



# Software Quality and Testing in MySQL

**Trim Pershad**  
**Omer BarNir**

**MySQL – User Conference and Expo 2009**

---

## Topics

- • Overview of QA
- Process
- Testing
- Bug Analysis and Metrics

## Overview of QA

- MySQL products
  - Server, Cluster, ET, Developer Tools, Connectors
  - We will talk mostly about server
- QA over the years
- QA group structure
- Continuously improving quality

## Overview of QA (cont)

- Internal Processes
  - Well defined release criteria
  - High quality is the goal
- All releases are tested by QA
- New Tests added all the time
- Build most test tools internally

## Topics

- QA Background in MySQL
- • Process
- Testing
- Bug Analysis and Metrics

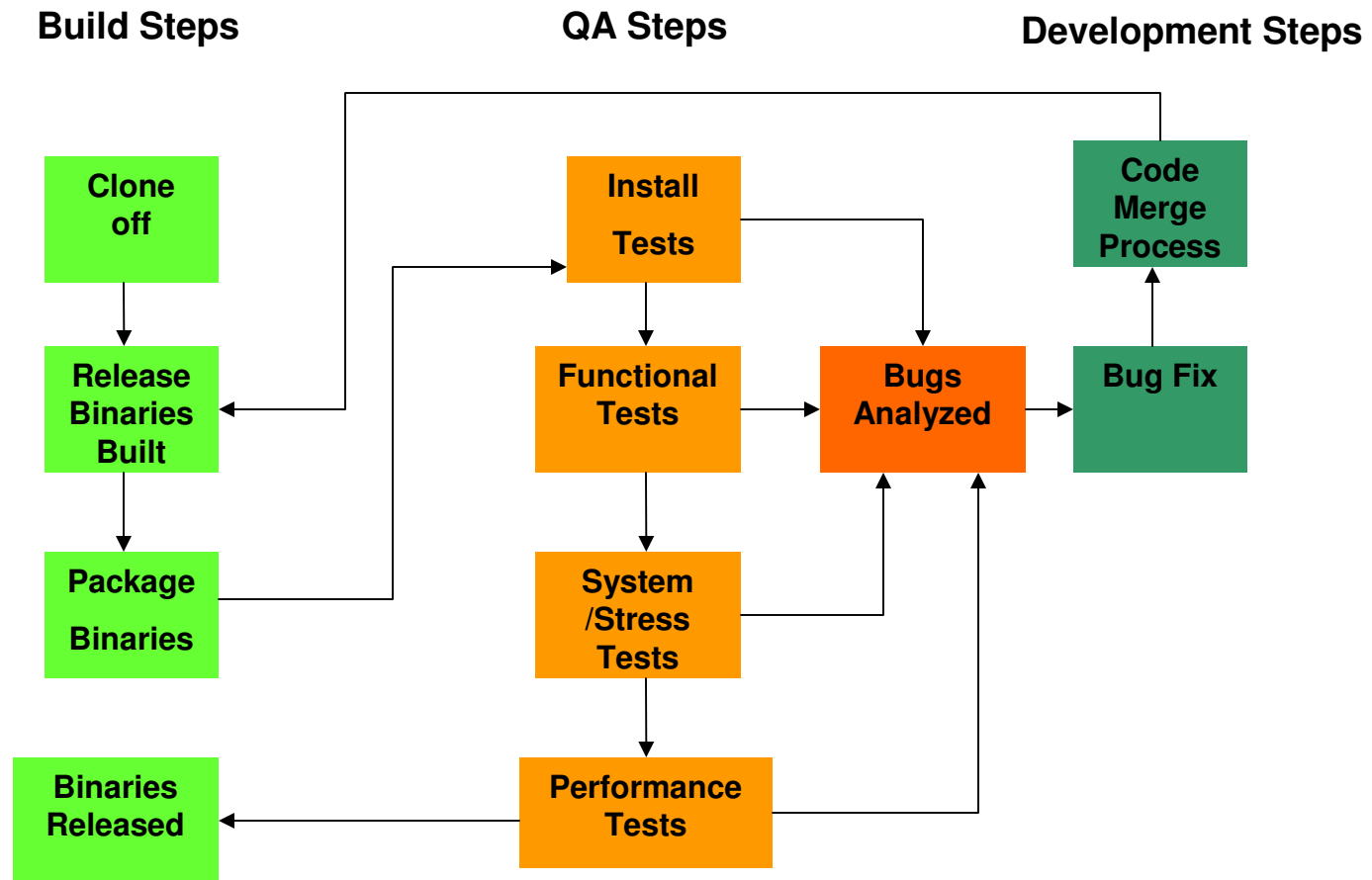
## Process ..Development Cycle

- Development
  - Specifications – Worklogs
  - Design reviews
  - Code reviews
  - Unit tests
  - Team Trees and Main Trees
- QA
  - Test Plans
  - Testing – functional, performance, platforms
  - Code coverage
  - Platforms

## Process ..Release Cycle

- Build
  - Clone off
  - Build binaries
  - Packaging
- QA
  - Run Install Tests
  - Run Functional Tests
  - System and Performance Testing
- Development
  - Bug Fixes

# Process ..Release Cycle



## Topics

- QA Background in MySQL
- Process
- • Testing
- Bug Analysis and Metrics

## Testing

- • Testing frameworks
- Testing suites
- Cross Product Integration Testing

## Testing - Frameworks

- MTR (mysql-test-run)
  - Single user test driver (based on 'test' files)
  - Used for Functional tests
  - Distributed with the MySQL Server
- mysql-stress-test
  - Multi connection scenarios (based on 'test' files)
  - Uses for system/concurrency tests
  - Distributed with the MySQL Server (framework only)

## Testing – Frameworks (cont.)

- Systems Test Framework
