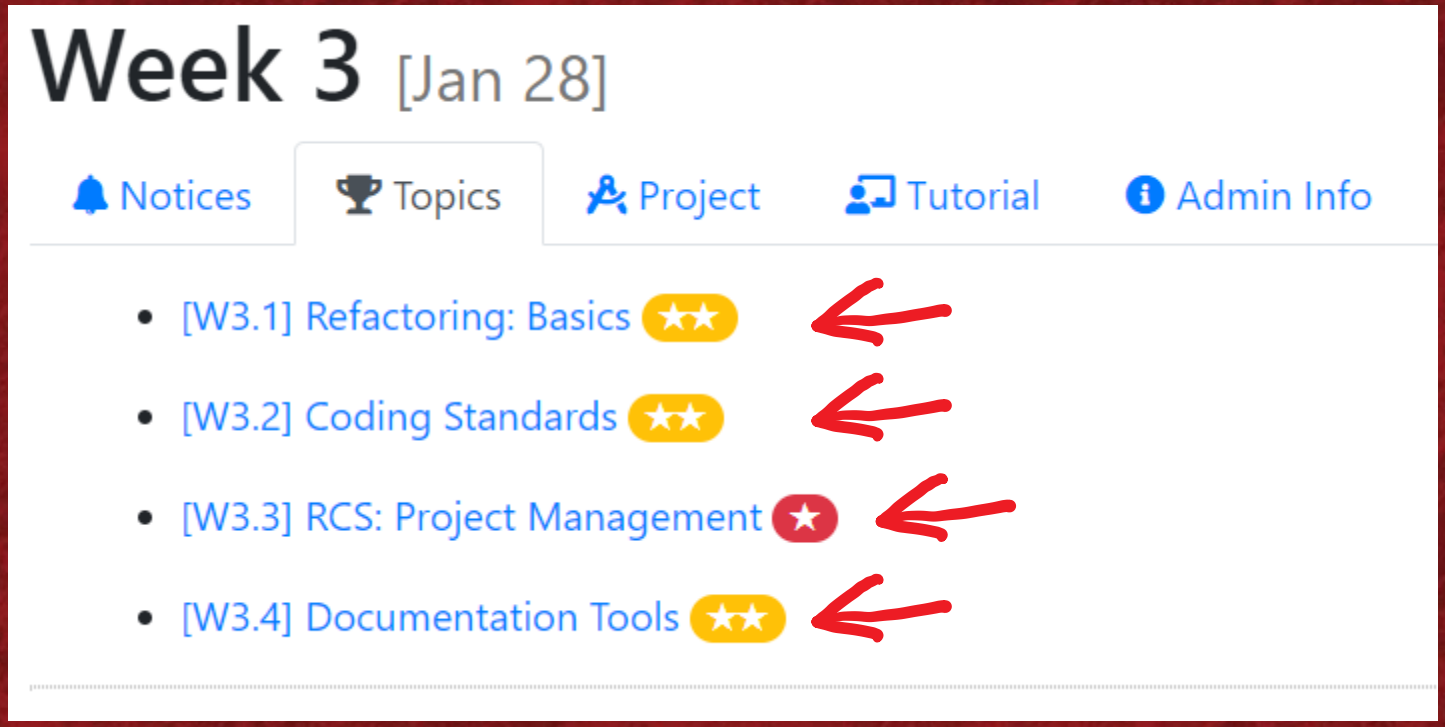
WARNING

These slides are not optimized for printing or exam preparation. These are for lecture delivery only.

These slides are made for PowerPoint 2010. They may not show up well on other PowerPoint versions. You can download PowerPoint 2010 viewer from here.

These slides contain a lot of animations. For optimal results, watch in slideshow mode.

Recap


Week 3 [Jan 28]
Notices
Topics
Project
Tutorial
Admin Info
[W3.1] Refactoring: Basics
[W3.2] Coding Standards
[W3.3] RCS: Project Management
[W3.4] Documentation Tools

Review questions

In a forking workflow ...

1. Developers contribute to the main repository directly
2. There are no merge conflicts
3. Pull requests are the main mechanism for making contributions ✓

Which is/are true about refactoring?

1. Applicable to any programming language ✓
2. Changes the function of the program
3. Testing is a must after each refactoring operation ✓
4. Can make the code slow ✓
5. Too hard to refactor code as it is a lot of manual labor

Bracketing style specified in OSS-NUS Java coding standard is ...

1. Sumerian style
2. Mesopotamian style
3. Egyptian style ✓
4. Roman style

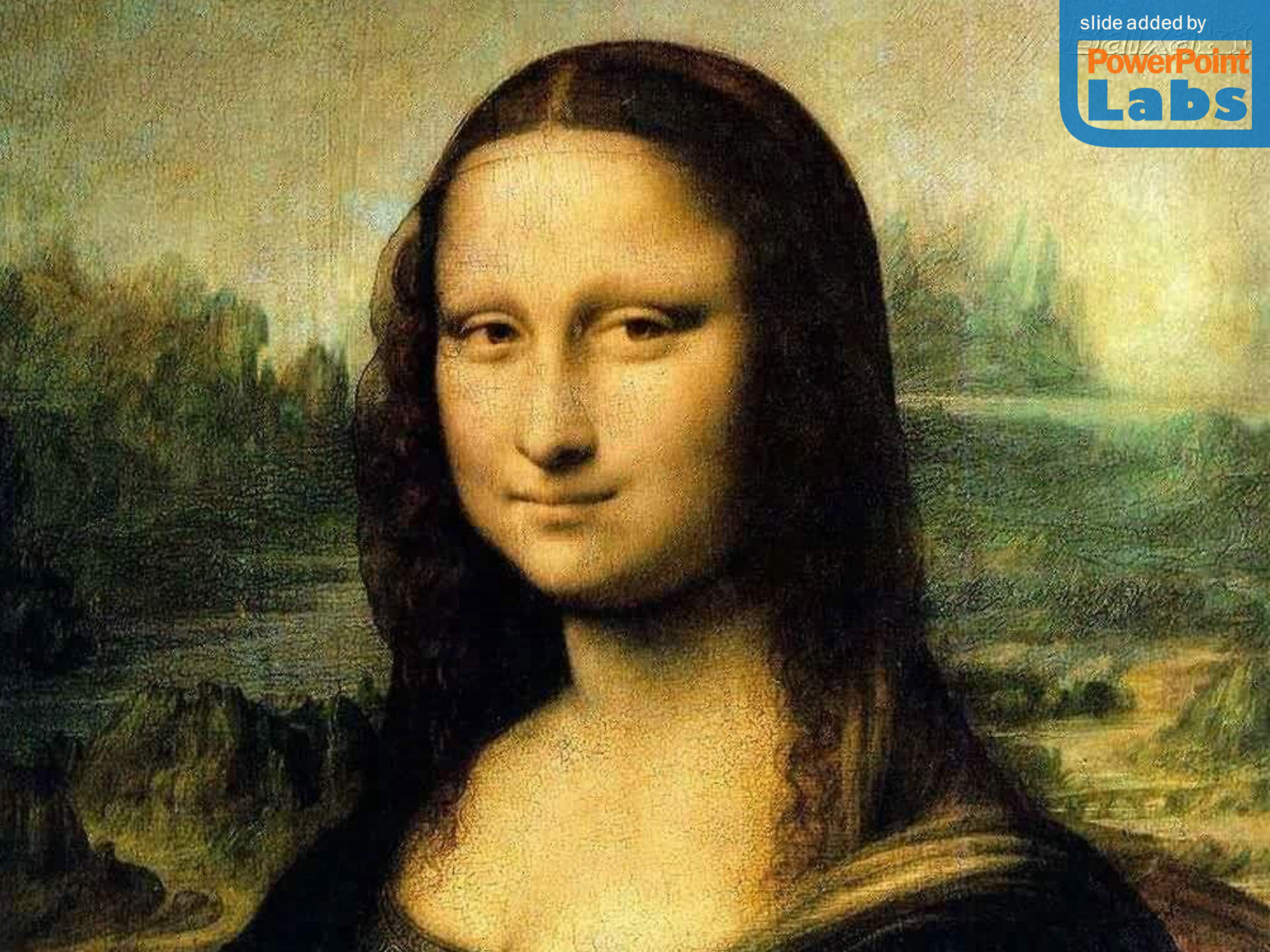
Week 4 [Feb 4]

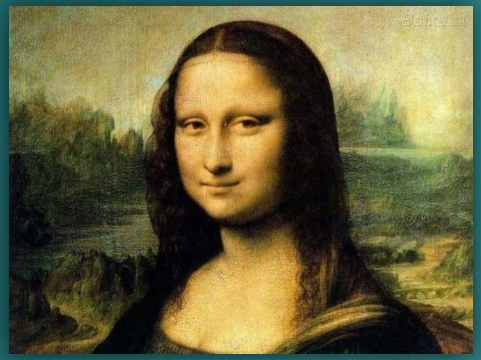
Notices | Topics | Project | Tutorial | Admin Info

- [W4.1] Requirements analysis
- [W4.2] CodeQuality
- [W4.3] Exception Handling
- [W4.4] OOP: Classes & Objects
- [W4.5] Java: enumerations
- [W4.6] OOP: Inheritance

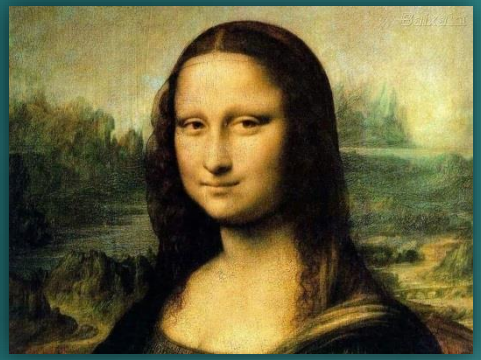
Practices for better quality code

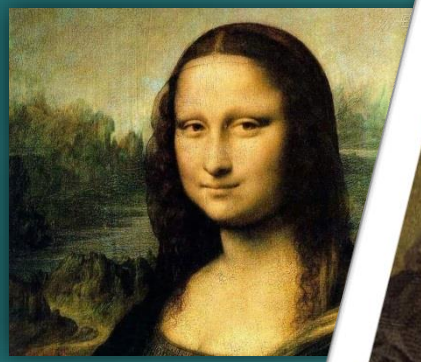

Baixat.tt

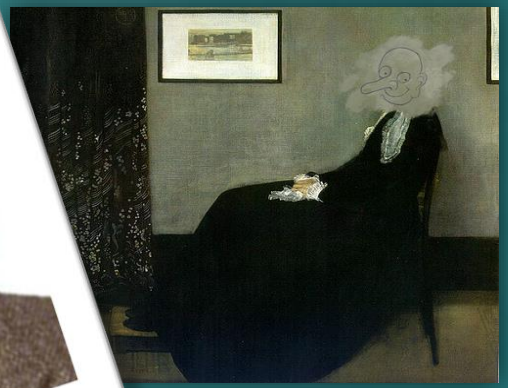

slide added by
PowerPoint
Labs



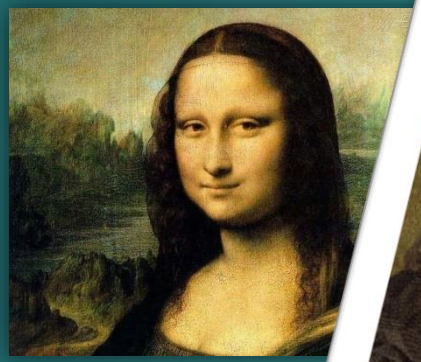
slide added by
PowerPoint Labs

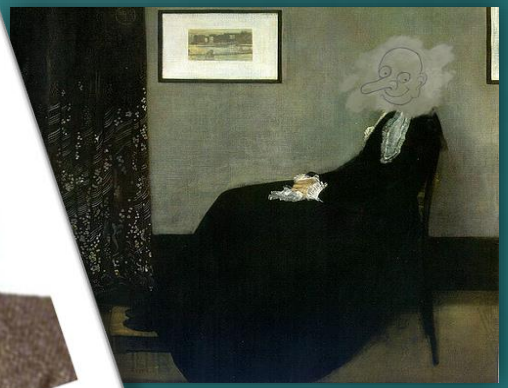






slide added by
PowerPoint Labs





THE MATRIX HAS YOU
ALL AROUND US
IT IS THERE WHEN YOU WATCH TELEVISION
IT IS THERE WHEN YOU WATCH
THE MATRIX HE IS THE ONE
IT IS ALL AROUND US
IT IS THERE WHEN
+ IOMAP_SIZE + 1)
#endif
/* segment base address (overwrit
/* length (overwritten) */
/* segment type */
/* priority level */
/* descriptor present */
/* default 16 size */
/* granularity */
d be a compile time error, but cpp doesn't grok size
the memory layout that we use for the vm86 region.
static void
stack
Page Tbl 1M + 64K = 272 entries = 1088 bytes
PCB size: ~240 bytes

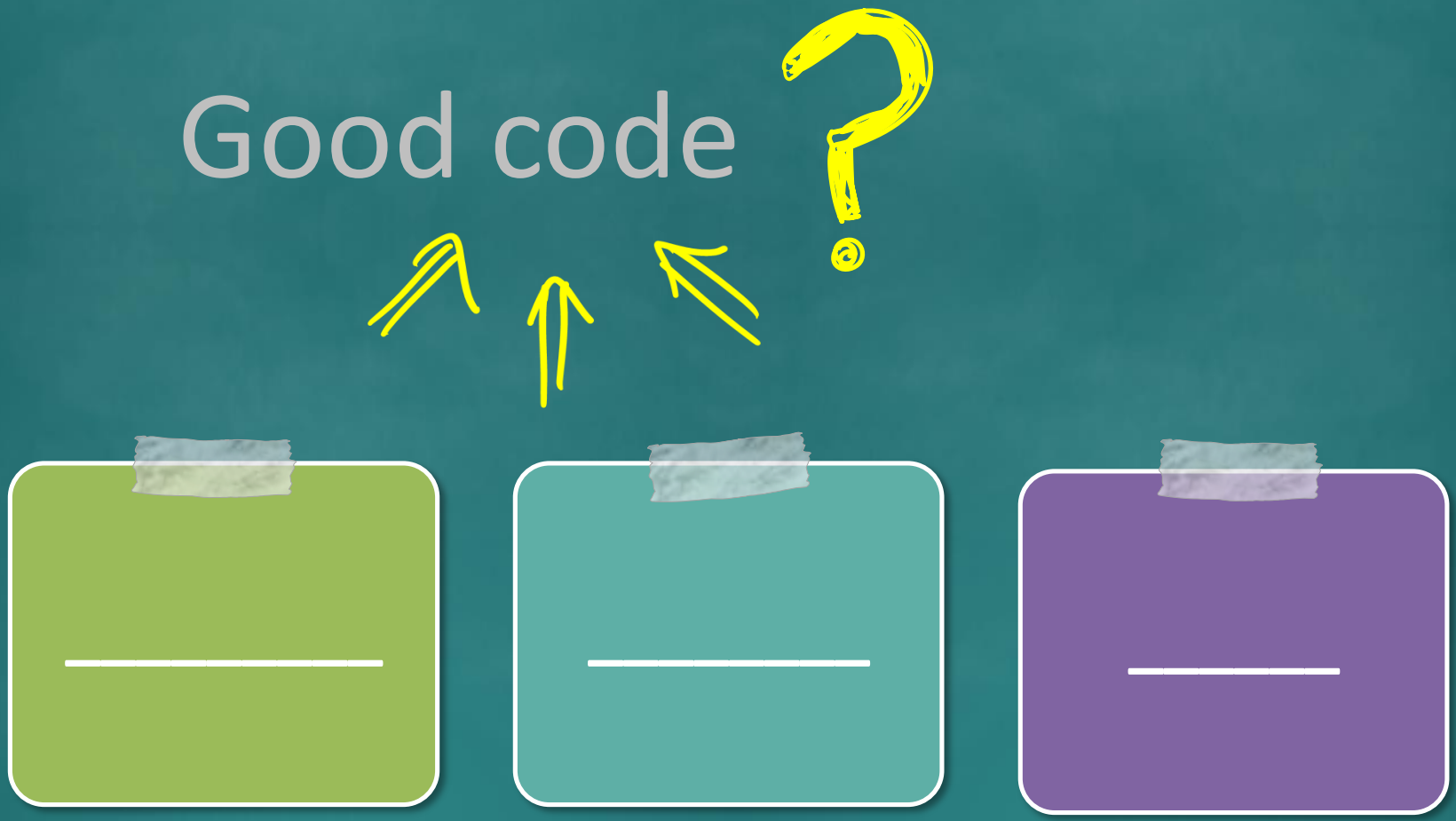
Your code quality

→ Your project grade

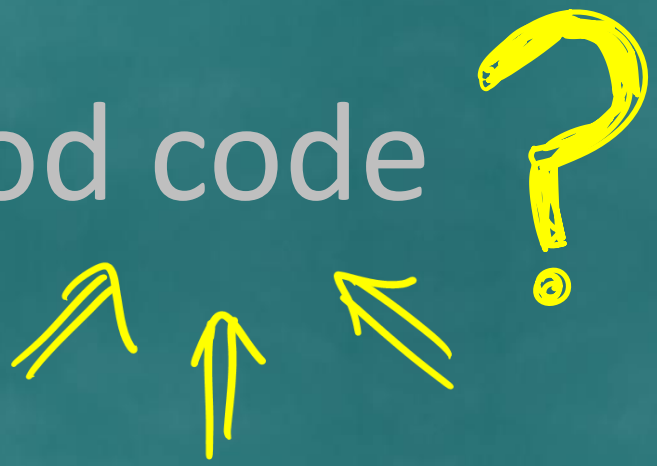
[~10-12%, individual]

→ Your career

What should be the characteristics of good quality code?



Good code



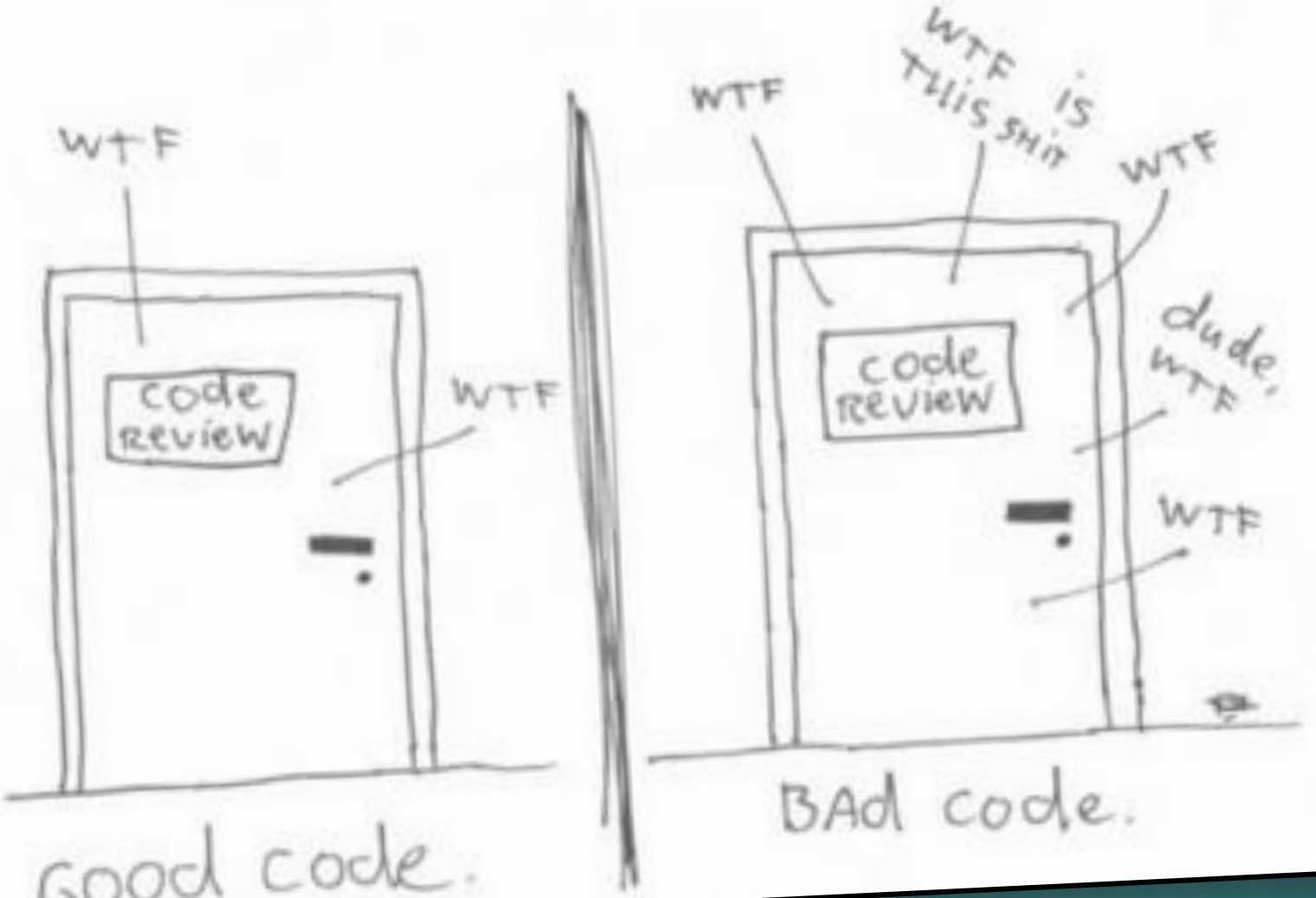
Fast

Secure

_____

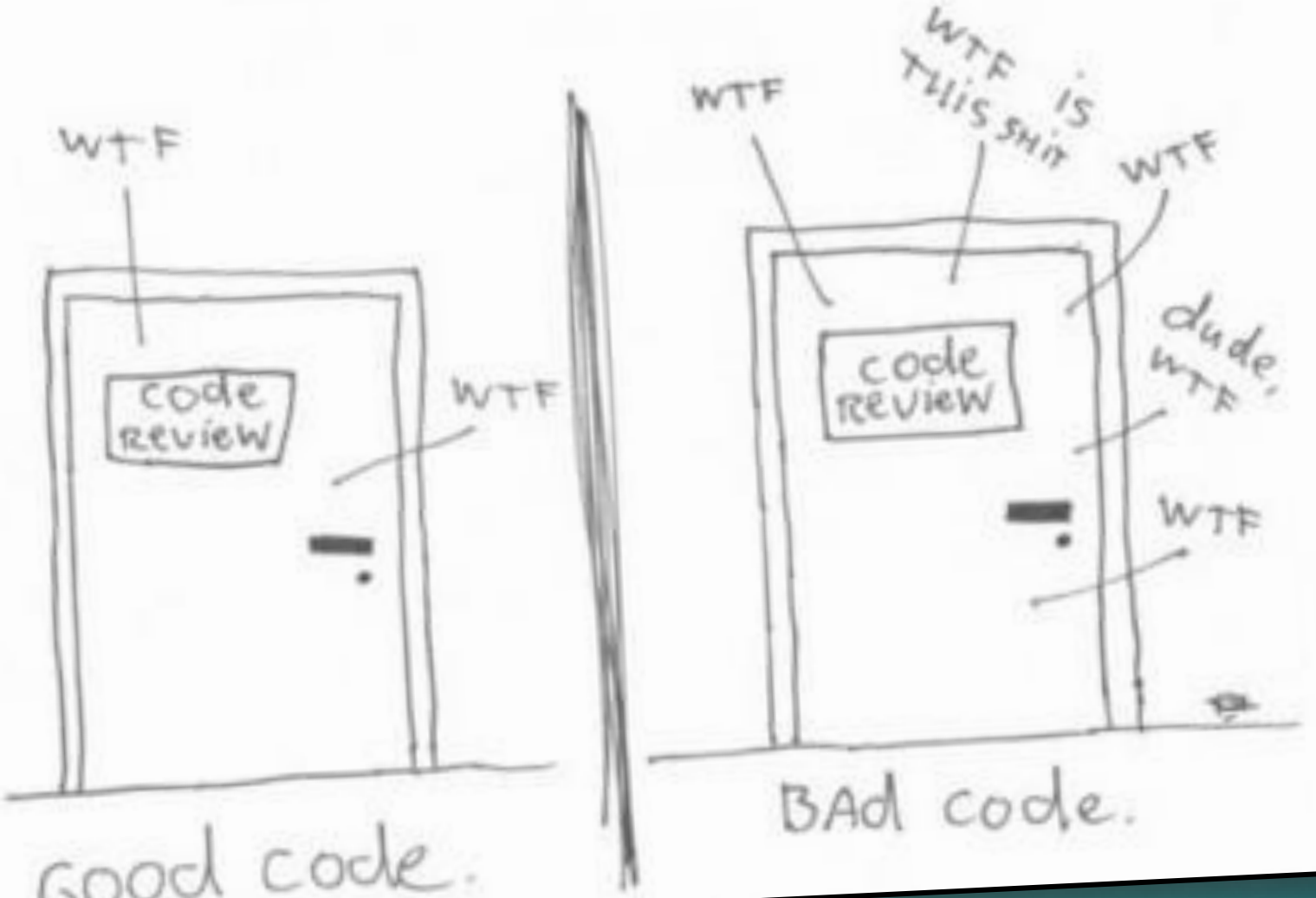
How do you measure the quality of code?

The ONLY VALID MEASUREMENT OF code QUALITY: WTFs/MINUTE



(c) 2008 Focus Shift

The ONLY VALID MEASUREMENT OF code QUALITY: WTFs/MINUTE



Any fool can write code that a computer can understand.
Good programmers write code that humans can understand.

--Martin Fowler

Practices for better quality code

1. .....
2. .....
3. .....
4. .....
5. .....

CONCRETE

SUBJECTIVE

Which one is the more subjective guideline?

a.

Class names should start with an upper case letter.

b.

Class names should be meaningful.

slide added by PowerPoint Labs

CONCRETE

SUBJECTIVE

a.

Class names should start with an upper case letter.

b.

Class names should be meaningful.

CONCRETE

a.

Class names should start with an upper case letter.

SUBJECTIVE

b.

Class names should be meaningful.

SUBJECTIVE

SUBJECTIVE

SUBJECTIVE

CONCRETE

SUBJECTIVE

SUBJECTIVE

CONCRETE

SUBJECTIVE

SUBJECTIVE

SUBJECTIVE

CONCRETE

SUBJECTIVE


Martin Fowler
Steve McConnell
Robert C Martin
Kent Beck
SUBJECTIVE
SUBJECTIVE

Practices for better quality code

1. .....
2. .....
3. .....
4. .....
5. .....

Practices for better quality code

1. .....
2. .....
3. .....
4. .....
5. .....

Practices for better quality code

1. Follow a standard — CONCRETE
2. ..... — SUBJECTIVE
3. ..... — SUBJECTIVE
4. ..... — SUBJECTIVE
5. ..... — SUBJECTIVE

Standardize what?

Answer on Archipelago

```
11  import javax.servlet.http.*;
12
13  /**
14   *
15   * @author Pourya
16   */
17  public class ActionHandler extends HttpServlet
18
19      /**
20       * Processes requests for both HTTP <code>G
21       * @param request servlet request
22       * @param response servlet response
23       */
24      protected void processRequest(HttpServletR
25              throws ServletException, IOException {
26          response.setContentType("text/html;cha
27          PrintWriter out = response.getWriter()
28          try {
29              response.addCookie(new Cookie("","
30
```

JUST DO IT

Is this comment OK?

What

/** Removes words not used by the
compiler during… */

```
String trimExtraWords(String inputString){
        …
        start = header.trim(header.indexOf('k')+2,
                min(footer.length()/2-2, temp++))
        header.compact();
}
```

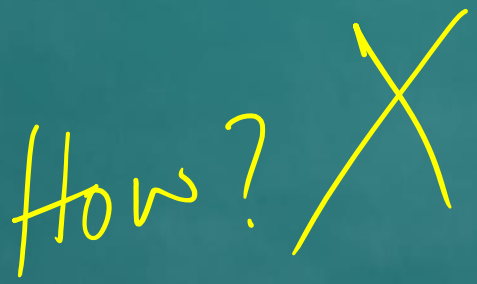
Is this comment OK?

Why

```
String trimExtraWords(String inputString){
        …
        start = header.trim(header.indexOf('k')+2,
                min(footer.length()/2-2, temp++))
        header.compact();
}
```

// to remove multiple spaces

Is this comment OK?




```
String trimExtraWords(String inputString){
        …
        start = header.trim(header.indexOf('k')+2,
                min(footer.length()/2-2, temp++))
        header.compact();
}
```

// first we get index of k and …

Practices for better quality code

1. Follow a standard
2. Comment, if you must!
3. ……
4. ……
5. ……

```
Boolean flag;
```

vs

```
Boolean isValidRespor
```

Unrelated things should be named distinctly

ClassName

variableName

STANT_NAME

ing()

tion()

```
String colorRed;
String colorBlue;     // ex value of blue
String hexGreen;      // hex value of green

int colorGreen;       // num. times green is used
int redColorCount;    // num. times red is used
```

Practices for better quality code

1. Follow a standard
2. Comment, if you must!
3. Naming
4. ……
5. ……

What is the discount if *already subsidized*?

```
int calculateDiscount() {
    int discount;
    if (!age) {
            if (!sub) {
             if (! notFTime) {
                    discount = 500;
              }else {
                    discount = 250;
              }
            } else {
                discount = 250;
            }
      }else {
           discount = 0;
    }
     return discount;
}
```

a) 0
b) 250
c) 500
d) 1000
e) Huh?

Q1. What is the discount if already subsidized?

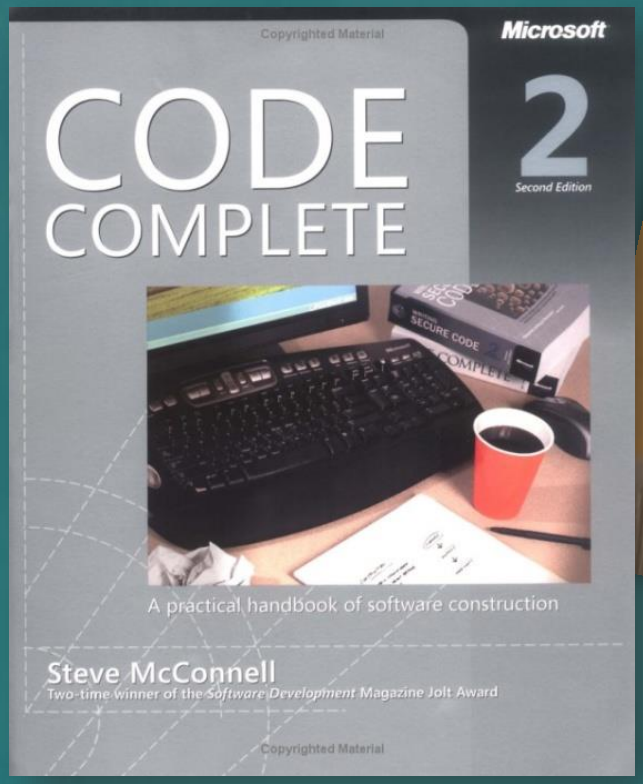
```
int calculateDiscount(){
    int discount;

    if (isSenior) {
        discount = 0;
    } else if (isAlreadySubsidized) {
        discount = 250;
    } else if (isPartTime)
        discount = 250;
    } else {
       discount = 500;
    }
    return discount;
}
```

Practices for better quality code

1. Follow a standard
2. Comment, if you must!
3. Naming
4. Make code Readable
5. ……

Good code is its own best documentation.

