Software technology

03 – Software Development Models and Methodologies

Dr. Attila Gludovátz

Online catalog – every week

- https://catalog.inf.elte.hu/
- Log in
- Username: yourUsername (@inf.elte.hu)
- Password: your email password
- Captcha: I generate a number for you…
- Lecture attendance is ***not*** optional! Max 3 misses and you are out

Developers are Different

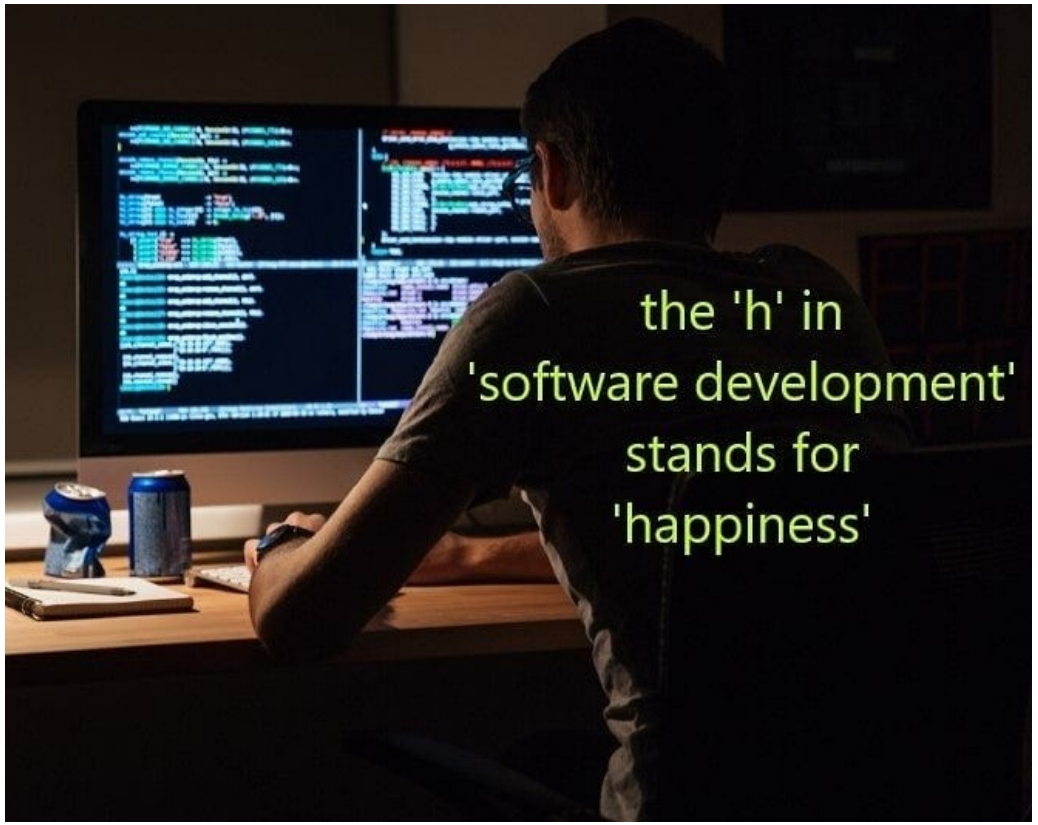
Tasks are Different

- Consider managing and estimating the following
  - Trendy, design-oriented web development company (making web shops)
  - Outsourcing company delivering information systems
  - Game developer company delivering on multiple platforms

Important thing is

- Know your
  - Team
  - Task
  - (Technology)

– in this order –



Different Types Of Programmers And Coders #1: The Innocent

- Dreamers
- Overestimate their skills

Different Types Of Programmers And Coders #2: The Regular Guy

- “Good enough”
- Right skills, perform well, but…

Different Types Of Programmers And Coders #3: The Hero

- The Anti-pattern
  - Coding immediately
  - Bugs come, we fix them
  - Sometimes we test
  - Sometimes we release
- There is also one more type of a hero or rather a wannabe hero worth-mentioning here – the *Code Cowboy*
- The code cowboy is a person who wants to help but does so in an irregular way
- He works quickly without much thinking
- If it comes to a deadline, the cowboy will do everything to meet it even if it means cutting off non-essential parts of the project

Different Types Of Programmers And Coders #4: The Jester

- *You only live once so why should you care?*

- Jesters live their lives to the fullest and prove to be one of the trickiest types of programmers to manage

Different Types Of Programmers And Coders #5: The Caregiver / Martyr

- The caregiver can quickly become the **Martyr**

- It is a person who will sacrifice themselves for their work, a workaholic in the caregiver’s shoes, to put it mildly.

Different Types Of Programmers And Coders #6: Ninja / Explorer

- Ninjas are people who do their work with precision and speed
- They work alone, know what they have to do even before you tell them
- Based on our observations, they’re one of the most valuable types of programmers out there

Different Types Of Programmers And Coders #7: The Rebel / Experimenter

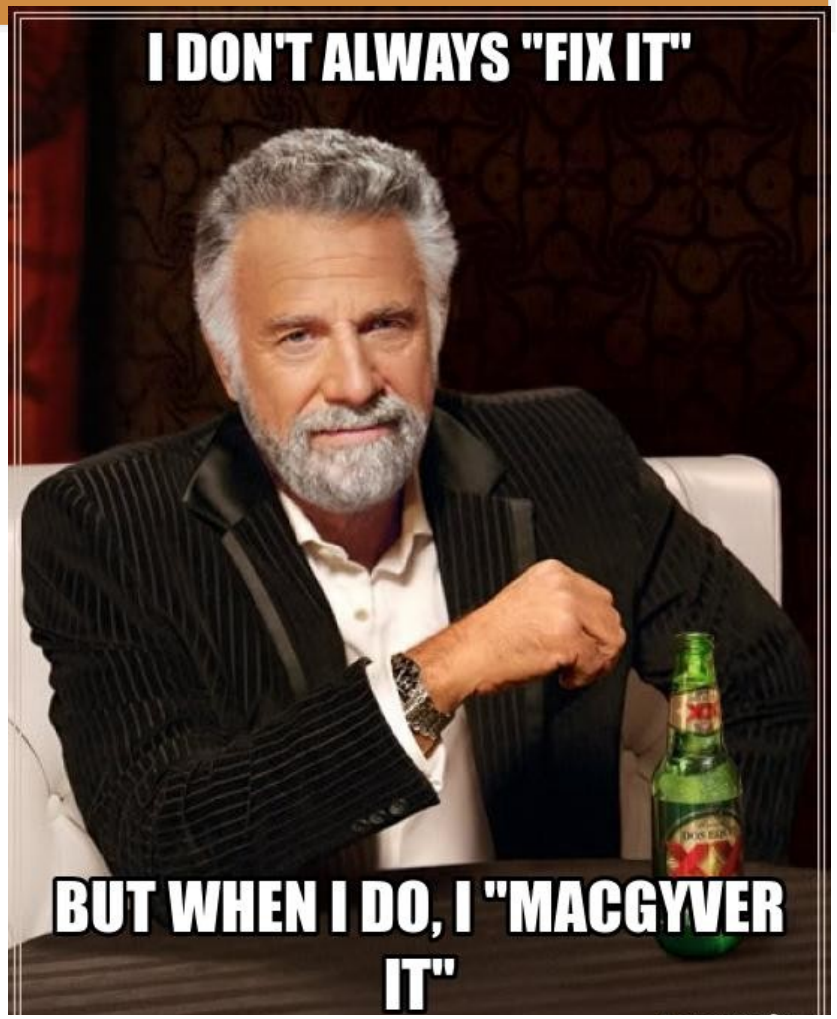
- Although ninjas can sound like they are rebels, they don’t have experiment
- One of the most creative types of programmers is the Rebel or Experimenter
- They are driven by the motto *“Rules are made to be broken”*
- Experimenters are constantly looking for new solutions, new frameworks, better languages, better code

Different Types Of Programmers And Coders #8: The Lover / Hardcore Geeks / Fanboys

- They love what they do
- The code is like their child
- They want to write the best code in the world, and they don’t like less-than-perfect solutions
- This can be a problem because a lot of work is based on finding *“good enough solutions”* rather than the **perfect ones** due to limited resources

