Defect Life Cycle and Defect Reporting/ Tracking

What is a bug in software testing?

The Bug is the informal name of defects, which means that software or application is not working as per the requirement.

While testing the application or executing the test cases, the test engineer may not get the expected result as per the requirement. And the bug had various names in different companies such as error, issues, problem, fault, and mistake, etc.

Why defect/bug occur?

- Wrong coding
- Missing coding
- Extra coding

Wrong coding

Wrong coding means improper implementation.

For example: Suppose if we take the Gmail application where we click on the **"Inbox"** link, and it navigates to the **"Draft"** page, this is happening because of the wrong coding which is done by the developer, that's why it is a bug.

Missing coding

Here, missing coding means that the developer may not have developed the code only for that particular feature.

For example: if we take the above example and open the inbox link, we see that it is not there only, which means the feature is not developed only.

Extra coding

Here, extra coding means that the developers develop the extra features, which are not required according to the client's requirements.

Bug tracking tool

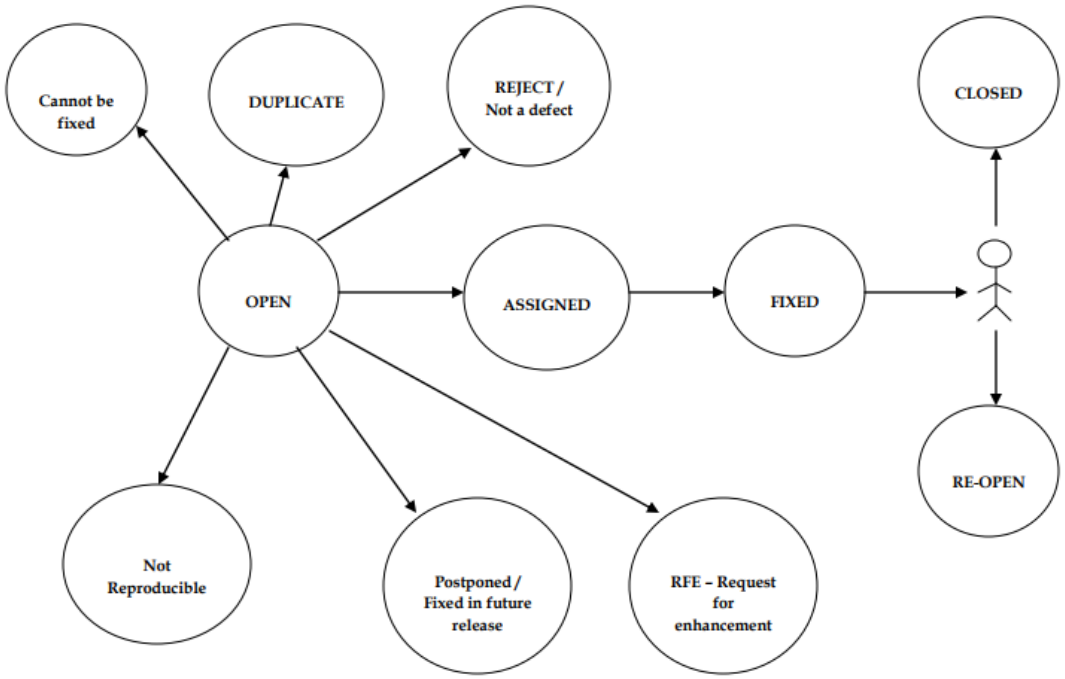
- Jira
- Bugzilla
- Redmine
- Mantis
- Backlog

Jira is one of the most important bug tracking tools. Jira is an open-source tool that is used for bug tracking, project management, and issue tracking.

Jira includes different features like reporting, recording, and workflow. In Jira, we can track all kinds of bugs and issues, which are related to the software and generated by the test engineer.

****

DEFECT LIFE CYCLE



Customer gives requirements – developers are developing the s/w – testing team is writing test cases looking at the requirements

Developer develops the product – the test engineer starts testing the product – he finds a defect – now the TE must send the defect to the development team.

As soon as the test engineer finds the bug, status is given as New, which indicates that a bug is just found.

He prepares a defect report – and sends a mail to the Development lead then the status is “new/ open”. Development lead looks at the mail and at the bug – and by looking at the bug – he comes to know which development engineer developed that feature which had a bug – and sends the defect report to that particular developer and says “bug assigned”.

This new bug needs to be reported to the concerned Developer by changing the status as Assigned so that the responsible person should take care of the bug.

Then the Developer first goes through the bug, which means that the Developers read all the navigation steps to decide whether it is a valid bug or not.

Based on this, if the bug is valid, the Developer starts reproducing the bug on the application, once the bug is successfully reproduced, the Developer will analyze the code and does the necessary changes, and change the status as **Fixed** and send it/ assigned it to test engineer saying “bug fixed” – he also “cc mail” to the development lead.
Once the code changes are done, and the bug is fixed, the test engineer retest the bug, which means that the test engineer performs the same action once again, which is mentioned in the bug report, and changes the status accordingly:

Close, if the bug fixes properly, and functionally working according to the requirement.

OR Re-open, if the bug still exists or not working properly as per the requirement, then the bug sends it back to the Developer once again

This process is going on continuously until all the bugs are fixed and closed.
Every bug will have a unique number.

To whom we can assign the bug:

- **Developers**: If we know who has developed that particular module.
- **Developers lead**: If we don't know the Developer who has developed the particular module.
- **Test lead**: When we don't have any interaction with the development team.

Important Point: Whenever we find a bug and Developers fix it, we have to check one round of impact area.

Invalid / “REJECT” BUG
Now, when the TE sends a defect report – the Development Lead will look at it and reject the bug.