As you're about to add a comment, ask yourself, 'How can I improve the code so that this comment isn't needed?' --Steve McConnell



```
int calculateDiscount(){

    int subsidy;
    if (isSenior) {
       subsidy =   -1

   } else if (isAlreadySubsidized) {
       subsidy = 250 ;

   } else if (isPartTime)
       subsidy = 250 ;

   } else {
       subsidy = 500;
   }
     return subsidy;
 }
```

We can write this instead: True | False
if (isAlreadySubsidized || isPartTime)

?

MAGIC NUMBERS

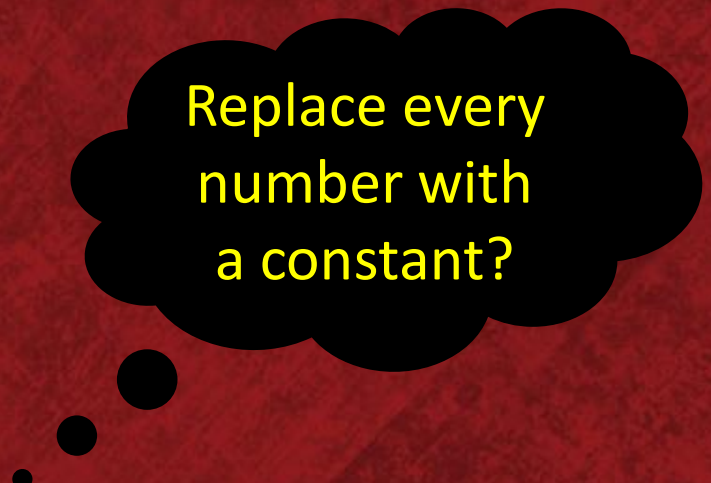
```
final int  FULLTIME_SUBSIDY = 500
```

```
int calculateDiscount(){

     int subsidy;
     if (isSenior) {
        subsidy = REJECT_SENIOR;

    } else if (isAlreadySubsidized) {
        subsidy = SUBSIDIZED_SUBSIDY ;

    } else if (isPartTime)
        subsidy = FULLTIME_SUBSIDY * RATIO ;

    } else {
        subsidy = FULLTIME_SUBSIDY;
    }
      return subsidy;
 }
```

MORE INFORMATIVE

Practices for better quality code

1. Follow a standard
2. Comment, if you must!
3. Naming
4. Make code Readable
5. Avoid magic numbers



Admin Matters

Fork codebase from semester fork

nus-cs2103-AY1819S2

NOT from the upstream se-edu

Reason – we use Java 9; upstream uses Java 8

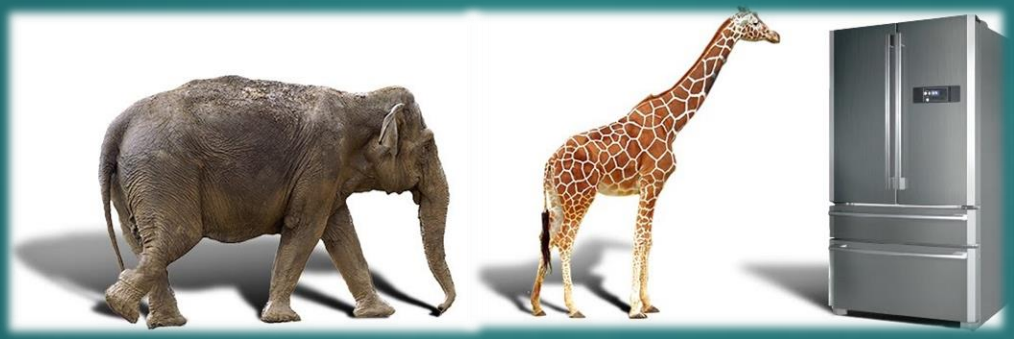
Team formation/registration

How do you put a giraffe in a fridge?

How do you put an elephant in a fridge?

slide added by
PowerPoint Labs

How do you put an elephant in a fridge?



How long do you need to understand how Addressbook-level1 code works?

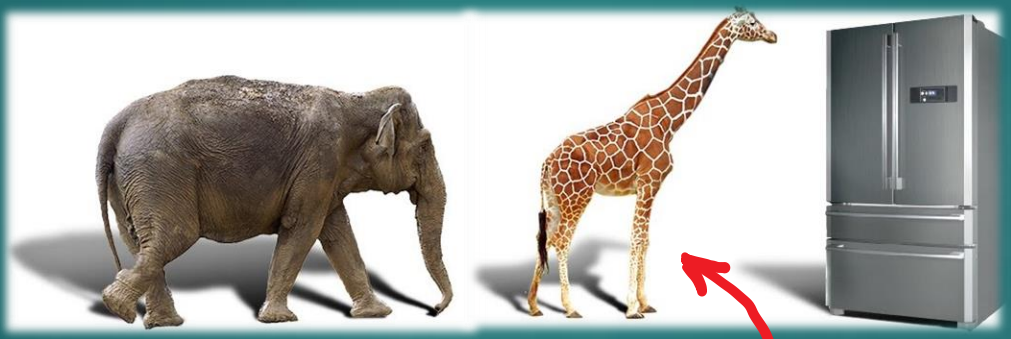
a) Less than 1 minutes

b) Around 10 minutes

c) 10-30 min

d) 30-60 min

e) >60 min



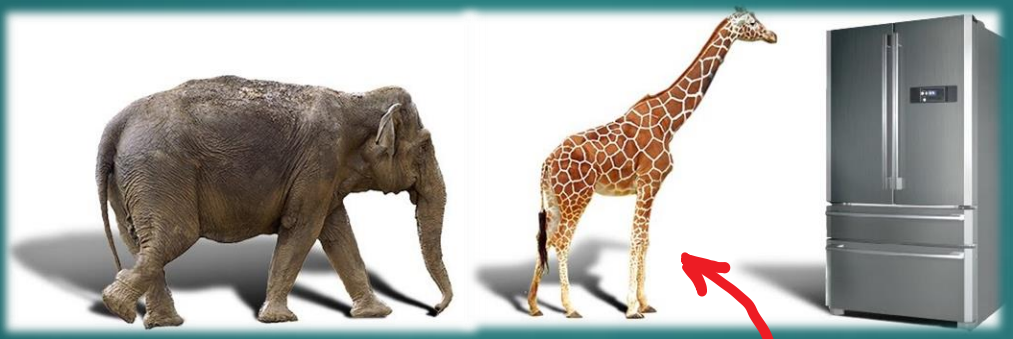
Let us read Addressbook-like code of another student

```
public static void main(String[] args) {
    exitIfIncorrectArguments(args);
    setEnvironment(args);
    printWelcomeMessage();
    executeCommandsUntilExitCommand();
}
```

ABSTRACTION

[capture concepts, ignore details]

slide added by PowerPoint Labs



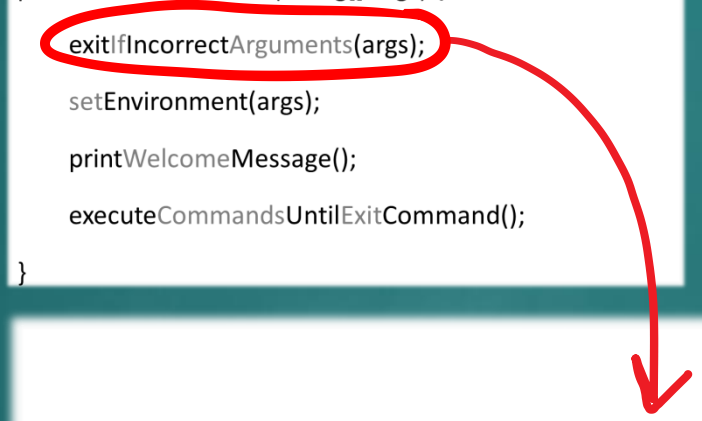
Let us read Addressbook-like code of another student

```
public static void main(String[] args) {
    exitIfIncorrectArguments(args);
    setEnvironment(args);
    printWelcomeMessage();
    executeCommandsUntilExitCommand();
}
```

ABSTRACTION

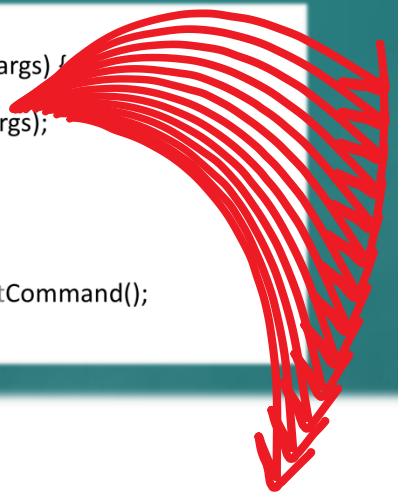
[capture concepts, ignore details]

```
public static void main(String[] args) {
    exitIfIncorrectArguments(args);
    setEnvironment(args);
    printWelcomeMessage();
    executeCommandsUntilExitCommand();
}
```



```
private static void exitIfIncorrectArguments(String[] args) {

}
```

```
public static void main(String[] args) {
    exitIfIncorrectArguments(args);
    setEnvironment(args);
    printWelcomeMessage();
    executeCommandsUntilExitCommand();
}
```



```
private static void exitIfIncorrectArguments(String[] args) {



}
```

slide added by PowerPoint Labs

```
public static void main(String[] args) {
    exitIfIncorrectArguments(args);
    setEnvironment(args);
    printWelcomeMessage();
    executeCommandsUntilExitCommand();
}
```

```
private static void exitIfIncorrectArguments(String[] args) {
    exitIfNoArguments(args);
    exitIfUnacceptableFileName(args);
}
```

```
private static void exitIfNoArguments(String[] args) {
    if (args.length==0) {
        stopWithErrorMessage("No arguments");
    }
}
```

```
private static void exitIfUnacceptableFileName(String[] args) {
    String fileName = getFileName(args);
    if (!isFileNameAcceptable(fileName)) {
        stopWithErrorMessage("Incorrect file name");
    }
}
```

```
public static void main(String[] args) {
        exitIfIncorrectArguments(args);
        setEnvironment(args);
        printWelcomeMessage();
        executeCommandsUntilExitCommand();
}
```

```
private static void exitIfIncorrectArguments(String[] a
        exitIfNoArguments(args);
        exitIfUnacceptableFileName(a
}
```

```
private static void exitIfNoArguments(String[] args) {
    if (args.length==0) {
        stopWithErrorMessage("No arguments");
    }
}
```

```
private static void stopWithErrorMessage(String me
        System.out.println(message);
        System.exit(0);
}
```

LEVELS OF ABSTRACTION

```
public static void main(String[] args) {
        exitIfIncorrectArguments(args);
        setEnvironment(args);
        printWelcomeMessage();
        executeCommandsUntilExitCommand();

}
```

```
private static void exitIfIncorrectArguments(String[] args) {
    exitIfNoArguments(args);
    exitIfUnacceptableFileName(args);

}
```

```
private static void exitIfNoArguments(String[] args) {
    if (args.length==0) {
            stopWithErrorMessage("No arguments");
    }
}
```

```
private static void stopWithErrorMessage(String message) {
    System.out.println(message);
    System.exit(0);
```

```
public static void main(String[] args) {
    if (args.length == 0) {
        System.out.println("No arguments");
        System.exit(0);
    }
    String fileName = args[0];
    if (!isFileNameAcceptable(fileName)) {
        System.out.println("Incorrect file nam
        System.exit(0);
    }
    File f = new File(fileName);
    if (!f.exists()) {
        try {
            f.createNewFile();
        } catch (IOException e) {
            System.out.println("Cannot create f
            System.exit(0);
        }
    }
    System.out.println("Welcome. " + fileNa
    while (!isExitCommand) {
        System.out.print("Enter command:");
        String command = scanner.nextLine();
        command = readLineFromConsole();
        if (isAddCommand(command)) {
```

```
public static void main(String[] args) {
        exitIfIncorrectArguments(args);
        setEnvironment(args);
        printWelcomeMessage();
        executeCommandsUntilExitCommand();

}
```

```
private static void exitIfIncorrectArguments(String[] args) {
    exitIfNoArguments(args);
    exitIfUnacceptableFileName(args);

}
```

```
private static void exitIfNoArguments(String[] args) {
    if (args.length==0) {
        stopWithErrorMessage("No arguments");
    }
}
```

```
private static void stopWithErrorMessage(String message) {
    System.out.println(message);
    System.exit(0);
```

```
public static void main(String[] args) {
    if (args.length == 0) {
        System.out.println("No arguments");
        System.exit(0);
    }
    String fileName = args[0];
    if (!isFileNameAcceptable(fileName)) {
        System.out.println("Incorrect file nam
        System.exit(0);
    }
    File f = new File(fileName);
    if (!f.exists()) {
        try {
            f.createNewFile();
        } catch (IOException e) {
            System.out.println("Cannot create f
            System.exit(0);
        }
    }
    System.out.println("Welcome. " + fileNa
    while (!isExitCommand) {
        System.out.print("Enter command:");
        String command = scanner.nextLine();
        command = readLineFromConsole();
        if (isAddCommand(command)) {
```

```
public static void main(String[] args) {
        exitIfIncorrectArguments(args);
        setEnvironment(args);
        printWelcomeMessage();
        executeCommandsUntilExitCommand();

}
```

```
private static void exitIfIncorrectArguments(String[] args) {
    exitIfNoArguments(args);
    exitIfUnacceptableFileName(args);

}
```

```
private static void exitIfNoArguments(String[] args) {
    if (args.length==0) {
        stopWithErrorMessage("No arguments");
    }
}
```

```
private static void stopWithErrorMessage(String message) {
    System.out.println(message);
    System.exit(0);
}
```

```
public static void main(String[] args) {
    if (args.length == 0) {
        System.out.println("No arguments");
        System.exit(0);
    }
    String fileName = args[0];
    if (!isFileNameAcceptable(fileName)) {
        System.out.println("Incorrect file nam
        System.exit(0);
    }
    File f = new File(fileName);
    if (!f.exists()) {
        try {
            f.createNewFile();
        } catch (IOException e) {
            System.out.println("Cannot create f
            System.exit(0);
        }
    }
    System.out.println("Welcome. " + fileNa
    while (!isExitCommand) {
        System.out.print("Enter command:");
        String command = scanner.nextLine();
        command = readLineFromConsole();
        if (isAddCommand(command)) {
```

SLAP

[Single Level of Abstraction Per function]

SLAP (Single Level of Abstraction Per function)

```
public static void main(String[] args) {
        exitIfIncorrectArguments(args);
        setEnvironment(args);
        printWelcomeMessage();
        executeCommandsUntilExitCommand();

}
```

```
private static void exitIfIncorrectArguments(String[] args) {
    exitIfNoArguments(args);
    exitIfUnacceptableFileName(args);

}
```

```
private static void exitIfNoArguments(String[] args) {
    if (args.length==0) {
        stopWithErrorMessage("No arguments");
    }
}
```

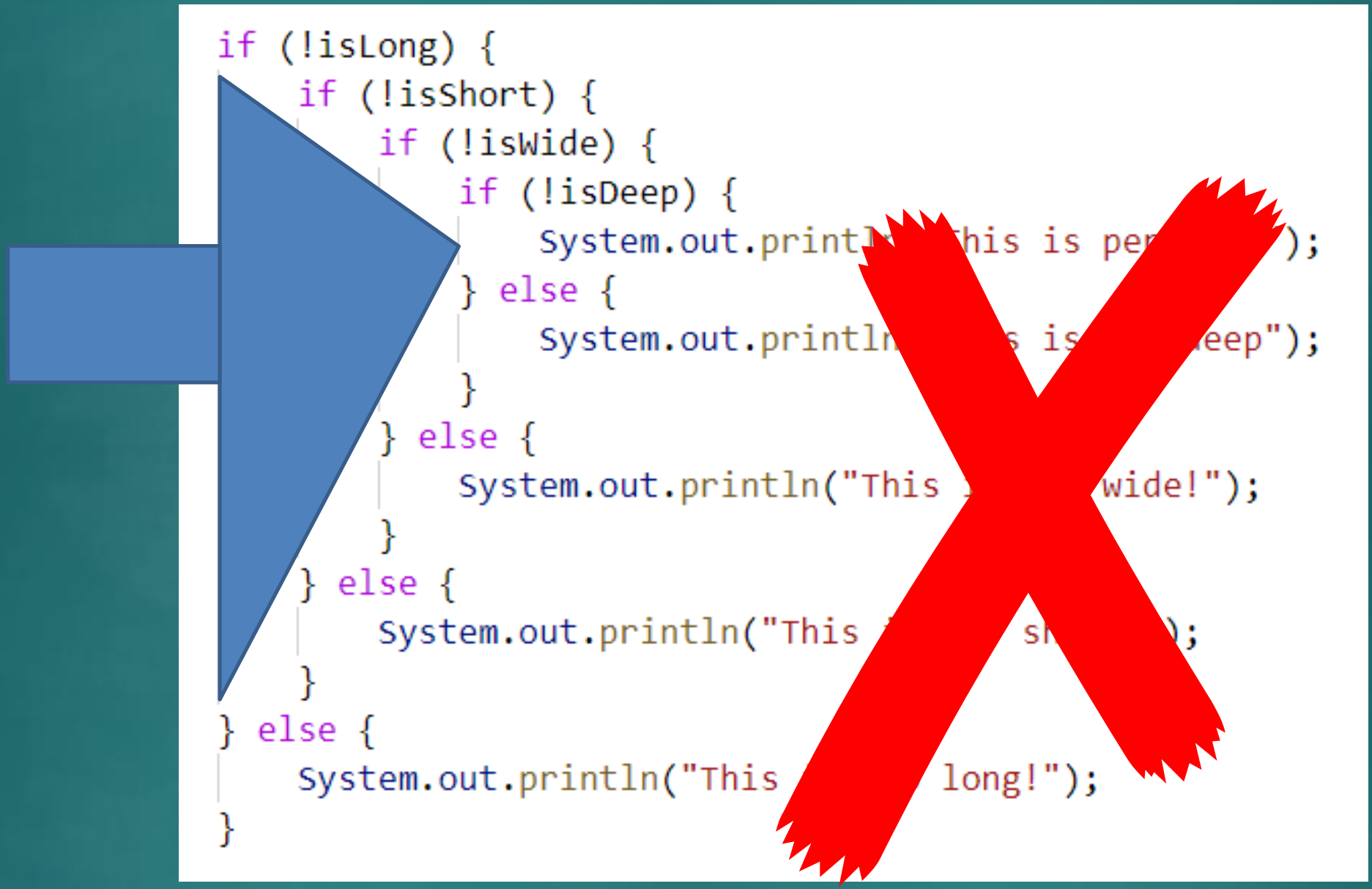
```
private static void stopWithErrorMessage(String message) {
    System.out.println(message);
    System.exit(0);
```

No SLAP

```
public static void main(String[] args) {

    exitIfIncorrectArguments(args);

    File f = new File(fileName);
    if (!f.exists()) {
        try {
            f.createNewFile();
        } catch (IOException e) {
            System.out.println(
                    "Cannot create file!");
            System.exit(0);
        }
    }

    printWelcomeMessage();

    executeCommandsUntilExitCommand()
}
```

Practices for better quality code

6. Exploit abstraction (SLAP)
7. ……
8. ……
9. ……
10. ……


if (!isLong) {
if (!isShort) {
if (!isWide) {
if (!isDeep) {
} else {
}
} else {
}
} else {
}
} else {
}

```
if (isLong) {
    System.out.println("This is too long!");
} else if (isShort) {
    System.out.println("This is too short!");
} else if (isWide) {
    System.out.println("This is too wide!");
} else if (isDeep) {
    System.out.println("This is too deep");
} else {
    System.out.println("This is perfe
}
```

Practices for better quality code

6. Exploit abstraction (SLAP)
7. Avoid deep nesting
8. ……
9. ……
10. ……

```
ArrayList<String> getSortedList(ArrayList<String> listToSort) {
    listToSort = Collections.sort(listToSort);
    return listToSort;
}
```

```
try {
    // something
} catch {

}
```

Minimize the scope of the variables

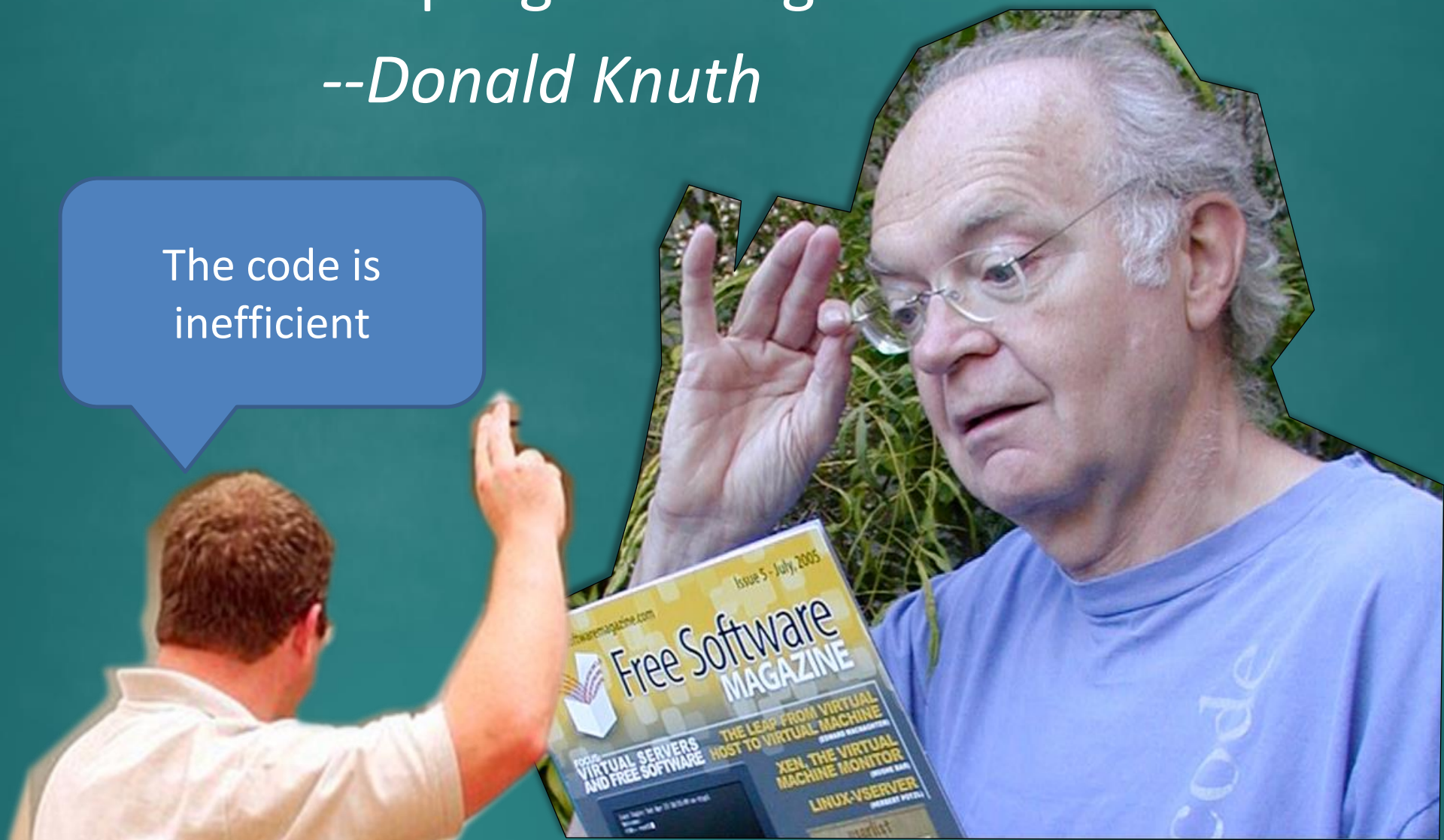
```
void  foo() {
    . . .
    if (isBar) {
        int i = 0;
        // do something with i
    }
    . . .
}
```

Practices for better quality code

6. Exploit abstraction (SLAP)
7. Avoid deep nesting
8. Avoid error prone shortcuts!
9. ……
10. ……

Premature optimization is the root of all evil in programming.

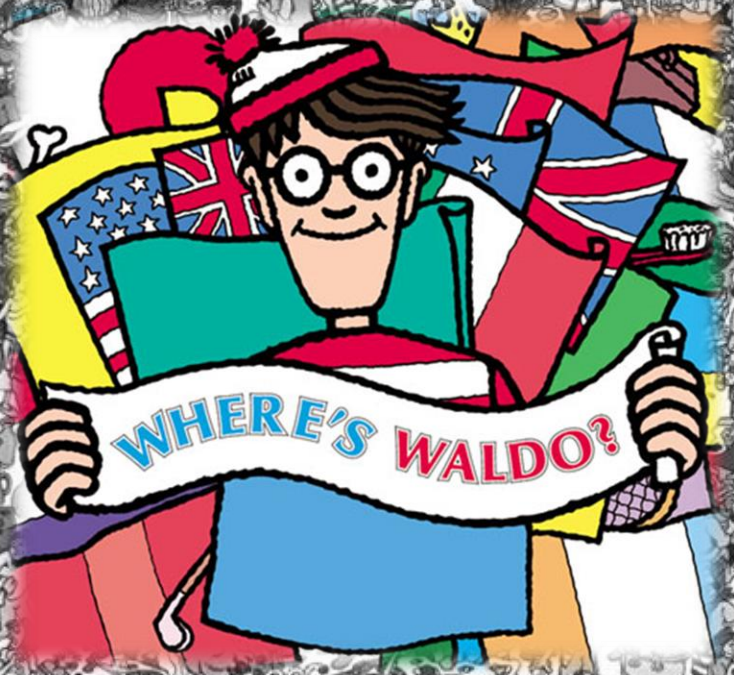
--Donald Knuth



Reason 1: it might not be needed

Reason 2: we don't know where

Reason 2: we don't know where

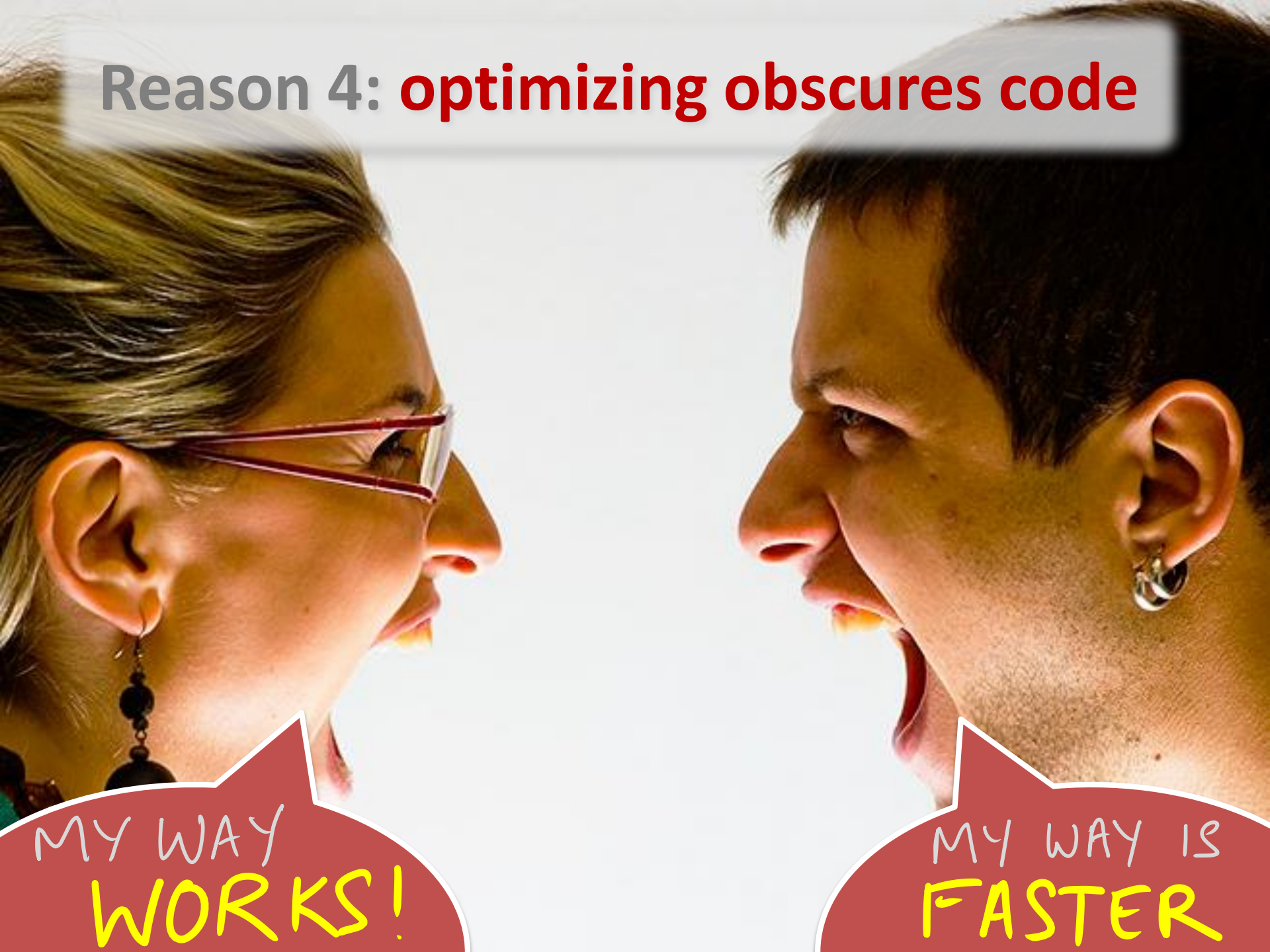


Profiler Tools

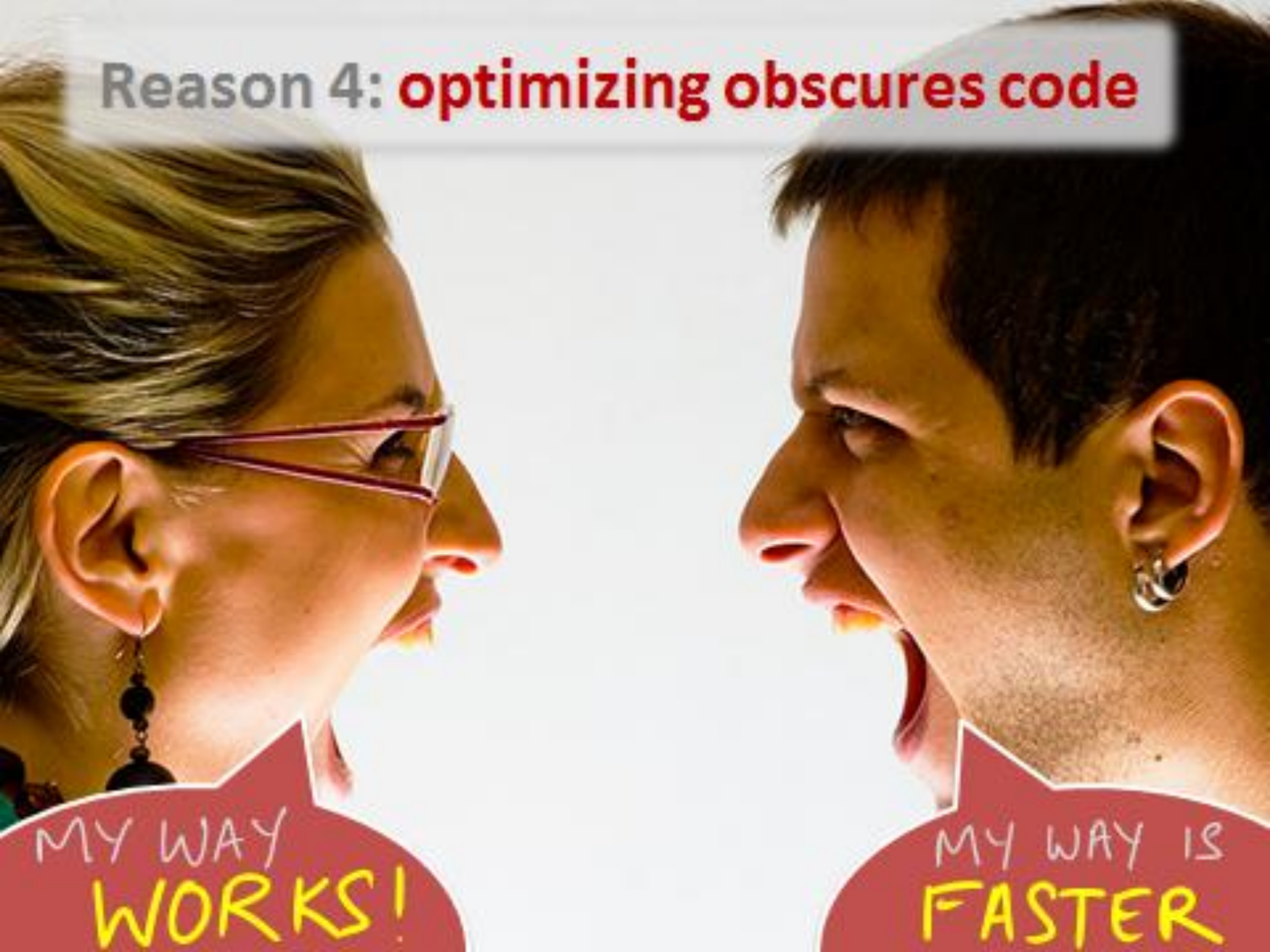
[tools that find actual performance bottlenecks]

