

From Use Cases to UI Specification

- How do I leap from Use Case to UI?
- Design-neutral use cases
- How to prototype a use case
 - Macro level organization techniques
 - Navigation paradigms
 - Exercise and revise
- The UI Functional Specification
 - Application Overview & Navigation Diagram
 - Page Layout & Description
 - Panel Map
 - Load / Close Events
 - UI Controls
 - Panel Properties
 - Panel Data Elements & Data Access Logic
- How the UI Specification is consumed



About your Speaker

David Ruble

David Ruble is a principal at Olympic Consulting Group, a software architecture and development firm. He is an analyst, designer, author and educator who is widely regarded as an expert in the field of information modeling, functional specification and GUI design. He has been a lead analyst and designer of many mission-critical corporate information systems as well as applications in the health and public safety sectors.

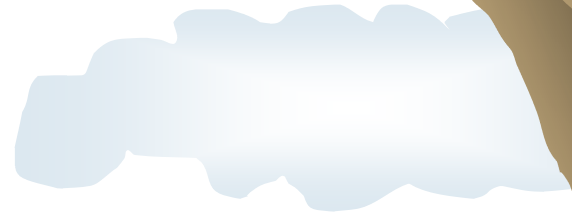
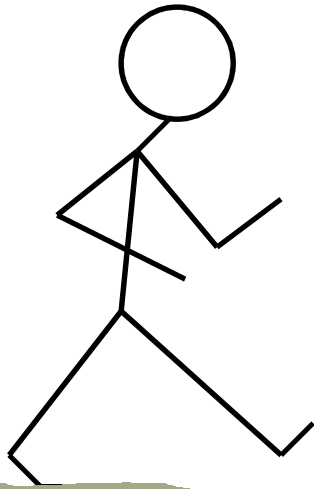
As an educator, he has taught software engineering techniques to hundreds of students throughout the United States. He is the author of, *Practical Analysis & Design for Client/server & GUI Systems*, (Prentice-Hall 1997). His book is used widely in colleges and universities throughout the United States and Thailand, Mexico and Argentina and he enjoys favorable reviews from as far away as Bosnia.



© 2006 – Olympic Consulting Group, all rights reserved.
Olympic Consulting Group
PO Box 4008
Federal Way, WA 98063
206.946.2690
www.ocgworld.com

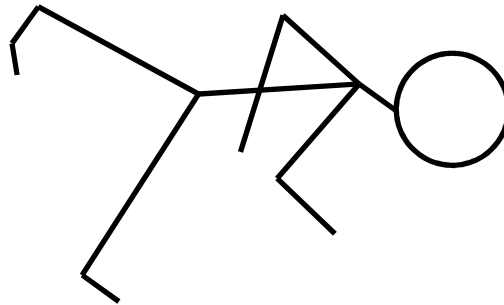


Q: How do I leap from Use Cases to UI design?





A: Beware what use cases ARE and ARE NOT.





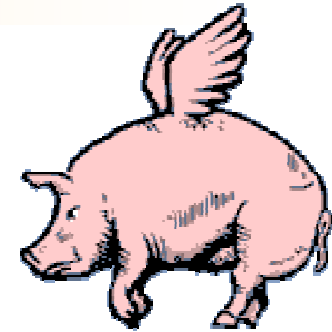
What's a Use Case?

"A use case is a sequence of transactions in a system whose task is to yield a measurable value to an individual actor of the system." [– Jacobson et al., 1995]

- Originally proposed as a method for creating early test scenarios
- Later adopted as a method for documenting requirements
- Promotes system analysis from the “user’s” point of view
- Provides a natural way to partition large systems
- Broadly accepted throughout the software industry



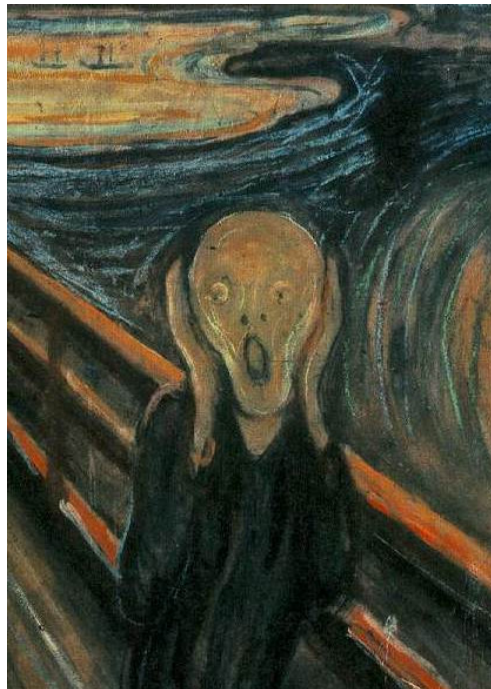
Use Case Challenges



- Highly generalized definition
- The original focus on test scenarios *can* result in:
 - Premature focus on interaction *design* (user/system dialog) rather than essential requirements
 - Long-hand RAD (Rapid Application Development)
 - Omission of UI specification (missing from most CASE tools and methodologies)
 - Highly behavioral and process-centric approach with risky deferral of data requirements definition

The UI designer's nightmare:

“The Use Case starts when the user clicks on the red icon on the left-hand side of the main navigation bar.”





