PNSQC

OCTOBER 10-12 2022

AMOL PATIL
Orthogonal RCA

RCA Goals

- Defect prevention
- Incident reduction

Root Cause Analysis

Defining Defect Prevention

- Defect prevention is a process whose purpose is to:
  - identify the common causes of defects, and
  - change the relevant process(es) to prevent that type of defect from recurring. (SEI)

- *Take what we already know and apply it to what we think we know to produce quality software.*

Root Cause Analysis

Orthogonal Defect Classification

- Orthogonal Defect Classification
- Developed at IBM in the 1990s by Ram Chillarege
- Methodology to characterize software defects and translate into process defects

Root Cause Analysis

Important Points about ODC

- *A defect in the software is a defect in the process*

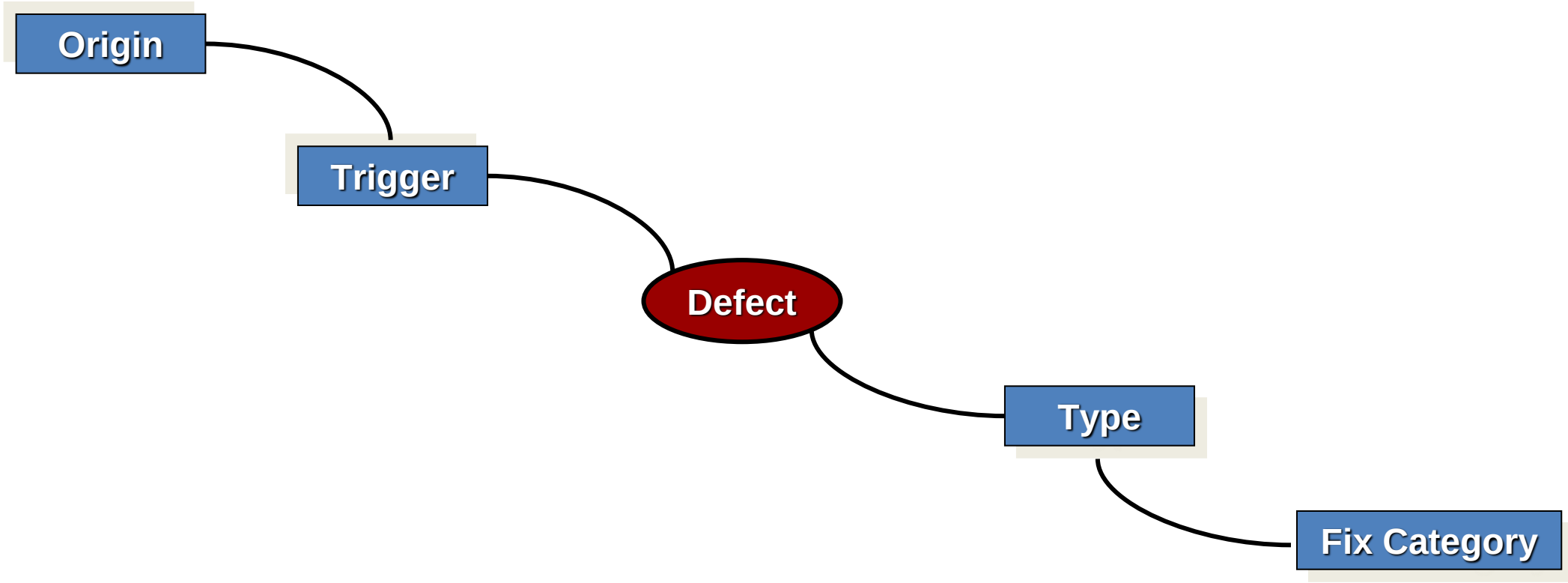
- Implementing ODC is very cost-effective
  - Enhances data already collected (software defects)
  - Adding fields that are completed real-time make data collection virtually free!
  - Tooled to quickly identify process defects (mapping)
- ODC can be implemented in stages
  - Start with field defects, then move to in-process analysis
  - Utilize defect profiling in-process to predict quality and project status
- Fields can be tailored to your own organization

Root Cause Analysis

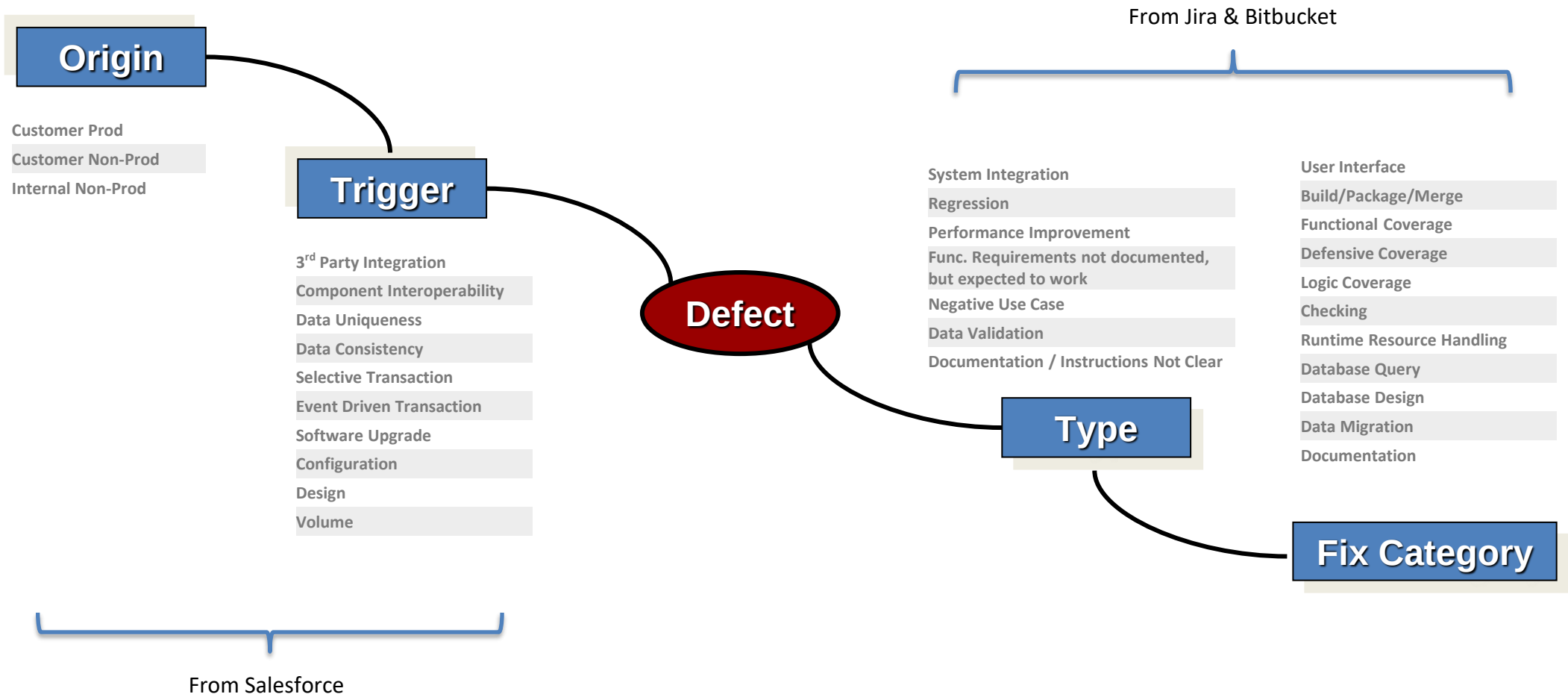
Defect/Incident Profiling

- Defect Origin
- Business Routine
- Defect Trigger
- Trace
- Volume
- Frequency
- Last Modification
- History
- How did the customer get to this point
- Why is this happening?
- Preliminary/Official Root cause

Simple Defect Classification Scheme

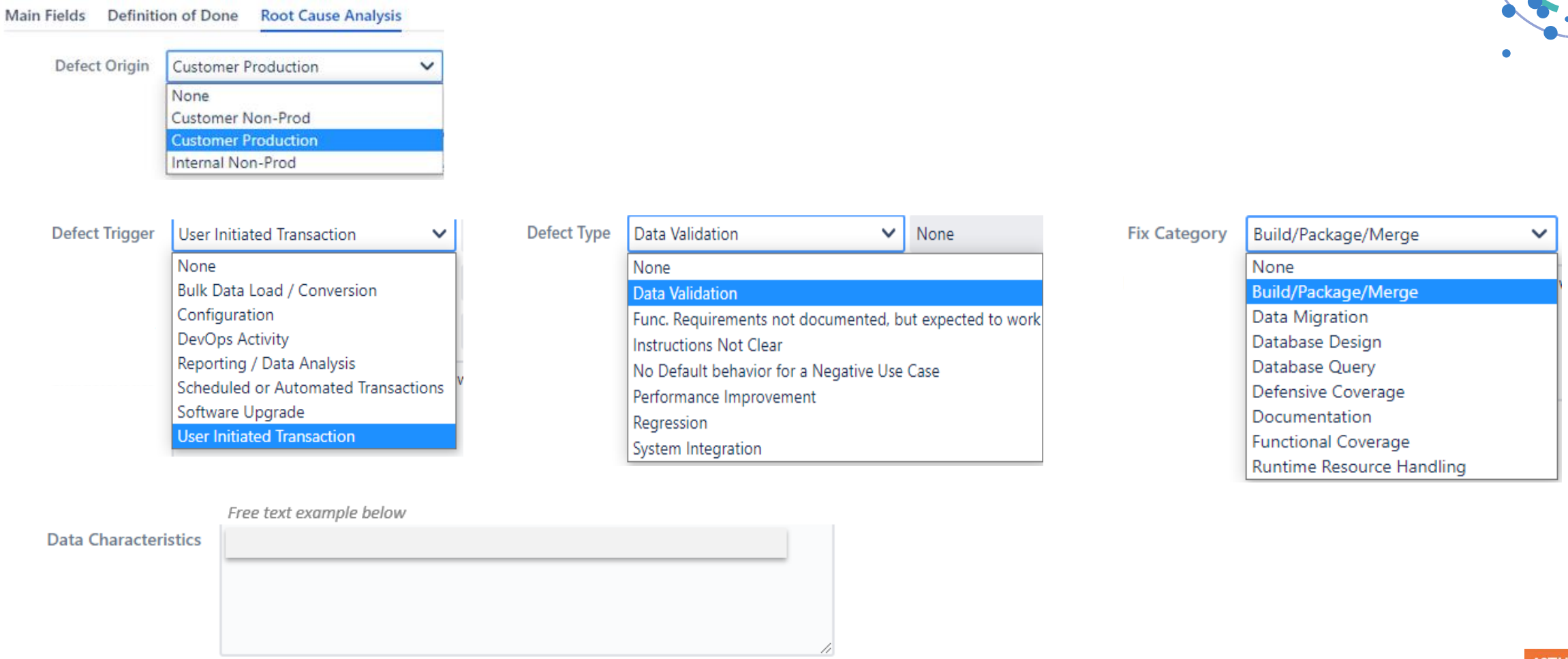


Simple Defect Classification Scheme



AMOL PATIL
Orthogonal RCA

Jira: RCA Section

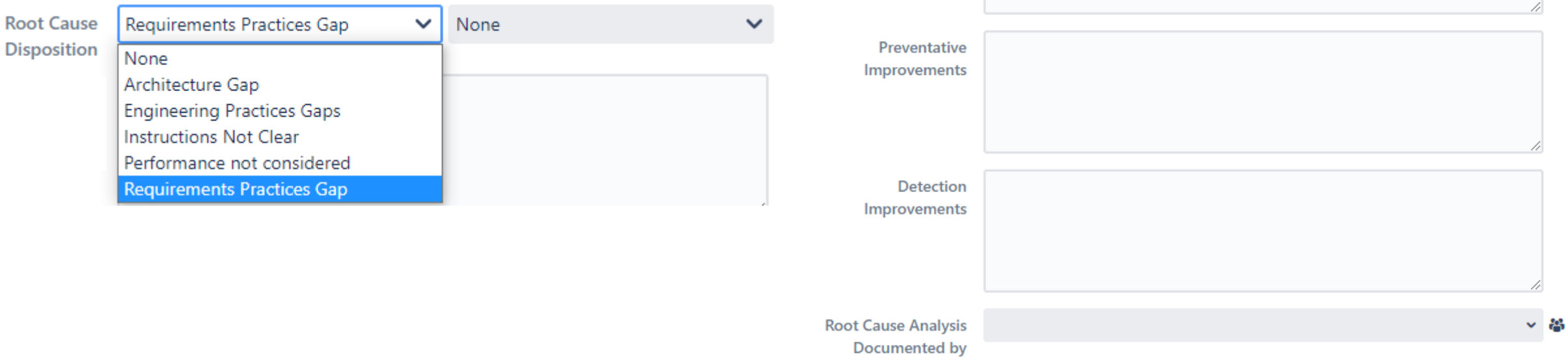


AMOL PATIL

Orthogonal RCA

40TH ANNUAL PACIFIC NW SOFTWARE QUALITY CONFERENCE
OCT. 10–12, 2022

Jira: RCA Section



Guiding framework for Preventative improvements

Use the hierarchy of controls chart to answer the question

How could these type of defects be prevented in future ?

AMOL PATIL
Orthogonal RCA

40TH ANNUAL PACIFIC NW SOFTWARE QUALITY CONFERENCE OCT. 10-12, 2022

Template for Preventative improvements

- Going forward as part of the <process>, the <role> will <action> the <improvement>

Process		Action	Improvement
HLR gathering	Tech lead	complete	
HLR breakdown to stories	Principal Programmer	review	
Story grooming	Tester	start	
Backlog triaging	Product owner	close	
Sprint planning / retro /demo	Tech Writer		
Design approach	Delivery Lead / Manager		
Coding / Programming	Bus Analyst		
Code review			
Unit testing			
Code compile / build / package / distribute			

- This will be measured using <tool>
- Adherence will be checked using <process>

Tool	Process
Jira report with <these> Jira fields	Audit
DoD	

Template for Detection improvements

> Going forward as part of the <process>, the <role> will <action> the <improvement>

Process		Action	Improvement
Unit Testing	Test Engineer	complete	Regression checklist
Integration testing	Tech lead	review	Test design document
Test design approach	Programmer/Developer	start	Default behavior in Acceptance criteria
Test plan writing	Product owner	close	Non-functional requirements (Perf., Scal., Compatibility)
Manual story testing	Tech Writer	address	
Ad-hoc/Buddy testing	QA manager	utilize	
Data migration testing	Bus Analyst	add	
Test review and approval with team		update	
System testing		monitor	
Quality Criteria measurement – Perf/Scal.			

> This will be measured using <tool>

> Adherence will be checked using <process>

Tool	Process
Jira report with <these> Jira fields	Audit
DoD	Sprint planning / Sprint retrospective
Daily bitbucket review	Story acceptance demo
Requirements DoR checklist	

Root Cause Analysis

Orthogonal Defect Classification

- Define smart categories that defects can be classified into
- Choices for selection for each categories based on being able to objectively answer
  - Did the team understand the business needs
  - Did the team brainstorm the solution to the best possible fit for the business need
  - Did the team implement the solution with best practices
  - Did the team examine the results through different levels of effective testing
- Objectively assign RC disposition

Why ODC?

- ODC helps to create a narrative flow that can lead to simplified way of reaching the RC conclusion
  - Defect Trigger in Defect Origin led to Defect Type that was addressed in this Fix category leading to RC Disposition
- Analysis over a large set of data provides hot spots and weak points
- Detailed RCA methods like 5 Why's and Fish Bone Diagram approach are time consuming
- ODC can be incorporated into the defect tracking system

Identifying hot spots from RCA data

What Engineering and Product leaders can learn from RCA data

Orthogonal RCA

40TH ANNUAL PACIFIC NW SOFTWARE QUALITY CONFERENCE OCT. 10–12, 2022

Defect Origin w/ Defect Type

→ Defect Type ↓ Defect Origin	Data Validation	Func. Requirements not documented, but expected to work	Instructions Not clear	No default behavior for a Negative Use Case	Performance Improvement	Regression	Grand Total
Customer Non-Prod	14	18	3	3	4	8	**50**
Customer Prod	35	53	5	7	14	27	**141**
Internal Non-Prod	1	7		1		3	**12**
Grand Total	**50**	**78**	**8**	**11**	**18**	**38**	**203**

Data Source: Sample created for example purpose only

- What is happening?
  1. 70% bugs found in prod environments
  2. Underlying faults in the products become failures due to business routines
  3. Data sources used by business users in Prod is triggering many data validation related problems in the software

- Why is this happening?
  1. Bugs missed being caught by Test cycles in Engineering and in Customer UAT cycles
  2. Business users use the product in ways that is not well known to customer and Engineering Dev and Test teams
  3. Configuration information and applicability of the changes only known to business users

