Process Modeling

Chapter 7

Process Design

- Seldom the responsibility of the database designer or DBA
- However, understanding the basics aids communication with the process designers and ensures that the database design supports the intended processes

Business Process Example

- Find all unshipped orders in the database.
- For each order:
  - Check for available inventory. If sufficient inventory for the order is not available, skip to the next order.
  - Check the customer's credit to make sure they are not over their credit limit or have some other credit problem, such as overdue payments. If there is a credit problem, skip to the next order.
  - Generate the documents required to pack and ship the order (packing slip, shipping labels, and so on), and route them to the shipping department.
  - When the shipping department has finished with the order, create the invoice for the order and bill the customer accordingly.

Flowcharts

- Diagrams used to help visualize simple processes
  - Both during the design process and as documentation of completed designs.
- Used by industrial engineers over 20 years before their introduction into data processing
- In the 1960s and 1970s, flowcharts were in mainstream use for designing and documenting computer programs,
  - Also an element of most computer science degree programs offered by colleges and universities

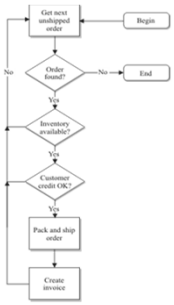
Flowcharts

- Use in computer program design diminished in the later 1970s, and they were seldom used in the 1980s, largely due to several factors:
  - Better diagramming methods were developed.
  - Structured programming techniques made flowcharts less necessary because programs were divided into single-purpose routines.
  - Third generation programming languages proved to be self-documenting as long as programmers added effective comments.

The Flowchart

- Process steps shown with rectangles
- Decision points shown with diamonds
- Flow of control shown with lines and arrows
- Start and end points shown with ellipses
- Connectors shown with “home plate” symbols

Flowchart Example



Flowchart Strengths

- Procedural language programmers find them naturally easy to learn and use
- Applicable to procedures outside of a programming context
- Useful for spotting reusable (common) components
- May be easily modified and can evolve as requirements change

Flowchart Weaknesses

- Not applicable to nonprocedural or object-oriented languages
- Cannot easily model some situations such as recursive processes

Function Hierarchy

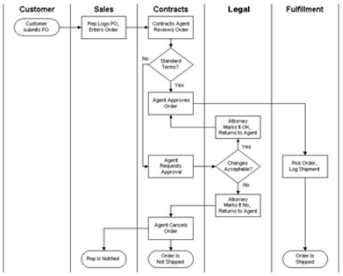
Function Hierarchy Strengths

- Quick and easy to learn and use
- Quickly document the bulk of the function (get to 80% quickly)
- Great overview at high and medium levels of detail

Function Hierarchy Weaknesses

- Checking quality is difficult and subjective
- Cannot handle complex interactions between functions
- Do not clearly show the sequence of process steps or dependencies between steps
- Not effective for large hierarchies or at very detailed levels

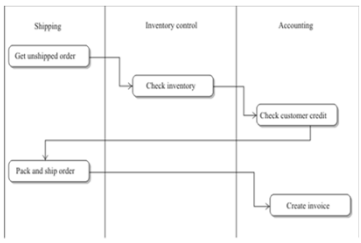
Swim Lane Diagrams



The Swim Lane Diagram

- An extension of the flowchart diagram that visually distinguishes which part of an organization (which organizational unit) is responsible for executing the subprocesses within a business process.
- First appeared in the 1940s as a variation of the flow chart
  - Originally known as “multi-column charts”
  - Geary Rummier and Alan Brache used the term “swim lane diagrams” in their book *Improving Performance* (Jossey-Bass, 1990)
  - Many people started using the term "Rummler-Brache Diagrams" in honor of the authors

Swim Lane Diagram Example



Swim Lane Diagram Strengths

- Unmatched ability to show who does what in the organization
- Excellent for identifying inefficiencies of existing processes
- Lends itself well to reengineering efforts

Swim Lane Weaknesses

- Does not represent complicated processes well (many steps or complex step dependencies)
- Does not show error and exception handling

The Data Flow Diagram

- A graphical representation of the flow of data through a computerized system
  - Diagrams combined hierarchically in an effort to achieve the best combination of the flowchart and the function hierarchy diagram
- Data Centric, unlike flow charts and swim lane diagrams
- Larry Constantine, the original developer of structured design, proposed data flow diagrams in 1974
  - Based on the work of D. Martin and G. Estrin
  - Refined and popularized by Ed Yourdon and Tom Demarco
  - Extended and refined by Chris Gane and Trish Sarson

The Data Flow Diagram

- Processes represented with rounded rectangles
- Data stores represented with open-ended rectangles
- Sources and destinations of data shown with squares
- Flows of data shown with lines with arrowheads indicating direction of flow

Data Flow Diagram Example

DFD Components

- Processes are represented with rounded rectangles, and are numbered hierarchically
- Data stores are represented with an open-ended rectangle.
  - A ***data store*** is a generic representation of data that is made persistent by being stored somewhere, such as in a file, database, or even a written document.
- Sources and destinations of data (external entities in relational terminology) are shown using squares
- Flows of data are shown using lines with arrowheads indicating the direction of flow

Data Flow Diagram Strengths

- Easily shows the overall structure of the system without sacrificing detail
- Good for top-down design work
- Good for presentation of systems designs to management and business users

Data Flow Diagram Weaknesses

- Time consuming and labor intensive to develop for complex systems
- Top-down design is ineffective when requirements are sketchy and continuously evolving
- Poor at showing complex logic, but lowest level diagrams easily supplemented

UML Process Diagrams

Type	Name	Description
Structure	Class diagram	Shows a collection of static model elements such as classes and types, their contents, and their relationships
Structure	Component diagram	Depicts the components that make up an application, system, or enterprise
Structure	Composite structure diagram	Depicts that internal structure of a classifier, (such as a class, component, or use case), including the classifier's interaction points to other parts of the system (added in UML 2.x)
Structure	Deployment diagram	Shows the execution architecture of systems, including nodes, hardware/software environments, and the middleware that connects them
Structure	Object diagram	Depicts objects and their relationships at a point in time
Structure	Package diagram	Shows how model elements are assembled into packages as well as the dependencies between packages
Behavior	Activity diagram	Depicts high-level business processes, including data flow
Behavior	State machine diagram	Describes the states an object or interaction may be in, and the transitions between states
Behavior	Use case diagram	Shows actors, use cases, and their interactions
Behavior	Communication diagram	Shows instances of classes, their interrelationships, and the message flow between them
Behavior	Interaction overview diagram	A variant of an activity diagram that depicts and overview of the control flow within a system or business process (added in UML 2.x)
Behavior	Sequence diagram	Depicts the time ordering of messages between classifiers, essentially showing the sequential logic of the system
Behavior	Timing diagram	Depicts the change in state or condition of a classifier instance or role over time (added in UML 2.x)

Relating Entities and Processes

- CRUD Matrix
  - One axis represents major processes
  - Other axis represents major entities
  - Letters represent entity usage for processes
    - C for Create
    - R for Read
    - U for Update
    - D for Delete

CRUD Matrix Example

	Product	Order	Customer	Invoice
Order Entry	R	CRU	RU	
Order Fullfillment	RU	RU	R	C
History Management		RD	R	

Problems Identified in CRUD Matrices

- Entities with no create process
- Entities with no delete process
- Entities with no update process
- Entities with no read process
- Processes that delete or update without reading
- Processes that only read (no create, delete or update processes)


"That's the last bug. Tomorrow we go production..."