context
  - Its relationships to other components are clear
  - No "stand alone" documents
- A logical place to start reading
  - Follow links to other specification components
- Prefers graphic to text wherever possible
  - a picture of the system or part of the system

Agree or disagree?
Components of a structured specification

- **Dataflow diagrams** (session 4)
  - Tie everything together
  - Provide the picture
  - Hierarchical ("leveled") to manage detail

- **Data item** definitions (session 3)
  - Comprise the project data dictionary

- **Process** definitions (session 6)
  - Formulas, procedures, algorithms, decision rules, etc.

---

**Dataflow diagram**

- Shows the relationships among other components.
- Four kinds of element:
  - Active (does something)
    1. process (DeMarco & Robertson's call it a "bubble")
  - Passive
    2. data store
    3. dataflow
    4. external interface (user or other system)
- DeMarco and Gane & Sarson versions are virtually identical except that they use different graphic symbols.

---

**DFD Graphic symbols**

1. **a process**
   - **DeMarco's symbol**
     ![DeMarco's symbol]
     He calls it a "bubble"
   - **Gane & Sarson's symbol**
     ![Gane & Sarson's symbol]
     Rectangle with rounded corners

   Which do you prefer?

2. **dataflow**
   - **DeMarco's symbol**
     ![DeMarco's symbol]
     labelled curved arrow
   - **Gane & Sarson's symbol**
     ![Gane & Sarson's symbol]
     labelled rectilinear connector

Which do you prefer?
Identifying a process on a DFD

- Give it a unique **number**
- Give it a **name** in verb-object form.
  Examples:
  - Edit transaction
  - Check customer's credit
  - Reserve stock
- Omit the object if it's obvious from an output dataflow.

![Diagram](attachment:process-diagram.png)

Graphic DFD symbols

- **DeMarco's versus Gane & Sarson's:**
  - Since the information content is virtually identical, you'd expect the choice to make little difference
  - Not so! Practitioners have argued vehemently in favor of one or the other.

- **Why do we care?**
  - In this course we don't.
  - Use whichever convention you prefer (or make up your own), but don't mix them in the same set of documents
  - The Robertson text book uses DeMarco
  - Visio prefers Gane & Sarson

Arguments for each

- Informal appearance is less intimidating to user audience
- Easier to draw by hand, esp. without a template
- Neater appearance
- Process mimics shape of a sheet of paper
- Easier to draw by **Visio**

Which graphics symbols do we use in this course?

- Your choice, but don't mix them in any one assignment or set of documents
- You can even invent your own as long as the meaning is clear
- How should we produce them?
  - Use Visio or a similar diagramming tool
  - Use a digramming template
  - Draw them by hand (carefully)
### More about DFDs in session 4

- Meanwhile, here are the rest of the symbols
- We’ve seen 2 of the 4

### More DFD symbols

3. **Terminator**
   - Represents something at the boundary of the system
     - a user
     - another system
   - Involved with input or output or both

4. **Data store**
   - Like a file or a database
   - Dataflows go in and come out; no timing assumption
   - That's all!

**What do they look like?**

### DFD characteristics

- A dataflow diagram, as the name implies, shows the relationships among
  - processes
  - data
  - external interfaces (end users or other systems)
- It usually fits on a single page

**What if it doesn’t?**

- Unlike a flowchart it does not show:
  - sequence of processing
  - decisions or conditional logic

**So, how do we specify those things?**

### The rest of a structured specification

- Dataflow diagrams have central importance in showing the relationships among system components, but we still need:
  - rigorous data definitions (session 3)
  - rigorous process definitions (session 6)
  - data model
    - entity-relationships
    - O.O. class definitions and hierarchy
  - miscellaneous optional information
    - constraints
    - use cases
    - . . .
Getting started: the top level

- The simplest DFD is a context diagram (also called a level-0 DFD)
- It shows the scope of an application system
  - Who uses the system?
  - What goes in?
  - What comes out?
- This is the obvious place to start
  - writing an ESD
  - getting the sponsoring users' OK

The Context diagram

- Shows
  - the whole application system as a single process
  - all users and external interfaces
  - major system outputs and system inputs
  - major data stores
- Example:
  - Robertson text fig. 2.6.17 (p.190)

More DFD Examples

- See textbook p. 107

- Context diagrams:
  - http://webpages.cs.luc.edu/~cweisert/COMP320/Context1b.pdf (DeMarco style)