Reason 3: compilers can optimize

Reason 4: optimizing obscures code



Reason 4: optimizing obscures code



Reason 4: optimizing obscures code

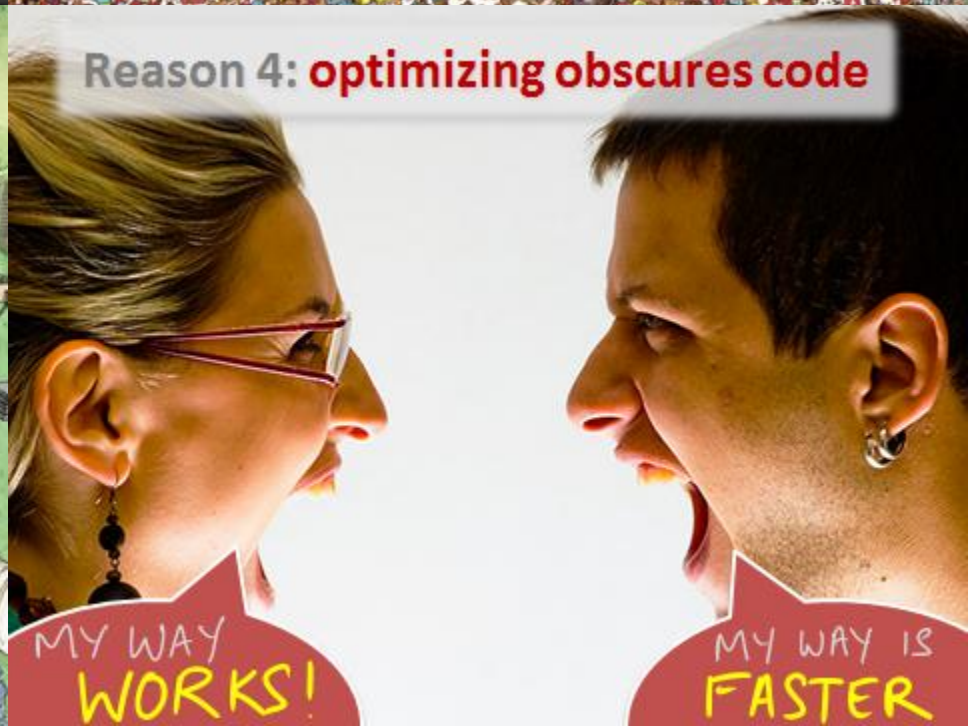
slide added by PowerPoint Labs

Reason 1: it might not be needed

Reason 2: we don't know where

Reason 3: compilers can optimize

Reason 4: optimizing obscures code



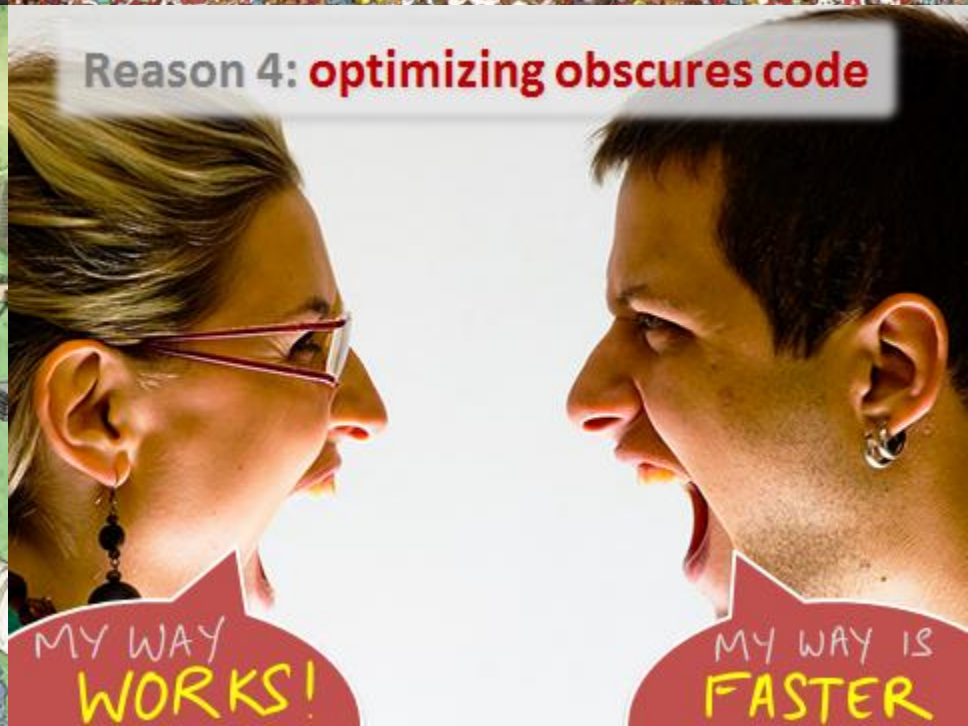
Reason 1: it might not be needed

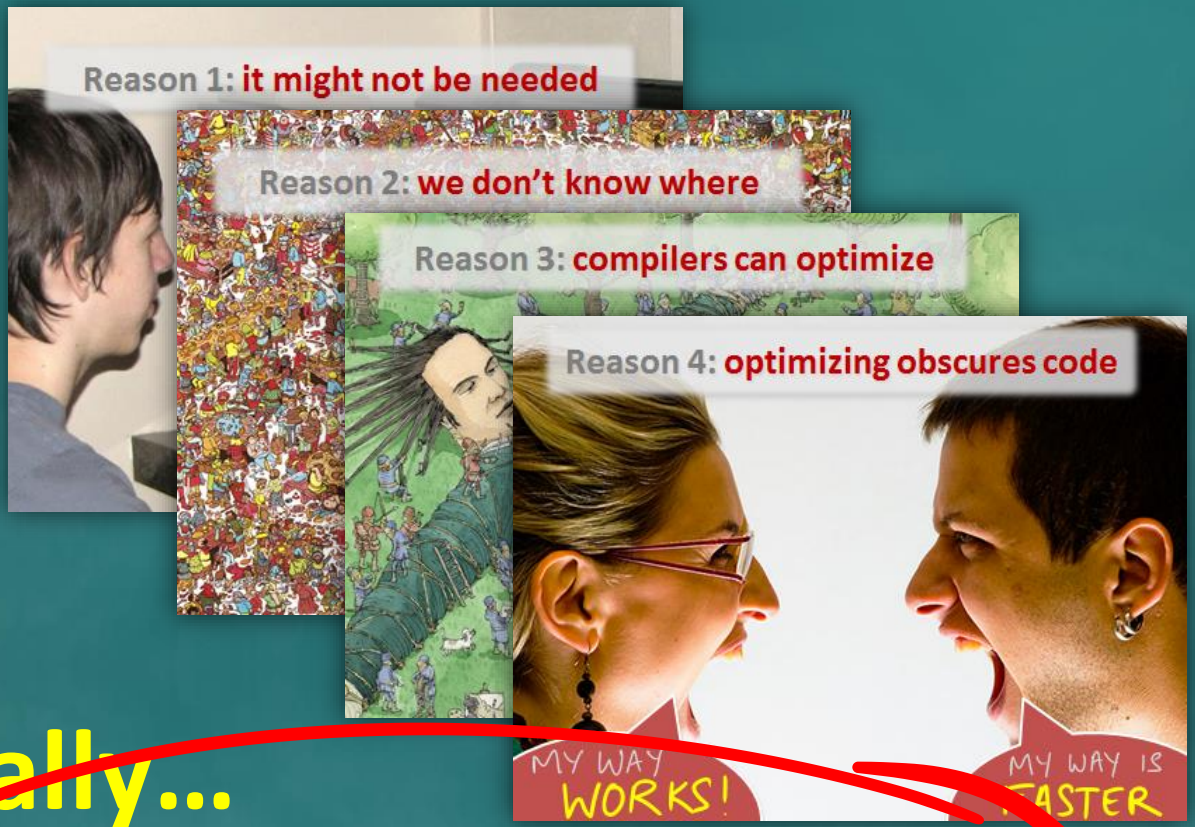
Reason 2: we don't know where

slide added by
PowerPoint Labs

Reason 3: compilers can optimize

Reason 4: optimizing obscures code





Typically…

Make it **work**

Make it **right**

Make it **fast**


Make it **work**
Make it **right**
Make it **fast**

Make it work

Make it right

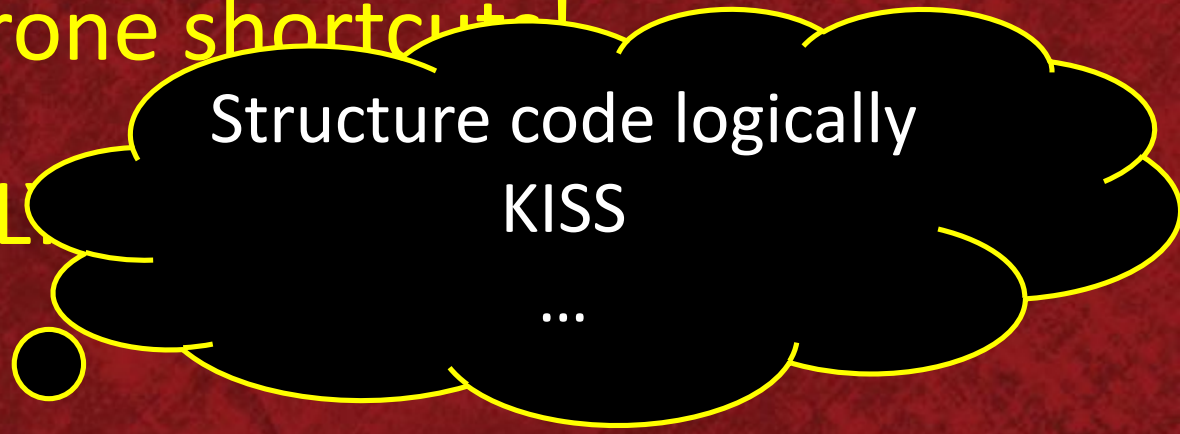
Make it fast

Practices for better quality code

6. Exploit abstraction (SLAP)
7. Avoid deep nesting
8. Avoid error prone shortcuts!
9. Optimize, ONLY if you must
10. ……

Practices for better quality code

6. Exploit abstraction (SLAP)
7. Avoid deep nesting
8. Avoid error prone shortcuts!
9. Optimize, ONLY if necessary!
10. More ….



Identify areas to improve in the code below

```
...
private static final String MESSAGE_COMMAND_HELP_PARAMETERS = "Parameters: %1$s";
private static final String MESSAGE_COMMAND_HELP_EXAMPLE = "Example: %1$s";
private static final String MESSAGE_DISPLAY_PERSON_DATA = "%1$s  Phone Number: %2$s  Email: %3$s";
private static final String GOODBYE_MESSAGE = "Exiting Address Book... Good bye!";
private static final String MESSAGE_INVALID_COMMAND_FORMAT = "Invalid command format: %1$s";
...
/** List of all persons in the address book. */
private static final ArrayList<String> person = new ArrayList<>();
...
public static void main(String[] args) {
    showWelcomeMessage();
    processProgramArgs(args);
    loadDataFromStorage();
    while (true) {
        System.out.print("Enter command: ");
        String userCommand = SCANNER.nextLine();
        userCommand = userCommand.trim();
        showToUser(userCommand);
        String feedback = executeCommand(userCommand);
        showResultToUser(feedback);
    }
}
...
/**
 * Show a message to the user
 */
private static void showToUser(String message) {
  System.out.println(LINE_PREFIX + m);
}
```

Identify areas to improve in the code below

```
...
private static final String MESSAGE_COMMAND_HELP_PARAMETERS = "Parameter
private static final String MESSAGE_COMMAND_HELP_EXAMPLE = "Example: %1$
private static final String MESSAGE_DISPLAY_PERSON_DATA = "%1$s  Phone N
private static final String GOODBYE_MESSAGE = "Exiting Address Book... G
private static final String MESSAGE_INVALID_COMMAND_FORMAT = "Invalid co
...
/** List of all persons in the address book. */
private static final ArrayList<String> person = new ArrayList<>();
...
public static void main(String[] args) {
    showWelcomeMessage();
    processProgramArgs(args);
    loadDataFromStorage();
    while (true) {
        System.out.print("Enter command: ");
        String userCommand = SCANNER.nextLine();
        userCommand = userCommand.trim();
        showToUser(userCommand);
```

Identify areas to improve in the code below

```
public static void main(String[] args) {
    showWelcomeMessage();
    processProgramArgs(args);
    loadDataFromStorage();
    while (true) {
        String userCommand = readUserCommand();
        System.out.print("Enter command: ");
        String userCommand = SCANNER.nextLine();
        userCommand = userCommand.trim();
        showToUser(userCommand);
        String feedback = executeCommand(userCommand);
        showResultToUser(feedback);
    }
}
...
/**
 * Show a message to the user
 */
private static void showToUser(String message) {
```

Higher level

Same level

Identify areas to improve in the code below

```
        processProgramArgs(args);
        loadDataFromStorage();
        while (true) {
            System.out.print("Enter command: ");
            String userCommand = SCANNER.nextLine();
            userCommand = userCommand.trim();
            showToUser(userCommand);
            String feedback = executeCommand(userCommand);
            showResultToUser(feedback);
        }
    }
    ...
    /**  Shows the message to the user
     * Show a message to the user
     */
    private static void showToUser(String message) {
        System.out.println(LINE_PREFIX + m);
    }
```

Identify areas to improve in the code below

```
            String userCommand = SCANNER.nextLine();
            userCommand = userCommand.trim();
            showToUser(userCommand);
            String feedback = executeCommand(userCommand);
            showResultToUser(feedback);
        }
    }
    ...
    /**
     * Show a message to the user
     */
    private static void showToUser(String message) {
        System.out.println(LINE_PREFIX + m);
    }
```

Answer on Archipelago

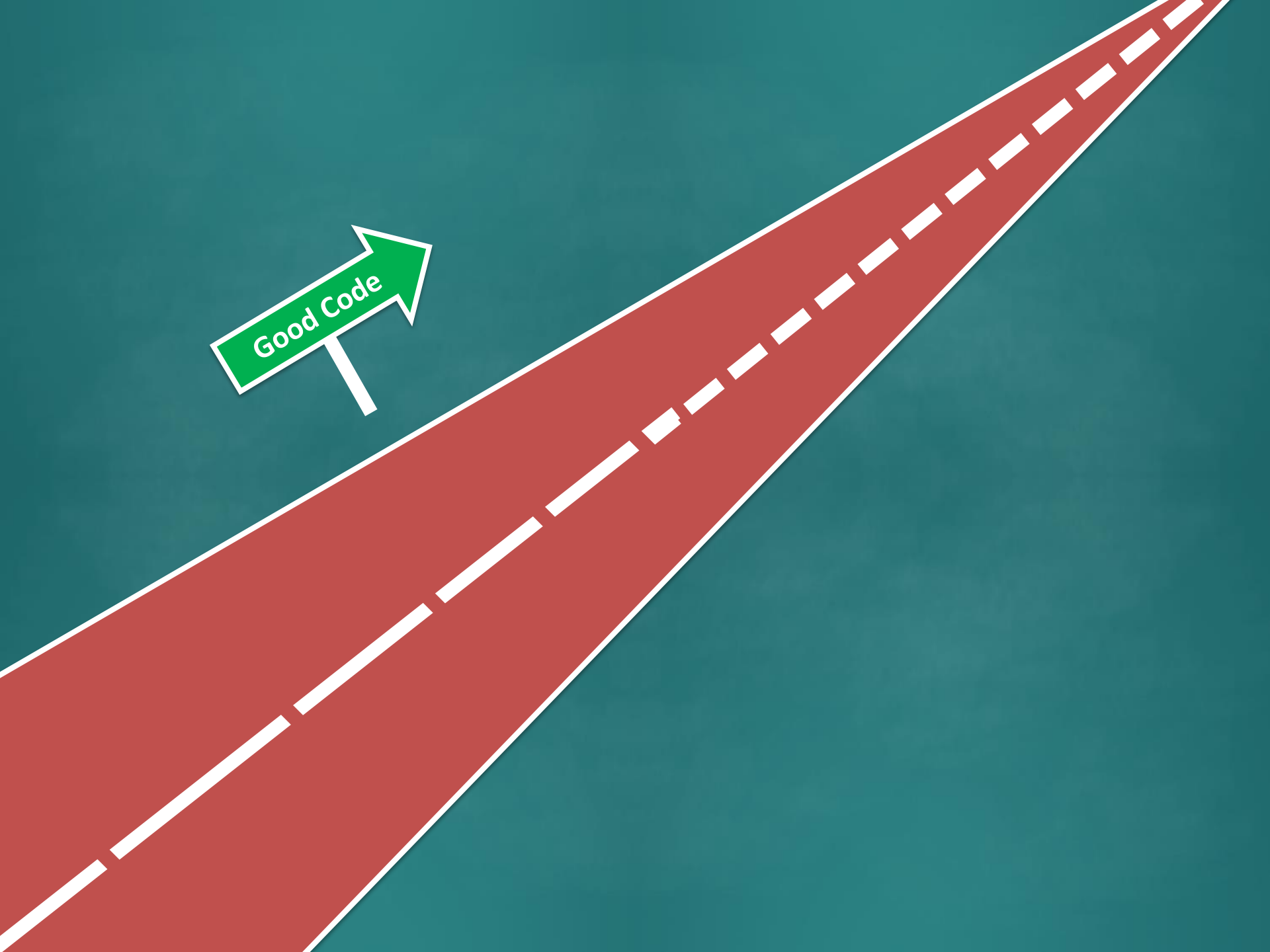
Intention

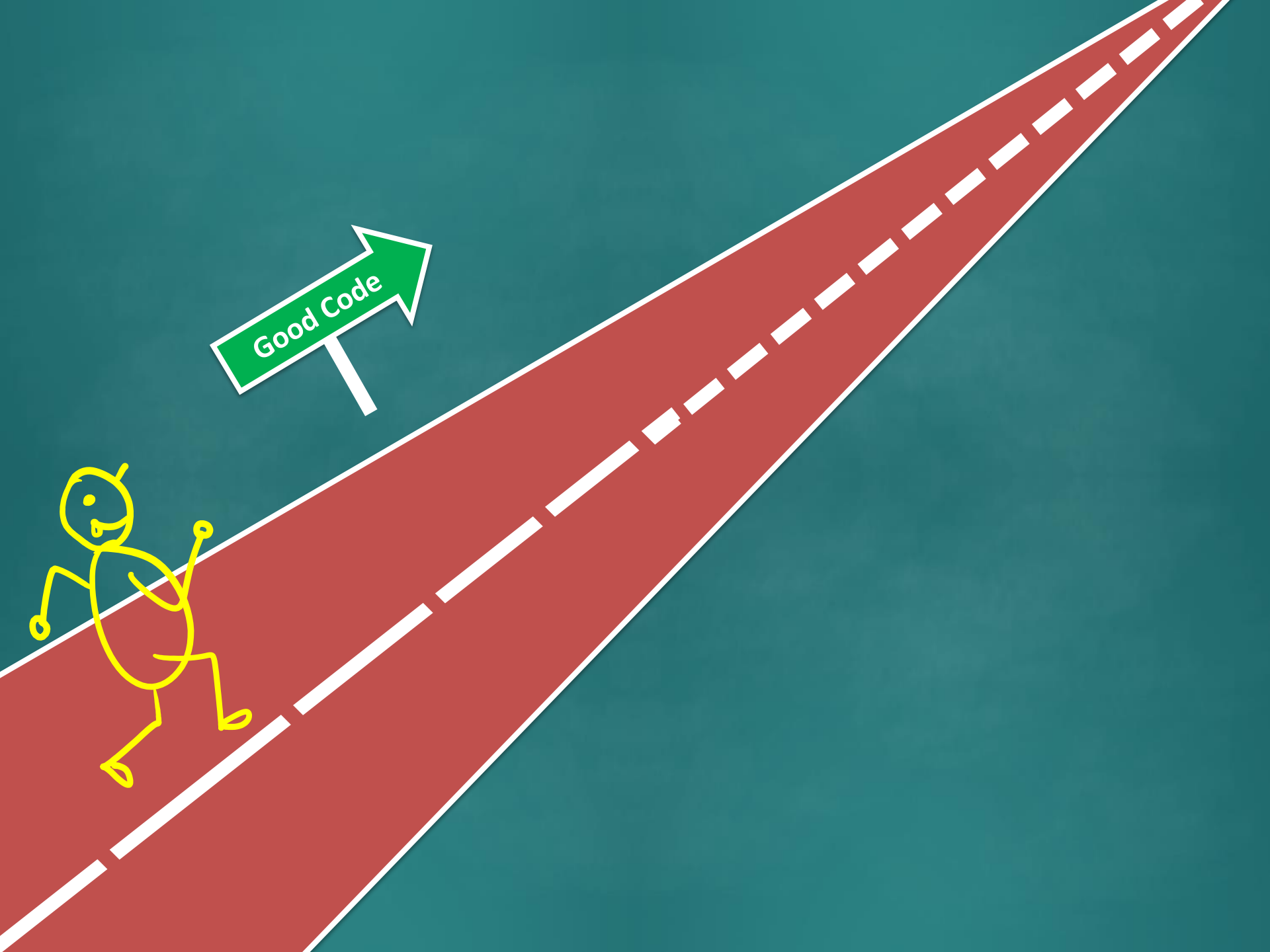
Mechanism

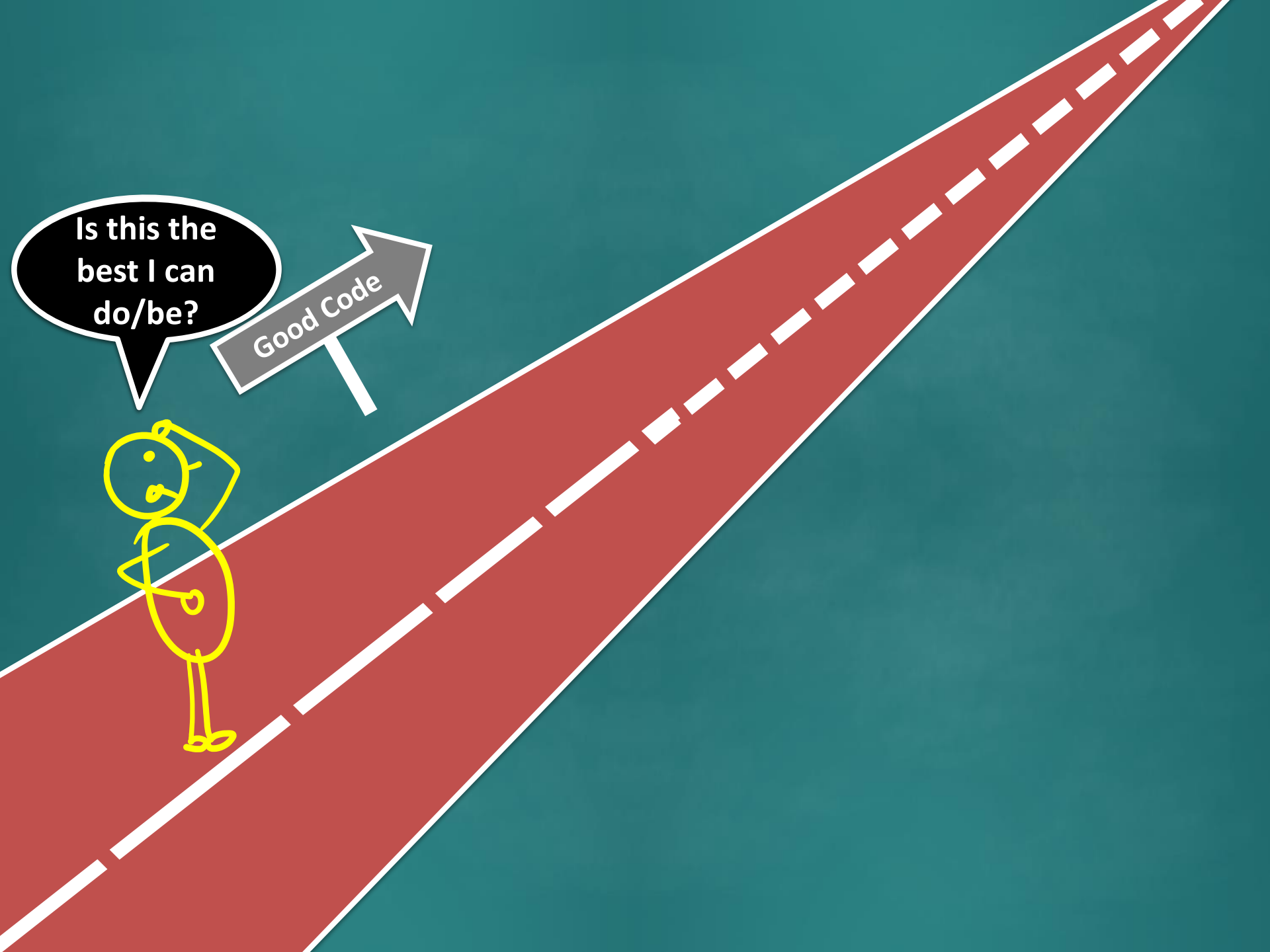
Too much abstraction?

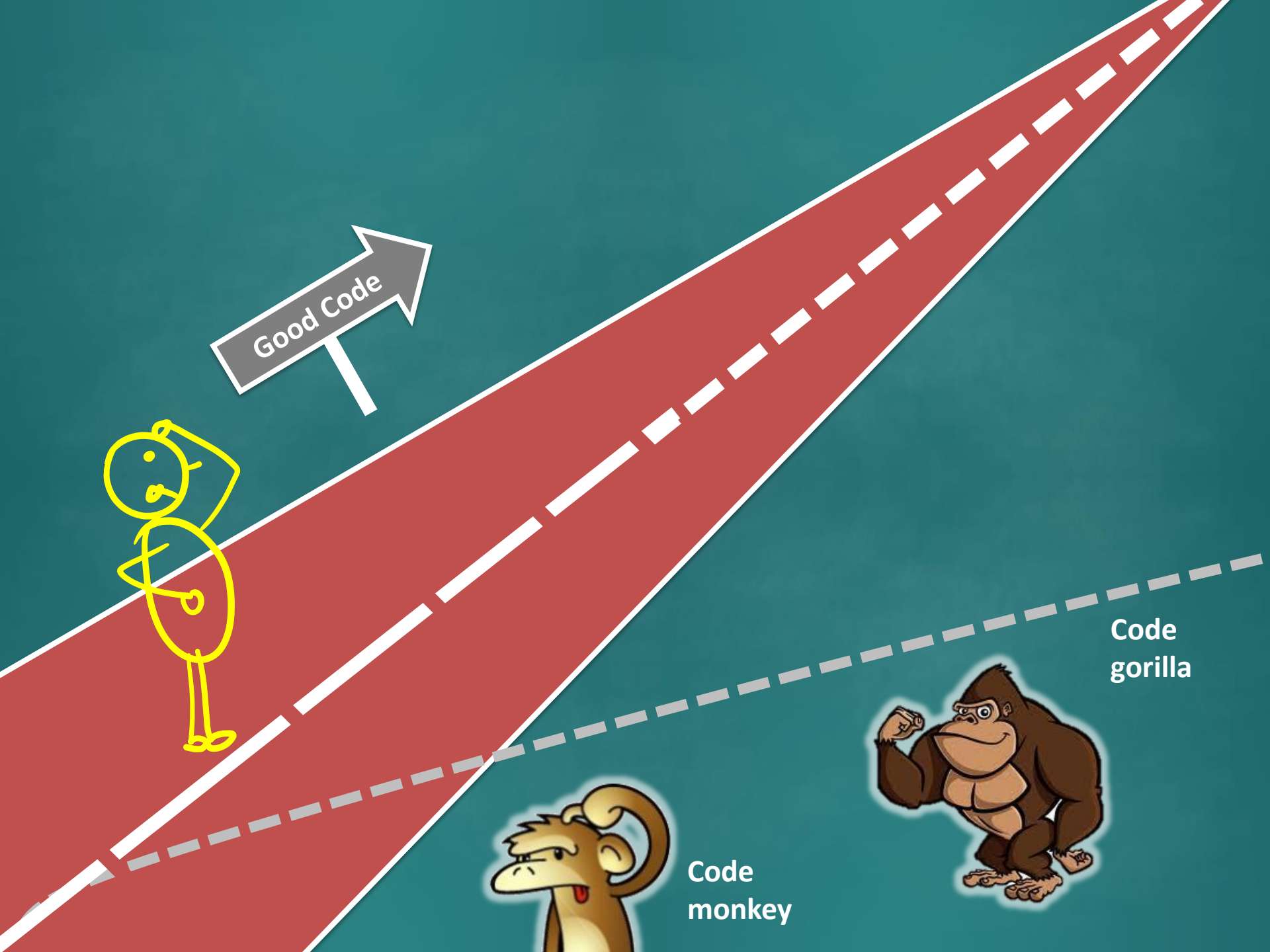
Good code looks neat.

