Requirements Modeling:
Flow, Behavior, Patterns, and Webapps (Ch. 7)

Cengiz Günay
CS485/540 Software Engineering

Fall 2014, Some slides courtesy of Joan Smith, Roger Pressman, and the Internets
Today:
Req. Modeling: Flow, Behavior, Patterns, and Webapps (Ch. 7)

“failing to write a spec is the single biggest unnecessary risk you take in a software project”
Fact/Fallacy Tidbit

• Fact 25
  
  Missing requirements are the hardest requirements errors to correct

• Discussion
  – Requirements come from people-to-people communication
  – Therefore naturally error-prone
  – Missed requirements = missed logic, potentially affecting all aspects of the delivered product
    • Easy to find an error that is in existing code
    • Hard to find an error in code that doesn’t exist!

From Robert Glass, “Facts & Fallacies of Software Engineering”
Flow-Oriented Modeling

- Represents how data objects are transformed as they move through the system.
- **Data flow diagram (DFD)** is the diagrammatic form that is used.
- Considered by many to be an “old school” approach, but continues to provide a view of the system that is unique—it should be used to supplement other analysis model elements.
Data-flow: good for parallelism

Used in LabView programming language and Matlab’s Simulink:
Data-flow: good for parallelism

Used in LabView programming language and Matlab’s Simulink:

- no variables
- clear indication to where to free memory
- independent data streams can be processed in parallel
The Flow Model

Every computer-based system is an information transform ....
Flow Modeling Notation

- External entity
- Process
- Data flow
- Data store
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
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
Data Flow

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

- base
- height
- compute triangle area
- area

Data Stores

Data is often stored for later use.

- sensor #
- report required
- sensor number
- sensor data
- sensor #, type, location, age
- type, location, age

Data Flow Diagramming: Guidelines

- 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
Constructing a DFD—I

- review user scenarios and/or 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
Level 0 DFD Example

Constructing a DFD—II

- 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
COMPLETE MAP OF OPTIMAL TIC-TAC-TOE MOVES

Your move is given by the position of the largest red symbol on the grid. When your opponent picks a move, zoom in on the region of the grid where they went. Repeat.

MAP FOR X:
The Data Flow Hierarchy

Flow Modeling 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)
Process Specification (PSPEC)

- narrative
- pseudocode (PDL)
- equations
- tables
- diagrams and/or charts
DFDs: A Look Ahead

Maps into

analysis model

design model

Control Flow Modeling

- 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.
Control Specification (CSPEC)

The CSPEC can be:

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

combinatorial spec
Behavioral Modeling

- The behavioral model indicates how software will respond to external events or stimuli. To create the model, the analyst must perform the following steps:
  - Evaluate all use-cases to fully understand the sequence of interaction within the system.
  - Identify events that drive the interaction sequence and understand how these events relate to specific objects.
  - Create a sequence for each use-case.
  - Build a state diagram for the system.
  - Review the behavioral model to verify accuracy and consistency.
State Representations

- In the context of behavioral modeling, two different characterizations of states must be considered:
  - the state of each class as the system performs its function and
  - the state of the system as observed from the outside as the system performs its function

- The state of a class takes on both passive and active characteristics [CHA93].
  - A *passive state* is simply the current status of all of an object’s attributes.
  - The *active state* of an object indicates the current status of the object as it undergoes a continuing transformation or processing.
State Diagram for the ControlPanel Class

- **reading**
  - key hit
  - password entered
  - password = incorrect & numberOfTries ≤ maxTries
    - do: validatePassword
    - timer < lockedTime
    - timer > lockedTime
  - password = correct
    - selecting
    - activation successful
  - numberOfTries > maxTries
    - locked
    - timer < lockedTime
    - timer > lockedTime
The States of a System

- **state**—a set of observable circumstances that characterizes the behavior of a system at a given time
- **state transition**—the movement from one state to another
- **event**—an occurrence that causes the system to exhibit some predictable form of behavior
- **action**—process that occurs as a consequence of making a transition
Behavioral Modeling

- make a list of the different states of a system (How does the system behave?)
- indicate how the system makes a transition from one state to another (How does the system change state?)
  - indicate event
  - indicate action
- draw a state diagram or a sequence diagram
Sequence Diagram

Control-flow Modeling

- Model View Controller (MVC) framework for WebApps
Requirements Modeling for WebApps

Content Analysis. The full spectrum of content to be provided by the WebApp is identified, including text, graphics and images, video, and audio data. Data modeling can be used to identify and describe each of the data objects.

