



Lab Course / “Praktikum”: *Project Management and Software Development for Medical Applications*

Documentation, Tests, Design Patterns & Integration Strategy – WS2021/22

Vanessa Gonzales Duque

Munich, 8 November 2022



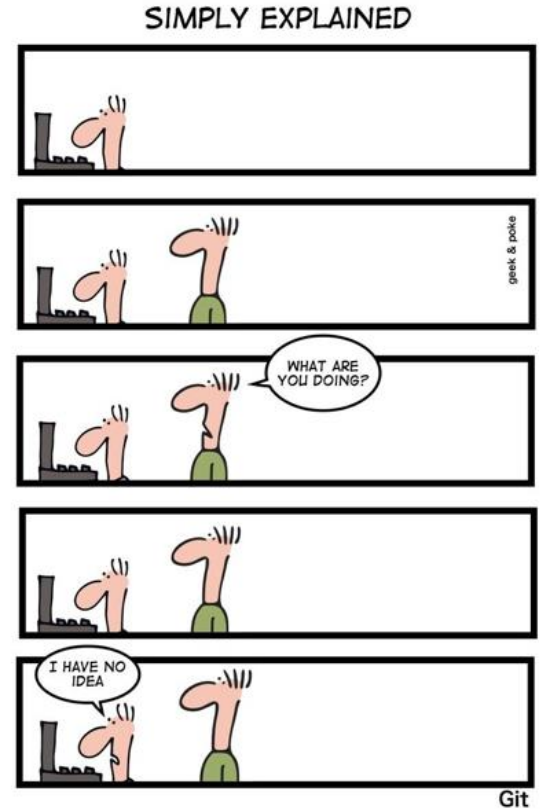
Technische Universität München



JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Disclaimer

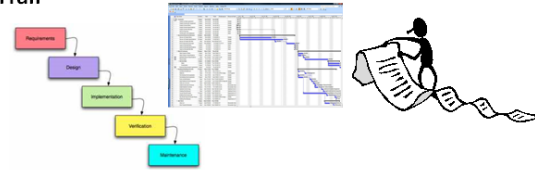
- This talk will not cover all aspects of SE!
- Familiarize with concepts and ideas
- Not every single detail matters



Software Engineering approaches

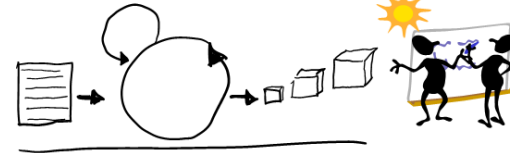
- Sometimes it is applied rigidly
- Many different contrasting ideas
- Do not get your attention drawn away from the problem at hand!

Waterfall



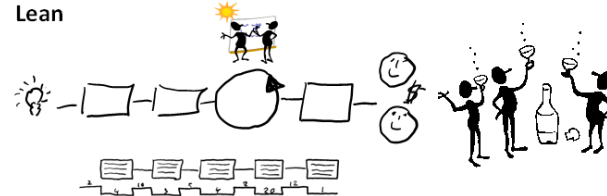
Schedule large work orders and align people by workflow

