CS 563: Software Maintenance And Evolution

Introduction

Oregon State University, Spring 2024

CS 563 Instructor

Manish Motwani
Office: KEC 3043
Office hours: Tuesdays 2-3 PM
Email: motwanim@oregonstate.edu
Web: mmotwani.com

A little about me:

- Grew up in India and did B-Tech in CSE (2011)
- Worked 4 years as a Systems Engineer/Researcher at TRDDC, Pune, India (2011-15)
- MS/PhD from UMass Amherst (2015-22)
- Postdoc from Georgia Tech (2022-23)
- Assistant Professor at Oregon State (2023 onwards)

Today’s plan

- Introductions
- What is CS 563 about and why you should take it?
- Foundations – Intro to software maintenance

Today’s plan

- **Introductions**
  - Your name
  - Your program and what year are you in
  - What do you expect to learn from this course?

What is CS 563 about? The bigger picture

- **CS 361, SOFTWARE ENGINEERING I**: front-end software development (requirements analysis and specification; design techniques; project management)
- **CS 362, SOFTWARE ENGINEERING II:** back-end software development (verification and validation; debugging; maintenance)
- **CS 561, SOFTWARE ENGINEERING METHODS:** methods and supporting tools in the context of different SDLC methods (e.g., agile, waterfall)
- **CS 562, SOFTWARE PROJECT MANAGEMENT:** managing software projects to deliver high-quality systems in timely and cost-effective manner
- **CS 563, SOFTWARE MAINTENANCE AND EVOLUTION:** maintaining and evolving mature and deployed software systems with constantly changing technology and system requirements
- **CS 569, SELECTED TOPICS IN SOFTWARE ENGINEERING: PROGRAM ANALYSIS AND EVALUATION:** methods and techniques to verify and validate software systems

What is CS 563 about?

- **CS 563, SOFTWARE MAINTENANCE AND EVOLUTION:** maintaining and evolving mature and deployed software systems with constantly changing technology and system requirements
- This course assumes that you know some basic programming!
- Please visit https://mmotwani.com/courses/SkillAssessment.html to brush-up the recommended skillset required to do homework and project.
- The course focuses on state-of-the-art techniques for software maintenance and evolution
- State-of-the-art means exploring or cutting-edge research
- Students will learn the latest techniques in improving and maintaining software quality
- Students will advance the state-of-the-art by developing their own techniques!

What is CS 563 about?

- Learning the state-of-the-art techniques to assist software maintenance and evolution
- Hands-on learning by doing in-class exercises, homework, and long-term project
- Hands-off learning by presenting and discussing about recently published research papers related to the topics
- **Please bring a laptop with admin access from next class**

What is CS 563 about?

- **CS 563, SOFTWARE MAINTENANCE AND EVOLUTION:** maintaining and evolving mature and deployed software systems with constantly changing technology and system requirements
- **TOPICS COVERED:**
  - Introduction and basics
  - Software change management
  - Fault localization
  - Program repair
  - Software debloating
  - Mining software repositories
  - Software refactoring

How to succeed in CS 563?

- **Communicate!** – software engineering by nature is collaborative, you need to develop your team skills in working with other people
- **Attend all classes** – the latter portion of the class time is kept for hands-on work (in-class exercises, project discussion, etc.), which may continue in the following class
- **Spend time in reading research papers** – important for you to develop critical-thinking and problem-solving ability. We will do some of that in class too!
- **Talk to me if you are struggling** – use office hours to seek guidance on anything that you are struggling with for doing well in course. it is my job to help you!

