

Skills for the Agile Designer

Rebecca Wirfs-Brock
rebecca@wirfs-brock.com
www.wirfs-brock.com



Wirfs-Brock Associates



What makes a designer agile?

Core values:

Simplicity

Communication

Teamwork

Curiosity

Satisfying stakeholder needs

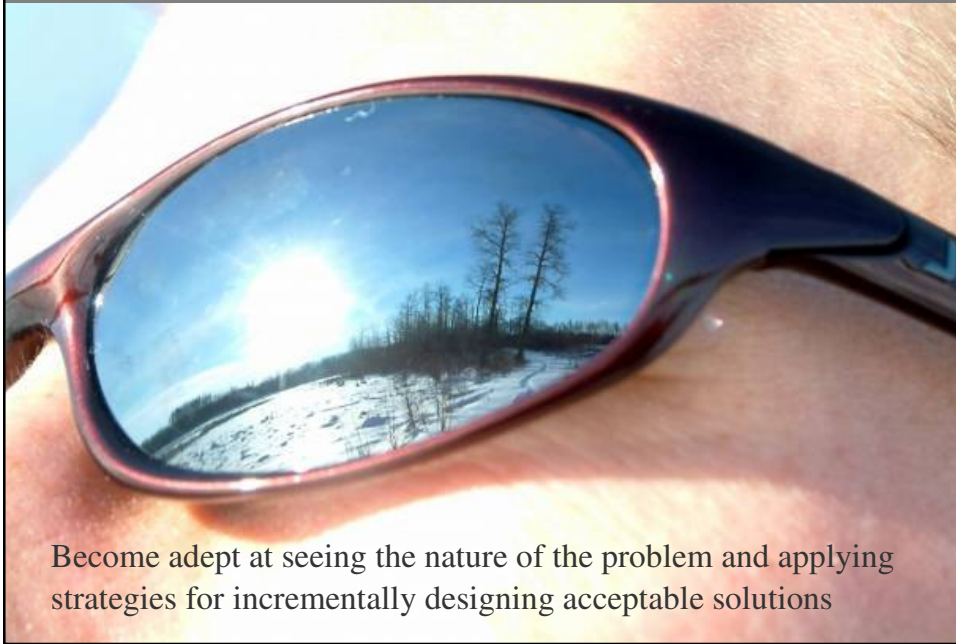
An attitude of learning



Wirfs-Brock Associates

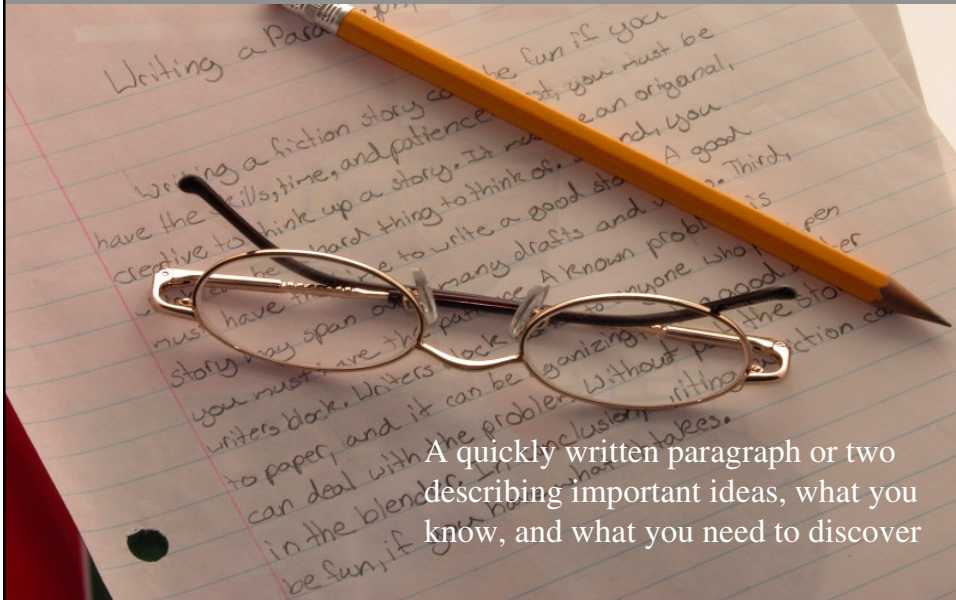
2

How do you become a world class agile designer?