Agile



Schedule small work orders and align people by schedule

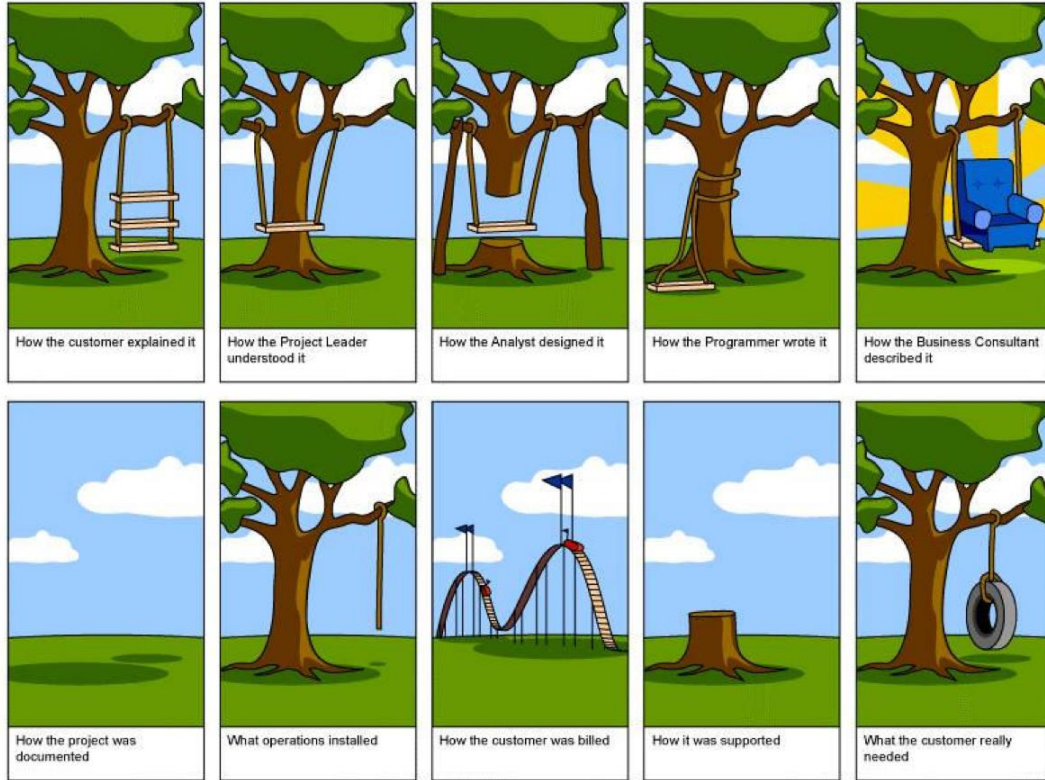
Lean



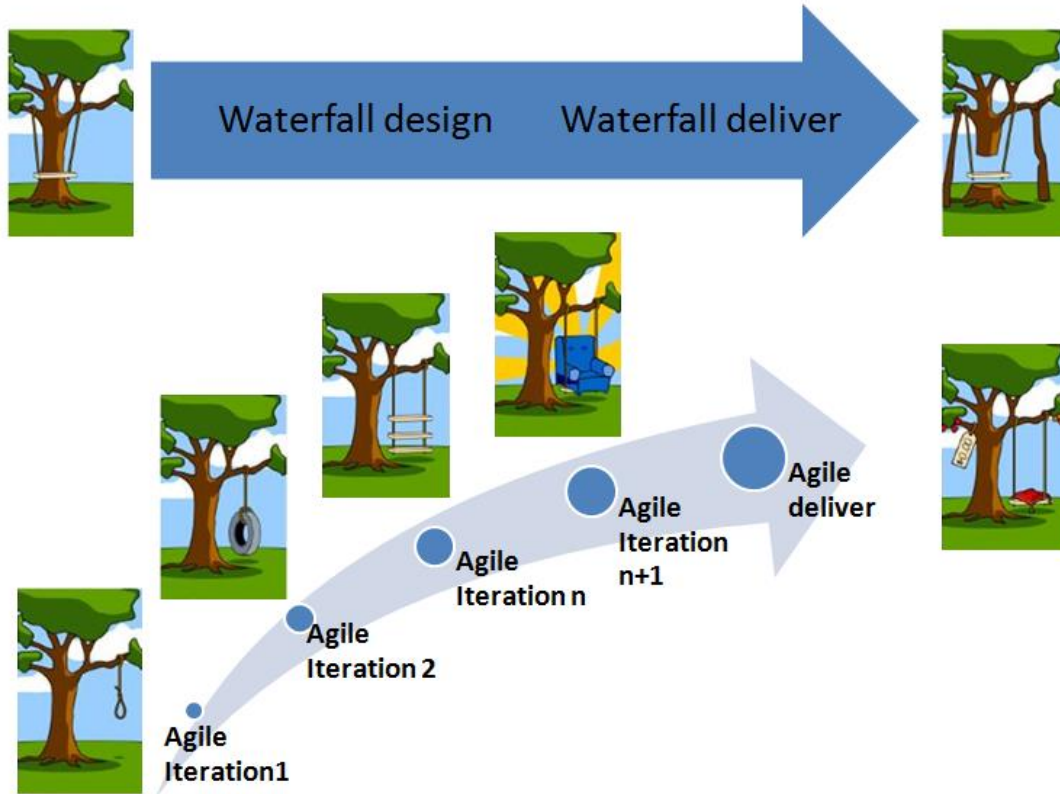
Schedule small work orders and align people by workflow



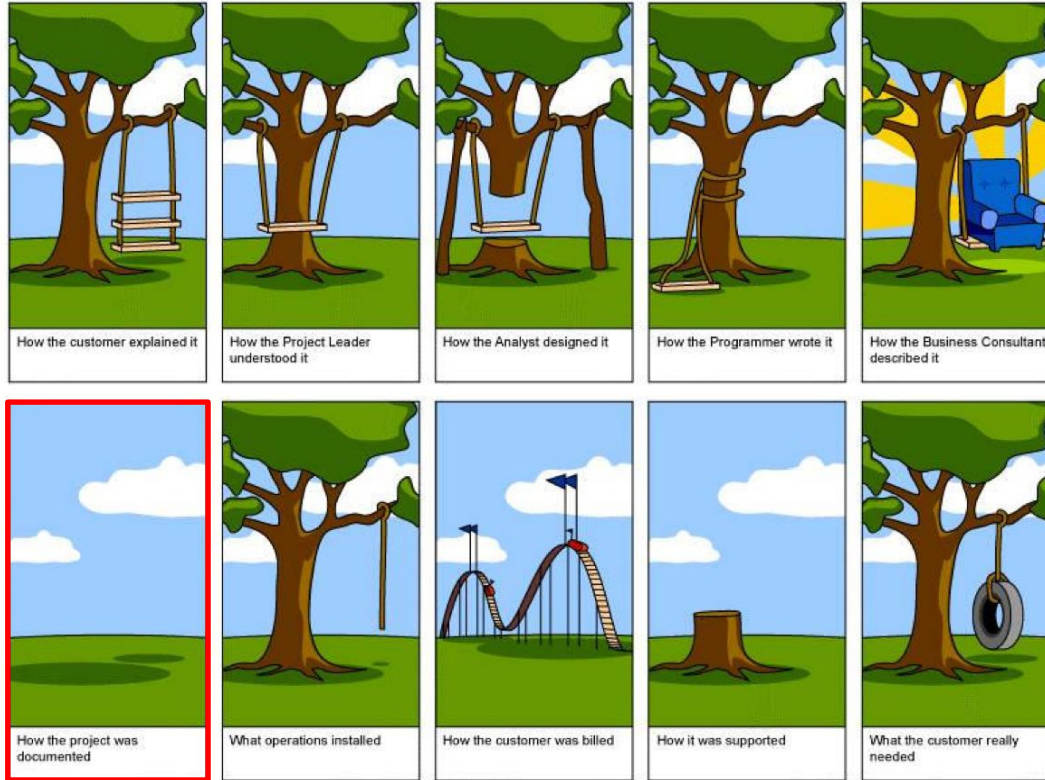
How Software Design and Engineering really works..



