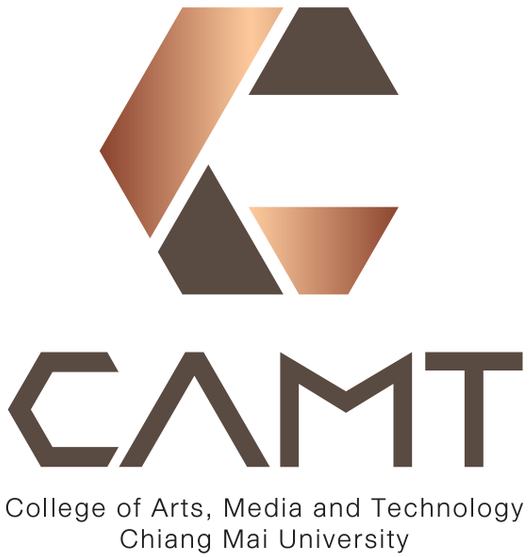
SE 234 Advance Software Development

#4 Issue Tracking



Lect Passakorn Phannachitta, D.Eng.

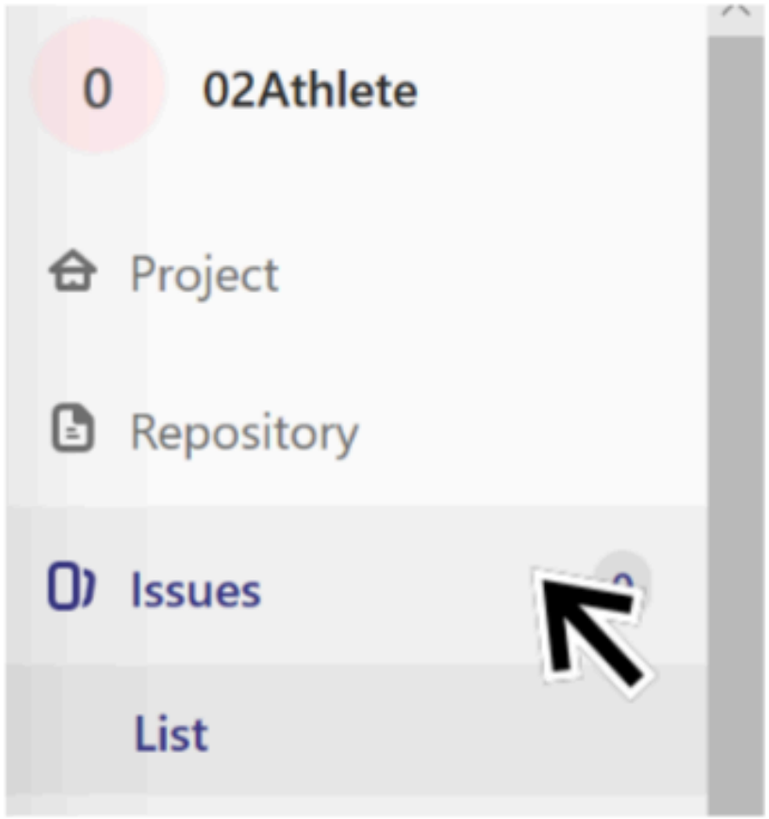
passakorn.p@cmu.ac.th

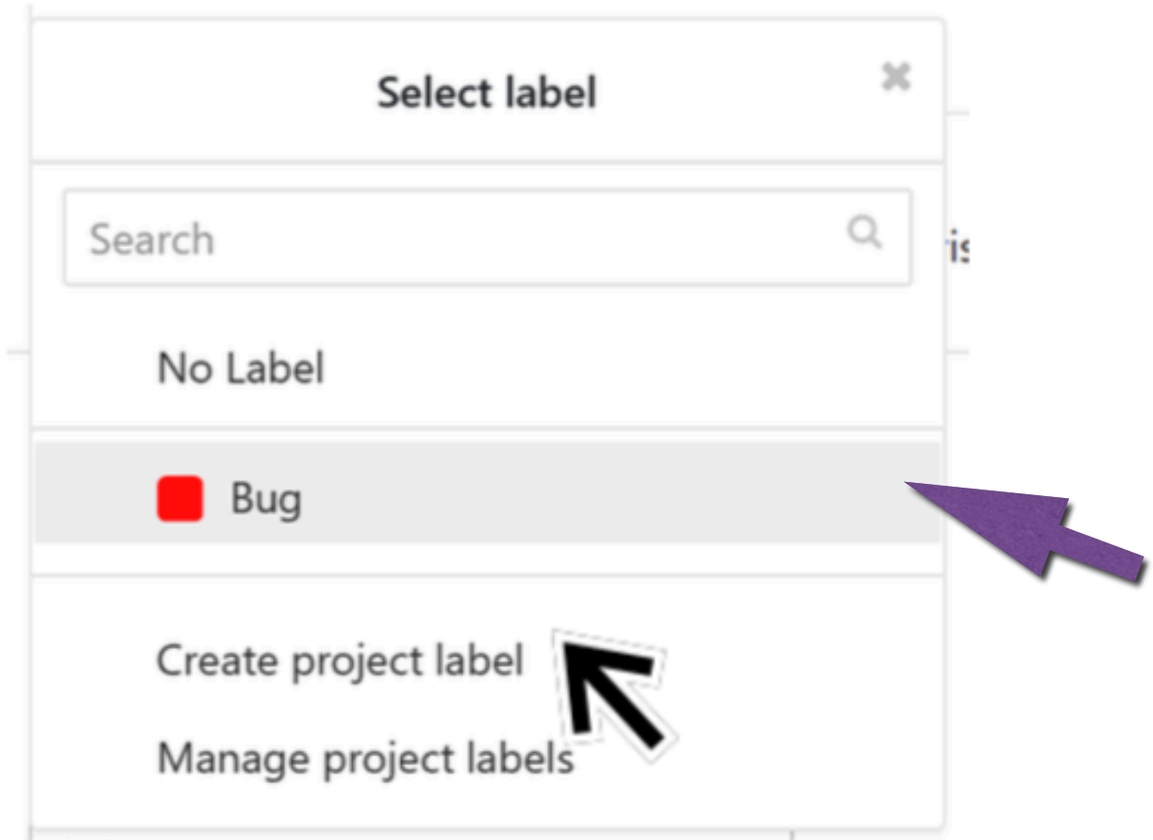
College of Arts, Media and Technology

Chiang Mai University, Chiangmai, Thailand

Issue Tracking

- In the previous lab



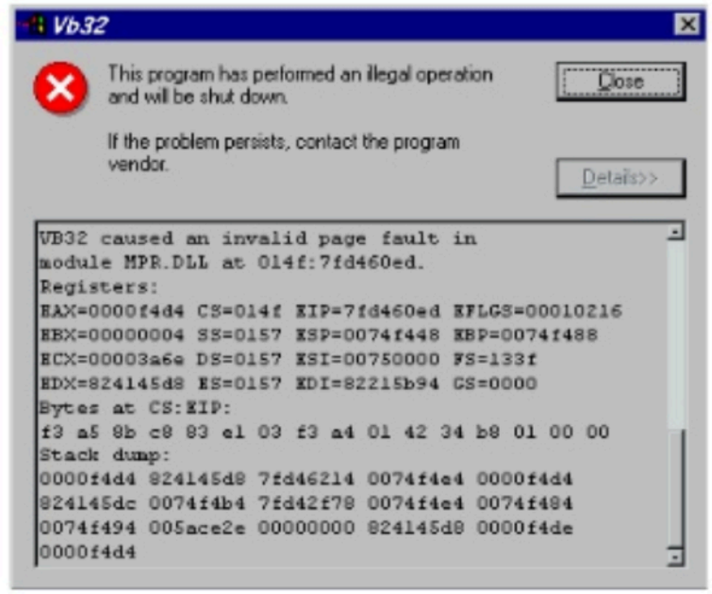


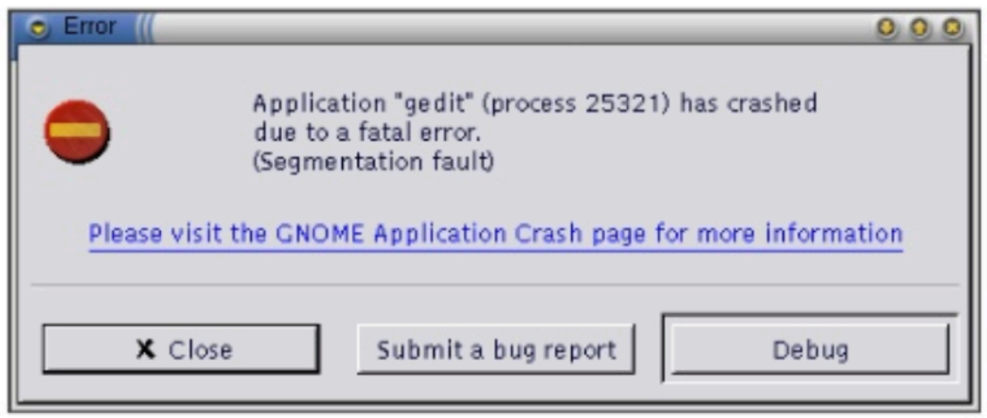