Keep your UI design options OPEN:

- Limit Use Case definition to the “WHAT” not the “HOW”
- Avoid premature design decisions such as navigation and actual screen layout
- Accompany the Use Case Model with a Data Model and State Models
- Design the UI with low-fidelity prototypes
- Confirm your design (with users & developers)
- THEN, commit the design to a UI specification



Example of a hard-wired design in a UC:

1. User logs in to MyBanking
2. User clicks on Apply for new IRA account
3. System displays IRA Account Holder form
4. User fills out IRA Account Holder form and clicks Next
5. System displays Beneficiary form
6. User fills out Beneficiary form and clicks Next
7. System displays IRA Rollover Authorization form
8. User fills out IRA Rollover Authorization form and clicks Next
9. System displays terms and conditions
10. User agrees to terms and conditions and clicks Next
11. System displays Electronic Signature form
12. User completes Electronic Signature form and clicks Submit
13. System confirms New IRA Account Application Confirmation in Printer-Friendly format
14. User may print New IRA Account Application Confirmation to local printer
15. User may log off or continue with other Online Banking offerings



Preconditions help free up the design options

1. User logs in to MyBanking
2. User clicks on Apply for new IRA account
3. System displays IRA Account Holder form
4. User fills out IRA Account Holder form and clicks Next
5. System displays Beneficiary form
6. User fills out Beneficiary form and clicks Next
7. System displays IRA Rollover Authorization form
8. User fills out IRA Rollover Authorization form and clicks Next
9. System displays terms and conditions
10. User agrees to terms and conditions and clicks Next
11. System displays Electronic Signature form
12. User completes Electronic Signature form and clicks Submit
13. System confirms New IRA Account Application Confirmation in Printer-Friendly format
14. User may print New IRA Account Application Confirmation to local printer
15. User may log off or continue with other Online Banking offerings

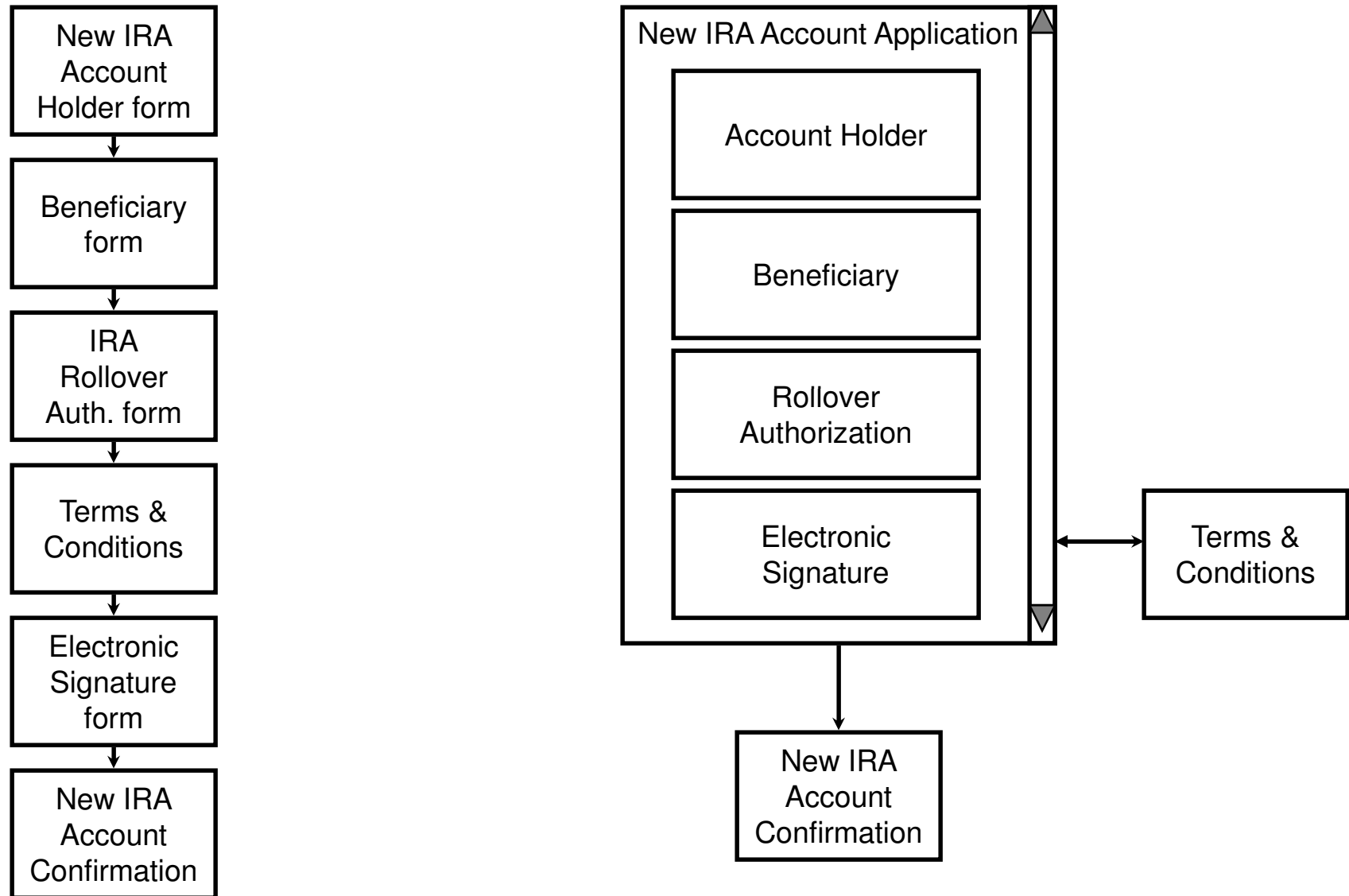
Precondition

Needlessly constrains the UI design. Why not allow the user to make the request, then test for whether logged in?

