Juntoz Development Cycle

Last update: 2019-01-13

Intro

Index

Agile

Agile Manifesto

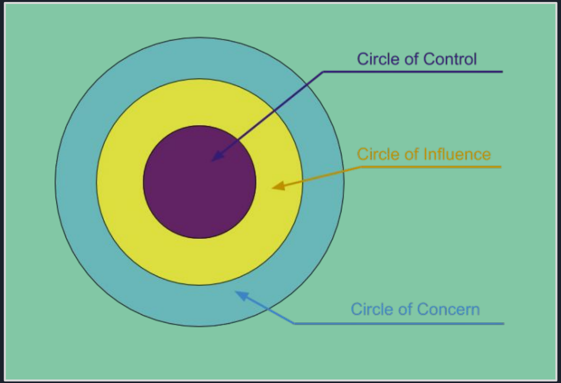
- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Some Agile Principles and Values

- Adapt / Embrace Changes
- Seek Early Feedback / Fail Often, Fail Fast
- Working Software is the best measure of progress
- Focus
- Quality
- Teamwork
- Refactor

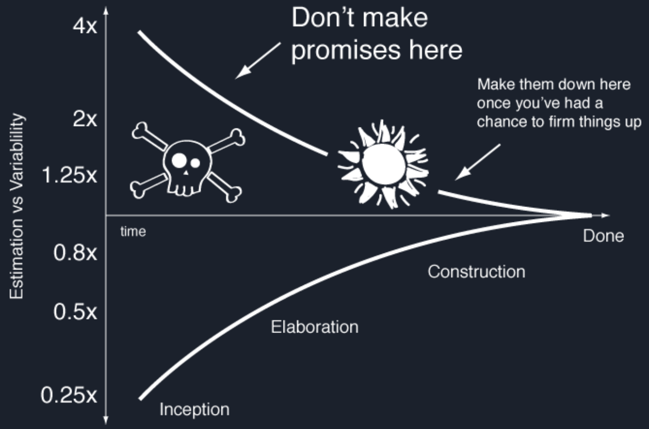
Adapt and Embrace Changes

- Adapt to keep competitive advantage.
- Push for decisions to be made every day.
- Question what you know and what you have to do every day.
- Every complex system can change at any time. Do not assume you can control it.
- Instead of fighting it, embrace it.



Embrace uncertainty

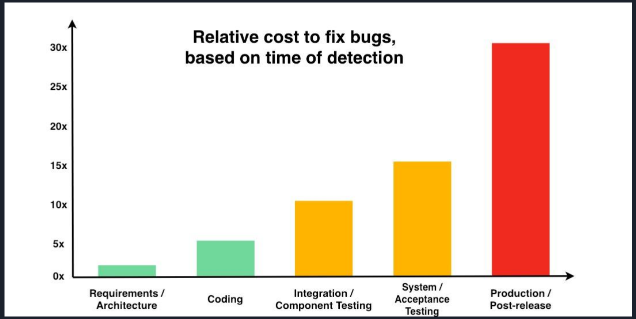
- We cannot even plan ahead of this day, how will you do 15 days ahead?
- Make a decision as responsibly late as possible.
- Do not plan everything up front
- Do not design everything up front
- Do not develop something that is not even designed or decided



Don't make promises here
Make them down here once you've had a chance to firm things up
Estimation vs Variability
4x
2x
1.25x
0.8x
0.5x
0.25x
time
Done
Inception
Elaboration
Construction

Seek Early Feedback / Fail Fast Fail Often Fail Cheap

- Feedback is not only from the customer
- Feedback is also from the tests we do
- The shortest feedback cycle is in the developer computer.
- To know if something works, you need to deliver fast, and cheaply.
- Paper / Board, Mockup PSD, Mockup Browser, Mockup HTML, Developer Local, Staging, Production.