Issue Tracking

- Before we start discussing about issue tracking, let's go through its fundamental backgrounds
  - What is it all about ?
  - What is meant by issue ?
  - The life cycle of issue

What is all about ?

- Any software can fail.

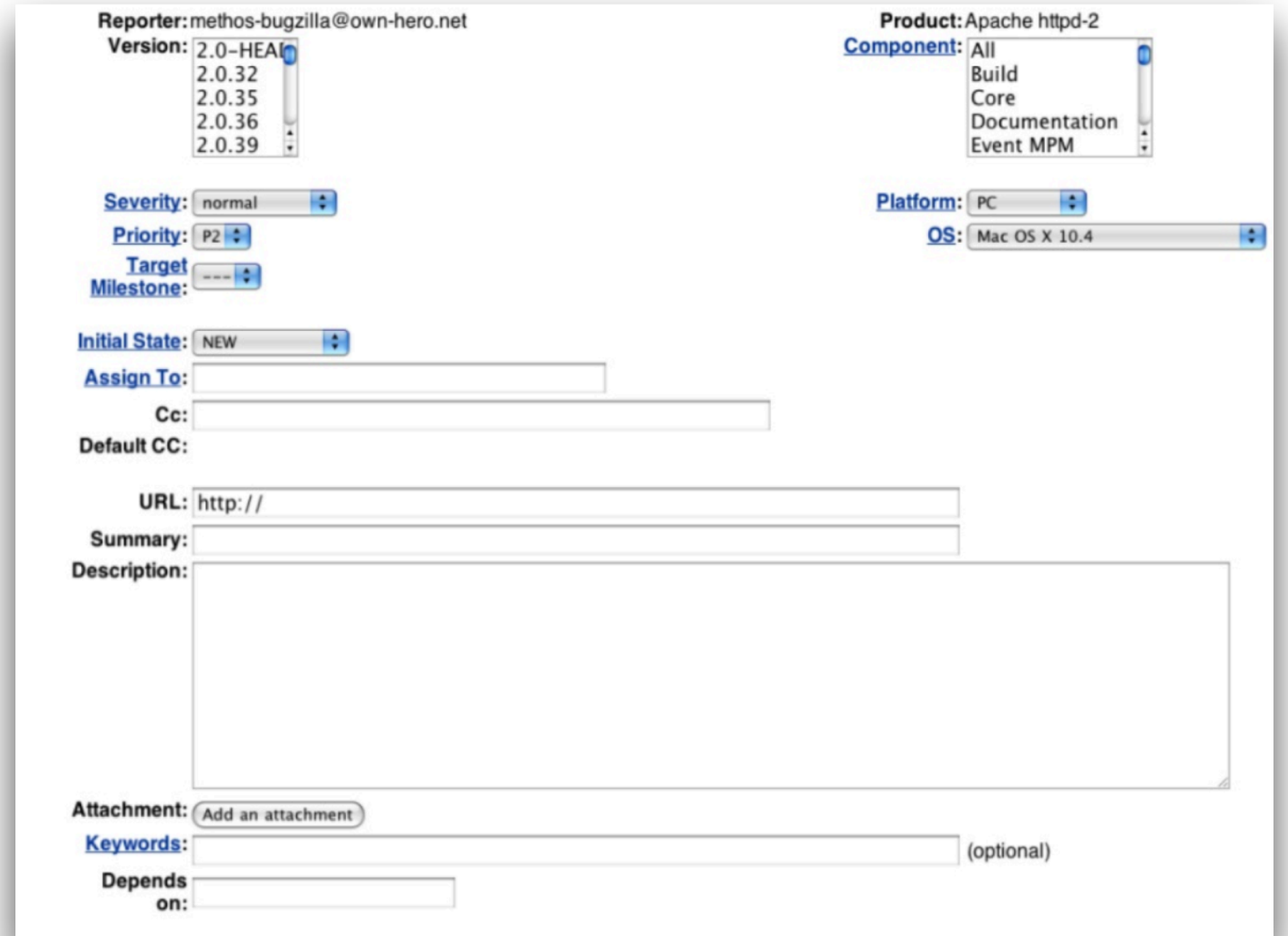




https://www.slideshare.net/tom.zimmermann (Accessed Jan 2018)

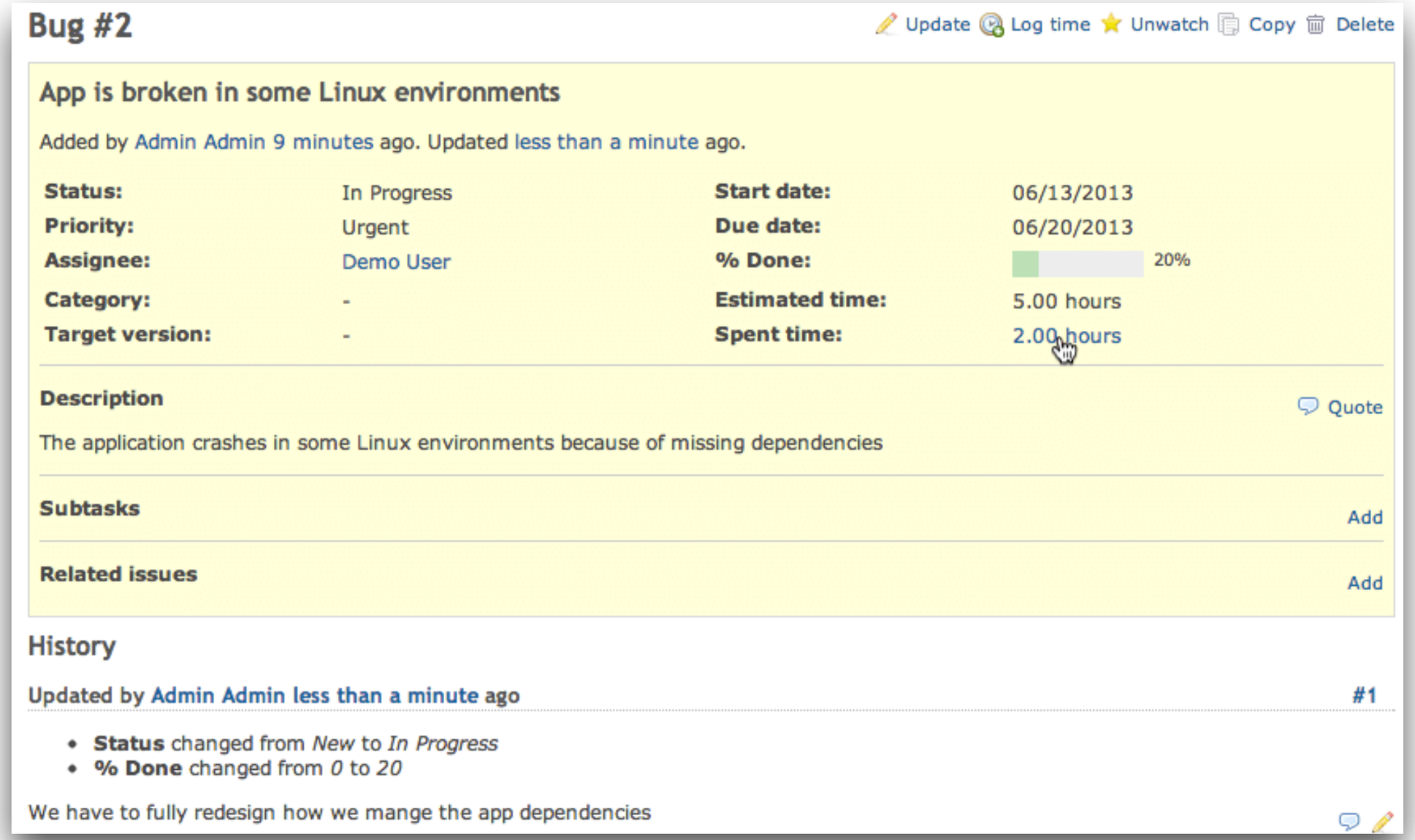
What is all about ?