Post condition

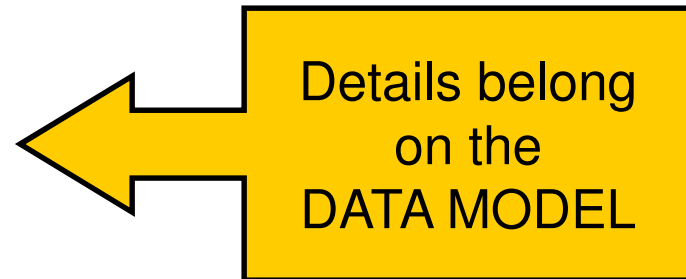


Sequential UI vs. Object-centric UI



Design-neutral version of the UC:

Preconditions	<p>Must be registered user of MyBanking</p> <p>Must be logged in & authenticated</p>
Inputs	<p>New IRA Account Application:</p> <ul style="list-style-type: none"> – Account Holder Profile – Account Beneficiaries – IRA Rollover Authorization – Electronic Signature – Agreement to Terms and Conditions
Process	<p>System displays New IRA Account Application</p> <p>Applicant completes and submits New IRA Account Application</p> <p>Applicant must agree to terms and conditions before application is accepted</p> <p>Systems saves Application and displays Confirmation</p> <p>Applicant may print a copy of the Confirmation</p>
Outputs	<p>Terms & Conditions</p> <p>Confirmation</p>
Post Conditions	<p>Applicant may log off or continue with other Online Banking offerings</p>





How do I get from Use cases to UI?

- Macro level: Evaluate the application's organization across all use cases
- Micro level: For each use case, focus on data arrangement & task flow before worrying about controls and navigation
 - Layout data panels without regard to screen size constraints
- Evaluate navigation options – respecting screen size constraints
- Apply agreed-upon UI quality vectors
- Apply style guide standards to use of controls
- Exercise & revise (e.g., whiteboard usability sessions)
- When firm, complete the UI functional specification



Macro level UI organization

- Based on all use cases for the application (with anticipated room to grow)
- Select the appropriate entry and navigation paradigm
- Consider how the user responds to business events . . .



Actor-centric UI

- Group use cases by actor to partition applications
 - + One stop shopping for each actor
 - Risk of actor's roles changing
- Examples:
 - Dashboards: Summarizes all things a particular actor needs to monitor
 - Role / organization based main menu



Object-centric UI

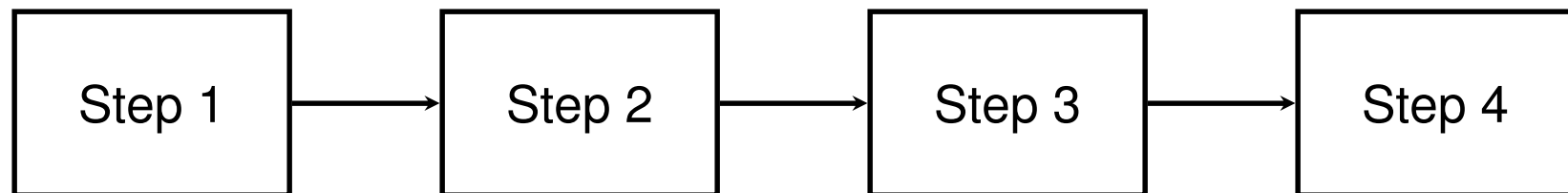
- Group use cases by the primary object on which they act
 - + One stop shopping for each major object (e.g., Order)
 - + Excellent for collaborative work groups
 - Complex user authorization and state management
 - Risk of many users exposed to items not relevant to their role
- The state-transition models declare what options are available at a point in time

Order Number	Order Date	Status	
219540	01.02.2000	New	Cancel
228560	01.15.2000	Vouched	Ship
229751	01.16.2000	Canceled	Activate
231140	02.03.2000	Shipped	Invoice



Sequentially-organized UI

- Sort event-based use cases chronologically to step users through the process
 - + Good for repetitive tasks
 - + More often applied at the micro level (e.g., wizards)
 - Poor for randomized activities





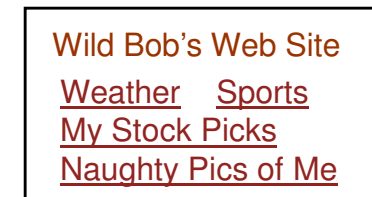
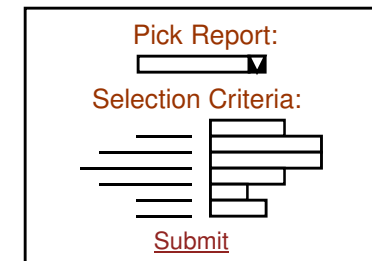
Temporally-organized UI

- Event-based use cases are grouped together *only* because they occur at a similar time
 - + Rarely used, but excellent for time-sensitive tasks
 - + Most often found in month-end activities
 - Not as flexible of an organization



Lower levels of event/use case grouping paradigms

- Logic-sharing: Mostly unrelated items grouped together to share code
 - Often done in the spirit of “reusability” but usually results in a mess
- Coincidental: “Only the designer knows for sure”
 - No apparent rationale for why the ability to execute use cases are aggregated together