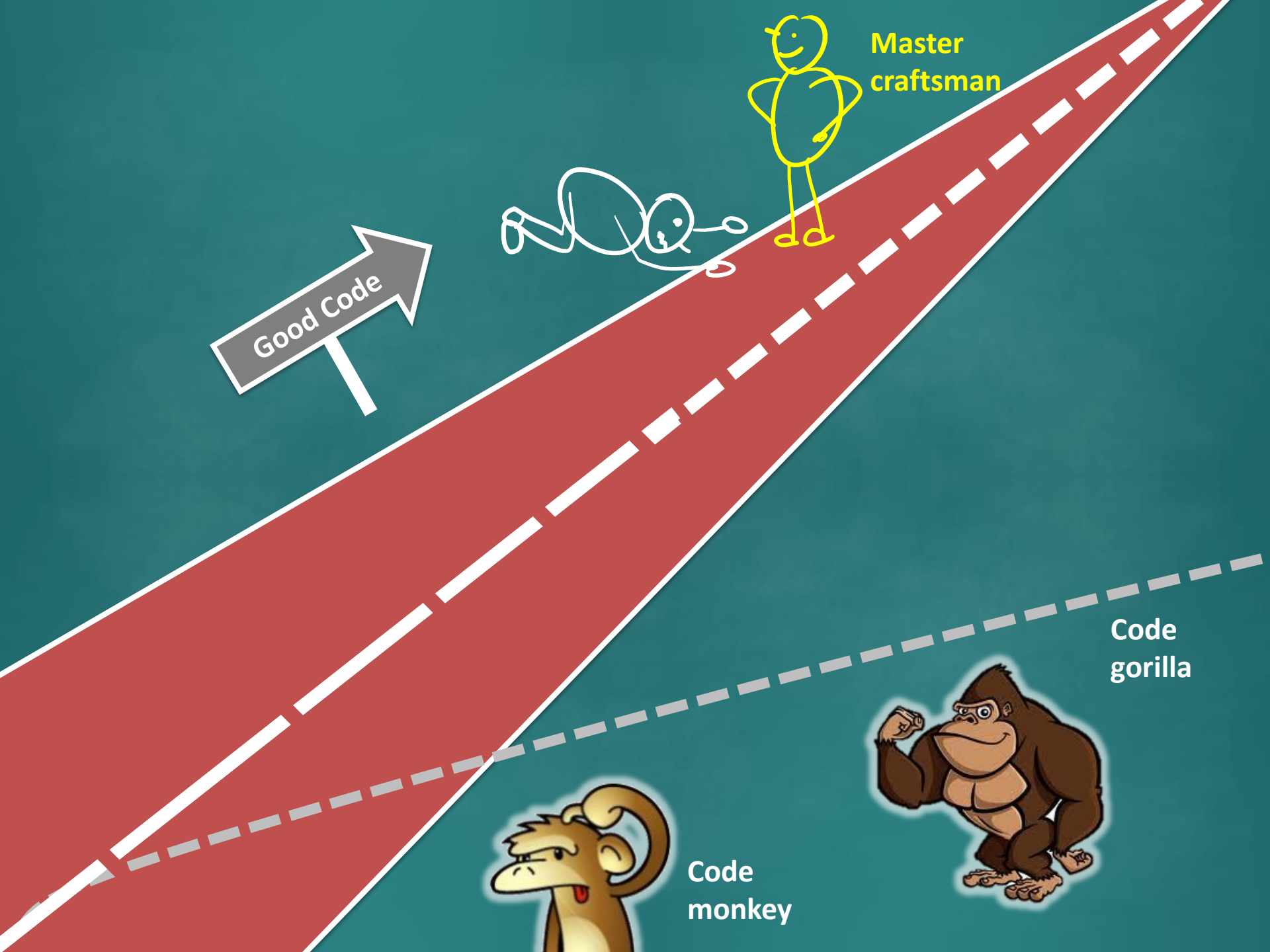
```
47
48      /**
49       * Constructs a TestRunner using the given ResultPrinter all the output
50       */
51      public TestRunner(ResultPrinter printer) {
52              fPrinter= printer;
53      }
54
55      /**
56       * Runs a suite extracted from a TestCase subclass.
57       */
58      static public void run(Class<? extends TestCase> testClass) {
59              run(new TestSuite(testClass));
60      }
61
62      /**
63       * Runs a single test and collects its results.
64       * This method can be used to start a test run
65       * from your program.
66       * <pre>
67       * public static void main (String[] args) {
68       *     test.textui.TestRunner.run(suite());
69       * }
70       * </pre>
71       */
72      static public TestResult run(Test test) {
73              TestRunner runner= new TestRunner();
74              return runner.doRun(test);
75      }
76
77      /**
78       * Runs a single test and waits until the user
79       * types RETURN.
80       */
81      static public void runAndWait(Test
82              TestRunner
83
84
85
86
```


Good Code


Good Code


Is this the best I can do/be?
Good Code

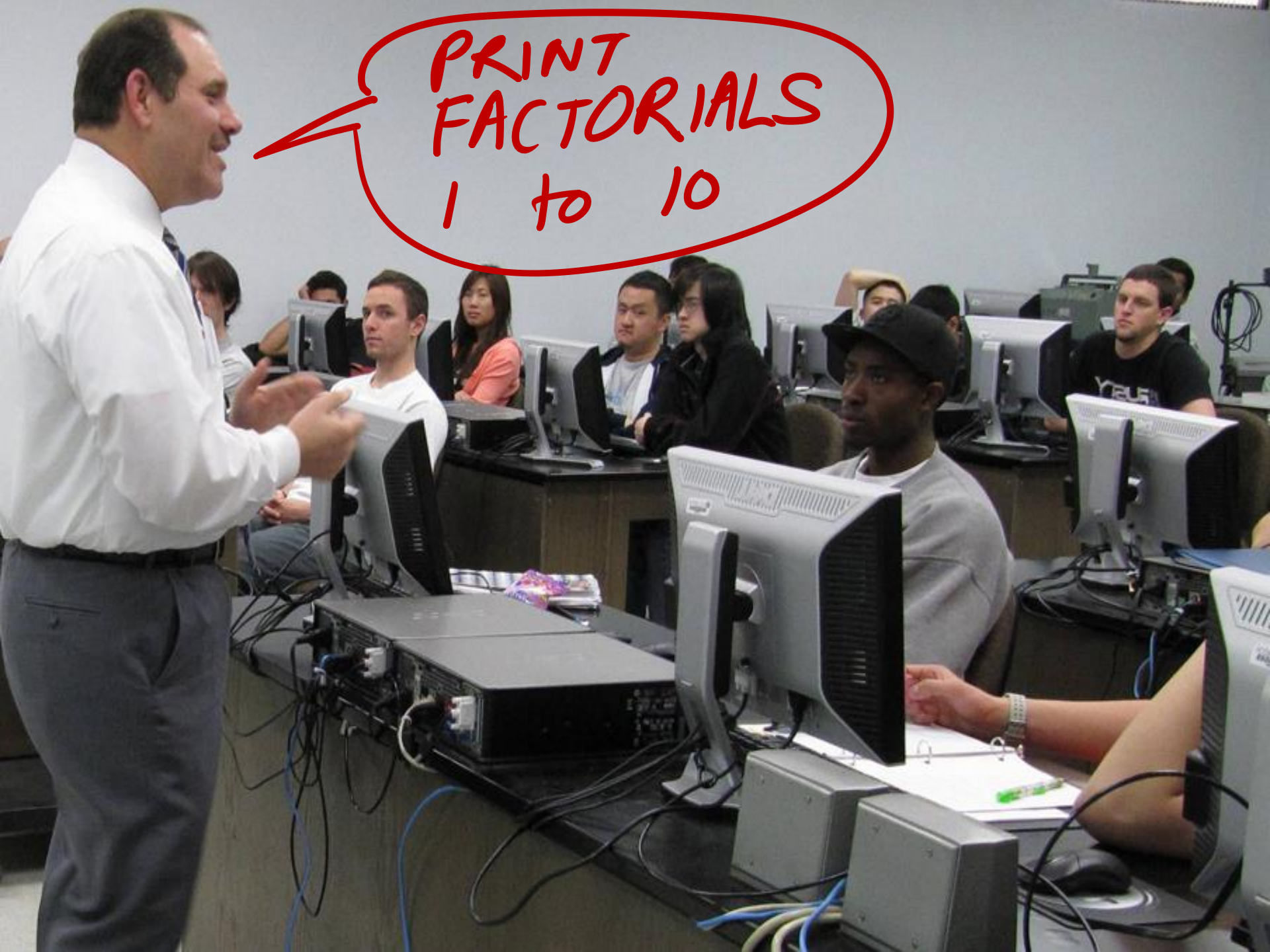

Good Code
Code gorilla
Code monkey


Master craftsman
Good Code
Code gorilla
Code monkey

Week 4 [Feb 4]

Notices | Topics | Project | Tutorial | Admin Info

- [W4.1] Requirements analysis
- [W4.2] CodeQuality
- [W4.3] Exception Handling
- [W4.4] OOP: Classes & Objects
- [W4.5] Java: enumerations
- [W4.6] OOP: Inheritance


PRINT
FACTORIALS
1 to 10


LET'S
BUILD A
DOOHICKEY


ANGRY BIRDS
1 billion downloads


BIRDS

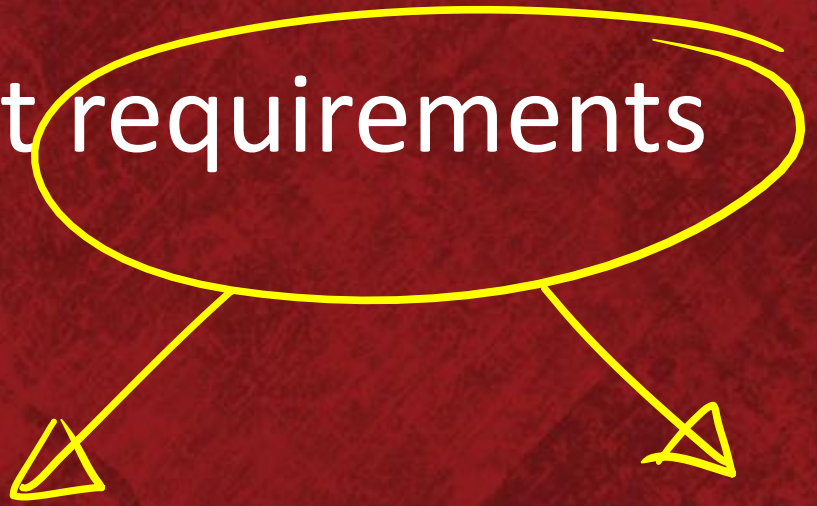
FAIL **FAIL** **WIN**

FAIL

WIN

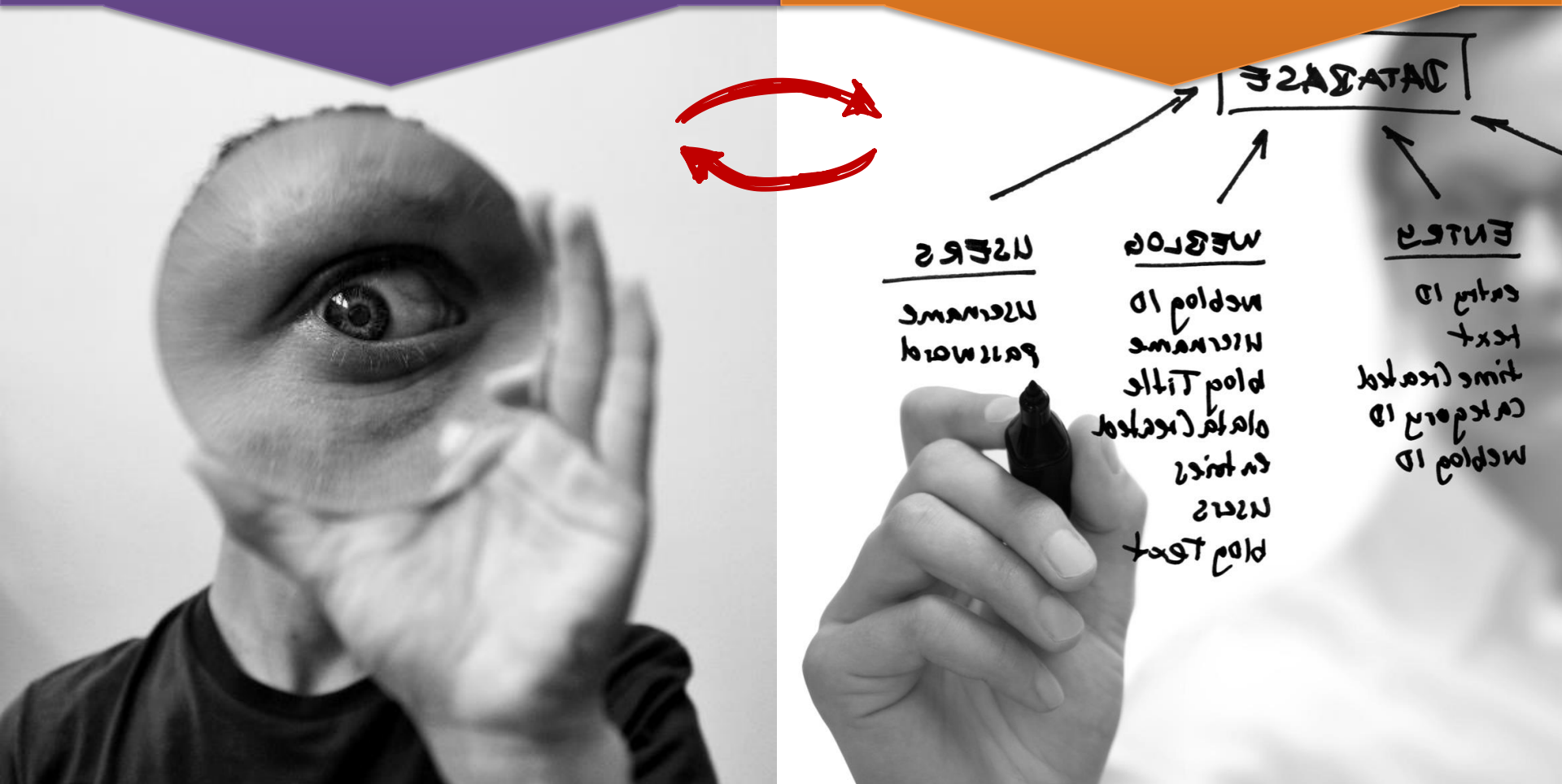
Problem before solution:

figuring out requirements



Establish

specify



ESTABLISHING requirements

ESTABLISHING requirements

If you drop a hammer and a feather from the same height, which one will hit the ground first?

a) Feather first

b) Hammer first

c) Both at the same time

d) Feather will be blown away

If you drop a hammer and a feather from the same height, which one will hit the ground first?

Don’t assume. Ask.

ESTABLISHING requirements

ASK USER

ESTABLISHING requirements

slide added by PowerPoint Labs

ASK USER

User surveys

ESTABLISHING requirements

slide added by PowerPoint Labs

SURVEYS

ESTABLISHING requirements

SURVEYS

ESTABLISHING requirements

SURVEYS

ESTABLISHING requirements

slide added by PowerPoint Labs

SURVEYS

Interview users

ESTABLISHING requirements

slide added by PowerPoint Labs

SURVEYS

INTERVIEWS

ESTABLISHING requirements

SURVEYS

INTERVIEWS

ESTABLISHING requirements

slide added by PowerPoint Labs

SURVEYS

INTERVIEWS

Focus groups

slide added by
PowerPoint Labs

ESTABLISHING requirements

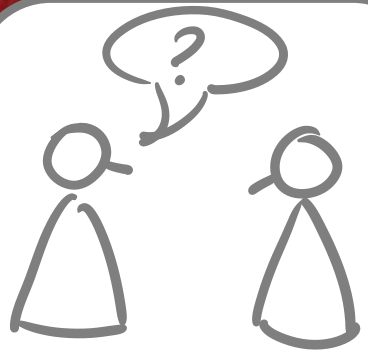
SURVEYS

INTERVIEWS

FOCUS GROUPS

ESTABLISHING requirements

SURVEYS



INTERVIEWS

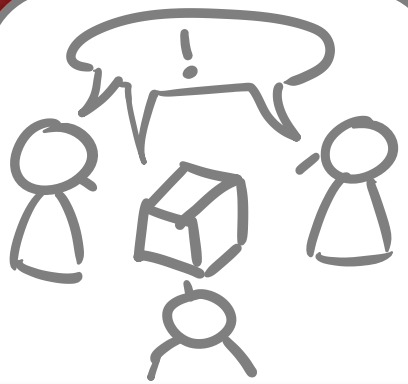
FOCUS GROUPS

ESTABLISHING requirements

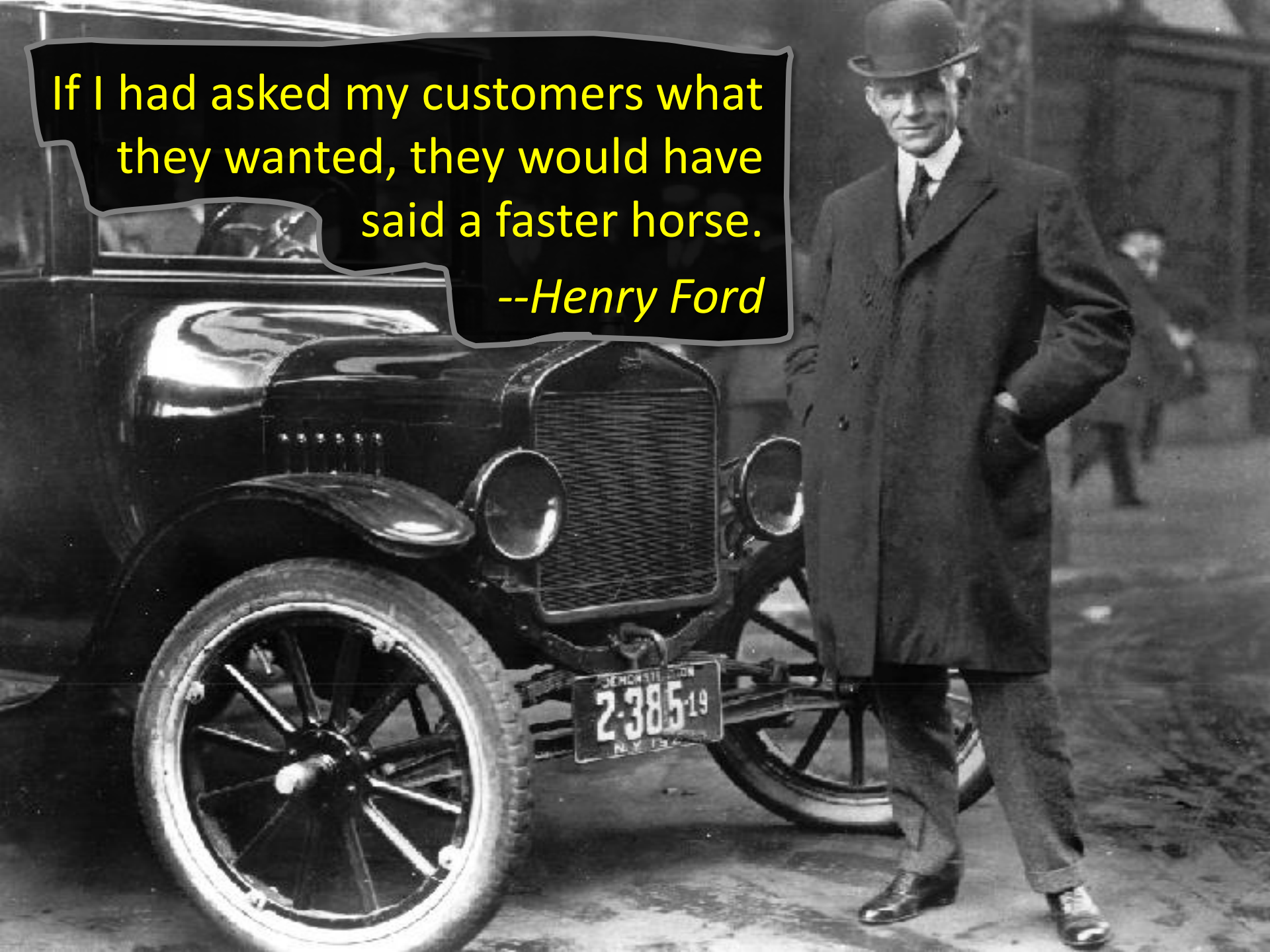
ASK USER

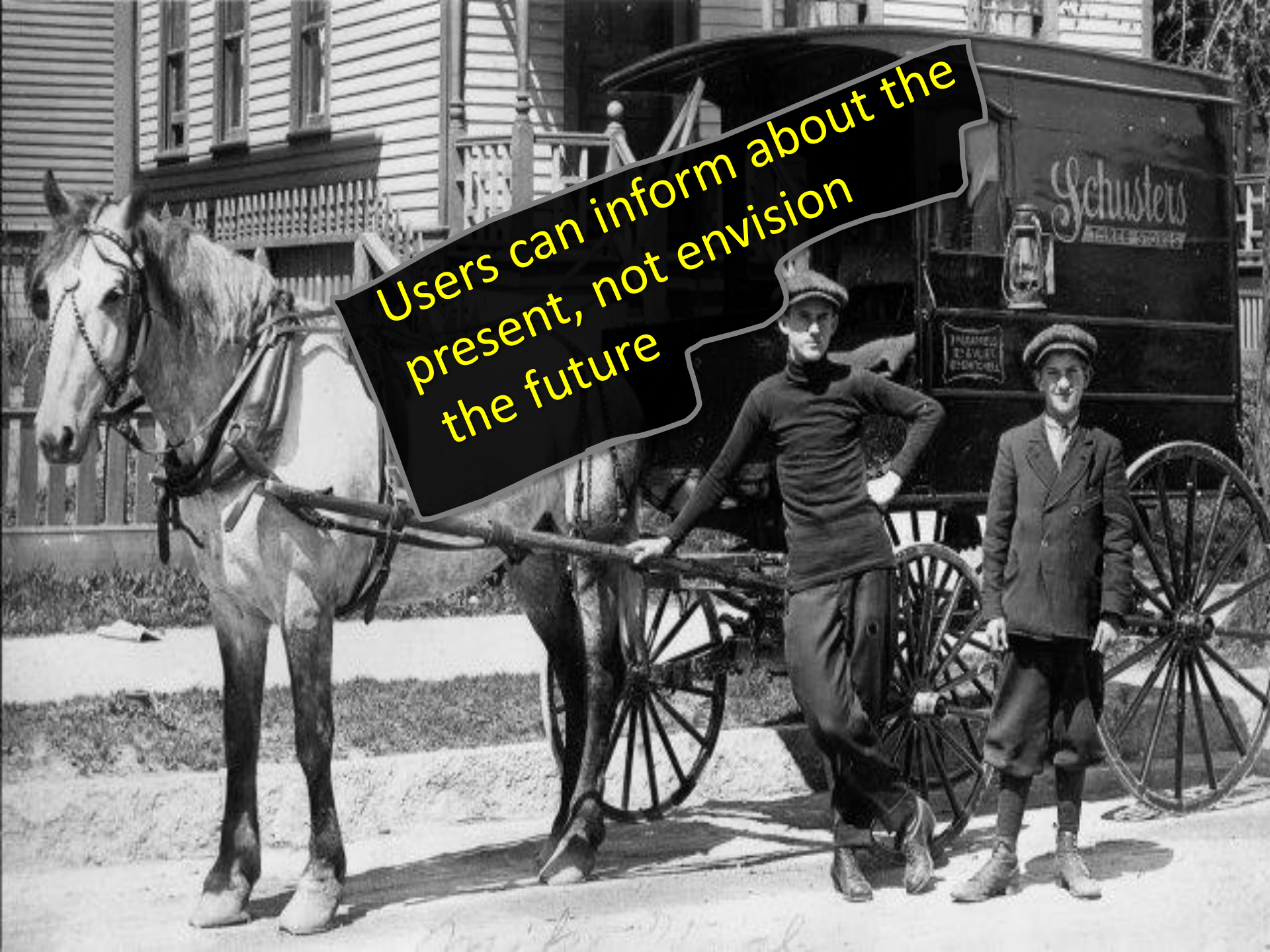
SURVEYS

INTERVIEWS



FOCUS GROUPS


If I had asked my customers what they wanted, they would have said a faster horse.
--Henry Ford