Become adept at seeing the nature of the problem and applying strategies for incrementally designing acceptable solutions

Designer's Story: A Tool for Seeing What's Important...



A quickly written paragraph or two describing important ideas, what you know, and what you need to discover

Why tell a designer's story?

To state your perspective

Describing the problem helps you own it

Sharing them builds understanding

Metaphors are hard to come by...identifying themes and key responsibilities from designer stories is one alternative



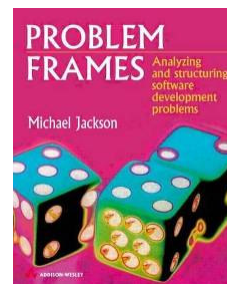
Wirfs-Brock Associates

5

Problem Frames: A tool for seeing typical patterns of software tasks



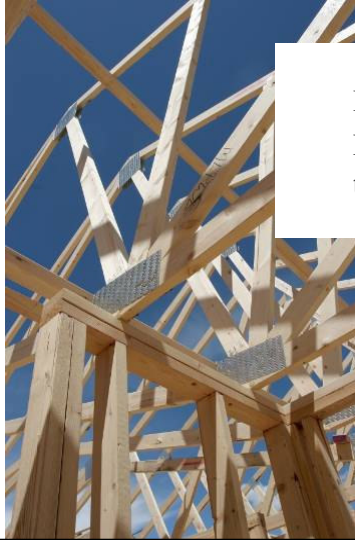
frame—a structure that gives shape or support



Wirfs-Brock Associates

6

Framing strategy: divide and conquer



Decompose problems
Focus on the requirements and
the concerns of each subproblem



7

Tactics for decomposing problems

Identify the core problem
Look for ancillary problems
Constructing mail
Managing mail folders
Sorting mail
Reading mail
Maintaining address lists
Examine problem concerns for more



Five basic problem frames

Workpiece- a tool that allows users to create and manipulate structures (Email editor)

Transformation- converting input according to certain rules (MIME Decoding)

Information- information derived from observed state and/or behavior (Determining a “junk mail” rating)

Commanded behavior- controlling behavior of some “thing” according to user commands (Sending an email message)

Required behavior- controlling the behavior of some “thing” (Sorting incoming mail according to pre-defined filters)



Wirfs-Brock Associates

9

Workpiece Frame Questions

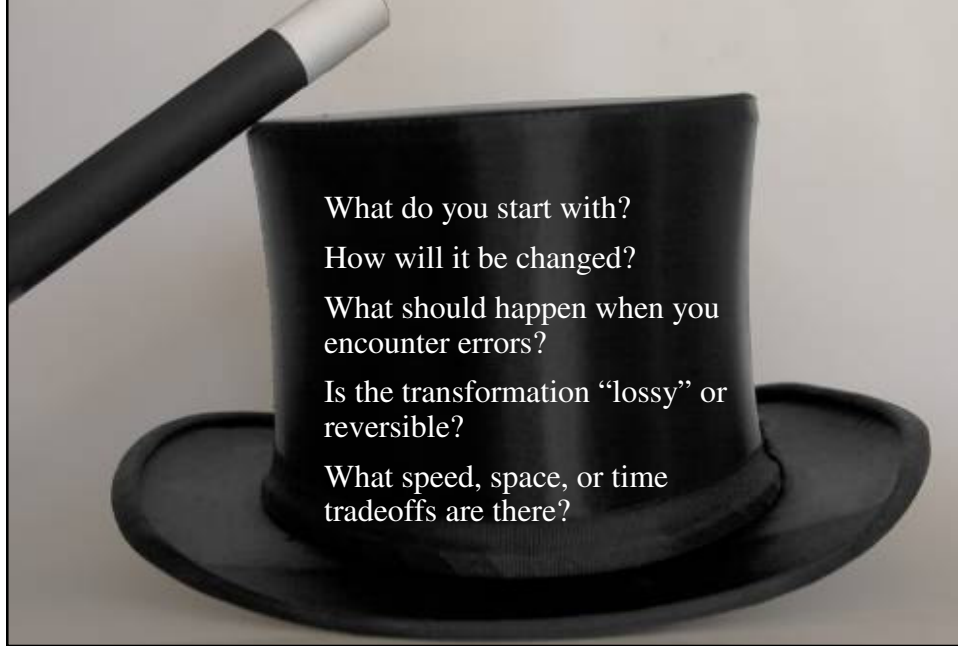
Will it take different forms?