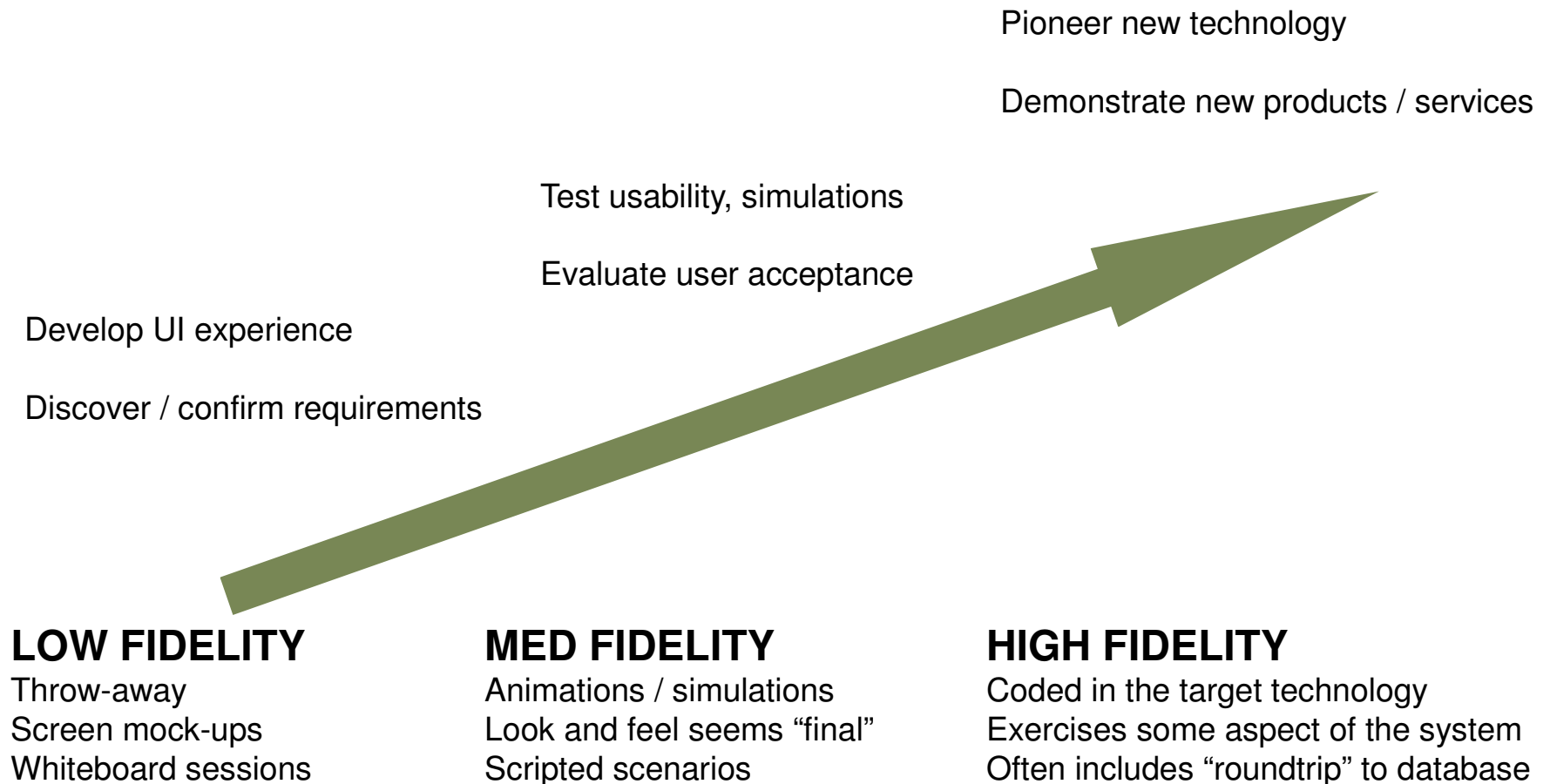


What is a Prototype?

- A facsimile of the real-thing that is not so fully functional as to pass for the final product
- A model created to demonstrate or learn about some specific aspects of the final product
- Prototyping is conducted to meet very *specific learning objectives*



The learning objectives drive the complexity of the prototype



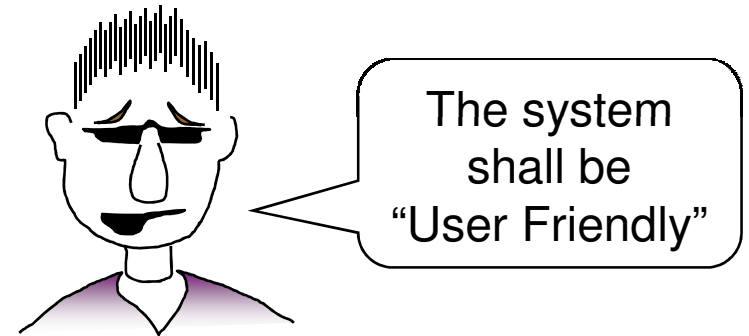


UI Prototyping elaborates on the use cases by taking into account:

- Target technology
 - Strengths
 - E.g., browser = zero-maintenance client, universal access
 - Limitations
 - E.g., bandwidth, client-side editing
- User profile
 - Assumed subject area knowledge
 - Prior computing experience
 - Expected level of training required
 - Language
 - Age
 - Level of education, literacy
 - Special considerations (e.g., color blindness, hearing-impaired)
- Style Guide standards from your shop

Let's ban the phrase: "User Friendly"

- What does "user friendly mean?"
 - Conservation of keystrokes
 - Comprehensive assistance
 - As few screens as possible
 - Speed of task completion
 - Rapidness of response
 - Etc...
- To achieve one, you may sacrifice another
- If left un-defined, each person defaults to their own opinion