Different Types Of Programmers And Coders #9: The Creator

- Every programmer must be a creator
- Among creators, there is one particular type who can cause a lot of problems when they are gone
- This person can fix anything in no time but in a way only they can understand
- To them, it really doesn’t matter what their work looks like as long as it is working



Different Types Of Programmers And Coders #10: The Sage

- The experienced programmer may appear outdated, but their knowledge and experience can be shared with others

- The representatives of this archetype can seem slow, but they know what they are doing and by working steadily way they win the race with great results

Different Types Of Programmers And Coders #11: The Magician

- Coding is like magic – you write some symbols and boom! There is a new thing
- Some programmers are like magicians – you don’t need to know the details or the technical aspects of their job, but you can still trust them
- They are making your vision a real thing

Different Types Of Programmers And Coders #12: The Ruler

- There are different types of Rulers
- One of them is **the VIP** – the kind of person who thinks he is the most important person in the project
  - They often look down on other team members and argue about everything that is against their vision
- A similar type is the **Perfectionist** – a person who won’t allow the project to go further unless the Perfectionist is contented with the results

Different Types Of Programmers And Coders #12: The Ruler

- The **Evangelist** is a person who insists on using a particular tool, language, solution, and attempts to revolutionize the workplace

- The **Clever Ambassador** is the face of the team
- The Ambassador has excellent communication skills and knows how to sell the work of the team and supervise

See also…

- **Hacker**, a subculture that relies on the creativity of individual programmers
- **Code monkey**, a pejorative term for programmers who are employed to write simple or repetitive code
- **Self-employment**, this is the state of working for oneself rather than an employer
- **Indie game developer**, have smaller budgets, usually sourcing from personal funds or via crowdfunding

Conclusion

- *Do any of these developer archetypes sound familiar?*
- *Do any of the types dominate your workforce?*
- The best idea is to welcome various types of programmers to your company, as non-homogenous teams are typically **more productive**

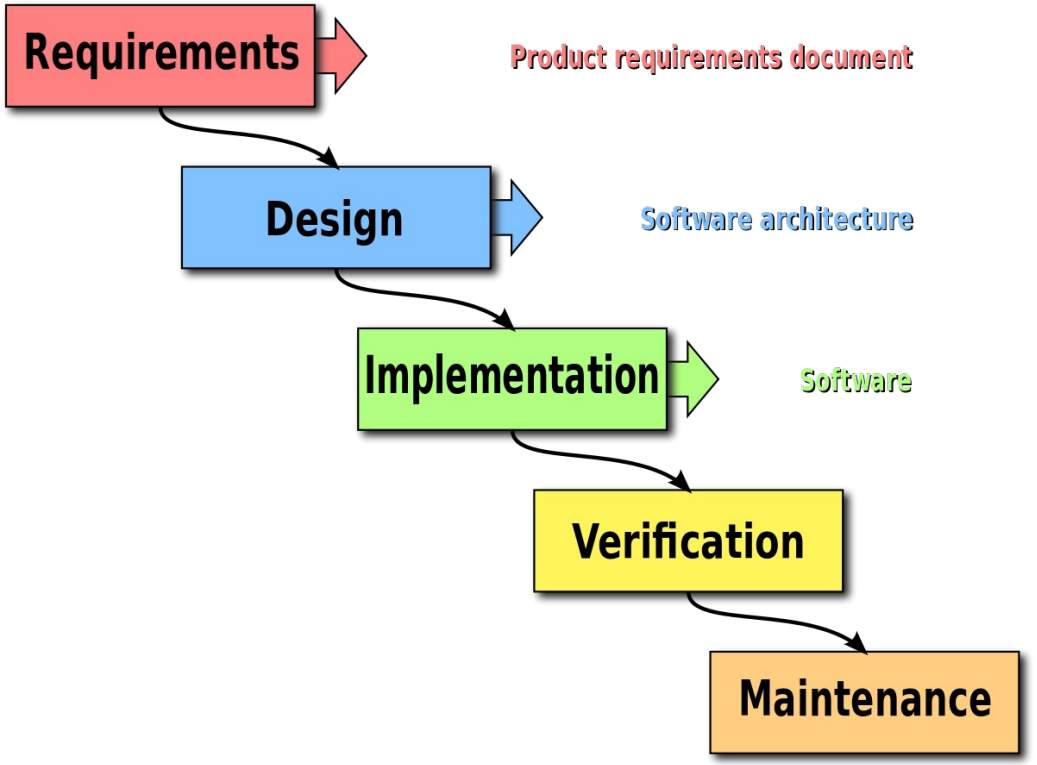
- Remember that each of these types of programmers come with their *advantages* and *disadvantages* which makes them more likely to perform under certain conditions

What kind of programmer are you?

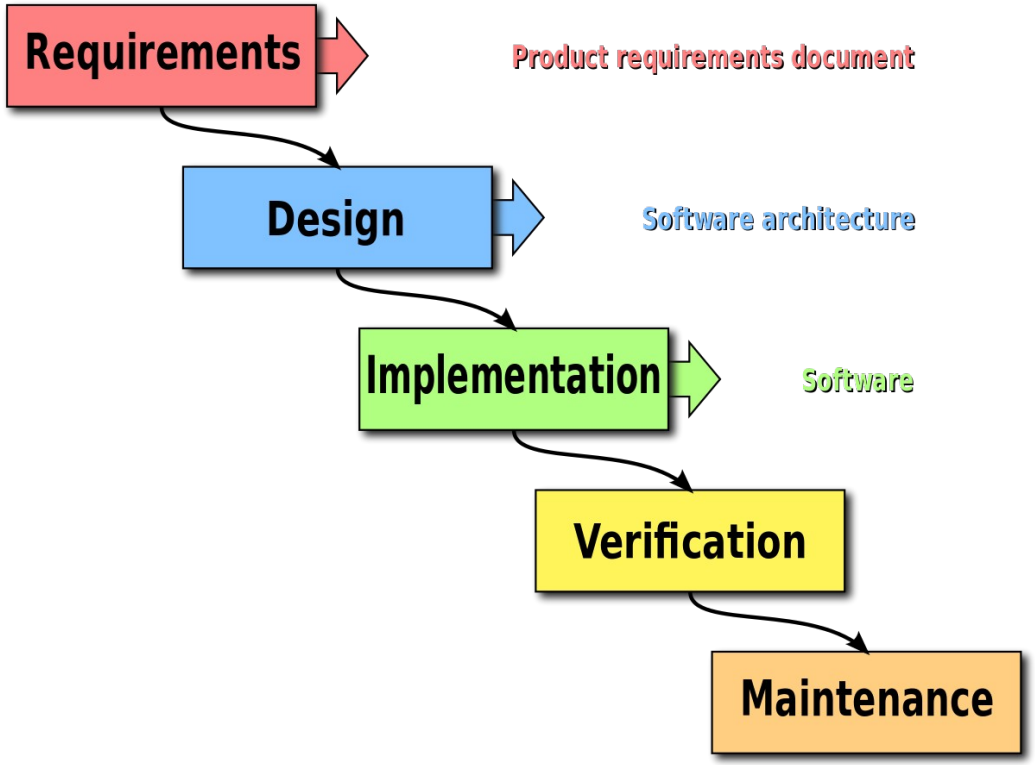
Software development models

Waterfall model

- Sequential phases, little overlap or feedback
- Schedules and target dates
- Implement all at once
- Tight control
- Document everything
- Big Design Up Front approach



The Phases of Waterfall model



The Phases of Waterfall model Subprocesses and deliverables

1. **Requirements:** project scope, stakeholder expectations, research (e.g., market), assemble team, kickoff (meeting)
2. **Design:** collect tasks (WBS diagram), create schedule (Gantt diagram)
3. **Implementation:** assign team tasks, monitor, track & trace, manage resources, report to stakeholders, test, deliver application
4. **Verification:** pay contracts, create template, close out paperwork, celebrate
5. **Maintenance:** This is an ongoing, post-launch phase that extends for as long as your contract dictates