Schusters
Users can inform about the present, not envision the future

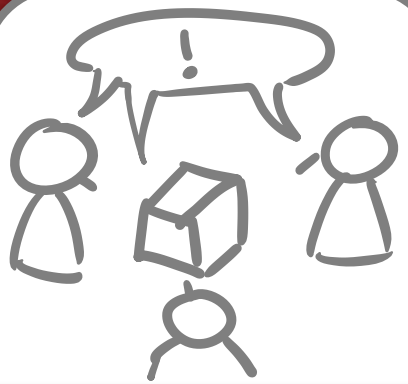

It's not the consumers' job
to know what they want
--Steve Jobs

ESTABLISHING requirements

ASK USER

SURVEYS

INTERVIEWS



FOCUS GROUPS

INVENT YOURSELF

ESTABLISHING requirements

slide added by PowerPoint Labs

ASK USER

SURVEYS

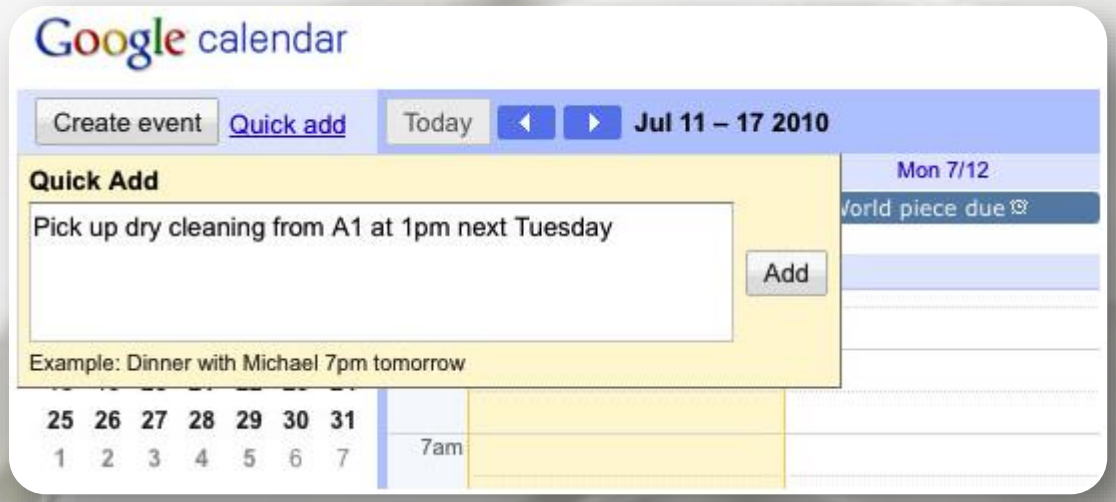
INTERVIEWS

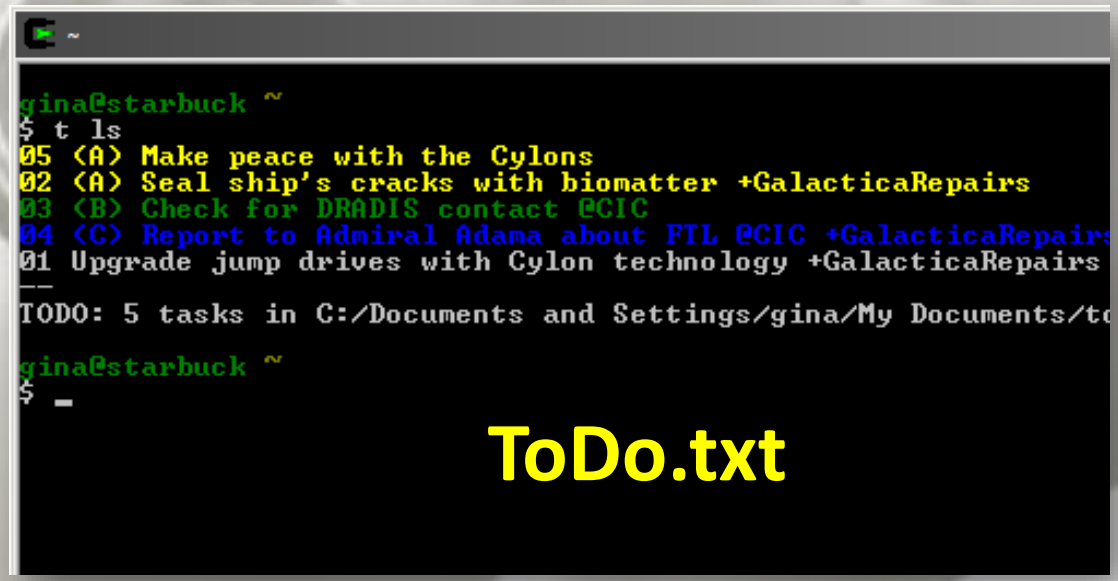
FOCUS GROUPS

INVENT YOURSELF

Study existing products

Study existing products





ToDo.txt

slide added by PowerPoint Labs

ESTABLISHING requirements

ASK USER

INVENT YOURSELF

SURVEYS

INTERVIEWS

FOCUS GROUPS

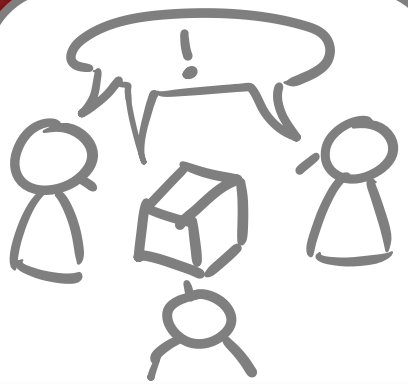
PROD. STUDY

ESTABLISHING requirements

ASK USER

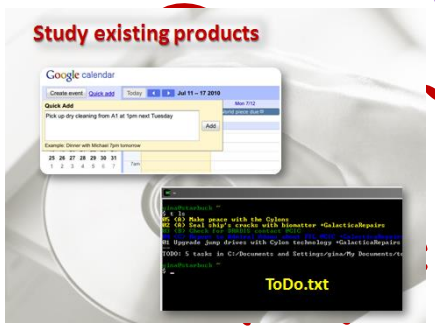
SURVEYS

INTERVIEWS



FOCUS GROUPS

INVENT YOURSELF



PROD. STUDY

ESTABLISHING requirements

slide added by PowerPoint Labs

ASK USER

SURVEYS

INTERVIEWS

FOCUS GROUPS

INVENT YOURSELF

PROD. STUDY

Observe users

slide added by PowerPoint Labs

ESTABLISHING requirements

ASK USER

INVENT YOURSELF

SURVEYS

INTERVIEWS

FOCUS GROUPS

PROD. STUDY

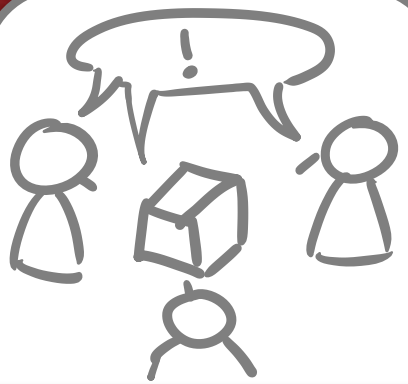
OBSERVE

ESTABLISHING requirements

ASK USER

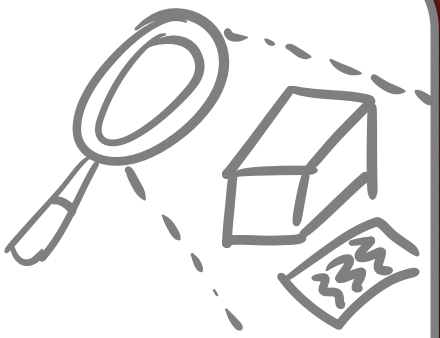
SURVEYS

INTERVIEWS



FOCUS GROUPS

INVENT YOURSELF



PROD. STUDY

OBSERVE

ESTABLISHING requirements

slide added by PowerPoint Labs

ASK USER

INVENT YOURSELF

SURVEYS

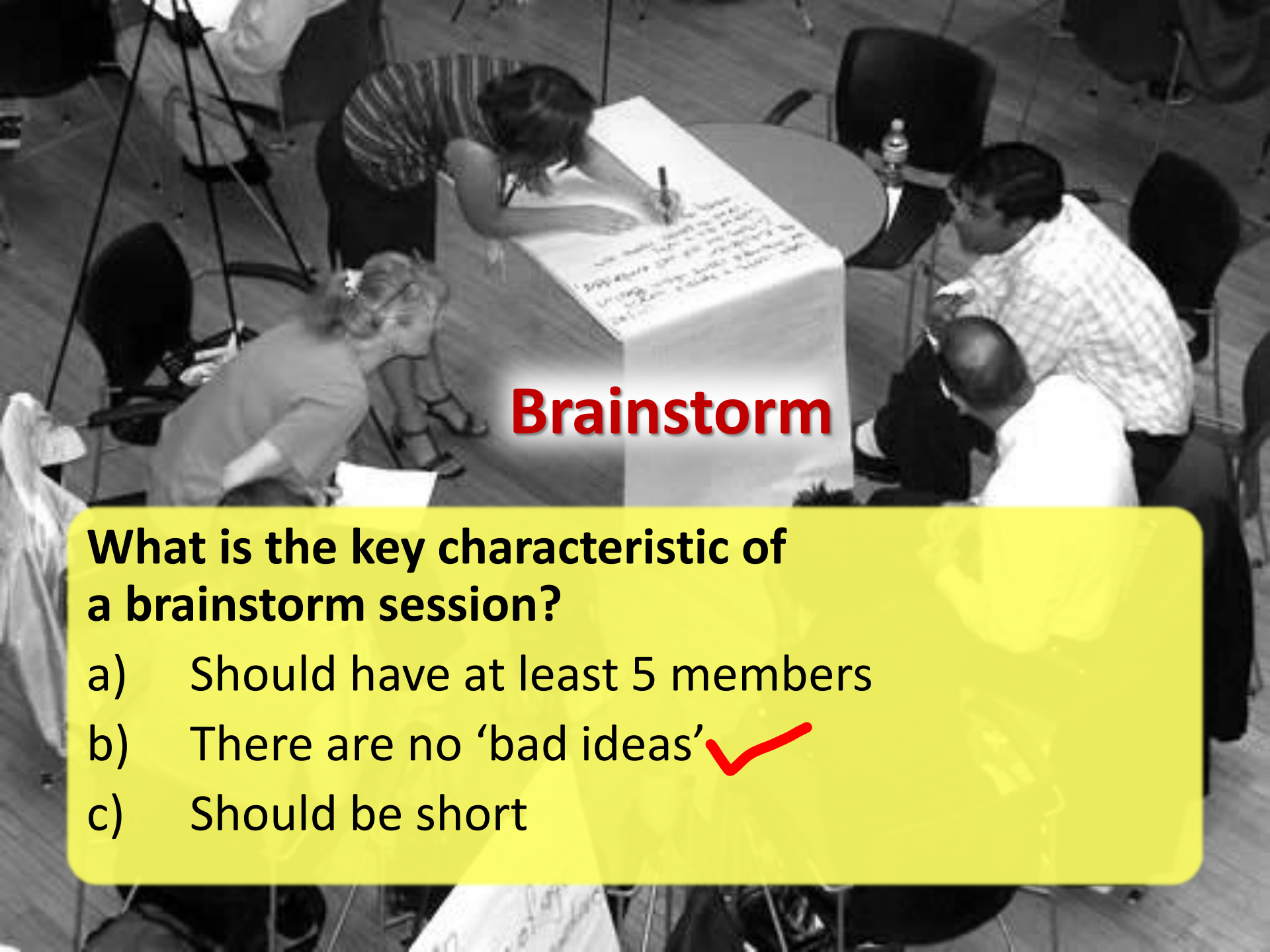
INTERVIEWS

FOCUS GROUPS

PROD. STUDY

OBSERVE


Brainstorm



Brainstorm

What is the key characteristic of a brainstorm session?

a) Should have at least 5 members

b) There are no ‘bad ideas’ ✓

c) Should be short

slide added by PowerPoint Labs

ESTABLISHING requirements

ASK USER

SURVEYS

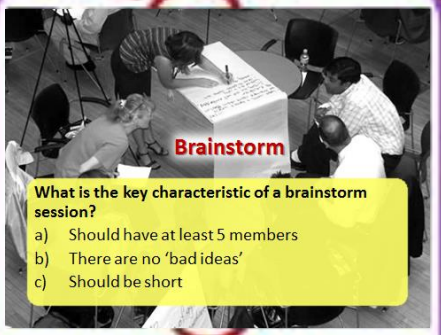
INTERVIEWS

FOCUS GROUPS

INVENT YOURSELF

PROD. STUDY

OBSERVE



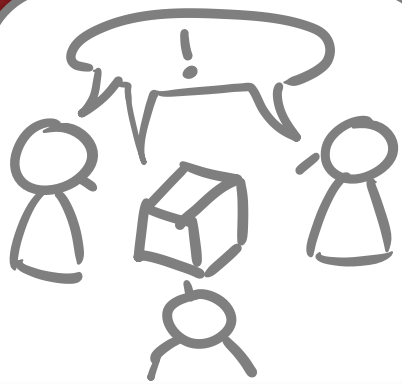
BRAINSTORM

ESTABLISHING requirements

ASK USER

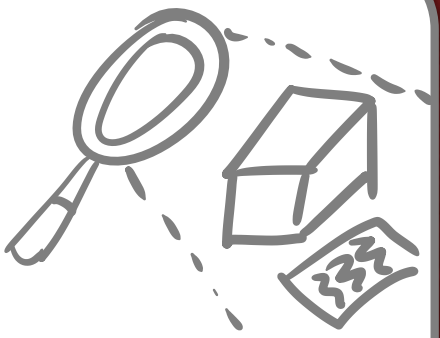
SURVEYS

INTERVIEWS

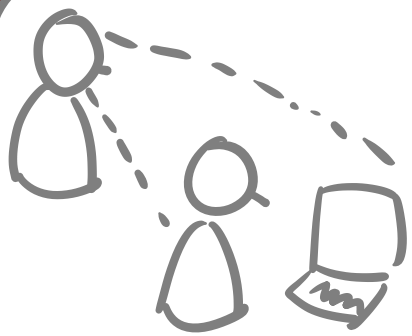


FOCUS GROUPS

INVENT YOURSELF



PROD. STUDY



OBSERVE

BRAINSTORM

ESTABLISHING requirements

ASK USER

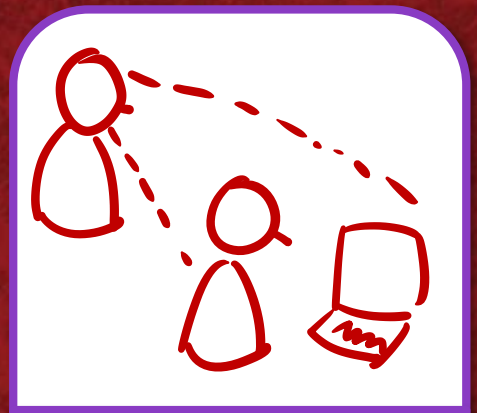
SURVEYS

INTERVIEWS

FOCUS GROUPS

INVENT YOURSELF

PROD. STUDY

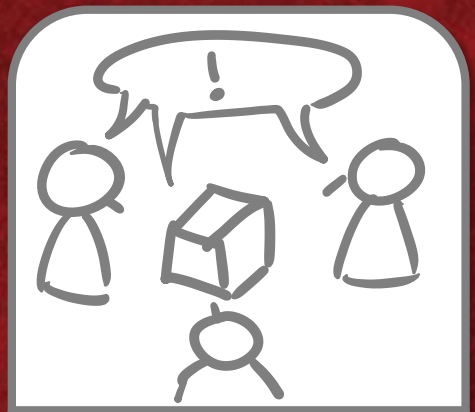


OBSERVE

BRAINSTORM

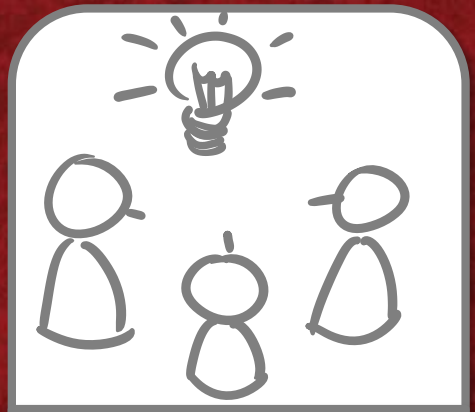
ESTABLISHING requirements

INTERVIEWS



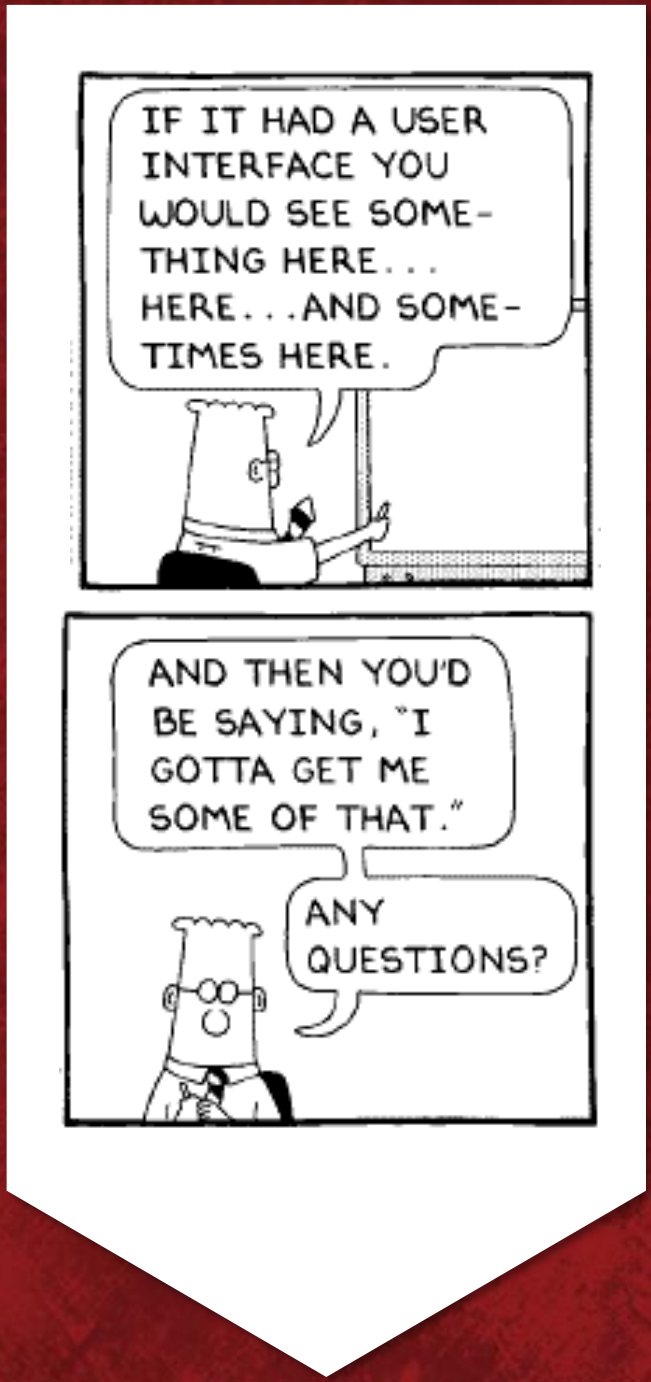
FOCUS GROUPS

OBSERVE



BRAINSTORM

ESTABLISHING requirements



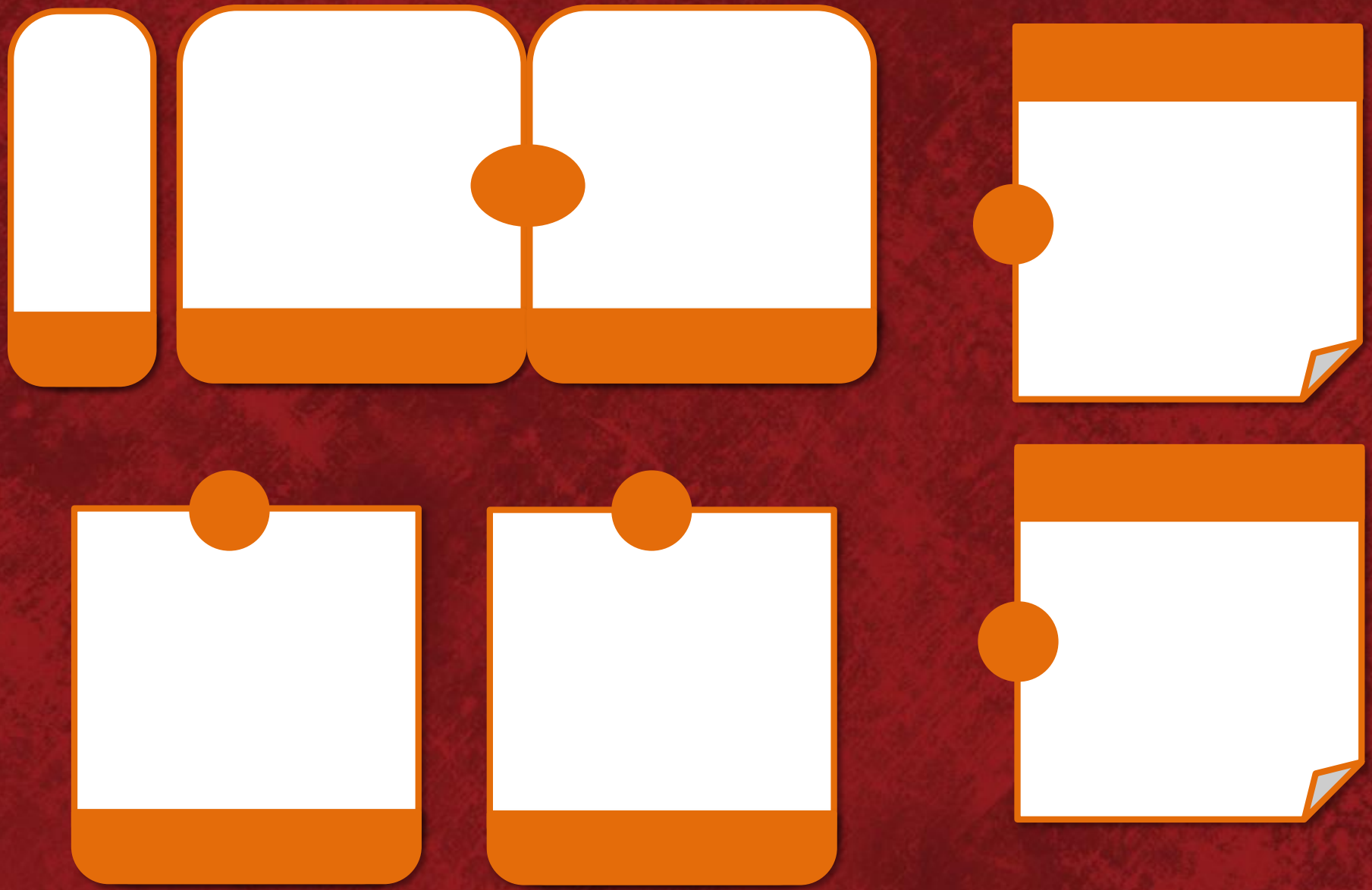
ESTABLISHING requirements

SPECIFYING requirements

ESTABLISHING requirements

SPECIFYING requirements

SPECIFYING requirements



TEAMMATES
Online Peer Evaluation & Peer Feedback System
[System Specification (Spec)]

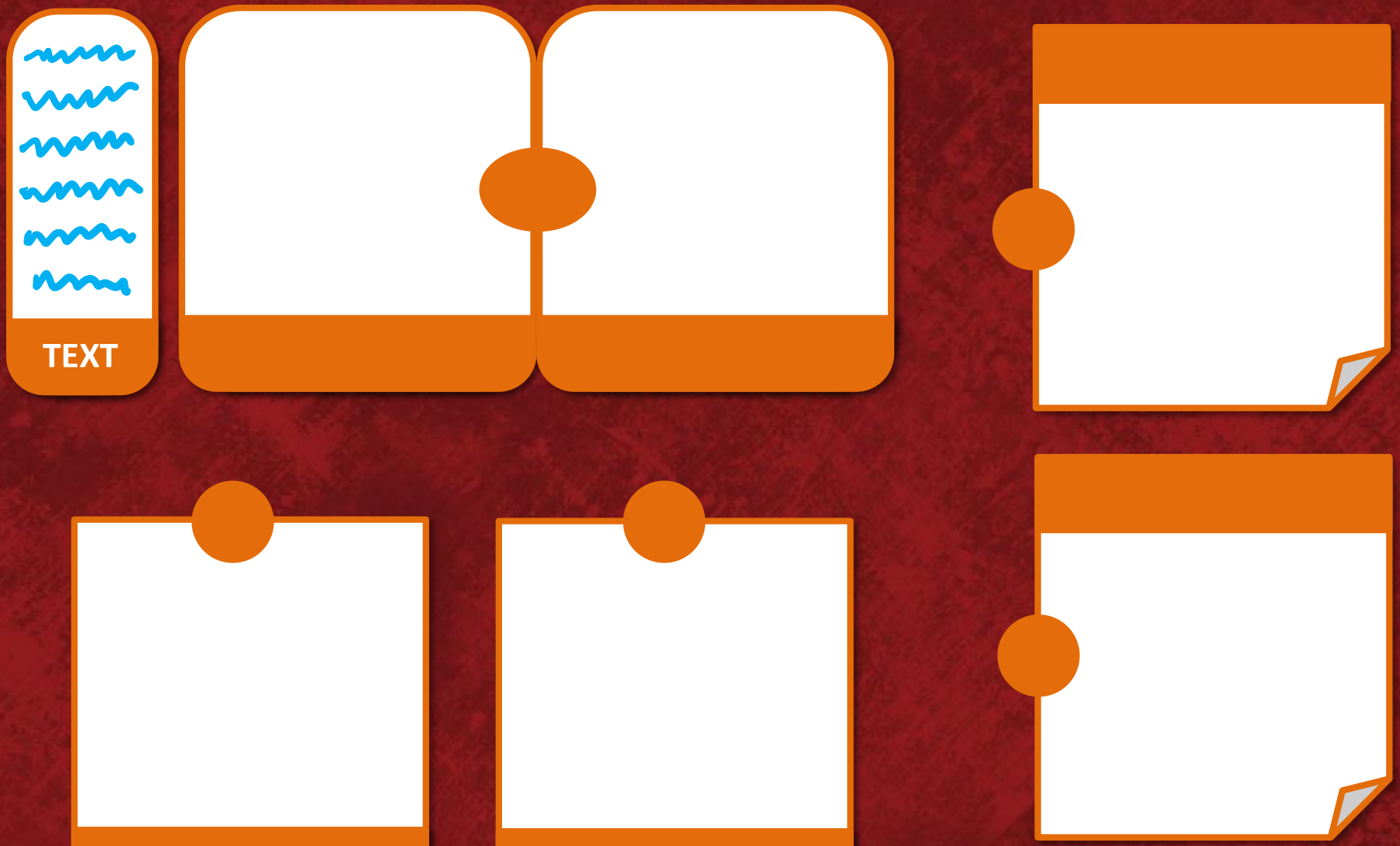
Text

TheProduct

Vision

TEAMMATES is a tool to **help instructors manage various forms of feedback in a class**, including peer evaluations.
Our target users are instructors. Their students are secondary users. i.e., Instructors decide to use the system, not students.

SPECIFYING requirements



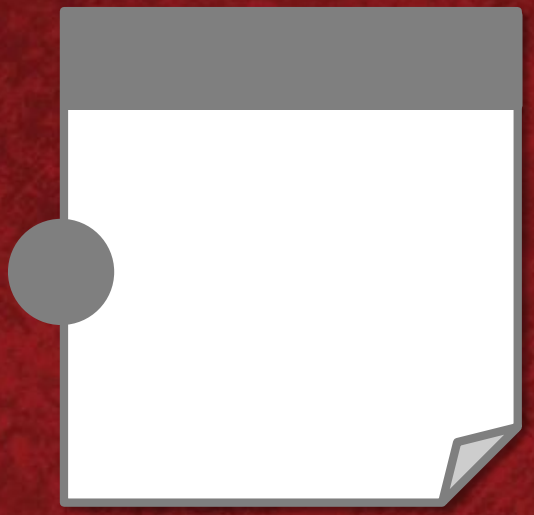
	Feature	Feature Description
1	User Interface	
1.1	Configuration generates order line item comments	Allow the configuration to populate order line item comments with detailed configuration infomation along with configuration comments. Stop change of line item notes other than from within the appropriate configurator.
1.2	Configrator UI Rework: Verbose wizard style	Rework user-interface to match the prototye discussed in collaborative review session. Fast, simple data entry. No unnecessary frills.
1.3	Configuration generates custon specific pricing	der customer for use in ing.
1.4	Allow adding a configuration and continue	e added to an order. To add a subsequent similar blind you'd need to go back into configurator and reenter all the infomation. This feature would allow the user to accept a finished configuration and immediately allow adding an additional configuration using the data from the previous configuration.
2	Special Order PO Generation	
2.1	PO generated correctly for configuration	Currently POs are generated for regular order line items. Change to allow configuration components to generate 1 or more purchase orders at a cost furnished by the component

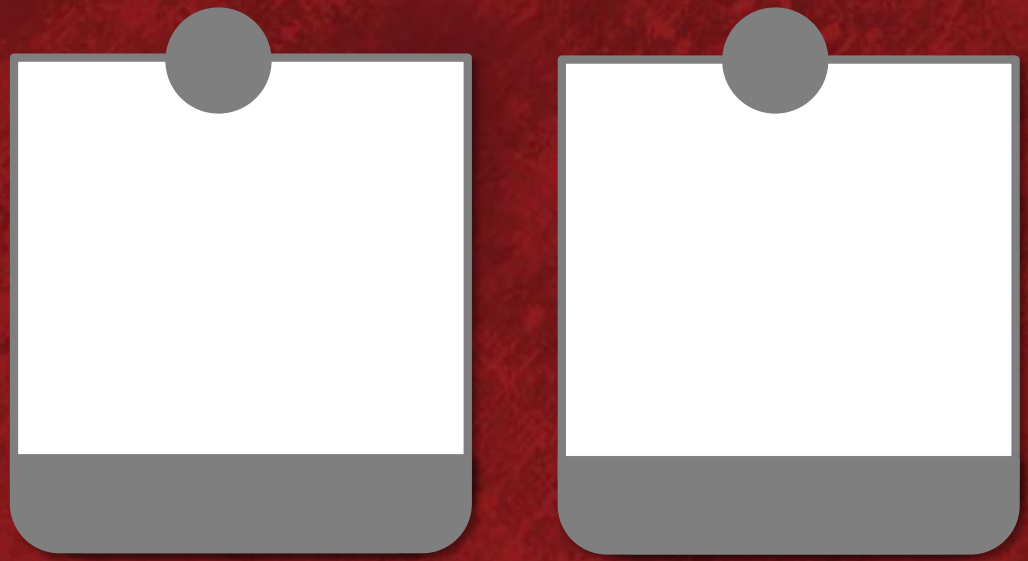
Feature lists

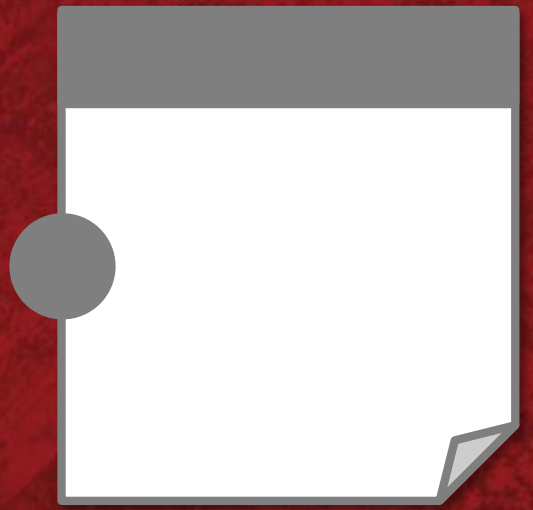
Workbench Features	Module
1 Hierarchical editing of content	Menu Node Edit / TAC / RIAT
1.1 Pluggable hierarchies	None
• Default is workbench owned taxonomy	
• 1.2 Editorial access CRUD by hierarchy	Menu Node Edit
• 1.3 Editorial access CRUD by node type by hierarchy	Menu Node Edit
• 1.4 Theme / menu / context adjustments by hierarchy (?)	Context / Menu Node Section
• 1.5 Exclude content types from hierarchies (media, esp.)	Menu Node Edit
• 1.6 Assign subedtiors	Menu Node Edit
2 Extensible workflow states	Workflow / Flag
• 2.1 Tied to actions / rules	Workflow / Flag Actions
• 2.2 Tied to Views (admin views)	VBO / Flag
• 2.3 Tied to notifications / email	Workflow / Flag / Actions
3 Version control	
• 3.1 Revisions	Core
• 3.2 Untrusted users put	Actions / Palantir custom action
• 3.3 Live / draft moderati	Revision Moderation / Actions
• 3.4 Logging	Core
• 3.5 Diffs	Diff
• 3.6 Scheduled publishing	Scheduler
• 3.7 Page owner notified of changes, may not be author	
4 Editorial dashboards	
• 4.1 Workflow state lists	Views / VBO / Flag / Workflow
• 4.2 Editor flags (flag for review)	Flag
• 4.3 Dashboard list of content per user	
• 4.4 Pages I own	
• 4.5 Pages I can edit	
• 4.6 Pages that have changes	
• 4.7 Pages that have been flagged	