Keep the problem as small as possible!



How Software Design and Engineering really works..





Documentation

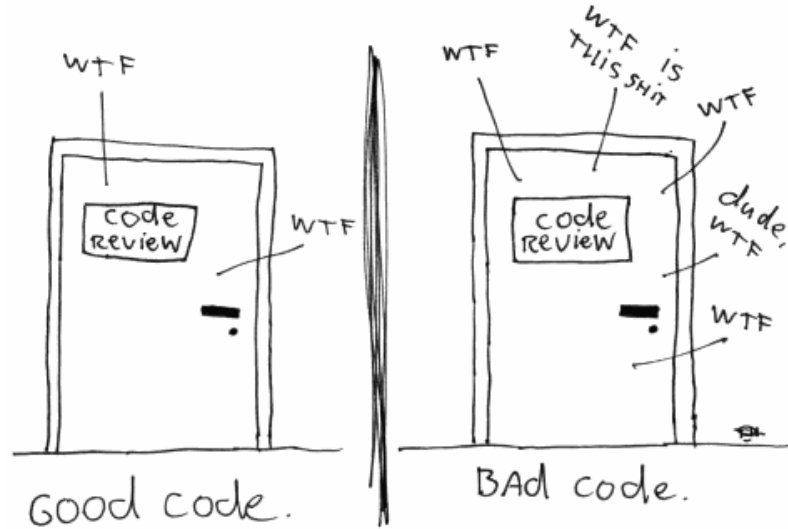


Documentation for developers

This includes:

- Your customers
- Your team
- Yourself!

The ONLY valid measurement
of code quality: WTFs/minute

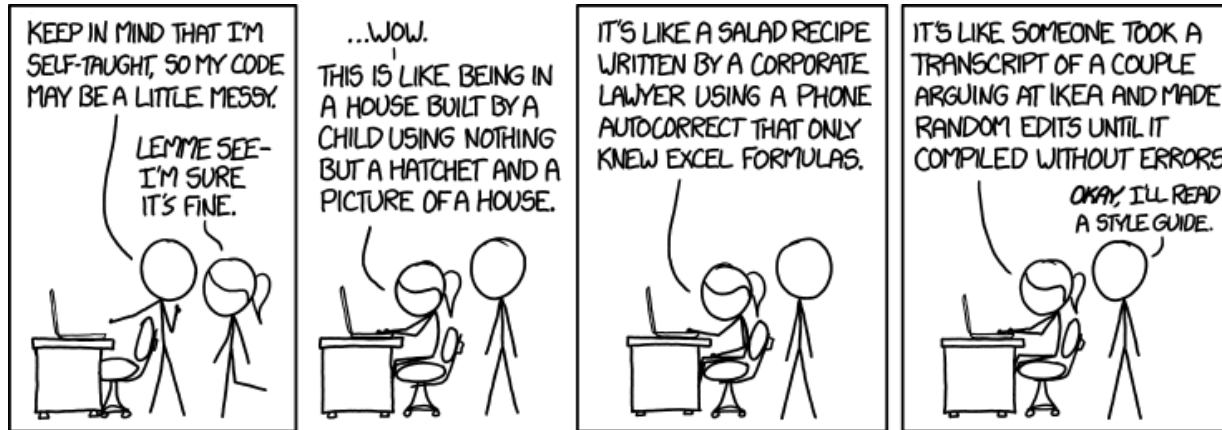


(c) 2008 Focus Shift



Documentation for developers – Code style

- Code is written once, but read many more times
- Don't be lazy:
 - Good variable names
 - Refactor code
 - Keep modular and generic



Documentation for developers – Comments

- No trivial comments
- Explain:
 - Assumptions
 - Corner cases
 - Non-trivial use of language features

BAD:

```
//Apply style.  
apply(style);
```

GOOD:

```
// Unlike the others, this image needs to be drawn in the user-requested style  
apply(style);
```



Documentation for developers – Doxygen

- Creates static docs from comments
- Close to source code, so USUALLY less out-of-date
- Useful only with non-trivial content

```
class Time {  
  
    public:  
  
        /**  
         * Constructor that sets the time to a given value.  
         *  
         * @param timemillis Number of milliseconds  
         *                 passed since Jan 1, 1970.  
         */  
        Time (int timemillis) {  
            // the code  
        }  
}
```



Documentation for developers – Doxygen

[Main Page](#) | [Class List](#) | [Class Members](#)

Time Class Reference

[List of all members.](#)

Public Member Functions

`Time (int timemillis)`

Static Public Member Functions

`Time now ()`

Detailed Description

The time class represents a moment of time.



Documentation for users

- Users as seen by developers:



- Usually the cause is bad documentation!
- You make a lot of assumptions that are clear in your head, but not to a new user



Design Patterns (and anti-Patterns)



Technische Universität München



JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

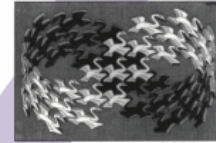
Design Patterns

- Reusable code structures
- Solve common problems
- Proven to work, common vocabulary
- Mostly created to work around rigid Object-Oriented type systems
- BUT: focus on the problem rather than where to stuff them in your program!