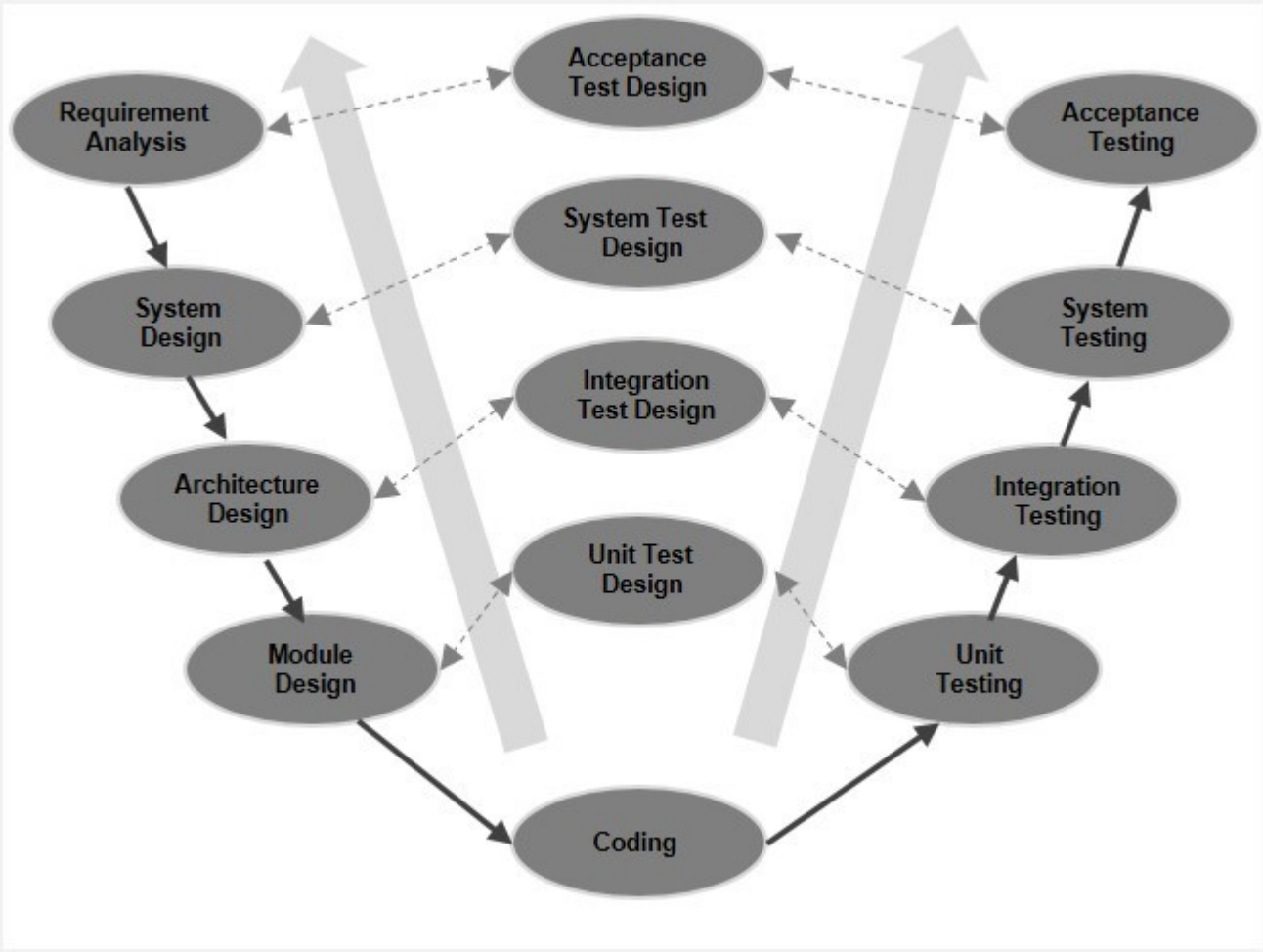
Waterfall model
Pros and cons

- Detailed analysis and planning reduces cost on later stages, may spot problems early
- Only 30% - 40% is implementation
- Structured approach leads to
  - Refined organization structure
  - Detailed documentation
  - Easy recovery (on team member change)
- Easy to monitor (everybody understands where we are)
- Works well if requirements, scope, technology is well understood

- If something is not understood well, like
  - Requirements
  - Scope
  - Technology
  - Goals, targets
- Too rigid, cannot handle problems with
  - Design
  - Major testing issues
  - Multiple things has to be redone then

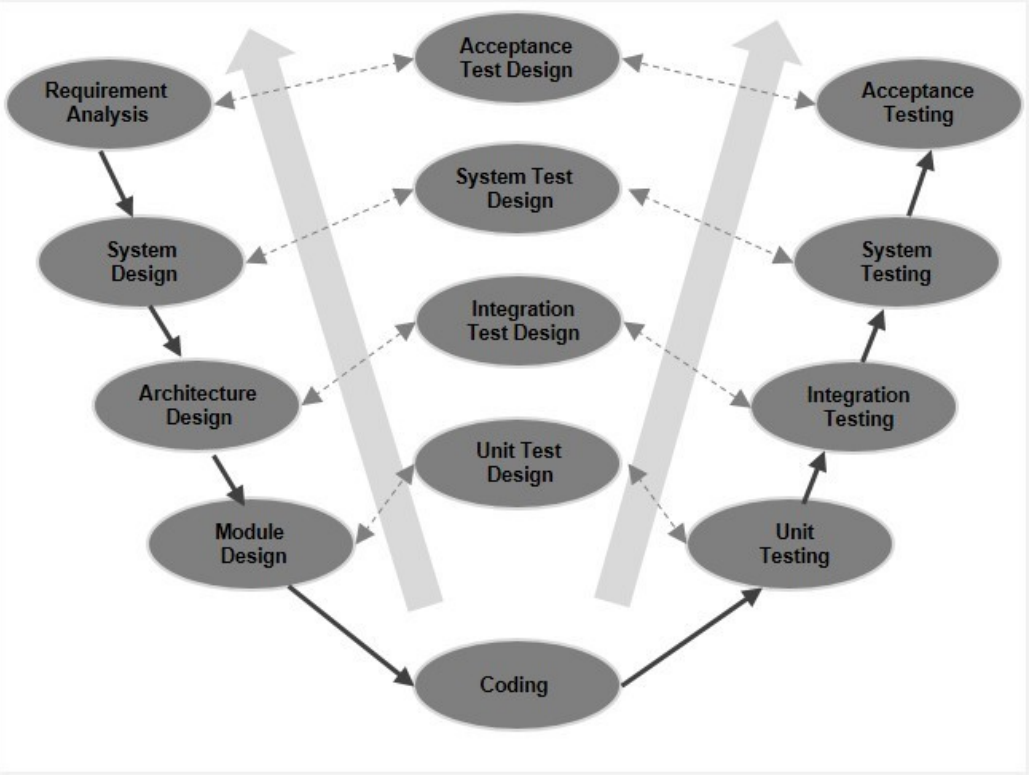
→ Modified Waterfall models

V-Model


Requirement Analysis
Acceptance Test Design
Acceptance Testing
System Test Design
System Design
System Testing
Integration Test Design
Architecture Design
Integration Testing
Unit Test Design
Module Design
Unit Testing
Coding

V-Model

- Only A Modified Waterfall
- Has different Levels of Abstraction
- Pairs development and testing phases (but do not unifies them)



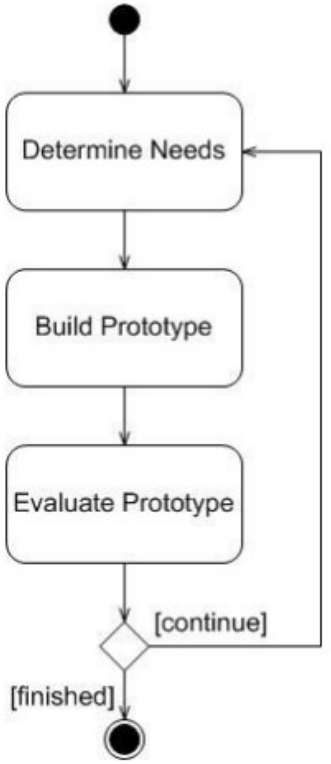
V-Model
Pros and cons

- The advantage of the V-Model method is that it is very easy to understand and apply
- The simplicity of this model also makes it easier to manage

- The disadvantage is that the model is not flexible to changes and just in case there is a requirement change, which is very common in today’s dynamic world, it becomes very expensive to make the change

Software Prototype Model

- Iterative approach
- Prototypes are
  - Horizontal (shallow, for User, GUI only)
  - Vertical (drill down in a function / feature)
- Prototyping can be
  - Throwaway
  - Evolutionary (prototype becomes product)
  - Merging (or Incremental)



What is Software Prototyping?