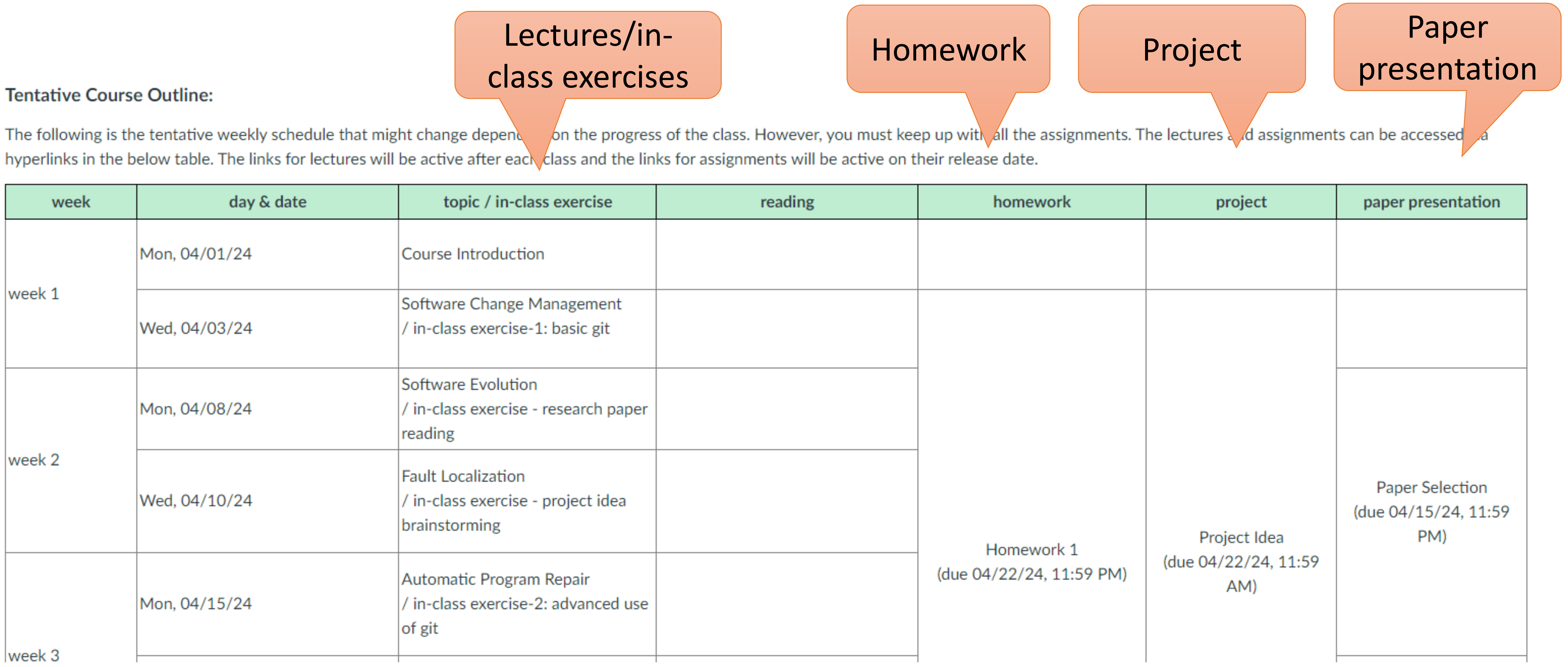
CS 563 policies

- **Grading policy:**
  - Class participation: 5%
  - Paper presentation: 20% (10% reviews, 10% oral)
  - Homework: 20% (10% each homework)
  - Mid-term exam: 15%
  - Project: 40% (15% project idea presentation and report,
    10% project plan presentation and report,
    15% final presentation and report)
  - > 90: A; 80 – 89: A-; 70 – 79: B+; 60 – 69: B; 50 – 59: B-; < 50: F
- **Late policy:** All submission deadlines are sharp, **but you will have automatic 1-day extension** on all assignments, no questions asked, and no emails needed.
- **Attendance policy: Attendance is mandatory** due to the nature of this course. All classes will be in-person and not recorded.

CS 563 material

- All the lectures and assignments related to in-class exercises, homework, paper presentation, and project will be available on Canvas (https://canvas.oregonstate.edu/)
- Please check announcements on Canvas to stay updated.
- All submissions and grading will be done via Canvas.
- Use Canvas Discussions for asking questions and sharing ideas.

CS 563 Content Organization



Tentative Course Outline:

The following is the tentative weekly schedule that might change depen on the progress of the class. However, you must keep up wit all the assignments. The lectures d assignments can be accessed a hyperlinks in the below table. The links for lectures will be active after eac class and the links for assignments will be active on their release date.