Is there a workflow associated with it?

Should it be published or printed?

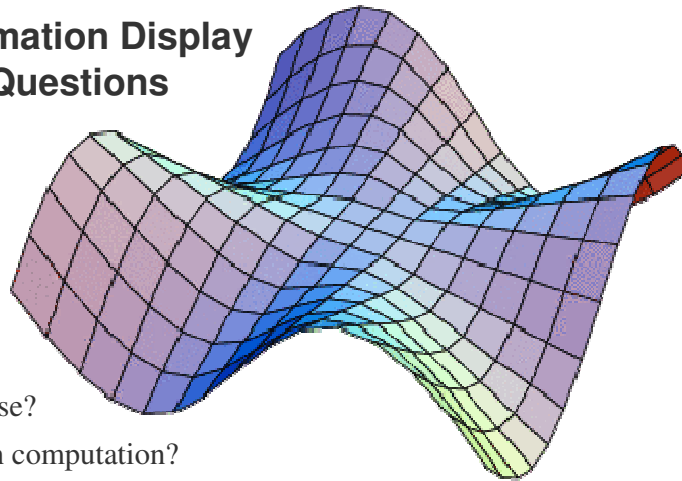


Transformation Frame Questions



What do you start with?
How will it be changed?
What should happen when you encounter errors?
Is the transformation “lossy” or reversible?
What speed, space, or time tradeoffs are there?

Information Display Questions



How precise?
How much computation?
Is the user only interested in current information? Is historical information important?
Are there questions that the user may want to ask about the information? What are they?



Wirfs-Brock Associates

12

Commanded Behavior Frame Questions



What does the user need to know?

Do commands always make sense?

Do commands need to be reversible? logged? monitored?

Required Behavior Questions

What external state must be controlled?

How does your software find out whether its actions have intended effect?

What should happen when things get “out of synch”?

Are there complex interactions?



Wirfs-Brock Associates

Problem frame expectations		
Frame	Complex	Simple
<i>Required behavior</i>	The thing you are controlling	
<i>Commanded behavior</i>	The thing you are changing/modifying	Operator commands
<i>Information display</i>	Interpretation/derivation of information	Information display
<i>Simple Workpieces</i>	Work pieces domain	User
<i>Transformation</i>		Input and Outputs

What about framing during design?

Jackson advocates fully understanding problems before starting design

Agile designers expect to refine ideas throughout the software lifecycle. Framing

- ...can focus design efforts
- ...can help you write better code
- ...can help you estimate complexity
- ...can lead to deeper understanding of what the system must do



Role Stereotypes: A tool for seeing and shaping behaviors



stereotype—A conventional, formulaic, and oversimplified conception, opinion, or image



Wirfs-Brock Associates

17

Object Role Stereotypes



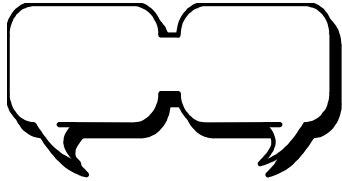
Information holder -
knows and provides
information

Structurer - maintains relationships
between objects and information
about those relationships



Wirfs-Brock Associates

Object Role Stereotypes



Interfacier - translates information and requests

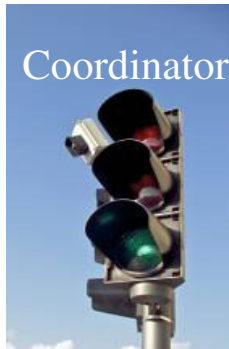
Service provider - performs work on demand