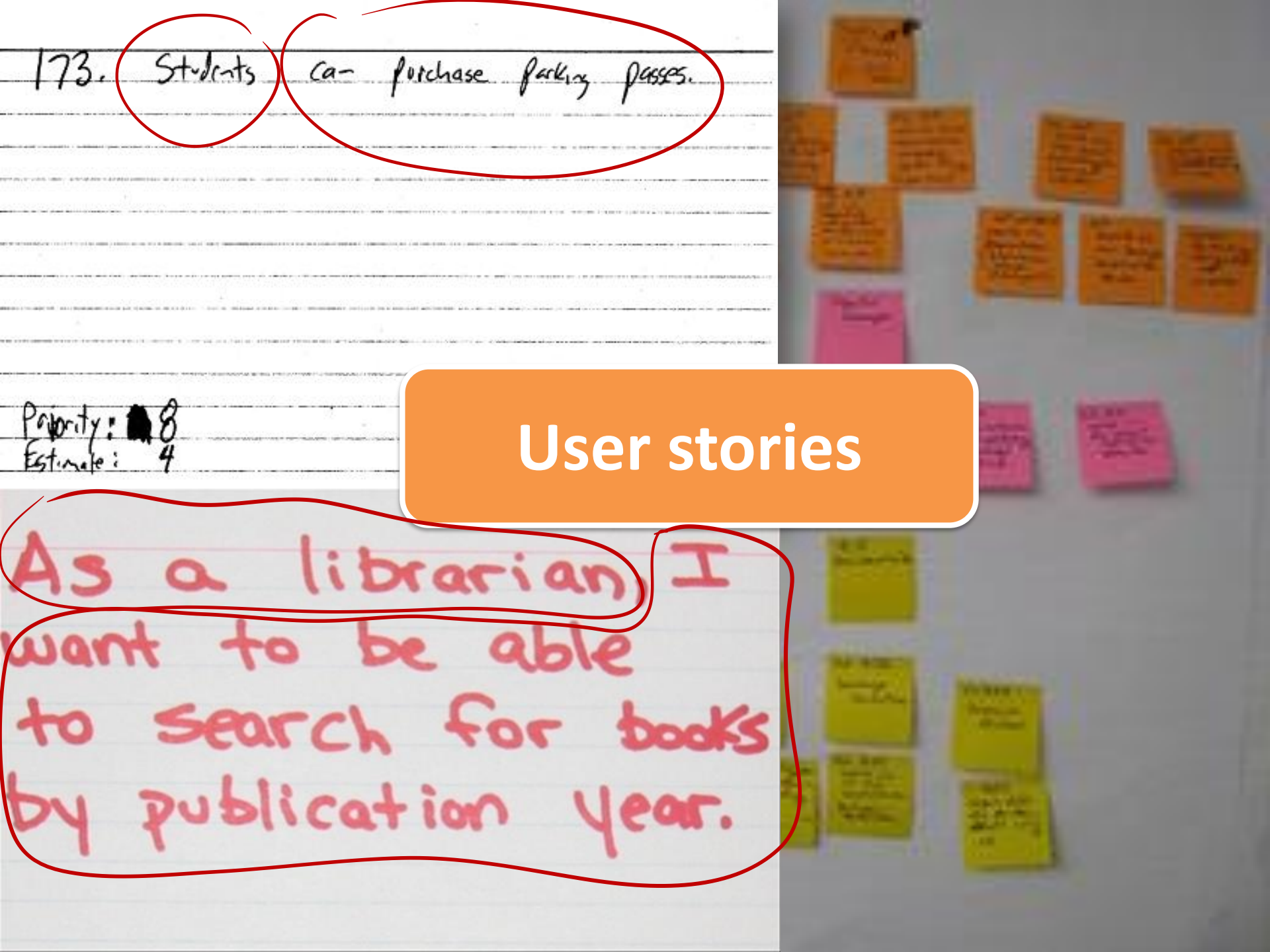
Feature lists

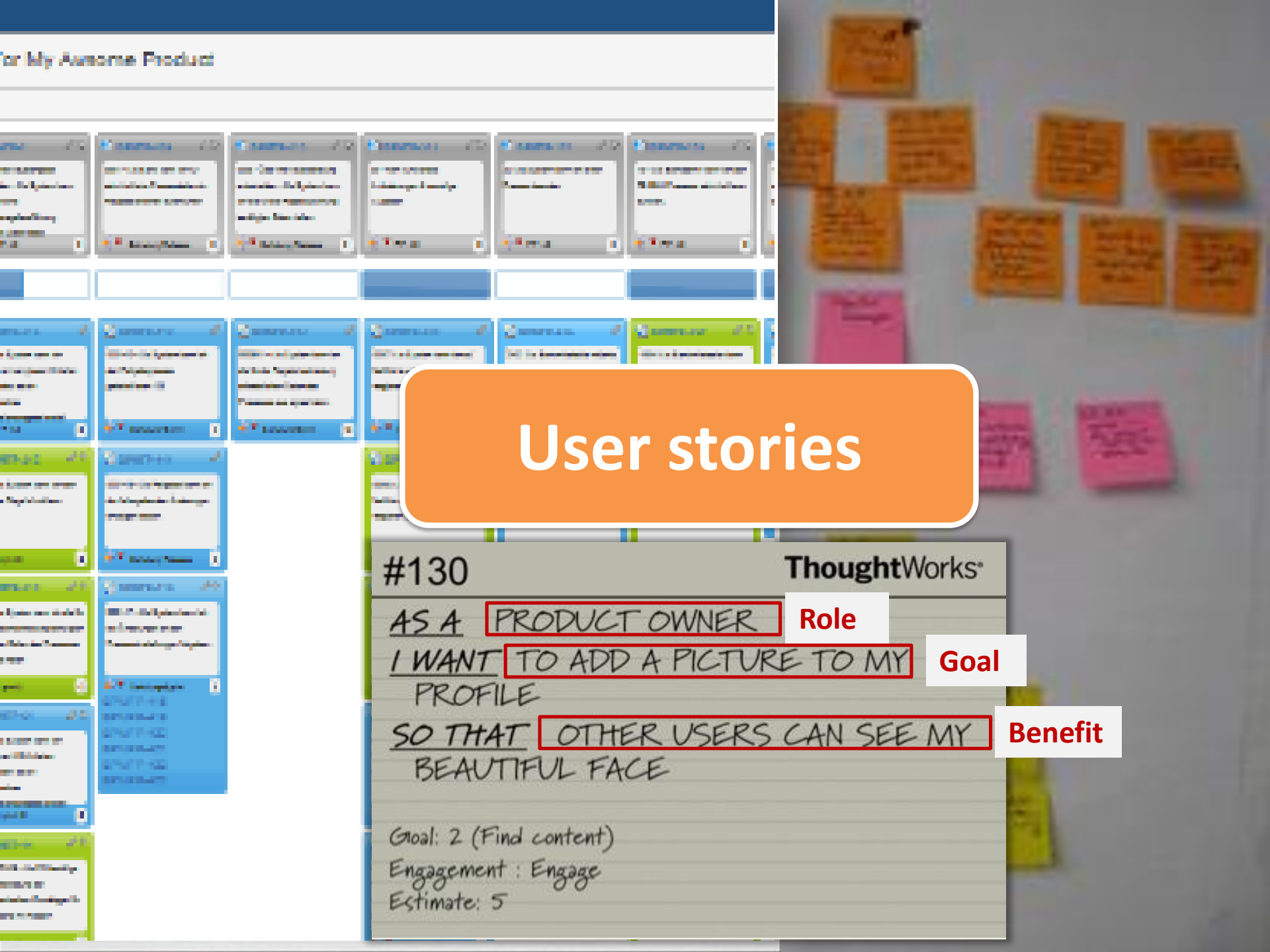
SPECIFYING requirements

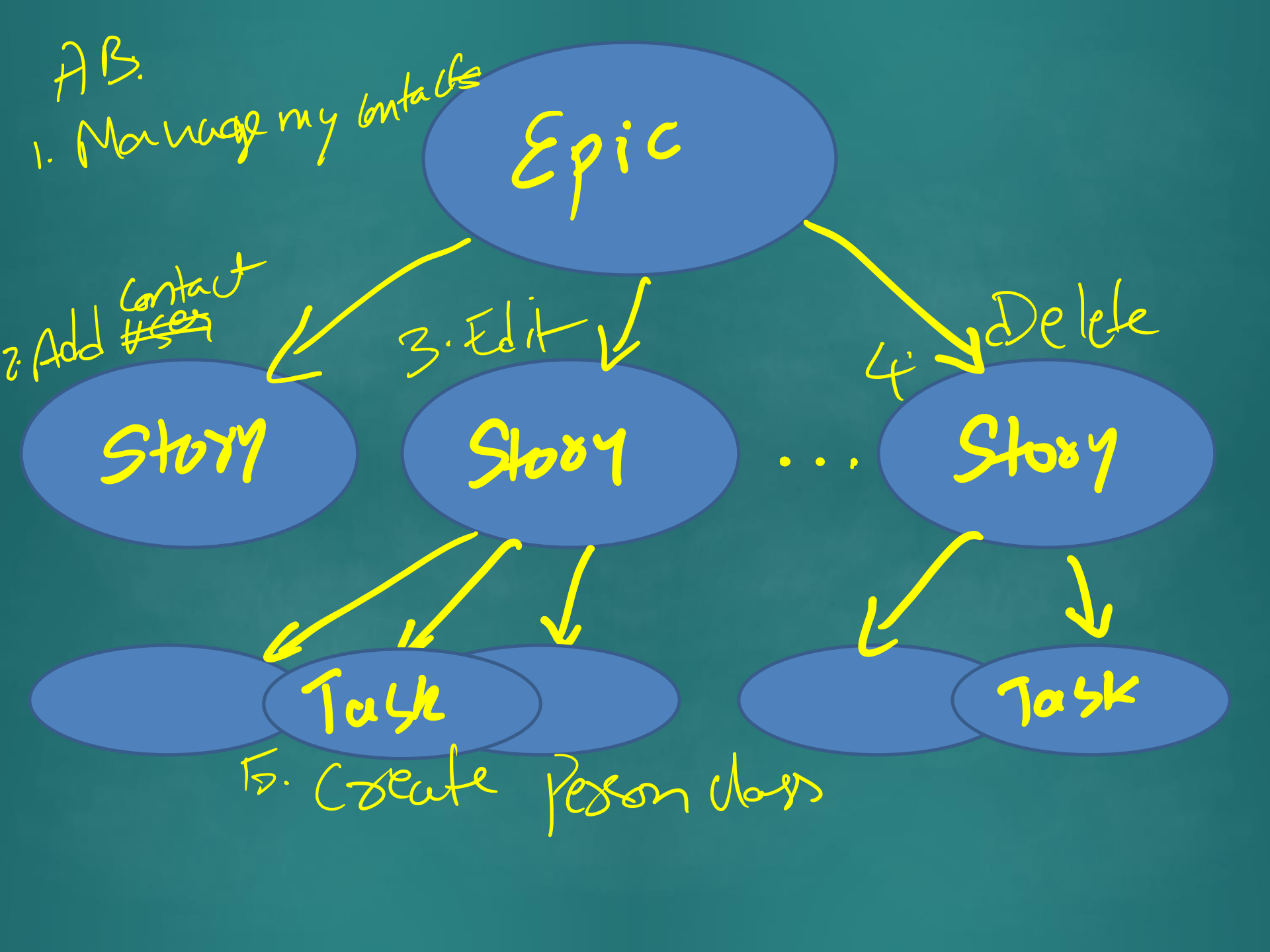




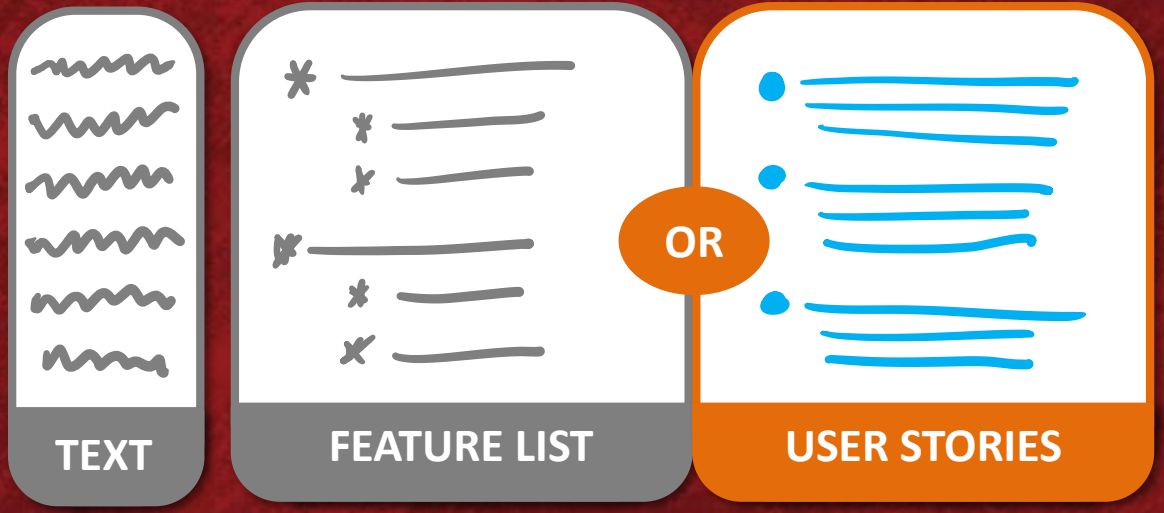


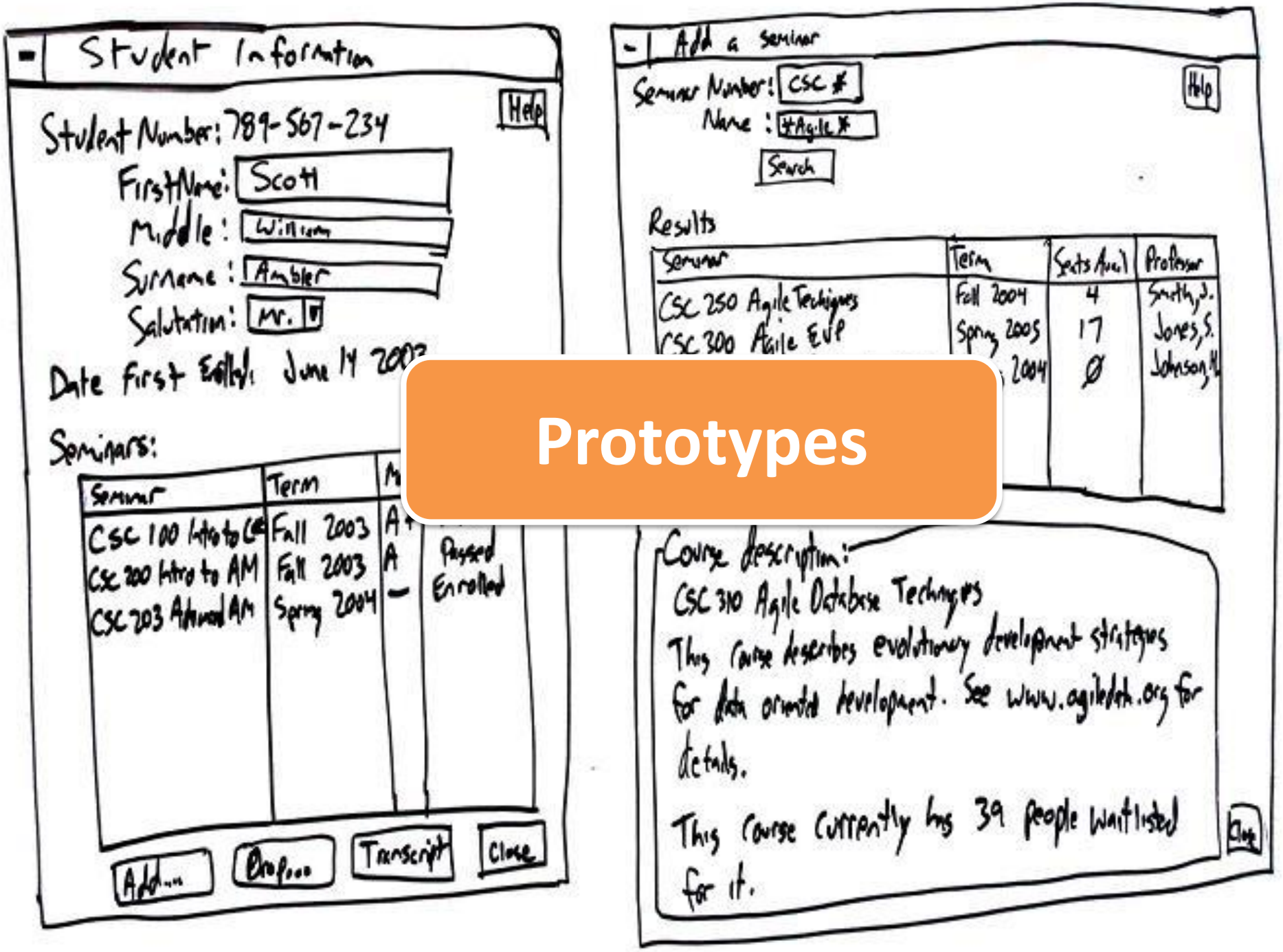

173. Students can purchase parking passes.
Priority: 8
Estimate: 4
User stories
As a librarian I want to be able to search for books by publication year.


User stories
#130
ThoughtWorks
AS A PRODUCT OWNER
Role
I WANT TO ADD A PICTURE TO MY PROFILE
Goal
SO THAT OTHER USERS CAN SEE MY BEAUTIFUL FACE
Benefit
Goal: 2 (Find content)
Engagement : Engage
Estimate: 5

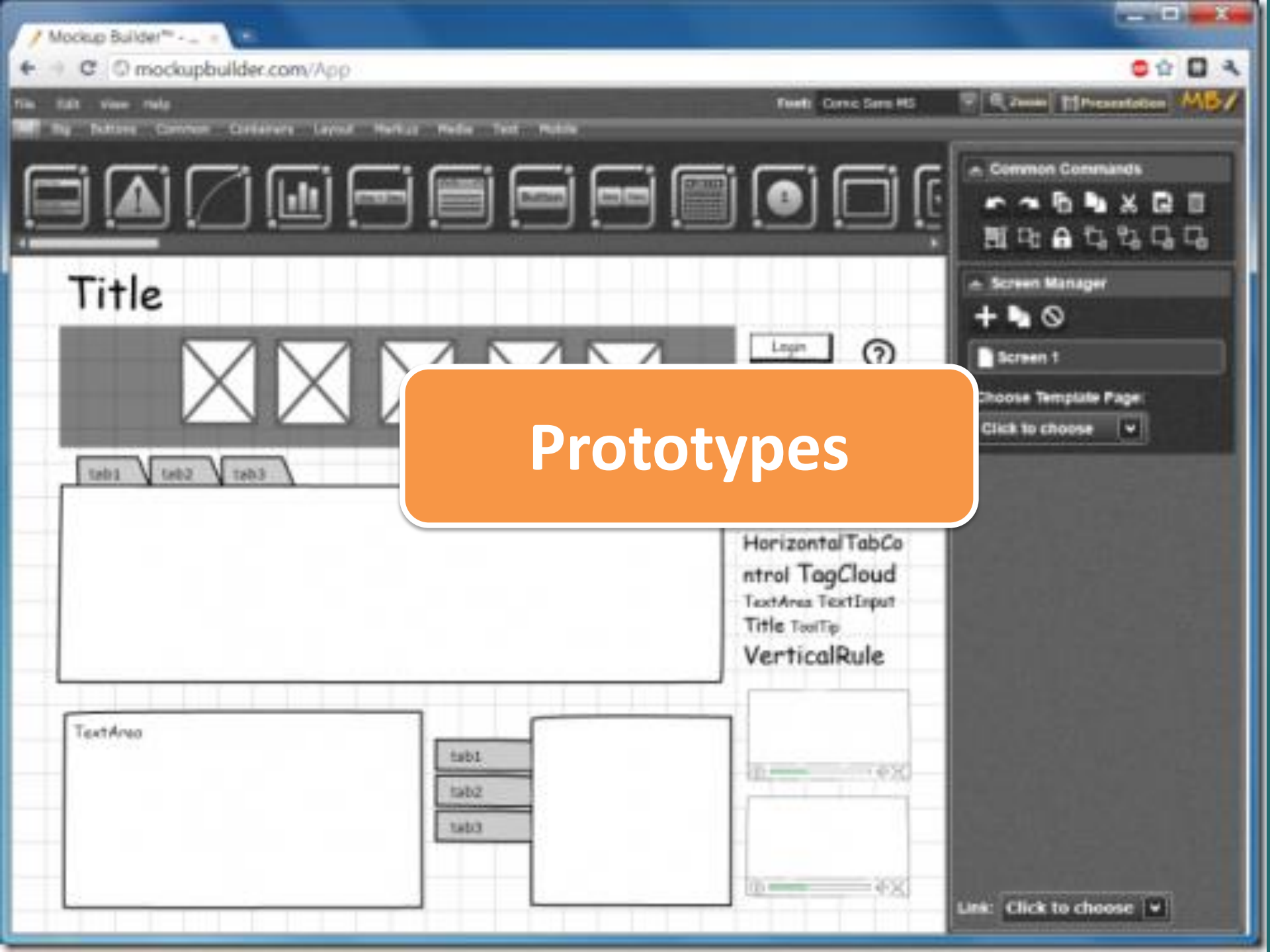

AB.
1. Manage my contacts
Epic
2. Add contact
3. Edit
4. Delete
Story
Story
...
Story
Task
Task
5. Create Person class

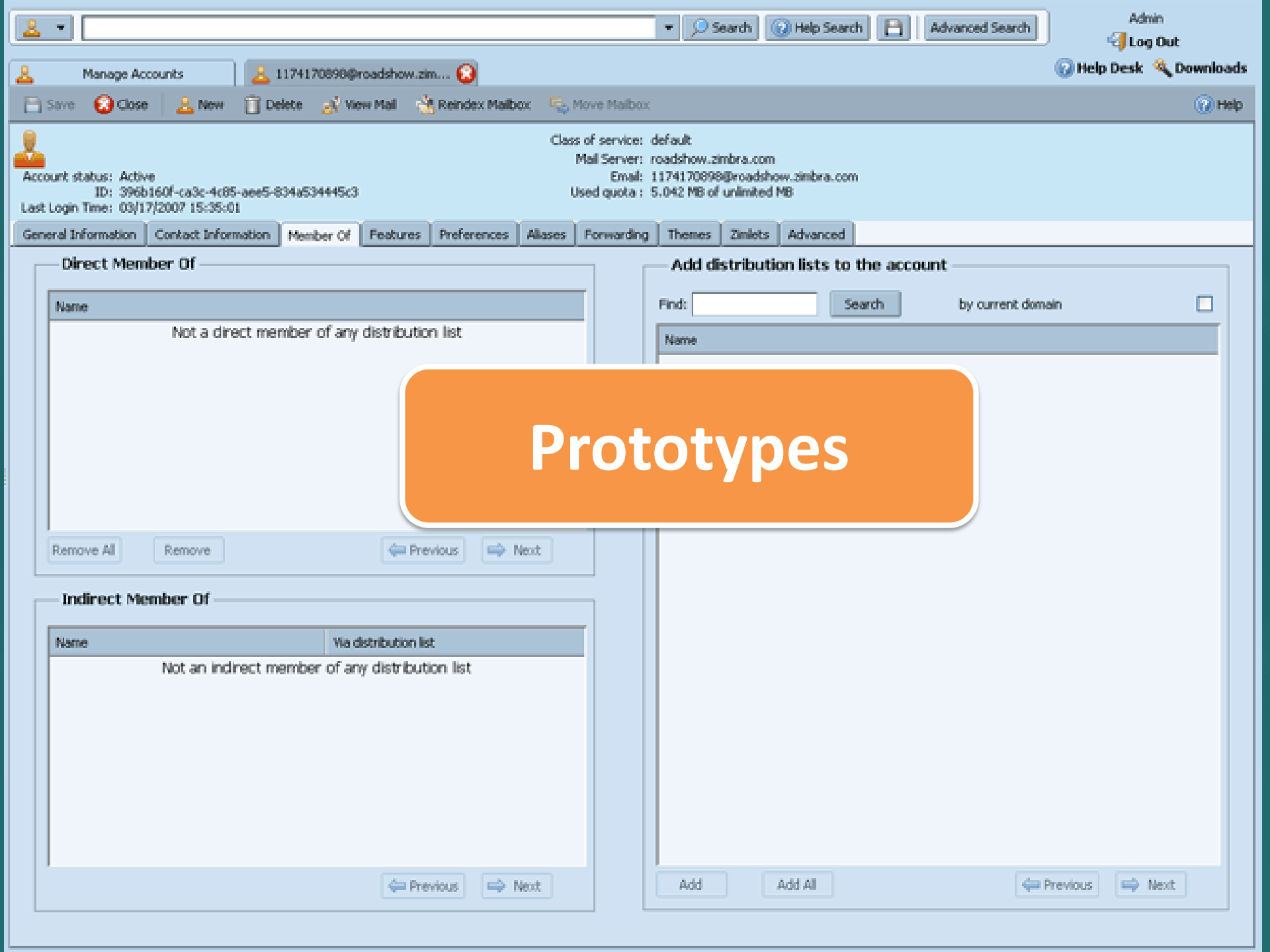
SPECIFYING requirements




Student Information
Help
Student Number: 789-567-234
FirstName: Scott
Middle: William
Surname: Amber
Salutation: Mr.
Date First Enrolled: June 14 2003
Seminars:
Seminar
Term
CSC 100 Intro to CS
Fall 2003
A+
CSC 200 Intro to AM
Fall 2003
A
Passed
CSC 203 Advanced AM
Spring 2004
—
Enrolled
Add...
Drop...
Transcript
Close
Add a Seminar
Seminar Number: CSC #
Name: %Agile%
Search
Help
Results
Seminar
Term
Seats Avail
Professor
CSC 250 Agile Techniques
Fall 2004
4
Smith, J.
CSC 300 Agile EJB
Spring 2005
17
Jones, S.
2004
0
Johnson, M.
Course description:
CSC 310 Agile Database Techniques
This course describes evolutionary development strategies for data oriented development. See www.agiledata.org for details.
This course currently has 39 people waitlisted for it.
Close

Prototypes


Mockup Builder
mockupbuilder.com/App
Common Commands
Screen Manager
Screen 1
Choose Template Page:
Click to choose
Title
Login
tab1
tab2
tab3
HorizontalTabCo
ntrol TagCloud
TextArea TextInput
Title ToolTip
VerticalRule
TextArea
tab1
tab2
tab3
Link:
Click to choose
Prototypes


Search
Help Search
Advanced Search
Admin
Log Out
Help Desk
Downloads
Manage Accounts
1174170898@roadshow.zim...
Save
Close
New
Delete
View Mail
Reindex Mailbox
Move Mailbox
Help
Account status: Active
ID: 396b160f-ca3c-4c85-aee5-834a534445c3
Last Login Time: 03/17/2007 15:35:01
Class of service: default
Mail Server: roadshow.zimbra.com
Email: 1174170898@roadshow.zimbra.com
Used quota : 5.042 MB of unlimited MB
General Information
Contact Information
Member Of
Features
Preferences
Aliases
Forwarding
Themes
Zimlets
Advanced
Direct Member Of
Name
Not a direct member of any distribution list
Remove All
Remove
Previous
Next
Indirect Member Of
Name
Via distribution list
Not an indirect member of any distribution list
Previous
Next
Add distribution lists to the account
Find:
Search
by current domain
Name
Add
Add All
Previous
Next
Prototypes

SPECIFYING requirements

TEXT

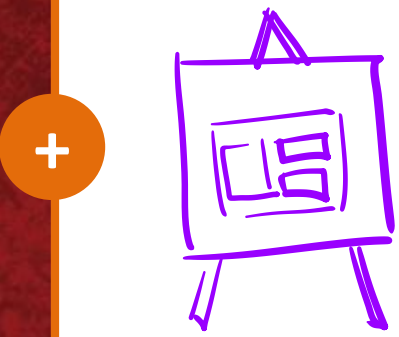
FEATURE LIST

OR

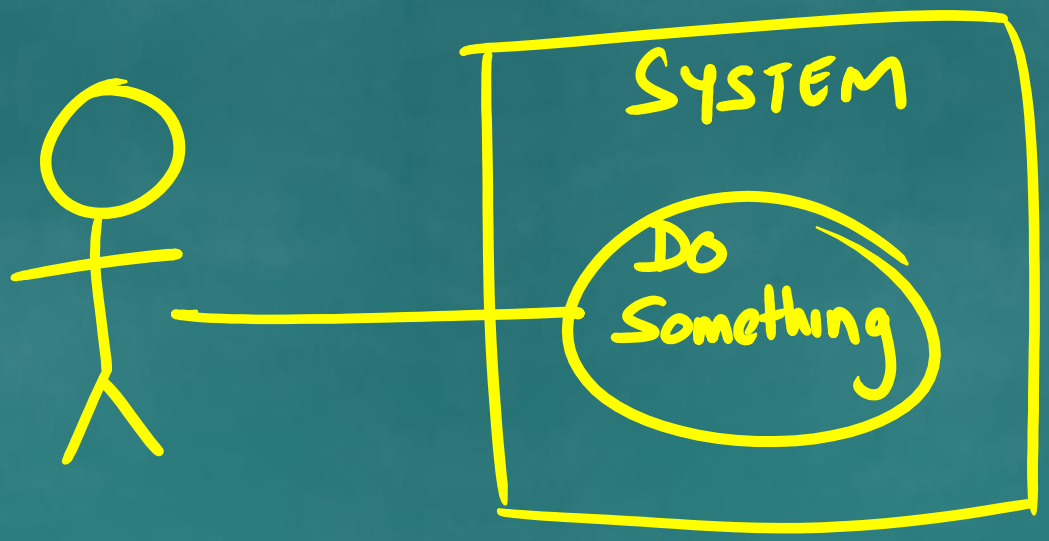
USER STORIES

PROTOTYPES

+



Use cases



Use cases

Use cases

As a librarian, I want to be able to search for books by publication year.

System: Online Banking System (OBS)

Use case: UC23 - Transfer Money

Actor: Customer

Main success scenario:

1. User chooses to transfer money.
2. OBS requests for details of the transfer.
3. User performs enter transaction details (UC14a).
4. OBS requests for confirmation.
5. User confirms transfer.
6. OBS transfers the money and displays status.

   Use case ends.

System: Online Banking System (OBS)

Use case: UC23 - Transfer Money

Actor: Customer

Main success scenario:

1. User chooses to transfer money.
2. OBS requests for details of the transfer.
3. User performs enter transaction details (UC14a).
4. OBS requests for confirmation.
5. User confirms transfer.
6. OBS transfers the money and displays status.

Use case ends.

Think "ROLE"

System: Online Banking System (OBS)

Use case: UC23 - Transfer Money

Actor: Customer

Main success scenario:

1. User chooses to transfer money.
2. OBS requests for details of the transfer.
3. User performs enter transaction details (UC14a).
4. OBS requests for confirmation.
5. User confirms transfer.
6. OBS transfers the money and displays status.

   Use case ends.

System: Online Banking System (OBS)

Use case: UC23 - Transfer Money

Actor: Customer

Main success scenario:

1. User chooses to transfer money.
2. OBS requests for details of the transfer.
3. User performs enter transaction details (UC14a).
4. OBS requests for confirmation.
5. User confirms transfer.
6. OBS transfers the money and displays status.

   Use case ends.

System: Online Banking System (OBS)

Use case: UC23 - Transfer Money

Actor: Customer

Main success scenario:

1. User chooses to transfer money.
2. OBS requests for details of the transfer.
3. User performs enter transaction details (UC14a).
4. OBS requests for confirmation.
5. User confirms transfer.
6. OBS transfers the money and displays status.

Use case ends.

System: Online Banking System (OBS)

Use case: UC23 - Transfer Money

Actor: Customer

Main success scenario:

1. User chooses to transfer money.
2. OBS requests for details of the transfer.
3. User performs enter transaction details (UC14a).
4. OBS requests for confirmation.
5. User confirms transfer.
6. OBS transfers the money and displays status.

Use case ends.

INCLUSION

What is the main success scenario for the buy drink can use case?

What is the main success scenario for the buy drink can use case?

System: Vending machine (VM)
Actor: Customer (C)

1. C: Insert coin into the machine
2. VM: Display options
3. C: Select drink
4. VM: Dispense drink

Answer on Archipelago

System: Online Banking System (OBS)

Use case: UC23 - Transfer Money

Actor: User

Main success scenario:

1. User chooses to transfer money.
2. OBS requests for details of the transfer.
3. User performs enter transaction details (UC14a).
4. OBS requests for confirmation.
5. User confirms transfer.
6. OBS transfers the money and displays status.