week	day & date	topic / in-class exercise	reading	homework	project	paper presentation
week 1	Mon, 04/01/24	Course Introduction				
	Wed, 04/03/24	Software Change Management / in-class exercise-1: basic git		Homework 1 (due 04/22/24, 11:59 PM)	Project Idea (due 04/22/24, 11:59 AM)	
week 2	Mon, 04/08/24	Software Evolution / in-class exercise - research paper reading				Paper Selection (due 04/15/24, 11:59 PM)
	Wed, 04/10/24	Fault Localization / in-class exercise - project idea brainstorming				
week 3	Mon, 04/15/24	Automatic Program Repair / in-class exercise-2: advanced use of git				

https://canvas.oregonstate.edu/courses/1970765

CS 563 Nondiscrimination policy

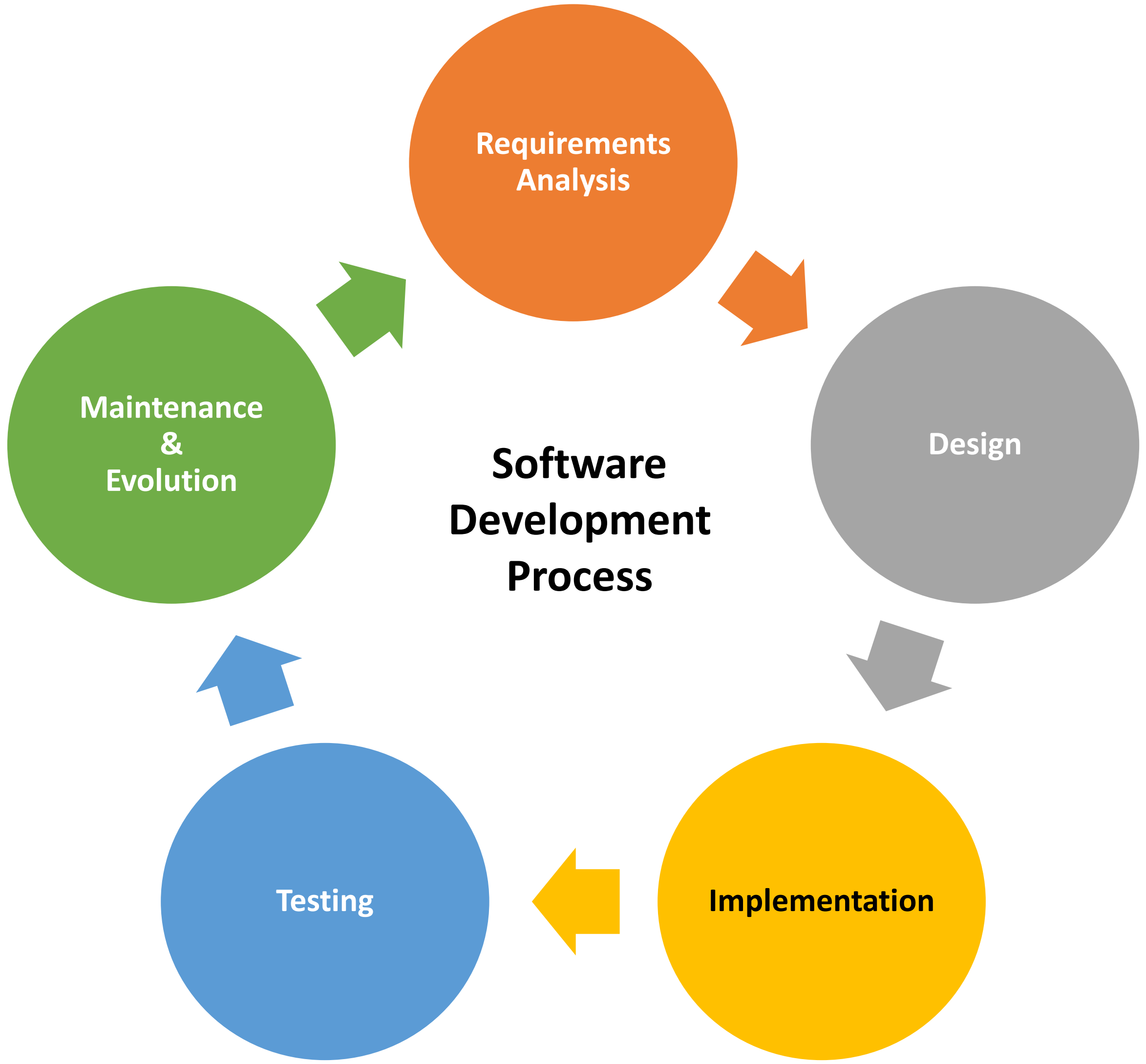
Software engineering is at its nature a collaborative activity and it benefits greatly from diversity. This course includes and welcomes all students regardless of age, background, citizenship, disability, sex, education, ethnicity, family status, gender, gender identity, geographical origin, language, military experience, political views, race, religion, sexual orientation, socioeconomic status, and work experience. Our discussions and learning will benefit from these and other diverse points of view.