- Prototype is a working model of software with some limited functionality
- The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation
- Prototyping is used to allow the users evaluate developer proposals and try them out before implementation
- It also helps understand the requirements which are user specific and may not have been considered by the developer during product design

Software Prototyping - Application

- Software Prototyping is most useful in development of systems having high level of user interactions such as online systems
- Systems which need users to fill out forms or go through various screens before data is processed can use prototyping very effectively to give the exact look and feel even before the actual software is developed
- Software that involves too much of data processing and most of the functionality is internal with very little user interface does not usually benefit from prototyping

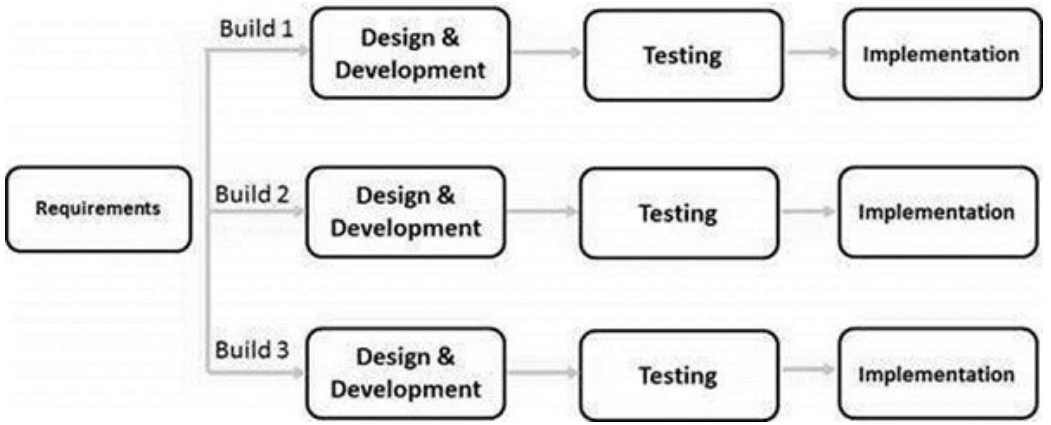
Software Prototyping
Pros and cons

- Fast
- Cheap
- Quick User Feedback
- User Involvement

- No planning → problems later or too much iterations
- Too much iterations → Too much prototypes → becomes Expensive and too long
- Developers dislike throwing away code
- Developers do not understand the real goals (they just have shallow GUI prototypes)

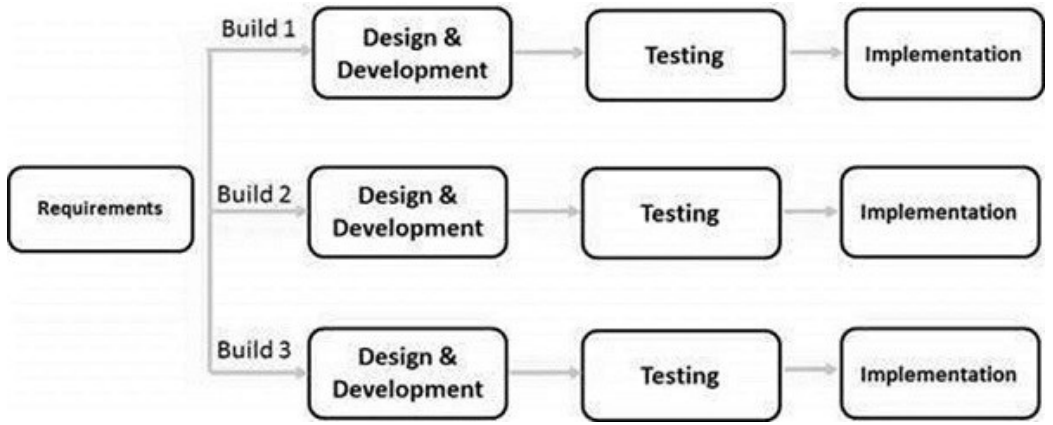
Iterative / Incremental Model

- In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed



Iterative / Incremental Model

- An iterative life cycle model does not attempt to start with a full specification of requirements
- Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements
- This process is then repeated, producing a new version of the software at the end of each iteration of the model



Iterative / Incremental Model Pros and ...

- Some working functionality can be developed quickly and early in the life cycle
- Results are obtained early and periodically
- Parallel development can be planned
- Progress can be measured
- Less costly to change the scope/requirements
- Testing and debugging during smaller iteration is easy
- Risks are identified and resolved during iteration; and each iteration is an easily managed milestone
- Risk analysis is better

- Easier to manage risk - High risk part is done first
- With every increment, operational product is delivered
- Issues, challenges and risks identified from each increment can be utilized/applied to the next increment
- It supports changing requirements
- Initial Operating time is less
- Better suited for large and mission-critical projects
- During the life cycle, software is produced early which facilitates customer evaluation and feedback

Iterative / Incremental Model
… and cons

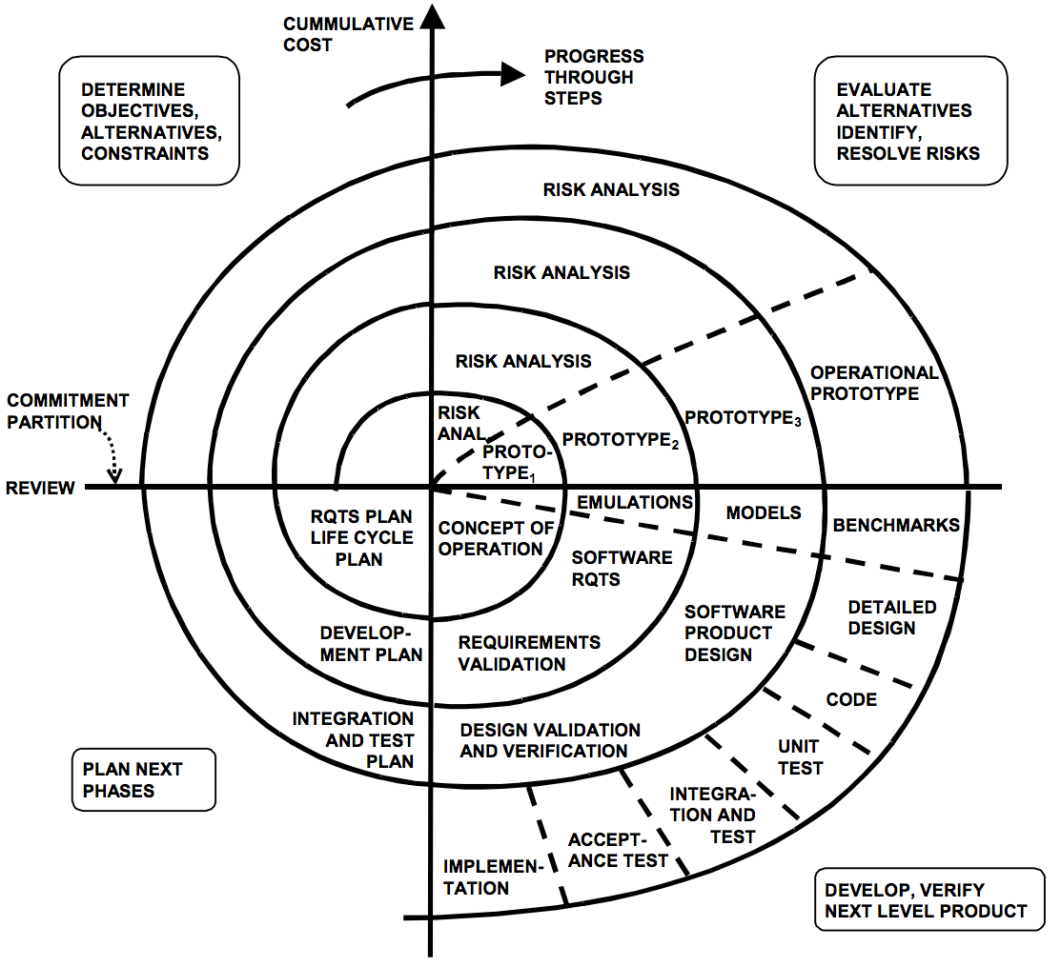
- More resources may be required
- Although cost of change is lesser, but it is not very suitable for changing requirements
- More management attention is required
- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle
- Not suitable for smaller projects