- To fix the problem, the failure must be reported.



What is all about ?

- The reported failure should be tracked until it is resolved.



Terminology

- **Bug** is an error found **before** the software goes into its production stage. — i.e., found by a dev or a tester
- **Defect** is an error found **after** the software goes into its production stage. — i.e., found by a user or a client
- **Error** is anything that we cannot confirm that it is compiled with our expected behavior in any stage.

> In this course, let the term **issue** denote the superset of all these terms.

Terminology - Note

Elsewhere, the terms **bug, defect, fault, failure, anomaly, incident, false positive, error, and mistake** are commonly used interchangeably.

Issue

- An issue can be considered as any exception that can influence the software to function **deviated** from our expectation, e.g., producing incorrect results.
- It may be due to **mistake** produced in any stage of software development.
- Actually, we should track not only issues, but also any suspicious behavior
  - This will allow us to properly handle them.

After an Issue is Reported

- We need to
  - Identify the cause of problem
  - Modify the code
  - Monitor
  - Verify
  - etc.

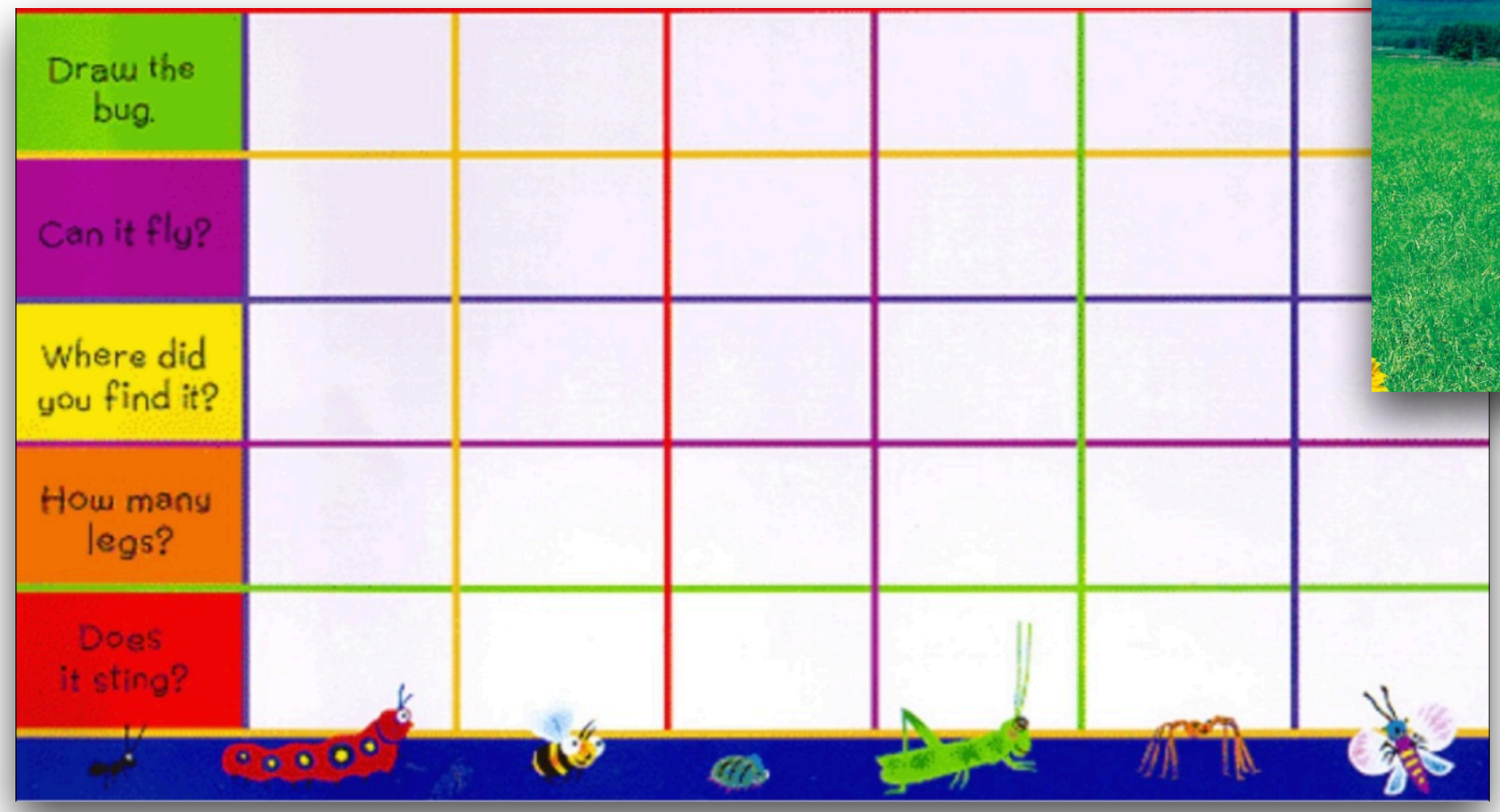
A Bug Report

- Identify the cause of problem
  - Modify the code
    - Verify
      - Monitor
        - Etc.

The document used during the communication of these activities is called a **bug report**.