Working Software as the measure of progress

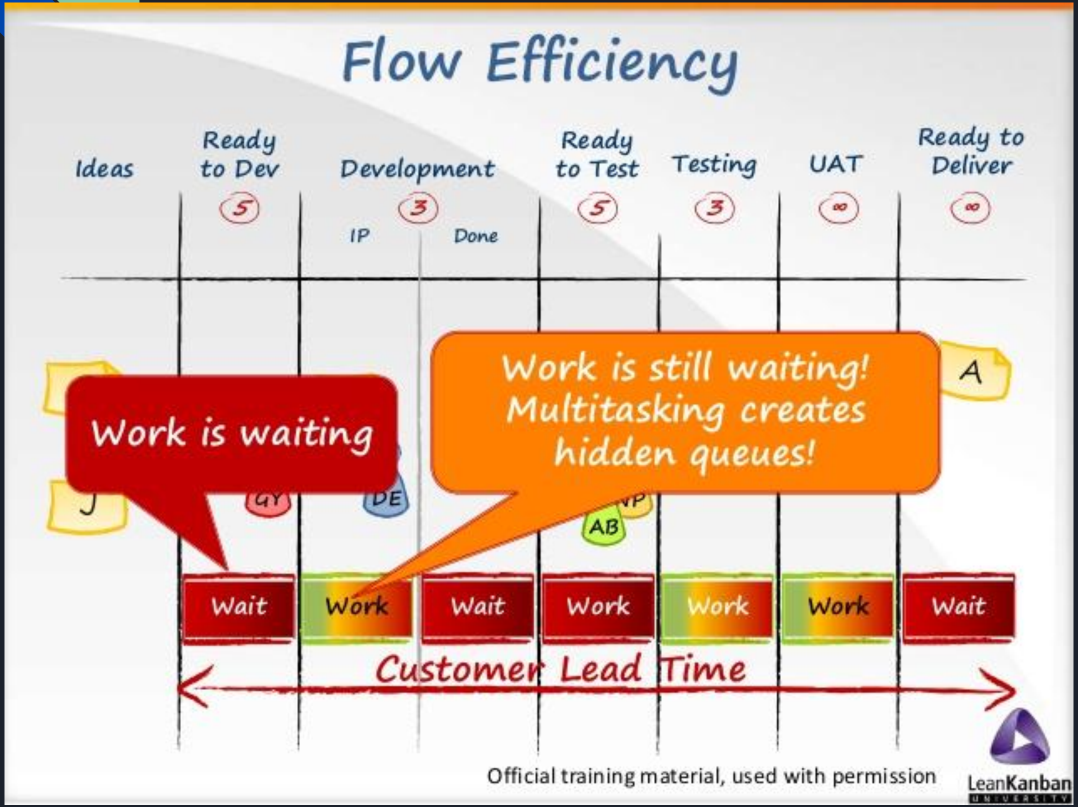
- Early Feedback
- Refactor Short Cycle
- Prioritize End to End
- Analysis Paralysis
- Do not design overly complex solutions when they are not needed
- Short Sprint
- Demo

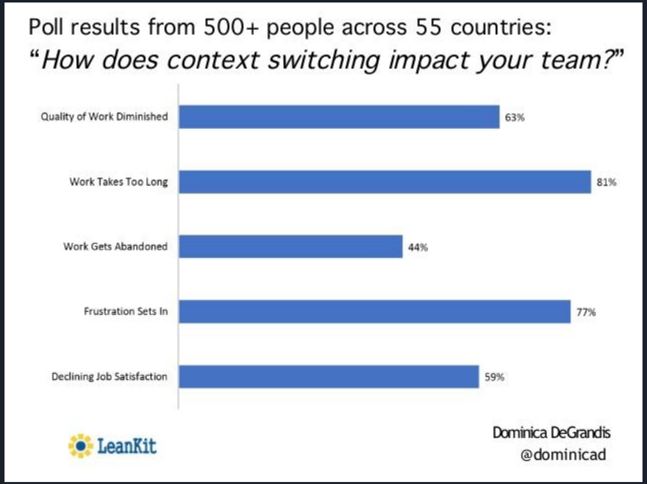
<Draw Customer Anxiety Over Time>

Focus

- Focus on value
- Focus on one PBI at a time
- The whole team focuses on one goal at a time
- Multitasking: it delays all the tasks in the end
- Focus on the team, not just myself

CONTEXT SWITCH

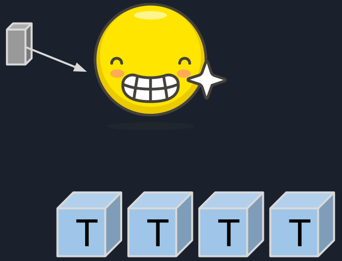



Poll results from 500+ people across 55 countries:
"*How does context switching impact your team?*"