Any kind of language or action displaying bias against or discriminating against members of any group or making members of any group uncomfortable are against the mission of this course and will not be tolerated. The instructor welcomes discussion of this policy and encourages anyone experiencing concerns to speak with him.

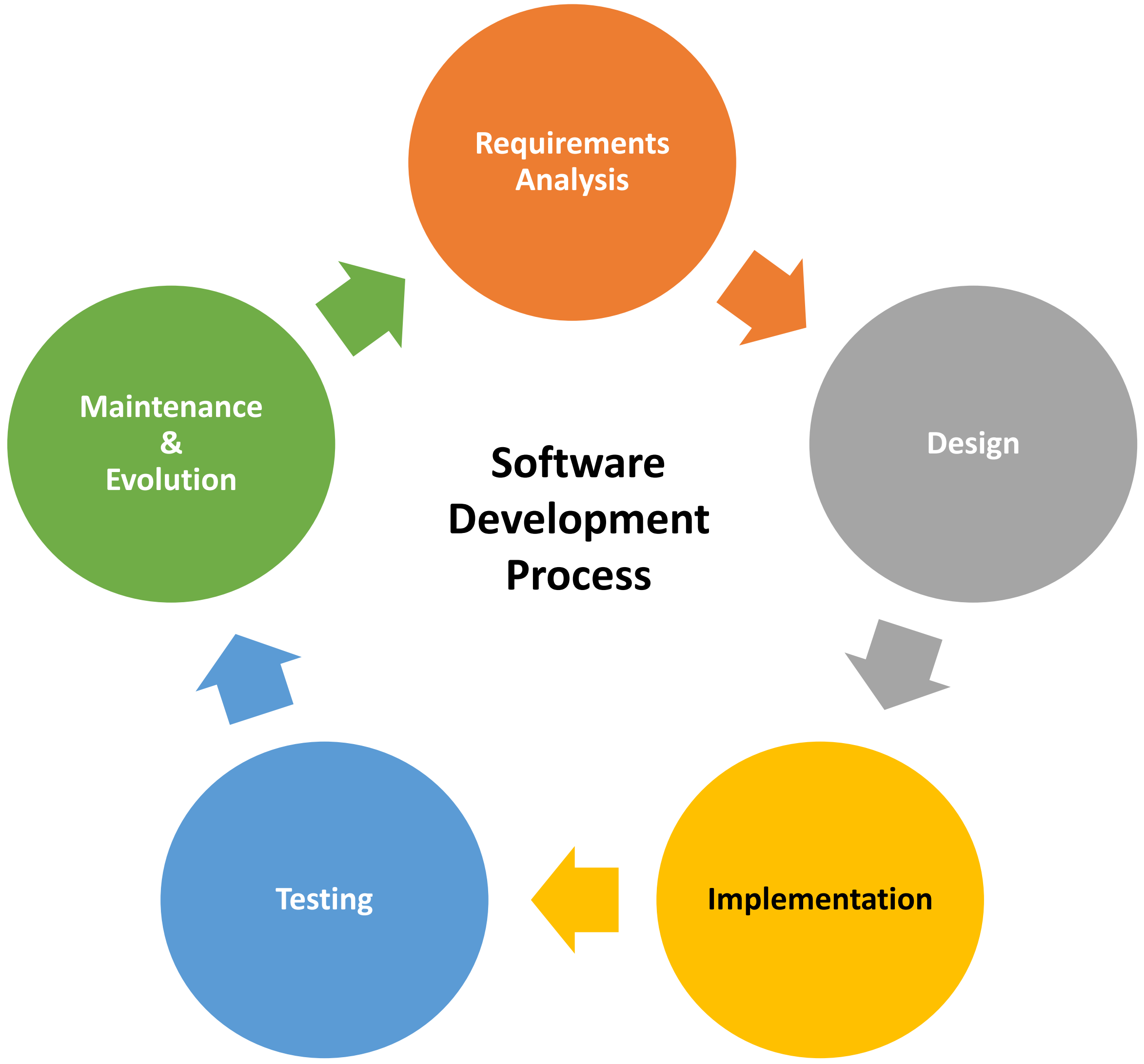
CS 563 Academic integrity

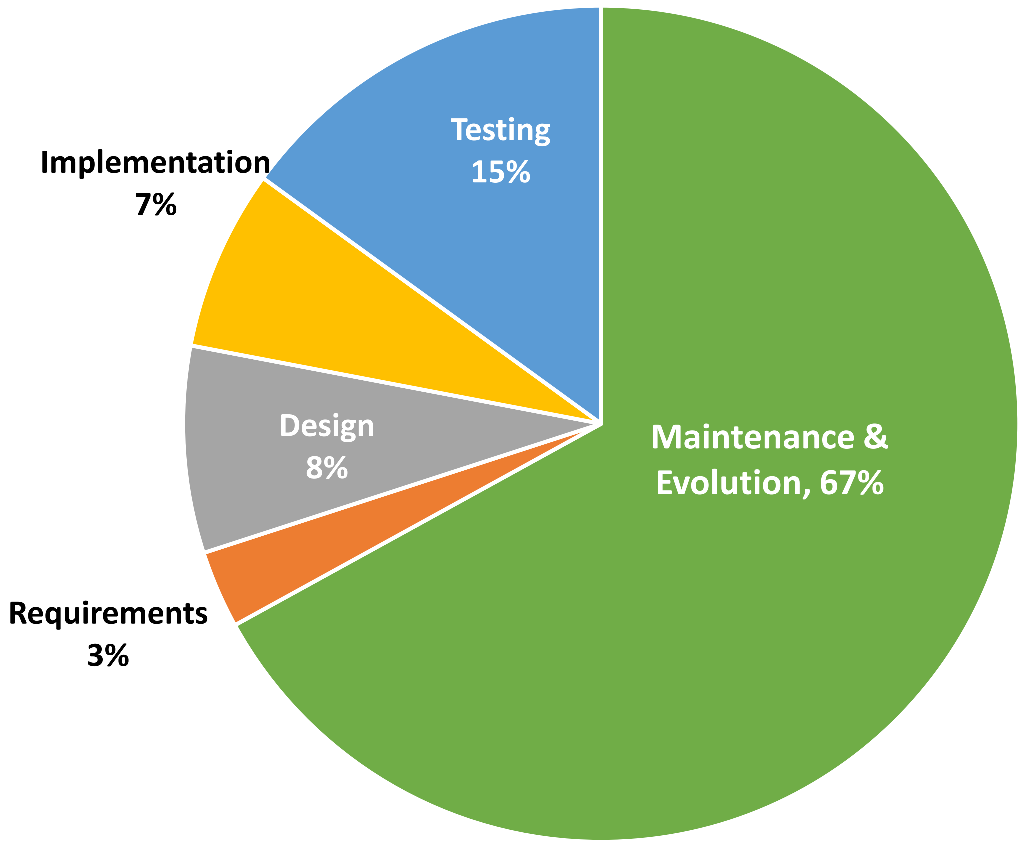
- **Students are allowed to work together on all aspects of this class. However, for the homework assignments, each student must submit his or her own write up, clearly stating the collaborators. Your submission must be your own. When in doubt, contact the instructor about whether a potential action would be considered plagiarism.** If you discuss material with anyone besides the class staff, acknowledge your collaborators in your write-up. If you obtain a key insight with help (e.g., through library work or a friend), acknowledge your source and write up the summary on your own. It is the student's responsibility to remove any possibility of someone else's work from being misconstrued as the student's. Never misrepresent someone else's work as your own. It must be absolutely clear what material is your original work. Plagiarism and other anti-intellectual behavior will be dealt with severely. **Note that facilitation of plagiarism (giving your work to someone else) is also considered to be plagiarism and will carry the same repercussions.**
- Students are encouraged to use the Internet, literature, and other publicly-available resources, except the homework solutions and test (including quizzes, midterms, finals, and other exams) solutions, from past terms' versions of this course and other academic courses, whether at OSU and at other institutions. **To reiterate, the students are not allowed to view and use past homework and test solutions, unless explicitly distributed by the CS 569 instructor as study material. Whenever students use Internet (including ChatGPT), literature, and other publicly-available resources, they must clearly reference the materials in their write ups, attributing proper credit. This cannot be emphasized enough: attribute proper credit to your sources**. Failure to do so will result in a zero grade for the assignment and possibly a failing grade for the class, at the instructor's discretion. **Copying directly from resources is not permitted, unless the copying is clearly identified as a quote from a source. Most use of references should be written in the words of the student, placing the related work in proper context and describing the relevant comparison.**

