



[www.primaryview.org](http://www.primaryview.org)

*from the notebooks of John Artim*

# Use Cases and Software Requirements Specification

## User Interface and Use Case Authoring

John M. Artim, ©2003, 2004. Permission to copy, modify, or distribute is granted.

# Acknowledgements

- Feel free to use this material, in whole or part, for your own training needs but please indicate that you based your work, with permission, on a copyrighted work by John M. Artim that is available from [www.primaryview.org](http://www.primaryview.org).
- A Microsoft® PowerPoint file of this notebook is available by request. Send an email to John Artim ([jartim@uistyle.com](mailto:jartim@uistyle.com)) requesting the Use Case Basics Notebook PowerPoint file in the Use Cases and Software Requirements Specification series. Please include your name, affiliation, and your intended use of the material.

# Goals of this Notebook

---

- Discuss usability and user interface (UI) issues in the context of requirements specification
- Introduce specification of user interface and usability requirements in the context of use cases

**Browsing Time for this notebook is about 1 hour**



# Background for this Notebook

---

- You should have a basic understanding of use case modeling such as is outlined in the *Use Case Basics* notebook
- You should have a basic understanding of concept modeling as is outlined in the *Concept Modeling for Analysis and Specification* notebook
- You should have a basic understanding of usability as is outlined in the *Usability* notebook



# Major Topics in this Notebook

---

- How does usability show up in requirements specification?
- What constitutes good user interface?
- How is good user interface specified?

# Usability

- Usability—the user’s ease of task completion
  - Applies to hammers, business processes, and GUIs
- Usability can be judged by positive criteria:
  - Which design works better?
  - Is a design built on good/appropriate design patterns?
- Or by negative criteria:
  - Does this design exhibit usability flaws?
    - Known poor design practice
  - Does this design exhibit usability problems?
    - Observed user difficulties



# Goal of Usability Practice

---

- Eliminate all severe usability problems before a product is used for real work!
- Reduce or eliminate moderate usability problems as quickly as possible
- Reduce minor usability problems over time
- But most problems should be caught during requirements specification or in design!

# First Get Requirements

- For each use case:
  - What must the user see
    - For a given Cargo Manifest, display the Shipments covered by the Cargo Manifest, the Containers associated with each Shipment, and the Cargo packed in each Container
    - For a given Container, display a list of the seals on that Container, where each seal is identified by the Owner of the seal, an ID number, if available, the kind of seal (manual or RFID), the state of the seal (separate lists for manual and RFID), and date seal placed
- ***Get rid of all scenario language that implies a UI design!***

# Presentation Requirements

- Ideally, all presentation requirements are lumped into presentation objects in the scenarios
  - All of the entities, attributes, and associations that the user needs to complete each task should be summarized in a presentation object in each scenario
  - See the *Sequence Diagrams for Use Case Authoring* notebook

# Then Aggregate Use Cases

- Many use cases often require display of the same set of entities and relationships
  - Before aggregating make sure the differences in display requirements are compatible
  - Where possible combine in sensible ways
  - Above all, make sure the set of entities and relationships is manageable in size
    - This is always context specific
    - Too small, big, or just right depends on how much the user normally keeps juggled in his or her head
    - Computerized systems can extend this limit but not usually by orders of magnitude

# Then Comes the High-Level UI Design

- For each user-task use case
  - Pull all user display requirements in one place
    - These are the requirements for what the user must see and how that information should be structured
  - Focus on the use cases forming a single *Unit-of-Work*
  - Determine which UI Design Patterns are appropriate to assisting the user complete these use cases
  - Combine the patterns supporting the *Unit-of-Work*
    - This will form one application window or browser page
    - Possibly with one or more dialogs or supporting pages

# Unit-of-Work

- The work done by an end-user in a single session
- Units-of-Work can sometimes interrupt one-another; keep them separate!
- Example: When taking an order a customer gives a service representative an address change
  - These are two separate units-of-work
    - *Take an Order*
    - *Edit Customer Address*
- **Note:** a Unit-of-Work is distinct from the many database transactions that typically support it.

# UI Design Pattern

- A description of a UI design approach and the kind of task (use case) it supports
- The UI design approach can be described
  - Abstractly and very generically
    - But this is mostly for a technical UI audience
  - Concretely and by example
    - This is more useful to a wide audience
- See the companion notebook: *Basic UI Design Patterns*

# Conclusions

- The UI should form a representation of the user's task that enables them to efficiently complete that task
- Don't focus on low-level tasks with many disconnected bits of UI
  - This forces the user to keep too much detail in their head
- Use the UI to organize the domain for the user
  - The UI should guide the user to solutions
  - If the SUD can completely automate a task, then let it
  - But if the SUD can't complete the task, don't let the SUD force the user into unnatural acts
  - Let the user's expertise drive task completion