Wirfs-Brock Associates

Object Role Stereotypes

Coordinator - mechanically reacts to events



Controller - makes decisions while closely directing others' actions



Wirfs-Brock Associates

How to Use Role Stereotypes

1. Think about objects or components needed
2. Study a design to learn what types of roles predominate and how they interact
3. Blend stereotypes to make objects more responsible
 - information holders that compute
 - service providers that maintain information
 - structurers that derive facts
 - interfacers that transform



Wirfs-Brock Associates

21

n-tier web applications

Layer	Functionality	Role	Technique
Client	User Interface	Interfacer	HTML, JavaScript
Presentation	Page Layout	Interfacer	JSP
Control	Command	Coordinator	Servlet
Business Logic	Business Delegate	Controller	POJO, Session EJB
Data Access	Domain Model	Information Holder, Structurer	JavaBean, Entity EJB
Resources	Database, Enterprise Services	Service Provider	RDBMS, Queues, Enterprise Service Bus



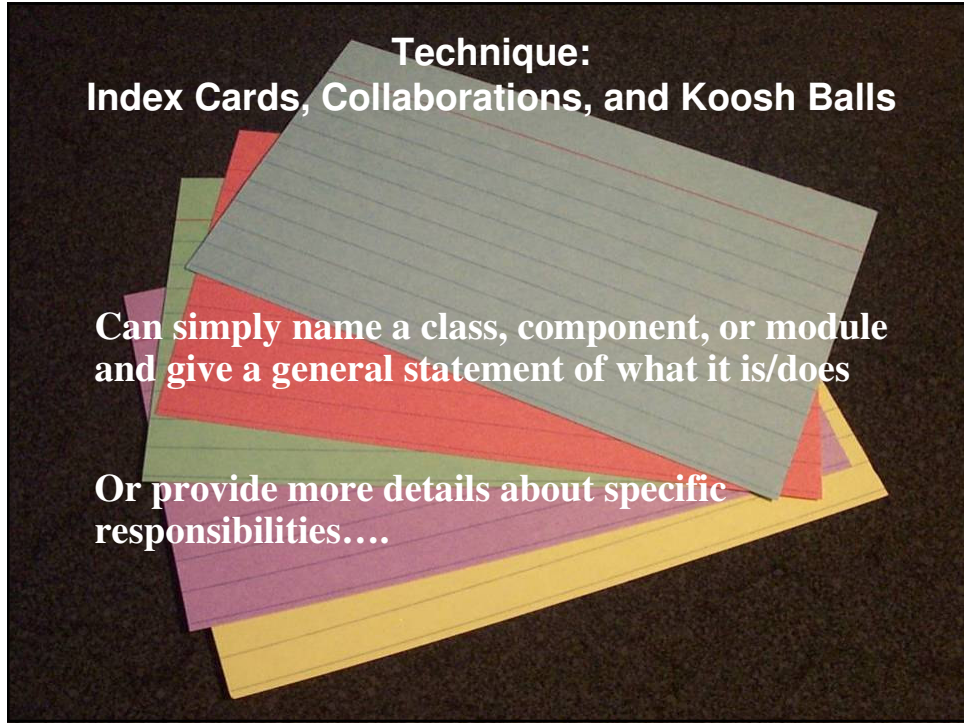
Wirfs-Brock Associates

22

Technique:
Index Cards, Collaborations, and Koosh Balls

Can simply name a class, component, or module and give a general statement of what it is/does

Or provide more details about specific responsibilities....




CRC Cards: An informal design tool
Candidate, Responsibilities, Collaborators

Sensor

Purpose: Represents what the Arbor 2000 system knows about devices that reports data that is physically sensed from the environment. Physical sensors can report light intensity, temperature, wind speed and direction, rainfall and other physical readings. Some kinds of sensors can sense multiple physical characteristics and are capable of reporting readings at different intervals (such as every minute, hourly, weekly, monthly) or based on a significant event (temperature rising x degrees in a period of time, x amount of rainfall, etc.).