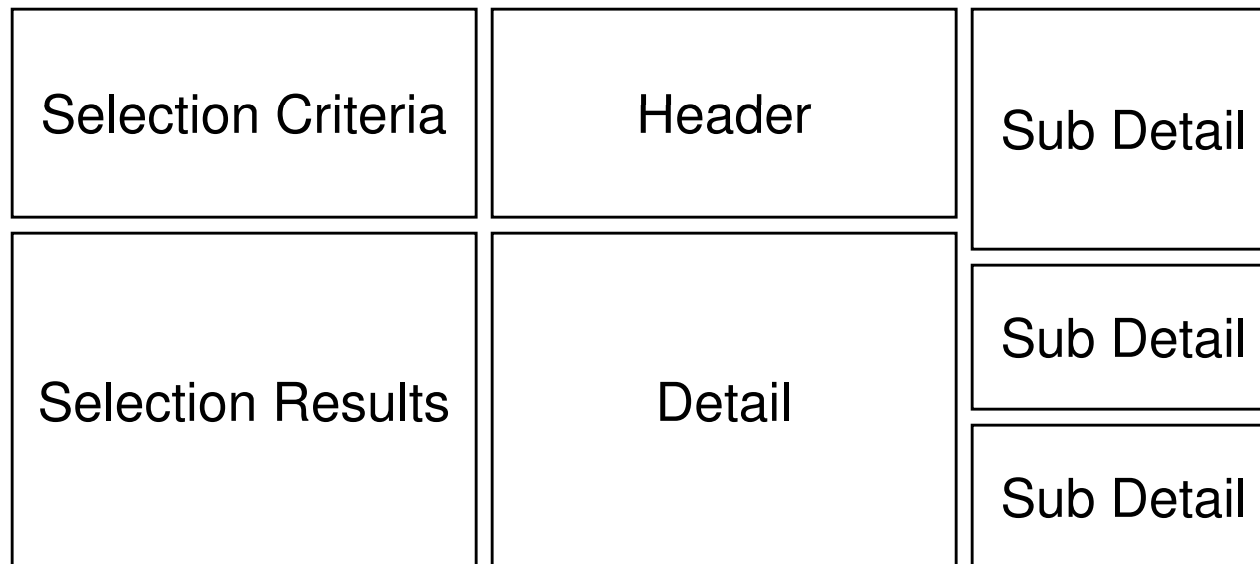
Style guide

- Menus – location, size, color
- Controls – naming, placement, size, color, enablement
- Fonts, size, style
- Labeling – alignment, placement, colon use
- Reserved words for menus
- Graphics – color, logos, images, resolution, fills, group boxes



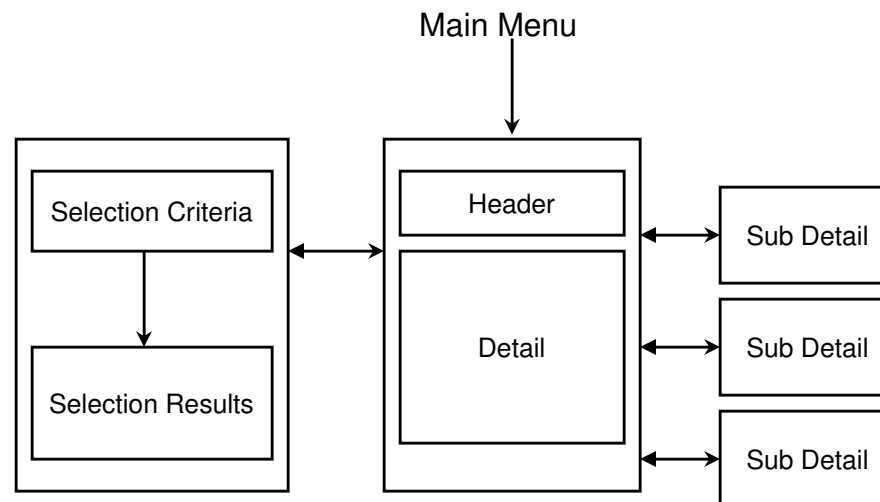
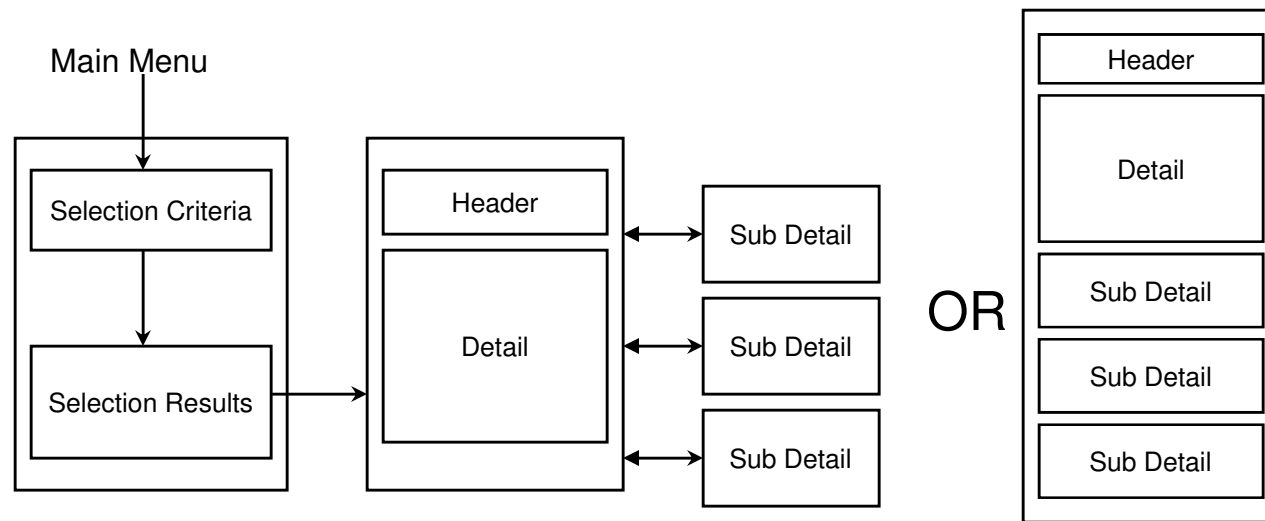
Prototyping a use case

- Layout the data panels for the use case's inputs and outputs – without worrying about navigation





Consider different navigational paradigms & page organizations



Exercise and Revise

- Story boarding and walkthroughs are excellent techniques to validate prototypes
- Keep on the alert for:
 - Verbal narrative not captured in any document
(write it down!)
 - Changes to relationships between data elements
(write it down!)
 - Discussions of heretofore undiscovered events
(write them down and check scope)





