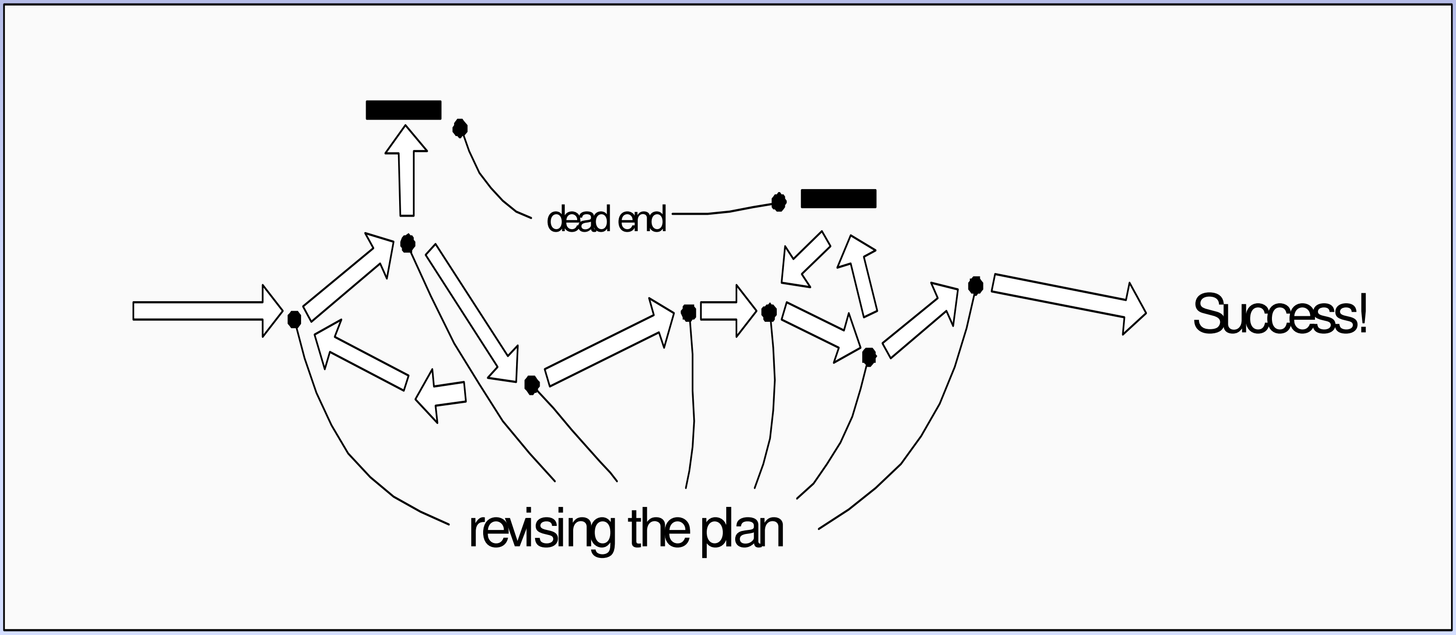
The Art of Telling a Design Story

Rebecca J. Wirfs-Brock
Wirfs-Brock Associates
rebecca@wirfs-brock.com

The Nature of Design



Design is an ongoing process of discovery and invention

A Design Storytelling Strategy

- Tell interesting stories—not every story is worth telling
- Establish scope, depth and tone
- Decide how detailed the story should be
- Choose the appropriate forms—tell it, draw it, describe it
- Present your design so important things get emphasized
- Revise, clarify, compress, and expand as needed

The Story Teller's Dilemma

- You can't avoid forward references.
  - If an object collaborates with another— and you haven't yet read a description of that second object's role and responsibilities— you can only guess at why it is being used.
- Even if you could manage to organize a story so that fundamentals were presented first, it makes for a very dull presentation.

Speaking to A Crowd

- Readers’ interests and backgrounds differ.
  - Some may know a lot and don’t want to be bothered by a review.
  - Others may be looking for specific facts.
  - Others may need some fundamentals before understanding details.
  - Still others may only want the punch line.
- You have to choose what to emphasize and how best to please the crowd while still catering to individuals’ needs.

What’s More Fundamental

- Things you cannot change are more fundamental than things under your control.
- Use cases are more fundamental than sequence diagrams that describe how a system supports them. *Problem descriptions are more fundamental than solutions which are your own creation.*
- A typical execution path is more fundamental than an exceptional one.
- Things are more fundamental than relations between them. Your audience needs to understand objects and their general behavior before they can comprehend structural relationships or interaction details.

Progressive Realization

- Landscape architects create views that purposely reveal things. They move someone to where they want in gradual, interesting steps.
  - When telling a high-level design story, stick to the main points. Tell it as if it were a news flash. Present the central objects and components and what is important about them. Reveal just enough to engage your readers.
  - After explaining typical situations, give your readers options to veer off in one of several directions: to more detailed views, to exceptional conditions, to alternatives.

Putting It All Together

- Stories build to more dramatic conclusions if important parts are first, followed by new material presented in novel ways.
- Be aware of monotony setting in. After seeing a bunch nearly identical drawings, attention wavers.
  - To hold attention, shift your readers' focus by inserting commentary that explicitly calls out some details.
  - Or, point out that the next five diagrams are similar, so all but the most eager reader can skim them in good conscience.

