

Objectives

- Usability
- Accessibility
- Issue Trackers
- Project

PHP for Class

- Example: Lab 0

```
<?php
$title="Lab 0: Remote Access to CS Lab Machines";
?>
<?php include("header.html"); ?>
<?php include("navtop.html"); ?>
<?php include("lab0.html"); ?>
<?php include("footer.html"); ?>
```

PHP for Class

- Example: Lab 0

lab0.php

```
<?php
$title="Lab 0: Remote Access to CS Lab Machines";
?>
<?php include("header.html"); ?>
<?php include("navtop.html"); ?>
<?php include("lab0.html"); ?>
<?php include("footer.html"); ?>
```

header.html

```
<title><?php echo $title ?></title>
```

- PHP Lite → much more you can do with PHP

Review

- Usability

- What is usability?

Review: Usability

- Engineering principles for designing and building software interfaces that are
 - Fast to learn
 - Speedy to use
 - Avoid user errors
- How to recognize and articulate the difference between “this page sucks” and “I can improve this page by changing X,Y, and Z”
- Life-long habits for engineering usable products

Simplicity

- An old quote :
“It’s easy to make things hard, it’s hard to make things easy”
- Or as Mark Twain said :
“It takes three weeks to prepare a good ad lib speech”
- Simple is hard!
 - A good interface is a lot like a good sports referee ... you never notice it’s there
- It takes knowledge, skills, talent, and diligence to design simple things

Fundamental software design principle: the 7 ± 2 Rule

- Human's short-term memory can only hold about 7 things at a time (plus or minus 2)
- That is all we can concentrate on!
 - Sports
 - Books
 - People and organizations
 - Software
 - User interfaces
- When we get more than about 7 items, we get confused

Semantic vs Syntactic Knowledge

Semantic

- What to do
- Stable in memory
- Independent of computer

Syntactic

- How to do it
- Easy to forget
- Specific to OS, language, hardware

Delete a file {
Right-click, choose delete
Drag to trash can
rm x.java
del x.java

Counting words
in a file {
wc -w x.txt
Tools → Word Count...
vim x.txt; s/ /rtn/g; :\$

Shneiderman's Measurable Criteria

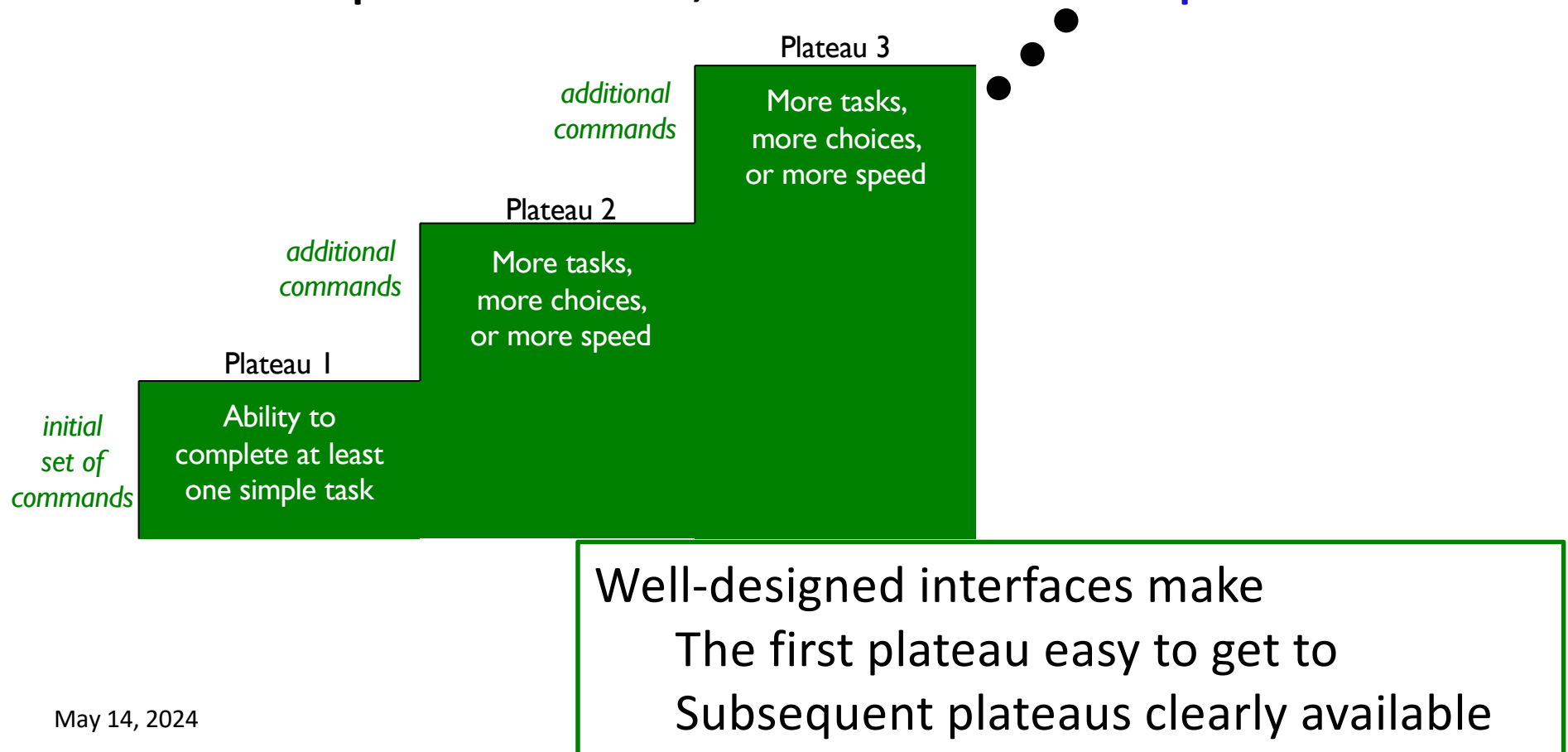
- User interface design has long been considered an **art** rather than a **science**
 - Decisions made subjectively rather than objectively
- There has been a lot of effort to make UI design more **objective**
 - an engineering activity
- Criteria is general for designing any user interface