  - Multi connection scenarios (Based on mysql-stress-test)
  - Concurrency and Longevity Testing
  - Not Distributed
- Random Query Generator
  - Test Generator
  - Single/multi-user test driver (based on ‘grammar’ files)
  - Used for Functional, Compatibility and System tests
  - Separate Tree

## Testing – Frameworks (cont.)

- Upgrade-downgrade
  - Functional scenarios
  - Uses for upgrade, downgrade
  - Not Distributed

## Testing – Frameworks, Performance

- Sysbench
  - C Based application, Multi threaded connections
  - Built in scenarios: OLTP, Memory, CPU, threads
  - Supports Lua extensions (v0.5.0)
  - Public Tree
- MySQLBench
  - Single user 'atomic' test scenarios
  - Perl Based
  - Included in the mysql distribution
- DBT2 Tests
  - Based on TPC-C benchmark

## Testing

- Testing frameworks
- • Testing suites
- Cross Product Integration Testing

## Test Suites

- Functional
  - Functional Test suite
  - NIST compliance tests
  - Large table tests
  - Random/Complex query testing
  - Upgrade/Downgrade Compatibility
- System/Longevity
  - Concurrent IUDS scenarios
  - Replications
  - Crash/Recovery tests

## Upgrade/Downgrade Compatibility

- Start Old Server
- Run test script (init\_xxx.test)
- Shut down old server start new one
- Run upgrade process (\*)
- Run test script (upgrade\_xxx.test)
- Shut down new server, start old one
- Run test script (downgrade\_xxx.test)
- Shutdown server
  
- (\*) Different upgrade path
  - Live (same database)
  - Dump/restore
  - Replication (old master new slave)

## Concurrent IUDS scenarios

- Start Server
- Run script initializing tables/users/Procedures etc.
- Launch multiple clients,
  - Continuously run until the end of the test
  - Each running a random scenario and exists
- Runs tasks
  - ‘management’ activities (truncate log tables etc)