- Defining increments may require definition of the complete system
- Management complexity is more
- End of project may not be known which is a risk
- Highly skilled resources are required for risk analysis
- Projects progress is highly dependent upon the risk analysis phase

Spiral Model

= Waterfall + Prototyping + Iterative (Incremental)

- Stakeholder perspective!
- Adaptive in
  - Planning depth (more risk, more detail)
  - Effort (more risk, more effort)



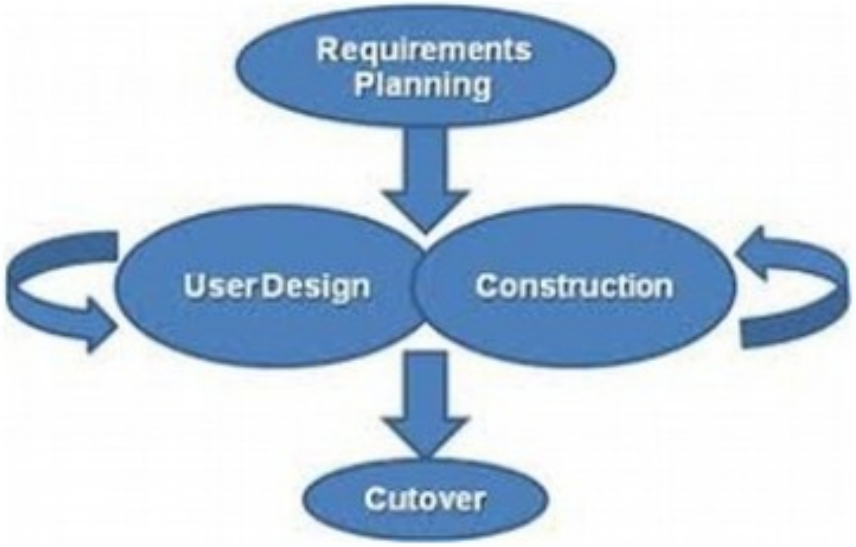
Spiral Model
Pros and cons

- Changing requirements can be accommodated
- Allows extensive use of prototypes
- Requirements can be captured more accurately
- Users see the system early
- Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management

- Management is more complex
- End of the project may not be known early
- Not suitable for small or low risk projects and could be expensive for small projects
- Process is complex
- Spiral may go on indefinitely
- Large number of intermediate stages requires excessive documentation

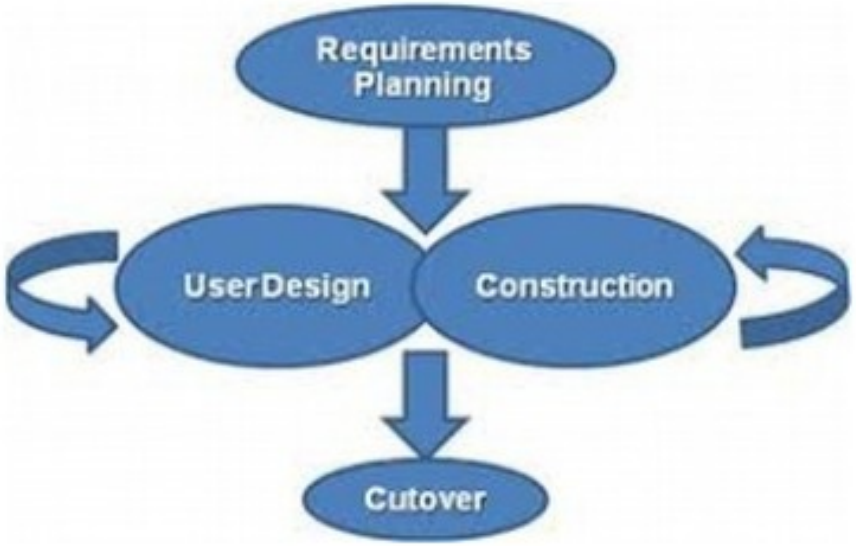
RAD Model

- Rapid Application Development
- Flexible process to gather knowledge
- Made for GUI and information systems
- The golden age of GUI builders



RAD Model

- Usually trendy methods are good for certain type of projects / products where no unique solutions are needed (no risk of bad design, bad planning… similar as our previous project)



RAD Model
Pros and cons

- Changing requirements can be accommodated
- Progress can be measured
- Iteration time can be short with use of powerful RAD tools
- Productivity with fewer people in a short time
- Reduced development time
- Increases reusability of components

- Dependency on technically strong team members for identifying business requirements
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers
- High dependency on Modelling skills.
- Inapplicable to cheaper projects as cost of Modelling and automated code generation is very high
- Management complexity is more

Big Bang / Chaos Model

- Almost anti-pattern
- Lightweight model
- Definition, implementation and integration on all levels respectively
- Issue is just an incomplete, not well-defined task
- All **issues** are executed according to **categorization**
  - **Value** (to the user)
  - **Urgency**
  - **Robustness** (completeness)

Big Bang / Chaos Model
Pros and cons

- This is a very simple model
- Little or no planning required
- Easy to manage
- Very few resources required
- Gives flexibility to developers
- It is a good learning aid for newcomers or students

- Very High risk and uncertainty
- Not a good model for complex and object-oriented projects
- Poor model for long and ongoing projects
- Can turn out to be very expensive if requirements are misunderstood