Shneiderman's Measurable Criteria

1. **Time to learn** : The time it takes to learn some basic level of skills
2. **Speed of UI performance** : Number of UI “interactions” it takes to accomplish tasks
3. **Rate of user errors** : How often users make mistakes
4. **Retention of skills** : How well users remember how to use the UI after not using for a time
5. **Subjective satisfaction** : The lack of annoying features

1. Time to Learn

- With complicated UIs, the users must plateau



2. Speed of UI Performance

Number of UI “interactions” (roughly) it takes to accomplish tasks

- About *navigating* through the interface, **not** how fast the software or network runs
- *Interaction points* are places where the users interact with the software:
 - Buttons
 - Text boxes
 - Commands

3. Rate of User Errors

- Users will always make mistakes
- UIs can encourage or discourage mistakes
 - Consistency, instructions, navigation, ...

4. Retention of Skills

- “Once you learn to ride a **bicycle**, you never forget”
- Some interfaces are **easy to remember**, some are hard
- If they **flow logically** (that is, match the user’s **mental model** or expectations), they are very easy to remember
- If an interface is very **easy to learn**, then the retention is not important
 - **users can just learn again**
- Retention is typically more important with UIs that are **hard to learn**

5. Subjective Satisfaction

- Subjective satisfaction is how much the users “like” the UI
 - depends on the user (thus the word “subjective”)
- Think of it in reverse: Users are **unhappy** when there is something annoying in the interface
 - Blinking
 - Ugly colors
 - Spelling errors in messages
- Most important in **competitive** software systems
 - Like ... everything on the Web !

A Case Study: Looking at a File's Info

- In Unix command-line:
`ls -l`
- In GUI, pull up File manager, select the file, get its info

Metric	Command-Line	GUI
1. Time to Learn		
2. Number of UI interactions		
3. Rate of User Errors		
4. Retention of Skills		
5. Satisfaction		

A Case Study: Looking at a File's Info

- In Unix command-line:
`ls -l`
- In GUI, pull up File manager, select the file, get its info

Metric	Command-Line	GUI
1. Time to Learn	Higher	Lower
2. Number of UI interactions	1	Multiple clicks
3. Rate of User Errors	Higher	Lower
4. Retention of Skills	Lower (until reach plateau)	Higher
5. Satisfaction	So satisfied	So satisfied

Establishing Criteria Priorities

Before designing,
decide what is acceptable
for each of the five criteria

- Order of priorities
- Minimally acceptable
- Optimistic goal

Don't Make Me Think Discussion

- What were the most important points?
 - What is easy to implement?
 - What is trickier?
- What was most surprising?
- Conventions: Why should our site adhere to conventions?
- What web site did you think had good usability? Why?
- What web site did you think had poor usability? Why?

<https://wlu.box.com/v/SiteUsability>

Analysis

- Consider the “good usability” sites and the “poor usability” sites
 - What do they have in common?
 - What lessons should we take from them?