## Concurrent IUDS scenarios - Example

```
#
# This test is deleting rows based on a random value
#
--source suite/systems/include/system_1_init.inc
let $NUM_VAL=`SELECT @NUM_VAL`;
USE systest1;

# Setting parameter information
--replace_result $NUM_VAL <NUM_VAL>
eval set @f1_nums=$NUM_VAL;
set @tmp_num=int_rand(@f1_nums);

delete from tb0_eng1 where f1=@tmp_num;
```

```
#
# This test is inserts rows based on a random value
# using a stored procedure
--source suite/systems/include/system_1_init.inc
let $NUM_VAL=`SELECT @NUM_VAL`;
USE systest1;

# Setting parameter information
--replace_result $NUM_VAL <NUM_VAL>
eval SET @f1_nums=$NUM_VAL;
SET @tmp_num=int_rand(@f1_nums);
SET @tmp_word=str_rand(4);

CALL ins_tb0_eng1 (@tmp_num, @tmp_word);
```

```
#
# This test is updating rows based on a random value
#
--source suite/systems/include/system_1_init.inc

let $NUM_VAL=`SELECT @NUM_VAL`;
use systest1;

# Setting parameter information
--replace_result $NUM_VAL <NUM_VAL>
eval set @f1_nums=$NUM_VAL;
set @tmp_num=int_rand(@f1_nums);
set @tmp_word=str_rand(4);

update tb0_eng1
set f2=concat('U:',@tmp_word,'-',@tmp_num), f3=f3+1
where f1=@tmp_num-1;
```

## Concurrent IUDS scenarios – Example (cont.)

```
# This procedure scans 'tb0_master' table for rows where f1 = 'num_pr'
# and for each row INSERTs a row in 'tb0_eng1'
```

```
#-----
```

```
CREATE PROCEDURE ins_tb0_eng1 (num_pr INT, str_pr CHAR(15))
```

```
BEGIN
```

```
  DECLARE done INT DEFAULT 0;
```

```
  DECLARE v3 DECIMAL(5,3);
```

```
  DECLARE cur1 CURSOR FOR
```

```
    SELECT f3 FROM tb0_master WHERE f1 = num_pr;
```

```
  DECLARE CONTINUE HANDLER FOR SQLSTATE '01000' SET done = 1;
```

```
  DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done = 1;
```

```
  OPEN cur1;
```

```
  FETCH cur1 INTO v3;
```

```
  wl_loop: WHILE NOT done DO
```

```
    INSERT INTO tb0_eng1 (f1, f2, f3, f4) VALUES
```

```
      (int_rand(@f1_nums),
```

```
       CONCAT('l:',str_pr,'-',num_pr),
```

```
       v3, NOW());
```

```
    FETCH cur1 INTO v3;
```

```
  END WHILE wl_loop;
```

```
  CLOSE cur1;
```

```
END//
```

```
#
```

```
# This test is inserts rows based on a random value
```

```
# using a stored procedure
```

```
CREATE TRIGGER tb0_eng1_ins AFTER INSERT ON tb0_eng1 FOR EACH ROW
```

```
  INSERT INTO tb0_logs (dt1, entry_dsc, f4)
```

```
  VALUES (NOW(), CONCAT('Insert row ', NEW.f1, ', '
```

```
          NEW.f2, ' ', NEW.f3, ' (tb0_eng1)'), NEW.f1);
```

```
CREATE TRIGGER tb0_eng1_upd AFTER UPDATE ON tb0_eng1 FOR EACH
```

```
ROW
```

```
  INSERT INTO tb0_logs (dt1, entry_dsc, f4)
```

```
  VALUES (NOW(), CONCAT('Update row ', OLD.f1, ', ', OLD.f2, '->',
```

```
          NEW.f2, ' ', OLD.f3, '->', NEW.f3, ' (tb0_eng1)'), NEW.f1);
```

```
CREATE TRIGGER tb0_eng1_del AFTER DELETE ON tb0_eng1 FOR EACH ROW
```

```
  INSERT INTO tb0_logs (dt1, entry_dsc, f4)
```

```
  VALUES (NOW(), CONCAT('Delete row ', OLD.f1, ', ', OLD.f2, ', '
```

