Use Cases

- Specify “actors” and how they interact with various component parts of a system
  - This is an external "black box" view of a system
- System is a collection of “use cases” (i.e., capabilities provided to users/actors)
- Represented using diagrams and schemas
  - Diagrams show flow of “uses” between actors and use cases
  - Schemas are more formal non-pictorial definitions

Example Use Case diagram

Use Case schema

Stakeholders and Questions

- Users of the system
  - How do I want it to behave
- Developers of the system
  - What capabilities do I need to implement
- Customers
  - What are these capabilities and behaviors worth

Message Sequence Diagrams

- Sometimes called “ladder charts”
- Represent a particular sequence of messages exchanged between entities
- Popular in object-oriented methods to represent communications between objects
- Shows one particular communication sequence in one run of the system
  - Shows behavior as well as communication
- Can be extended with conventions to represent looping, casing, timeouts, synchronization, global conditions across different entities, delayed message reception, etc.
Example Message Sequence Diagram

Stakeholders and Questions

- Users
  - What behaviors do I want/need
- Developers
  - What interactions need to be supported
  - What are the entities
  - What operations are to be supported
- Inspectors?
  - Are there safety issues
  - ??

Class Diagram

- In widespread use. Consists of
  - Name
  - Attributes
  - Operations/Methods
  - Associations
    - Cardinalities
    - Annotations
    - Qualifiers
    - Interfaces
    - More..... (much more)

Collaboration Diagrams

- Popular in object-oriented methods to represent message exchanges between objects
- Object specification augmented by annotations that represent dataflows between the communicating objects
- Differ from other notations
  - Nodes represent objects, not activities (as in DFDs, activity diagrams, activity charts, and block diagrams)
  - Notes represent object instances, not object classes
- As in sequence diagrams, represent the sequence of messages in one particular scenario, not all possible communications scenarios.
Representing Other Types of Things

- **Data, Objects, Artifacts**
  - These are clearly secondary in all of the above diagrams
  - Often are more important than functional view
  - Harder to depict diagrammatically

- **Process artifacts and views**
  - Primary interest of management and customer stakeholders for much of the time
  - Typical questions:
    - What is the (development) plan? schedule?
    - Are we almost done?
    - What are we going to do next?
    - What if Joe quits?
  - Different representations are needed to reply effectively

Primitive Data Representations

- **Record Structures**
- **Array Structures**
- **Linked List Structures**

Representation of Data/Objects

- Complement to emphasis on representation of activities
  - Foregoing representations all focused on activities
  - Weak capabilities for describing data and objects
  - Seen mostly as effects of activities
  - Numerous places where data descriptions were needed
  - Supposed to be sorted (which way?)
  - Elements had fields (what types?)

- Problems in doing this well
  - What information needed/what questions need answers?
    - Hierarchical decomposition of data
    - Legal actions on data
    - Typing information
    - What forms of representation will be useful?
      - Natural language
      - Diagrammatic
      - Formal language

- Problems in doing this well
  - What information needed/what questions need answers?
    - Hierarchical decomposition of data
    - Legal actions on data
    - Typing information
    - What forms of representation will be useful?
      - Natural language
      - Diagrammatic
      - Formal language

Much more will be said about this Later in the course

Primitive Process Representations

- **PERT/CPM Charts**
- **Gantt Charts**

Pert and CPM Charts

- Depict the process as a network of tasks
- Each step is a circle
- Incoming arrows are steps that must complete before this one
- Outgoing arrows are steps that might follow this one
- Each step has a time estimate
- No loops allowed
- So that maximum "flow time" can be computed
  - Along the "critical path"
- Early management tool
- Very naive and oversimplified view
  - no loops!!
  - Simplicity is its strength and weakness
Gantt Charts

- Familiar milestone charts, progress charts, ...
- Time represented along a horizontal axis
- Each task (person, ...) represented by a solid bar plotted against the time line
- Bar starts at "start time" and ends at "end time"
- Key Milestones represented by triangles placed along the bar
- Shows how tasks juxtapose
- Shows who should be doing what at all times
- Shows how product is supposed to evolve over time
- Effective for spotting schedule slippages

Example Gantt Chart

Multirepresentation Systems

- Have seen that different representations are of different uses
- One diagram may be useful in different ways to different stakeholders
- But most stakeholders require a variety of diagrams
- Several different diagrams can be expected to be needed to satisfy the different stakeholders
- Problems with different views/diagrams
  - Are they all representing the same software product?
  - How to assure that they are all consistent with each other?
  - If the product changes, then ALL views must change correspondingly

STATEMATE

- Focus on Statecharts—an enhancement of FSM's
- Augmented by other views (e.g. activity Diagrams)
- Key feature is maintenance of consistency among views
- Rigorously defined semantics
  - Including specification of needed consistency
- References
- Commercially available software system
Multiple Views in Statemate

- Rationale for multiple views: Too much information in a single diagram creates clutter, confusion, defeats clarity
- Advantage of multiple views: Each represents a different viewpoint, different model, with a different diagram
- Disadvantage: Reader needs to synthesize views, assure that they are really consistent with each other
- Multiple views in Statemate:
  -- Module Charts (a hierarchy representing capabilities)
  -- Activity Charts (hierarchical dataflow charts)
  -- Statecharts (hierarchical finite state machines)
  -- Sequence Charts
- All facilitated by a slick user interface
- Statemate views depict some different views, but also overlap with each other: facilitates cross-checking for consistency and easier comprehension