Software Development Tools

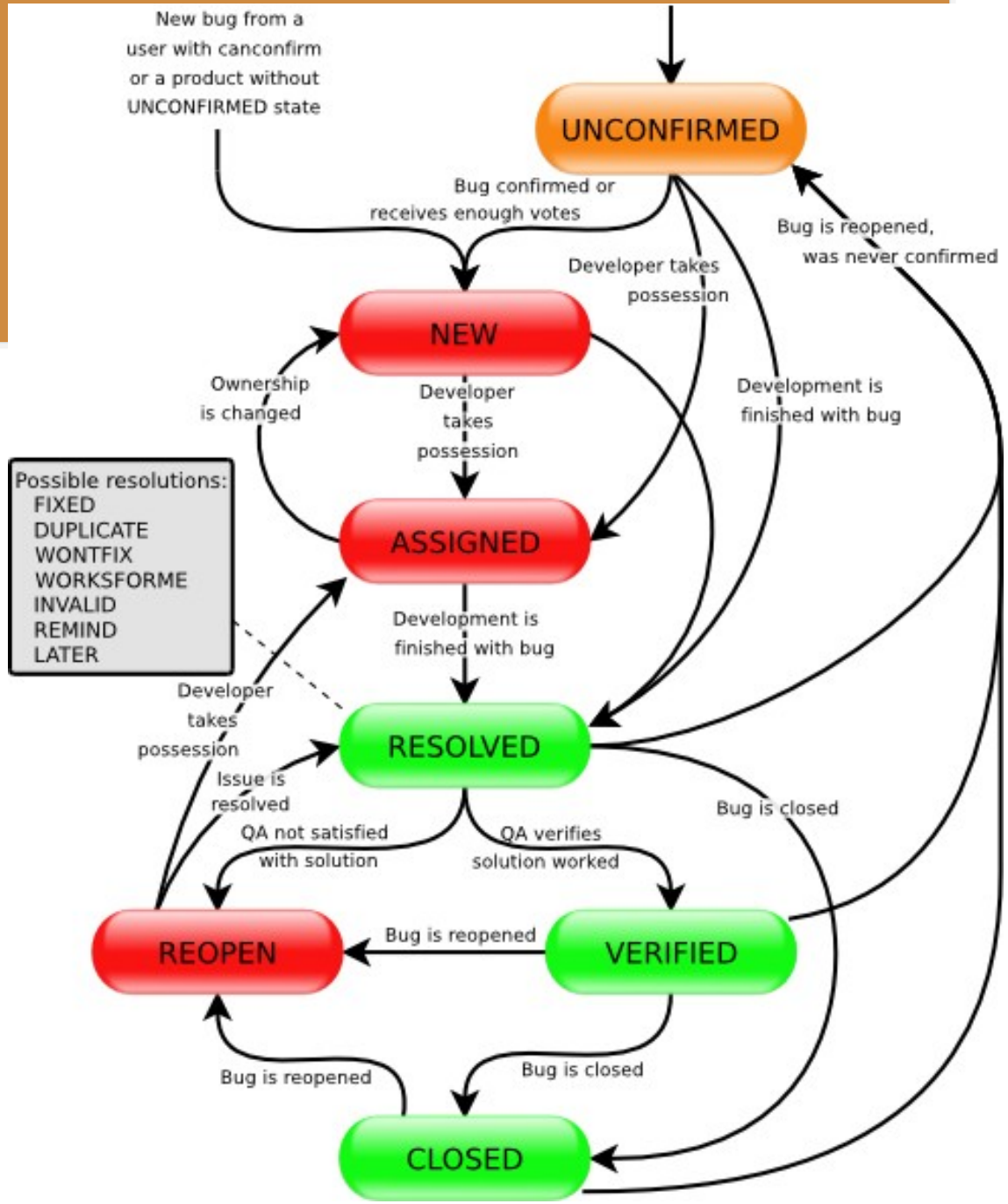
Time Management Matrix

Bug Life-cycle
Bugzilla

- Bugzilla is a robust, featureful and mature defect-tracking system, or bug-tracking system
- Defect-tracking systems allow teams of developers to keep track of outstanding bugs, problems, issues, enhancement and other change requests in their products effectively
- Simple defect-tracking capabilities are often built into integrated source code management environments such as Github or other web-based or locally-installed equivalents
- We find organizations turning to Bugzilla when they outgrow the capabilities of those systems - for example, because they want workflow management, or bug visibility control (security), or custom fields

Bug Life-cycle Bugzilla

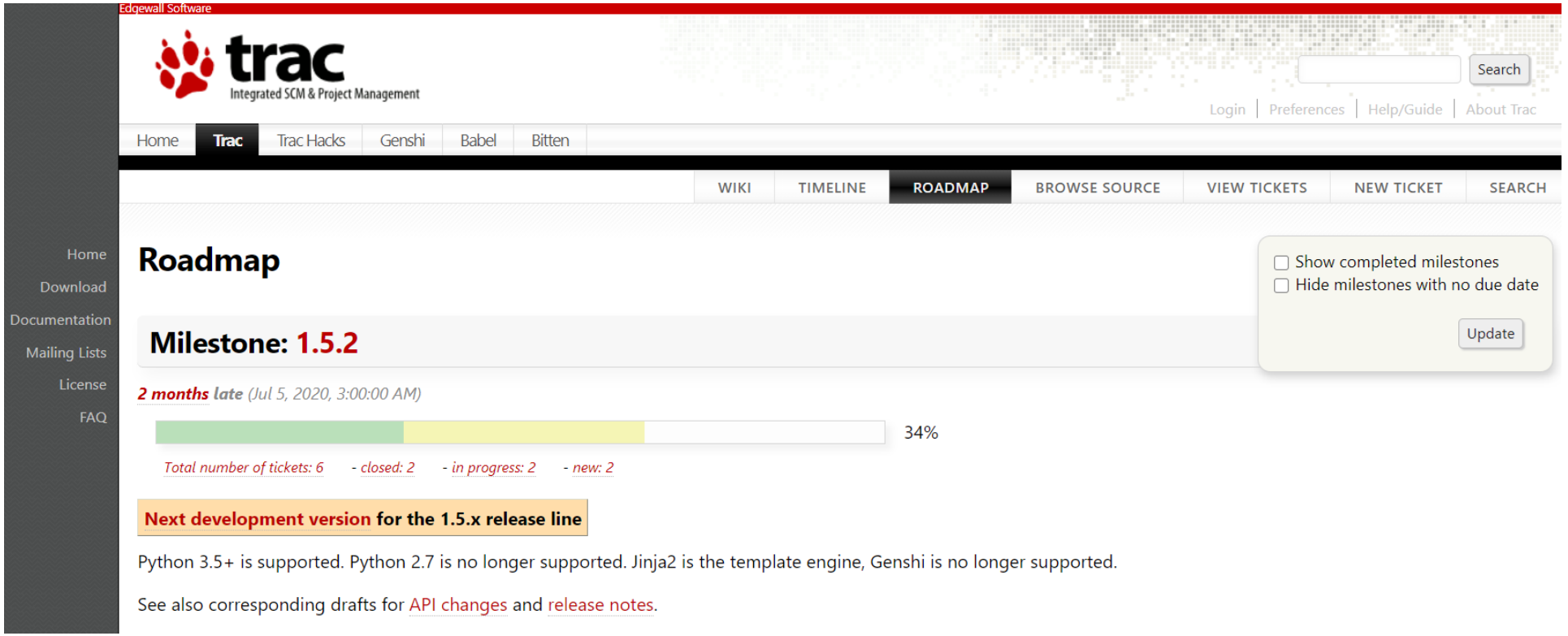
- The life cycle, also known as workflow of a bug is currently hardcoded into Bugzilla
- This figure contains a graphical representation of this life cycle



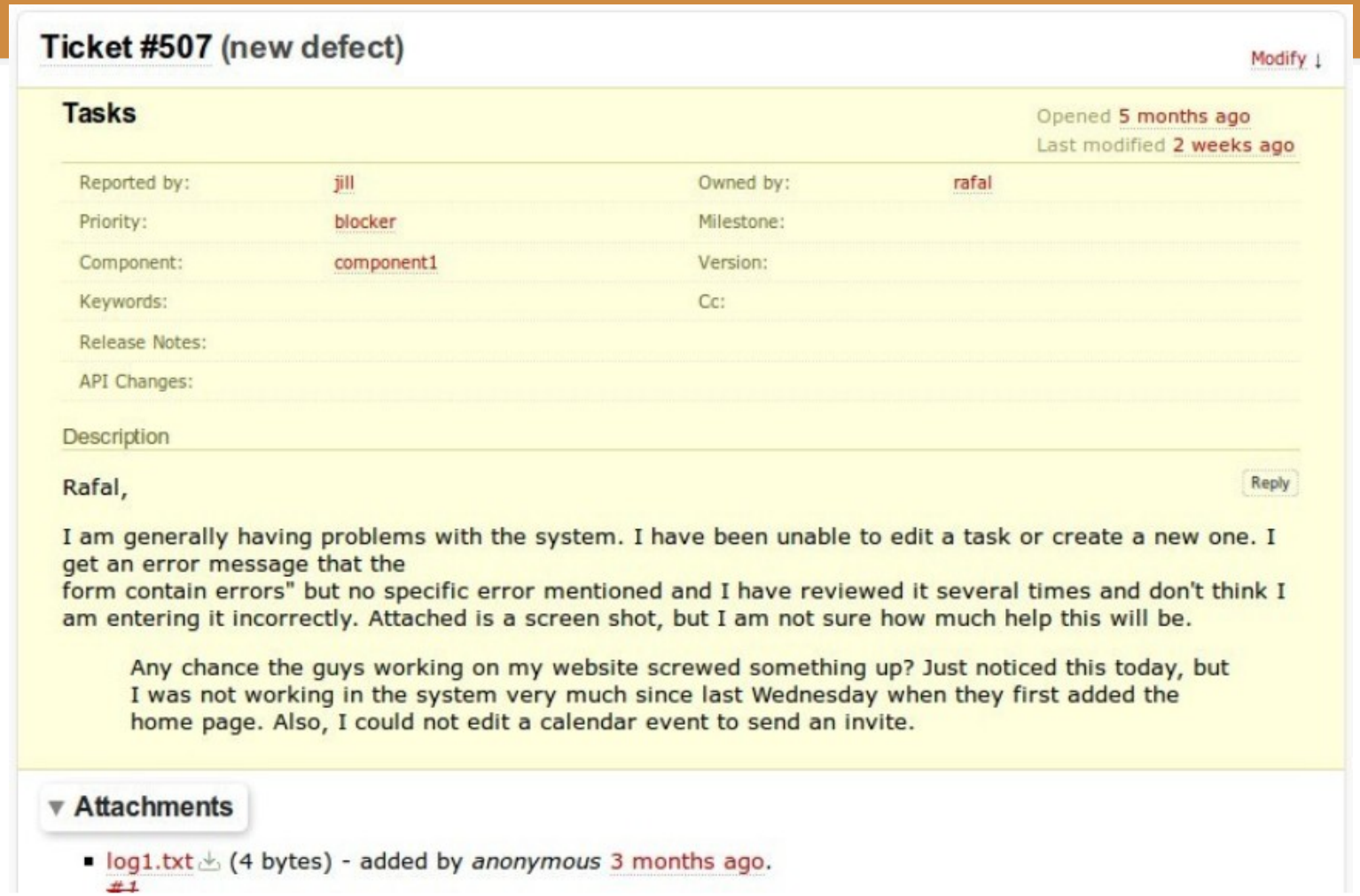
Tools for WBS and beyond (Trac)