Stereotypes: Service Provider, Information holder

 Wirfs-Brock Associates 24

Explaining Purpose

A candidate does and knows certain things. Briefly, say what those things are. A pattern to follow:

An object is or represents a *thing* that *knows or does certain things*. And then mention one or two interesting facts about the object, perhaps a detail about what it does or who it works with.



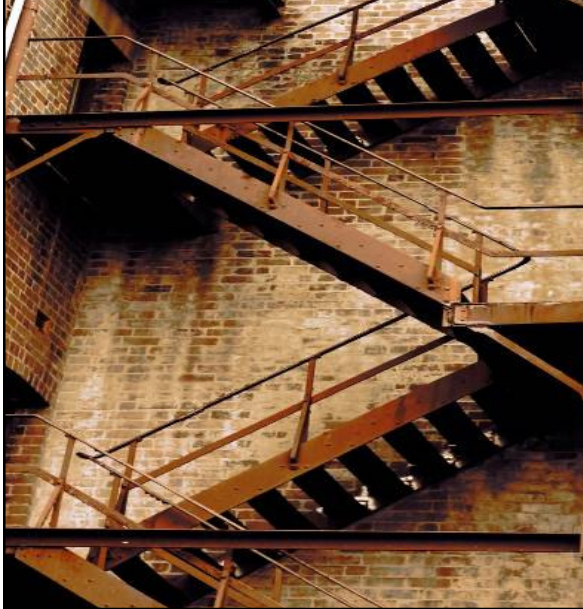
CRC Cards: An informal design tool

Candidate, Responsibilities, Collaborators

Sensor	
knows physical characteristics it can detect	Parser — Collaborators
knows reporting interval	Measurement — Collaborators
maintains configuration parameters	
knows sensor make and model	Responsibilities
knows location	
knows if activated	
creates measurements from reports	



Technique: seeing a design at different levels



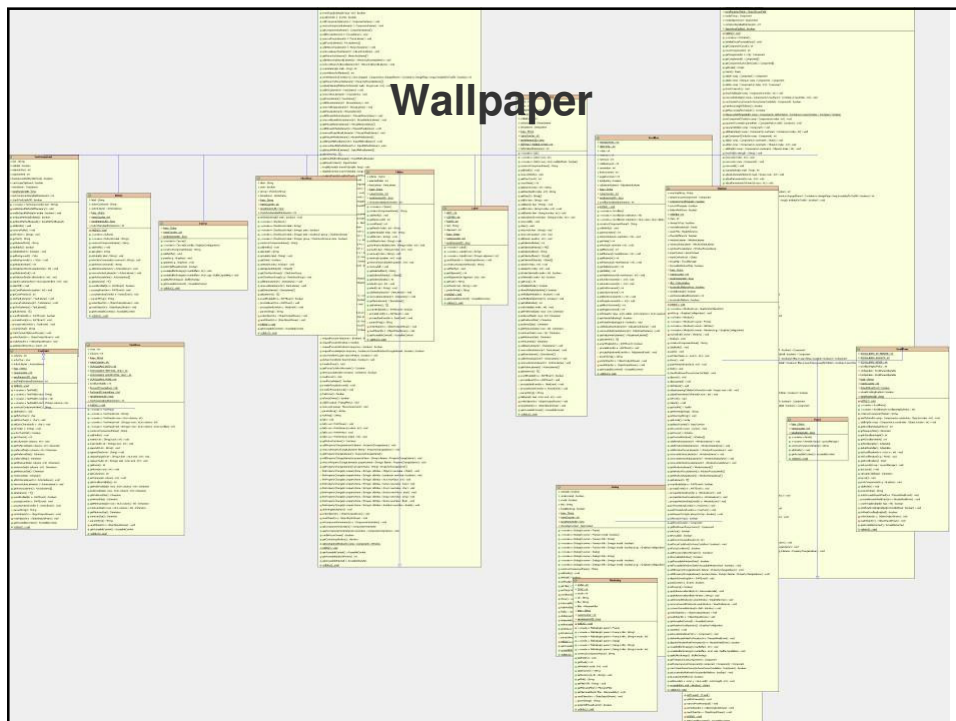
We can talk about design elements at different levels:

conceptual- a set of responsibilities

specification- methods, functions and attributes

implementation- code and data

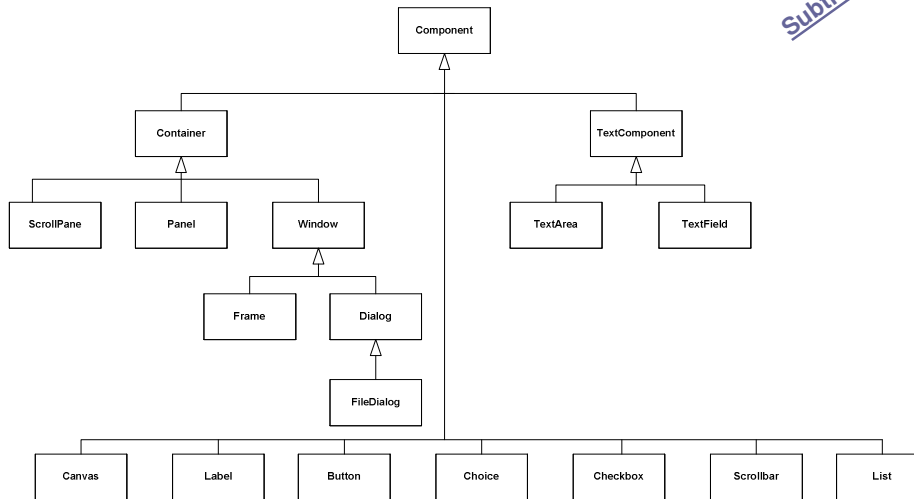
27



Class Diagrams

can show inheritance hierarchies

Subtle points



This diagram shows only Component hierarchy of the AWT library without all the classes' gory details. An implementation view doesn't have to show every detail. What you leave out is just as important a decision as what to include.



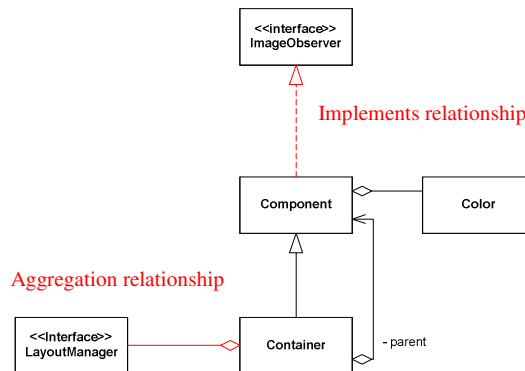
Wirfs-Brock Associates

29

Class Diagrams

can highlight key relationships

Subtle points



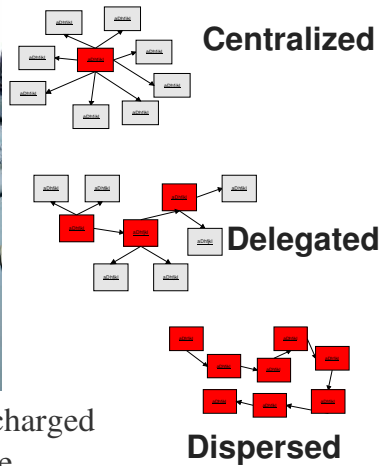
The interface stereotype describes a type that contains only operation signatures. Thus, a Container 'contains' one or more types that conform to the LayoutManager interface. A Component 'implements' the ImageObserver interface.



Wirfs-Brock Associates

30

Control centers and collaboration styles: Tools for shaping solutions



control center—a place where objects charged with controlling and coordinating reside



Wirfs-Brock Associates

31

Control Design

There are many control centers in your design, each may have a different style

Involves decisions about

- how to control and coordinate tasks,
- where to place responsibilities for making domain-specific decisions (rules), and
- how to manage unusual conditions (the design of exception detection and recovery)

Goal: Make similar parts of a design consistent



Wirfs-Brock Associates

32

Characteristics of Centralized Control

Generally, one object (the controller) makes most of the important decisions and figures out what to do.