Design Patterns

Elements of Reusable
Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides



Foreword by Grady Booch



Some design Patterns

- Singleton: class with only one instance in whole program
- Abstract factory: allows to create an instance of several families of classes
- Observer: way of notifying change to a number of classes
- Decorator: add functionality to class without inheriting
- Facade: single class that represents an entire subsystem



SourceMaking, “Design Patterns,” [Online] Available: https://sourcemaking.com/design_patterns

Design anti-Patterns

- Too many classes
- Functions too long

```
img_filter = ImageFilter()  
img_filter.set_image(img)  
img_filter.set_radius(2.5)  
filtered_img = img_filter.get_output()
```



```
filtered_img = filter_img(img, radius=2.5)
```



Design anti-Patterns

- Too many classes
- Functions too long
- Mixed functionality
- Reinventing the wheel
- Premature optimization





Testing



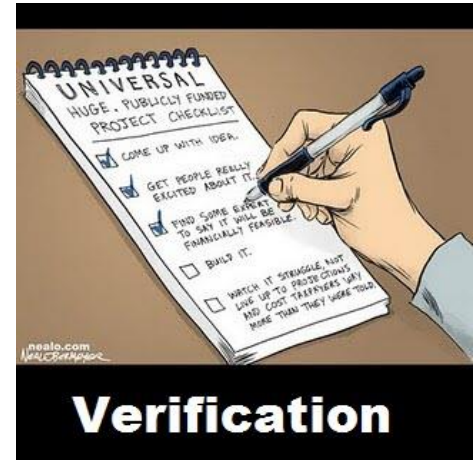
Technische Universität München



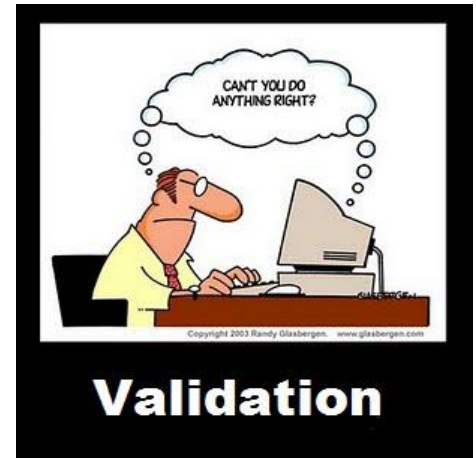
JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Testing – Definitions

- Verification and Validation (V&V)
 - **Verification**: The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of the phase [IEEE-STD-610]
 - **Validation**: The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements [IEEE-STD-610]



Verification



Validation



Testing – Definitions

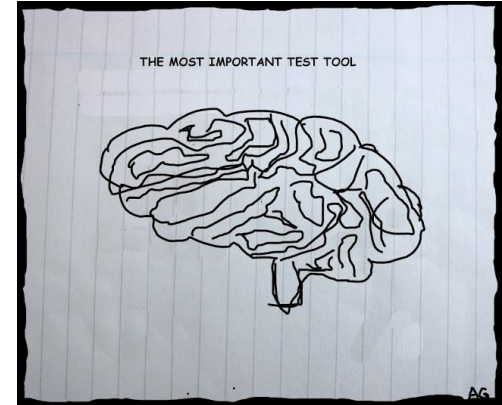
Criteria	Verification	Validation
Definition	The process of evaluating work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements for that phase.	The process of evaluating software during or at the end of the development process to determine whether it satisfies specified business requirements.
Objective	To ensure that the product is being built according to the requirements and design specifications. In other words, to ensure that work products meet their specified requirements.	To ensure that the product actually meets the user's needs, and that the specifications were correct in the first place. In other words, to demonstrate that the product fulfills its intended use when placed in its intended environment.
Question	Are we building the product <i>right</i> ?	Are we building the <i>right</i> product?
Evaluation Items	Plans, Requirement Specs, Design Specs, Code, Test Cases	The actual product/software.
Activities	<ul style="list-style-type: none"> •Reviews •Walkthroughs •Inspections 	<ul style="list-style-type: none"> •Testing



Test types

- Runtime Test: Sanity check for invalid program states during runtime
- Test Run: Developer runs the software and looks for obvious errors
- Systematic Test: Carefully chosen test data, comparison with expected results
- Regression Test: Extended and automated systematic test, run repeatedly (e.g. after every commit), test results are documented
- Performance Test: Testing performance of the software (runtime, memory usage, ...)

Testing may be a pain in the neck, but with the right combination of the above test types you get a good cost-return value



Test levels

- Unit Test: Checks a single piece of code (e.g. class) in isolation
- Integration Test: Verifies the interfaces between components
- System Test: Checks that the whole software meets the requirements
- Operational Acceptance Test: Put the software to test with real end users and in realistic conditions



Unit test

```
import unittest

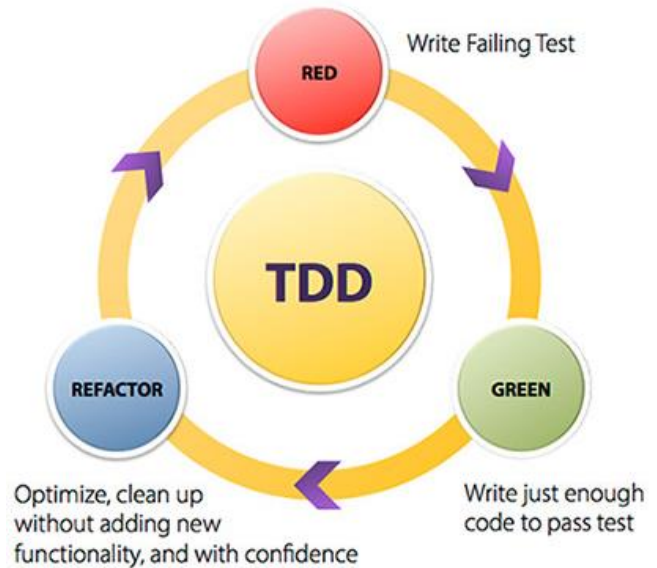
def fun(x):
    return x + 1

class MyTest(unittest.TestCase):
    def test(self):
        self.assertEqual(fun(3), 4)
```