Have a Point, Make Your Point!

You have less than two minutes to convince first time visitors to stay on your web site

Every page must justify
WHY the user should stay

Analysis of Usability from DMMT

- “Trunk test” questions - Chapter 7
 - What site is this? (Site ID)
 - What page am I on? (Page name)
 - What are the major **sections** of this site?
 - What are my options at this level? (Local navigation)
 - Where am I in the grand scheme of things?
 - How can I search?
- For your sites (good and poor), answer these questions
 - Are these answers related to your thoughts on usability?

WEB ACCESSIBILITY

Web Accessibility

- Original web goal: “This is for everyone”
- Accessibility goal: Everyone should be able to access all web content
 - Global accessibility benefits everyone
 - Federal Section 508 standards – ADA law

Basic Accessible Design Principles

- Provide appropriate alternative text
 - Within the `alt` attribute of the `img` element.
 - Within the context or surroundings of the image itself
- Caption video, provide transcripts for audio
- Make file downloads (e.g., PDFs) accessible
 - Checkers for various file types

Basic Accessible Design Principles

- Do not rely on color alone to convey meaning
- Make content structured, clearly written, and easy to read
 - Make use of semantic elements

Web Accessibility Evaluation Tool



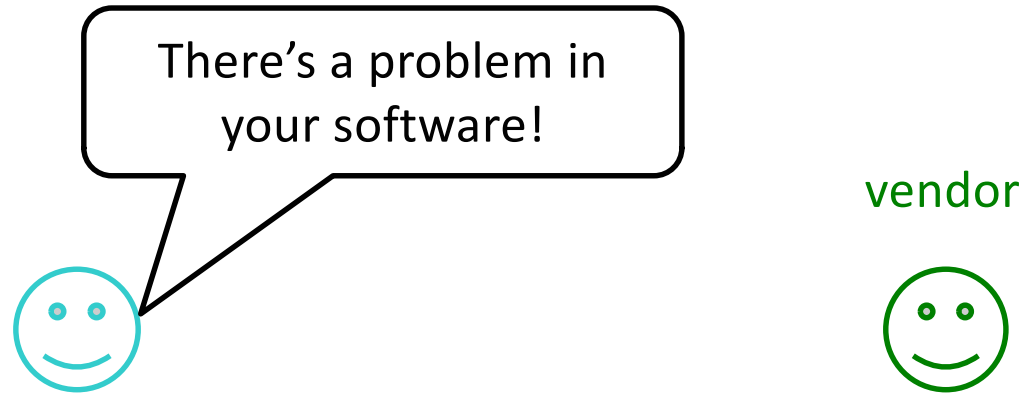
- <https://wave.webaim.org/>
 - Provides feedback on accessibility issues on web page
 - Tells you the good and the bad (what needs fixed)
 - Does not give you a “pass”
 - Requires a human

Accessibility Resources

- Intro to Web Accessibility
 - <https://www.w3.org/WAI/fundamentals/accessibility-intro/>
- Health and Human Services – check lists for various file types, explanations, ...
 - <https://www.hhs.gov/web/section-508/accessibility-checklists/index.html>

ISSUE TRACKING

Problem Life Cycle



user

** Amount of happiness may vary*

vendor

1. reproduces the problem
2. isolates the circumstances
3. locates and fixes the *defect*
4. delivers the fix to the user

What's a Problem?

- A *problem* is a questionable property of a program run
 - It becomes a *failure* if it's incorrect...
 - ...a *request* for enhancement if missing...
 - ... and a *feature* if normal behavior

It's not a bug, it's a feature!

Challenges

- How do I organize the life cycle?
- Which problems are currently *open*?
 - Haven't been diagnosed, fixed
- Which are the most severe problems?
- Did similar problems occur in the past?

Problem Report

- A problem comes to life with a *problem report*
- Includes all information vendor needs to fix problem
- Also known as *change request* or *bug report*


Example Problem Report

```
From: me@dot.com  
To: you@there.org  
Subject: Crash  
Your program crashed.  
(core dumped)
```

- Core dump: recorded state of the working memory of a computer program at a specific time, generally when the program has terminated abnormally (crashed)
- Email content similar to students' emails to me when they want to know why something went wrong in their program

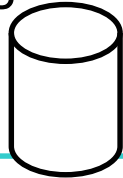
What does the report tell you?