Tendencies to avoid:

- Overly complex control logic
- No work done by other objects

Benefit:

Easy to follow what's going on

Drawback:

Changes can ripple among controlling and controlled objects



Wirfs-Brock Associates

33

Characteristics of Delegated Control

A coordinator passes some decision making and actions off to objects surrounding a control center. Each delegate has a significant role:

- Coordinators tend to know about fewer objects
- Messages are higher-level

Tendency to avoid:

- Helper objects that don't do enough

Benefit:

Changes typically localized and simpler
Easier to divide interesting design work among a team



Wirfs-Brock Associates

34

Characteristics of Dispersed Control

Decision-making and action are spread among a chain of objects who each perform a small, well-defined task

Tendencies to avoid:

Little or no value-added by an object in the chain

Hardwired dependencies

Opportunity:

Design so responsibilities can be chained in novel ways



Wirfs-Brock Associates

35

Trust Regions: A tool for seeing where “defensive” behavior is or isn’t needed



Collaborate



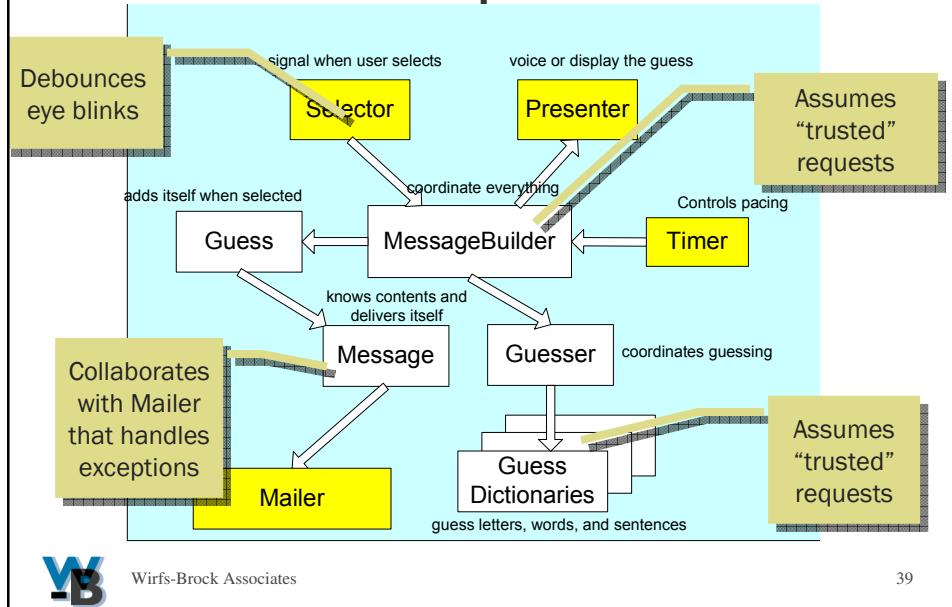
To work together, especially in a joint intellectual effort