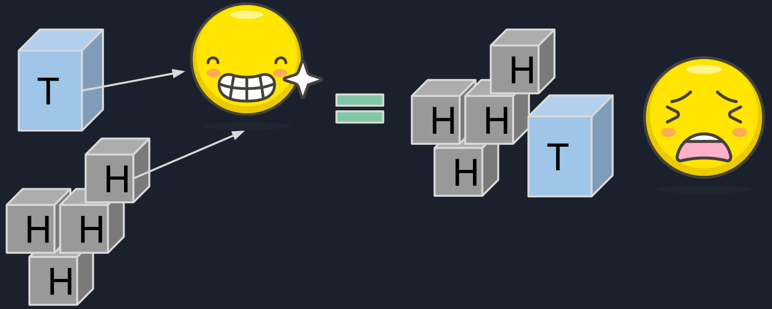
Quality of Work Diminished	63%
Work Takes Too Long	81%
Work Gets Abandoned	44%
Frustration Sets In	77%
Declining Job Satisfaction	59%

Hand Workers vs Mental Workers

- Repetitive Tasks with no/small context
- Small tasks, several tasks can be accomplished per day. Predictable.
- Offloading and Uploading are small and almost no back history
- Physical Exhaustion



- Tasks of variable sizes and variable goals
- You might accomplish one, many or none per day. Not Predictable.
- Offloading and Uploading are costly, with a lot of back history
- Mental Exhaustion



Exercise: Context Switching

Quality

- Quality is **<u>everyone's responsibility</u>**. Not only by testers.
- Quality must be applied to all steps in the cycle:
  - Flow (prioritize value, prioritize output, question everything, constant review of deadlines, prioritize end-to-end flows).
  - Development (Unit Tests, Integration Tests, Code Standards, Code Metrics, Best Practices).
  - Testing (UI, UX, Functionality, Performance, Scalability, Blackbox, Whitebox).
  - Deployment (automatic, no user interaction, repeatable. reliable).
- If a bug is found, report it in Jira (triage will happen later) and report to team lead.
- UI and UX are bugs, "yellow screen of death" is a bug, unexpected behavior is a bug (which can be converted to a story), slowness is a bug.

Teamwork

Great things in business are never done by one person, they're done by a team of people.
Steve Jobs
AZ QUOTES

Teamwork

- In waterfall mode, teamwork means to do my task and pass on the result.
- In agile, it means that all three interact and work (based on their own role) towards the same task or goal.

	Task 1	Task 2	Task 3
Waterfall	PO	DEV	QA
Agile	PO DEV TEST	PO DEV TEST	PO DEV TEST

- Teamwork also means that you must realize someone depends on you, therefore it is important to do a timely handover with good quality.

Refactor

“Always leave the campground cleaner than you found it”

- Refactor can be small:
  - Rename a variable
  - Delete a variable
  - Better error handling
  - Reorder methods
  - Adding a TODO comment
- Question what you see and know
- Envision how things should be
- Do it by phases, baby-steps. Return to main branch as fast as possible.
- **Take <u>controlled</u> risks**

Done and Ready

DONE

- Dev **DONE** (Development, Code Review and Unit Tests).
- Tests **DONE** (Manual and Automated, Acceptance and Alternative).
- Production **DEPLOYED**

READY

- User Need **STABLE**
- Tech Design **STABLE**
- Acceptance Criteria **DEFINED**
- User Story **ESTIMATED**
- Acceptance Tests **IDENTIFIED**

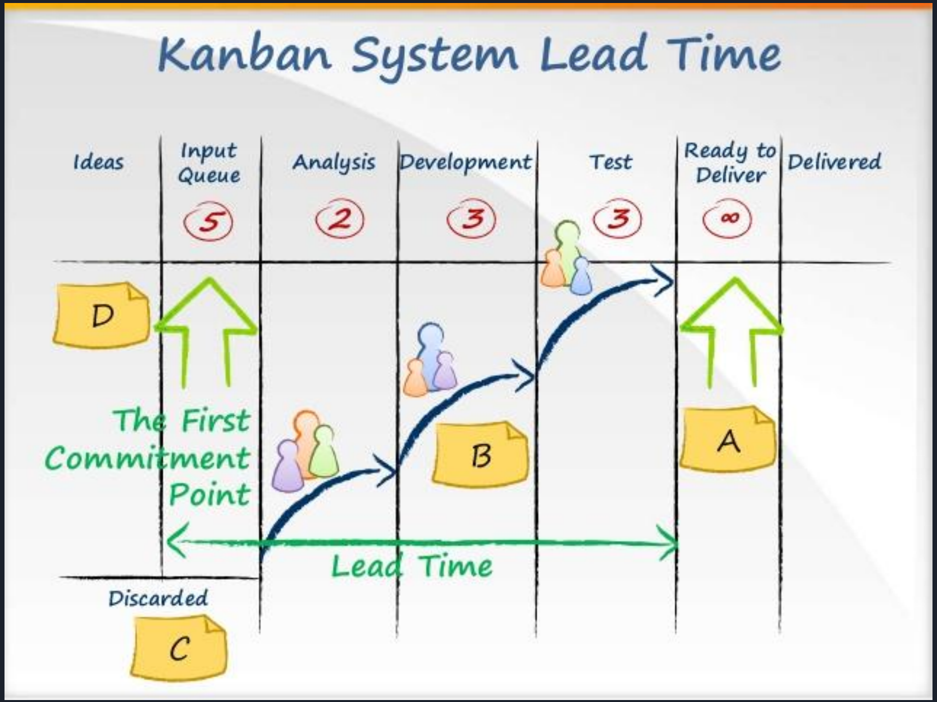
DO NOT START THE STORY UNTIL IT IS READY

