UML Sequence Diagrams for Process Views

CSE 403, Spring 2008, Alverson
With some material from MartyStepp lectures, Wi07.
Outline

- UML class diagrams – recap
- UML sequence diagrams
- UML wrapup

More detail:
- http://dn.codegear.com/article/31863#sequence-diagrams
UML classes

- class name
- attributes - all data fields of the object
  - visibility name : type
- operations – omit trivial, inherited methods
  - visibility name (parameters) : return_type
- visibility:  + public
  - protected
  - private
- underline static values

CSE 403, Spring 2008, Alverson
Class relationships

- generalization – inheritance between classes

- association – connection between classes
  - dependency
    - Solid line, or dotted if temporary dependency
  - aggregation
    - class contains another class
  - composition variation
    - contained class will not exist without the container class
  - multiplicity, navigability
Leap Year Table

Calendar

Executive Calendar

Calendar Entry

Conference Room

Relationship exercise

Generalization
Association
Aggregation
Composition
**UML Sequence Diagrams**

**sequence diagram:**
- details how operations are carried out -- what messages are sent and when

- capture the *process view* of an architecture – provide a *dynamic* view of behavior

- organized according to *time* - time progresses as you go down the page

- objects are listed from left to right, based on when they take part in the *message sequence*
MVP Sequence diagram
How do you start?

1. Identify the process/algorithm/activity you want to capture (may be a use case)

2. Identify the major objects involved

3. Map out the flow of control/messages to achieve the result
Representing objects

InstanceName : ClassName

Alverson : Instructor

anonymous object

: Instructor

object of unknown class

Alverson

CSE 403, Spring 2008, Alverson
Messages between objects

- message (method call) indicated by horizontal arrow to other object
  - with message name and arguments above arrow

```
method name
requestClassGrades(ClassID)
```

Alverson : Instructor

method argument types

CSE 403, Spring 2008, Alverson
More on messages

- message indicated by horizontal arrow
  - dashed arrow back indicates return
  - different arrowheads for normal / concurrent (asynchronous) methods

Levy : DepartmentHead

Alverson : Instructor

requestClassGrades(ClassID)

Grades

giveRaise(BigNumber)

Levy

Alverson

type of return value

return (dashed line)

synchronous (full arrow)

asynchronous (half arrow)
Indicating method calls

- **activation**: thick box over object's life line; drawn when object's method is on the stack
  - either that object is running its code, or
  - it is on the stack waiting for another object's method to finish
Lifetime of objects

- **creation**: an arrow with 'new' written above it
  - an object created after the start of the sequence appears lower than the others

- **deletion**: an X at bottom of object's lifeline
  - how do objects get deleted in Java? in C?

CSE 403, Spring 2008, Alverson
Conditionals and loops (UML2)
Example sequence diagram #1

UML1 style of iteration and conditionals

If a room is available for each day of the stay, make a reservation and send a confirmation.
Example sequence diagram #2

- StoreFront
- Cart
- Inventory

- Loop
- AddItem
- ReserveItem
- Checkout
- ConfirmOrder
- ProcessOrder
- PlaceItemInOrder

Object calls itself
Using visio to create a SD

- Create a sequence diagram to represent a skier booking a lesson

- Objects: Skier, Booking System, Calendar
Forms of system control

- What can you say about the control flow of each of the following systems?
  - Is it centralized?
  - Is it distributed?
  - Does the sequence diagram help show this?
What control pattern?
What control pattern?
What’s wrong with this SD?

- Look at the UML syntax and the viability of the scenario
What about with this one?
Why not just code it?

- Sequence diagrams can be somewhat close to the code level. So why not just code up that algorithm rather than drawing it as a sequence diagram?
Why not just code it?

- Sequence diagrams can be somewhat close to the code level. So why not just code up that algorithm rather than drawing it as a sequence diagram?

  - A good sequence diagram is an abstraction
  - Sequence diagrams are language-agnostic
  - Non-developers can do sequence diagrams
  - Can see many objects/classes at a time on same page (visual bandwidth), enabling
    - Easier understanding
    - Easier review for correctness
  - Good communication medium
UML closing thoughts

- What’s good about UML?
  - A common language
    - makes it easier to share requirements, specs, designs
  - Visual syntax is good
    - summarizes information
    - good for non developers/less technical
  - Tool support is available
    - Visio, Violet, Rational, Eclipse (June 2007), …
    - Some tools convert from UML to code
UML closing thoughts

- What’s not so good?
  - Rich language (good and bad)
  - Visual syntax does not always work or scale
    - Features hard to depict
    - Large diagrams would be required, which are hard to understand

- UML is happening!
  - UML is widely known by users, tool vendors, developers, customers
  - Seems a step forward – a standard language for representing software architecture and design