

## Objectives

- Review: Maven, Project Organization
- Software Engineering Tool
  - Issue Tracking: Jira
- Project Organization
  - Preparing for Use Cases

May 10, 2021

Sprenkle - CSCI335

1

1

## Review

- DB: How can you access the database for your project from the terminal?
- What is Maven?
  - What are the benefits of Maven?
  - What is the philosophy of Maven? (“C over C”)
    - Why is that beneficial?
- How does the MVC design pattern apply to your project?
  - Yes, you’re working on different projects. Explain to your neighbors!
- What do **you** do to understand a new code base?
- We’re at the halfway point of the semester: how does your resume look?

May 10, 2021

Sprenkle - CSCI335

2

2

# 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 Lite → much more you can do with PHP

May 10, 2021

Sprenkle - CSCI335

3

3

# 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

May 10, 2021

Sprenkle - CSCI335

4

4

# ISSUE TRACKING

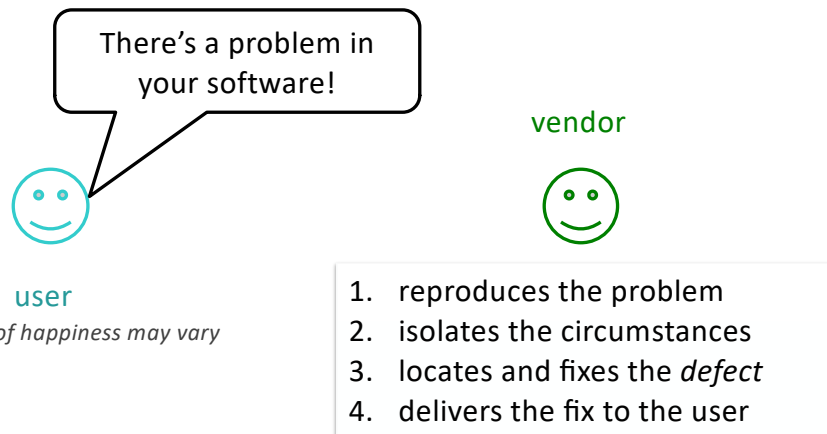
May 10, 2021

Sprenkle - CSCI335

5

5

## Problem Life Cycle



May 10, 2021

Sprenkle - CSCI335

Zeller

6

6

## 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!*

7

## 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?

8

## 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**

May 10, 2021

Sprenkle - CSCI335

Zeller

9

9

## 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?

May 10, 2021


Sprenkle - CSCI335

Zeller

10

10

## Example Problem Report #2

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

May 10, 2021

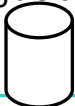
Sprenkle - CSCI335

Zeller

11

11

## 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?

May 10, 2021

Sprenkle - CSCI335

Zeller

12

12

## 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?

May 10, 2021

Sprenkle - CSCI335

Zeller

17

17

## 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?

May 10, 2021

Sprenkle - CSCI335

Zeller

18

18

## 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 Problem File***
  - Only one person at a time can work on it
  - History of earlier (fixed) problems is lost
  - *Does not scale*
- **Alternative #2: *A Problem Database***
  - Examples: Bugzilla, JIRA

## Classifying Problems

- Severity
- Priority
- Identifier
- Comments
- Notification

23

## 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

24

## 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

25

## 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

26

## Comments

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

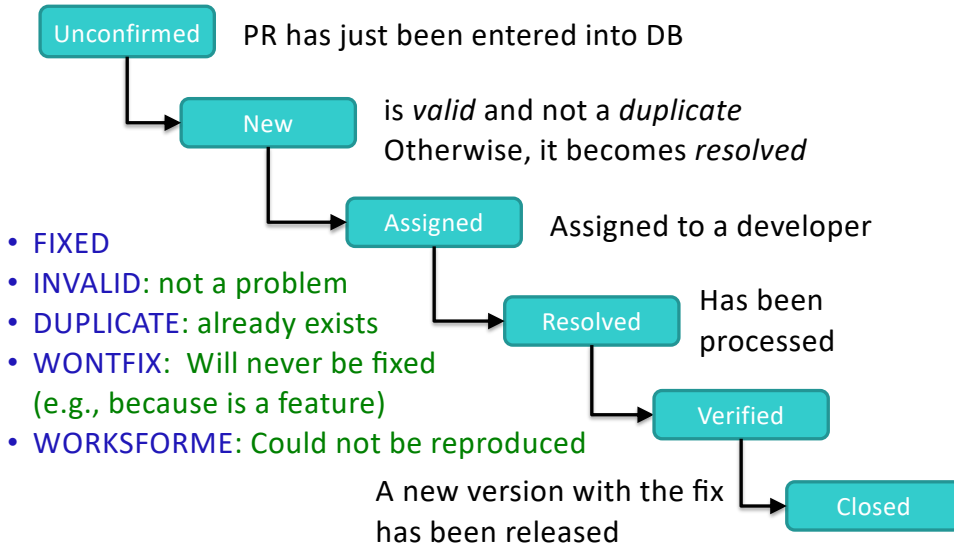
```
I have a patch for this. It's  
just an uninitialized variable,  
but I still need a review.
```