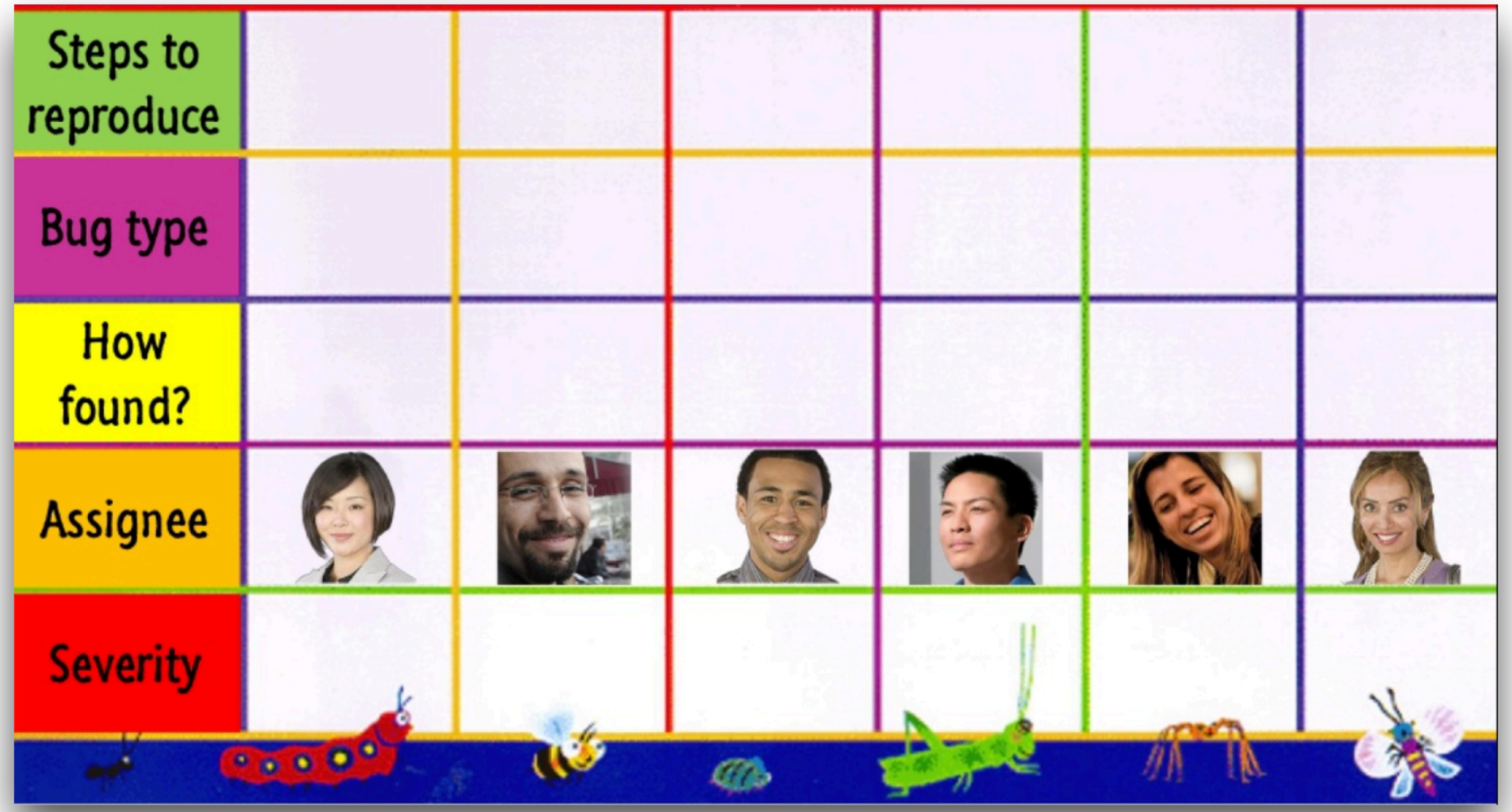
The standardization of this activities is referred to as a **bug life cycle**

A Bug Report



https://www.slideshare.net/tom.zimmermann (Accessed Jan 2018)

A Bug Report



https://www.slideshare.net/tom.zimmermann (Accessed Jan 2018)

A Bug Report

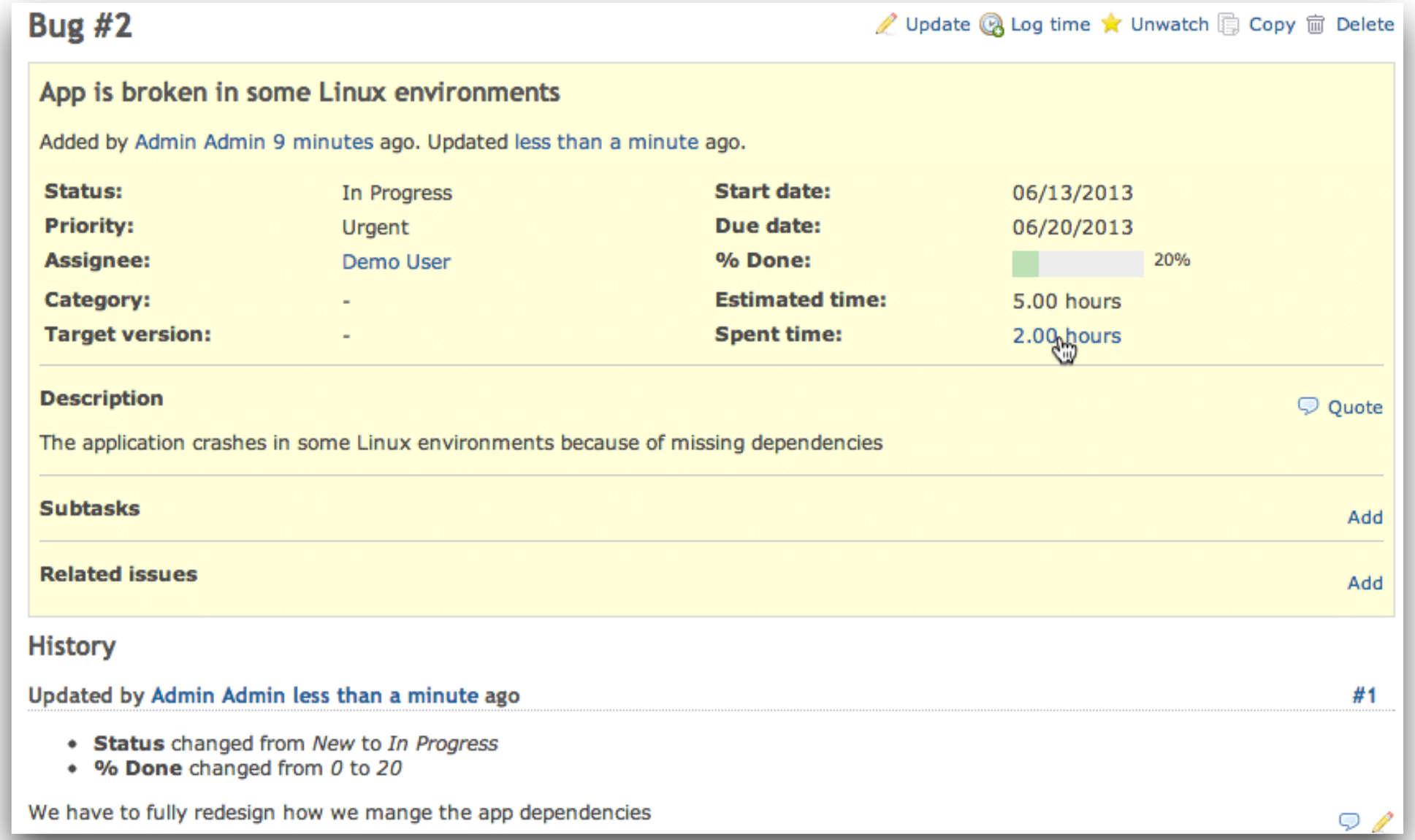
- A technical document describing failure.
- Structure — At **reporter** side
  - ID
  - Project
  - Product
  - Type
  - Version
  - Detected version
  - Summary
  - Description
  - Step to replicate
  - Actual result
  - Expected result
  - Attachments
  - Remarks
  - Reporter

A Bug Report

- A technical document describing the failure.
- Structure — At **manager** side
  - Assigned to
  - Fixed by
  - Current State
  - Dated Closed
  - Severity
  - Priority

A Bug Report — Example

- In Redmine



A Bug Report — Example

- In Bugzilla

Bug 65535 - Running large tests with 500 or more users results in an NPE

Status: CLOSED DUPLICATE of ~~bug 65532~~
Alias: None
Product: z_Archived
Component: Hyades (show other bugs)
Version: unspecified
Hardware: PC Windows XP