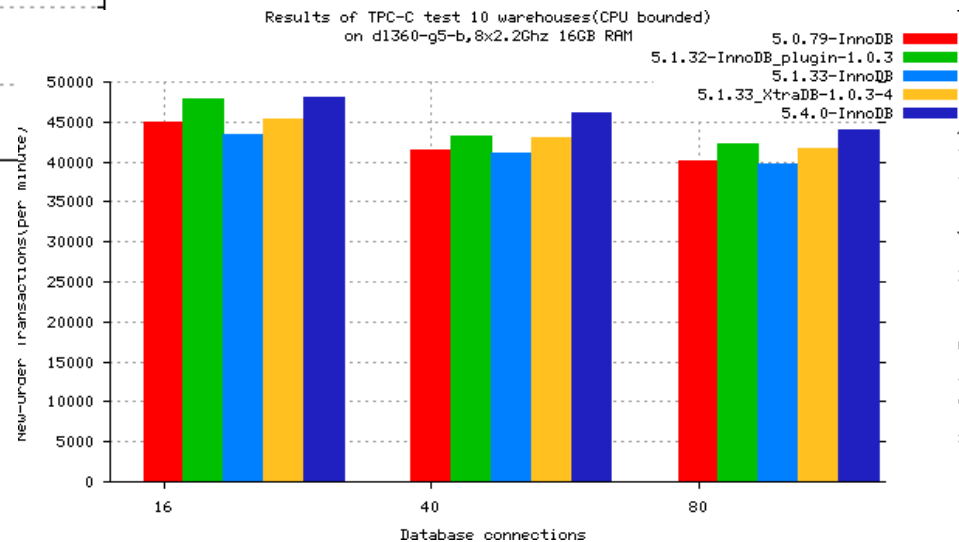
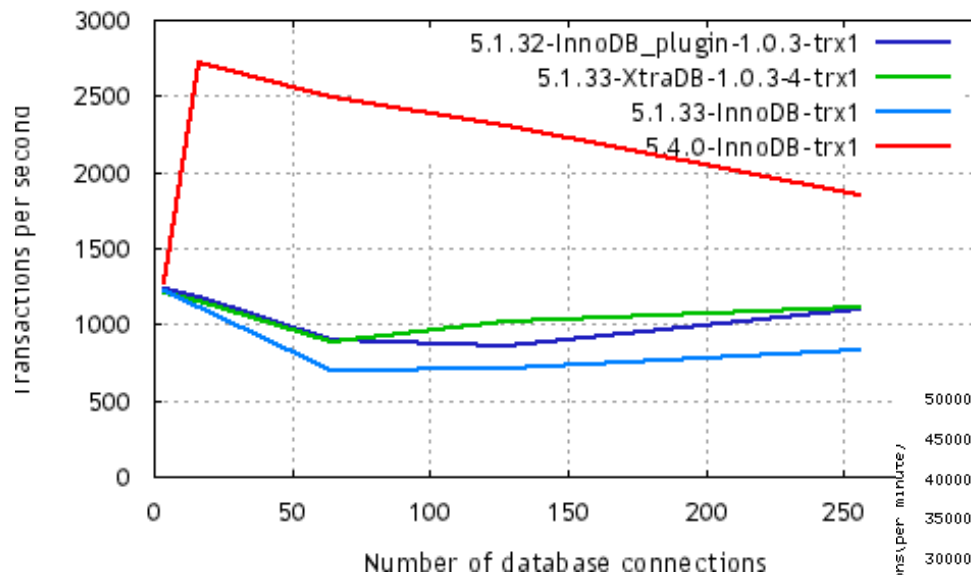
```
          OLD.f3, ' (tb0_eng1)'), OLD.f1);
```