Example Problem Report #2

```
From: me@dot.com  
To: you@there.org  
Subject: Re: Crash  
Sorry, here's the core  
 <core, 14MB>
```

Example Problem Report #3

```
From: me@dot.com  
To: you@there.org  
Subject: Re: Crash  
You may need this too,  
just in case
```

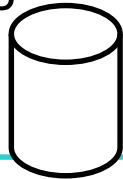


```
<data, 148GB>
```

- What's the problem with these problem reports?

Example Problem Report #3

```
From: me@dot.com  
To: you@there.org  
Subject: Re: Crash  
You may need this too,  
just in case
```



```
<data, 148GB>
```

- What's the problem with the problem reports?
 - Limited information about what the problem is, what caused it
 - Information is across 3 emails

What To Report

- The product release
- The operating environment
- The problem history
- A one-line summary
- Expected and experienced behavior

Product Release

- Typically, some *version number* or otherwise unique identifier
 - Required to reproduce the problem

Perfect Publishing Program 1.1 (Build 7E47)

- Generalize: Does the problem occur only in this release?

Operating Environment

- Typically, *version information* about the operating system
- Can be simple (“Windows 10”) or complex (“Ubuntu Linux 20.04.1LTS with the following packages...”)
- Generalize: In which environments does the problem occur?

Problem History

- Steps needed to *reproduce* the problem
 1. Create “bug.ppp”
 2. Print on the default printer...
- If the problem cannot be reproduced, it is unlikely to be fixed
- Simplify: Which steps are relevant?

Expected Behavior

- What should have happened according to the user:

The program should have printed the document.

- Reality check: What is the understanding of the user?

Observed Behavior

- The *symptoms* of the problem — in contrast to the expected behavior

```
The program crashed with the following
information:
```

```
*** STACK DUMP OF CRASH (LemonyOS)
```

```
Back chain  ISA  Caller
00000000    SPC  0BA8E574
03EADF80    SPC  0B742428
03EADF30    SPC  0B50FDDC PrintThePage+072FC
SnicketPC unmapped memory exception at
          0B512BD0 PrintThePage+05F50
```

A One-Line Summary

- Captures the essence of the problem

PPP 1.1 crashes when printing

If we're developing a large software application,
as good as we may be, we're going to have bugs...

A lot of them....

Managing Problems

- **Alternative #1: *A Shared Problem File***
 - Anyone with permission can update any part of it (do you want that?)
- **Alternative #2: *A Problem Database***
 - Examples: Bugzilla, JIRA

Classifying Problems

- Severity
- Priority
- Identifier
- Comments
- Notification

Problem Severity

- **Enhancement.** A desired feature
- **Trivial.** Cosmetic problem
- **Minor.** Problem with easy workaround
- **Normal.** “Standard” problem
- **Major.** Major loss of function
- **Critical.** Crashes, loss of data or memory
- **Showstopper.** Blocks development

Priority

- Every new problem is assigned a *priority*
- The higher the priority, the sooner the problem will be addressed
- Priority is *independent* from severity
- Prioritizing problems is the main tool to control development and problem solving

Identity

- Every new problem is assigned an *identifier*
 - Also known as PR—problem report—number or bug number
- The identifier is referenced in all documents during the debugging process

Subject: PR #3427 is fixed?

- Included in your commit comments

Comments

- A developer can attach *comments* to a problem:

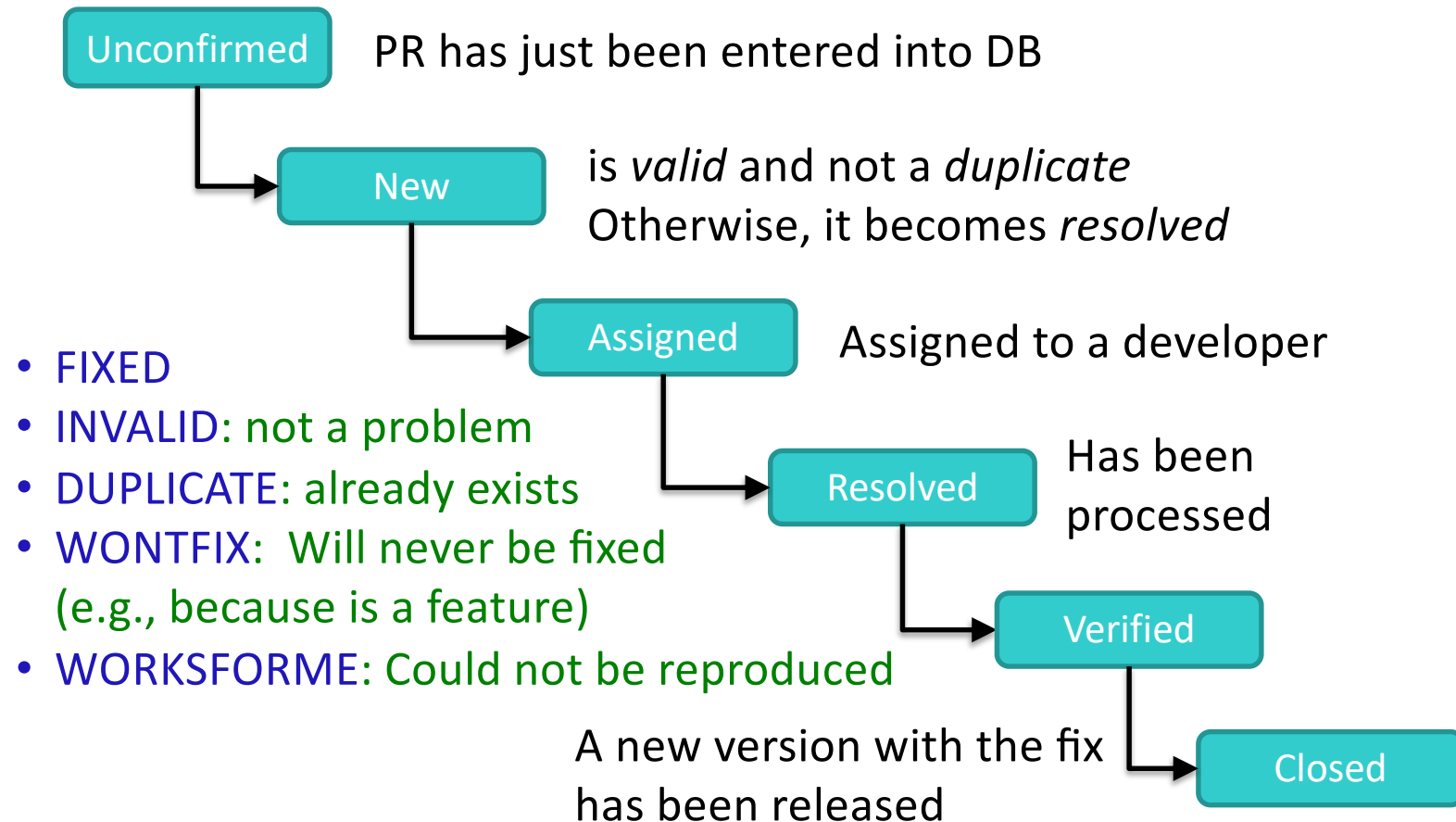
```
I have a patch for this. It's  
just an uninitialized variable,  
but I still need a review.
```

- Comments may also include files, documents, etc.

Notification

- Developers and users can attach an e-mail address to a problem report
- They will be notified every time the report changes

Simplified Problem Lifecycle



Management

- Who enters problem reports?
- Who classifies problem reports?
- Who sets priorities?
- Who takes care of the problem?
- Who closes issues?

Summary

- Reports about problems encountered in the field are stored in a *problem database*
- A problem report must contain everything relevant to reproduce the problem
- It is helpful to set up a standard set of items that users must provide (product release, operating environment...)

Problem Reports Summary

- An effective problem report...
 - is well-structured
 - is reproducible
 - has a descriptive one-line summary
 - is as simple and general as possible
 - is neutral and stays with the facts

Issue Tracking Summary

- A typical problem life cycle starts with an *unconfirmed* status
- It ends with a *closed* status and a specific resolution (such as fixed or workforme)

Issue Tracking Tools

- Bugzilla
- Jira
- TRAC (+ wiki)
- GitHub Issues

PROJECT

May 14, 2024

Sprenkle - CSCI335

59

Justifications

- Why Starting with Writing/Diagramming?
 - Reveals misunderstandings and gaps in knowledge
 - To you (I hope!) and to me

Issues in AGP

- Issues aren't always well-specified
 - We don't have an analyst to create the issues
 - Looking for ideas/solutions
- Issues with issues
 - We may find that something isn't possible or isn't something we actually want

Review

- What is the *production* server vs the *development* server?
- Review use cases, tracing through the code
 - User selects the list of properties
 - User selects one property from the list of properties
- How does Spring (e.g., Boot, Data, MVC) make web development easier for us? How does it make it harder?

Understanding Use Cases

- List of properties
 - Select one property
-
- What does Spring handle for us?

Looking Ahead

- Warm up: writing Javadoc comments
 - Expecting merge conflicts to resolve
- Starter functionality: Thursday midnight
- Exam Friday