Bug is rejected because,

1) **Misunderstanding of requirements (Test Engineer misunderstood the requirements OR Developer misunderstood the requirements)**
2) **While installing or configuring the product** – wrongly configured or installed the product and we found a bug in the product – send it to development team – developer says “reject” because he looks at the defect report and comes to know that the Testing team has not installed the product correctly

3) Referring to old requirements
Chances are there that the Testing team may be referring to old requirements while testing the product. For ex, - in the 1st build – developers have given the product for testing in which a

sales link button is there, test engineer does testing and reports any defects – in the 2nd build, customer has given a requirement change to the development team where-in he has asked them to remove the sales link button – but the testing team is not aware of this change – so when the development team gives the new build – obviously they would have removed the sales button – test engineer when he looks at the feature – he is referring to old requirements which has sales link button – but in the new requirements it has been removed which the test engineer is not aware of – so he reports a bug – development team then reply back saying "refer to new requirements in which sales link button has been removed".

4) Because of extra features

Consider the example and figure shown in Pg 130. Here, "help" button is a feature which has been developed by the developer as an extra feature not part of the requirements – obviously when the TE looks at the application and requirements, since Help feature is not there – he reports it as a bug – the developer rejects it saying "it is an extra feature and that he won't test it" – Test Engineer replies back by re-opening the bug saying "update the requirements" – updating the requirements requires lot of running around for the development team, talking to the customer and all that – so he either removes it or talks to the customer.

"DUPLICATE" BUG

The TE finds a bug and sends a defect report and also assigns the bug report a number – let"s say he sends a bug report numbered as Bug 25 – now the developer replies back by saying "Bug 25 is duplicate of Bug 10" i.e, another Test engineer has already sent that bug earlier and it is being fixed.

Why do we get duplicate bugs?

Because of common features – Lets say Test Engineer A and B are testing an application – Test Engineer A and B to test their features must Login to the application using valid username and password before they can go deep into the application and start testing their features – A enters valid username and password and clicks on Login button – it goes to a blank page – this is a bug – so A 134 | P a g e prepares a defect report for this bug and sends it to developer – After sometime, B also tries to login and finds the same bug – he also prepares a defect report and sends it to developer – but developer sends back defect report of B saying "it is a duplicate".

B finds bug in A's module – let"s say A and B are testing their respective features – A is doing functional testing and integration testing on all his features – Similarly B is also testing all his features – In one scenario, B needs to go to A"s feature(say m12) and send data to B"s feature(say m23) – when B clicks on m12, he finds a bug – although its feature of A, he can still prepare a defect report and send it to developers – and B sends the defect report to development team – now, A after testing all his features comes to his feature m12 – he also finds the same defect and sends a defect report – but the developers send it back saying "it is duplicate".

How to avoid duplicate bugs?
Let us consider Test Engineer A is testing the Sales module and finds a defect where-in – he enters all the information and then when he clicks on the Submit button, it is going to the Blank page – he prepares a defect report in which he explains the defect.

Now, before he sends it to developer – A logs into a server named QA – there, he goes into a folder called “Defect Repository” which has all the defect reports prepared by TE(s) and sent to developers and clicks on MS-Search and gives Sales as the search keyword and hits on search button – now, it starts looking inside the folder if there are any defect report which has Sales in it – if the search results in No files – then A will first copy-paste his defect report in the defect repository and then send the defect report to the development team who then assign it to the development engineer.

Now, after a few days – another TE B finds the same defect and he also prepares a report – but before he sends it to the development team. He first logs onto QA and Defect Repository and searches for Sales keyword – he finds the report which was earlier sent by TE A – so he doesn‟t send the defect to development lead

But, always – the TE if he finds a defect similar to his if the Search operation yields some results – before he concludes that his defect has already been sent – he must first read the defect report and ascertain whether it is the same bug which he has found – chances are there that it might have the keyword Sales, but it is about some other defect in Sales feature.

For example, - TE A is testing the password text field – he finds that it is accepting 12 characters also (when the requirement says “it must accept only 10 characters”) – that‟s a bug. So he prepares a defect report which has the keyword password and stores it in the defect repository and also sends it to the Dev team. Now, another TE B is also testing password field – he finds there is a bug where-in password text field is also accepting blank space character – before he prepares a report, he goes to the defect repository and searches for Password keyword – there he finds the earlier report sent by A – but that‟s a different defect – he reads it and realizes that it‟s not the same defect which he has found – so, he prepares a defect report regarding the blank space acceptance in the password text field – and stores it in the defect repository and also sends it to the development team.

- Generally, we don't search for each bug in the repository to check the duplicity.
- To save time, we only search that bug that have common features and dependent features.

Cannot be fixed

Chances are there – Test Engineer finds a bug and sends it to Development Lead – development lead looks at the bug and sends it back saying “cannot be fixed”.

Why does this happen? – Because,

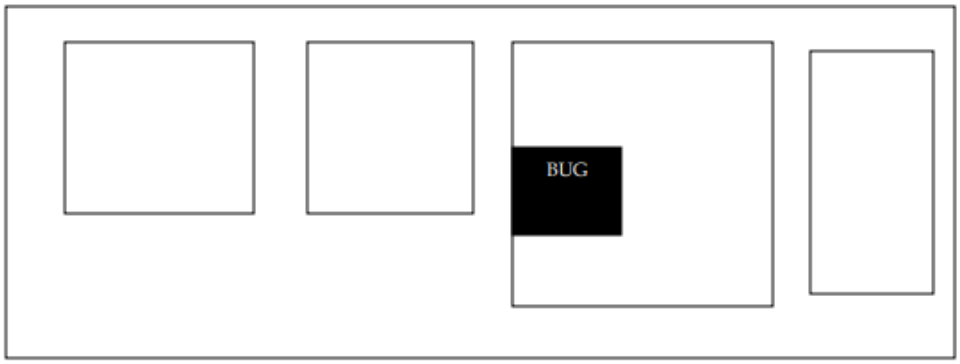
- Technology itself is not supporting i.e, programming language we are using itself is not having capability to solve the problem
- Whenever there is a bug in the root of the product. If it‟s a minor bug, then the development lead says “cannot be fixed”. But, if it‟s a critical bug, then development lead cannot reject the bug
- If the cost of fixing the bug is more than the cost of the bug itself – cost of the bug means loss incurred because of the bug.