Importance: P3 critical (vote)
Target Milestone: ---
Assignee: Marius Slavescu — ECA
QA Contact:

URL:
Whiteboard: closed460
Keywords:

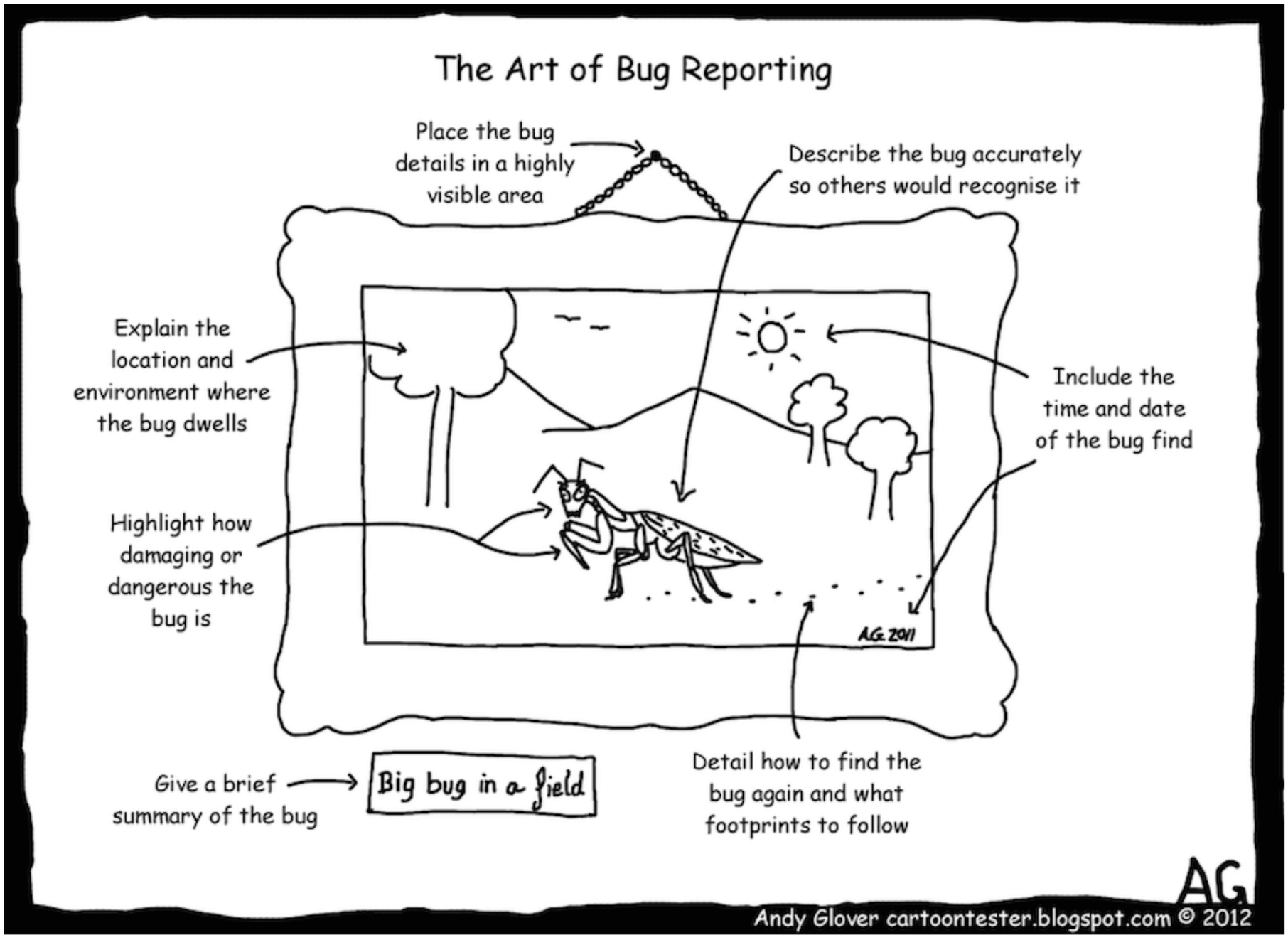
Depends on:
Blocks:

Reported: 2004-06-03 09:36 EDT by Ashish Mathur — ECA
Modified: 2012-02-14 16:41 EST (History)
CC List: 0 users

See Also:

Who	When	What	Removed	Added
jptoomey	2004-06-03 10:02:00 EDT	Assignee	jptoomey	slavescu
slavescu	2004-06-03 15:28:34 EDT	Status	NEW	RESOLVED
		Resolution	---	DUPLICATE
sluiman	2004-11-03 11:22:48 EST	Target Milestone	---	3.0 M10
paulslau	2009-06-30 12:37:55 EDT	Status	RESOLVED	CLOSED
		Whiteboard		closed460
webmaster	2012-02-14 16:41:08 EST	Component	Models.test	Hyades
		Version	3.0	unspecified
		Product	Hyades	z_Archived
		Target Milestone	3.0 M10	---

A Recommended Reporting Practice



https://usersnap.com/blog/easy-bug-tracking-hacks-developers/ (Accessed Jan 2018)

Severity vs Priority

- **Severity** implies technical damages
  - Critical / Major / Medium / Low
- **Priority** implies business damages
  - High / Medium / Low