Foundations: SDLC



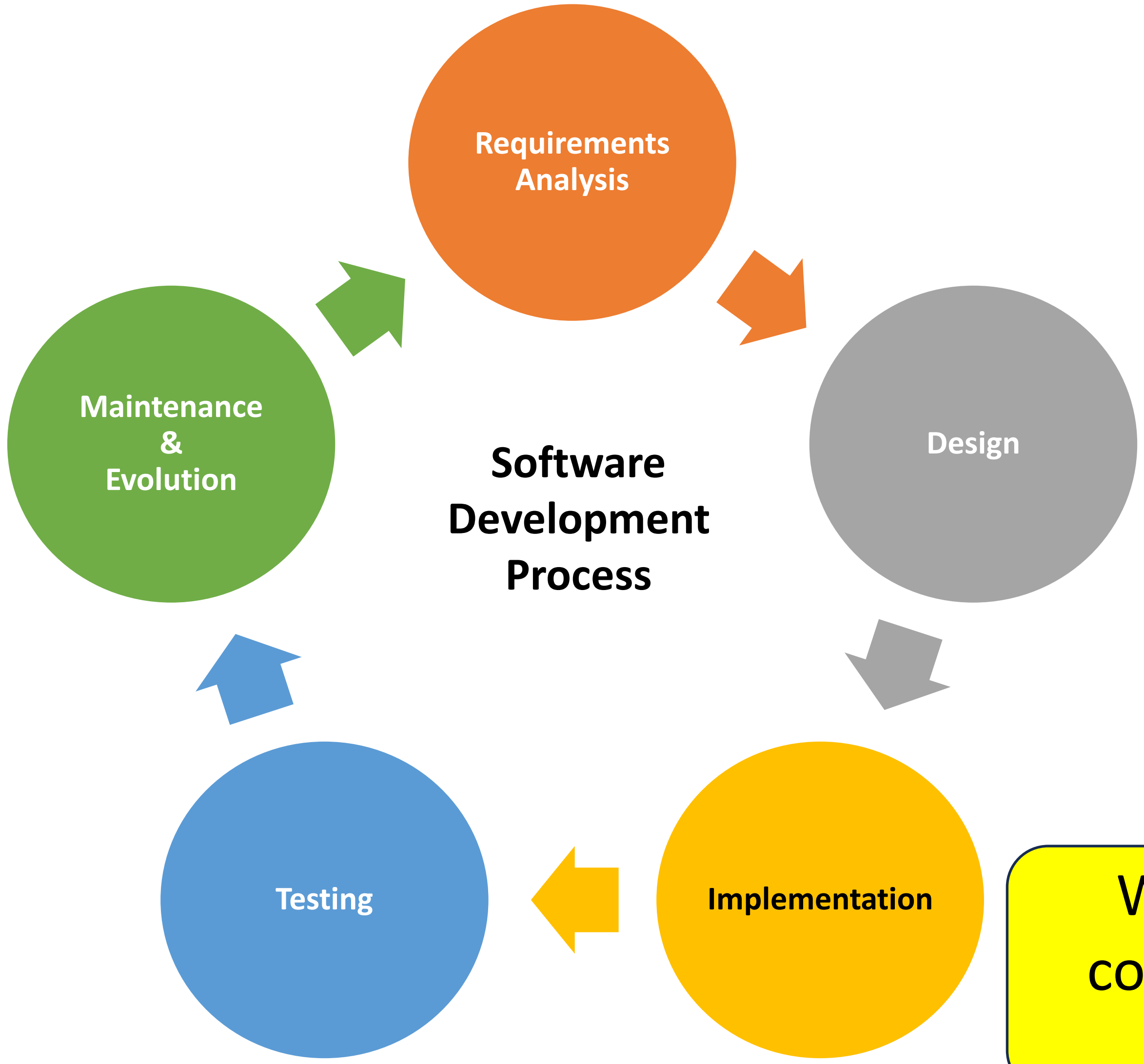
True Cost of Software Development

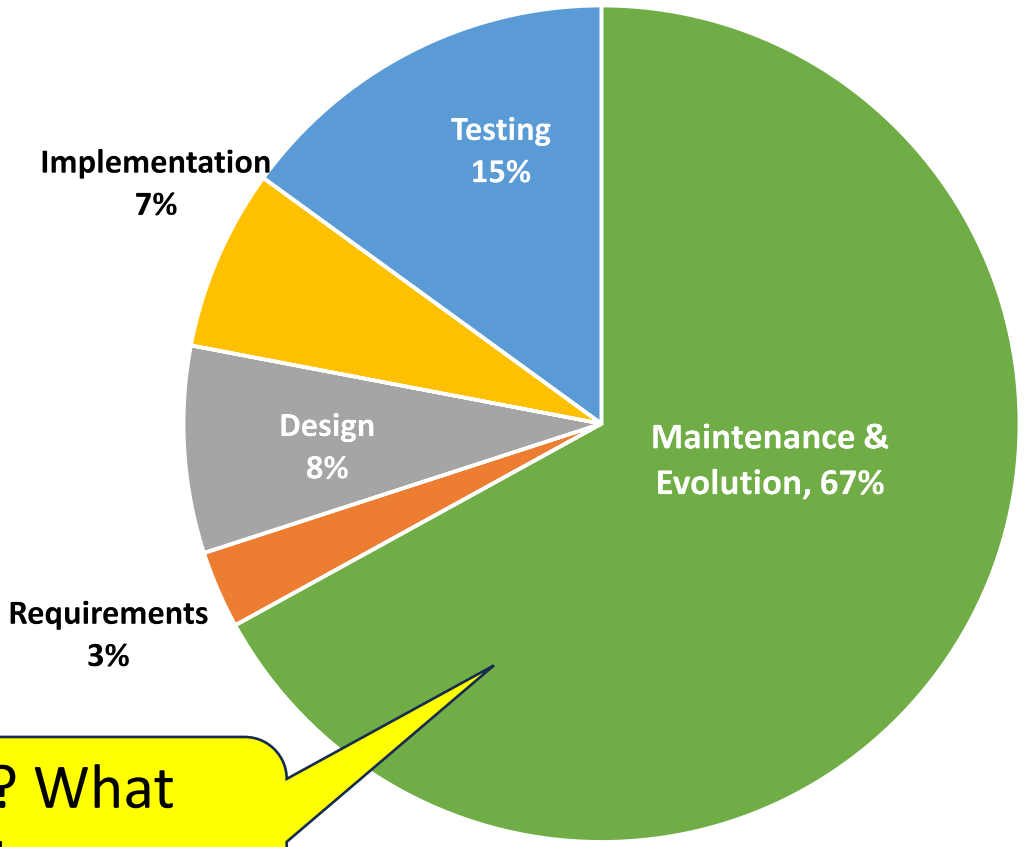




Object-Oriented and Classical Software Engineering, Schach 1999

True Cost of Software Development





Why? What contributes to this cost?

The Costs of Maintenance

- “Help Desk” – type support (person hours)
- Defect / Failures fixes
- New features
- Changes in the environment
- Business needs change
- Changes to all supplemental material (docs, manuals, etc.)

Maybe not “Maintenance”?

- “Maintenance” is almost seen as a dirty word in development
- Something “less than” new development
- Maybe:
    - Software Evolution?
    - Improvement and Modification?
    - Lifecycle support?
    - For simplicity, lets call it “Maintenance” but its much more!
- Software is a “living” creation and is expected to change
- The first version of software is rarely the “right” version.

Scenario

- Consider a custom software/hardware product that costs $2M to develop
- Problems have arisen!
  - The custom hard disk in which all the data is stored has run out of room!
  - However, the company that build the original disk is no longer in business…
  - You could buy from a new vendor, but to make it work (Adapter pattern!) would cost upwards of $100K
  - Also, the original devs have left the company …
  - So, you now have new devs trying to make sense of everything …

Maintenance is “all the SDLC phases”

- In this scenario, we have:
  - A new/modified requirement created from a defect report
  - A need for doing redesign (how will the adapter pattern work here?)
  - Implementing the new interface
  - Testing (unit, integration, system, and regression!)
  - Pushing the change
  - And now supporting the new version into the future
- To be good at maintenance, you need to be good at ALL the SDLC phases

But who does software maintenance?

