Discussion Slides

Deeper A1

- Why might a use case description (e.g. on a user story card) be inadequate as a requirements specification?
  - *It might not enumerate all of the various cases that must be handled.*
  - *It doesn't describe distinct steps of the process or the over-arching user-level metaphors.*
  - *We might be automating an existing system, and it is important that the new process be very similar to the old one.*

Deeper A2

- What is the difference between a behavioral requirement and a behavioral specification?
  - *Simple answers are "what" vs. "how", and "external behavior" vs. "internal implementation".*
  - *In practice, requirements and specifications are the points on a continuous path of progressive refinement … and it may be difficult to tell when one turns into the other.*

Deeper B1

- Why is a graphical design language interesting?
  - *Any language can be evaluated on the basis of range, ease of expression and precision of expression.*
  - *While graphical languages are (in general) less precise and powerful than textual ones, the human mind can often absorb structures and relationships much more easily and quickly from a visual representation than a textual one.*

Deeper B2

- How is UML different from other graphical flow-charting & design systems of the last 50 years?
  - *It supports precise and complete high level descriptions of objects and their relationships.*
  - *Its wide range of sub-languages permit meaningful description of many more aspects of system behavior and structure.*
  - *The sub-languages are related in ways that enable easy integration of multiple models.*

Deeper B3

- Why is it useful to standardize a graphical design language?
  - *People can more quickly understand a language they already know.*
  - *Designs captured in a standardized language can be more easily shared and reused.*
  - *A standardized language can express much more nuanced concepts (compare AMSLAN with charades).*
  - *A large population will attract tool developers.*

Deeper B_4

- What distinguishes a sketching language from a specification language?
  - *A sketching language can quickly and easily describe general concepts.*
  - *A specification language can precisely describe objects and their behavior.*
- What distinguishes a specification language from a programming language?
  - *A programming language can completely specify data types and flow control.*

Deeper B_5

- How does a textual representation enable the development of CAD tools?
  - *A standard textual representation permits a design to be easily passed between tools.*
  - *A graphical design tool can output the textual representation for the created program.*
  - *Browsers, type-checkers, interpreters and other tools can parse the textual representation as a means of importing the model on which they will.*

Useful Concepts C_1

- What is a synchronous interaction
  - *One where the sender waits until the recipient finishes processing the request. Remote procedure calls are the most common form of a synchronous interaction.*
- What is an asynchronous interaction
  - *One where the sender sends a message to the recipient, and then continues processing with no further interest in what the recipient might decide to do about the message.*

Useful Concepts C_2

- What is meant by thread creation and destruction?
  - *This is a popular concept in client/server computing. It refers to an independent thread of execution (something like a separately executing program) that is created just to process a request, and ends as soon as the request completes.*
  - *There are actually a great many different thread models. Take my OS class if you would like to learn more about these.*

Deeper C_3

- What is the difference between descriptive and prescriptive diagrams.
  - *A descriptive diagram illustrates what a system does or would do. Diagrams used in brain-storming sessions or user documentation are descriptive.*
  - *A prescriptive diagram illustrates what a system is required to do. Specifications in requirements documents are prescriptive.*

Deeper C_4

- Why are UML interaction diagrams poorly suited for expressing algorithms?
  - *Basic UML interaction diagram do not seem to have a reasonable way of describing decisions or iteration.*
  - *Actually, UML does define ways of representing iteration and decision in an interaction diagram … but it is painfully awkward.*
  - *This problem is solved by UML activity diagrams.*

Deeper

- Why do the action boxes in an activity diagram have rounded corners?
  - *In UML, a box with squared corners is a "thing" (a class, object, or component)*

Deeper

- Why do activity diagrams begin (and end) with a circle?
  - *In UML, a circle represents an interface, which means an activity diagram can be substituted into an object/interface diagram.*

Deeper

- Why would we want to show the name of a routine or method name inside of a general action description?
  - *To more clearly specify the way the action is to be implemented, or how objects/methods will be used in a proposed design.*

Deeper

- Are UML activity diagrams powerful enough to fully express an algorithm?
  - *Given that actions could include assignments, and that loops can be implemented with IF statements, this notation might be up to the task.*
  - *I question, however, whether this language could offer any significant advantages in ease, power, or precision of expression over a traditional programming language.*

Deeper

- Why would you use a swim-lane vs. an interaction diagram?
  - *To show not only interactions, but the computations that drive them.*
- Why would you use a swim-lane vs. an activity diagram?
  - *If the algorithm of interest would be executed in distinct objects/threads/hosts.*
  - *To show the interactions of the algorithms running in the different objects/threads.*

Deeper

- When would we choose a state diagram rather than an activity diagram?
  - *If we expected to implement the algorithm as a state machine.*
  - *Even systems that are implemented with normal code, may have several interesting and distinct operational states or modes. Diagramming those states and the events that trigger transitions between them may be a very useful way to view the system's dynamics.*

Deeper

- Why is it awkward to describe a data flow model in UML?
  - *UML was designed to describe OO systems.*
  - *It is a language for describing hierarchies of objects, algorithm to implement their methods, and messages to mediate their interactions.*
  - *Data flow models follow data streams through a succession of transformations and routings.*
  - *These represent very different perspectives.*

Answer

- Choose the model that is best suited to the aspects of the system we are trying to describe:
  - user/system or inter-component interactions
  - algorithms or distributed computations
  - system state transitions
  - data transformations
- You don't have to choose one
  - You can use different diagrams to describe different aspects of system behavior