- Trac is an enhanced wiki and issue tracking system for software development projects (bug tickets)
- Trac uses a minimalistic approach to web-based software project management
- It provides an interface to Subversion and Git (or other version control systems), an integrated Wiki and convenient reporting facilities

Tools for WBS and beyond (Trac)



Trac Ticket System



Ticket #507 (new defect) Modify ↓

Tasks

Opened 5 months ago
Last modified 2 weeks ago

Reported by:	jill	Owned by:	rafal
Priority:	blocker	Milestone:	
Component:	component1	Version:	
Keywords:		Cc:	
Release Notes:			
API Changes:			

Description

Reply

Rafal,

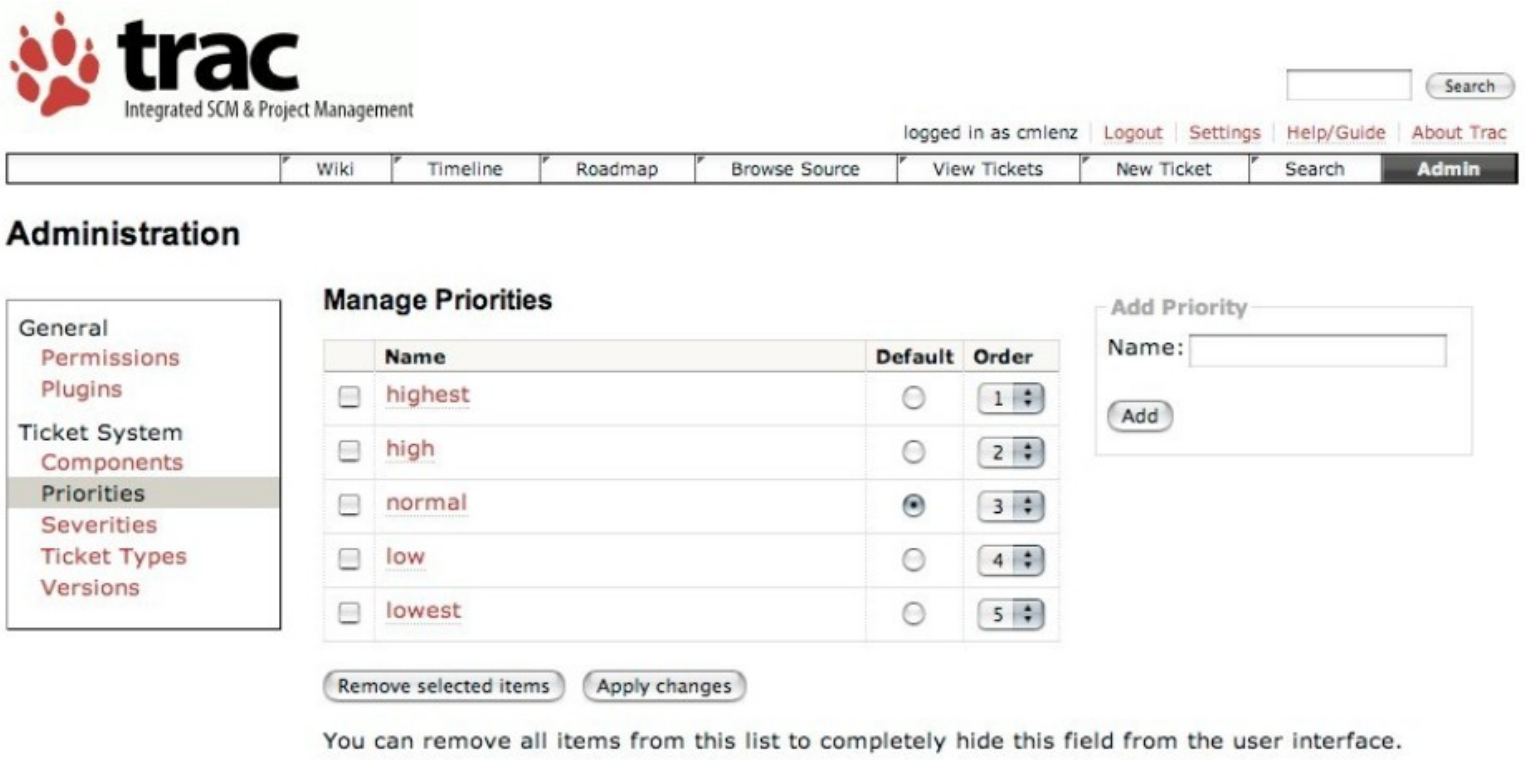
I am generally having problems with the system. I have been unable to edit a task or create a new one. I get an error message that the
form contain errors" but no specific error mentioned and I have reviewed it several times and don't think I am entering it incorrectly. Attached is a screen shot, but I am not sure how much help this will be.

> Any chance the guys working on my website screwed something up? Just noticed this today, but I was not working in the system very much since last Wednesday when they first added the home page. Also, I could not edit a calendar event to send an invite.

▼ **Attachments**

- log1.txt (4 bytes) - added by *anonymous* 3 months ago.