Postponed/ Deferred or Fixed in the next release

The deferred status of the bug was not fixed in the initial build because of the time constraints.

a) We find a bug during the end of the release (could be major or minor but cannot be critical) – developers won‟t have time to fix the bug – such a bug will be postponed and fixed in the next release and the bug status will be “open”

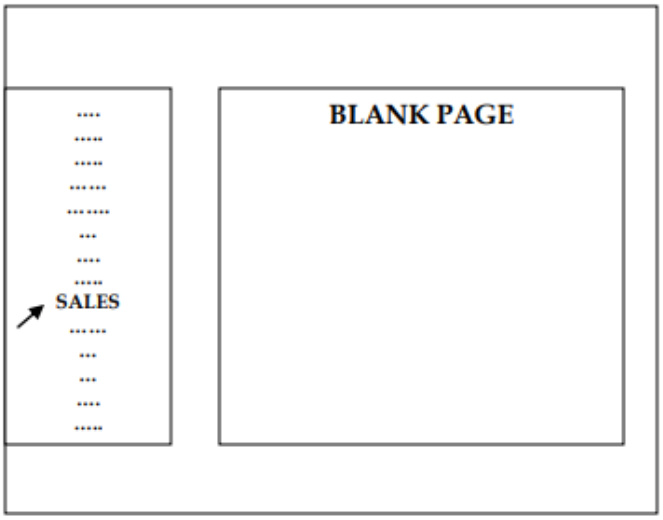
b)



Developers have built the above application – Test Engineers are testing the application and they find a bug – now, the bug is sent to the development team.
Development team looks at the report – and replies back to the testing team saying “they will not fix the bug for now, as the customer is thinking of making some changes to that module or he might as well ask for a requirement change where-in the module itself might have to be removed” – so they won‟t fix the bug for now – but if no requirement change happens, then they will fix the bug.

c) If the bug is minor and it is in feature exposed to internal users. For ex, consider Citibank employee using a s/w where-in he is making keeping track of accounts and customer details – now, for example if the “Sort by name” feature is not working – it‟s ok because it‟s a minor bug because the development team can always fix the bug during the next release

Not Reproducible



Open the application in Mozilla FireFox

1) Open browser and enter www.citibank.com and click on Go button

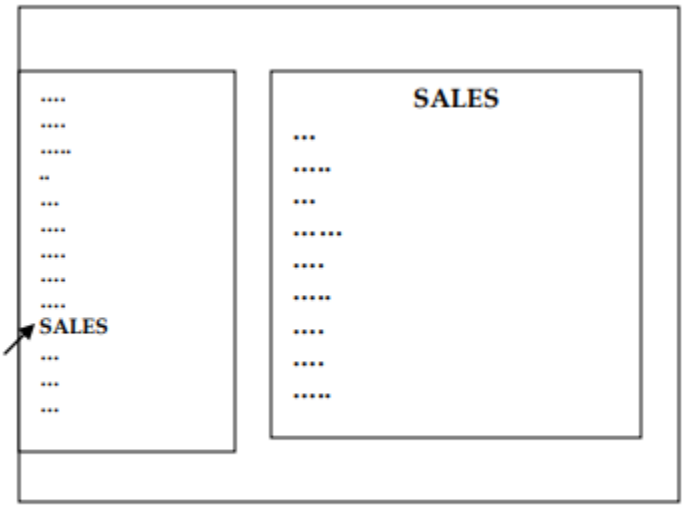
2) Login using valid username and password

3) In the homepage, click on “SALES link button”

4) It goes to blank page

DEFECT REPORT

Let us consider the above application – the TE is testing the above application in Mozilla Firefox– when he clicks on the SALES link button – it goes to a blank page – this is a bug – the TE prepares a defect report and sends it to the development team.



Open the application in Internet Explorer

The development lead looks at the defect report and opens the application in Internet Explorer – when he clicks on the Sales link button, the Sales page opens and the application works perfectly.

Then, what was the problem? – the answer is, the TE didn‟t specify in the defect report in which browser to open the application – so, the application didn‟t work for the TE in Mozilla FireFox – but, when the developer opened it in Internet Explorer, the application worked totally fine.

Since the developer is not able to see any defect – he replies back by saying “I am not able to find the defect” or “I am not able to reproduce the defect”.

Why do we get “not reproducible” defects?

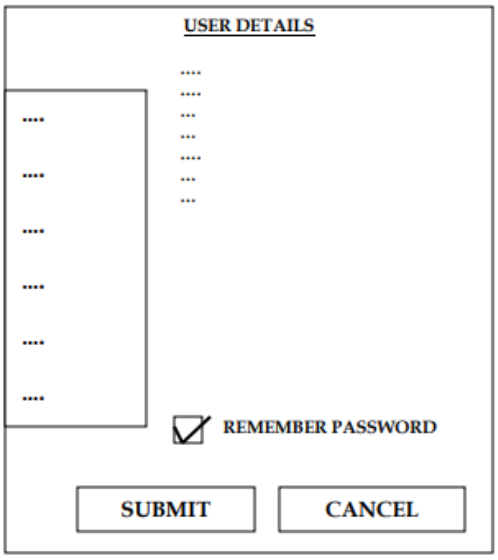
1. Because of platform/ Environment mismatch
2. Because of improper defect report / Incomplete Bug Report
3. Because of data mismatch
4. Because of build mismatch
5. Because of inconsistent defects

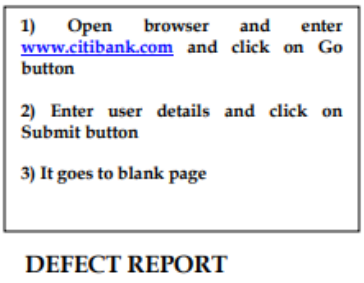
1) Because of platform mismatch

Because of OS mismatch
Because of browser mismatch
Because of „version of browser‟ mismatch
Because of „settings of version‟ mismatch