UI Functional Specification

For On-Shore AND Off-Shore Development

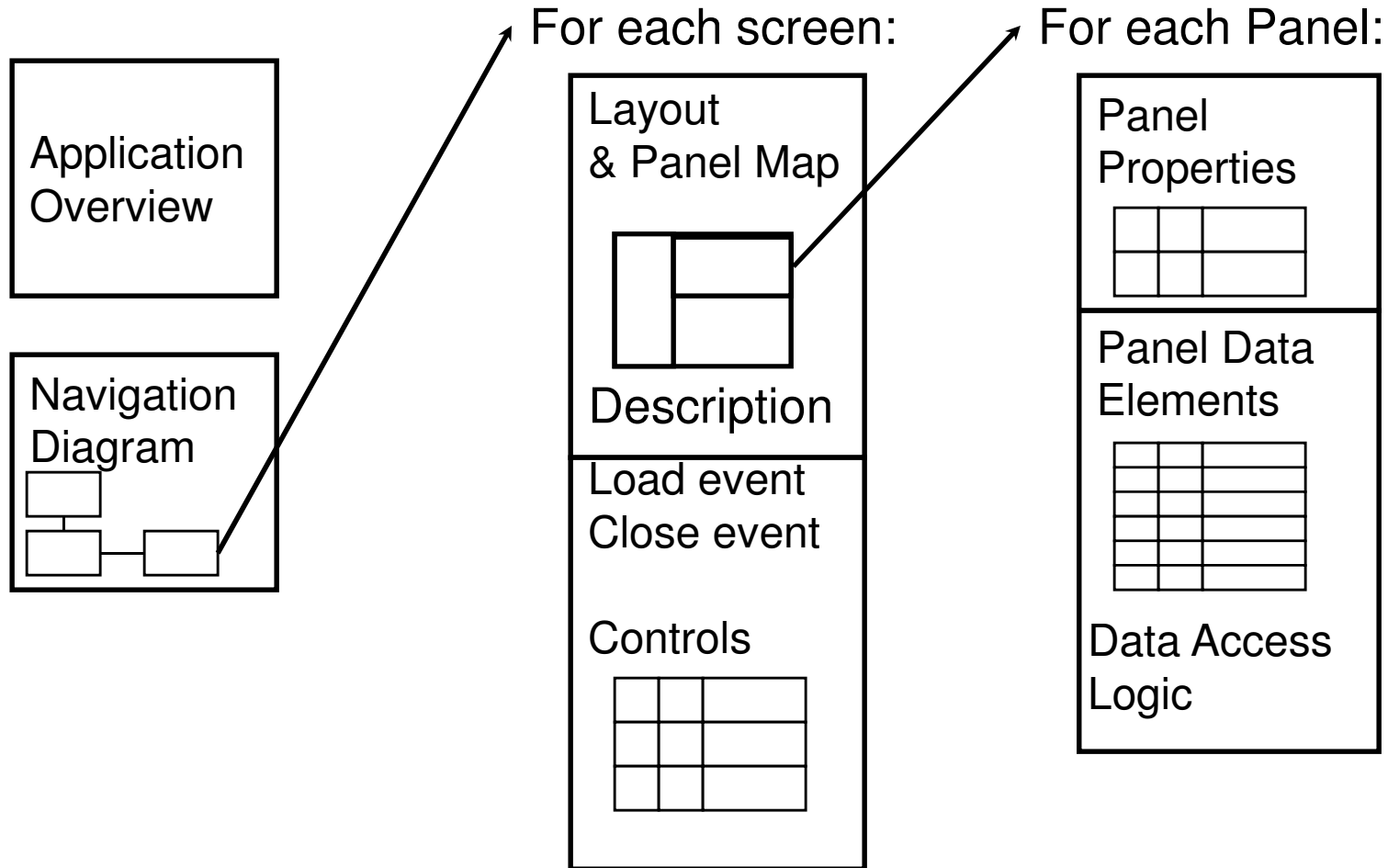


The Purpose of the UI Specification

- Blueprint for development
- Defines all elements of the externally-observable application
- Fully consumes the interface prototype – which represents the use cases
- References the Use Case model – for internal processing requirements
- Provides a rich lode of detail for devising test cases
- Survives construction as an as-built specification
- **CRITICAL FOR OFF-SHORE DEVELOPMENT!!!**



UI Functional Specification Contents





Application Overview

- Describes the purpose of the individual application in language a user can understand
- Orients the reader to the business events managed by the application
- Lists the specific use cases referenced within the spec

Application Overview

The SwapMeat.com application allows consumers to buy, sell or trade frozen food within their trading region.

SwapMeat.com was started by Horga Snowtread, homemaker from Ballard, WA, who was frustrated by the amount of food spoiling in her freezer. People who got free beef with the purchase of their tires can swap it with their neighbors for, say, halibut or chicken tenders.

The SwapMeat application is accessed from the home page by clicking on the SwapNow tab. Buyers and Sellers must first register before participating in a transaction. There is no fee to buy. A nominal charge to the seller's credit card is processed at the time of the sale . . .