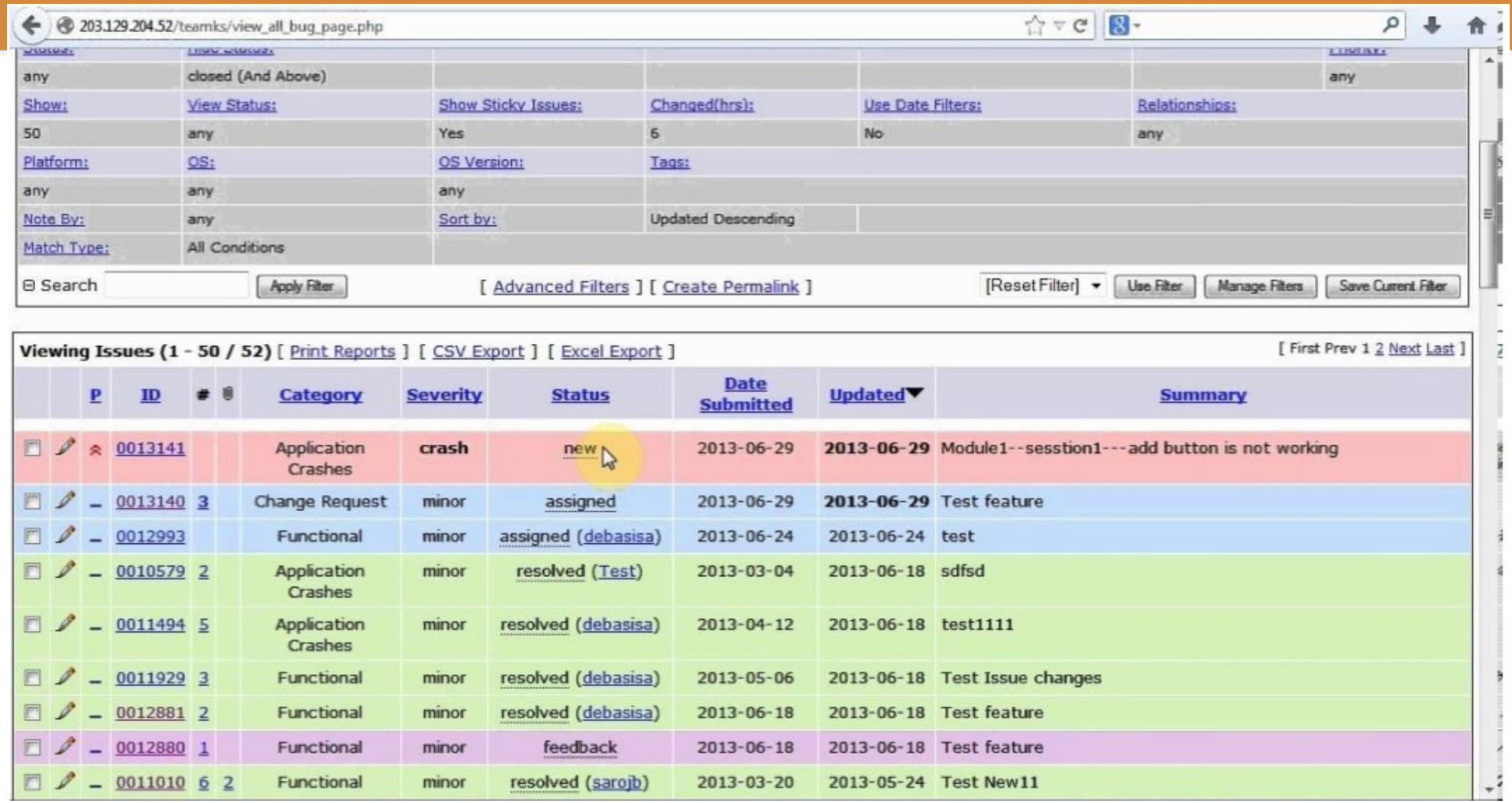
Trac Ticket Data



Mantis Bug Tracker

- MantisBT makes collaboration with team members & clients easy, fast, and professional
- MantisBT is an open source issue tracker that provides a delicate balance between simplicity and power
- Users are able to get started in minutes and start managing their projects while collaborating with their teammates and clients effectively
- Try for free: https://mantishub.com/

Mantis Issue



203.129.204.52/teamks/view_all_bug_page.php

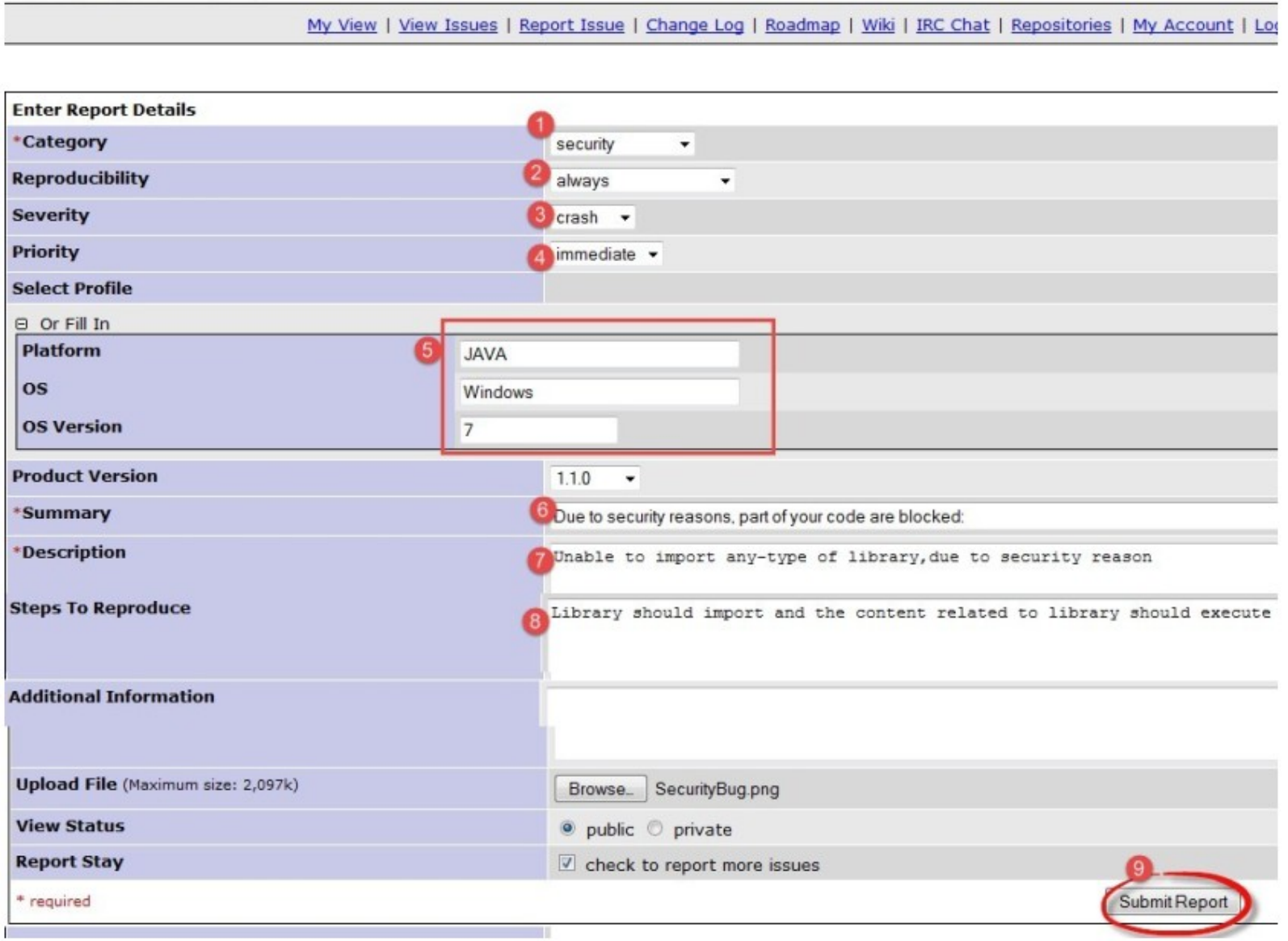
any	closed (And Above)				any
Show:	View Status:	Show Sticky Issues:	Changed(hrs):	Use Date Filters:	Relationships:
50	any	Yes	6	No	any
Platform:	OS:	OS Version:	Tags:		
any	any	any			
Note By:	any	Sort by:	Updated Descending		
Match Type:	All Conditions				

⊟ Search | Apply Filter | [Advanced Filters] [Create Permalink] | [Reset Filter] | Use Filter | Manage Filters | Save Current Filter

Viewing Issues (1 - 50 / 52) [Print Reports] [CSV Export] [Excel Export] [First Prev 1 2 Next Last]

P	ID	#		Category	Severity	Status	Date Submitted	Updated	Summary
	0013141			Application Crashes	crash	new	2013-06-29	**2013-06-29**	Module1--sesstion1---add button is not working
	0013140	3		Change Request	minor	assigned	2013-06-29	**2013-06-29**	Test feature
	0012993			Functional	minor	assigned (debasisa)	2013-06-24	2013-06-24	test
	0010579	2		Application Crashes	minor	resolved (Test)	2013-03-04	2013-06-18	sdfsd
	0011494	5		Application Crashes	minor	resolved (debasisa)	2013-04-12	2013-06-18	test1111
	0011929	3		Functional	minor	resolved (debasisa)	2013-05-06	2013-06-18	Test Issue changes
	0012881	2		Functional	minor	resolved (debasisa)	2013-06-18	2013-06-18	Test feature
	0012880	1		Functional	minor	feedback	2013-06-18	2013-06-18	Test feature
	0011010	6	2	Functional	minor	resolved (sarojb)	2013-03-20	2013-05-24	Test New11

Mantis Issue Data



Bug Triage

- Collect information (enough?)
- Identify duplicates / similar problems
- How to reproduce?
- Set Priority + Importance
- Assign
- Fast paced
- Face-to-face (real-time)
- (Support levels)

Questions?

- …
- Or write me an email to gla@inf.elte.hu

Market Research - Canvas

Goal: Create Work Breakdown Structure diagram

https://creately.com/plans/

Free for ...

- 5 Public Documents
- 1 Folder
- Only 3 Collaborators