- 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



May 10, 2021

Sprenkle - CSCI335

Zeller

29

29

## Management

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

May 10, 2021

Sprenkle - CSCI335

Zeller

30

30

## 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 worksforme)

33

## Issue Tracking Tools

- Bugzilla
- Jira 
- TRAC (+ wiki)

34

## Using Jira

- Add Requirements/Features/Bugs to JIRA
  - assign to team members
- Creates TODO lists
- Can mark when you've completed task, including the git commit code

May 10, 2021

Sprenkle - CSCI335

35

35

## Project Organization Discussion

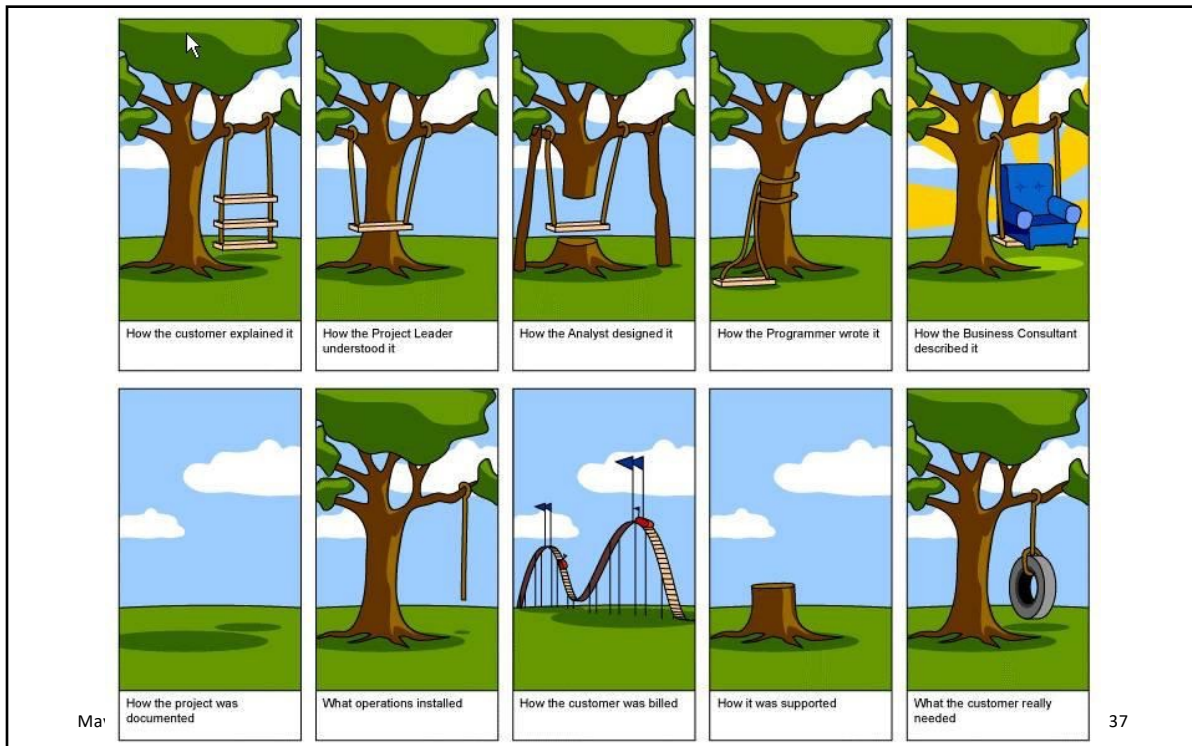
- Professor Sprenkle rotating among teams
  - Discussing documentation and understanding
  - Revise, clarify documentation based on discussion

May 10, 2021

Sprenkle - CSCI335

36

36



37

## Exam Preparation

- Wed on Canvas, 2 hours between 9 a.m. – 11:59 p.m.
  - Part on Canvas
  - Part in Word – downloadable document that you complete
  - Open notes, slides, book, and mind. Closed everything else
- Prep document on course web site

38

## To Do

- HW – Read Don't Make Me Think
  - Respond on Canvas
- Review initial project functionality
- Exam: Wednesday