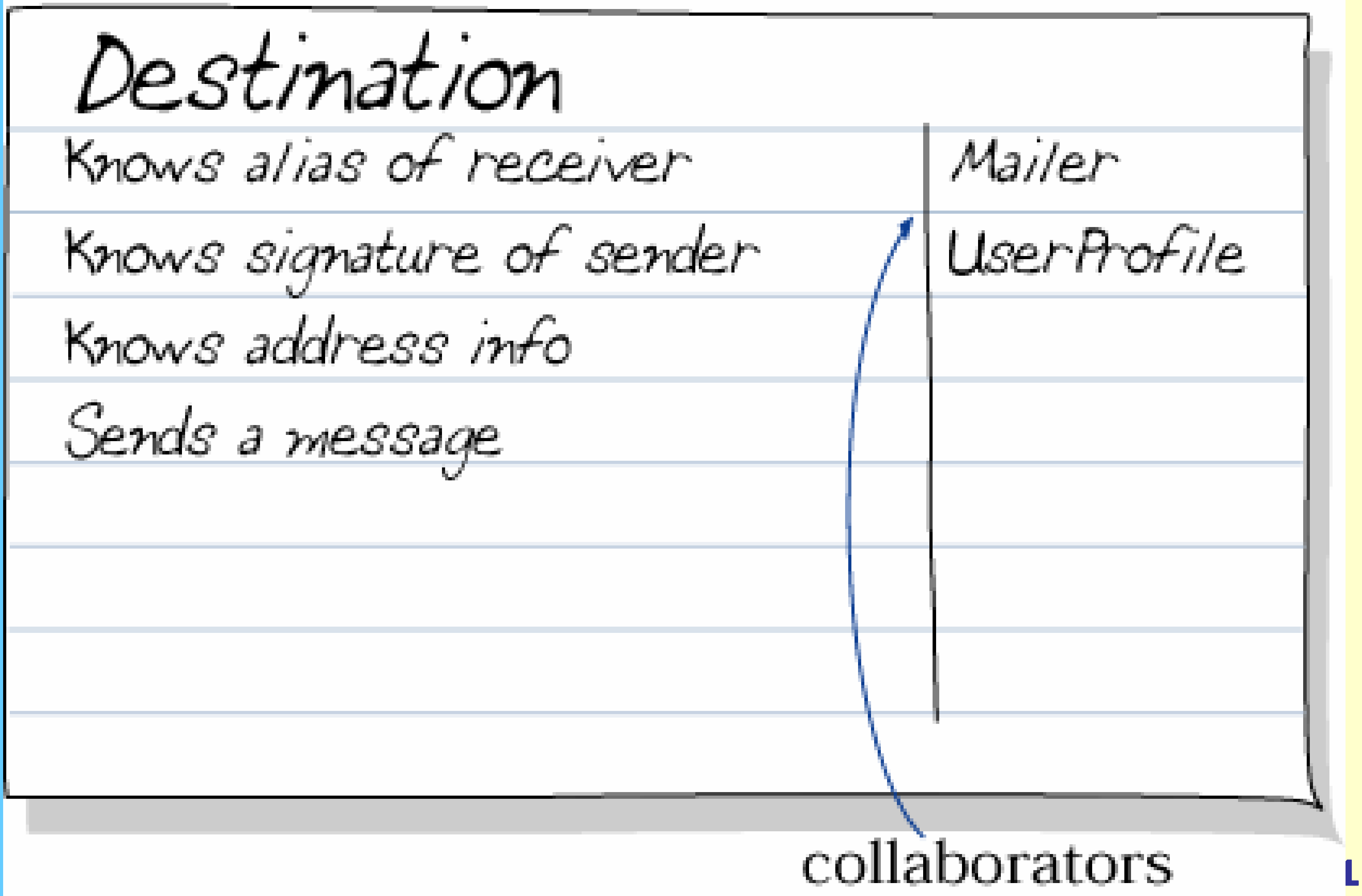
Dealing with an Impatient Audience

- Progressive realization works when your readers want to follow your lead.
  - Those seeking specific facts won’t sit still for too much nonsense. So include a section that answers Frequently Asked Questions. Point those readers there.
  - You needn’t present fundamentals first. Important things that deserve special emphasis should be stated first. Background information can be presented afterwards.
  - Don’t be a defensive storyteller—assume your audience wants to hear what you have to say.