Test Driven Development

- Write tests first, then develop until pass
- Pros:
 - Help focusing on objectives
 - Think about corner cases
 - More rewarding experience
 - More confident about later changes



Testable code

- Keep functions small

```
def add_to_cart(user, article):
    price = database.get_article(article)
    if user.age > 35 and article.category == 'food':
        price *= 0.90
    elif user.city == 'Munich' and article.category == 'electronics':
        price *= 0.85
    database.reduce_availability(article)
    user.add_to_cart(article, price)
```

```
def compute_price(user, price, article):
    if user.age > 35 and article.category == 'food':
        price *= 0.90
    elif user.city == 'Munich' and article.category == 'electronics':
        price *= 0.85
    return price
```

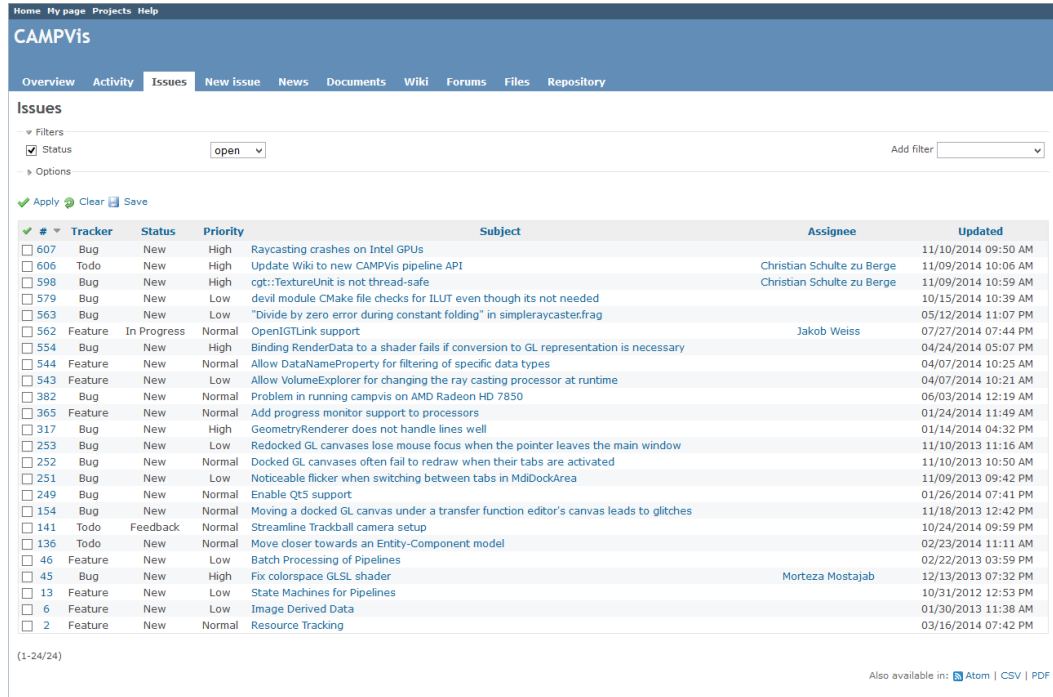
- Do not mix functionality

```
def add_to_cart(user, article):
    price = database.get_article(article)
    price = compute_price(user, price, article)
    database.reduce_availability(article)
    user.add_to_cart(article, price)
```



Bug tracker

- Help tracking defects present in software



The screenshot shows the 'Issues' page of the CAMPVis project. The page has a navigation bar with 'Home', 'My page', 'Projects', and 'Help'. Below the navigation bar, there are tabs for 'Overview', 'Activity', 'Issues', 'New Issue', 'News', 'Documents', 'Wiki', 'Forums', 'Files', and 'Repository'. The 'Issues' tab is selected. The page displays a list of issues with columns for Tracker, Status, Priority, Subject, Assignee, and Updated. The issues are sorted by status and priority. The first issue is a bug with status 'New' and priority 'High' titled 'Raycasting crashes on Intel GPUs'. The last issue is a feature with status 'New' and priority 'Normal' titled 'Resource Tracking'.

