CMPE 276
Software Engineering

Lecture 4
UML

Projects

- Paper repository
  - Jason Reimr, Ian Brown
- Delta-V
  - Song Kim, Guaheng Ge, Kai Pan
- Servlet configuration manager
  - Rita Garcia
- Program for motion control in animation
  - Mark Slater
- Lego Mindstorms / Pocket PC networking
  - Venkatesh Rajendran, Vaibhav Bhadani

Assignment - due Oct 10

- Everybody:
  - Read Harel: Statecharts: A Visual Formalism for Complex Systems
  - Read B.P. Douglass: UML Statecharts
  (Both available on the class web page)

Assignment - due Oct 10

- Tools track:
  - Write use cases and UML Statecharts for:
    - Car locking logic
  - VCR
  - Submission:
    - Pdf format preferred
    - Can be created with any editor, or drawing tool (UML drawing tool, xfig, powerpoint, ...)
    - Email as an attachment to luca @ see

Assignment - due Oct 10

- Tools track: Car locking logic
  - Think at the events that are most relevant:
    - key_in_door, key_out_door
    - key_in_ignition, key_starter, key_run
    - user_light_ons, user_light_offs (input)
    - switch_light_ons, switch_light_offs
    - cabin_light_ons, cabin_light_offs
    - is_window_open, is_window_closed
    - *
    - and above all, ring_door_ding!!
  - Design the logic of your dream car

Assignment - due Oct 10

- Tools track: VCR
  - Think at the various states (typing, programming, ...)
  - Be sure you specify what appears on the display, and what the machine is doing
  - Can you make one that is easy to program?
Assignments - due Oct 10

• Project track:
  - Prepare presentation on project requirements and specifications (the report will be due on Oct 15)
  - Discussion will take place

  Social Sciences 2
  Room 14
  6-7:45pm (or until done)
  October 10

Projects

• What are the projects?
• Who is on each project?
• Let’s make a decision now.

Modeling

• Describing a system at a high level of abstraction
  - A model of the system
  - Used for requirements and specification
  - Many notations over time
  - State machines
  - Entity-relationship diagrams
  - Dataflow diagrams
  - ... see last lecture ...
  - UML

UML

• UML stands for
  Unified Modeling Language
  - Design by committee
    - Many interest groups participating
    - Everyone wants their favorite approach to be “in”
  - Resulting design is huge
  - Many features
  - Many loosely unrelated styles under one roof
  - Could also be called
  Union of all Modeling Languages

This Lecture

• We discuss
  - Use Case Diagrams
  - Class Diagrams
  - Sequence Diagrams
  - Activity Diagrams
  - State Diagrams

• This is a subset of UML
  - But probably the most used subset

Running Example: Automatic Train

• Consider an unmanned people-mover
  - Also as in many airports

  - Train
    - Moved on a circular track
    - Visits each of two stations in turn
    - Each station has a “request” button
    - To stop at this station
    - Each train has three “request” buttons
    - To stop at a particular station
Use-Cases

- Describe functionality from the user's perspective
- One (or more) use-cases per kind of user
  - May be many kinds in a complex system
- Use-cases capture requirements

An Example Use-Case in UML

- **Event-flow**
  - Passenger presses request button
  - Train arrives and stops at platform
  - Doors open
  - Passenger steps into train
  - Doors close
  - Passenger presses request button for final stop
  - ...
  - Doors open at final stop
  - Passenger exits train

Use Case Diagram

- **Graph showing**
  - Actors
  - Use case
    - Edge to actor if that actor is involved in that case
  - Actors
    - Stick figures
  - Use cases
    - Oscillate

Exceptional Situations

- Some use cases are unusual
  - I.e., error situations
- UML has a special notation
  - The "extend" relationship
  - Nothing to do with OO extension/inheritance
  - These are just rare cases
    - May be nearly unrelated to normal cases

Extension

- Dotted arrow pointing to "normal" case
Summary of Use Cases

- Use Case Diagram
  - Shows all actors, use cases, relationships
- 5 parts to each use case
  - Name, Actors, Entry/Exit Conditions, Event Flow
  - Actors are agents external to the system
    - E.g., users
  - Event flow are sequence of steps
    - In English

Class Diagrams

- Describe classes
  - In the OO sense
- Each box is a class
  - List fields
  - List methods
- The more detail, the more like a design it became

Class Diagrams: Relationships

- Many different kinds of edges to show different relationships between classes
- Mention just a couple

Associations

- Capture n-m relationship
  - Subsumes ER diagrams
- Label endpoints of edge with cardinalities
  - Use * for arbitrary

Aggregation

- Show contains a relationships
  - Station and Train classes can contain their respective buttons
- Denoted by open diamond on the 'contains' side

Generalization

- Inheritance between classes
  - Denoted by open triangle

- One request button per station; each train has three request buttons
Sequence Diagrams

- A table
  - Columns are classed or actors
  - Rows are time steps
  - Entries show control/data flow
    - Method invocations
    - Important changes in state

Example Sequence Diagram

- Method invocation
  - Note: These are all synchronous method calls. There are other kinds of invocations.

Example Sequence Diagram

- Invocation lifetime spans lifetimes of all nested invocations

Example Sequence Diagram

- "Lifelines" fill in time between invocations

Sequence Diagrams Notes

- Sequence diagrams
  - Refine use cases
    - Gives view of dynamic behavior of classes
  - Class diagrams give static class structure

- Not orthogonal to other diagrams
  - Overlapping functionality
  - True of all UML diagrams
Activity Diagrams

- Reincarnation of flow charts
  - Uses flowchart symbols

- Emphasis on control-flow

- Two useful flowchart extensions
  - Hierarchy
    - A node may be an activity diagram
  - Swim lanes

Example Activity Diagram

Example Activity Diagram

Example Activity Diagram

Another Example Activity Diagram

StateCharts

- Hierarchical finite automata
  - Invented by David Harel, 1983

- Specify automata with many states compactly

- Complications in meaning of transitions
  - What it means to enter/exit a compound state
Example Simple StateChart

StateChart for the Train

- A train can be
  - At a station
  - Between stations

- Pending requests are subset of \( \{A, B\} \)

- 16 possible states
  - Transitions: pushA, pushB, departA, departB, ...

StateChart for Buttons + Train

StateChart for Buttons + Train

Opinions about UML: What’s Good

- A common language
  - Makes it easier to share requirements, specs, designs

- Visual syntax is useful to a point
  - A picture is worth 1000 words
  - For the non-technical, easier to grasp simple diagrams than simple pseudo-code

- To the extent UML is precise, forces clarity
  - Much better than natural language

- Commercial tool support
  - Something natural language could never have

Opinions On UML: What’s Bad

- Hodge-podge of ideas
  - Lines of most popular modeling languages
  - Sublanguages remain largely unintegrated
  - Verbose, over-designed.

- Visual syntax does not scale well
  - Many details are hard to depict visually
  - Ad hoc text attached to diagrams
  - No visualization advantage for large diagrams
  - 1000 pictures are very hard to understand

- Semantics is not completely clear
  - Some parts of UML underspecified, inconsistent
  - Plans to fix
UML is Happening

- UML is being widely adopted
  - By users
  - By tool vendors
  - By programmers

- A step forward
  - Seems useful
  - First standard for high-levels of software process
  - Expect further evolution, development of UML

- Design interchange
  - XML (using XML to describe UML designs)