## Test Suites (cont.)

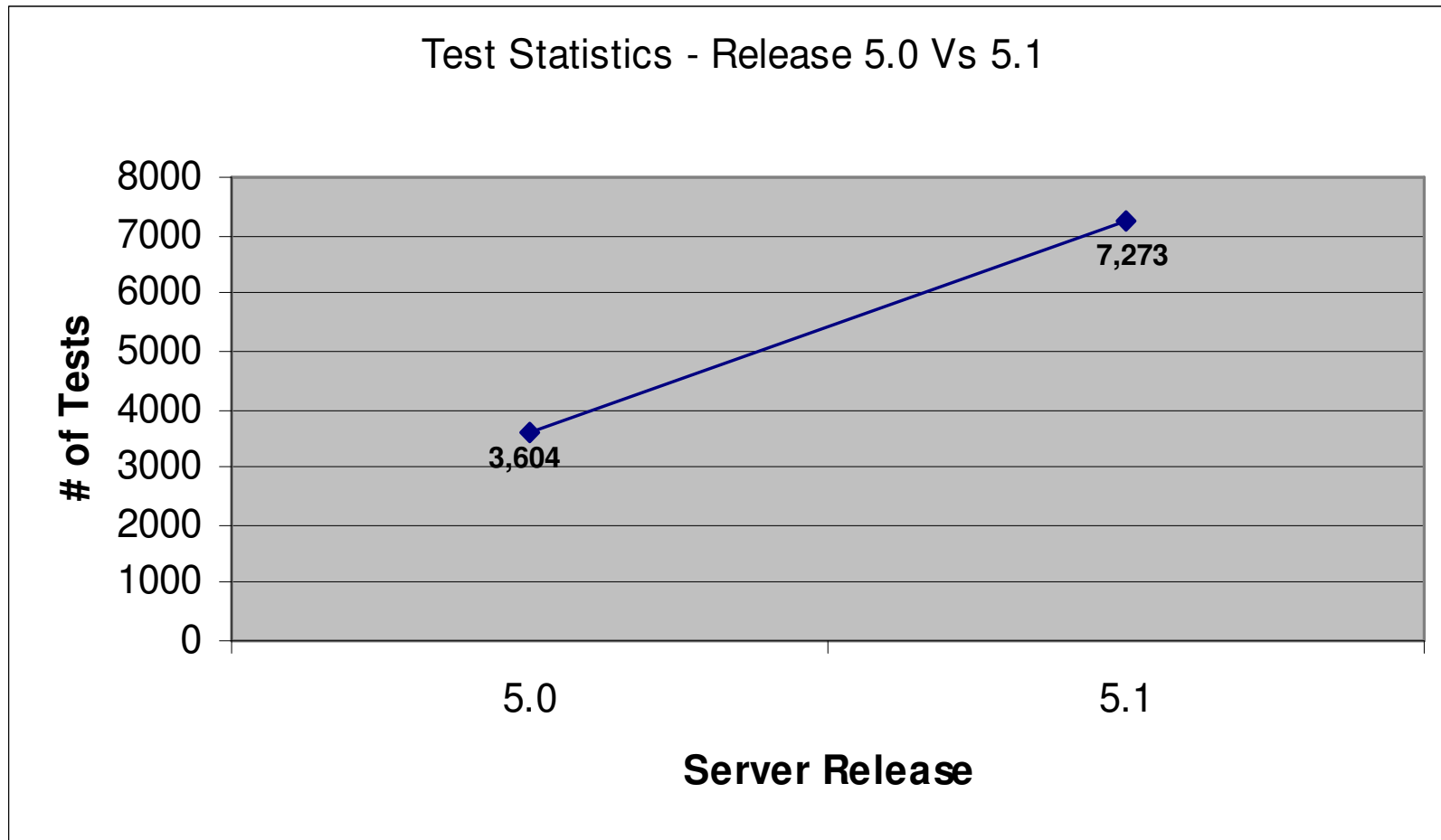
- Install Testing
- Load/Stress
  - Testing the system under high user load with connect/disconnect and IUDS operations
- Performance
  - Performance testing of SQL operations
  - Performance trend and changes with changing number of threads

# Test Suites – Performance Charts

Test: OLTP\_RW  
 InnoDB sysbench 1M rows. Linux x64 dl360-g5-b  
 innodb\_flush\_log\_at\_trx\_commit=1