Tracker	Status	Priority	Subject	Assignee	Updated
<input type="checkbox"/>	Bug	New	High		11/10/2014 09:50 AM
<input type="checkbox"/>	Todo	New	High	Christian Schulte zu Berge	11/09/2014 10:06 AM
<input type="checkbox"/>	Bug	New	High	Christian Schulte zu Berge	11/09/2014 10:59 AM
<input type="checkbox"/>	Bug	New	Low		10/15/2014 10:39 AM
<input type="checkbox"/>	Bug	New	Low		05/12/2014 11:07 PM
<input type="checkbox"/>	Feature	In Progress	Normal	Jakob Weiss	07/27/2014 07:44 PM
<input type="checkbox"/>	Bug	New	High		04/24/2014 05:07 PM
<input type="checkbox"/>	Feature	New	Normal		04/07/2014 10:25 AM
<input type="checkbox"/>	Feature	New	Low		04/07/2014 10:21 AM
<input type="checkbox"/>	Bug	New	Normal		06/03/2014 12:19 AM
<input type="checkbox"/>	Feature	New	Normal		01/24/2014 11:49 AM
<input type="checkbox"/>	Bug	New	High		01/14/2014 04:32 PM
<input type="checkbox"/>	Bug	New	Low		11/10/2013 11:16 AM
<input type="checkbox"/>	Bug	New	Normal		11/10/2013 10:50 AM
<input type="checkbox"/>	Bug	New	Low		11/09/2013 09:42 PM
<input type="checkbox"/>	Bug	New	Normal		01/26/2014 07:41 PM
<input type="checkbox"/>	Bug	New	Normal		11/18/2013 12:42 PM
<input type="checkbox"/>	Todo	Feedback	Normal		10/24/2014 09:59 PM
<input type="checkbox"/>	Todo	New	Normal		02/23/2014 11:11 AM
<input type="checkbox"/>	Feature	New	Low		02/22/2013 03:59 PM
<input type="checkbox"/>	Bug	New	High	Morteza Mostajab	12/13/2013 07:32 PM
<input type="checkbox"/>	Feature	New	Low		10/31/2012 12:53 PM
<input type="checkbox"/>	Feature	New	Low		01/30/2013 11:38 AM
<input type="checkbox"/>	Feature	New	Normal		03/16/2014 07:42 PM



Bug tracker

The screenshot shows a bug tracker interface with several key elements and annotations:

- Issue Title:** "Broken URL"
- Assignee:** "No one is assigned" with a gear icon. A red arrow points to this icon with the annotation "assigned to".
- Milestone:** "No milestone" with a gear icon. A red arrow points to this icon with the annotation "milestones".
- Content:** "The url is broken"
- Buttons:** "Write" and "Preview" (with a red arrow pointing to the "Write" button), and "Submit new Issue" (with a red arrow pointing to it and the annotation "new issue").
- Milestone Modal:** A modal window titled "Milestone" with a search bar "Filter milestones", "Open" and "Closed" tabs, and a list of milestones:
 - Clear this milestone (highlighted in blue)
 - Version 1 (Due in 24 days)
 - Version 2 (Due in about 1 month)
- Add Labels:** A list of labels on the right side:
 - bug (red)
 - duplicate (grey)
 - enhancement (blue)
 - invalid (grey)
 - question (purple)
 - wontfix (grey)A red arrow points to the "wontfix" label with the annotation "labels".





Integration strategies



Technische Universität München



JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

The Big-Bang Integration Strategy

- *Unordered implementation of the components / all components implemented at the same time*
- Problems
 - Errors are very hard to locate: Which component is the cause?
 - Design errors (errors in interfaces) not distinguishable from implementation errors
- Always prefer incremental integration strategy



Top-Down Integration Strategy

- *Start with the components from the top-most layer (e.g. GUI). Incrementally add layers further down*
- Pros/Cons
 - Early prototype available (with limited functionality)
 - Design errors can be detected in an early state
 - Many stubs required → cumbersome
 - No functionality until a very late stage



Bottom-Up Integration Strategy

- *Start with the components from the bottom-most layer (e.g. entity classes). Incrementally add upper layers.*
- Pros/Cons
 - No stubs required
 - Functionality available in early stages
 - Nothing to show to customers until the very end
 - Errors may be expensive, because they may be found late and solving them might require cumbersome changes