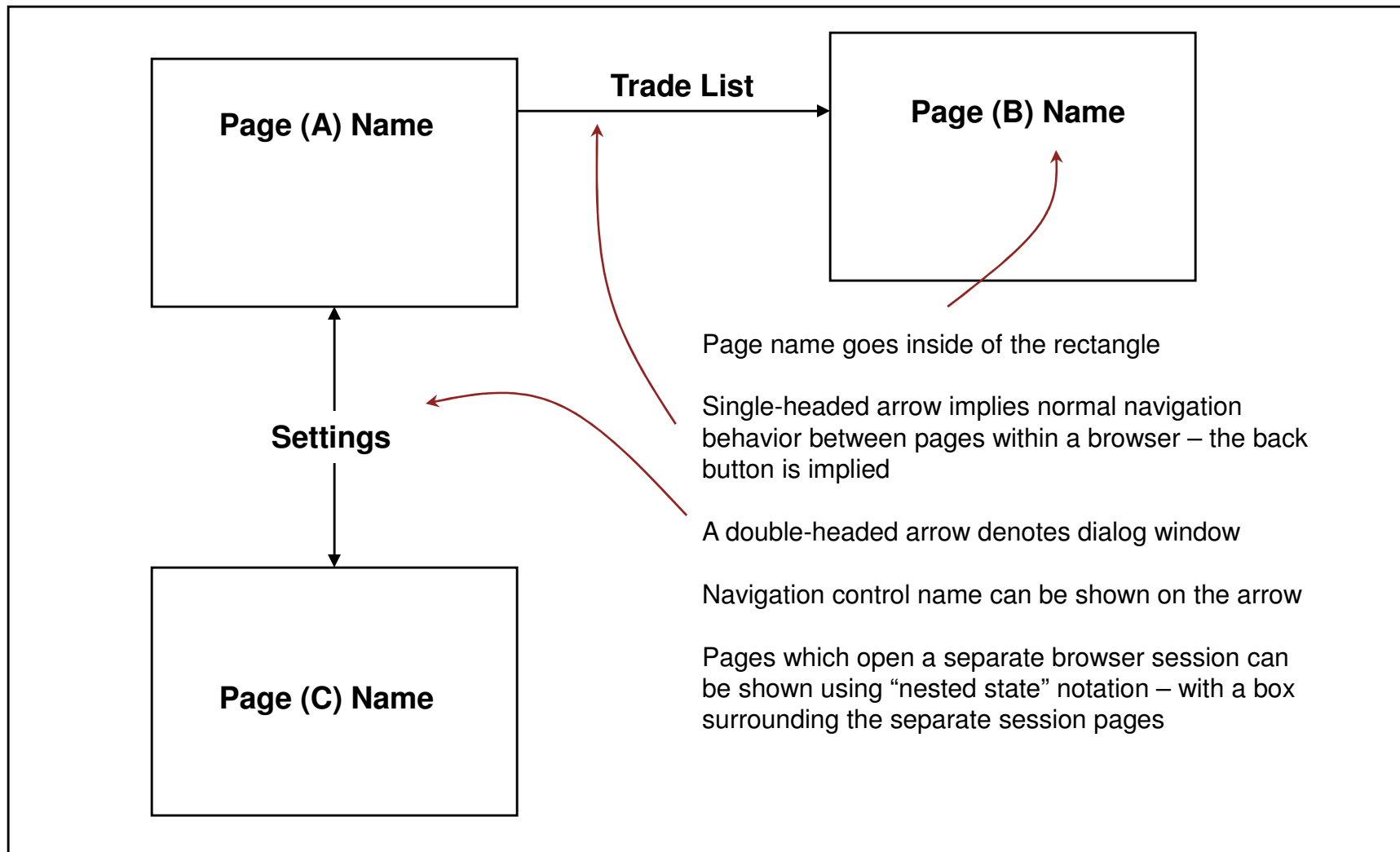


Navigation Diagram

- A form of prototyping
- Planning and communication tool
- Used to model navigation between pages within an application
- Can be restricted to show navigation for specific events



Navigation Diagramming Notation





Page Layout

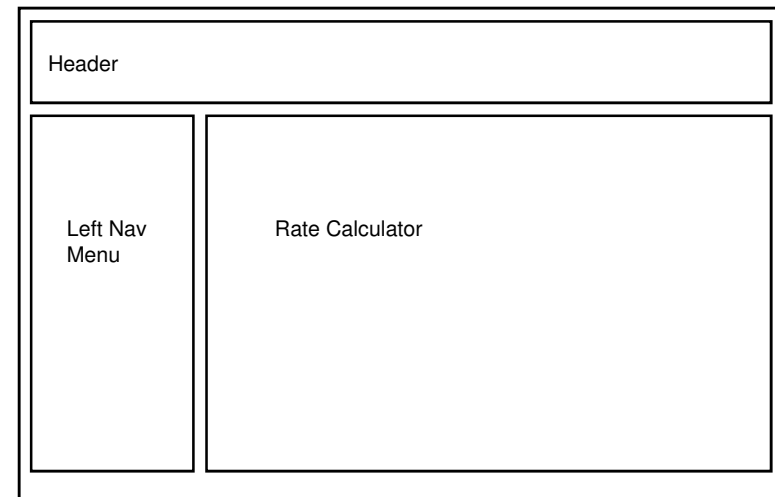
- Screen shot, mock-up or prototype representation of how the page should look
- OK to eliminate redundant features such as main menus, banners
- Paste together multiple shots if the windows scrolls down
- Can be replaced or augmented with actual screen shot after development

The screenshot shows the DHL website's Rate Calculator interface. The top navigation bar includes links for Ship, Track, Transit Times, Rates, Pickups, Locations, Supplies, and Billing. Below this, a secondary bar contains Rate Calculator, Rate Guide, Zone Charts, and Zone Maps. The left sidebar features a menu with options like LOGIN, REGISTER, Customer Service, Domestic Services, International Services, Small Business Center, Technology Tools, and About DHL. The main content area is titled 'Rate Calculator' and includes a note: 'Required fields are marked with an asterisk (*)'. The form is divided into three sections: 'Origin Location' (Country: United States Of America, Origin Postal Code: DR), 'Destination Location' (Country: United States Of America, Destination Postal Code, Service Type: DHL Next Day 10:30 am, Residential Delivery: No), and 'Package Information' (Number of Pieces: 1, Actual Weight, Dimensions: Length, Width, Height in inches, Estimated Monthly Packages, Shipment Date: Apr 19 2005).