# Test Statistics



## Testing

- Testing frameworks
- Testing suites
- • Cross Product Integration Testing

## Cross-Product Integration Testing

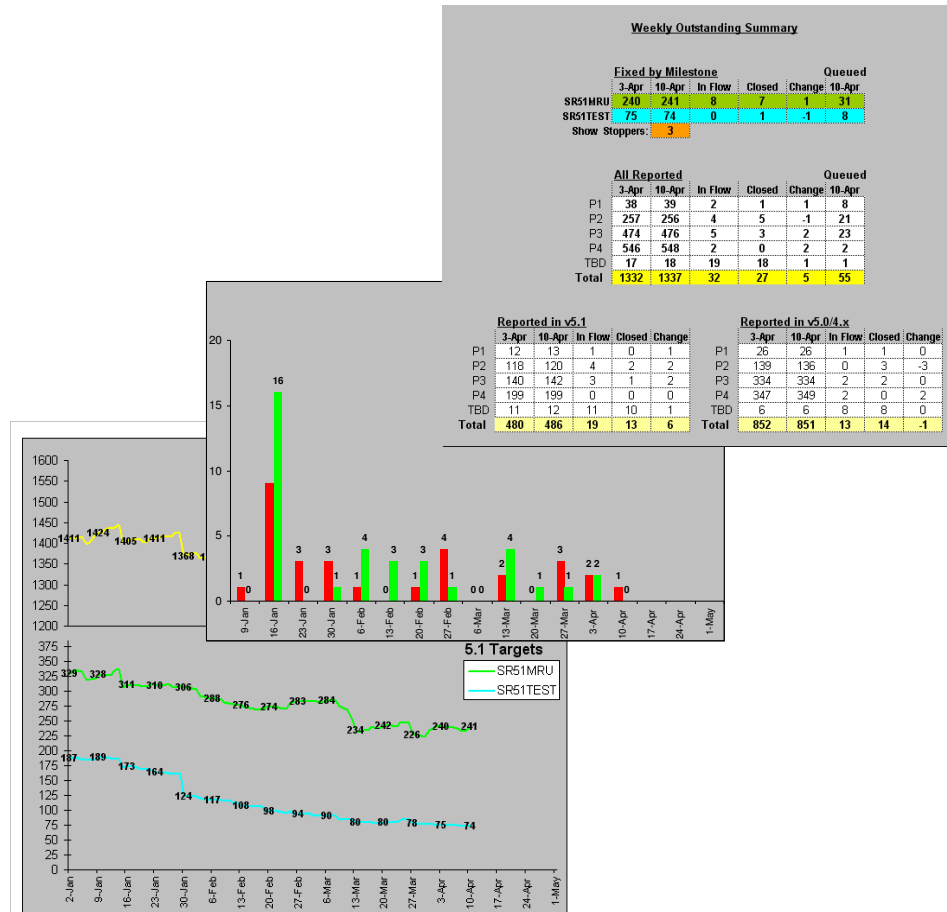
- Server Testing
  - Based on using client tools based on the C-API Interface
- Connector Testing
  - Systematic testing of connectors
  - Testing using sample applications built with common tools
    - Microsoft Access
    - Crystal Reports

## Topics

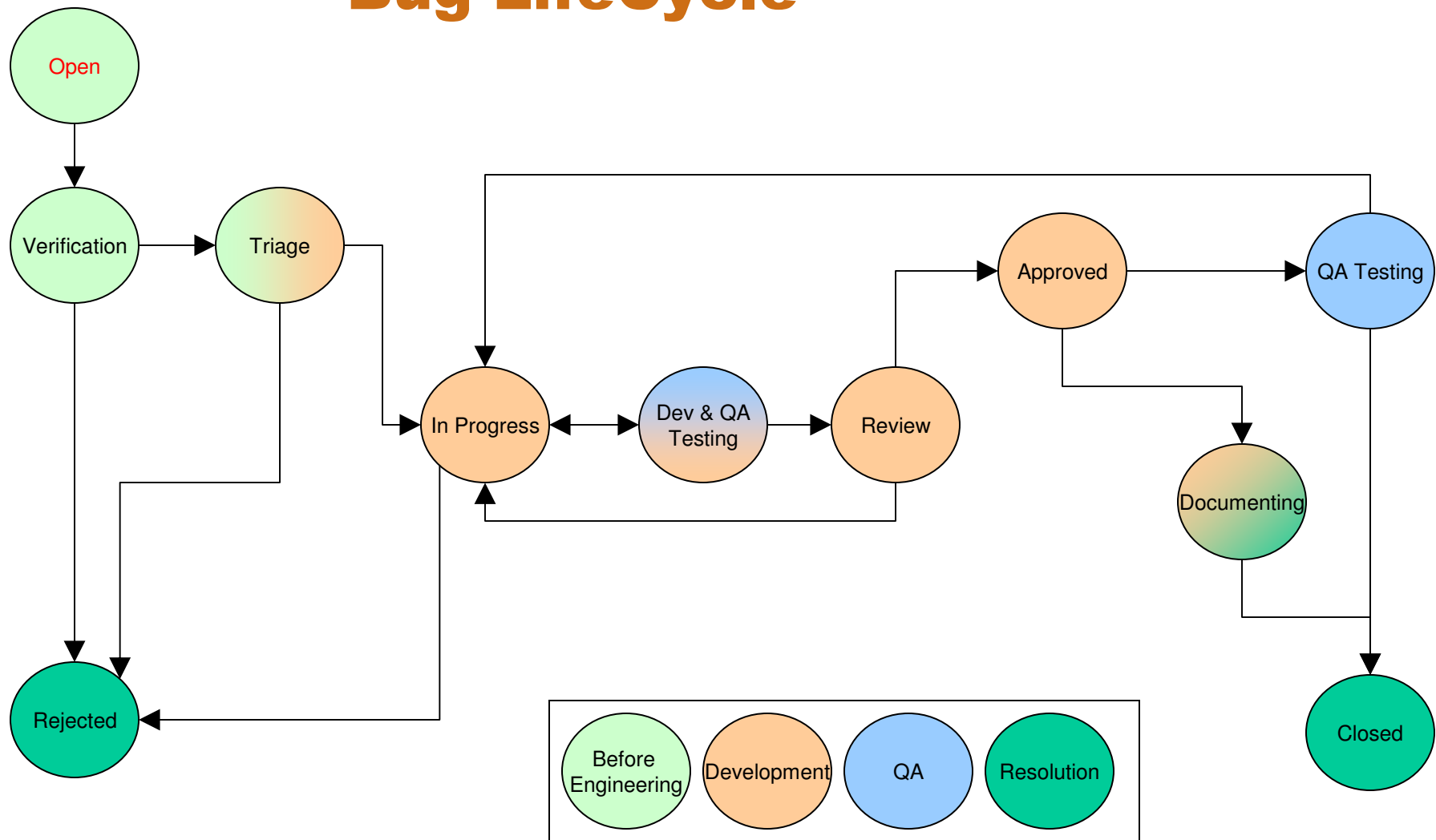
- QA Background in MySQL
- Process
- Testing
- • Bug Analysis and Metrics