- Depending on the company you are at, maintenance can be seen as the "lowest rung on the ladder"
- Developers often want to be doing "green field" development (new systems, new designs, new problems)
- Developers that have the most clout can work their way on those teams…
- … so maintenance is often left to less experienced developers…

Don't get in that mindset!

But who does software maintenance?

- Cultural problem in software development and maintenance
- Often the hardest job and seen as the least glamorous
- But companies need the best folks to be on maintenance!
  - Institutional / system knowledge is CRUCIAL
  - Being able to work with customers, developers, testers, etc. is important
  - Being a really good debugger and really good at comprehending code are needed skills
- Some companies do a “round-robin” rotation on maintenance

Software Maintenance Strategies

Adaptive Maintenance

Adapt to the changes in the environment (sw/hw)

Corrective Maintenance

Fix bugs, no matter where they appear (code, docs, etc.)

Perfective Maintenance

Consistently improve the system to stay competitive

Preventive Maintenance

Proactively address potential weaknesses before they become problematic

Software Maintenance Strategies Quiz

What kind of maintenance strategy do the following scenarios fall in?

- Your professor tells you to use Postgres instead of SQLite in your class project.
- Your customer tells you to make system work for color blind users.
- Your application uses a library that does not support the old Nvidia GPU architecture your cluster uses.

Software Maintenance Strategies Quiz

What kind of maintenance strategy do the following scenarios fall in?

- Your professor tells you to use Postgres instead of SQLite in your class project. – **corrective!**
- Your customer tells you to make system work for color blind users. – **corrective!**
- Your application uses a library that does not support the old Nvidia GPU architecture your cluster uses. – **adaptive!**

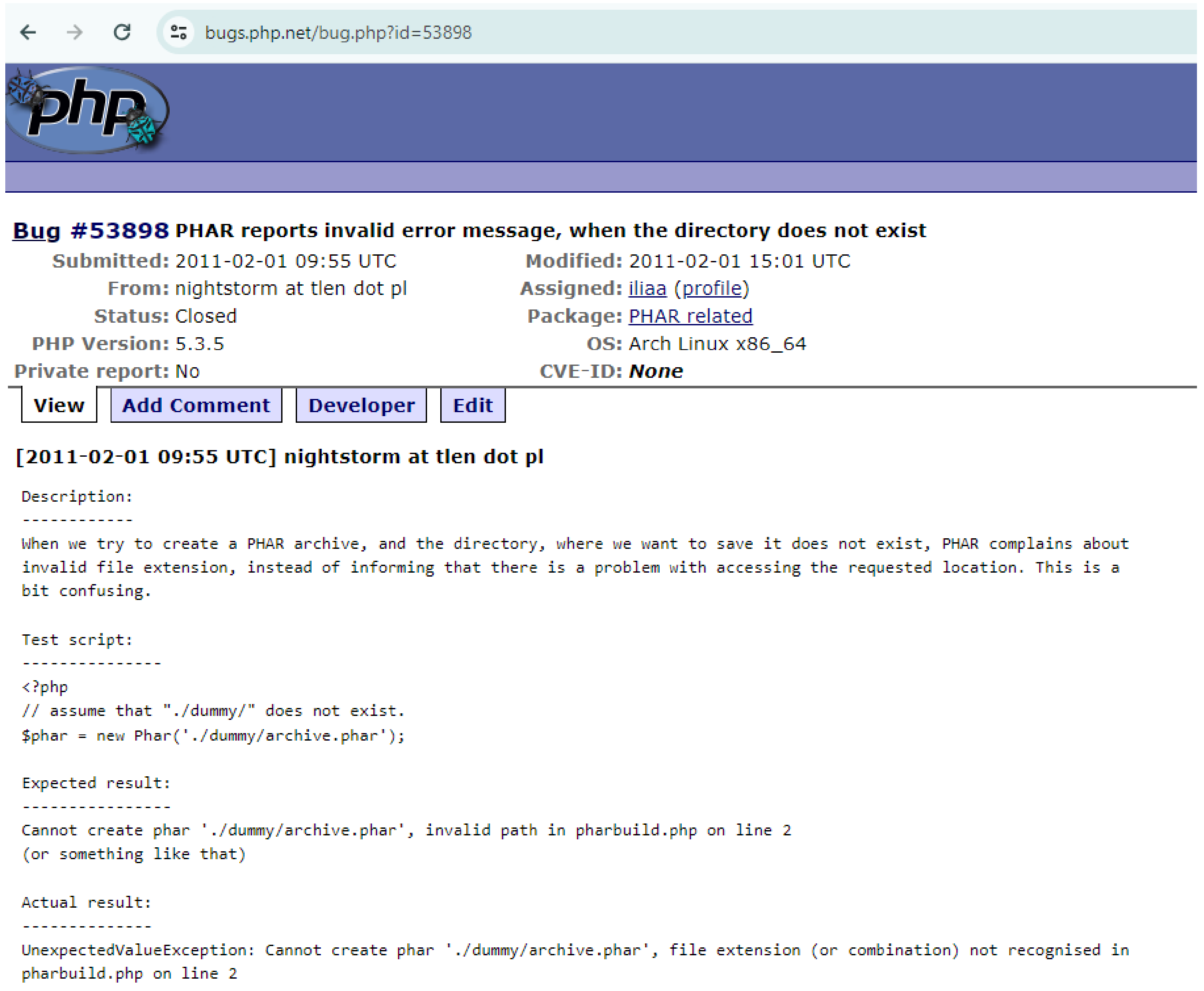
Software Maintenance Versioning

Version Numbers: X.Y.Z bxxxx

- X: major version (e.g., architectural change)
- Y: minor version (perfective maintenance)
- Z: patch/bug-fix version (no major change other than s/w doesn’t fail now in some cases)
- bxxxx: build version (optional, similar to commit-id to keep track of the branch and commit used for the release)

Where does maintenance start?