The Importance of Redundancy

- Specifying or doing the same thing more than once
- Usually considered undesirable in computing
- Typically regarded as desirable in engineering
- Particularly useful in safety engineering
  -- NASA 5-way redundancy
- Can help assure that multiple views are not inconsistent
  -- Different views should not be inconsistent about things in their intersection

Module Charts

- Hierarchy shown by
  -- Indentation
  -- Nesting module-charts inside each other
- How many levels of nesting without losing clarity?
Activity Chart

- A Data Flow Diagram
  - Hierarchical
    - Focus (depicted by solid boxes) on functions
    - Arrows depict data flows
  - All of this helps user/reader to associate features of one with features of the other
  - New dataflow diagram feature:
    - Control box (rounded): at most one per activity
    - Suggests need to depict how and when data will flow among functions—not just what
  **Example:** How to represent an activity consisting of a set of cases with DFD’s?
    - Dashed arrows represent flow of control information (e.g., signals, commands, status reporting/changing)
    >> This is redundant with arrows in Module Charts
- This, in turn, anticipates new view represented using the third type of chart
Statecharts

- Extension of basic notion of FSM
- FSM's are effective in modeling systems that are
  - clearly and accurately modelled as being in only one of a
    finite number of states at a time
  - considered to move from state to state driven by events drawn from a finite set of possibilities
- Statecharts add some features to what basic FSM’s can represent
  - Hierarchy:
    - Keeps charts from getting too big, hard to understand
  - ANDing and ORing of states:
    - to model simultaneously being in >1 state
    - example: elevator in moving/not or doors_open/not
  - Elaborate specification of transition conditions
- Correlation with Activity Charts helps comprehensibility

Add Activities and Actions

- Activities
  - Associated with a state
  - Start when the state is entered
  - Take time to complete
  - Interruptible
- Actions
  - Associated with a transition
  - Take an insignificant amount of time to complete
  - Non-interruptible

Activities and Guards in Statecharts

- Activities
  - An activity can also send an event
- Transitions
  - A transition may have a guard conditions as well as an event specified
  - Transitions can also specify an action that happens in response to the receipt of an event
Statechart

- Initialize
  - do: Initialize course object
  - do: Assign professor to course

Statechart with Nested States

- Initialize
  - superstate
  - substate
  - Register

- Open
  - entry: Register a student
  - Unassigned
  - do: Assign professor to course

- Closed
  - do: Report course is full

- Canceled
  - do: Send cancellation notices

State Sequence Chart View

- addStudent / numStudents = 0

- RegistrationComplete
  - do: Generate class roster

- Add student / numStudents = 0

- RegistrationClosed
  - numStudents = 10

- RegistrationClosed
  - numStudents > = 3

- RegistrationClosed
  - numStudents < 3

- cancelCourse

Message Sequence Chart View

- Very much like what we have seen before
- Vertical red lines augment diagram with timing information
  - Simultaneous activities
  - Specification of time lag between messages
Statemate Support Environment

- Tools to support drawing/changing diagrams
- Tools to support input of textual information through forms/templates
- Diagrams enhanced by use of color (?)
- Tools to generate simulations automatically—support “stepping through” the system
- System assures consistency among the diagrams—changes automatically depicted consistently in all diagrams
- Tools to automatically generate Ada code that emulate Statechart behavior

Template Input

Cross-Checking/Redundancy Checking
Discrete Event Simulation Too

- Does not seem to scale all that well
  - Hierarchy depicted by nesting all on one 2-dimensional surface
- Data still treated as secondary
- Focus still on functionality
  - Other characteristics and views are worth thinking about too:
    >> Speed
    >> Implementation approaches and issues
    >> ...

Statemate Weaknesses

UML (Unified Modeling Language): The Latest (?)

- Merger of Booch, Rumbaugh, Jacobsen work
  - "The three amigos"
    - All worked for Rational (now IBM)
- Comprehensive suite of diagrams
- Some semantics in place
  - But not all
  - International task forces (!) working on this
- Process for using them was developed too
  - Rational Unified Process (RUP)
- UML blew away the opposition
  - Not clear this was good
(Some) UML representations

- Class Diagrams
- Use Cases
- Sequence Diagrams
- Package Diagrams
- State Diagrams
- Activity Diagrams
- Collaboration Diagrams
- Deployment Diagrams

Different combinations used by Different users for different projects

Major UML Problems/Objections

- What are semantics of all of these features of all of these diagrams?
  - Task forces working on them
  - Maybe there is just too much there (?)
- Diagram semantics overlap
  - Which diagram to use when
  - How to tell when they are inconsistent
- Extensibility
  - Use of "stereotype" feature
  - How to reconcile semantics of new features with existing ones

UML Tries to cover everything

- A diagram type for everything
- But they are not well connected to each other
- Few rules on what to use when
- Long reach with uncertain grasp

Evaluation of Diagrammatic Approach

- Pictures considerably aid clarity
- Significantly reduce possible ambiguity
- Increasingly strong semantics of increasingly intricate pictures yield increasing completeness and increasing assurance of consistency
- Increasingly intricate pictures are decreasingly clear, decreasingly modifiable
  - Modern approach is to provide tools to help
- In place of one intricate and complex diagram, many systems substitute a set of coordinated diagrams, each of which is relatively simple (e.g., Statemate)
  - Leads to problems in assuring consistency of diagrams, but tools can help here too

BUT ALSO:
- Most diagrams help depict functionality, but not other characteristics, (e.g., data, process, etc.)