Use case ends.

System: Online Banking System (OBS)

Use case: UC23 - Transfer Money

Actor: User

Main success scenario:

1. User chooses to transfer money.
2. OBS requests for details of the transfer.
3. User performs enter transaction details (UC14a).
4. OBS requests for confirmation.
5. User confirms transfer.
6. OBS transfers the money and displays status.

Use case ends.

Main success scenario:

1. User chooses to transfer money.
2. OBS requests for details of the transfer.
3. User performs enter transaction details (UC14a).
4. OBS requests for confirmation.
5. User confirms transfer.
6. OBS transfers the money and displays status.

Use case ends.

Main success scenario:

1. User chooses to transfer money.
2. OBS requests for details of the transfer.
3. User performs enter transaction details (UC14a).
4. OBS requests for confirmation.
5. User confirms transfer.
6. OBS transfers the money and displays status.

Use case ends.

Extensions

3a. OBS detects an error in the entered data.

3a1. OBS requests for the correct data.

3a2. User enters new data.

Steps 3a1-3a2 are repeated until the data entered are correct.

Use case resumes from step 4.

Main success scenario:

1. User chooses to transfer money.
2. OBS requests for details of the transfer.
3. User performs enter transaction details (UC14a).
4. OBS requests for confirmation.
5. User confirms transfer.
6. OBS transfers the money and displays status.

Use case ends.

Extensions

3a. OBS detects an error in the entered data.

3a1. OBS requests for the correct data.

3a2. User enters new data.

Steps 3a1-3a2 are repeated until the data entered are correct.

Use case resumes from step 4.

5. User confirms transfer.
6. OBS transfers the money and displays status.

Use case ends.

Extensions

3a. OBS detects an error in the entered data.

3a1. OBS requests for the correct data.

3a2. User enters new data.

Steps 3a1-3a2 are repeated until the data entered are correct.

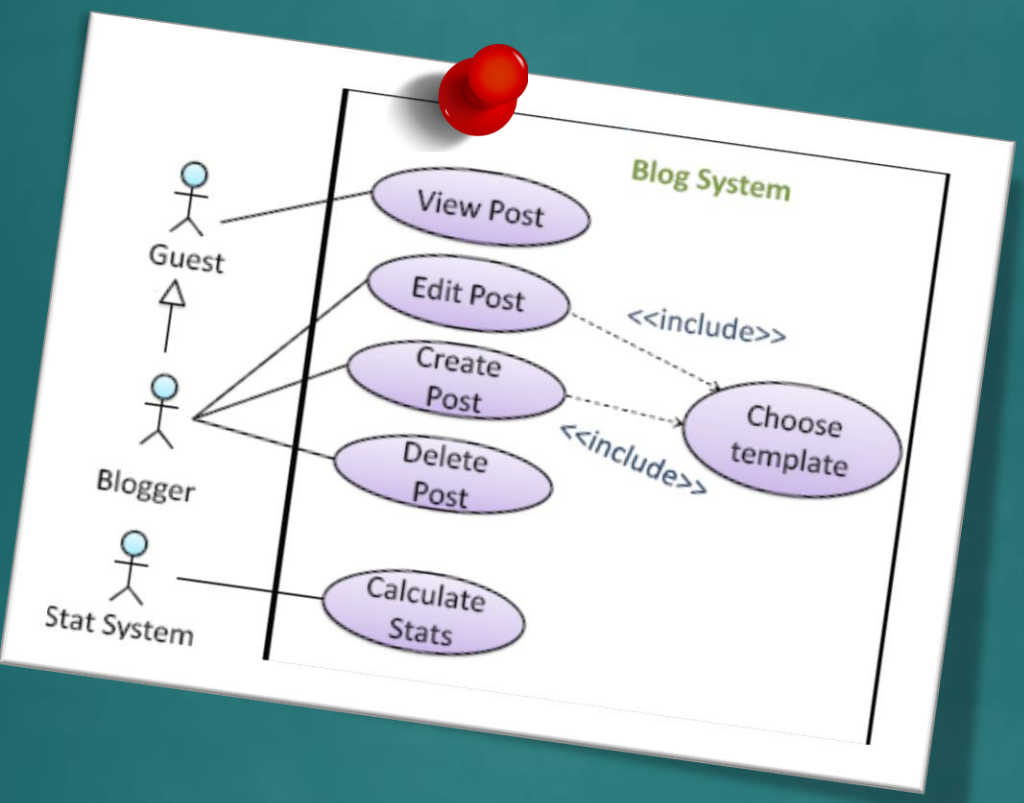
Use case resumes from step 4.

*a. At any time, User chooses to cancel the transfer

*a1. OBS requests to confirm the cancellation

*a2. User confirms the cancellation

Use case ends.



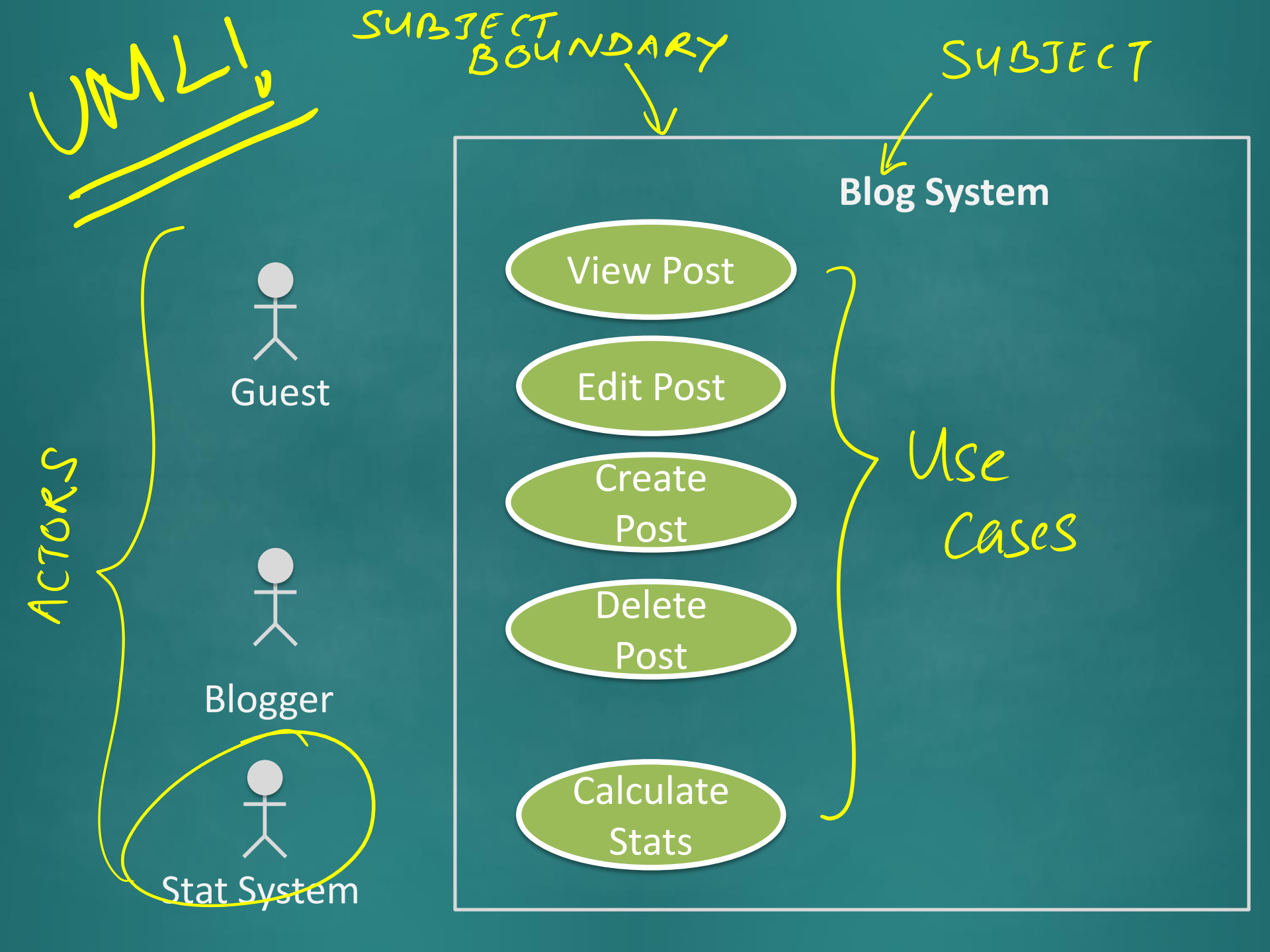
'stem: Online Banking System (OBS)
se case: UC23 - Transfer Money
:tor: User
SS:
1. User chooses to transfer money.
2. OBS requests for details of the transfer.
3. User enters the requested details.
4. OBS requests for confirmation.
5. User confirms transfer.
6. OBS transfers the money and displays the new account balance.
Use case ends.

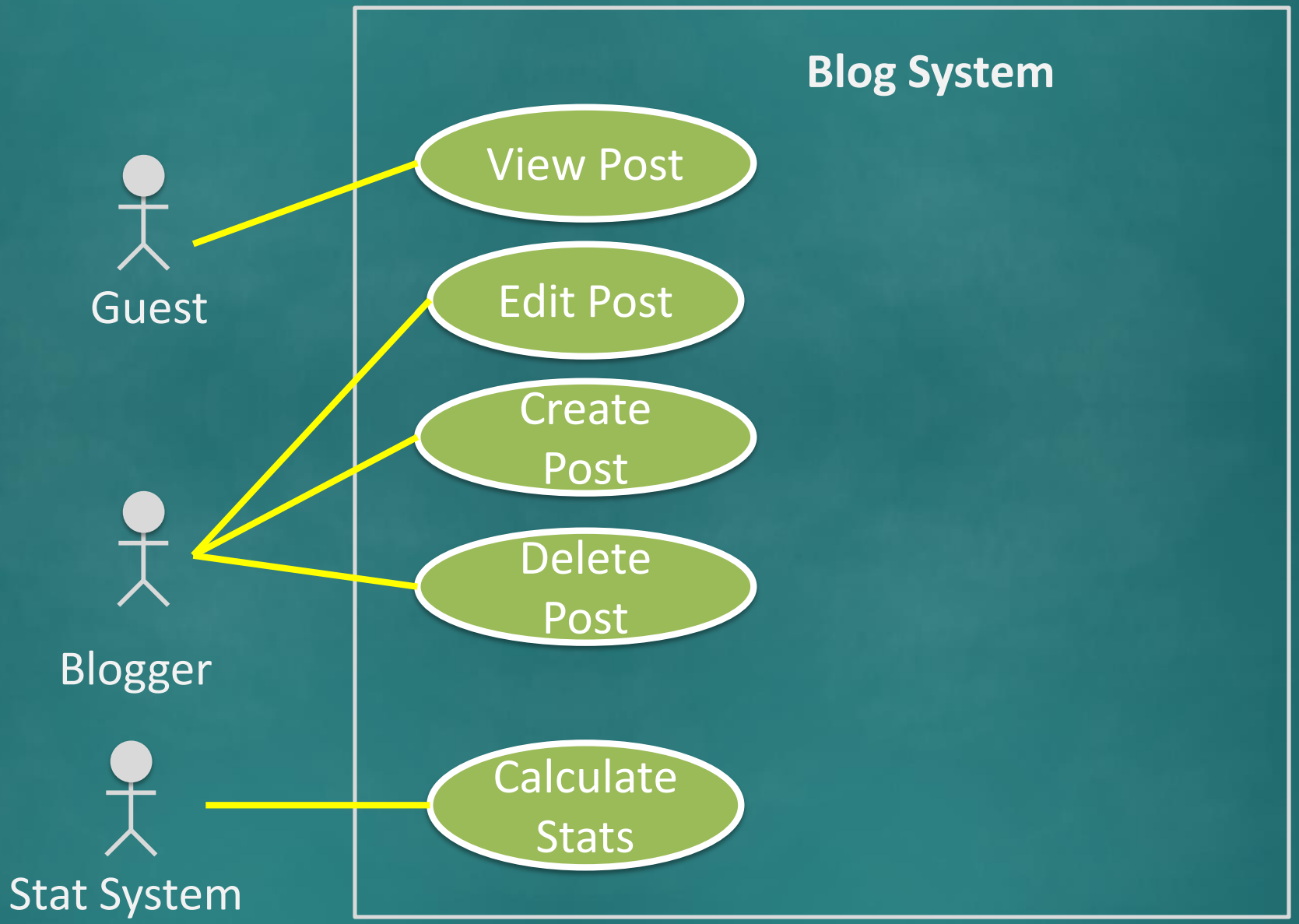
ntensions:
3a. OBS detects an error in the entered data.
3a1. OBS requests for the correct data.
3a2. User enters new data.
Steps 3a1-3a2 are repeated until the data entered are correct.
Use case resumes from step 4.

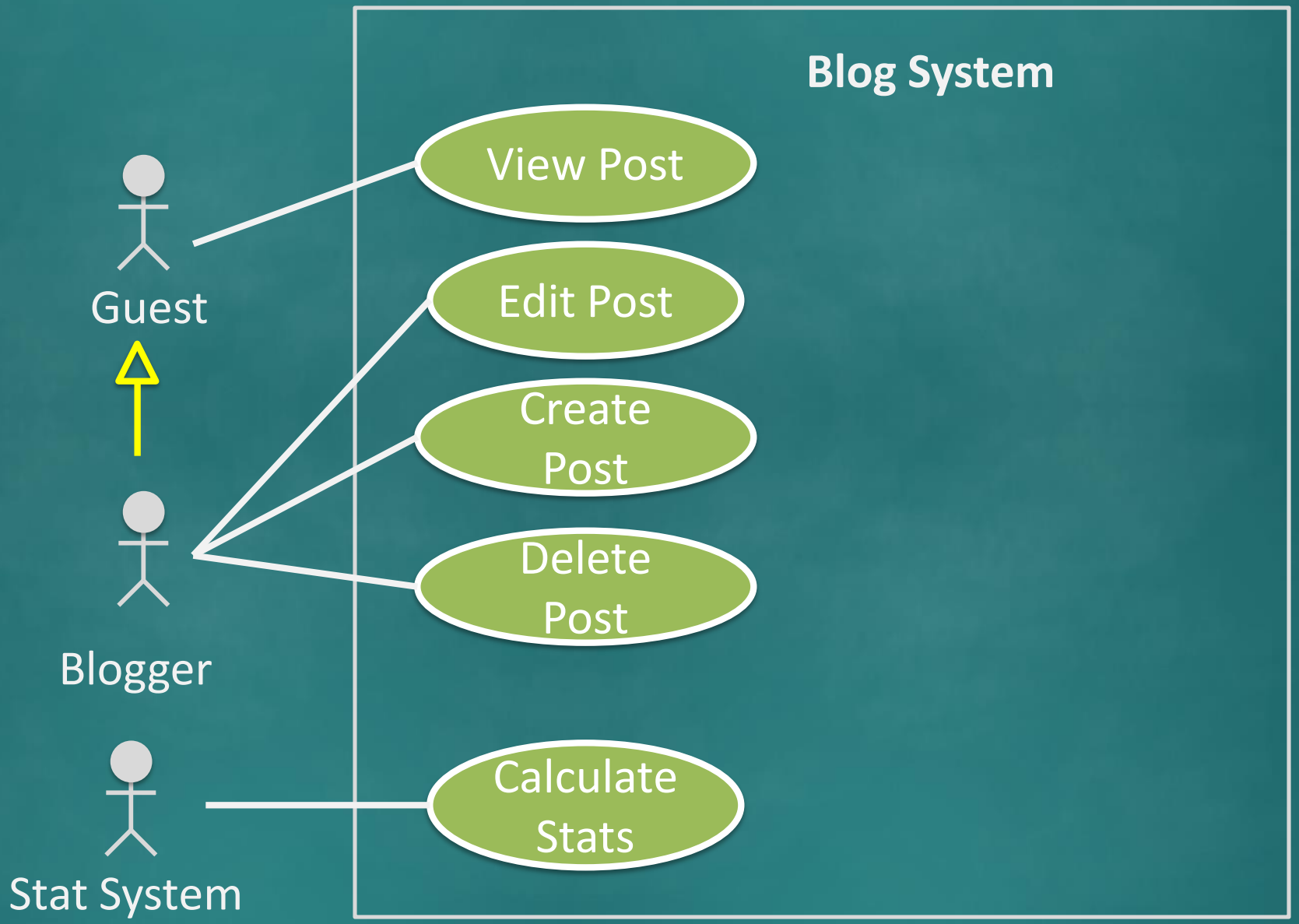
3b. User requests to effect the transfer in a future date.
3b1. OBS requests for confirmation.
3b2. User confirms future transfer.
Use case ends.

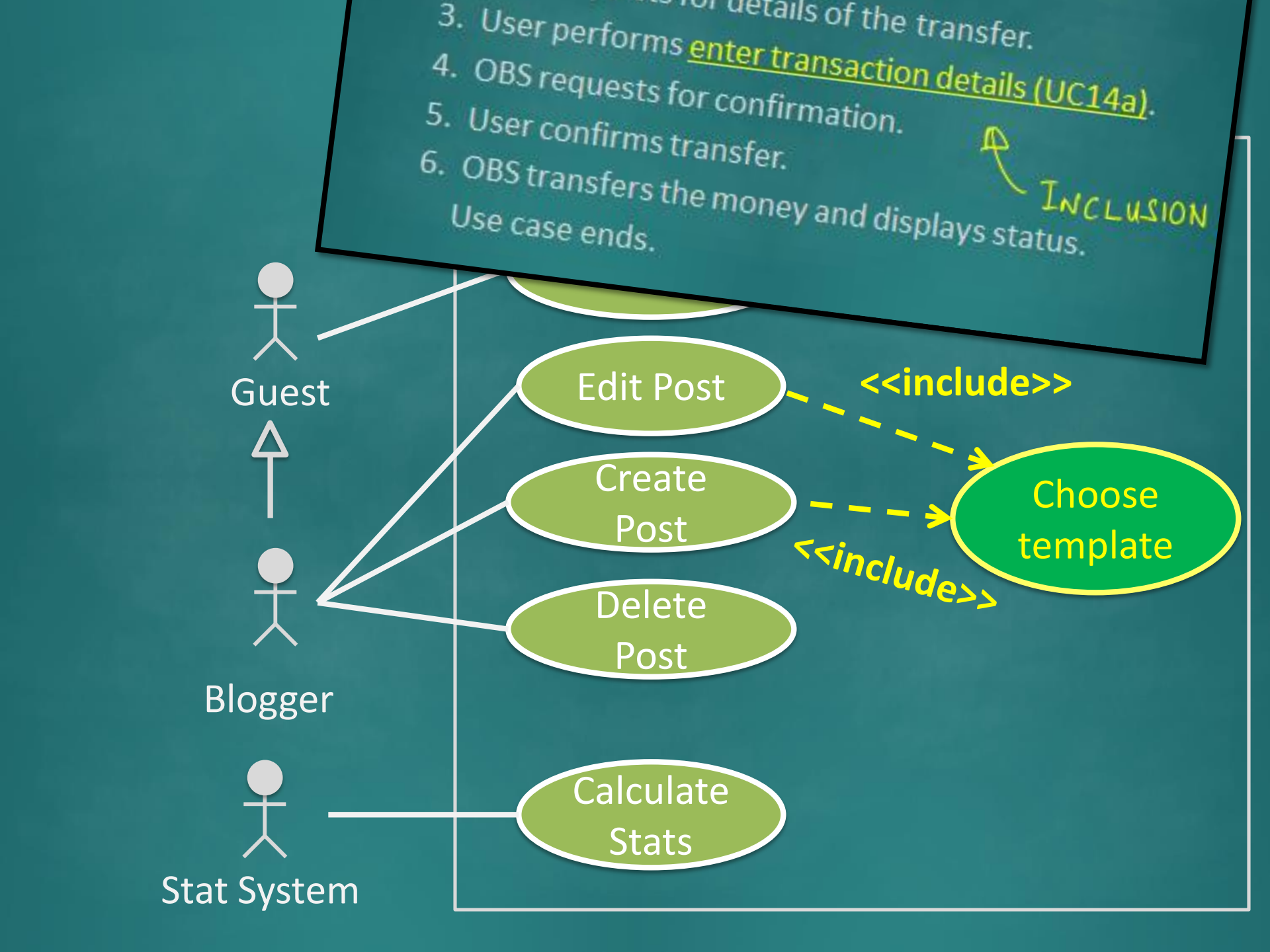
*a. At any time, User chooses to cancel the transfer
*a1. OBS requests to confirm the cancellation
*a2. User confirms the cancellation
Use case ends.

*b. At any time, 120 seconds lapse without any input from the User
*b1. OBS cancels the transfer
*b2. OBS informs the User of the cancellation


UML!
SUBJECT BOUNDARY
SUBJECT
Blog System
View Post
Edit Post
Create Post
Delete Post
Calculate Stats
Use Cases
ACTORS
Guest
Blogger
Stat System


Blog System
View Post
Edit Post
Create Post
Delete Post
Calculate Stats
Guest
Blogger
Stat System


Blog System
View Post
Edit Post
Create Post
Delete Post
Calculate Stats
Guest
Blogger
Stat System


for details of the transfer.
3. User performs enter transaction details (UC14a).
4. OBS requests for confirmation.
5. User confirms transfer.
6. OBS transfers the money and displays status.
Use case ends.
INCLUSION
Guest
Edit Post
<<include>>
Create Post
Choose template
<<include>>
Blogger
Delete Post
Calculate Stats
Stat System

Use case ends.

Extensions

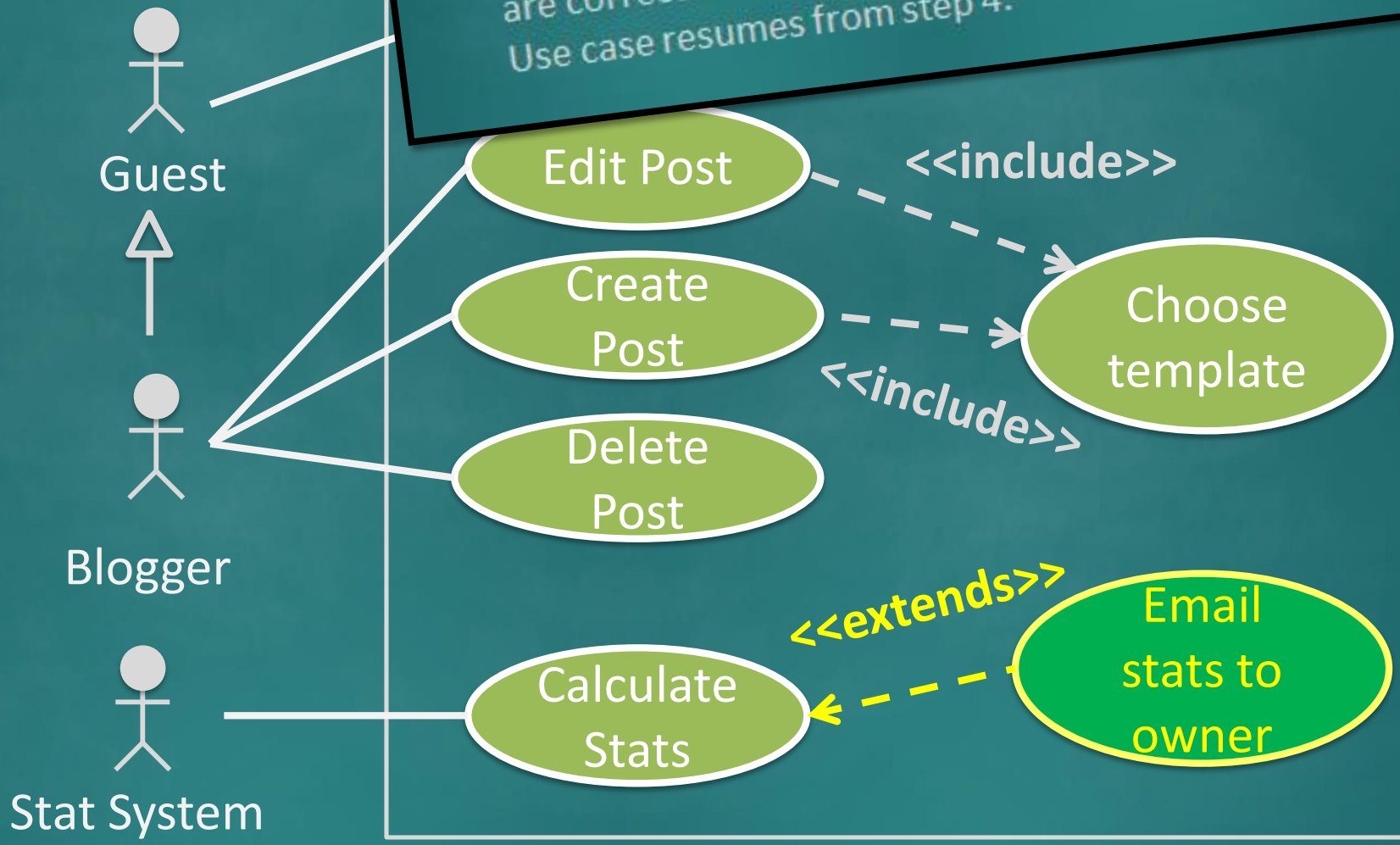
3a. OBS detects an error in the entered data.

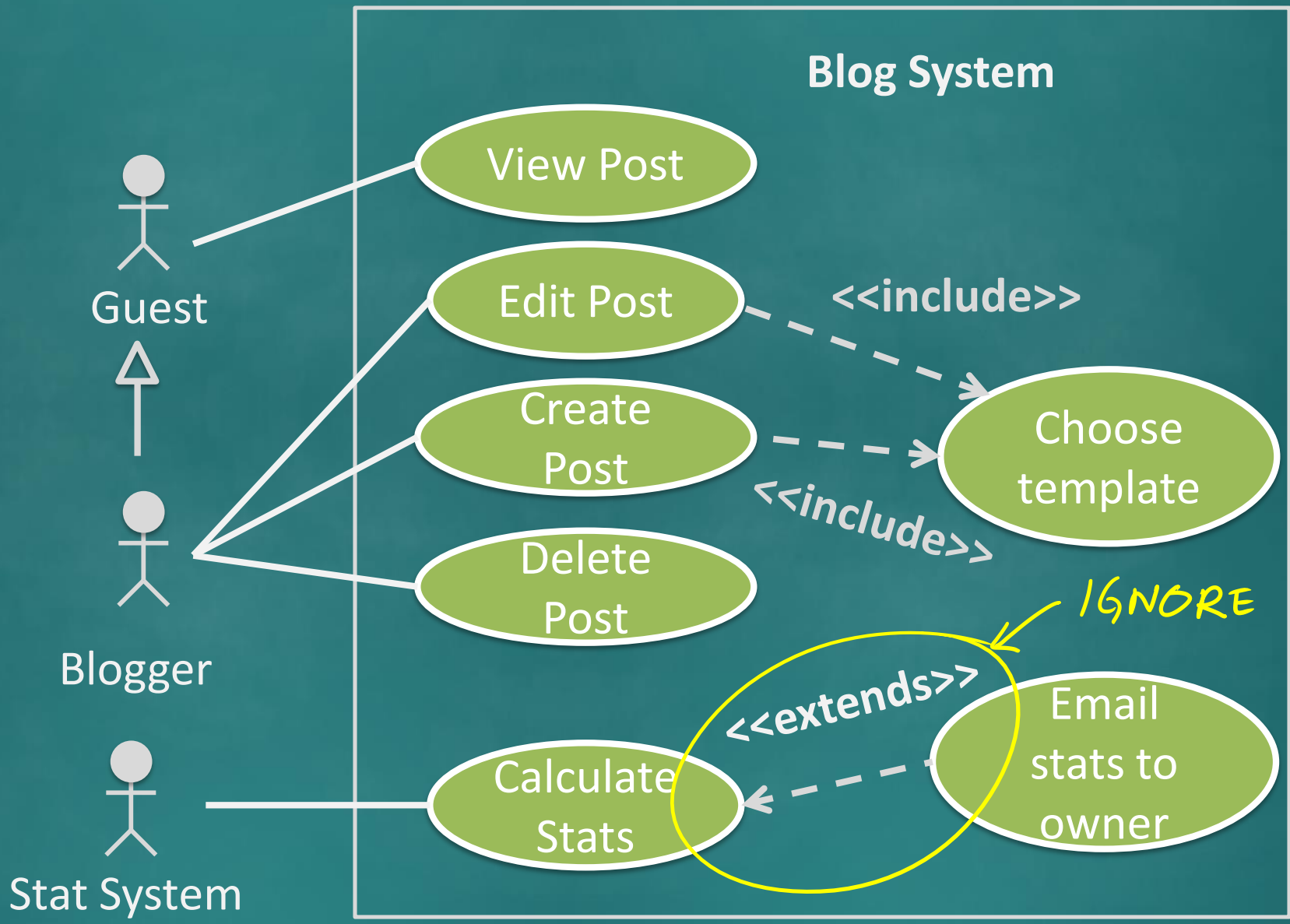
3a1. OBS requests for the correct data.

3a2. User enters new data.

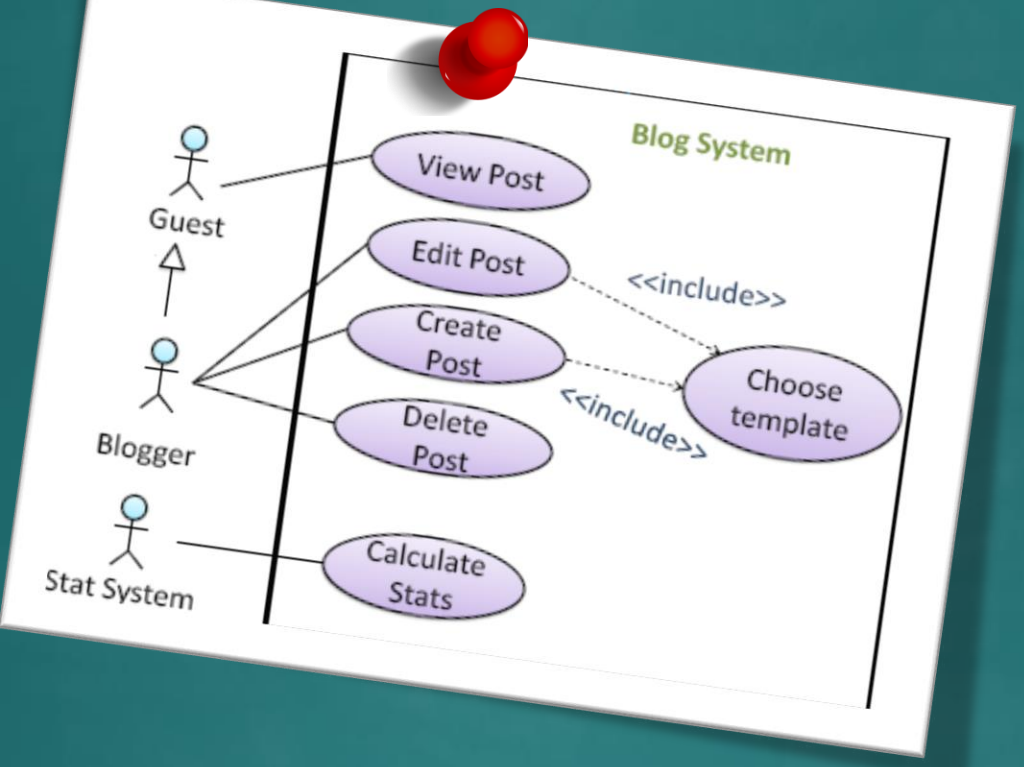
Steps 3a1-3a2 are repeated until the data entered are correct.