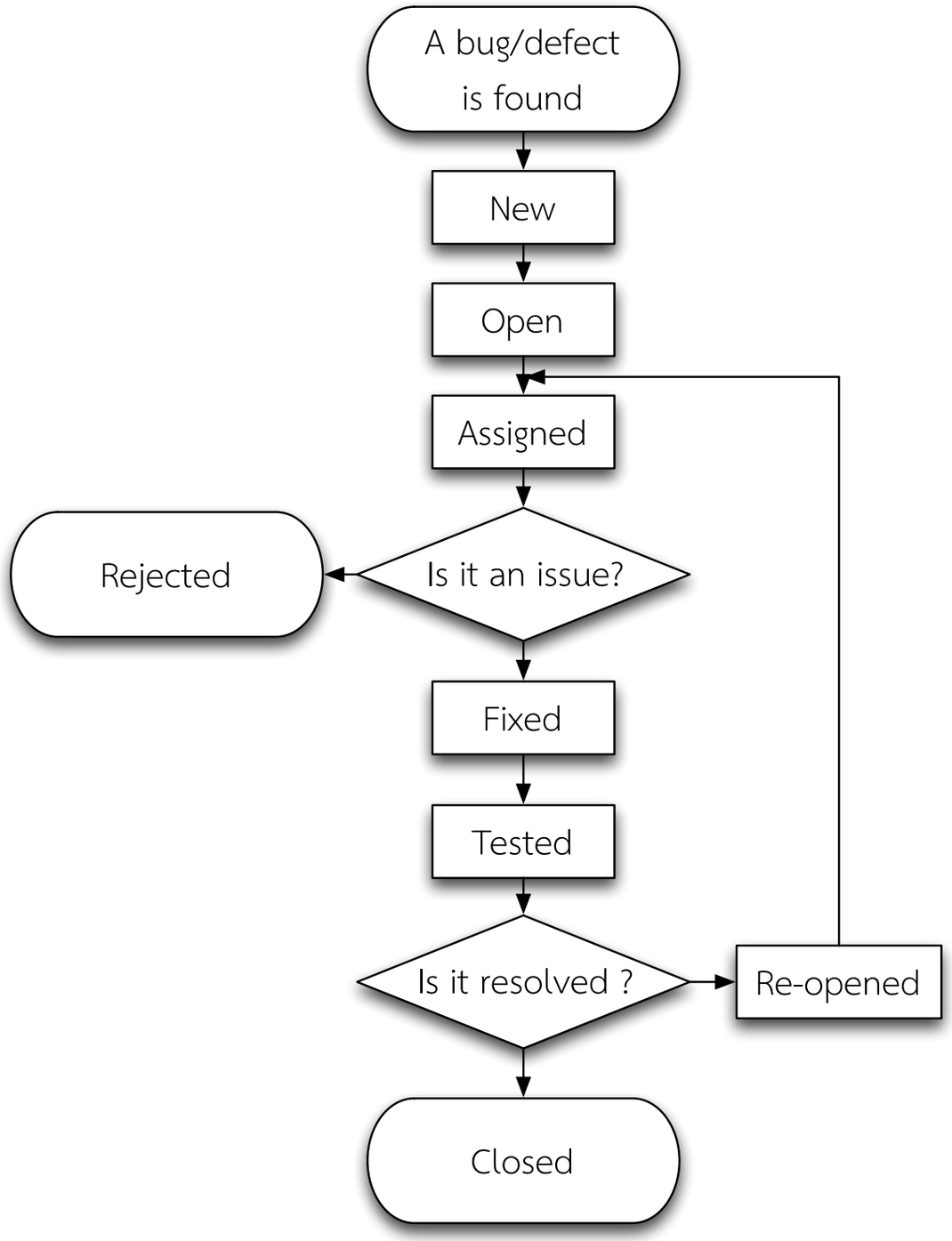
Quality Indicators of a Bug Report

- Atomic, i.e., one report for one matter
- Clear for understanding
- Actionable
- Move quickly from open to close
- Never reopen

Issue life cycle

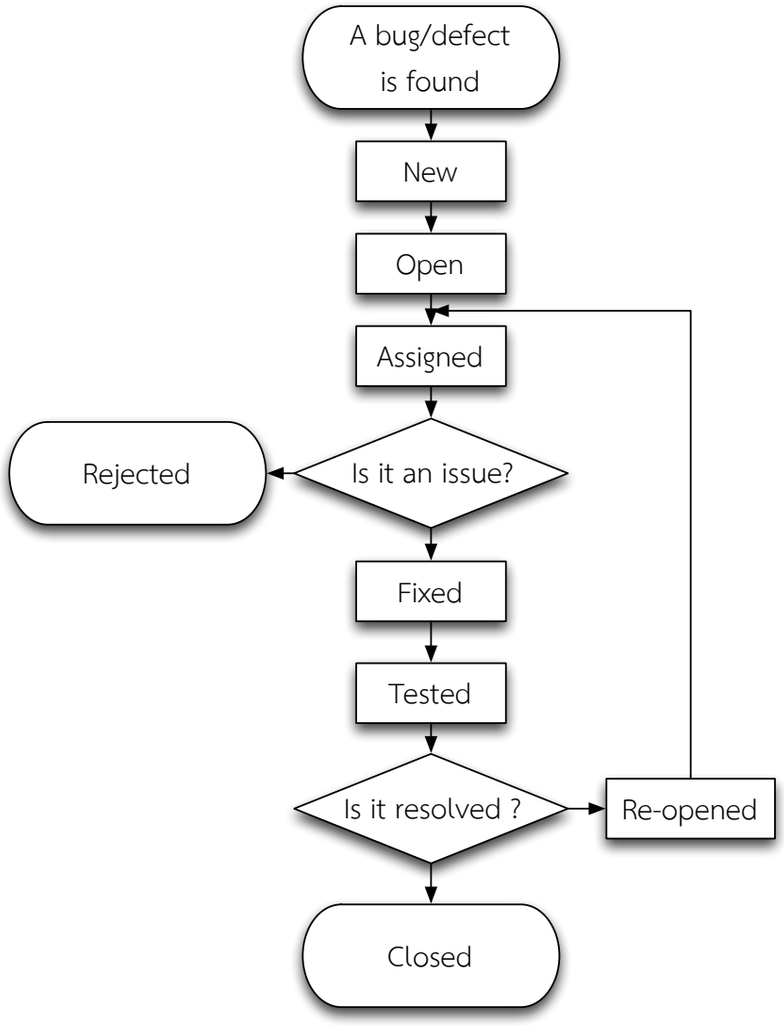
- In software development process, an issue has its life cycle.
  - The duration between the time it was found and the time it is closed successfully or rejected.
- An explicit life cycle allows us to ensure that the process is standardized
  - Generally, in any certain point of time —
    - More than one issues are being taken care of;
    - More than one people are actively interact with an issue.

Issue life cycle



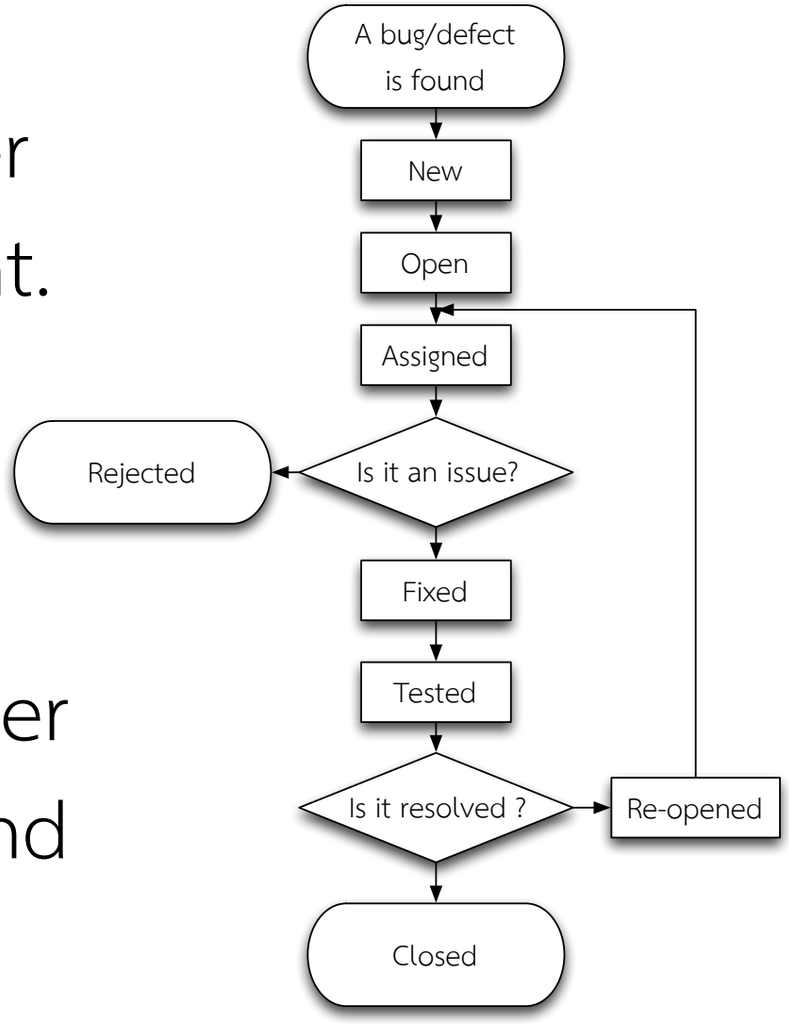
Stages of Issue Life Cycle

- **New**
  - When a bug or a defect is introduced for the first time.
  - It is not yet approved/acknowledged by the core developer team/manager.