- The bug report!
- Process:
  - Bug/defect is reported by stakeholder
  - Defect is investigated by developer
  - A patch is created
  - Testing (including regression testing) occurs
  - Change request is then processed (varies widely with organizations)



Bug #53898 **PHAR reports invalid error message, when the directory does not exist**

Submitted:	2011-02-01 09:55 UTC	**Modified:**	2011-02-01 15:01 UTC
From:	nightstorm at tlen dot pl	**Assigned:**	iliaa (profile)
Status:	Closed	**Package:**	PHAR related
PHP Version:	5.3.5	**OS:**	Arch Linux x86_64
Private report:	No	**CVE-ID:**	***None***

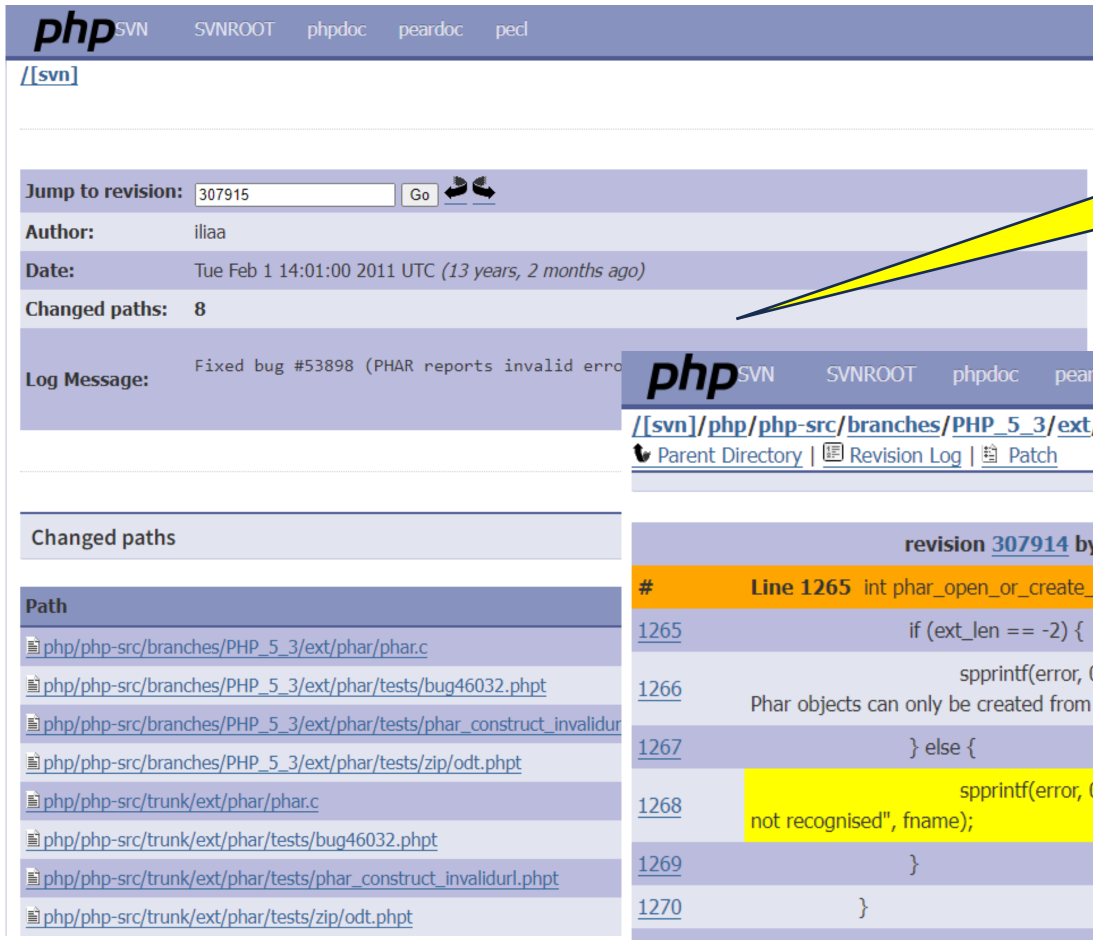
[2011-02-01 09:55 UTC] nightstorm at tlen dot pl

```
Description:
------------
When we try to create a PHAR archive, and the directory, where we want to save it does not exist, PHAR complains about
invalid file extension, instead of informing that there is a problem with accessing the requested location. This is a
bit confusing.

Test script:
---------------
<?php
// assume that "./dummy/" does not exist.
$phar = new Phar('./dummy/archive.phar');

Expected result:
----------------
Cannot create phar './dummy/archive.phar', invalid path in pharbuild.php on line 2
(or something like that)

Actual result:
--------------
UnexpectedValueException: Cannot create phar './dummy/archive.phar', file extension (or combination) not recognised in
pharbuild.php on line 2
```

Where does maintenance start?