# Metrics and Bug Analysis

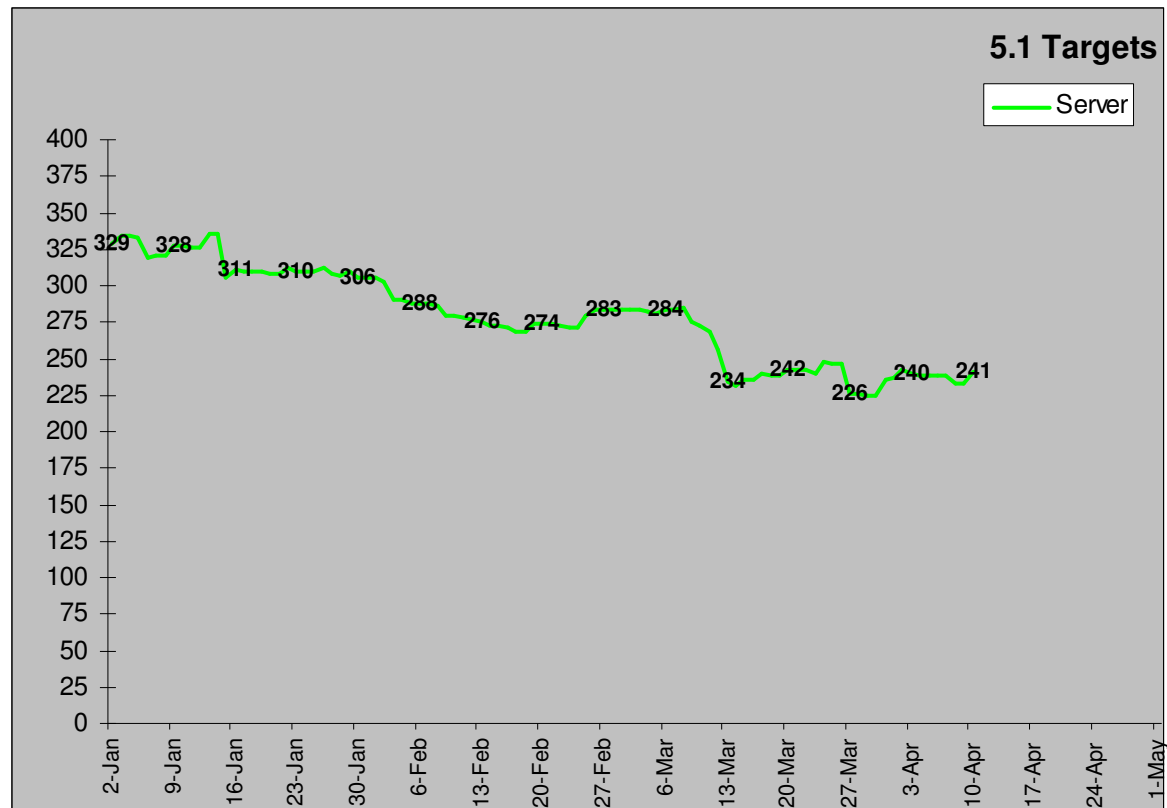
- Bug KPI Tracking
  - Used in making Quality decisions
- Bug Analysis
  - Bug Analysis
  - Prioritization of bugs
    - (cross team activity)