Interaction Analysis. The manner in which the user interacts with the WebApp is described in detail. Use-cases can be developed to provide detailed descriptions of this interaction.

Functional Analysis. The usage scenarios (use-cases) created as part of interaction analysis define the operations that will be applied to WebApp content and imply other processing functions. All operations and functions are described in detail.

Configuration Analysis. The environment and infrastructure in which the WebApp resides are described in detail.
The Content Model

- **Content objects** are extracted from use-cases
  - examine the scenario description for direct and indirect references to content
- **Attributes** of each content object are identified
- The **relationships** among content objects and/or the hierarchy of content maintained by a WebApp
  - Relationships—entity-relationship diagram or UML
  - Hierarchy—data tree or UML
Data Tree

The Interaction Model

- Composed of four elements:
  - use-cases
  - sequence diagrams
  - state diagrams
  - a user interface prototype

- Each of these is an important UML notation and is described in Appendix I
Sequence Diagram


new customer

:Room

:FloorPlan

:Product Component

:Bill of Materials

Floor Plan Repository

BoM Repository

describes room*

places room in floor plan

save floor plan configuration

selects product component*

add to BoM

save bill of materials

...
The Functional Model

- The functional model addresses two processing elements of the WebApp
  - user observable functionality that is delivered by the WebApp to end-users
  - the operations contained within analysis classes that implement behaviors associated with the class.
- An activity diagram can be used to represent processing flow
These slides are designed to accompany *Software Engineering: A Practitioner’s Approach, 7/e* (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.
The Configuration Model

- **Server-side**
  - Server hardware and operating system environment must be specified
  - Interoperability considerations on the server-side must be considered
  - Appropriate interfaces, communication protocols and related collaborative information must be specified

- **Client-side**
  - Browser configuration issues must be identified
  - Testing requirements should be defined
Navigation Modeling-I

- Should certain elements be easier to reach (require fewer navigation steps) than others? What is the priority for presentation?
- Should certain elements be emphasized to force users to navigate in their direction?
- How should navigation errors be handled?
- Should navigation to related groups of elements be given priority over navigation to a specific element.
- Should navigation be accomplished via links, via search-based access, or by some other means?
- Should certain elements be presented to users based on the context of previous navigation actions?
- Should a navigation log be maintained for users?
Navigation Modeling-II

- Should a full navigation map or menu (as opposed to a single “back” link or directed pointer) be available at every point in a user’s interaction?
- Should navigation design be driven by the most commonly expected user behaviors or by the perceived importance of the defined WebApp elements?
- Can a user “store” his previous navigation through the WebApp to expedite future usage?
- For which user category should optimal navigation be designed?
- How should links external to the WebApp be handled? overlaying the existing browser window? as a new browser window? as a separate frame?
Fact/Fallacy Tidbit

• Fact 26
  The list of “derived requirements” can be 50x longer than the list of original requirements

• Discussion
  – Requirements inform the design; the design informs the solution; the solution creates new requirements
  – Complexity may increase as well
  – Requirements traceability is affected
    • Traceability involves mapping a customer requirement to each part of the design/code/test/documentation
    • Sometimes used for code analysis – identifying dangling code, for example (i.e., code that no longer meets a requirement)
    • Design “consequences” (implicit requirements) aren’t tied directly to an original customer requirement, so mapping is unclear at best
    • Each phase of the project and addition of further implicit requirements adds to the traceability problem; no just a simple linked-list problem but linked-lists of linked-lists – extremely complex

From Robert Glass, “Facts & Fallacies of Software Engineering”