Use Multiple Descriptions

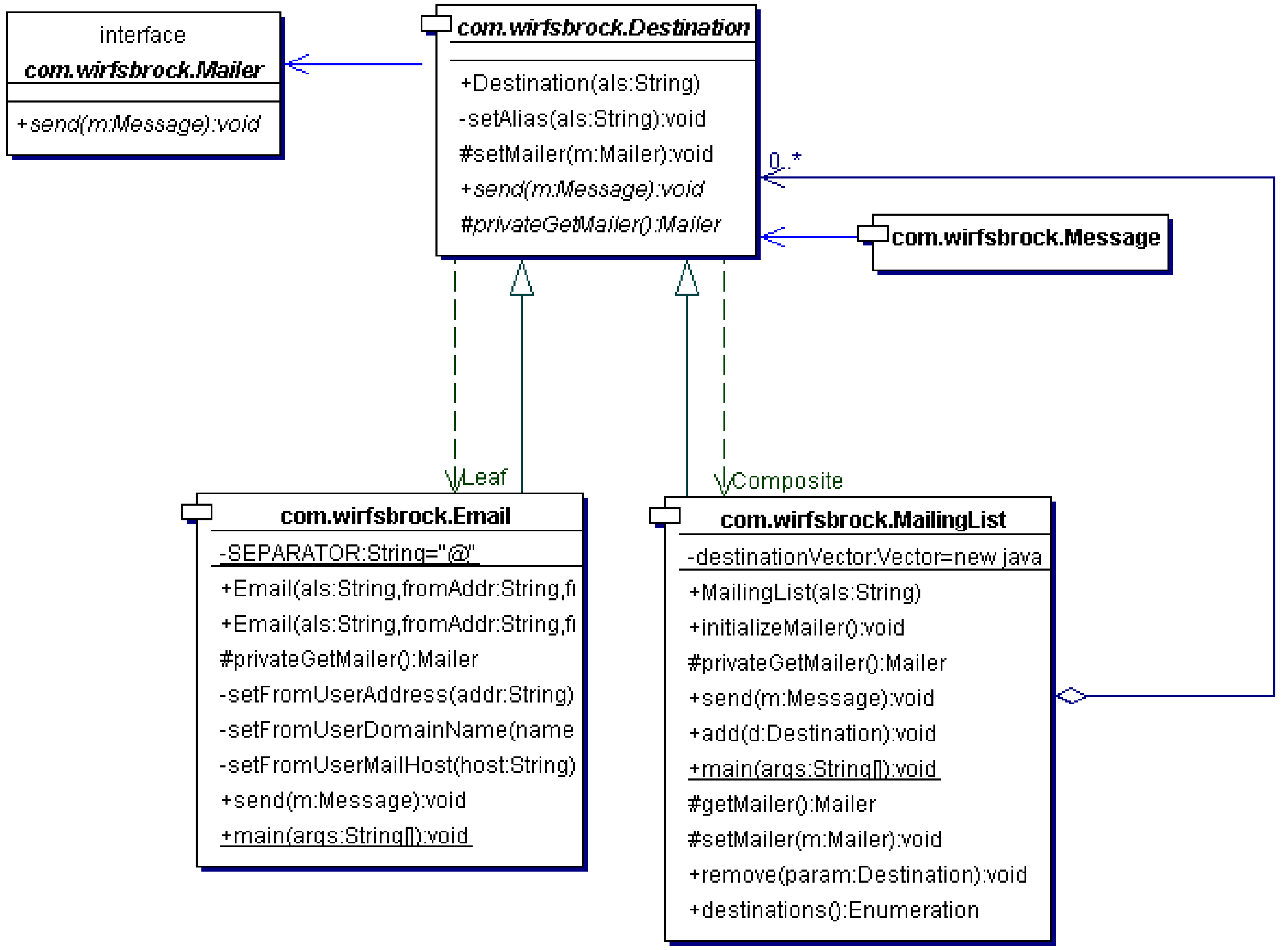
- Get comfortable describing and looking at a design at different levels of abstraction. Describe a design multiple ways:
  - Use CRC cards to set the stage and explain object roles and responsibilities (a conceptual design view)
  - Use class diagrams to show specification level details (attributes, operations, relationships)
  - Draw UML sequence diagrams to show specific interaction sequences
  - Write text to explain the details of objects' behavior during a specific interaction

Candidate-Responsibilities-Collaborators Card



Destination	
Knows alias of receiver	Mailer
Knows signature of sender	UserProfile
Knows address info	
Sends a message	

Class Diagrams



interface
com.wirfsbrock.Mailer

+*send(m:Message):void*

com.wirfsbrock.Destination

+Destination(als:String)
-setAlias(als:String):void
#setMailer(m:Mailer):void
+*send(m:Message):void*
#*privateGetMailer():Mailer*

0..*

com.wirfsbrock.Message

Leaf

Composite

com.wirfsbrock.Email

-SEPARATOR:String="@"

+Email(als:String,fromAddr:String,fi
+Email(als:String,fromAddr:String,fi
#privateGetMailer():Mailer
-setFromUserAddress(addr:String)
-setFromUserDomainName(name
-setFromUserMailHost(host:String)
+send(m:Message):void
+main(args:String[]):void

com.wirfsbrock.MailingList

-destinationVector:Vector=new java

+MailingList(als:String)
+initializeMailer():void
#privateGetMailer():Mailer
+send(m:Message):void
+add(d:Destination):void
+main(args:String[]):void
#getMailer():Mailer
#setMailer(m:Mailer):void
+remove(param:Destination):void
+destinations():Enumeration

Increasing Emphasis

- Things gain prominence by their position and appearance. To increase an item's emphasis:
  - Put it first
  - Highlight it