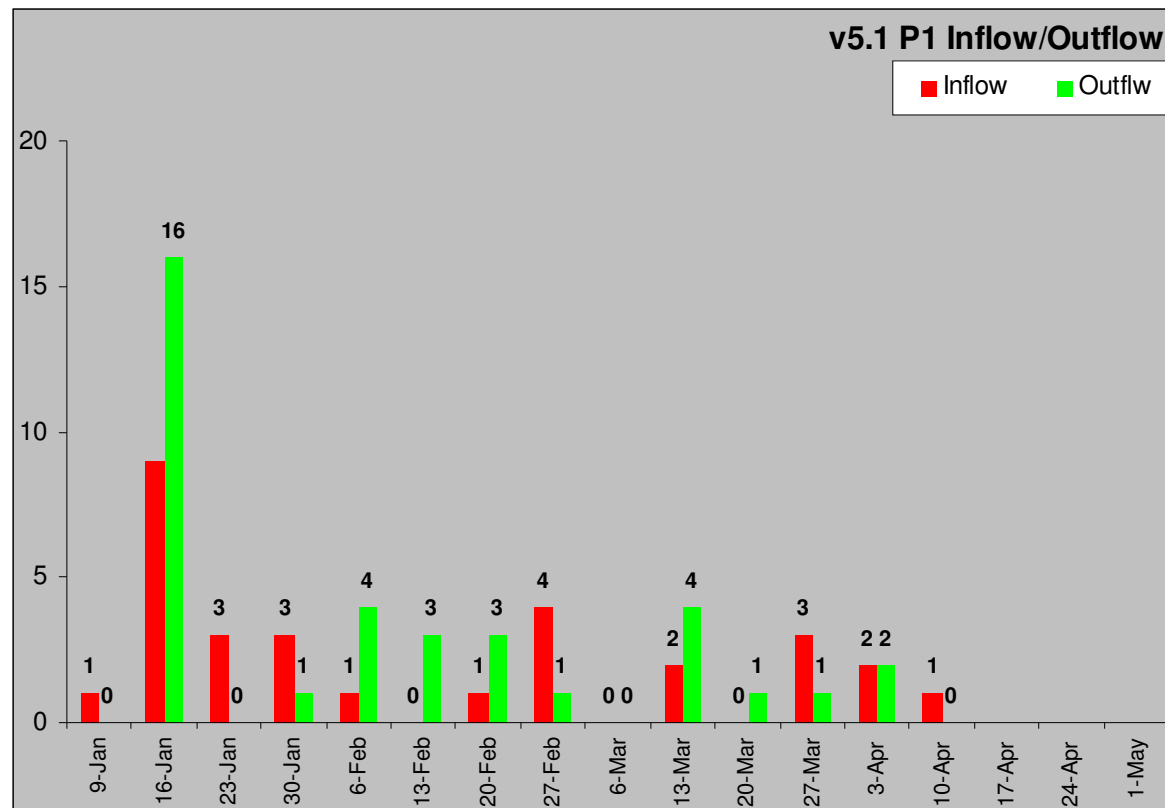
# Bug LifeCycle



## Metrics and Charts



# Metrics and Charts





# Questions?

## Contact Information

- Trim Pershad [\*trim.pershad@sun.com\*](mailto:trim.pershad@sun.com)
- Omer BarNir [\*omer.barnir@sun.com\*](mailto:omer.barnir@sun.com)