Deeper

- Why is it useful to identify different sub-classes of users?
  - *They may need the product to provide different functions.*
  - *They are likely to use the system to accomplish different goals.*
  - *They are likely to have different world-models and experience.*
  - *We may want to present different interfaces to different types of users.*

Deeper

- How might role based classification be useful?
  - *There may, in fact, be fundamentally different types of users, who will use the product in different ways.*
- Why might role based classification be invalid?
  - *Some roles may distinguish momentary activities rather than people (e.g. shopper vs. purchaser).*

Deeper

- Why might knowledge based classifications be important?
  - *If knowledge differences between users give rise to different models of the system, we might want to show each the expected model.*
- Why might knowledge differences not greatly affect scenarios?
  - *If the difference in knowledge is just a matter of experience we might provide the same basic scenarios, but with optional help/guidance for trainees.*

Deeper

- How are "softer" user characterizations like Rouse's useful?
  - *They establish "fundamental goals" … which are probably more important than "sample use cases".*
  - *They give us a richer model of the user, whom we are trying to satisfy.*
  - *They do give us usage scenarios, and goal- (rather than task-) oriented use-cases.*

Deeper

- Why is a good use case easier to get than a good declarative requirement?
  - *It is more natural for people to describe what they do, than an imaginary system.*
- Why are use cases easier to review than declarative requirements?
  - *Use cases are stories, and users can easily see how well the story matches their own experience.*

Deeper

- How might a use case more honestly capture requirements than declarative statements?
  - *Because the real requirements may actually be to facilitate the specified use cases.*
  - *Decoupling the requirements from the use cases adds opportunities for translation errors and/or the introduction of new elements that are not necessarily implied by those use cases.*

Deeper

- What is good about a prototype as an interface specification?

  Potential users can review and comment on it.

- What is bad about a prototype as an interface specification?

  It goes well beyond requirements, perhaps specifying an almost complete implementation.

Deeper

- Why should user stories be kept to one 3x5 card?

  To prevent calling large projects one feature.

  To force us to distill what we want to simple, clear and concise statement.

  To prevent us from calling it a complete specification (those come from users).

  To give us a useful management token.

  … but the truth is that we will associate task descriptions and test cases with them.

Deeper

- Why are detailed behavioral expectations not included in User Story cards?

  (a) This information is expensive to develop, and is not needed for planning purposes.

  (b) This information is evolved between the developer and user, and becomes the basis for the acceptance test suite. There is no need for any intermediate representation.

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 (and revealing) way to view the system's dynamics.

Deeper

- How would you generate a list of different types of users for a product?

  Some roles are obvious.

  You can ask someone with domain expertise if there are different sub-classes of potential users.

  You could interview potential users, ask them what they do, and learn about different roles from them.

Deeper

- 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 <u>much</u> more easily and quickly from a visual representation than a textual one.

Deeper

- Why are use cases:
  - concise … *simple list of steps*
  - real-world … *scenario has a purpose*
  - what, not how … *story only tells "what"*
  - user world-view . *from user perspective*

Deeper

- 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

- 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

- When does it makes sense to develop use cases by observing or interviewing potential users?
  - *When the product is intended to assist, enhance, or extend an existing process, users of the current process can provide good descriptions of how they use it.*

Deeper

- 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

- When does it makes sense to develop use cases by brain storming?
  - *If the whole point of the new product is to enable new usage scenarios, there may be no people who are familiar with them.*
  - *When the new product use will be very different from existing processes, there are limits to what we can expect from users of the existing products.*

Deeper

- If existing users don't know how they will use a product they've never seen, why should we interview them?
  - *To understand the needs to which our new product will respond.*
  - *To understand the world-view into which our new product may need to integrate.*
  - *To understand what users of products find to be convenient and inconvenient.*

Deeper

- If use case diagrams do not describe the interactions, what are they useful for?
  - *They are the basis for discussions of the basic sub-classes of users, and the way that each will use the system.*
  - *They provide a good high level overview of the system's expected capabilities.*
  - *They could be a table of contents for more detailed behavioral descriptions of each interaction.*

Deeper

- What does it mean for one use case to extend another?
  - *A more complex use case might be a superset of the steps of a simpler (and previously defined) use case.*
  - *Variations on a common underlying mechanism might front-end the simpler use case.*
  - *This both shows those relationships, and simplifies the overall use-case collection.*

Deeper

- How would it affect understandability if it was necessary to represent a system in multiple distinct use case diagrams?
  - *If they were truly distinct (different actions on different types of objects) separating them out to separate diagrams would probably make the system easier to understand. Packages can be used to for hierarchical groupings.*
  - *Otherwise, one larger diagram would probably make more sense than breaking things arbitrarily into smaller diagrams.*

Deeper

- Would it make sense to use both UML use case diagrams and XP user story cards?
  - *Each story card might show up as a use case in the use case diagram.*
  - *The use case diagram would provide an overview of the capabilities in a graphical and more structured (broken down by user types) way that is much easier to grasp.*