8 files changed to fix the bug

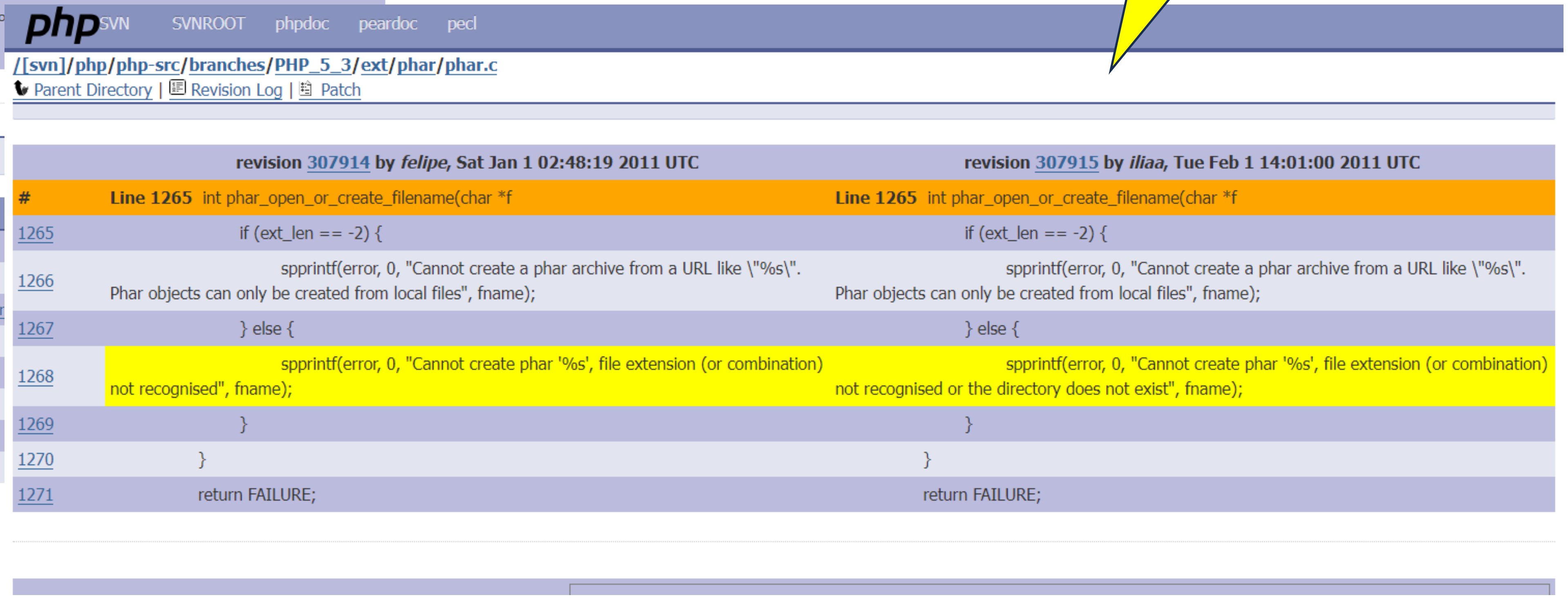
Patch for one of the files



Jump to revision:	307915 Go
Author:	iliaa
Date:	Tue Feb 1 14:01:00 2011 UTC *(13 years, 2 months ago)*
Changed paths:	8
Log Message:	`Fixed bug #53898 (PHAR reports invalid erro`

Changed paths

Path
php/php-src/branches/PHP_5_3/ext/phar/phar.c
php/php-src/branches/PHP_5_3/ext/phar/tests/bug46032.phpt
php/php-src/branches/PHP_5_3/ext/phar/tests/phar_construct_invalidur
php/php-src/branches/PHP_5_3/ext/phar/tests/zip/odt.phpt
php/php-src/trunk/ext/phar/phar.c
php/php-src/trunk/ext/phar/tests/bug46032.phpt
php/php-src/trunk/ext/phar/tests/phar_construct_invalidurl.phpt
php/php-src/trunk/ext/phar/tests/zip/odt.phpt


php SVN SVNROOT phpdoc peardoc pecl
/[svn]/php/php-src/branches/PHP_5_3/ext/phar/phar.c
Parent Directory | Revision Log | Patch

#	revision 307914 by *felipe*, Sat Jan 1 02:48:19 2011 UTC	revision 307915 by *iliaa*, Tue Feb 1 14:01:00 2011 UTC
	Line 1265 int phar_open_or_create_filename(char *f	**Line 1265** int phar_open_or_create_filename(char *f
1265	`if (ext_len == -2) {`	`if (ext_len == -2) {`
1266	`spprintf(error, 0, "Cannot create a phar archive from a URL like \"%s\". Phar objects can only be created from local files", fname);`	`spprintf(error, 0, "Cannot create a phar archive from a URL like \"%s\". Phar objects can only be created from local files", fname);`
1267	`} else {`	`} else {`
1268	`spprintf(error, 0, "Cannot create phar '%s', file extension (or combination) not recognised", fname);`	`spprintf(error, 0, "Cannot create phar '%s', file extension (or combination) not recognised or the directory does not exist", fname);`
1269	`}`	`}`
1270	`}`	`}`
1271	`return FAILURE;`	`return FAILURE;`

Software Maintenance Quiz

What are some of the things that can be done during the SDLC that can make maintenance tasks easier?

Software Maintenance Quiz

What are some of the things that can be done during the SDLC that can make maintenance tasks easier?

- Documentation and/or self-documenting code
- Following code conventions (language, system, and company)
- Use of design patterns
- Don’t over-design … but don’t under-design either
  - **System:** Fruit Inventory System for the Fruit Commission
  - **Over-design:** “I will design a system in which ‘fruit’ is an abstract class that can be expanded to all new, undiscovered kinds of fruit!”
  - **Under-design:** “There are seven fruits that we grow. The database will thus have seven columns for the seven types of fruit.”

Modern Software Maintenance

- Invention of issue-tracking systems to ease maintenance (Bugzilla, Jira, Redmine, Mantis, etc.)
- The concept of bug reports and change requests have evolved into the pull request (ref: GitHub)
- The same process from before is followed
- In fact, some open-source projects have strict requirements on submitted pull request (e.g., must include new test cases, show what new features are, etc.)

Risk of poor software maintenance & evolution: Ariane 5 Disaster

What did we learn & why should you take CS-563?

- Quick overview of software maintenance
- Software maintenance **is ALL of SDLC wrapped into its own time-frame** post the first release
- Software is an evolving, growing, and changing thing and we should have the skills and ability to support this (**ability to debug, develop new features, adapt with change, etc.)**
- At the end of the day, “green-field” development is just a start, **but it is not the heart of the software engineer**.
- It is more important that the released software continues to stay valued for stakeholders (users, managers, CEOs, CFOs, etc.).