2) Because of improper defect report

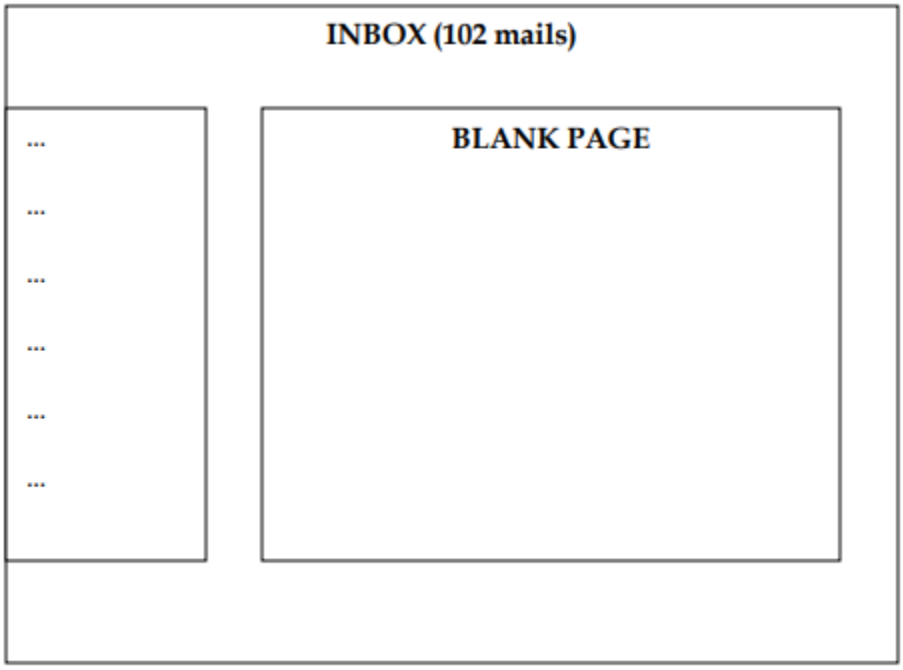
Example to explain this - take figure and explanation in Pg 136 and 137. Here, the TE must have specified to open the application in Mozilla Firefox
Let us consider another example shown below to explain “improper defect report”





In the above application – TE is testing the above application – when he enters all the user details and selects checkbox and clicks on Submit button – it goes to blank page – but, when the TE does not select the checkbox and clicks on Submit button, a new user account is created, i.e, it works fine. The TE prepares a defect report and sends it to the development lead. The Dev team looks at the report(as shown in box) – and tests the application – it is working absolutely fine. Then what happened? – Observe the defect report sent by the testing team – nowhere as he mentioned to “select checkbox and click on submit button” – thus because of the improper defect report – the development team is not able to reproduce the defect.

3) Because of data mismatch



Login as ABC and click on Inbox in homepage of ABC

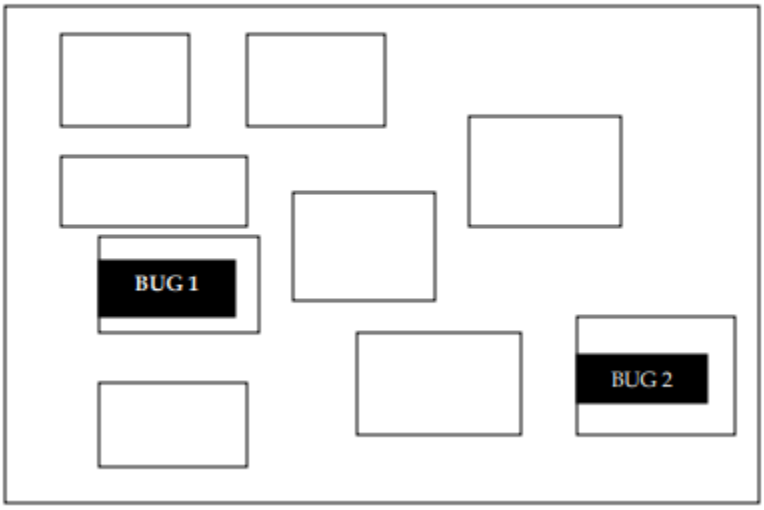
The above email application has been developed – the TE starts testing the above application – after testing and testing the above application using username ABC and valid password – there are almost 101 mails in ABC‟s inbox – now, again when the TE opens the application and goes to test the application – no mails are being displayed when he clicks in Inbox and instead he gets a blank page. So – the defect he has found is that – when the number of mails in Inbox exceeds 100 mails , he gets a blank page – so he prepares a defect report as shown in the box below,

1) Open the browser and enter the following URL www.email.com
2) Login using valid username and password
3) Click on Inbox
4) Blank page is displayed

The development team looks at the defect report – he logs in to the application using username XYZ and clicks on Inbox – no blank page is displayed and the Inbox is working fine

What happened here? – The TE should have mentioned in the defect report as “Login as ABC” or “Login to any mailbox which has more than 100 mails”. Thus, the developer does not find any defects and thus he mentions it as “not reproducible defect”.

4) Because of build mismatch



Developer develops build 1 and gives it for testing – in Build 1, there are 2 defects – bug 1 and bug 2 – TE while testing finds bug1 and reports it to development team – development team when they fix bug1, bug2 automatically gets fixed – development team is developing build 2 and gives it to testing team for testing – testing team just when it finishes testing build1 – he finds Bug 2 and reports it to development team – the development team however do not find any defect as they are testing build 2 – so they report back saying “bug is not reproducible”.

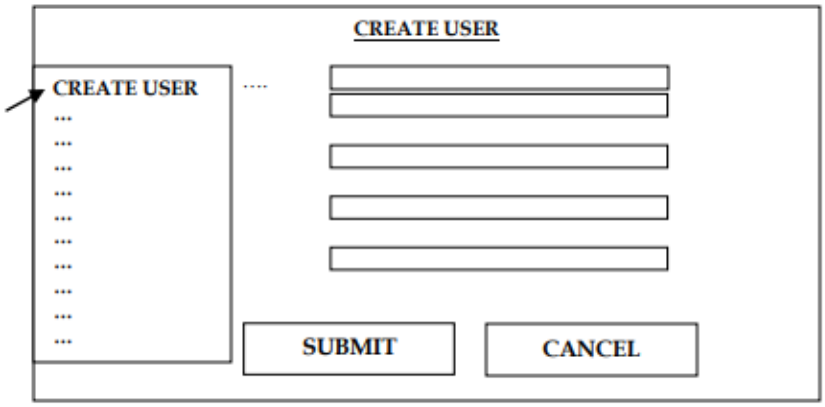
5) Because of inconsistent defects

To explain this, let us consider an example : When TE is testing an email application – he composes mail in User A and sends it to User B – he then logs out from user A and logs into User B and checks B"s inbox – but no mail is there in the Inbox – thus he finds a defect – now, just to confirm the defect before sending a defect report – the TE again logs into User A and

sends a mail to User B – he then logs out from User A and logs into User B and checks B‟s inbox – but, this time the mail is there in B‟s inbox!! Thus, we don‟t know when the defect comes and when the feature works fine. This is called an inconsistent defect.

REQUEST FOR ENHANCEMENT (RFE)

Test engineer finds a bug and sends it to the development team – when the development team looks at the report sent by the TE – They know it‟s a bug, but they say it's not a bug because it's not part of the requirements. Let us consider the example shown below,



In the above example, the TE is testing the above fields. After he enters all the data into the required fields, he realizes that he needs to clear all the data – but there is no CLEAR button – and he has to manually clear all the fields. He reports this as a bug – but the development team will look at the defect report and see that the clear button is not mentioned in the requirements – so they don‟t reject or close the bug – instead they reply back to TE saying that it is RFE.

Developers agree that it is a bug – but will say that the customer has not specified these details and hence will give it a name as RFE.

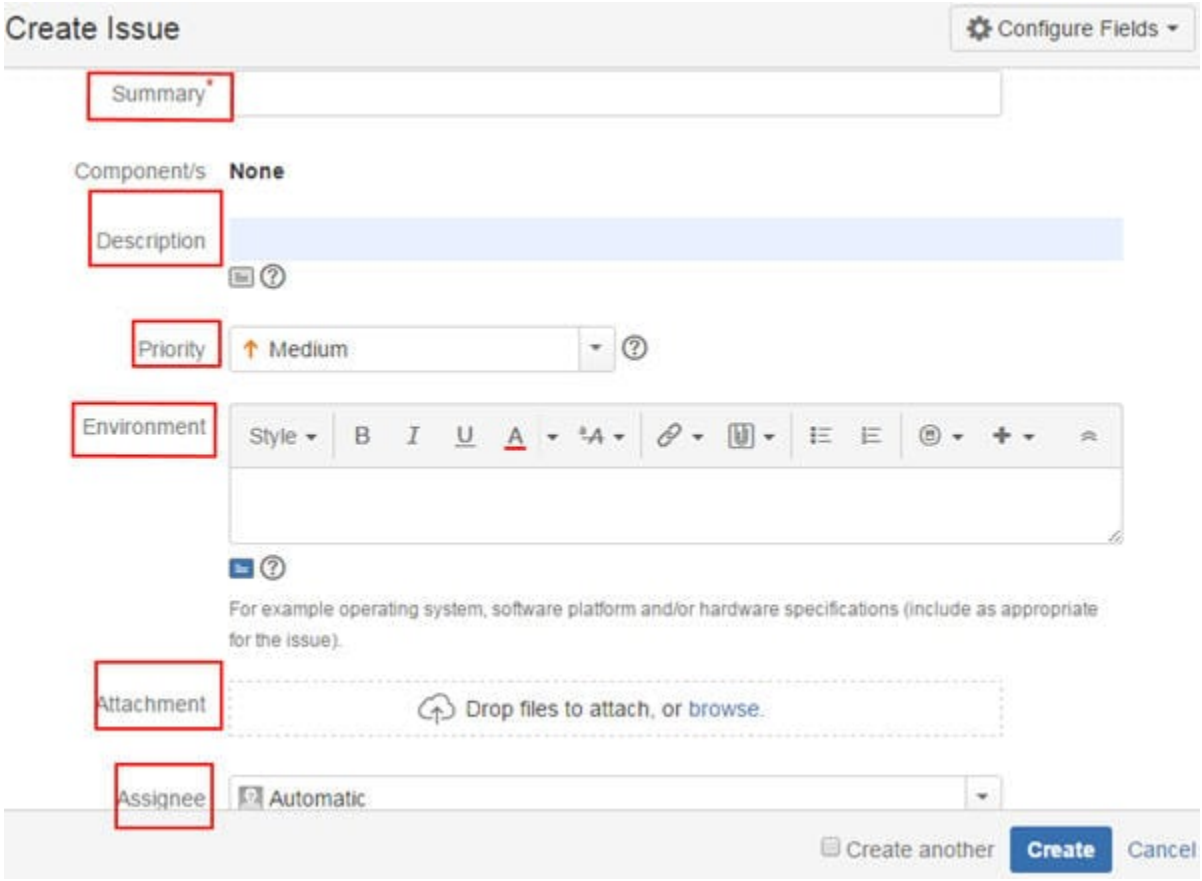
If a defect is known as RFE, then we can bill the customer. Otherwise, development team needs to fix the bug

Development teams always have a tendency to call a defect as RFE, so a TE needs to check the justification given by the development team – if it is valid, then he will accept it as a RFE – but if it is not, then he will respond to the development team with proper justification.

DEFECT REPORT / DEFECT TRACKING

Differentiate Error, Defect, and Failure?

- In the Development phase, if developers find any mismatch they call it an Error or mistake
- In Testing phase if Testers find any mismatch they call it as Defect or Bug or Fault
- In Production phase if Customers or End Users find any mismatch they call it as Failure



Refer https://instabug.com/blog/jira-bug-reporting-tool/

What is a Defect Report?

A document reporting on any flaw in a component or system that can cause the component or system to fail to perform its required function.

Defect ID – it is an unique number given to the defect

Test Case Name – whenever we find a defect, we send the defect report and not the test case to the developer. For defect tracking, we only track the defect report and not the test case. Test case is only for reference for the TE. We always only send the defect report whenever we catch a bug.
When we are doing ad-hoc testing – no test case is written for ad-hoc testing because they are "out of the box" testing scenarios – if we find a critical bug – then we convert that ad-hoc scenario into a test case and send it to the development team.

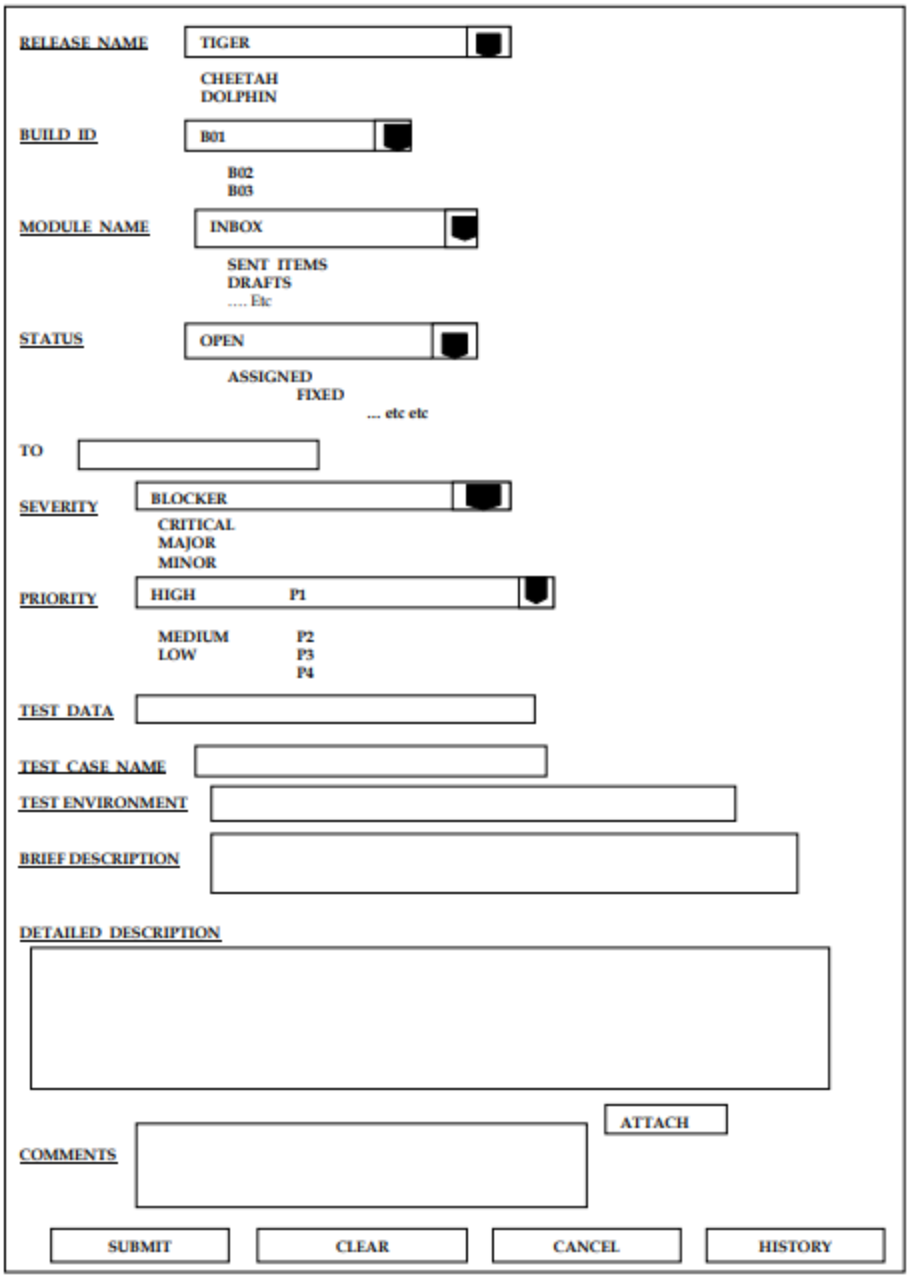
How to track a defect manually?

- 1) Find a defect
- 2) Ensure that it is not duplicate (i.e, verify it in Defect Repository)
- 3) Prepare defect report
- 4) Store it in defect repository
- 5) Send it to development team
- 6) Manage defect life cycle (i.e, keep updating the status)

There are 2 categories of tools,

1) Service based – In service based companies – they **might have** many projects of different clients – each project will have a different defect tracking tool

2) Product based – in product based companies – they use only one defect tracking tool.



When we click on log a defect link, we get the following page as shown above. (depends on the configuration)
We enter all the details and then when we click on submit button, the following happens,
It generates a bug ID automatically, ex – bugID 1578
It stores the data in the database
It automatically sends the mail to the Development Lead and also the copy of the mail goes to the TE himself

Development Lead opens the mail – he logs in to the tool using his username and password – then he gets the welcome page – he then enters the BugID into the defect id text box and clicks on search button When he clicks on search button – he gets the particular bug report. He looks at the report and after checking it‟s a valid bug, he then changes the status in the report to „assigned‟ and when he does that – he gets a TO text field as shown in the figure above – there he enters the email id of the concerned developer who has to fix that bug – and then when he clicks on submit button – whatever changes he has made is stored in the database – and the mail goes to the concerned TE, Developer and DL himself.

When the particular developer gets the the BugID, then he searches for the bug using the bugID and after seeing what‟s the bug – he goes to his module and fies the bug. When developer fixes the bug – he goes and changes the status to fixed and again he submits it. This is also saved in database. Then mail goes to TE, DL and to the developer himself. TE now opens the same bugID and checks the status whether it is fixed or not – if it is changed to fixed – then he re-tests the bug and if it is really fixed, then he goes to status and changes it to closed – if the bug is not fixed, then he changes the status to reopen. This comprises the entire defect life cycle.

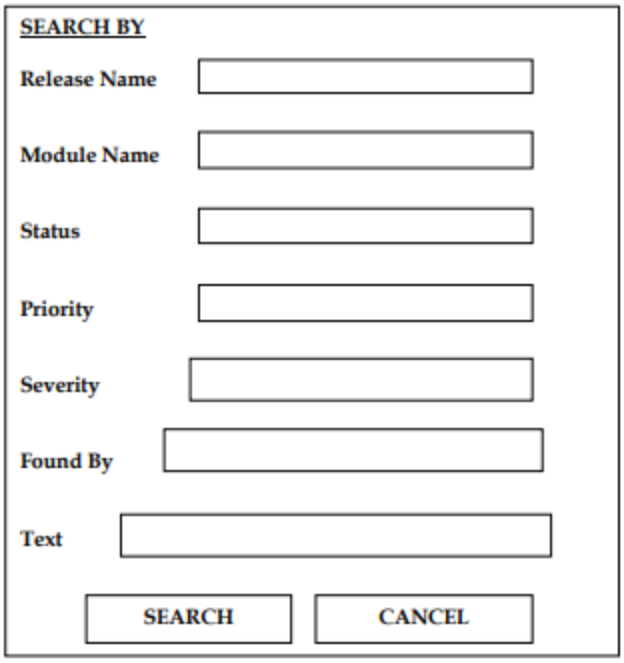
As soon as TE submits the bug report, then he or anybody cannot change two fields in the entire report – they are „brief description‟ and „detailed description‟.

If any changes are to be made in these fields – then they must be mentioned in the comments field. If a new defect is found within a old defect within a short period (say 10days), then instead of opening a new defect – he can open the old defect and login the new defect in the comments page.

How to check for duplicate bugs ?

- When the developer changes the status to duplicate, then the TE should check whether the previous bug & the sent bug is the same or not.
- To check whether its duplicate or not, click on advance search & get it confirmed if the bug is duplicate or not. If it‟s not duplicate, then TE should give proper justification.
- Click on Advanced Search (see figure next page)
- To avoid duplicate bugs, go to Advanced Search.

Whenever we catch a bug – Before logging the bug to the developer for fixing – first go & check whether it is logged before or not. To do so, click on Advanced Search & enter the data in the text field & click on Search. You will get the bug ID(S). if you enter „password‟ & search, then it will give you a different bug ID having password text. We must go & check it & log it for fixing if its not logged before. Defects found by the Testing team should never be closed just like that by the Development team. TE looks the product from the customer point of view – so if a developer says it‟s a minor bug, testing team always considers it as a major bug.



RESULT SHEET

Release Name	BugID	STATUS	Severity	Priority
TIGER	BugID 1578	OPEN	...	...
TIGER	BugID 1890	CLOSED	...	...
TIGER	BugID 1235	ASSIGNED	...	...

Blocker Defect

There are 2 types in blocker defect,

- **Major flow is not working –** Login or signup itself is not working in CitiBank application
- **Major feature is not working –** Login to CitiBank. Amount Transfer is not working

The defect report varies from company to company. But the following are the mandatory attributes of a defect report in all companies,

- Defect ID
- Severity

- Priority

Defect Density = (Total no. of defects found – total defects fixed) / (Total no. of TC(s) executed

Things to Include While Creating Defect Report:

The information provided in the defect reports is helpful to all stakeholders as well as other tester who might face a similar problem in the future. Therefore, to ensure that the defect report is perfect and conveys all the necessary information, it is essential to include some important details in it, which can make the process of understanding the defect report easy. The elements that should be included in a defect report are:

1. **Testing Condition:** There should be a summary of the condition in which the testing was executed. For example, if the tester is testing a web app, then the report should include information on the type of browser used by the tester.
2. **Steps Taken to Resolve Issues:** The report should include a clear breakdown of exactly what was done to detect and resolve the issues. Furthermore, information regarding the used methods, techniques and tools should be included, along with what the outcome of each step was, and whether that outcome was expected or not.
3. **Steps Indicating Successful Defect Duplication:** An indication should be provided whether or not the tester was able to reproduce the issue, along with the details of how many times did the issue reoccurred during the whole testing process.
4. **Any additional details:** Any other information that can improve the quality of the report should be included, such as charts, screenshots and more.

What Is Reproducible Defect?

If a Defect is appearing every time (in every execution) then that Defect is a producible defect, we can provide steps for locating it.

What Is Not Reproducible Defect?

If a Defect is sometimes only appearing then that Defect is not a producible defect, we have to provide proof (snap shots or database dumps) for locating it.

What are the different types of Status of Defects?

- New: Tester provides new status while Reporting (for the first time)

- Open: Developer / Dev lead /DTT opens the Defect
- Rejected: Developer / Dev lead /DTT rejects if the defect is invalid or defect is duplicate.
- Fixed: Developer provides fixed status after fixing the defect
- Deferred: Developer provides this status due to time etc…
- Closed: Tester provides closed status after performing confirmation Testing
- Re-open: Tester Re-opens the defect with valid reasons and proofs

What Is Defect Masking?

An occurrence in which one defect prevents the detection of another.

Give A Sample Defect Report Template?

- **Defect ID**: Unique No
- **Description**: Summary of the defect
- **Feature**: Module / Function / Service , in these module TE found the defect
- **Test Case Name**: Corresponding failing test condition
- **Reproducible** (Yes / No): Yes -> Every time defect appears during test execution

No -> Rarely defect appears

If **Yes**, attach test procedure:

If **No**, attach snapshot & strong reasons:

- **Status**: New / Reopen
- **Severity**: Seriousness of defect w.r.t functionality (high / medium / low)
- **Priority**: Importance of the defect w.r.t customers (high / medium / low)
- **Reported bug**: Name of Test Engineer
- **Reported on**: Date of submission
- **Assign to**: Name of the responsible person in development team -> PM
- **Build Version ID**: In which build, Test Engineer fount the defect
- **Suggested fix** (Optional): Tester tries to produce suggestion to solve this defect
- **Resolved by**: Developer name
- **Resolved on**: Date of solving By Developers
- **Resolution type**: check out in next page
- **Approved by**: Signature of Project Manager (PM)

Note: Defect Report Template may vary from one company to another and one project to another

Why does software have bugs ?

- Mis-communication OR No communication -> as to specific of what an application should or shouldn‟t do(the application‟s requirements)
- Software complexity
- Programming errors
- Changing requirements
- Time pressure

When do we stop the testing ?

- When the time span is less, then we test the important features & we stop it
- When the functionality of the application is stable
- When the basic feature itself is not working correctly
- Budget
- In the last 2 days of release, the code will be freezed. When the code is freezed, the defect found will be fixed in later stages.

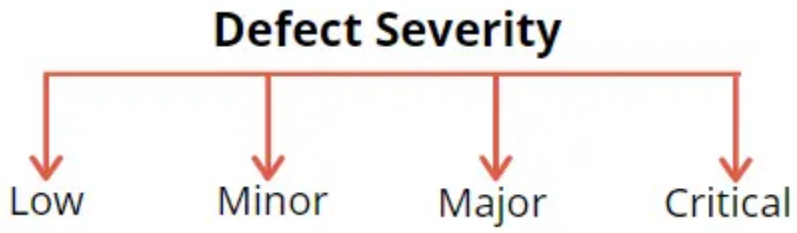
Severity and Priority of a Defect

Bug Severity:

Severity defines how impactful can a bug be to the system. Defect Severity means how badly the defect has affected the application's functionality.

SEVERITY of a Bug

Severity is the impact of the bug on a customer"s business.



Critical – A major issue where a large piece of functionality or major system component is completely broken. There is no work around & testing cannot continue.

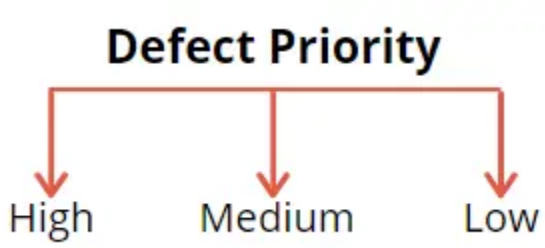
Major – A major issue where a large piece of functionality or major system component is not working properly. There is a work around, however & testing can continue.

Minor – A minor issue that imposes some loss of functionality, but for which there is an acceptable & easily reproducible workaround. Testing can proceed without interruption.

LOW – As the name suggests, these bugs do not harm the system in any critical way. Although they are harmless, yet valid and need to be removed.

Bug Priority

Priority defines the urgency of fixing the bug. Priority is considered from the customer's point of view. Priority indicates how soon the defect needs to be fixed by the developer. Priority is set by the product manager/customer and it determines the time frame given to the developer to fix the bug.



PRIORITY of a Bug

- It is the importance to fix the bug (OR) how soon the defect should be fixed (OR) which are the defects to be fixed first.
- Generally, severity is assigned by Tester / Test Lead & priority is assigned by Developer/Team Lead/Project Lead.

HIGH – Such defects need immediate attention as it might lead to complete failure of the system. The earliest resolution of such defects should be conducted.
Delay in finding the solution might affect the system adversely. Often leading to even more complications, **This has a major impact on the customer. This must be fixed immediately.**

MEDIUM – Such bugs do not affect the working of the system. These can be solved simultaneously with the testing as well as the design phase.These bugs do need to be fixed but do not need any immediate attention. **This has a major impact on the customer. The problem should be fixed before release of the current version in development**

LOW – These bugs are at the lowest priority. These are fixed once the developer is done with the high and medium-priority bugs. **This has a minor impact on the customer. The flow should be fixed if there is time, but it can be deferred with the next release.**

High Priority and High Severity Examples

1. The products added to the cart of an e-commerce website are not visible on the payment page.
2. The login button of the application is not working./ The user enters the username and password and clicks on the ‘Login’ button.
3. You log in to your amazon.com account, add items to the cart and click the “Proceed to Checkout” button. You make the payment and the system crashes. This defect makes the whole buying functionality unusable and so the severity is high.

High Priority and Low Severity Examples

1. The logo of the company’s welcome page is distorted.
2. The action buttons are not visually appealing or the information on the page appears hazy.
3. Suppose that in the amazon.com website, the logo is displayed as ”amazn.com” with the letter “o” missing. This defect does not affect the buying/selling or any other functionality in any way. So, the severity of this defect is low. But, a mistake in the company logo affects the brand identity and impacts the user experience. So, the defect is of high priority.
4. Suppose if the Flipkart logo is misspelled as Flipkart. That time it directly impacts the online business for the Flipkart company. People will think it’s not a genuine product and they won’t buy it. Business impact is huge. So it became a very high priority issue. But for developers fixing this issue is not that difficult. It is not even breaking any workflow also. So the severity is very low.
5. company logo put opposite. so this is not a functionality impact but highly business impact.

Low Priority and High Severity Examples

1. If the application is crashing on passing very large input for processing (which is very rarely done)
2. There are some buttons on the website which are overlapping. Although clickable, are creating a fuss.
3. Web page not found when user clicks on a link (user's does not visit that page generally)

Low Priority and Low Severity Examples

1. A spelling mistake on the page of the site which is not frequently visited.
2. The color of any text does not match the theme of the website.
3. Any cosmetic or spelling issues which is within a paragraph or in the report

Who decides the Severity and Priority of a Defect?

The organization decides the standards regarding who sets the priority and severity of a defect. However, in most cases, the severity type of a defect is set by the tester based on the product functionality and the written test cases. The priority is decided by the product manager based on customer requirements.

Difference between Priority vs Severity

Priority indicates how quickly the bug should be fixed.	Defined by the impact on the application's functionality.
Priority is based on the results of technical meetings and directions received from the project manager or client.	Severity is decided by the Test Engineer or QA team.
Priority is related to scheduling to resolve the problem.	Severity is related to the quality standard.
Its value doesn't change from time to time.	Its value changes from time to time.

For Jira tutorial you can refer to the below link https://www.youtube.com/watch?v=HzJzB0nQYIk