Defect Type w/ Defect Trigger

→ Defect Type ↓ Defect Trigger	Data Validation	Func. Requirements not documented, but expected to work	Instructions Not clear	No default behavior for a Negative Use Case	Performance Improvement	Regression	Grand Total
Configuration	1	1			1	1	4
Software upgrade	5	1	1		1	5	13
Scheduled or Automated Transactions	38	50	1	8	11	13	121
User Initiated Transaction	4	20	3	3	5	14	49
Reporting / Data Analysis	1	6				2	9
DevOps Activity	1		3			1	5
Bulk Data Load / Conversion						2	2
Grand Total	**50**	**78**	**8**	**11**	**18**	**38**	**203**

Data Source: Sample created for example purpose only

- What is happening?
  1. Story implementation missing many boundary and edge conditions, but valid business processes from a customers POV
  2. Data ingestion inconsistency
  3. Data validation by design happens differently

- Why is this happening?
  1. Treating each feature/defect as an individual occurrence and not using checklists with all other supported behavior
  2. Highly configurable product allows creation of valid and invalid data source for ingestion
  3. Connector and payor do data validation differently on certain fields by design

Defect Origin w/ Defect Type

→ Defect Type ↓ Defect Origin	Configuration	Software upgrade	Scheduled or Automated Transactions	User Initiated Transaction	Reporting / Data Analysis	DevOps Activity	Bulk Data Load / Conversion	Grand Total
Customer Non-Prod	2	7	30	11				50
Customer Prod	2	5	84	36	7	5	2	141
Internal Non-Prod		1	7	2	2			12
Grand Total	**4**	**13**	**121**	**49**	**9**	**5**	**2**	**203**

Data Source: Sample created for example purpose only

- What is happening?
  1. Scheduled & Automated transactions triggering large number of bugs
  2. User Initiated transactions are the second highest category
  3. DW Bootstrap issues are not reported in Non Prod

- Why is this happening?
  1. Variation of Data sources and data volumes intended for processing
  2. The business user intentions of is not known clearly to the dev and test teams at Customer and HE
  3. Configuration information and applicability of the changes only known to business users

Fix category w/ Defect Trigger

→ Defect Type ↓ Fix Category	Configuration	Software upgrade	Scheduled or Automated Transactions	User Initiated Transaction	Reporting / Data Analysis	DevOps Activity	Bulk Data Load / Conversion	Grand Total
Build/Package/Merge		2						2
Data Migration	1	2			1			4
Database Design		4	4					8
Database Query		1	7	2		2		12
Defensive Coverage		2	39	11	3		1	56
Documentation		1	3	5		3		12
Functional Coverage	2	1	66	28	5		1	103
Runtime Resource Handling	1		2	3				6
Grand Total	**4**	**13**	**121**	**49**	**9**	**5**	**2**	**203**

Data Source: Sample created for example purpose only

- What is happening?
  1. Business users at times using the product in a way that does not make logical sense to HE
  2. Event driven / scheduled operational items that are part of regular business routines getting stuck
  3. Upgrades introducing regressive behavior when processing new behavior
  4. User is creating or changing system configuration and encounters an error in the configuration process itself

- Why is this happening?
  1. Usability of the product allows ingenuine usage
  2. Default behavior on failure event(s) not allowing manual processing of stuck items
  3. Real world data, configuration and process missing from dev and test cycles
  4. Product does not restrict users from configuring processes/policies/categories and maintenance in unintended ways

Defect Type w/ Fix category

→ Defect Type ↓ Fix Category	Data Validation	Func. Requirements not documented, but expected to work	Instructions Not clear	No default behavior for a Negative Use Case	Performance Improvement	Regression	Grand Total
Build/Package/Merge						2	2
Data Migration	2	1				1	4
Database Design	2	1			3	2	8
Database Query	1	5			2	4	12
Defensive Coverage	29	5		7		15	56
Documentation	1	1	8			2	12
Functional Coverage	15	65		4	7	12	103
Runtime Resource Handling					6		6
Grand Total	**50**	**78**	**8**	**11**	**18**	**38**	**203**

Data Source: Sample created for example purpose only

> What is happening?

1. Major codebase churn is around introducing new functional logic to handle missing or wrong functionality
2. Reactive changes in the codebase to Address poorly defined code boundaries and data validation for unexpected data resources
3. Traces left are Java and Oracle Errors

> Why is this happening?

1. Story implementation primarily uses representative coverage
2. Corner cases with low occurrence probability not in the primary path of operating cases, but cases can happen and do happen
3. Framework not in place to give meaningful error messages for users to take remediation steps

PNSQC

OCTOBER 10-12 2022

THANK YOU