Use case resumes from step 4.




Blog System
View Post
Edit Post
Create Post
Delete Post
Choose template
<<include>>
<<include>>
IGNORE
<<extends>>
Email stats to owner
Calculate Stats
Guest
Blogger
Stat System

≃ Table of Contents



'stem: Online Banking System (OBS)
se case: UC23 - Transfer Money
:tor: User
SS:
1. User chooses to transfer money.
2. OBS requests for details of the transfer.
3. User enters the requested details.
4. OBS requests for confirmation.
5. User confirms transfer.
6. OBS transfers the money and displays the new account balance.
Use case ends.

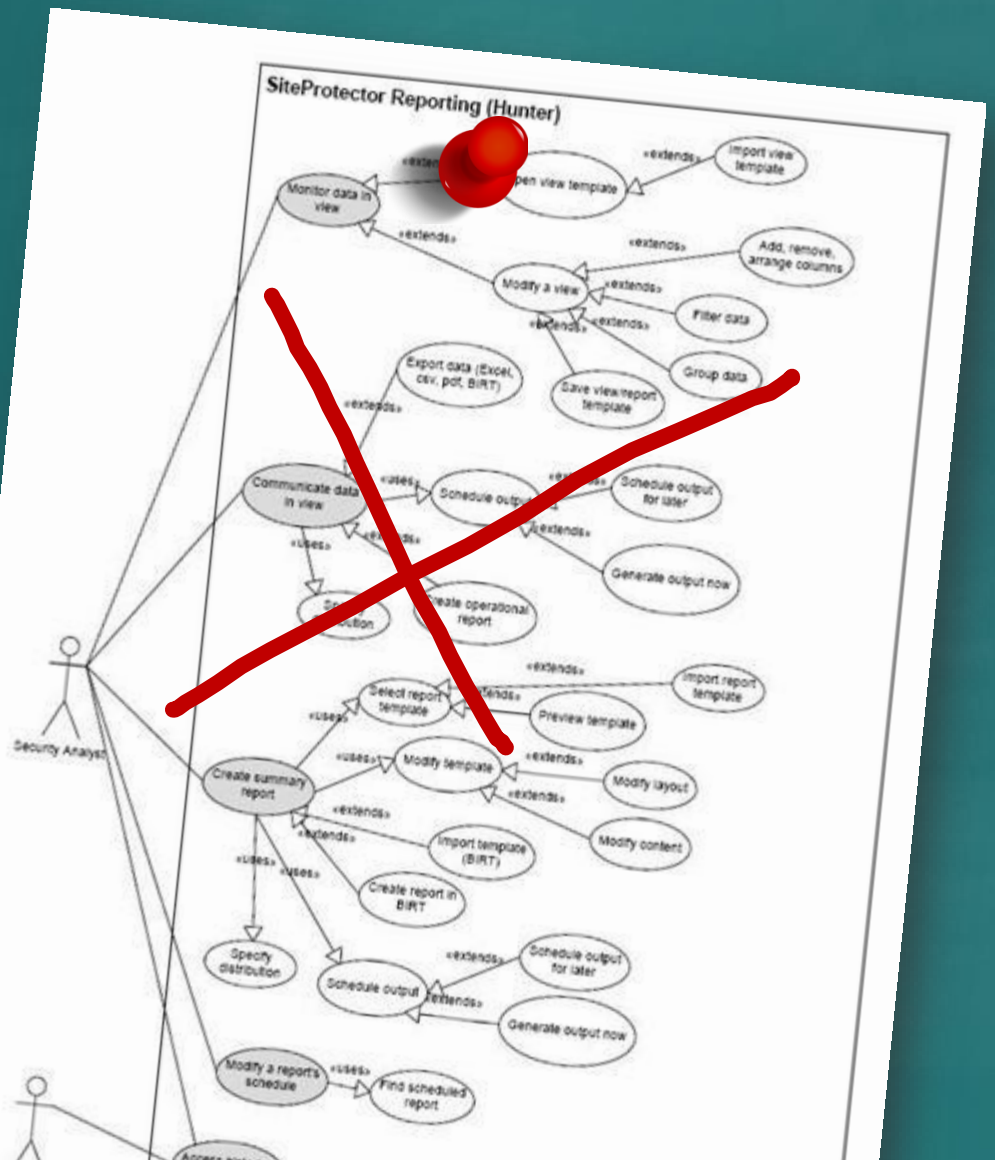
ntensions:
3a. OBS detects an error in the entered data.
3a1. OBS requests for the correct data.
3a2. User enters new data.
Steps 3a1-3a2 are repeated until the data entered are correct.
Use case resumes from step 4.

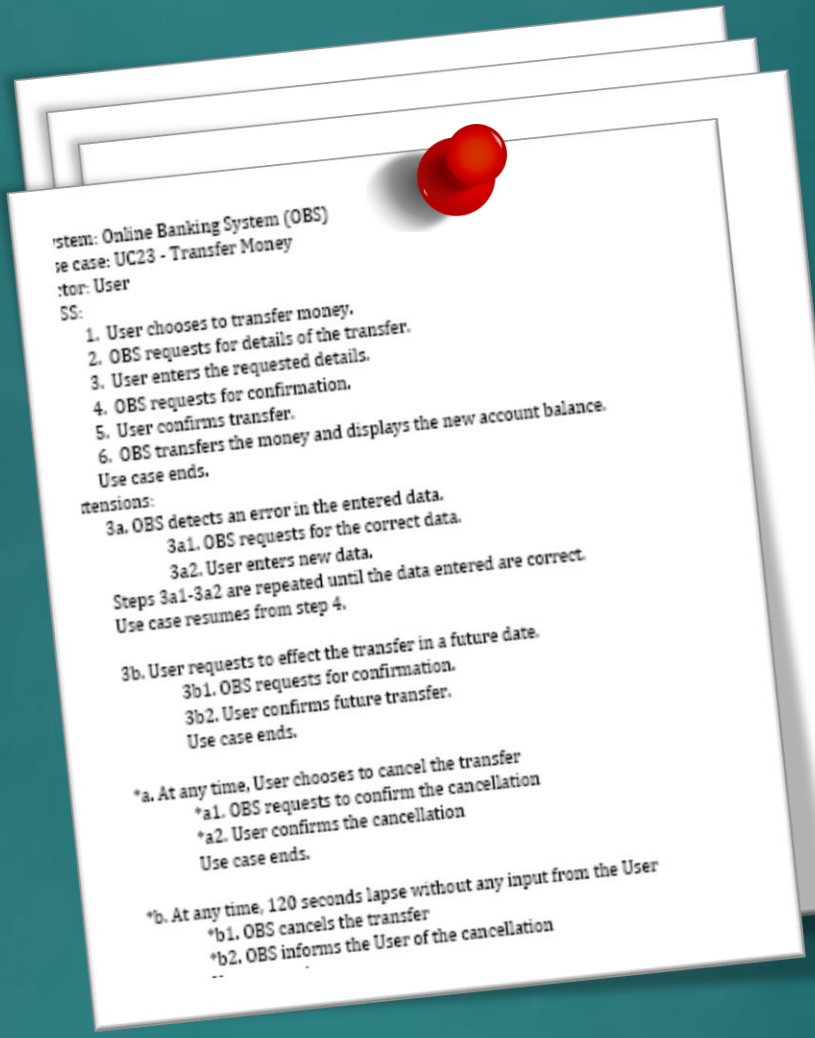
3b. User requests to effect the transfer in a future date.
3b1. OBS requests for confirmation.
3b2. User confirms future transfer.
Use case ends.

*a. At any time, User chooses to cancel the transfer
*a1. OBS requests to confirm the cancellation
*a2. User confirms the cancellation
Use case ends.

*b. At any time, 120 seconds lapse without any input from the User
*b1. OBS cancels the transfer
*b2. OBS informs the User of the cancellation

Guarantees, Preconditions, and more…. [see handout]





stem: Online Banking System (OBS)
e case: UC23 - Transfer Money
tor: User
SS:
1. User chooses to transfer money.
2. OBS requests for details of the transfer.
3. User enters the requested details.
4. OBS requests for confirmation.
5. User confirms transfer.
6. OBS transfers the money and displays the new account balance.
Use case ends.

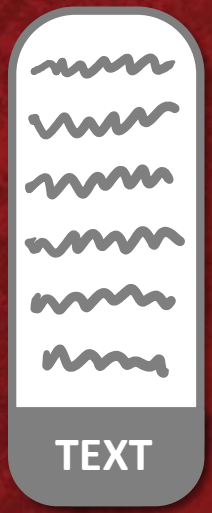
tensions:
3a. OBS detects an error in the entered data.
3a1. OBS requests for the correct data.
3a2. User enters new data.
Steps 3a1-3a2 are repeated until the data entered are correct.
Use case resumes from step 4.

3b. User requests to effect the transfer in a future date.
3b1. OBS requests for confirmation.
3b2. User confirms future transfer.
Use case ends.

*a. At any time, User chooses to cancel the transfer
*a1. OBS requests to confirm the cancellation
*a2. User confirms the cancellation
Use case ends.

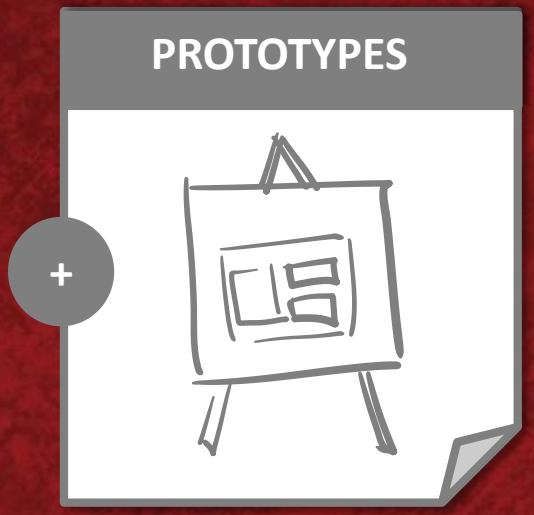
*b. At any time, 120 seconds lapse without any input from the User
*b1. OBS cancels the transfer
*b2. OBS informs the User of the cancellation

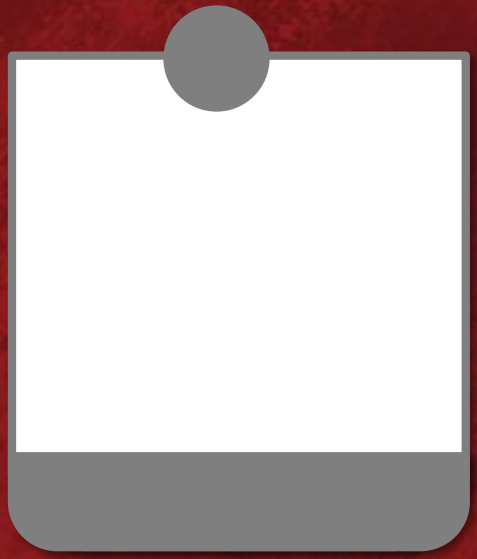
SPECIFYING requirements

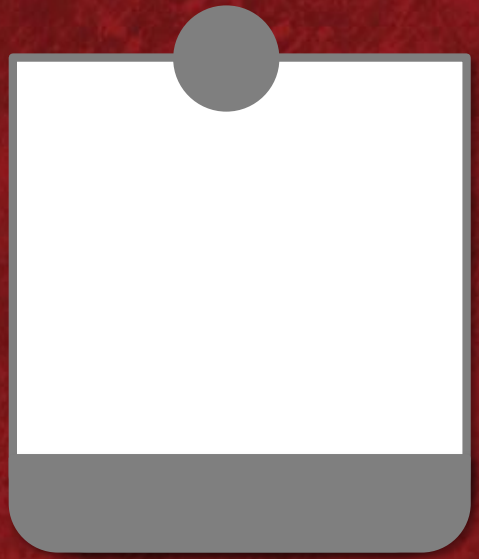


OR

+

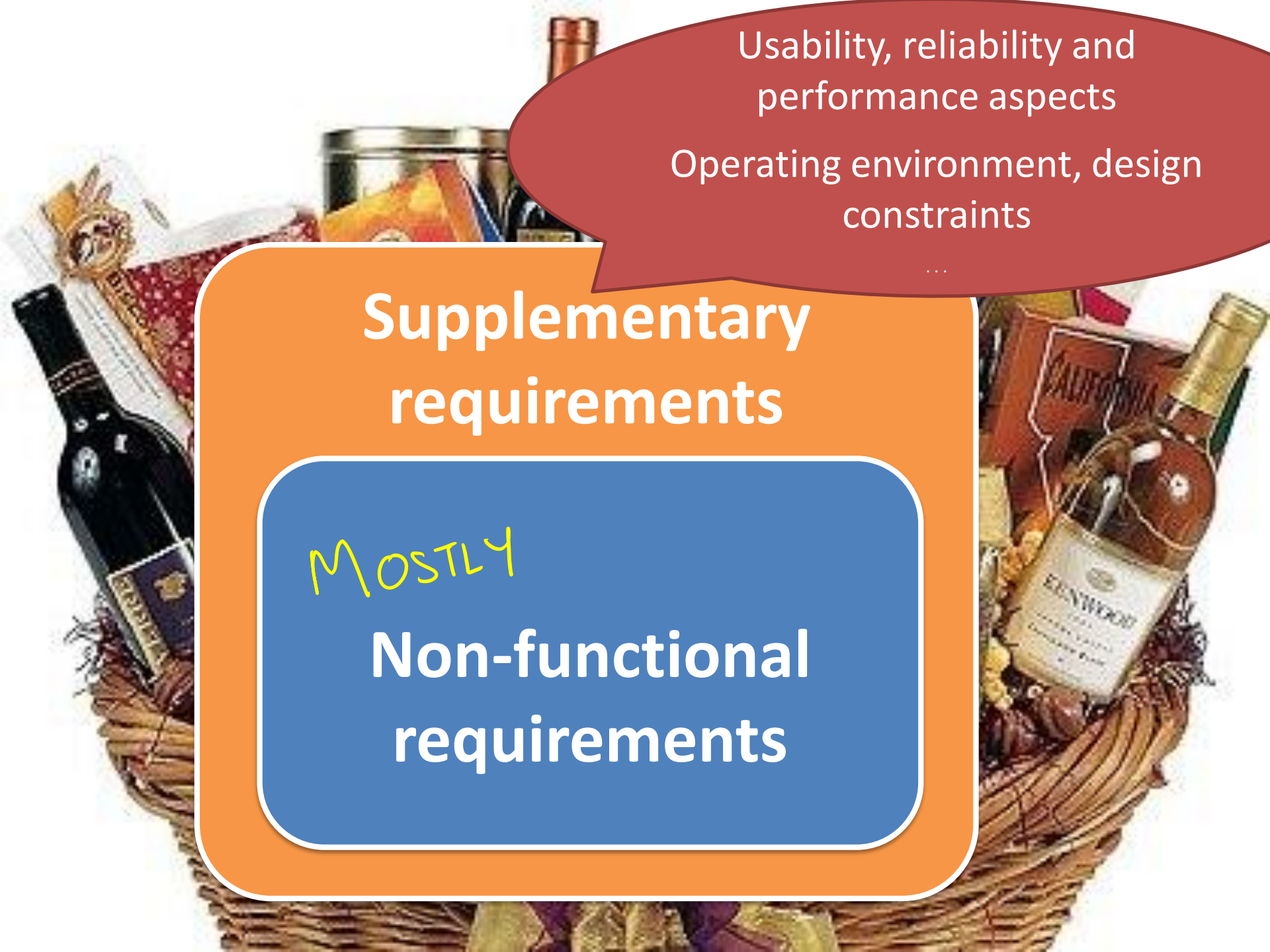




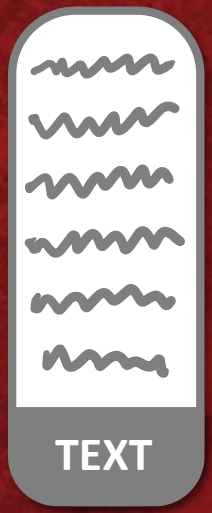


+

Supplementary requirements

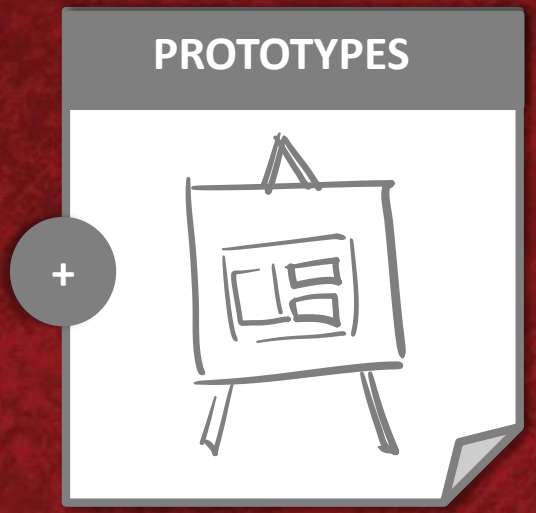

Usability, reliability and performance aspects
Operating environment, design constraints
…
Supplementary requirements
Mostly
Non-functional requirements

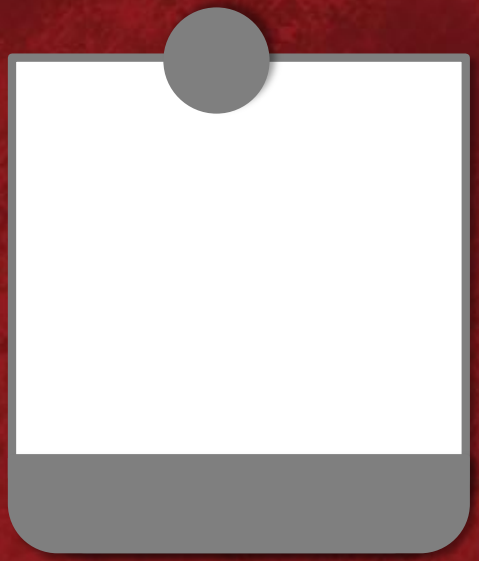
SPECIFYING requirements



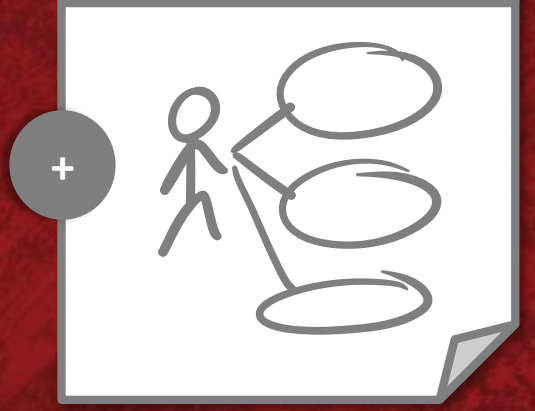
OR

+





+



"Okay your father
managed to get a mouse.
Now how do we use it?"

Glossary

SPECIFYING requirements

TEXT

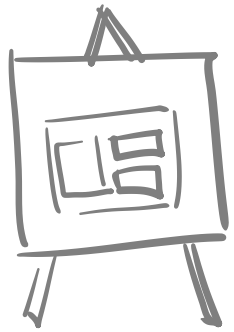
FEATURE LIST

OR

USER STORIES

\+

PROTOTYPES

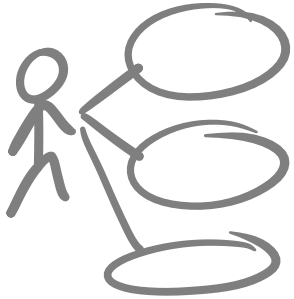


GLOSSARY

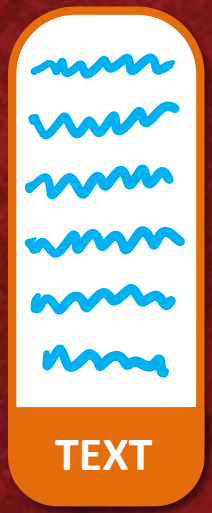
SUPPLEMENTARY

\+

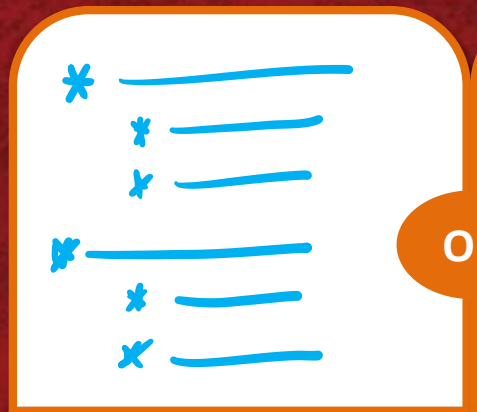
USE CASES



SPECIFYING requirements



TEXT



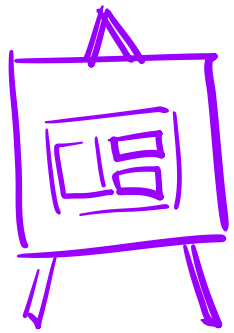
FEATURE LIST

OR

USER STORIES

\+

PROTOTYPES

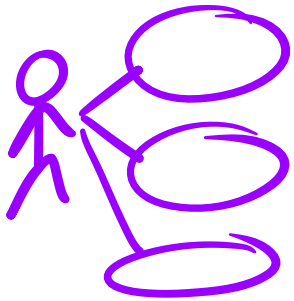


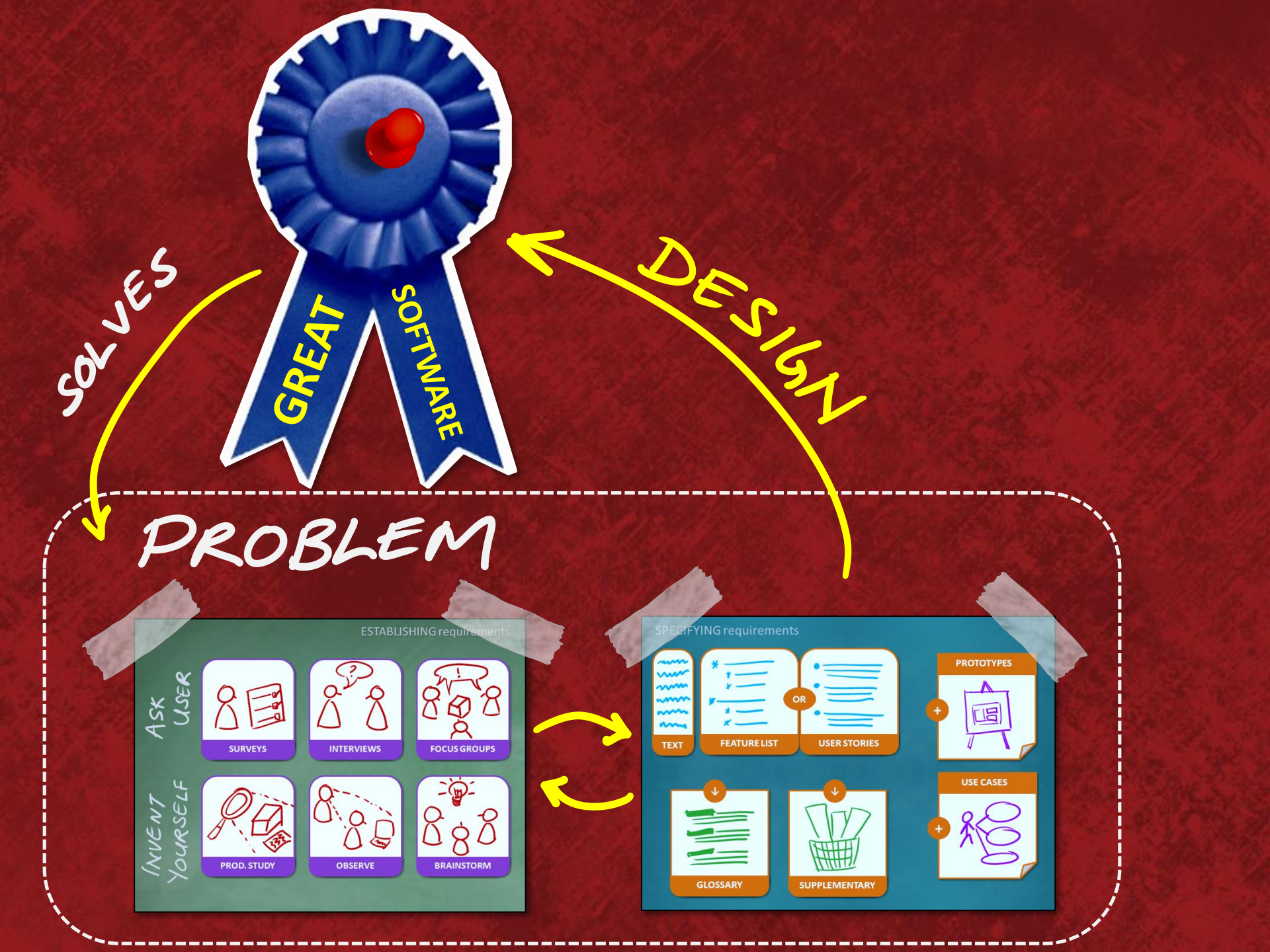
GLOSSARY

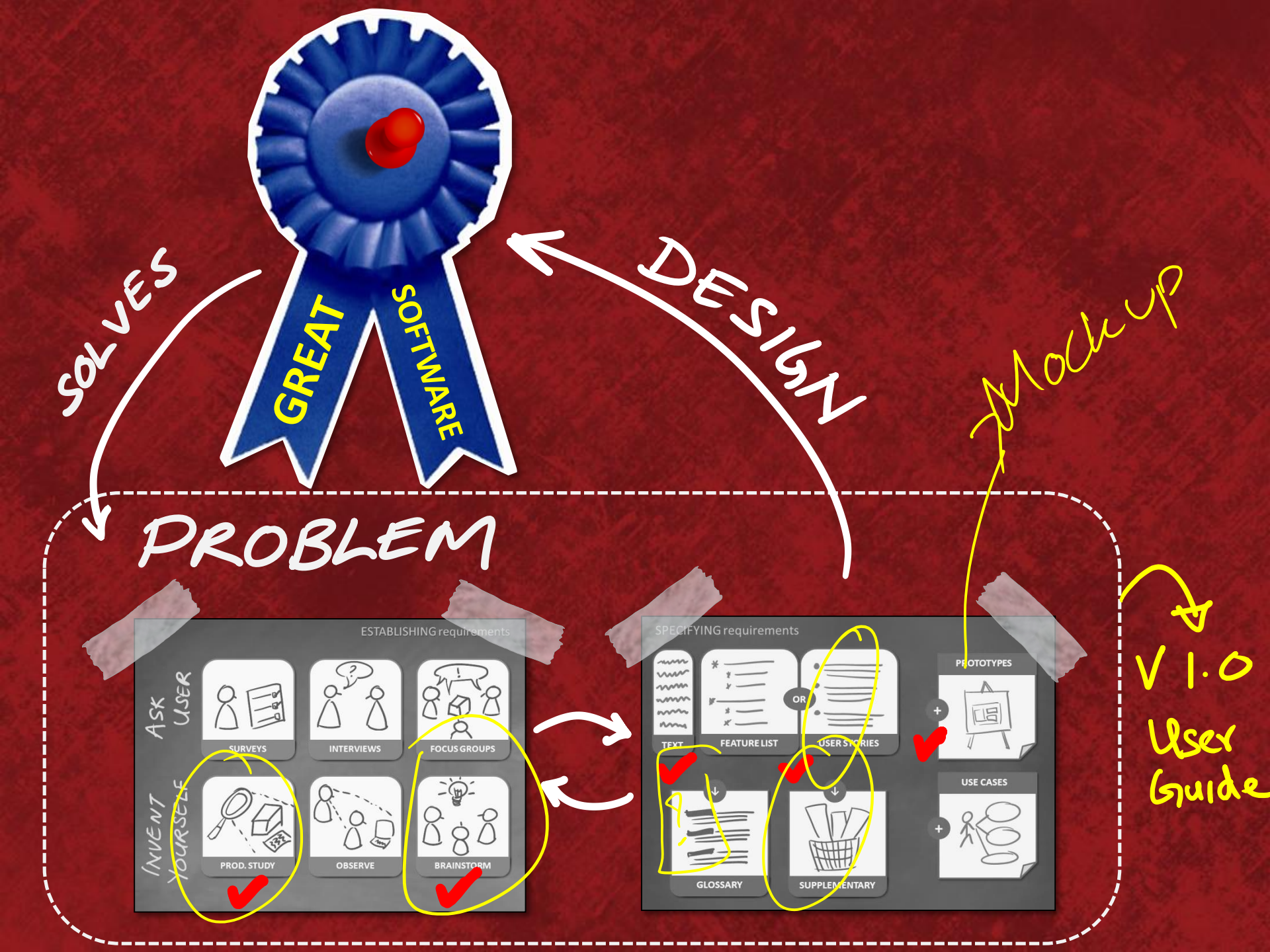
SUPPLEMENTARY

\+

USE CASES




GREAT SOFTWARE
SOLVES
DESIGN
PROBLEM
ESTABLISHING requirements
ASK USER
SURVEYS
INTERVIEWS
FOCUS GROUPS
INVENT YOURSELF
PROD. STUDY
OBSERVE
BRAINSTORM
SPECIFYING requirements
TEXT
FEATURE LIST
OR
USER STORIES
PROTOTYPES
USE CASES
GLOSSARY
SUPPLEMENTARY


GREAT SOFTWARE
SOLVES
DESIGN
Mockup
PROBLEM
ESTABLISHING requirements
ASK USER
SURVEYS
INTERVIEWS
FOCUS GROUPS
INVENT YOURSELF
PROD. STUDY
OBSERVE
BRAINSTORM
SPECIFYING requirements
TEXT
FEATURE LIST
OR
USER STORIES
PROTOTYPES
USE CASES
GLOSSARY
SUPPLEMENTARY
V 1.0
User Guide