Panel Map

- Wire frame drawing which divides the page into panels
- Panels are display areas which differ by type, or subject matter
- Panels can be reused on multiple pages
- Panel name establishes an unambiguous way to refer to sections of the screen



Panel Types

- Free form: single record
- Grid or table: multi-record
(can be dynamic, fixed)
- Graph
- Tree (e.g., Explorer)
- Image
- Motion Picture
- Copy

Current customer? Log in here

Phone number:

example: 0051236543

-Or-

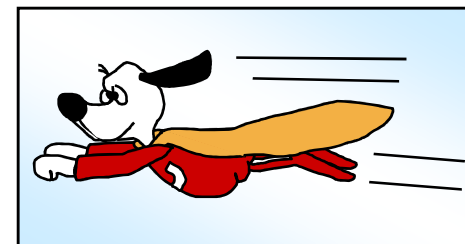
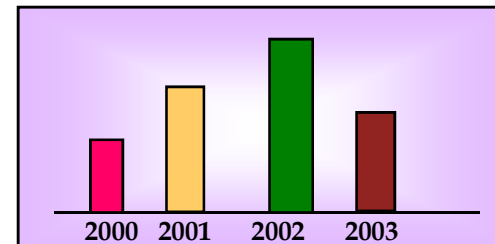
Username:

-And-

Password: [Forgot Password?](#)

Log in Remember me

TITLE	ARTIST	LISTEN	BUY
Smile	Jessica Simpson	<input type="checkbox"/>	<input type="button" value="Buy"/>
Fallen	Evanesence	<input type="checkbox"/>	<input type="button" value="Buy"/>
The College Dropout	Kanye West	<input type="checkbox"/>	<input type="button" value="Buy"/>
When The Sun Goes Down	Kenny Chesney	<input type="checkbox"/>	<input type="button" value="Buy"/>
Speakerboxxx/The Love Below	OutKast	<input type="checkbox"/>	<input type="button" value="Buy"/>
Clear	Josh Groban	<input type="checkbox"/>	<input type="button" value="Buy"/>
The Very Best Of Steryli Crow	Steryli Crow	<input type="checkbox"/>	<input type="button" value="Buy"/>
Kamikaze	Twista	<input type="checkbox"/>	<input type="button" value="Buy"/>



Warning: General Sturgeon has determined that fishing can be hazardous to fish.



Page Description

- Clear, concise text describing the use of the page
- Mixed audience = users, developers, testers, trainers

DON'T START FROM SCRATCH!

- Compile descriptions which already exist
 - Interface prototype accompanying text
 - Use case descriptions
- Then add
 - Page behavior
 - Control, link and menu item descriptions



Load Event / Close Event

- Load: Specifies any special processing or behavior when the page initially loads (e.g., setting default values, displaying messages)
- Close: Specifies any special processing when the page closes (e.g., prompt for unsaved changes)

Event	Description
Load	If user not logged in Display log in panel Else Display my_account panel Endif
Close	If unsaved changes Prompt "Do you want to save your work before closing?" End if



UI Controls

- All controls (links, buttons, menu items) on a page are listed in one table
- Defines enabled state and user authorization rules
- Specifies what happens when clicked (may reference a Use Case specification for further detail)

Label	Enabled	Authorization	Clicked
Approve	If order = unapproved	Manager only	Confirm action Set order status = approved
Deny	If order = unapproved	Manager only	Confirm action Set order status = approved
Details	If focus on single line item	All	Open pop-up window: <i>Line item detail description</i>
Close	Always	All	Close window



Panel Properties

- Defines the characteristics of the panel
- Different panel types have varying properties
 - Tables and Grid panels may have: sorting, grouping, suppression of repeating values, column ordering, dynamic vs. static rules

Panel Name	Accessories
Panel Type	Table
Sorting	By category, by product name, ascending
Grouping	Group by category
Supression	Supress repeating category name



Panel Data Elements

- Label (to correlate to screen shot)
- Type (e.g., drop down list box, radio button, text box, copy, jpeg, gif . . .)
- Required vs. optional?
 - >>> Conditions which change optionality
- Visible vs. Hidden?
 - >>> Conditions which change visibility

- User editable?

- >>> Conditions which change editability

- Rules:

- Derivations
 - Valid Ranges
 - Domain values
 - Constraints

Label	Rqrd	Vis	Edit	Rules
Account	Y	Y	N	
Balance	Y	Y	N	Sum of charge column
Payment	N	Y	Y	Must be positive number. May not exceed Balance.
Past Due	N	>>>	N	Visible only if > 0



Data Access Logic

- Informs the developer how to apply input parameters to retrieve the desired result set.
- Pseudo-SQL

```
Input:      order_date  
           customer_ID  
  
Select     Customer_Name  
           Order_Number  
           Order_Status  
           Order_Date  
  
From       Customer  
           Order  
  
Where      Order.Customer_ID = customer_ID  
AND       Order.Order_Date >= order_date
```



How the UI Specification is Consumed

- Design & Construction
 - Used as a blueprint for constructing the interface
- QA
 - Provides detail for constructing test cases for:
 - Page content and layout
 - Page load event / Close event
 - UI control enablement
 - UI control clicked events
 - Panel element properties, edits, behavior and conditions
- On going
 - Used as an “as-built” specification for future enhancements of the application



Summary

- Business Use Cases should be design-neutral to keep your options open

- To convert Use Cases to UI:
 - Macro level: Evaluate the application's organization across all use cases
 - Micro level: For each use case, focus on data arrangement & task flow before worrying about controls and navigation
 - Layout data panels without regard to screen size constraints
 - Evaluate navigation options – respecting screen size constraints
 - Apply agreed-upon UI quality vectors
 - Apply style guide standards to use of controls
 - Exercise & revise (e.g., whiteboard usability sessions)

- When firm, complete the UI functional specification