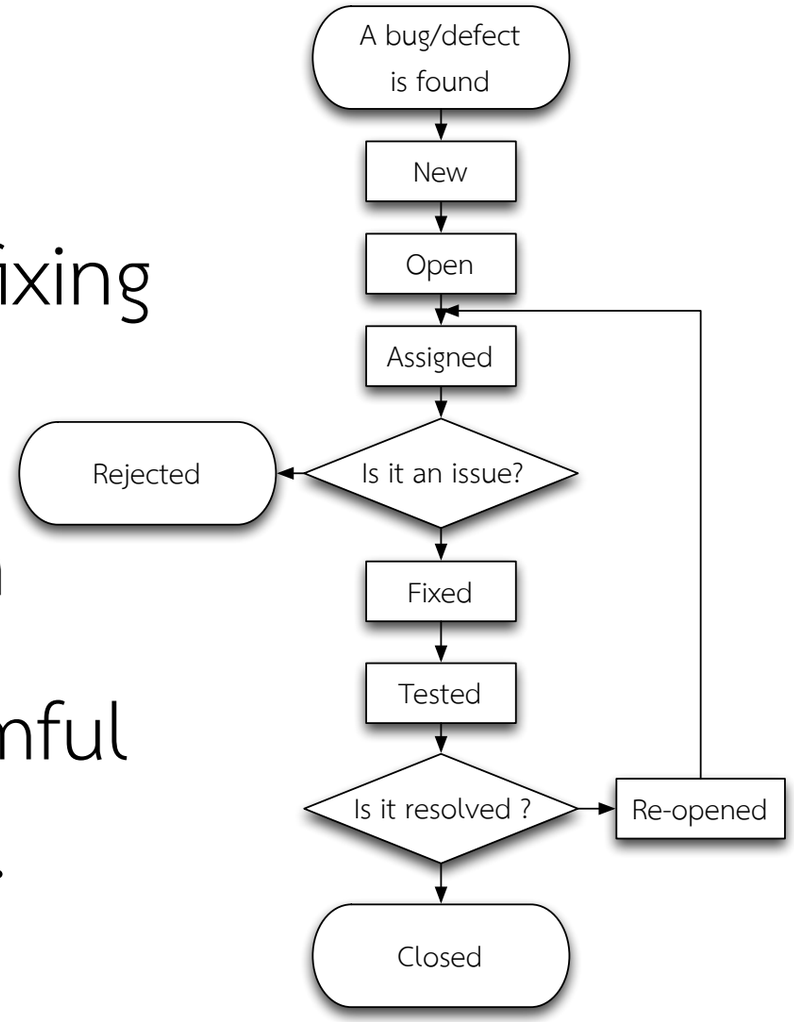
Stages of Issue Life Cycle

- Open
  - It is acknowledged by the core developer team / manager, waiting of an assignment.
  - **Open** can be assigned immediately, or **deferred** until it is needed to be fixed.
  - A common practice — the lead tester approves that the bug is genuine and changes the state from **New** -> **Open**



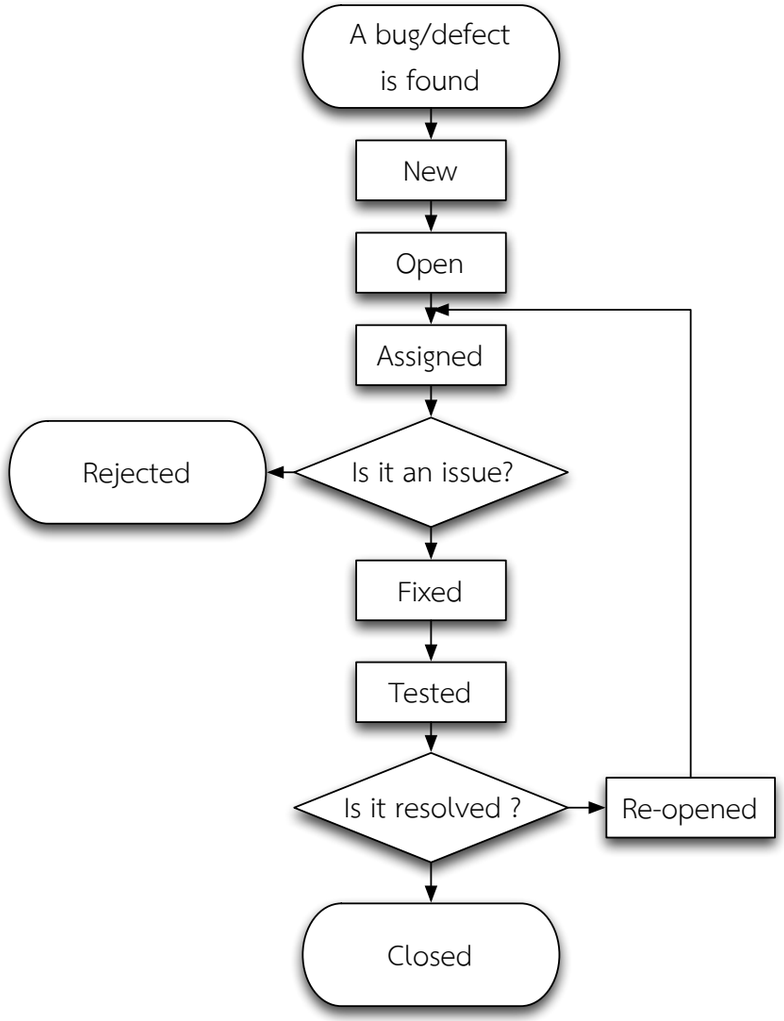
Stages of Issue Life Cycle

- Assign
  - After an assignment, there will be a responsible person who takes care of fixing the issue.
  - This step needs a careful consideration
    - Excessive assign/reassign cycles is harmful to the software development process.



Stages of Issue Life Cycle

- **Fixed**
  - After the assignee makes necessary code changes and verifies its correctness (by herself/himself), the stage can be changed from **open** to **fixed.**

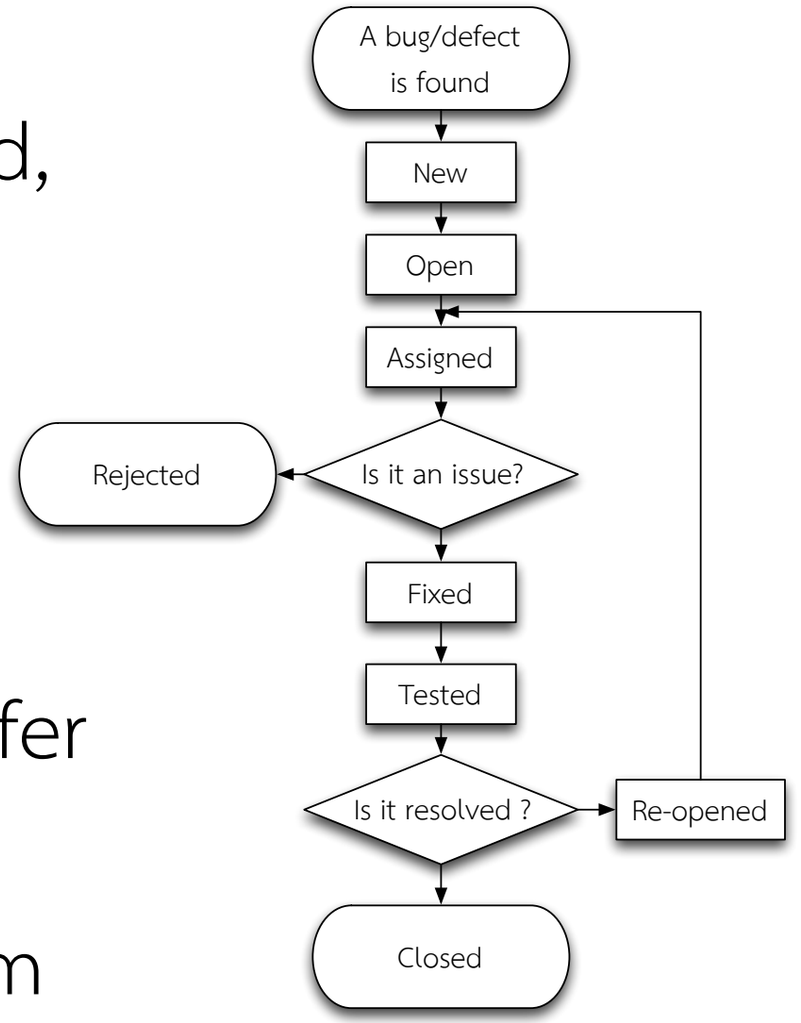


Variation of the Fixed stage

- **Wont't fix** — If a bug/defect will never be fixed.
- **Duplicate** — If a bug/defect is reported twice or two reports mention the same concept.
- **Incomplete / Cannot reproduce / Need more information** — The associated bug report does not contain sufficient information towards the resolving.
- **Works as designed / Not a bug** — the reported problem is not a considered as a bug