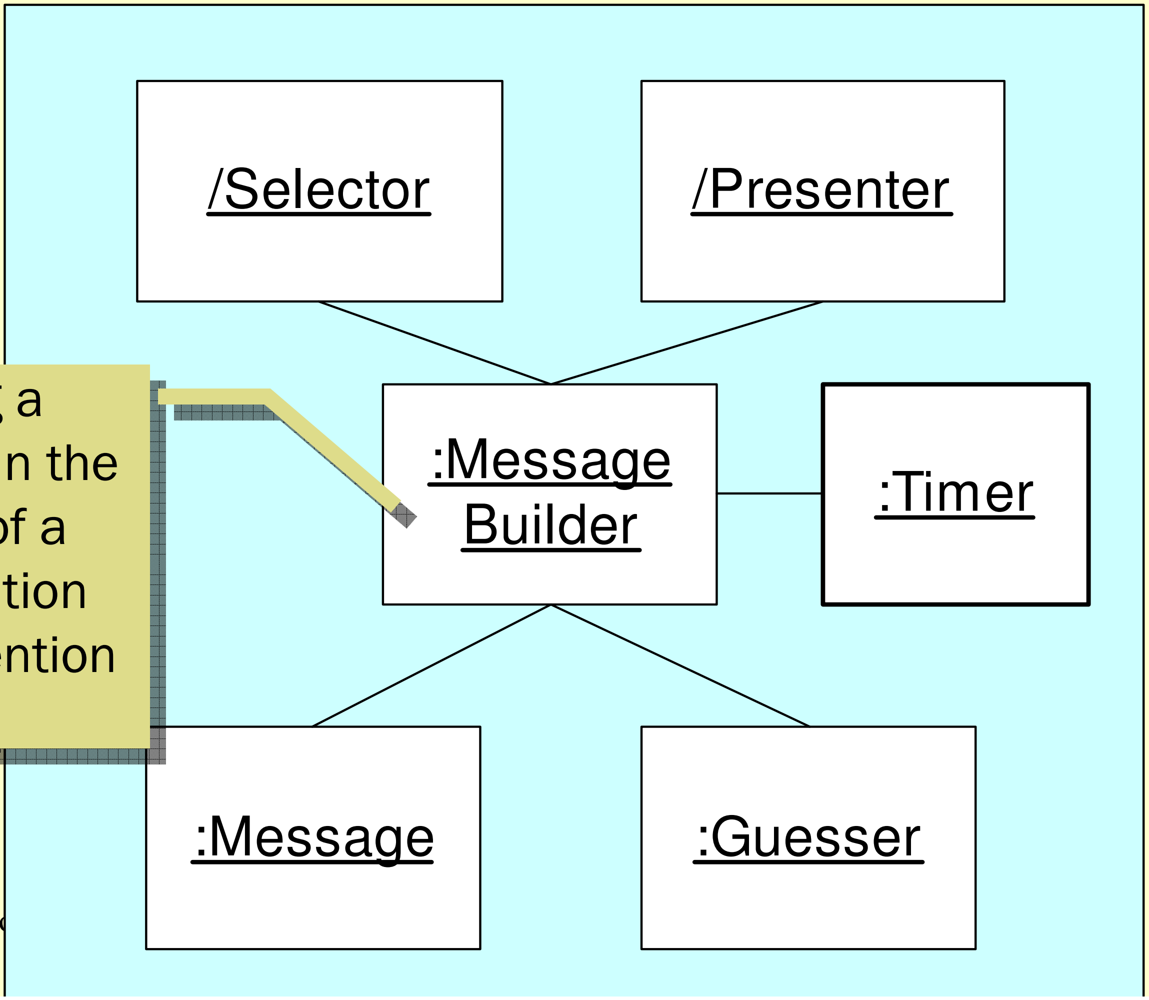
Surround it with space

  - Put it in a bulleted list
  - Mention it in multiple places
  - **Give it more room**
  - Repeat or restate it in different forms

Show Focused Interactions

- Decide what to emphasize!
- Consciously decide to ignore certain details
  - User interface
  - Delegated action
  - Details of an algorithm

Central location, size, and boldness add emphasis

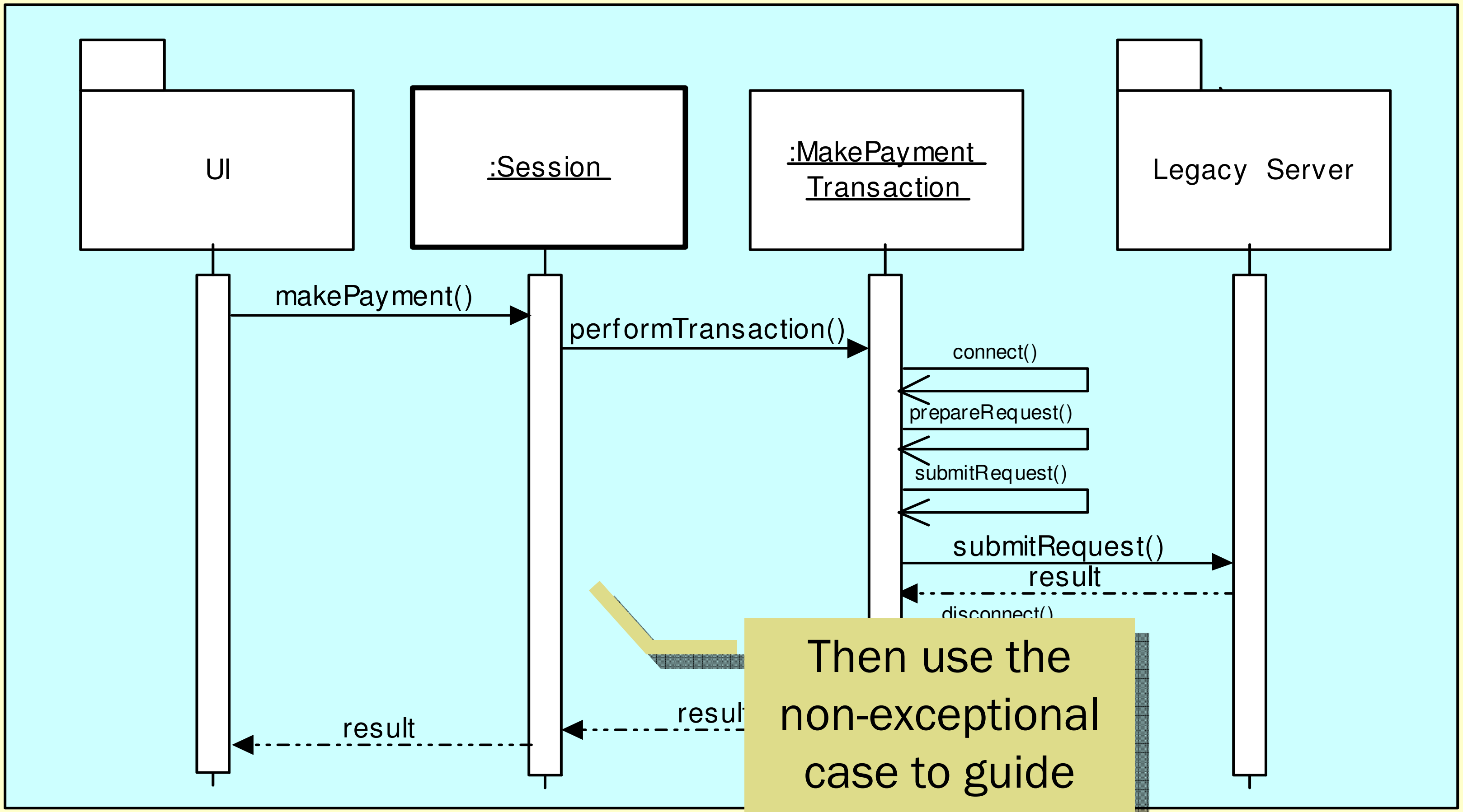


Placing a controller in the middle of a collaboration draws attention to it

Strunk and White's Elements of Style

- *Do not overwrite.* 10 pictures are not worth 10k words.
  - If collaborations are similar, show a typical case first, then note how remaining ones differ.
  - Draw representative interactions.

Illustrate The Non-Exceptional Case



Then use the non-exceptional case to guide your consideration of exceptional cases.

Then Explain And Document Policies

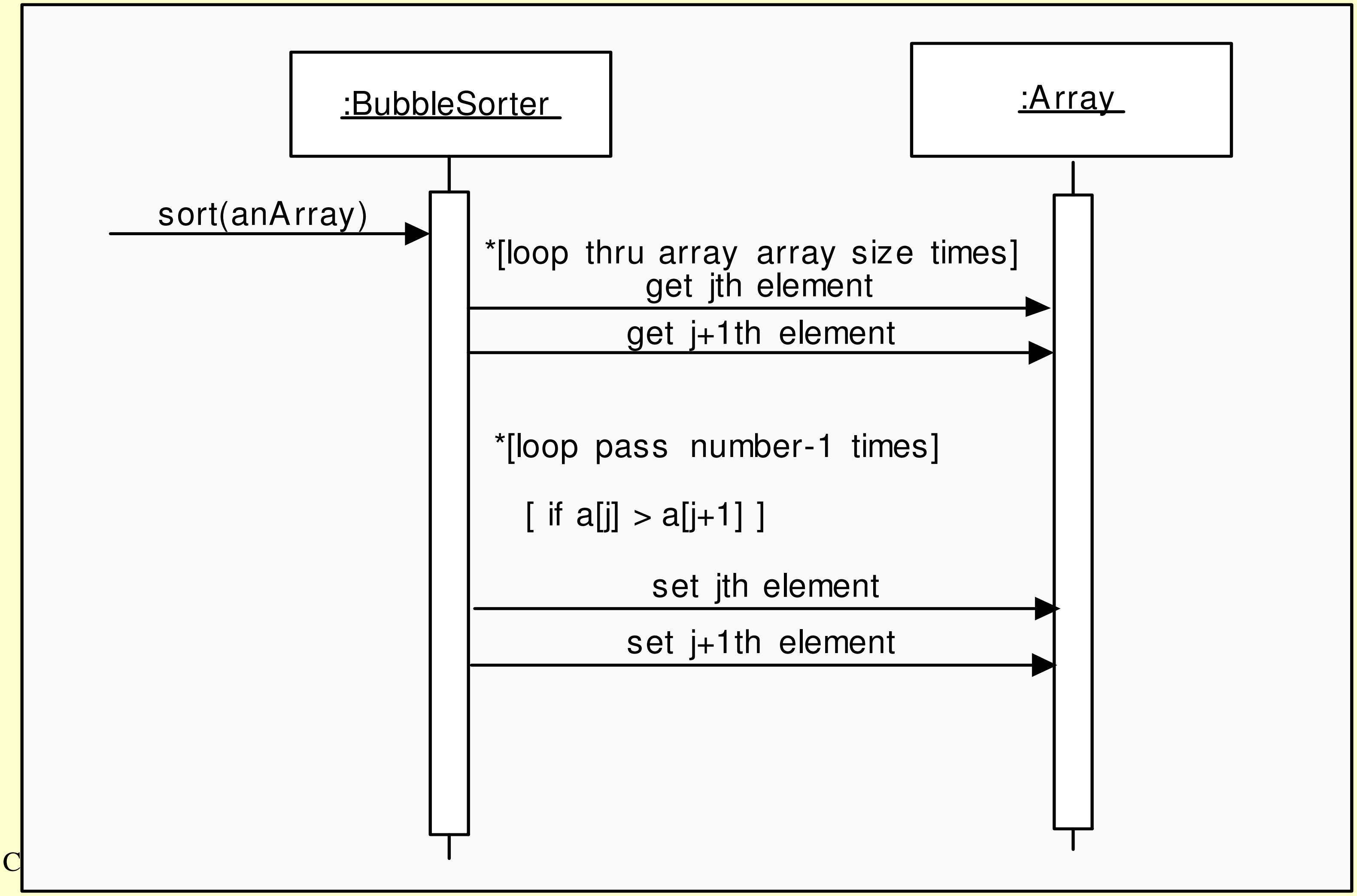
Exception or Error	Recovery Action	Affect on User
Connection is dropped between UI and Domain Server after transaction request is issued.	Transaction continues to completion. Instead of notifying user of status, transaction is just logged. User will be notified of recent (unviewed) transaction results on next login.	User session is terminated... user could've caused this by closing his or her browser, or the system could have failed. User will be notified of transaction status the next time they access the system.
Connection dropped between domain server and backend bank access layer after request is issued.	Attempt to re-establish connection. If this fails, transaction results are logged as "pending" and the user is informed that the system is momentarily unavailable.	User will be logged off with a notice that system is temporarily unavailable and will learn of transaction status on next login.

These descriptions are approachable to marketers, developers and other stakeholders

Strunk and White’s Elements of Style

- *Omit needless words.*
  - Visual equivalents of “needless words”on a collaboration or sequence diagram:
    - Return values
    - Internal algorithmic details
    - Details of caching and lazy initialization
    - Object creation and destruction
  - Visual equivalents of “needless” words: diagrams that don’t add much to your storyline.

Bubble Sort: A UML Sequence Diagram



:BubbleSorter

:Array

sort(anArray)

*[loop thru array array size times]

get jth element

get j+1th element

*[loop pass number-1 times]

[if a[j] > a[j+1]]

set jth element

set j+1th element

Bubble Sort Explained

The algorithm for a bubble sort consists of two nested loops. The inner loop traverses the array, comparing adjacent entries and swapping them if appropriate, while the outer loop causes the inner loop to make repeated passes. After the first pass, the largest element is guaranteed to be at the end of the array, after the second pass, the second largest element is in position, and so on. That is why the upper bound in the inner loop decreases with each pass; we don’t have to revisit the end of the array.

Bubble Sort: A Visual Illustration

- Consider the array 42,56,13,23
- Let's start sorting.........
- 42,56,13,23 no swap
- 42,56,13,23 swap
- 42,13,56,23 swap – end of 1st pass outer loop
- 42,13,23,56 swap
- 13,42,23,56 swap – end of 2nd pass outer loop
- 13,23,42,56 no swap – end of 3rd pass

Bubble Sort: Some Code

```
class BubbleSorter{
 void sort(int a[])
{ for (int i = a.length; –i>=0; )
   { boolean swapped = false;
     for (int j = 0; j<i; j++ ) {
           if (a[j] > a[j+1]) {
              int T = a[j];
              a[j] = a[j+1];
              a[j+1] = T;
              swapped = true; }
      if (!swapped) return; } } }
```

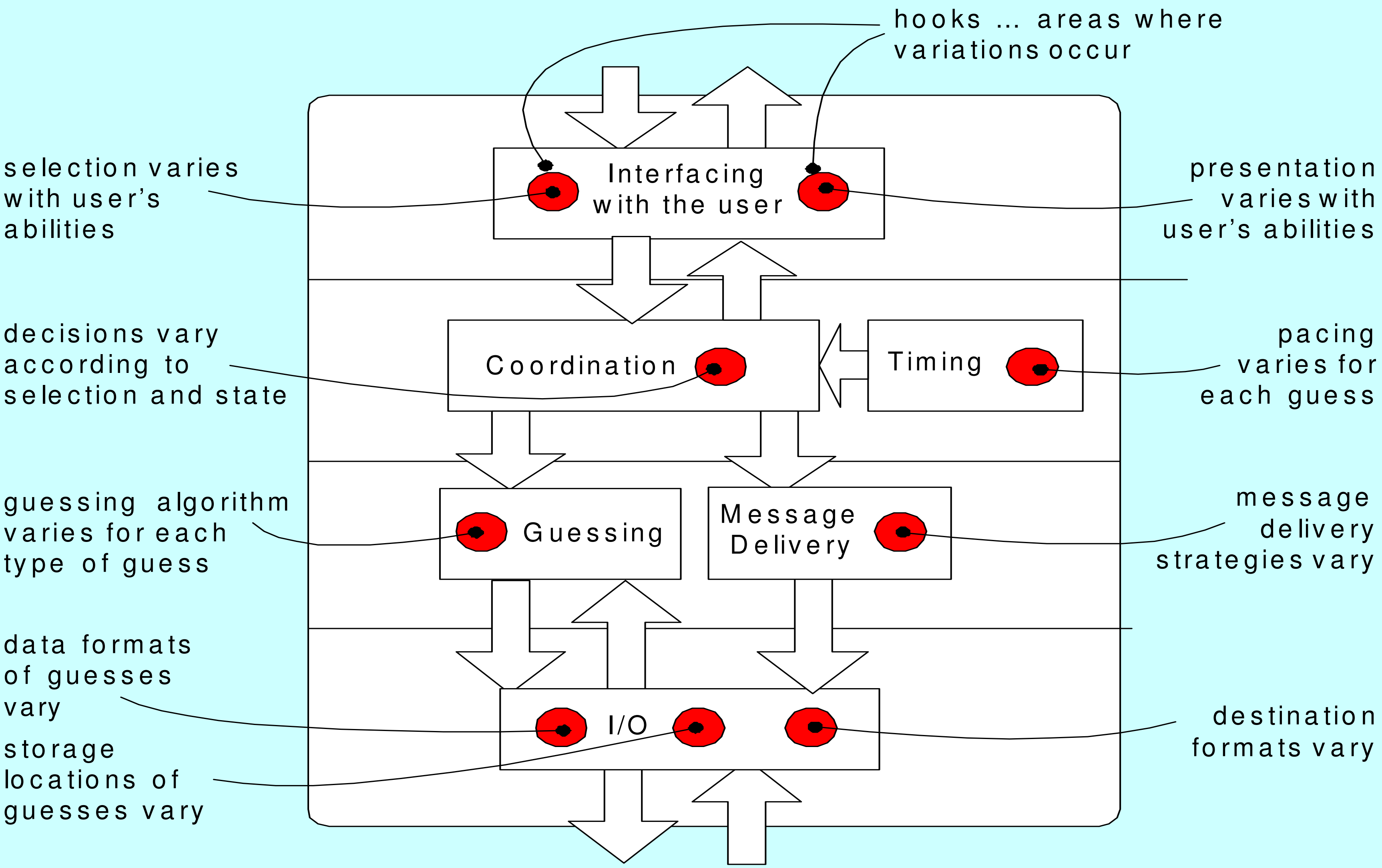
Where UML Diagrams Fall Short

- The best way to see isn't always with a standard diagram.
- Use words, pseudo-code, code, BNF grammar, decision tables, state tables, or pictures that emphasize certain features.
- Sequence diagrams sometimes fall short:
  - They do not show side-effects
  - It's hard to emphasize special areas
  - It's hard to interpret algorithms
  - Control flow is mostly hidden
  - Hard to see iteration

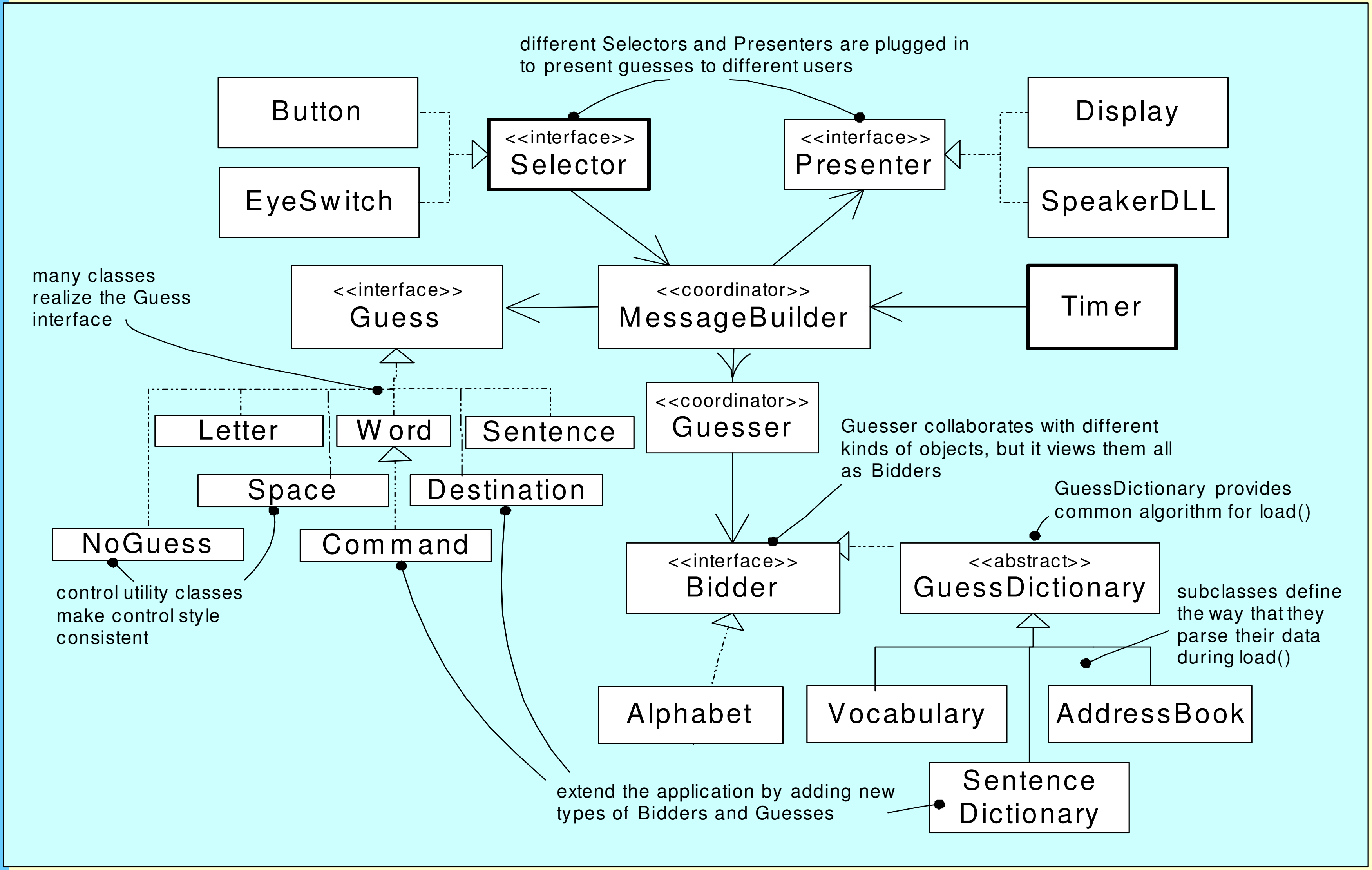
Strunk and White's Elements of Style

- *Revise and rewrite.* If someone doesn't "get it", it could be your problem.
  - A designer drew two views showing the same collaboration. One view omitted the interface details, the other included them. Some developers wanted to know what interfaces to use. Others who only wanted to how their parts of the system were activated didn't want to see these details. Both views were needed to get her design across.

Consider Your Audience


hooks ... areas where variations occur
selection varies with user's abilities
Interfacing with the user
presentation varies with user's abilities
decisions vary according to selection and state
Coordination
Timing
pacing varies for each guess
guessing algorithm varies for each type of guess
Guessing
Message Delivery
message delivery strategies vary
data formats of guesses vary
I/O
destination formats vary
storage locations of guesses vary

The Same View In UML



different Selectors and Presenters are plugged in to present guesses to different users

many classes realize the Guess interface

Guesser collaborates with different kinds of objects, but it views them all as Bidders

GuessDictionary provides common algorithm for load()

control utility classes make control style consistent

subclasses define the way that they parse their data during load()

extend the application by adding new types of Bidders and Guesses

Strunk and White's Elements of Style

- *Do not overstate.* Don't tell more than what you believe at any given point.
  - Precision should be a conscious decision. Use UML when you want to be precise and rough sketches when you want to convey the gist of your design.
  - The same diagram can be drawn with different degrees of precision.
    - If you only know general paths of collaboration—don't show specific messages.
    - If you know specific messages, but not the arguments—don't invent arguments just to fill in the blanks.

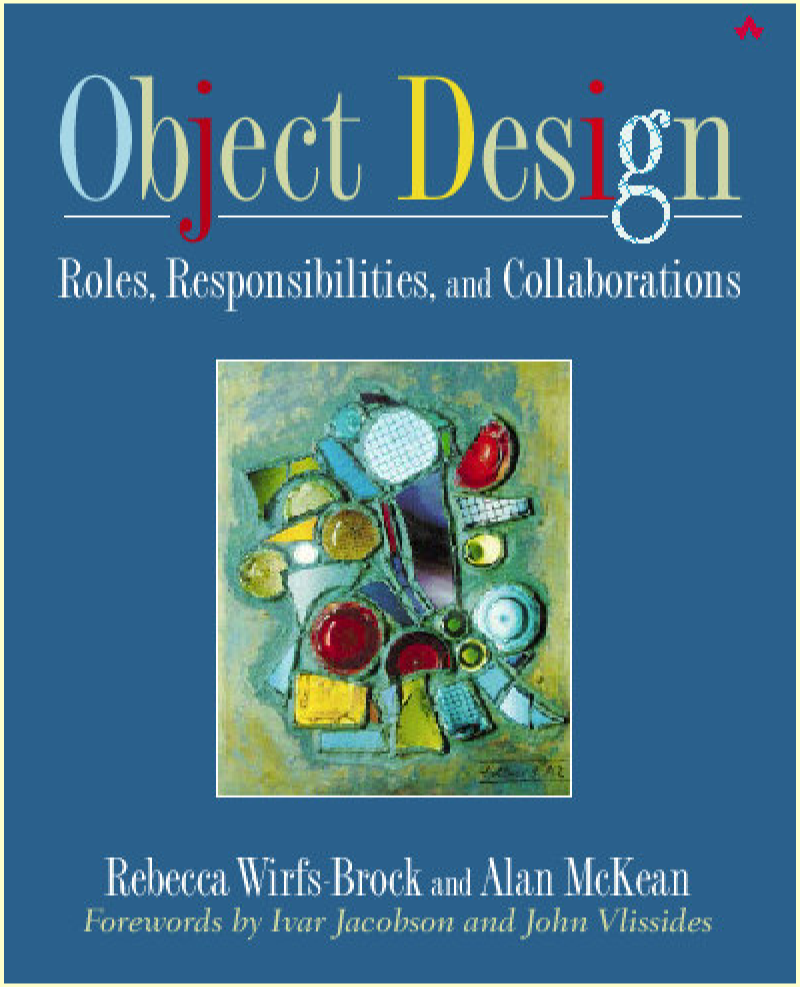
Strunk and White's Elements of Style

- *Do not affect a breezy manner.* Don't leave things understated, undrawn, or unexplained
  - CRC cards are too breezy if you want to explain an interaction sequence.
  - Don't arbitrarily limit your diagrams to a single page, or to ten or less objects. Stick with your story. Get it down, then figure out how to show it.
- *Be clear.* Choose the right form of expression
  - To emphasize message ordering, use a sequence diagram. If timing is critical, add timing marks.
  - Add running commentary to explain.

Strunk and White's Elements of Style

- ***Make sure the reader knows who is speaking.*** Tell a story from one perspective and stick to that storyline.
  - If you are explaining how subsystems collaborate, don't drop down two levels and show objects inside those subsystems collaborating with objects from a standard library.
  - If your intent is to show how a complex responsibility is divided among collaborators, show what helper methods are invoked. Stop there.

Resources


Object Design
Roles, Responsibilities, and Collaborations
Rebecca Wirfs-Brock and Alan McKean
Forewords by Ivar Jacobson and John Vlissides

- Read more about seeing, thinking, shaping, and describing object designs in our new book

  Object Design: Roles, Responsibilities and Collaborations, Rebecca Wirfs-Brock and Alan McKean, Addison-Wesley, 2003
- www.wirfs-brock.com for articles & resources