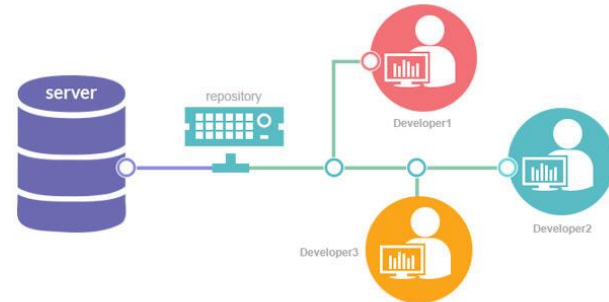


Version control

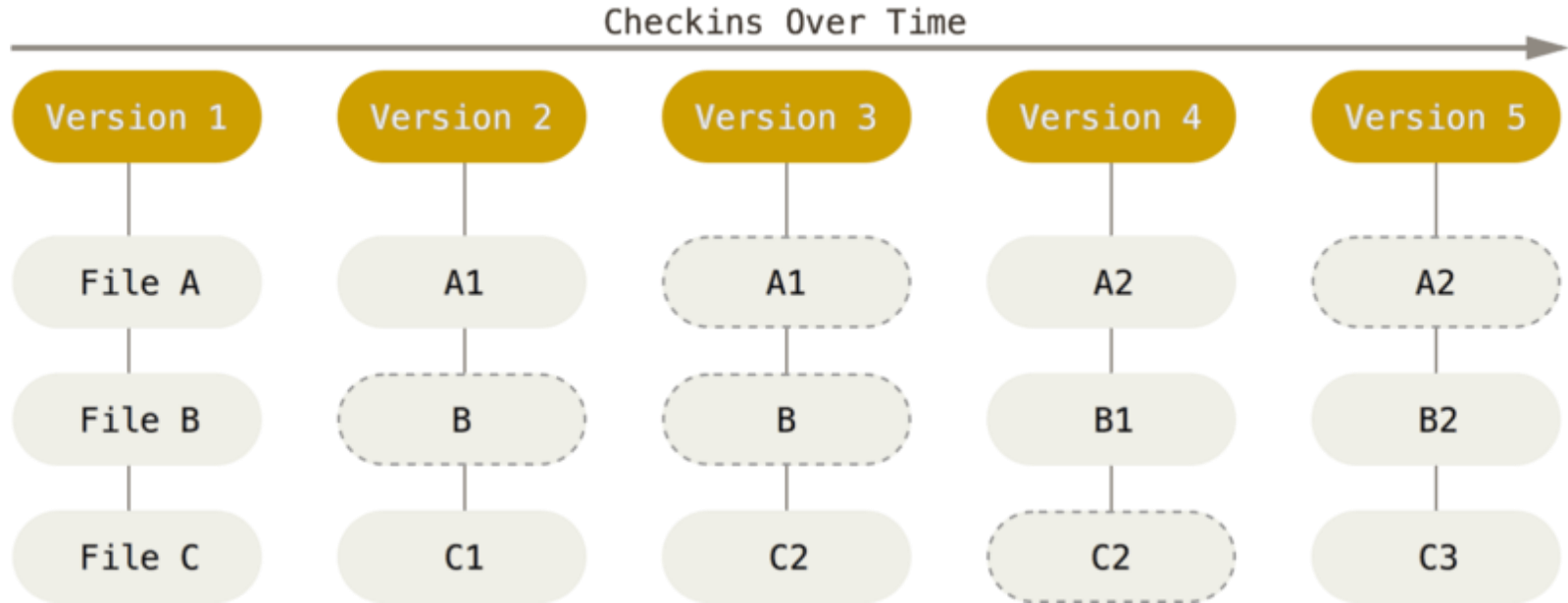


Version Control Systems

- Keep a history of changes to code
- Share code with others
- Integrate changes from others
- Manage concurrent versions



Version history



Changes history

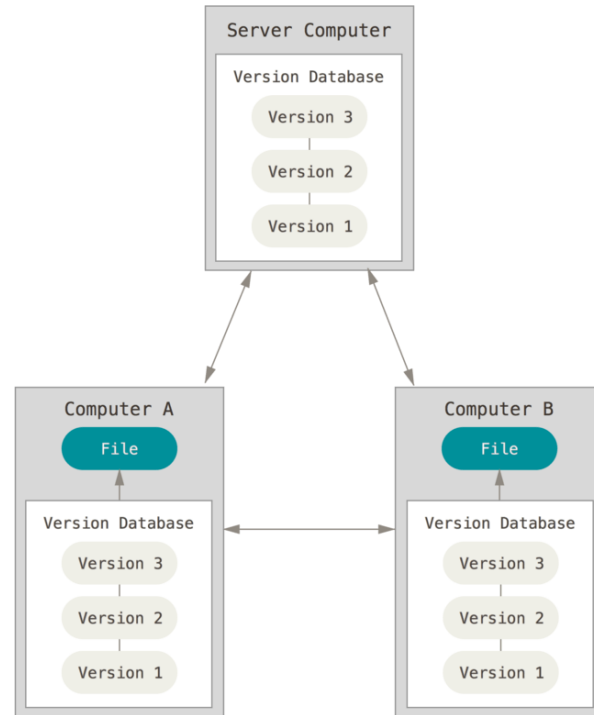
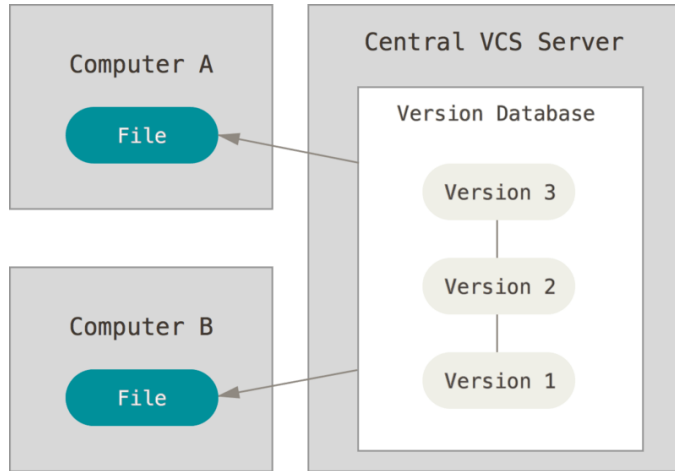
<p>2d4e9353 » streeter 2013-07-18 Add a missing quote so copy/...</p>	<pre>6 # "soupselect": "0.2.0"</pre>
<p>989e48f7 » nickhammond 2013-05-18 Specify underscore & undesc...</p>	<pre>7 # "underscore": "1.3.3" 8 # "underscore.string": "2.3.0"</pre>
<p>3406d66b » technicalpickles 2012-06-08 Update "w" help comments</p>	<pre>9 # 10 # Configuration: 11 # None 12 # 13 # Commands: 14 # hubot wiki me <query> - Searches for <query> on Wikipedia. 15 # 16 # Author: 17 # h3h</pre>
<p>97d63d4a » h3h 2011-11-09 Add a Wikipedia script for p...</p>	<pre>18 19 _ = require("underscore") 20 _s = require("underscore.string") 21 Select = require("soupselect").select 22 HTMLParser = require "htmlparser" 23 24 module.exports = (robot) -> 25 robot.respond /(wiki)(me)? (.*)/i, (msg) -></pre>
<p>374b8bfe » nickhammond 2013-05-18 change @http to @robot.http ...</p>	<pre>26 wikiMe robot, msg.match[3], (text, url) -></pre>
<p>97d63d4a » h3h 2011-11-09 Add a Wikipedia script for p...</p>	<pre>27 msg.send text</pre>