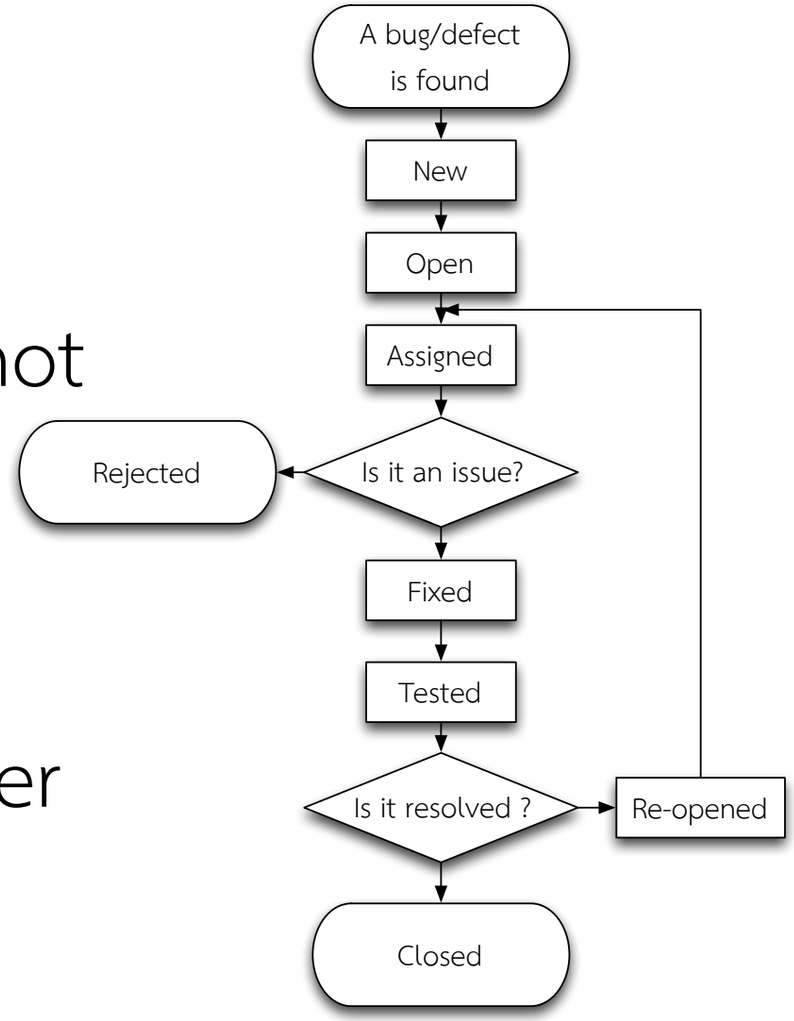
Late Stages of the Issue Life Cycle

- **Verified**
  - After it is thoroughly reviewed and tested, the stage can be changed from **fixed** to **verified**
- **Closed**
  - In some systems, **verified** and **closed** refer to the same stage. For the otherwise, **closed** refers to the stage when the team feels that the issue no longer exists.



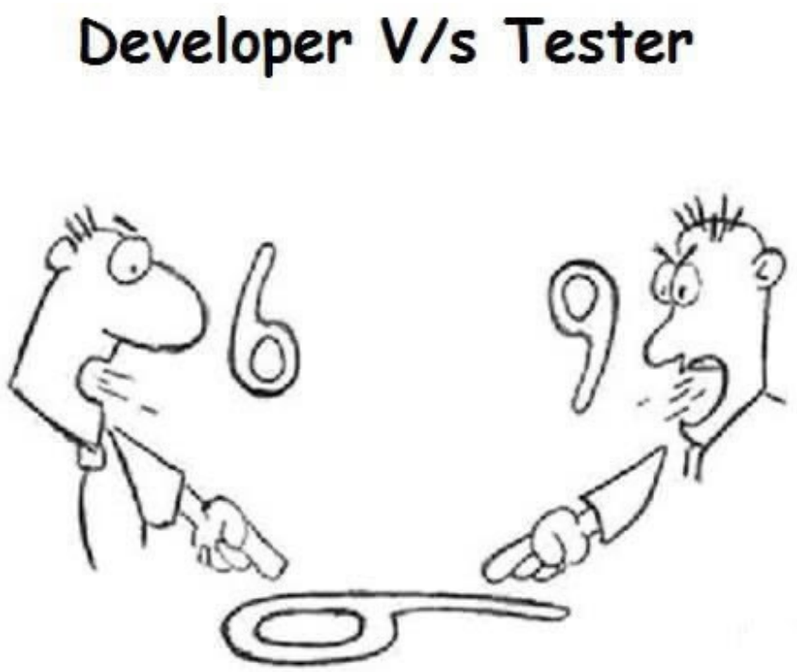
Other Stages which are not in a Normal Flow

- **Deferred**
  - Acknowledged but fixing it is not urgent.
- **Rejected**
  - If the team feels that the bug/defect is not genuine, she/he can reject the bug.
- **Reopened**
  - It was once **verified** or **closed**, but is later required to be re-examined again.
  - Reopen due to slackness of the team is practically unacceptable.



Practical Suggestion

- Communicate as much as possible
- Avoid noise in the comments
- Agree upon what a closed issue meant for



Issue Tracking

Issue Tracking

- A process used by quality assurance and developers to keep track of software problem and resolutions.
- Important because complex software products can have numerous defects throughout its lifetime.
  - Some are due to human errors.
  - Some are due to software evolution.
  - Some are due to software aging.

Issue Tracking

- At the scale of thousands of bugs and defects existed on one product, it is commonly not easy to manually evaluate, prioritize, and analyze all the reported issues, mainly due to the resource and time constraint.
- In turn, an issue tracking software, a.k.a., bug tracking system are made fully utilized.
  - A kind of database system that stores important information about bugs and defects, and helps the team members to manage them.
  - Transparent the entire process to all the stakeholders.

Issue Tracking

- In short

DB of records - i.e., **Bug reports**

+

A workflow driven by the state and the owner fields -

i.e., **Stage in the life cycle for each bug tied with the assignee**

Issue Tracking — Ultimate Goal

- Prevent defects
- Reduce the defect density

Of cause, this is not simply achieved by solely making utilized an issue tracking system. The information stored in the system will be useful in indicating whether the software development process is imperfect or flawed, major cause of most defects.

... will be further discussed in 953322

Some introduction

- In fact, we only ever know about the defects that are found — meaning that we never know the “true” value of defects

- It is observed that the more defects we found, the more and more defects will be introduced

Some introduction

- It is not true that “After we have found and fixed a lot of problems, now our software is OK”

- But .. “If testing reveals lots of bugs, likely that final product will be very buggy too”

Defect density

$$\frac{\#\text{Defects}}{\text{Software size}}$$

- E.g., 1 defects per #line of code
- Generally, it discusses that in the released code
- **Defect density at release** is a known measure of Organizational capability

Summary — Issue Tracking

- Keep track of all the bugs/defects that have been discovered.
- Keep track of all the steps required to validate, correct and take preventive action for any bug/defect.

To avoid the loss of any reported issue.

To ensure that developers do not work on any non-defect.

To control the correction activity.

Question Time