Collaborate



To cooperate treasonably, as with an enemy occupation force

Objects At The “Edges” Often Take On Added Responsibilities



Trust regions

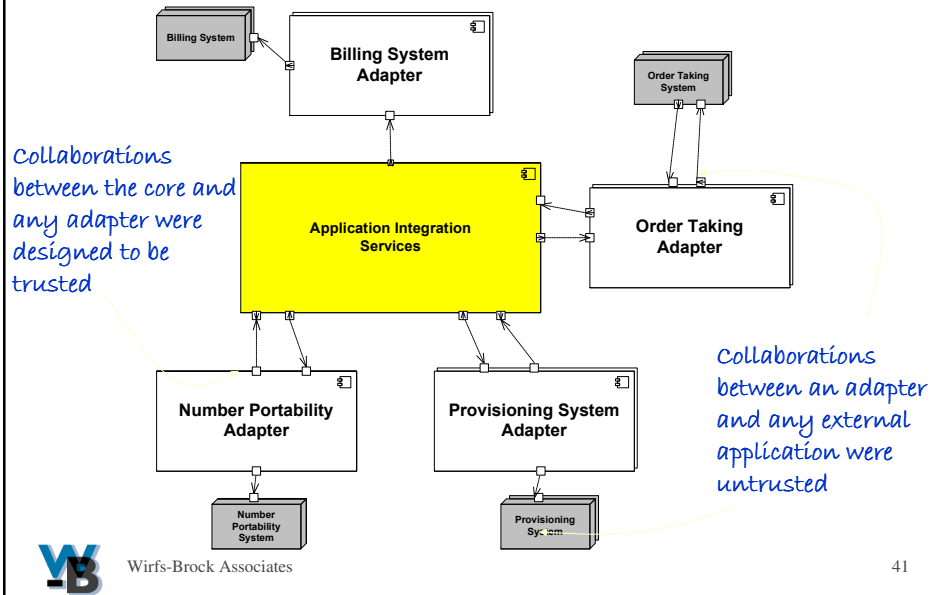
Components and objects at the “edges” may take on extra responsibilities

Within a trust region, collaborations can be more collegial

Check once, then proceed...

Code deep inside a trust region need not check for well-formed or timely requests

Trust In A Telco Integration Application



Collaboration Cases To Consider

Collaborations between objects or components...

- that interface to the user and the rest of the system
- in different layers or subsystems
- inside your system that interface to external systems
- you design and those designed by someone else

There are degrees of trust

Don't design every object to behave defensively

- Redundant checks are hard to keep consistent and lead to brittle code and poor performance

Requests From Untrusted Sources

The receiver is likely to check for timeliness, relevance, and correctly formed data

There are degrees of trust

Don't design every object to behave defensively

Redundant checks are hard to keep consistent and lead to brittle code

It leads to poor performance



Wirfs-Brock Associates

43

Design problem types: A tool for balancing priorities



Photo courtesy Drum Journey www.drumjourney.com

Each design problem category warrants a different approach and has a different rhythm

Design tasks aren't alike

Core design problems include those fundamental aspects that are essential to your software's success

Revealing design problems when pursued, lead to a fundamentally new, deeper understanding

The rest, while not trivial requires far less creativity or inspiration




Deciding what's core

What are the consequences of “fudging” on that part?

Would the project fail or other parts of your design be severely impacted?

When there are fundamentally different expectations, dig deeper. Someone may know something that others have ignored






Revealing design problems

What distinguishes revealing problems is their degree of difficulty and the element of surprise, discovery and invention


Some core problems may be revealing, many are not



47

Revealing design problems are always hard...

- coming up with a solution may be difficult—even though implementation may be straightforward
- they may not have a simple, elegant solution
- they may not be solvable in a general fashion
- they may require new inventions



Wirfs-Brock Associates

48

Agility and design problem types

Sort work into “problem buckets” making sure each iteration gets enough core work accomplished

Track how much time is spent on “the rest”

Use post-iteration reflections to ask why things were harder than they first appeared

Break out of planned iteration cycles to tackle revealing problems (they’ll need more than just a design spike)

Make sure the team gets involved on core design issues



Wirfs-Brock Associates

49

Group Decision Making

Common ways to decide don't always contribute to a design's integrity

- Voting
- Dictatorship
- Reaching consensus
- ✓ Gathering
- ✓ Sub-team



Wirfs-Brock Associates

50

A Two-Stage Decision Process

After gathering options, sort through them: No, no, no, maybe, no, no, no, no, maybe

Examine remaining options carefully



Wirfs-Brock Associates

Designing Responsibly

Use the best tool for the job

Tools for thinking, abstracting, analyzing, simplifying

Tools for making your design flexible

Learn design techniques, and practice, practice, practice

Keep learning



Wirfs-Brock Associates

52

How do you grow agile designers?



Seed your organization with some skilled in agile development practices *and* design

Allow design discussions to be a part of development

Grow best practices

The best designers never give up, they just know when to call it a day!

Thank you

-Rebecca
rebecca@wirfs-brock.com

www.wirfs-brock.com



Wirfs-Brock Associates

54