Branches

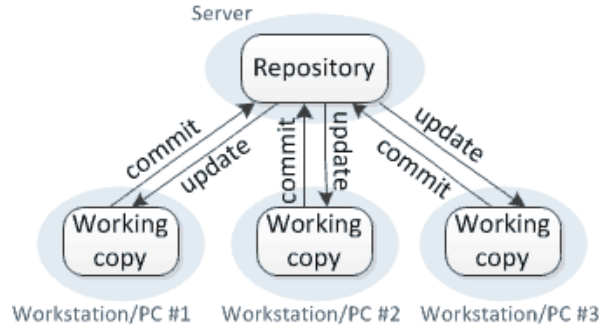


Centralized vs Distributed Version Control Systems

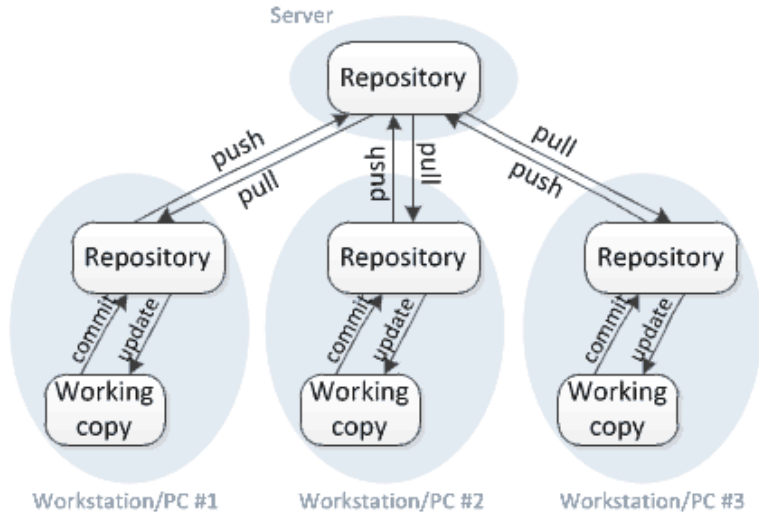


Centralized vs Distributed Version Control Systems

Centralized version control



Distributed version control



Software Configuration Management Guide, "Centralized vs Distributed Version Control Systems,"
[Online] Available: <https://scmquest.com/centralized-vs-distributed-version-control-systems/>



Continuous Integration

- Compile automatically on every change uploaded to VCS

The screenshot shows the Jenkins web interface for a project named 'Projekt campvis-gtest'. The page title is 'Projekt campvis-gtest' and the subtitle is 'Building CAMPVis on Linux/GCC'. The interface includes a navigation sidebar on the left with options like 'Zurück', 'Status', 'Änderungen', 'Arbeitsbereich', 'Jetzt bauen', 'Projekt Löschen', 'Konfigurieren', 'Redmine - tumvis', 'Embeddable Build Status', and 'GNU C Compiler Warnungen'. The main content area is divided into several sections: 'Arbeitsbereich' (Arbeitsbereich), 'Letzte Änderungen' (Letzte Änderungen), 'Letztes Testergebnis' (Kein Test fehlgeschlagen.), and 'Permalinks' (a list of build links). On the right side, there are two charts: 'GNU C Compiler Warnungen Trend' (a yellow area chart showing a peak at build #28) and 'Trend der Testergebnisse' (a blue area chart showing test results over time). The top navigation bar includes the Jenkins logo, a search bar, and user information for 'Christian Schulte zu Berge'.

GNU C Compiler Warnungen Trend

Build #	Count
#22	8
#24	8
#25	8
#26	14
#27	4
#28	4
#29	4
#30	4
#31	4
#32	4
#33	4
#34	4
#35	4
#36	4
#37	4
#38	4
#39	4
#40	4

Trend der Testergebnisse

Build #	Count
#29	0
#30	0
#31	0
#32	0
#33	0
#34	0
#35	0
#36	0
#37	0
#38	0
#39	0
#40	0
#41	0
#42	0
#43	0
#44	0
#45	0
#46	0
#47	0
#48	0
#49	0
#50	0
#51	0
#52	0
#53	0
#54	0
#55	0
#56	0
#57	0
#58	0
#59	0
#60	0
#61	0
#62	0
#63	0
#64	0
#65	0
#66	0
#67	0
#68	0
#69	0
#70	0
#71	0
#72	0
#73	0
#74	0
#75	0
#76	0
#77	0
#78	0
#79	0
#80	0
#81	0
#82	0
#83	0
#84	0
#85	0
#86	0
#87	0
#88	0
#89	0
#90	0
#91	0
#92	0
#93	0
#94	0
#95	0
#96	0
#97	0
#98	0
#99	0
#100	0
#101	0
#102	0
#103	0
#104	0
#105	0
#106	0
#107	0
#108	0
#109	0
#110	0
#111	0
#112	0
#113	0
#114	0
#115	0
#116	0
#117	0
#118	0
#119	0
#120	0
#121	0
#122	0
#123	0
#124	0
#125	0
#126	0
#127	0
#128	0
#129	0
#130	0
#131	0
#132	0
#133	0
#134	0
#135	0
#136	0
#137	0
#138	0
#139	0
#140	0





Thank you

Happy coding

Ardit Ramadani, M.Sc.
Research Assistent

Deutsches Herzzentrum München des Freistaates Bayern
Klinik an der Technischen Universität München
Lazarettstr. 36
80636 München

Technische Universität München
Fakultät für Informatik - I16
Chair of Computer Aided Medical Procedures and Augmented Reality
Boltzmannstr. 3
85748 Garching bei München

<https://www.in.tum.de/campar/members/ardit-ramadani/>
ardit.ramadani@tum.de
ramadani@dhm.mhn.de



Technische Universität München



JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING