
BUG
HUNTERS
THINK LIKE DEVELOPER
HUNT IT LIKE TESTER

Think Like Developers ..Hunt It like Tester

Introduction to software testing

Code Quality & Unit Testing for Software Craftsmanship

Testing always gives confidence in the developed software testing

Why We Are Here Today

Agenda

- Introduction
- Why…Idea
- Code quality
- Unit testing
- Role of testing
- References

Why…The idea

- **Multinationals are driven by**
  - Diversity
  - Distributed teams
  - Collaborative teams

Code Quality

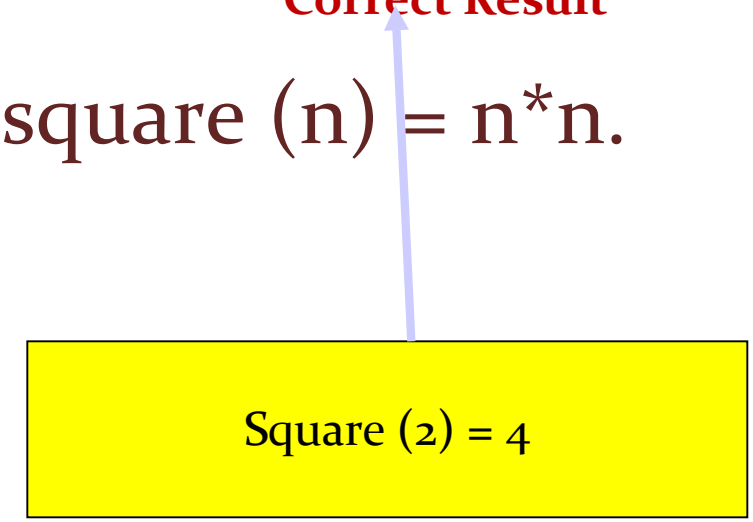
Why we care about writing quality code?

Let's do Small development task

Function that calculate square of int numbers

For any integer n, square (n) = n*n.

```
int square (int x)
{
    return x*2;
}
```



Developer is Happy ☺

Code is moved to Testing phase

For any integer n, square (n) = n*n.

```
int square (int x)
{
    return x*2;
}
```

Failure

Square (3) = 6

Bug

Let's do another Small development task

Function that takes array of int and display # of Zeros in it

numZero([0,5,0])=1

```
public static int numZero (int[] x) {
// Effects: if x == null throw NullPointerException
// else return the number of occurrences of 0 in x
  int count = 0;
  for (int i = 1; i < x.length; i++) {
    if (x[i] == 0) {
      count++;
    }
  }
  return count;
}
```

Bug

Failure

(numZero[4,5,0])=1

Correct Result

Actual Result != Expected Result

We caught bugs !!

Common SW Bugs

Real SW Business Examples

Examples Of Common SW Bugs (2)

Pepsi - $42 Billion Error

- In May 1992, Pepsi ran a promotion in the Philippines. It told customers they could win a million pesos (approx. $40,000) if they bought a bottle of Pepsi and found number 349 stamped on the underside of the bottle cap. Unfortunately, ***due to a software error***, 800,000 bottle caps were produced with number 349 instead of one, which was an equivalent of $42 billion in prize money. It cost the company dearly as some people pursued their claims through the courts and Pepsi paid out millions of dollars in compensation.
- Now, does it matter if there are mistakes in what we do?
- Does it matter if we don't find some of those flaws?
- We know that some of our mistakes do not matter, and some are very important

Examples Of Common SW Bugs (2)

The Lion King Animated Storybook

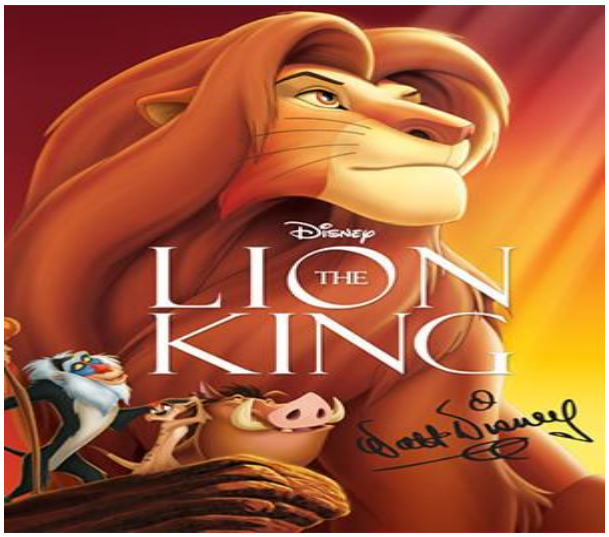
Disney's first multimedia CD-ROM game for kids.

Released at Christmas season.

26th December..... Customer Support's Nightmare.

The CD was testing only for specific PC platform.

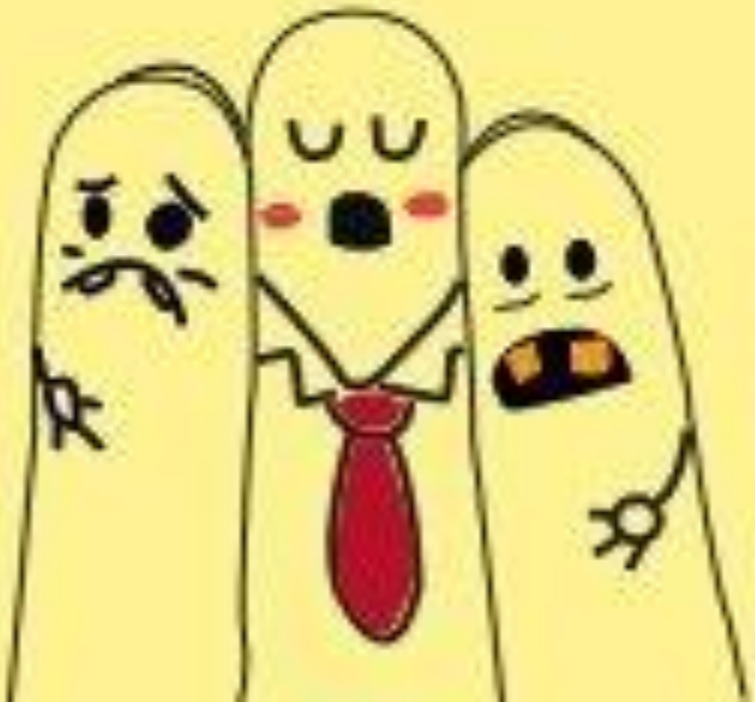
It failed on many popular PC operating system.



Because Bugs Costs A lot ☹

Beginning of

Difficult Times



Error - bug- Failure

A Developer makes an *error* ...

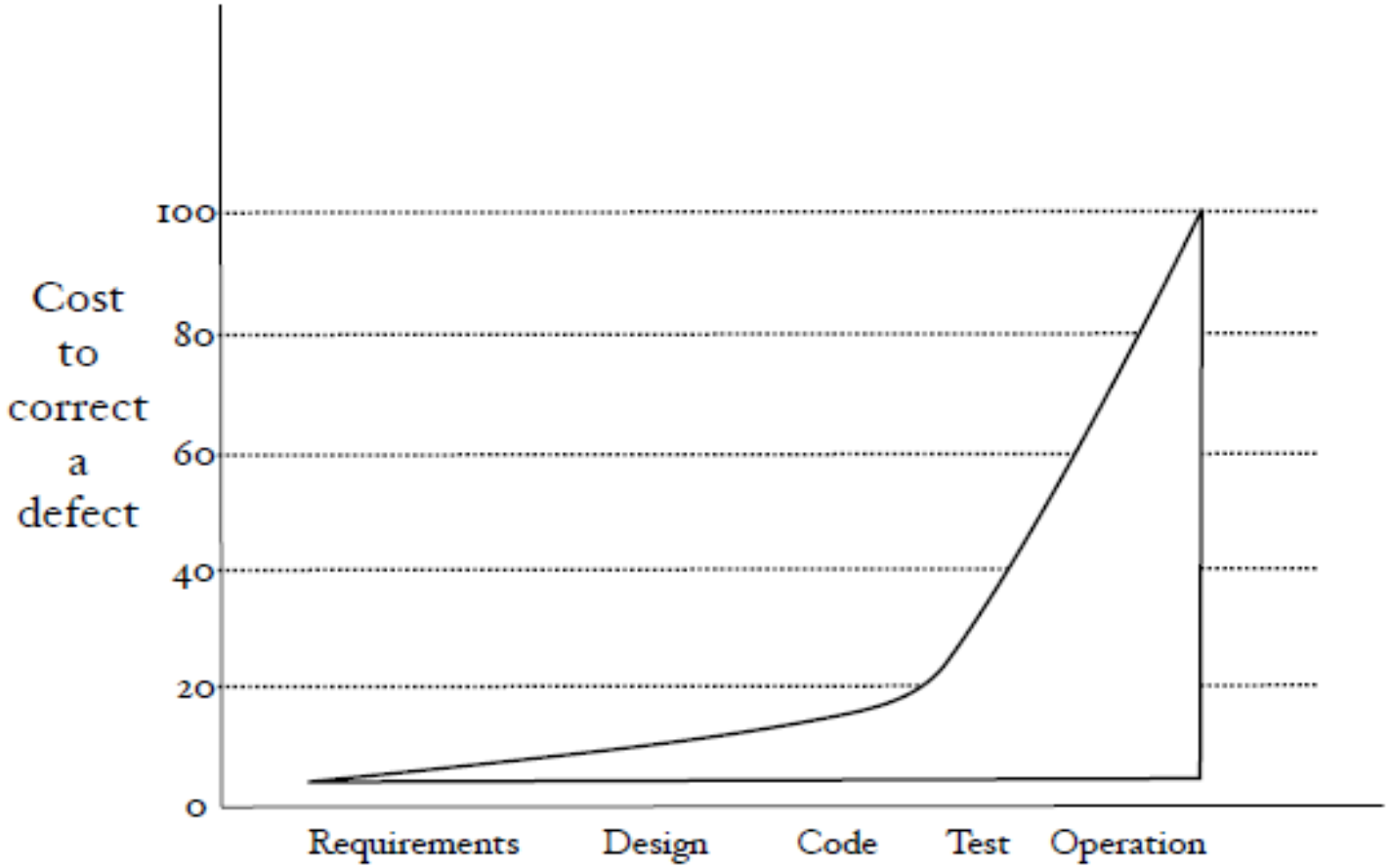
... that creates a *bug* in the software ...

... that can cause a *failure* in operation

Causes of software errors

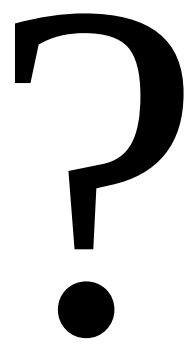
- Faulty requirements definition
- Client-developer communication failures
- Deviations from software requirements
- Logical design errors
- Coding errors
- Non-compliance with documentation and coding instructions
- shortcomings of the testing process
- documentation errors

Cost of Defect



[BOEH01]

How to write Quality Code?



It is all about writing and unit testing your code for the sake of….

Reducing Bugs

Lower Maintenance Effort

Ugly code

```
Ext.define('AM.store.fund', {
extend: 'Ext.data.Store',
model: 'AM.model.fund',
requires: [
'Ext.data.proxy.JsonP'
],
autoLoad: true,

proxy : {
type: 'jsonp',
//callbackKey: 'callback',
url: 'http://localhost:8090/spring-rest-sample/rest/funds/',
headers: {
'Accept': 'application/json'
},
reader: {
type: 'json',
root: 'data',
successProperty: 'success'
},
afterRequest: function(req, res) {
console.log("Ahoy!", req.operation.response);
}

},
```

Ugliest code

```
var Ext=Ext||{};if(!Ext.Direct){Ext.Direct={}}if(!Ext.Toolbar){Ext.Too
main={}}if(!Ext.button){Ext.button={}}if(!Ext.chart){Ext.chart={}}if(
art.series={}}if(!Ext.chart.theme){Ext.chart.theme={}}if(!Ext.containe
t.data={}}if(!Ext.data.association){Ext.data.association={}}if(!Ext.da
={}}if(!Ext.data.reader){Ext.data.reader={}}if(!Ext.data.writer){Ext.d
={}}if(!Ext.dom){Ext.dom={}}if(!Ext.draw){Ext.draw={}}if(!Ext.draw.eng
rm){Ext.form={}}if(!Ext.form.Action){Ext.form.Action={}}if(!Ext.form.a
}}if(!Ext.fx){Ext.fx={}}if(!Ext.fx.target){Ext.fx.target={}}if(!Ext.gr
t.grid.feature){Ext.grid.feature={}}if(!Ext.grid.header){Ext.grid.head
lugin){Ext.grid.plugin={}}if(!Ext.grid.property){Ext.grid.property={}
layout.boxOverflow={}}if(!Ext.layout.component){Ext.layout.component=
}}if(!Ext.layout.container){Ext.layout.container={}}if(!Ext.layout.con
ontainer.boxOverflow){Ext.layout.container.boxOverflow={}}if(!Ext.list
el={}}if(!Ext.perf){Ext.perf={}}if(!Ext.picker){Ext.picker={}}if(!Ext.
tton){Ext.rtl.button={}}if(!Ext.rtl.dd){Ext.rtl.dd={}}if(!Ext.rtl.dom)
form.field){Ext.rtl.form.field={}}if(!Ext.rtl.grid){Ext.rtl.grid={}}if
.plugin){Ext.rtl.grid.plugin={}}if(!Ext.rtl.layout){Ext.rtl.layout={}
Ext.rtl.layout.component.field){Ext.rtl.layout.component.field={}}if(
rtl.layout.container.boxOverflow){Ext.rtl.layout.container.boxOverflow
xt.rtl.resizer={}}if(!Ext.rtl.selection){Ext.rtl.selection={}}if(!Ext.
if(!Ext.rtl.tip){Ext.rtl.tip={}}if(!Ext.rtl.tree){Ext.rtl.tree={}}if(
}}if(!Ext.selection){Ext.selection={}}if(!Ext.slider){Ext.slider={}}if
{Ext.tip={}}if(!Ext.toolbar){Ext.toolbar={}}if(!Ext.tree){Ext.tree={}
={}}if(!Ext.ux){Ext.ux={}}if(!Ext.ux.form){Ext.ux.form={}}if(!Ext.view
=[],m=["constructor","toString","valueOf","toLocaleString"],k={},p={},
;for(r=m.length;r-->0;){q=(1<<r);p[k[q]=m[r]]=q}for(r in p){b|=p[r]}b=
r("config").fn;for(h in c){if(c.hasOwnProperty(h)){l.push(h)}}j.derive
numerableMembers,v=y.prototype,t,w,s,q;if(!u){return}for(t in u){q=u[t
dentityFn){v[t]=w=q;w.$owner=y;w.$name=t}else{v[t]=q}}for(s=1;r;s<<=1)
```

Beautiful code

```
// @todo: delete this file after testing

/**
 * # Defines CompanyStore - Only for testing, unrelated to the main app
 *
 *
 * @author Jozef Sakalos, Saki
 * @date 19.8.2013
 * @copyright (c) 2013, Jozef Sakalos, Saki
 * @license This file is proprietary and it is only meant to be
 * run as a part of Omni8.net blog application.
 * All other uses (reading, copying, reverse engineering
 * to name a few) are prohibited.
 */
Ext.define('Od.store.CompanyStore',{
    extend:'Ext.data.Store'
    ,requires:['Od.model.CompanyModel']
    ,model:'Od.model.CompanyModel'
    ,pageSize:15
    ,remoteSort:true
    ,remoteFilter:true

    /**
     * @return {Boolean} true if the store is dirty, false otherwise
     */
    ,isDirty:function() {
        var dirty = false;
        this.each(function(r){
            dirty = dirty || r.dirty || r.phantom;
        });
        dirty = dirty || this.getRemovedRecords().length;
        return dirty;
    } // eo function isDirty
});

// eof
```

No more spaghetti code

Make it Easier to Extend Good Code

Give your client the warm and fuzzy feeling. Quality Code = Trust

“Testing always Gives Confidence In The Developed Software.”

Two perspectives on code quality

Does the software correctly implements requirements? Is it easy to use? Does it crash?

How is the code organized? Where is <some> functionality implemented?

```
void CruiseControl_init(_C_CruiseControl * _C_)
{
    CruiseSpeedMgt_init(&(_C_->_C0_CruiseSpeedMgt));
    CruiseStateMgt_init(&(_C_->_C3_CruiseStateMgt));
    (_C_->_M_condact_0) = true;
    ThrottleCmd_init(&(_C_->_C4_ThrottleCmd));
    (_C_->_M_init) = true;
}

/* ========= */
/* MAIN NODE */
/* ========= */

void CruiseControl(_C_CruiseContr
{
    bool BrakePressed;
    bool AcceleratorPressed;
    bool SpeedOutOffLimits;
    bool _L19;
/*#code for node CruiseControl
/* call to node not expanded Dete
    (_C_->_Cn_DetectPedalsPressed._I
    (_C_->_Cn_DetectPedalsPressed._I1_Accelerator) =
    DetectPedalsPressed(&(_C_->_Cn_DetectPedalsPressed)
    BrakePressed = (_C_->_Cn_DetectPedalsPressed._O0_Br
    AcceleratorPressed =
        (_C_->_Cn_DetectPedalsPressed._O1_AcceleratorPr
/* call to node not expanded DetectSpeedLimits */
    (_C_->_Cn_DetectSpeedLimits._I0_speed) = (_C_->_I8_
    DetectSpeedLimits(&(_C_->_Cn_DetectSpeedLimits));
    SpeedOutOffLimits = (_C_->_Cn_DetectSpeedLimits._O0_
/* call to node not expanded CruiseStateMgt */
```

External (user's): *what?*

Internal (programmer's): *how?*

Components of Code Quality

- Writing according to coding standards
- Unit testing your code

Coding Standards

- Pleasant to read and easy to grasp
- Abide by standards
- Modular
- Almost bug-free
- Finished
- Clean
- Documented
- working

Unit Testing

- Test individual units of code
- Isolate each part
- Show that the individual parts are correct
- Document your code
- Executed within a framework(IDE)

Is Unit Testing Enough for Software quality

?

IEEE Definition of "Software Quality"

1. The degree to which a system, component, or process meets specified requirements.

2. The degree to which a system, component, or process meets customer or user needs or expectations.

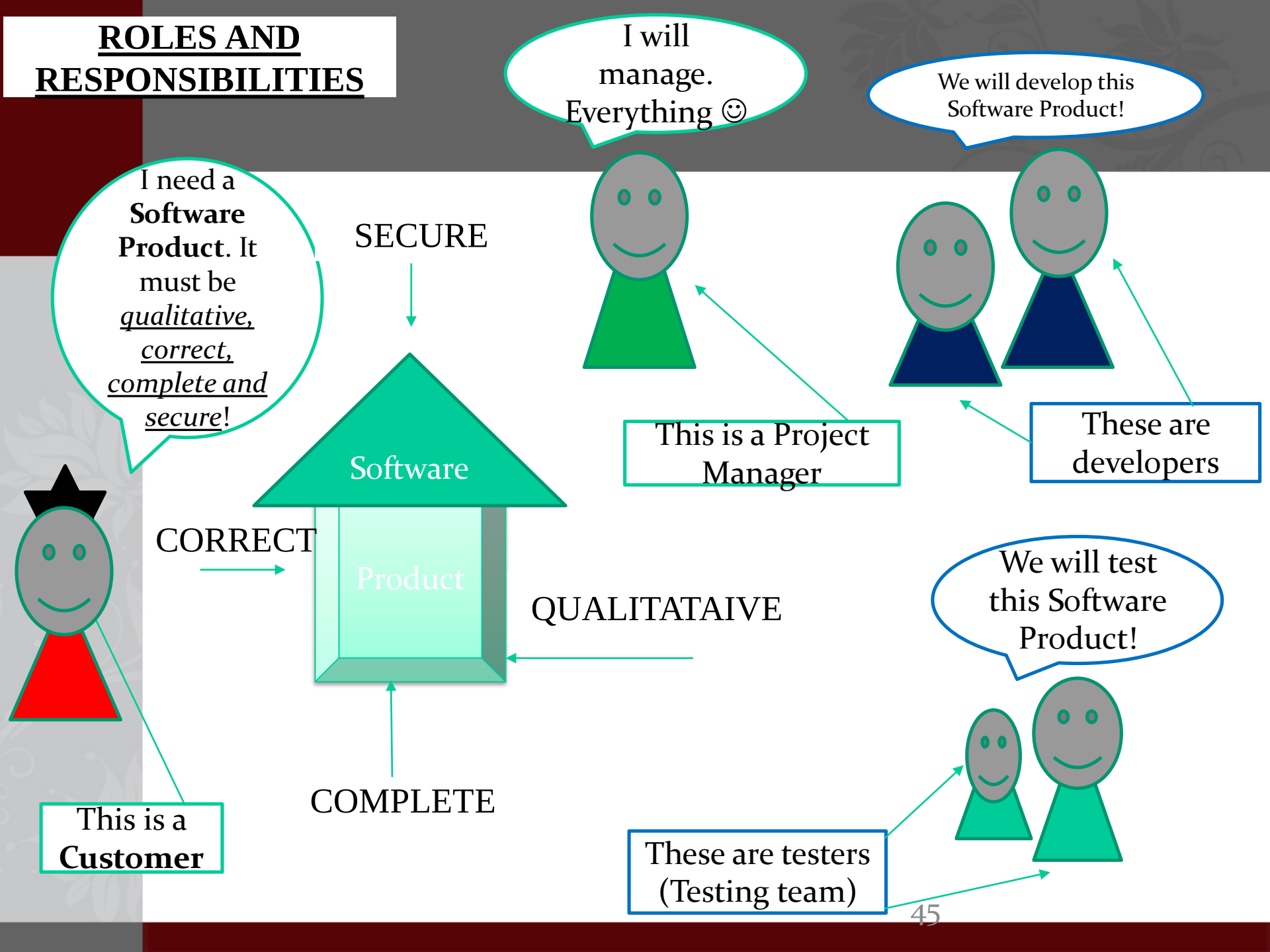
We need another detailed oriented Eye

We need testing team

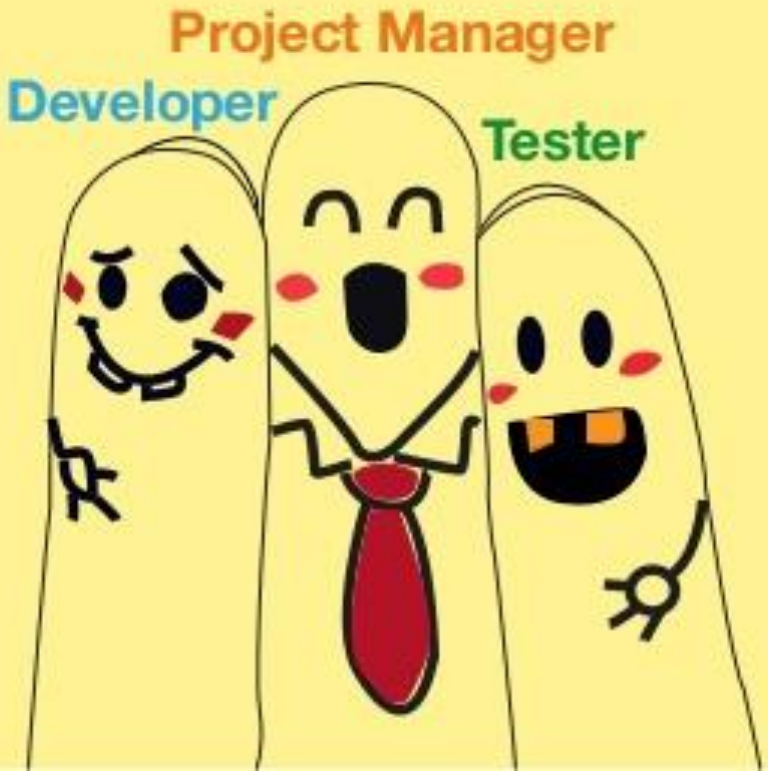
Testers Roles

- Find Bugs after unit testing
- Complete testing at different levels
  - Integration testing
  - System Testing
  - Acceptance testing (Business needs)
- Do all testing types
  - Functional Testing (Requirements)
  - Non Functional
    - Performance Testing
    - Security Testing…

ROLES AND RESPONSIBILITIES



They were all very good friends.

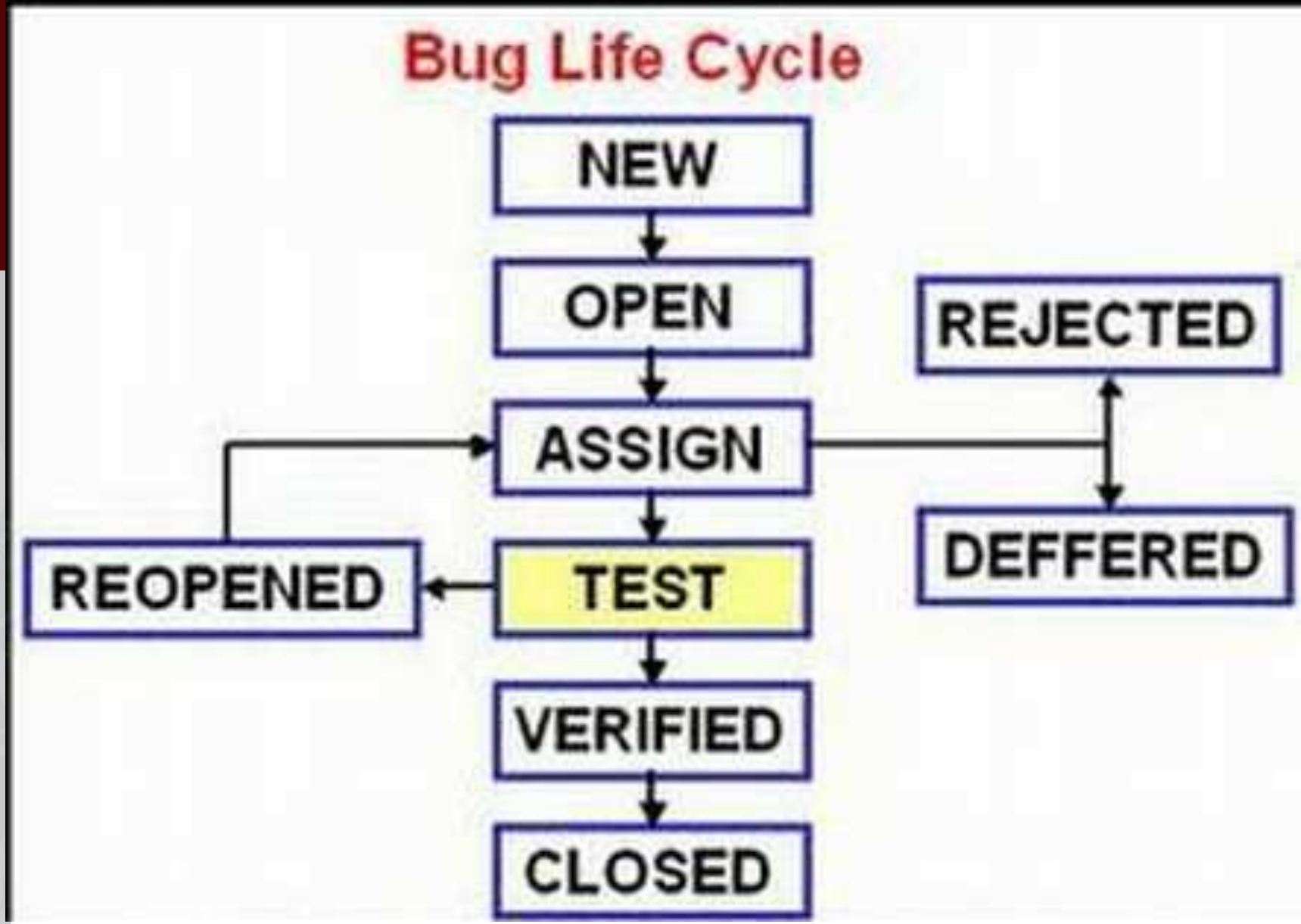


Testing & Bugs

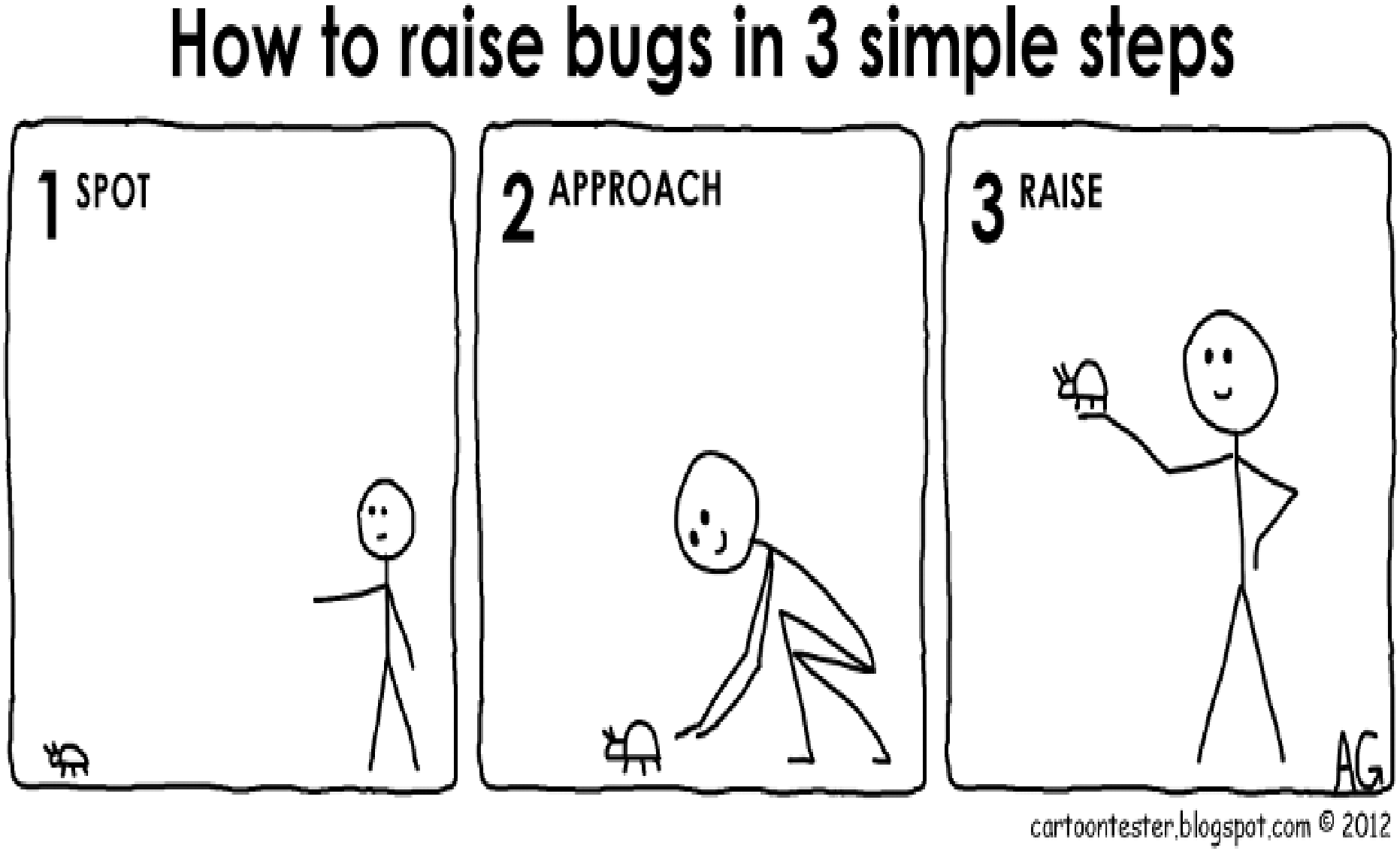
- Testing is all about find Bugs
- The earlier we find them

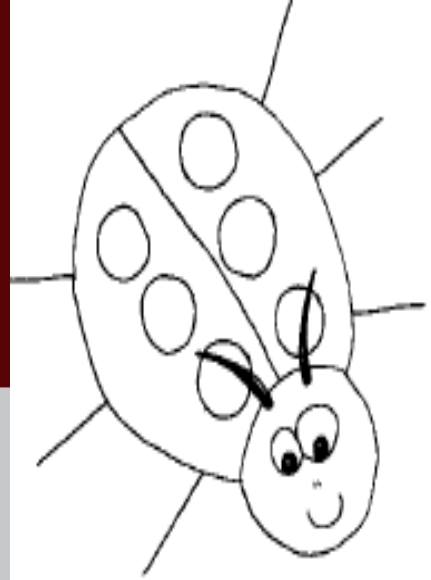
  ...the lower will be the cost to fix them

  ....the higher will be the quality of SW


Bug Life Cycle
NEW
OPEN
ASSIGN
TEST
REOPENED
VERIFIED
CLOSED
REJECTED
DEFFERED

Raise Bugs

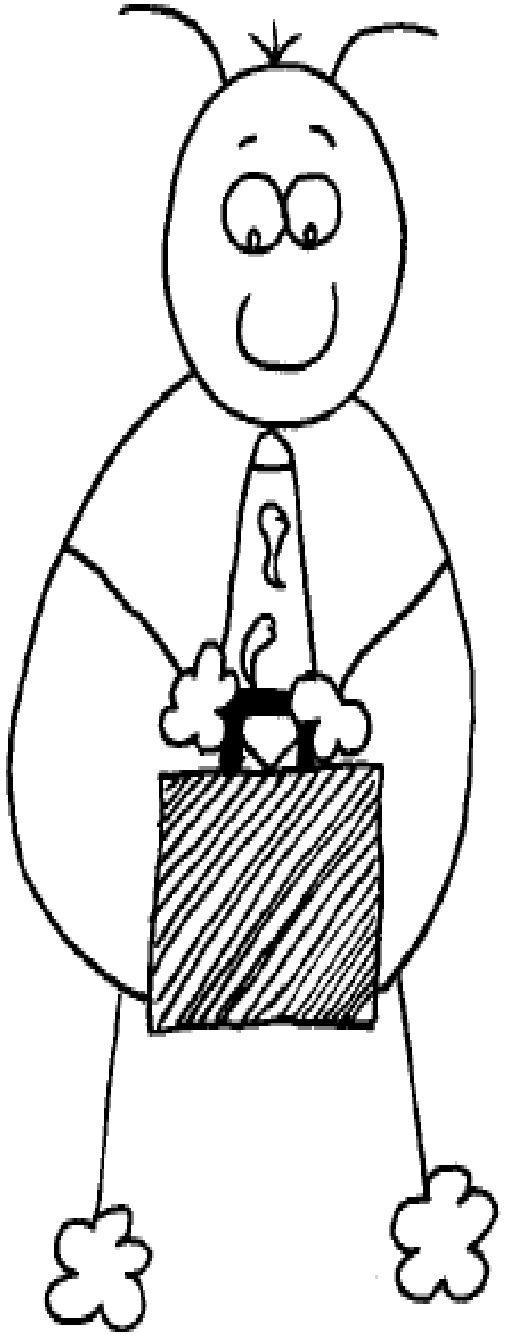

How to raise bugs in 3 simple steps
1 SPOT
2 APPROACH
3 RAISE
AG
cartoontester.blogspot.com © 2012



Bug Priority

- How important is it?
  - Urgent
  - Not Urgent

- Define Priority Scheme
  - P1
    - _____
  - P2
    - _____
  - P3
    - _____

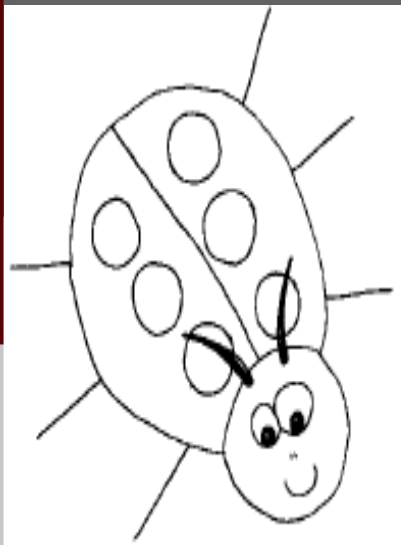


Priority

P1 - Fix it now

P2 - Fix it later

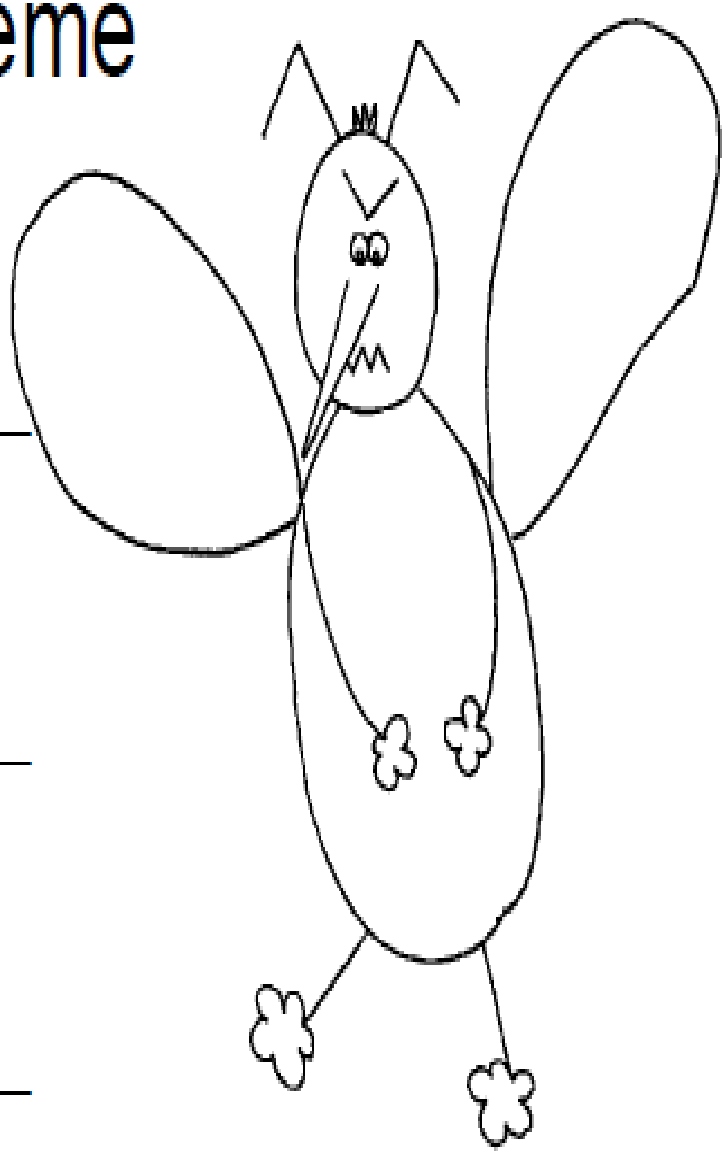
P3 - Don't fix it



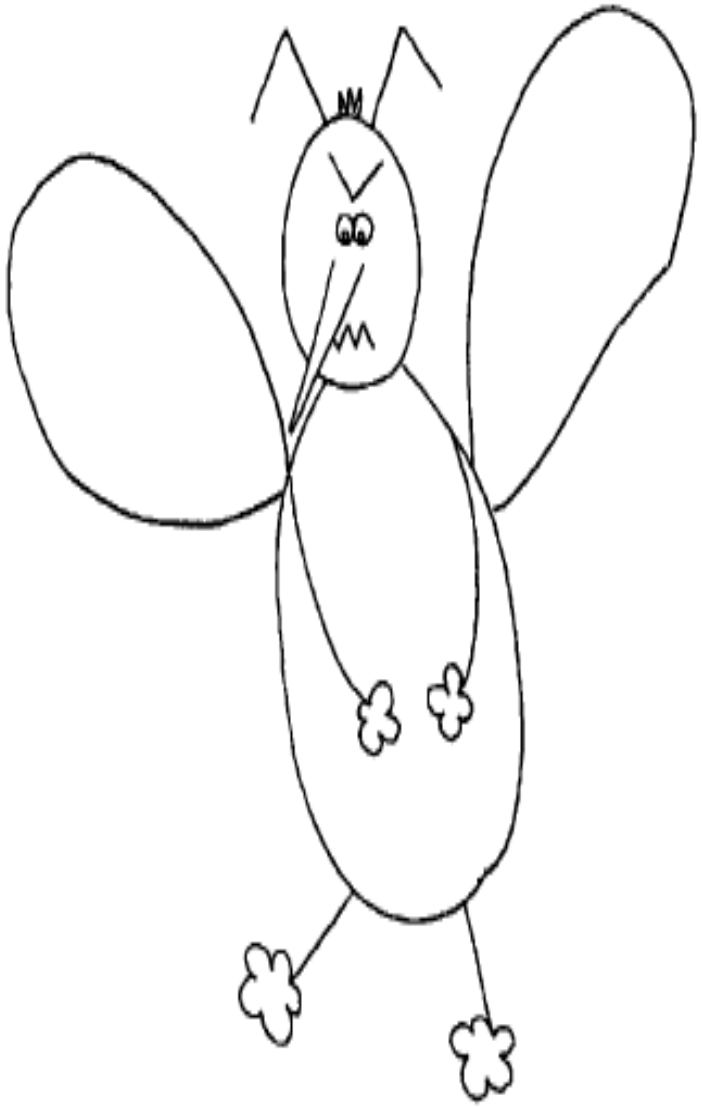
Bug Severity

- How much damage it causes
  - severe
  - not severe

- Define Severity Scheme
  - S1
    - _____
  - S2
    - _____
  - S3
    - _____



- Severity Scheme
  - S1
    - Unusable no straight forward work around
  - S2
    - Work around possible
  - S3
    - Cosmetic

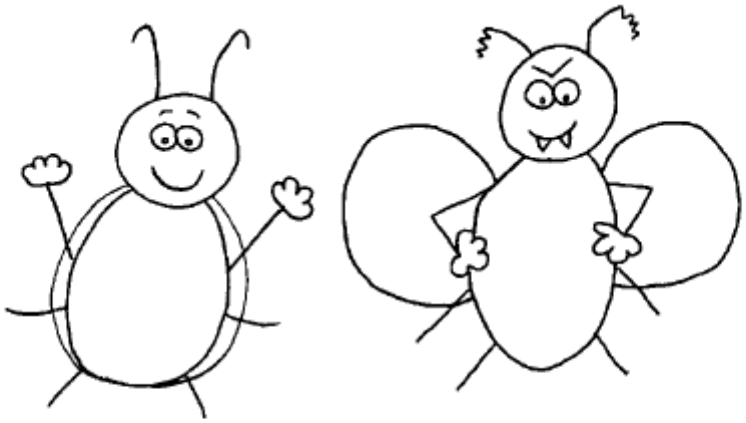


Severity

S1 - Show Stopper

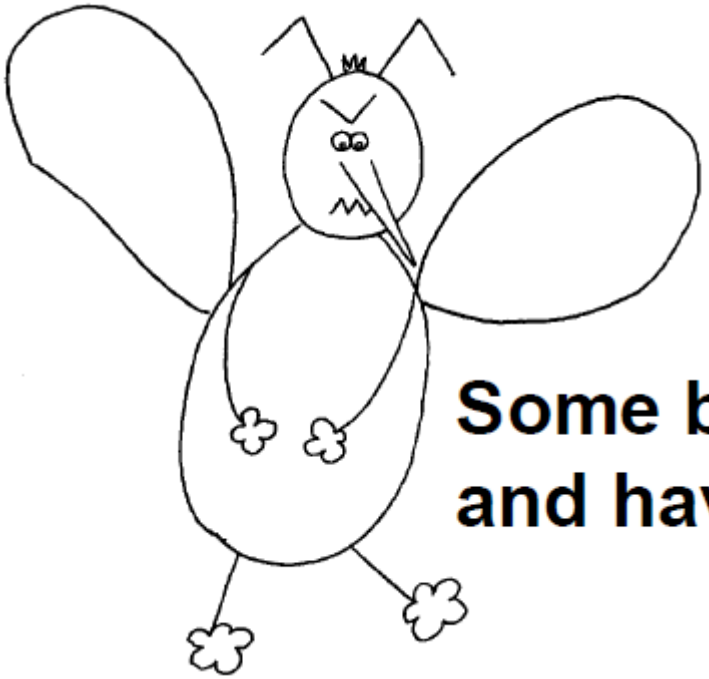
S2 - Work around

S3 - Cosmetic

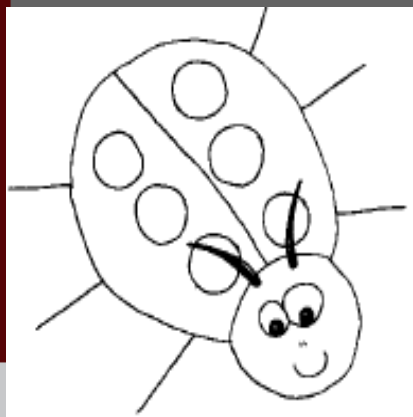


Bugs are not Good or Bad

Some bugs are important and have a high priority!

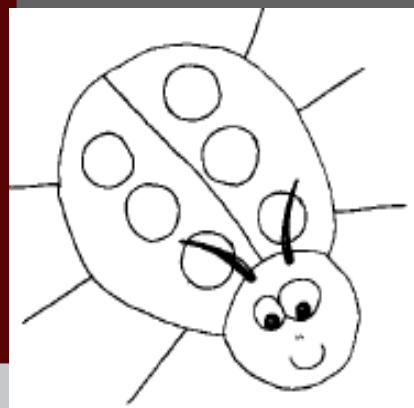


Some bugs are dangerous and have a high severity!



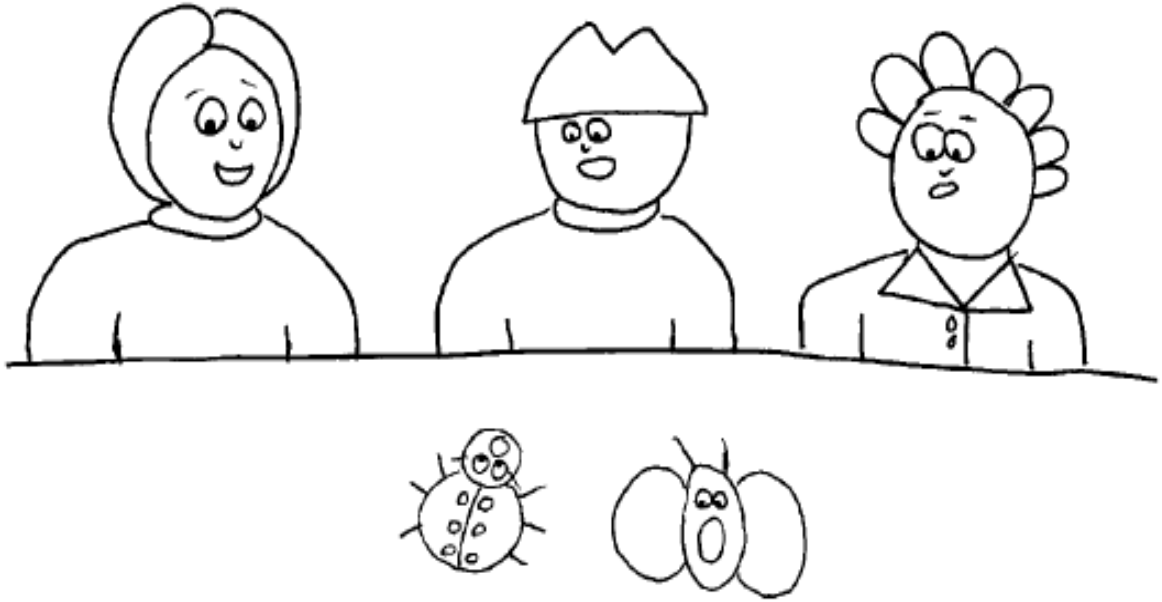
Bug Quadrants

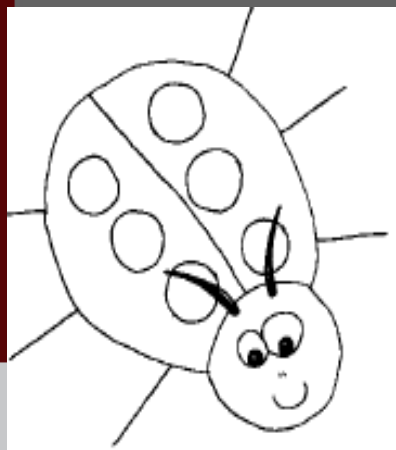
Urgent Severe	Urgent Not Severe
Not Urgent Severe	Not Urgent Not Severe



Finished?

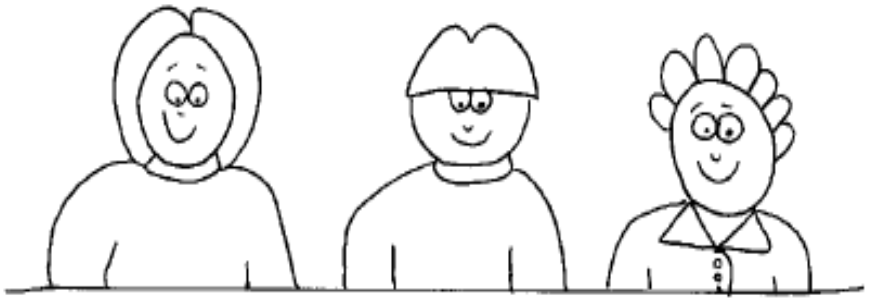
- How do you know you are finished?

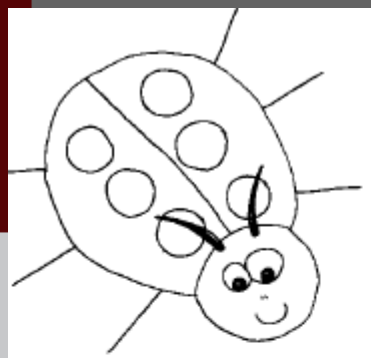




You know you are finished when …

- … the only bugs left are the ones that are acceptable (based on objective SQA input) …

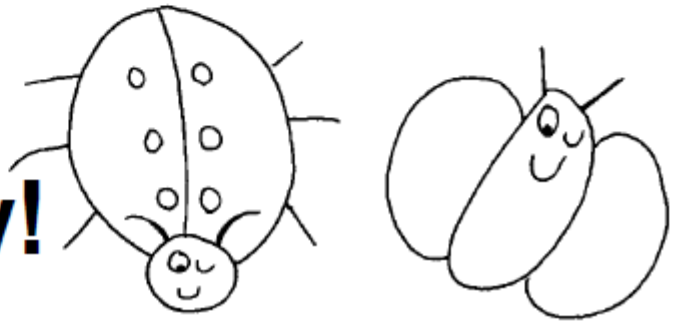




You know you are finished when …

- … the only bugs left are the ones that are acceptable (based on objective or was it subjective? input) …

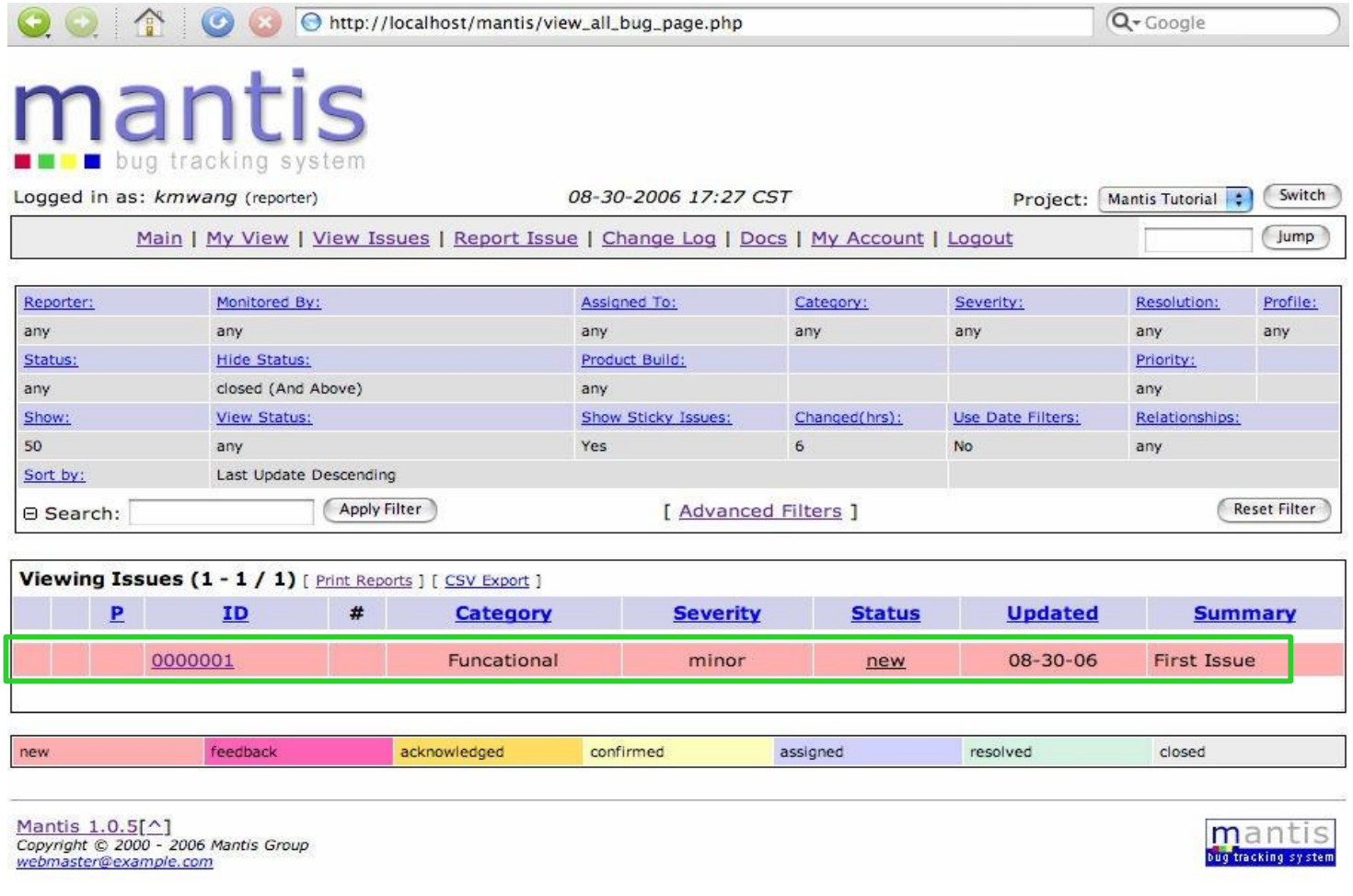
At least for now!



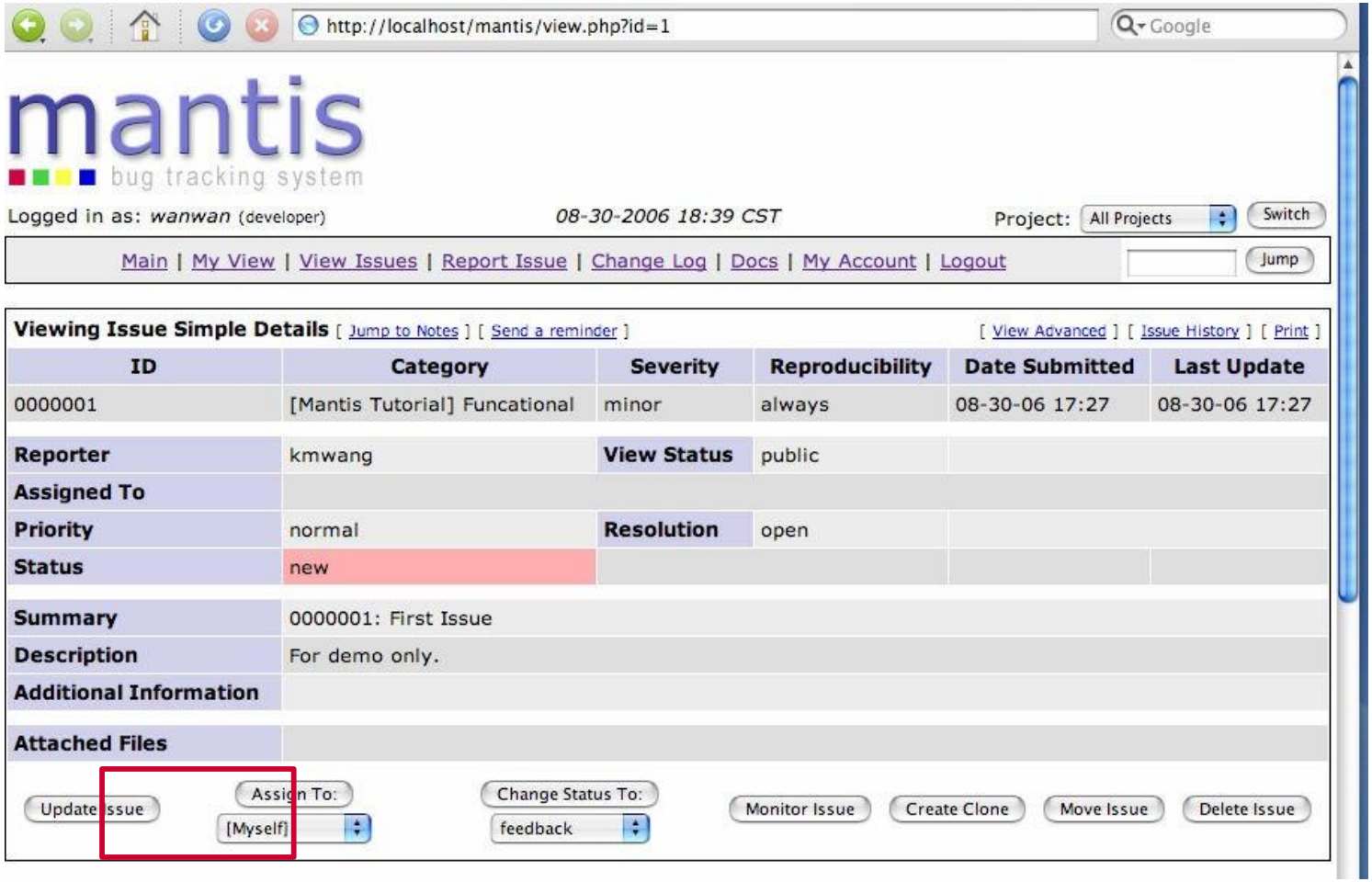
Bug Tracking Tool

Introduction to Mantis

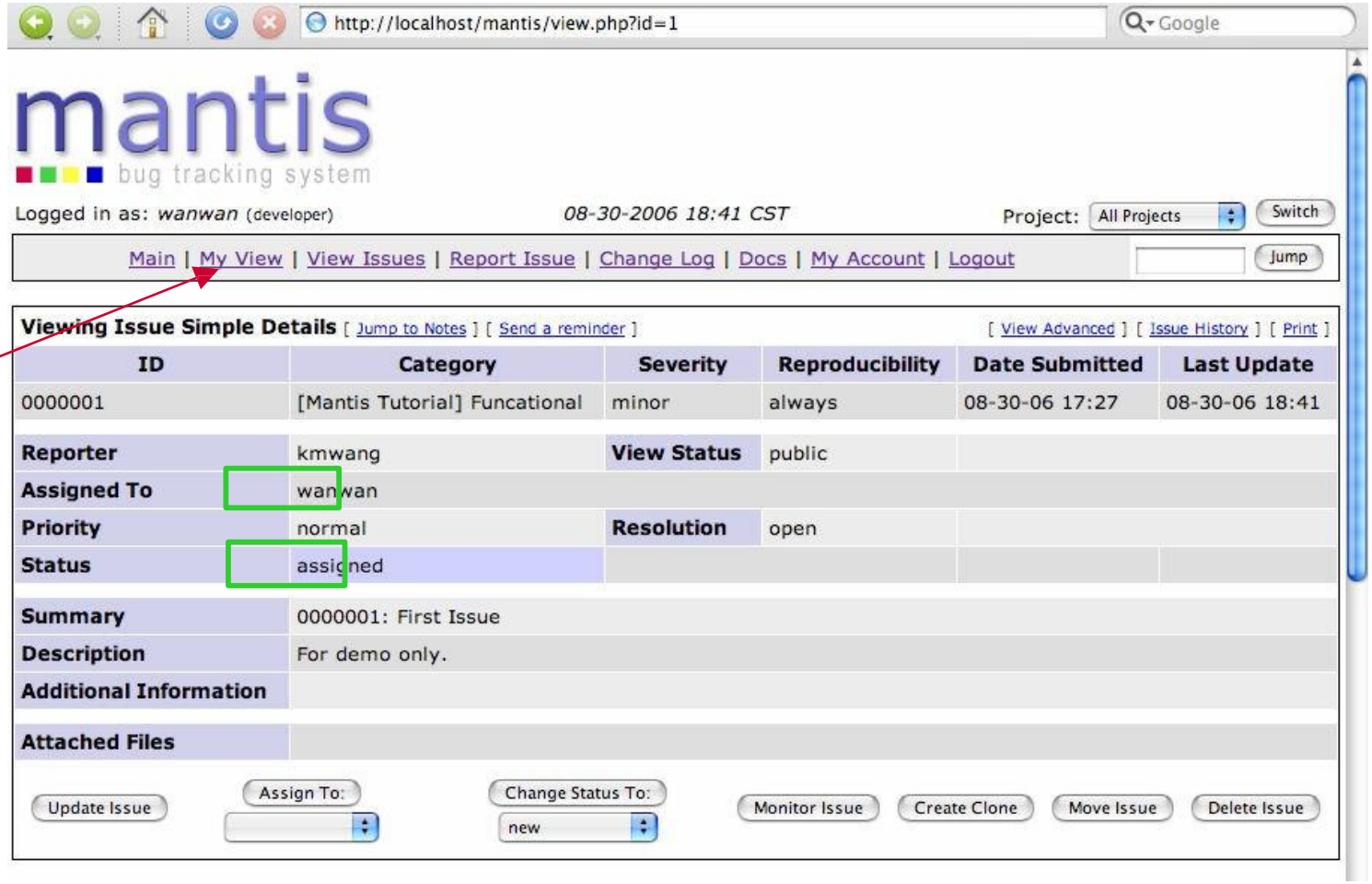
View Issue - After Reporting Issue



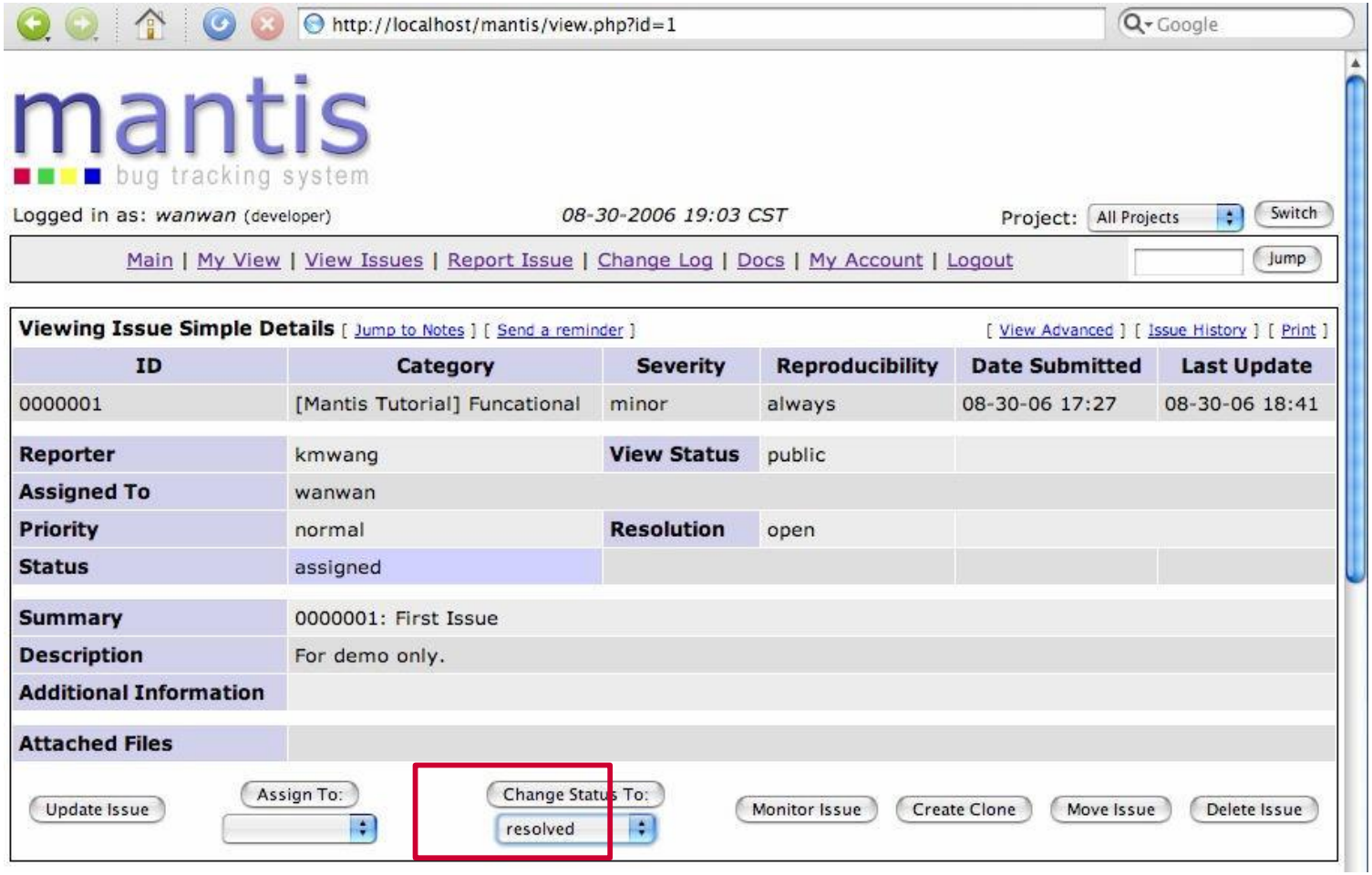
Issue Details



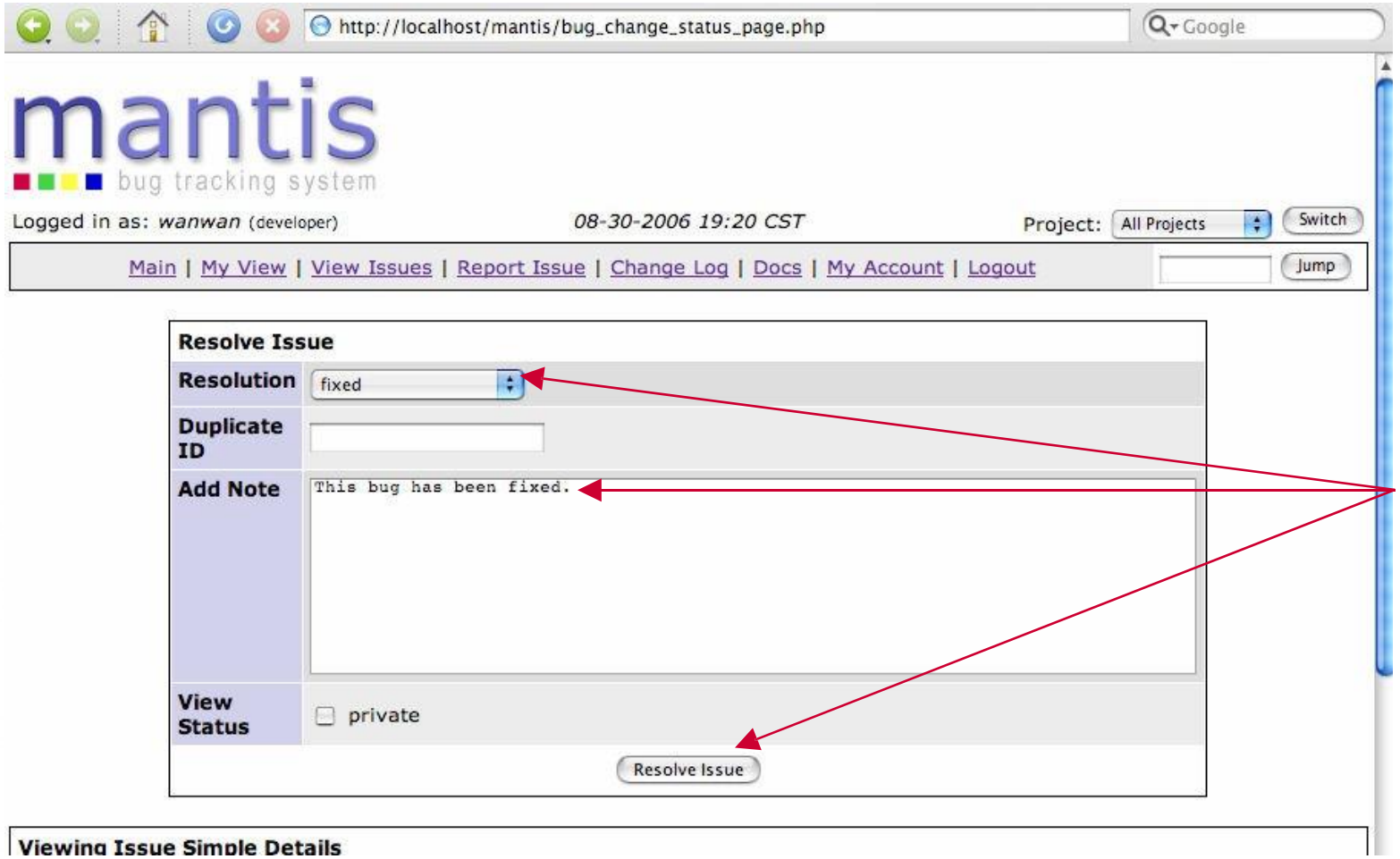
Issue is Assigned



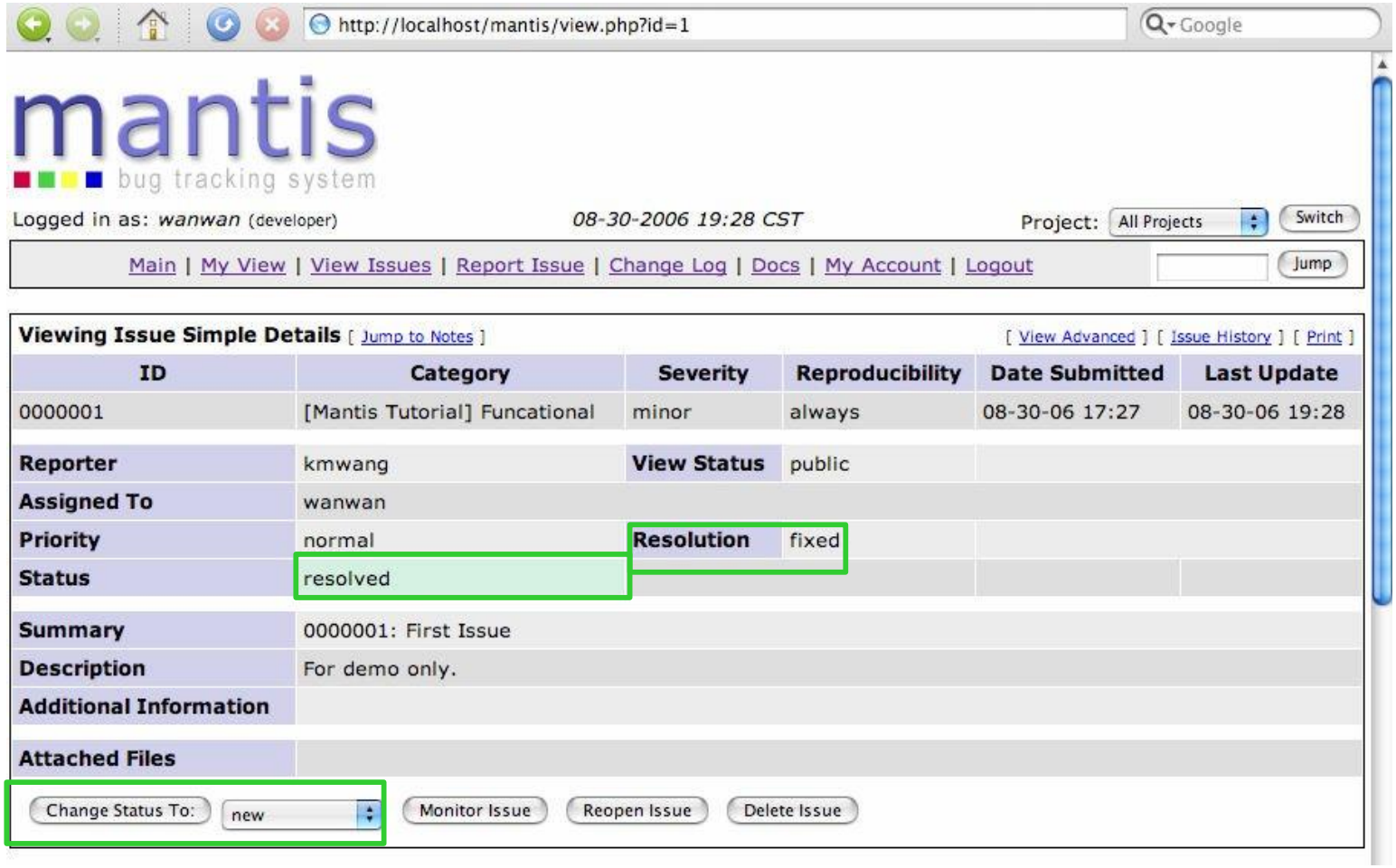
Select New Status



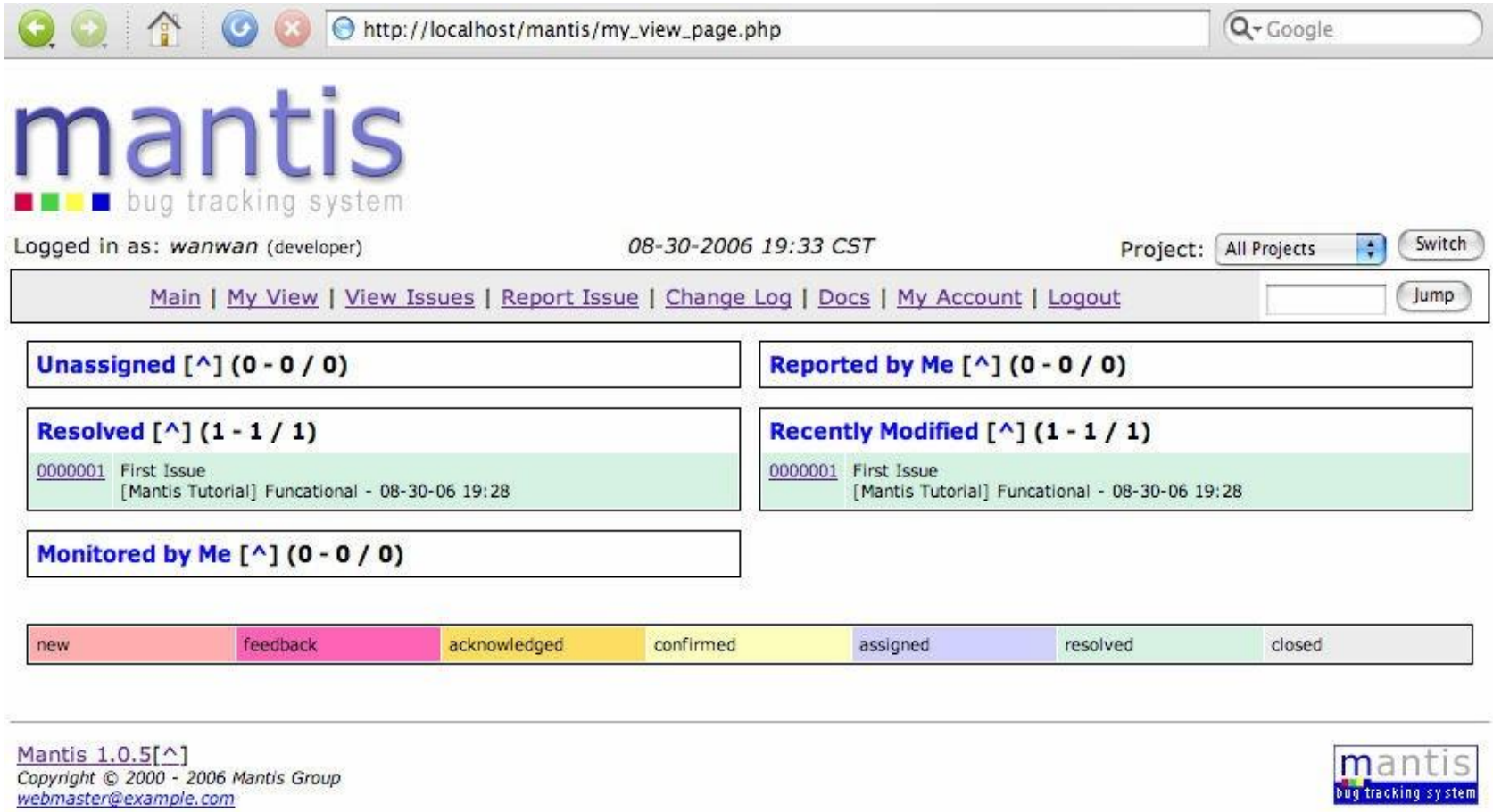
Enter Note of Resolution



Issue be Resolved

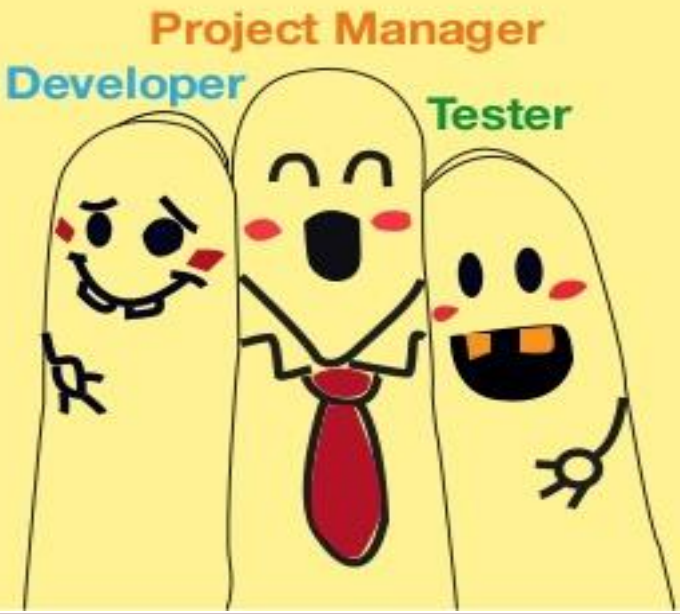


My View After Resolving



Testing Team are here for supporting you

They were all very good friends.



References

- Software testing online courses (ITI Online Platform)

https://uteena.com/courses/course-v1:ITI+ST01+2017_T1/about

- Developer Mindset

http://codebetter.com/blogs/darrell.norton/archive/2003/12/03/4222.aspx

http://www.sqnz.org.nz/documents/ShipHappens/Software%20Quality%20Group%20Presentation_frame.htm

http://pragprog.com/the-pragmatic-programmer

http://en.wikipedia.org/wiki/Fixing_Broken_Windows

http://programmer.97things.oreilly.com/wiki/index.php/The_Boy_Scout_Rule

- Software Craftsmanship

http://en.wikipedia.org/wiki/Software_craftsmanship

http://en.wikipedia.org/wiki/Robert_Cecil_Martin

http://vikashazrati.wordpress.com/2009/11/18/dissecting-software-craftsmanship/

http://clean-code-developer.de/

References

Unit Test

- http://en.wikipedia.org/wiki/Unit_testing
- http://en.wikipedia.org/wiki/XUnit
- http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks

TDD

- http://en.wikipedia.org/wiki/Test_Driven_Development

BDD

- http://en.wikipedia.org/wiki/Behavior_Driven_Development
- http://dannorth.net/introducing-bdd/
- http://behaviour-driven.org/

Code Coverage

http://en.wikipedia.org/wiki/Code_coverage
http://www.ibm.com/developerworks/java/library/j-cq01316/

References

Continuous Integration

- http://en.wikipedia.org/wiki/Continuous_integration
- http://www.codinghorror.com/blog/archives/000818.html
- http://www.stevemcconnell.com/ieeesoftware/bp04.htm
- http://www.joelonsoftware.com/articles/fog0000000023.html
- http://jenkins-ci.org/

Static Code Analysis

http://en.wikipedia.org/wiki/Static_code_analysis

http://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis

Code Review

http://en.wikipedia.org/wiki/Code_review

http://en.wikipedia.org/wiki/Pair_programming

Contacts

E-mail:-

amanyshousha@gmail.com

Linked in:-

https://www.linkedin.com/in/amanyshousha/

Mostly, its been about helping others …
Thank You


Testing
Software
development
test
quality
management
articles
design
code
system
People
Project
information
risk
good
usability
Tools
testers
Agile
web
use
check
process
work
James
Ruby
online
magazine
systems
exploratory
approach
requirements
time
team
bug
introduction
TOPS Technologies:- Software testing course http://www.tops-int.com/software-testing-training.html