TRY NOT TO START ANOTHER STORY UNTIL THE CURRENT ONE IS DONE

How to Deliver Value?

- What is Value? What matters to the software end user the most (either final user, merchant user, admin, etc).
- Delivering value also means it has to be useful and given in a timely manner.
- Acceptance Flow : happy flow of basic operation
- Alternative Flows: other operations that compliment the acceptance flow.
- Prioritize END TO END Acceptance flow
    - Later, identify alternative flows, and prioritize them by recurrence.
    - **Avoid multitasking (exercise) per person. DECREASE CONTEXT SWITCHING.**
    - **Avoid waterfall work (exercise) in the team. DECREASE LEAD TIME.**
    - Divide in stories **INVEST**.

What is LEAD TIME?

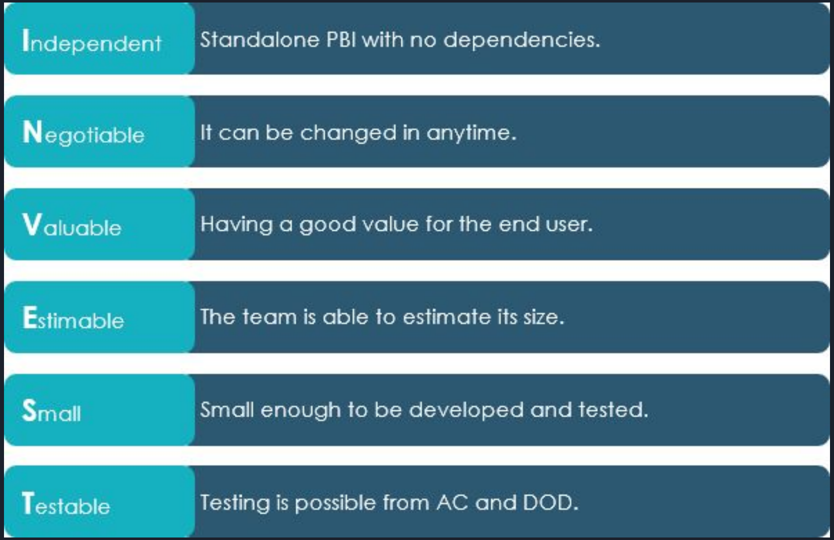


Exercise: Lead Time

Coin game

- Switch coins
- Batch = 10, 5, 2
  - Measure first coin delivery time
  - Measure last coin delivery time

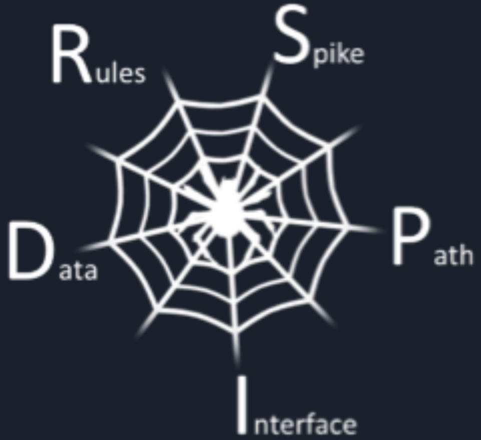
INVEST



Independent	Standalone PBI with no dependencies.
Negotiable	It can be changed in anytime.
Valuable	Having a good value for the end user.
Estimable	The team is able to estimate its size.
Small	Small enough to be developed and tested.
Testable	Testing is possible from AC and DOD.

TO BE AS MUCH AS POSSIBLE

SPLITTING



Or

https://agileforall.com/wp-content/uploads/2018/02/Story-Splitting-Flowchart.png

Exercise: User Story Splitting

Group 1: As a merchant user, I want to create a coupon that gives 20% discount over the entire order for some of my products.

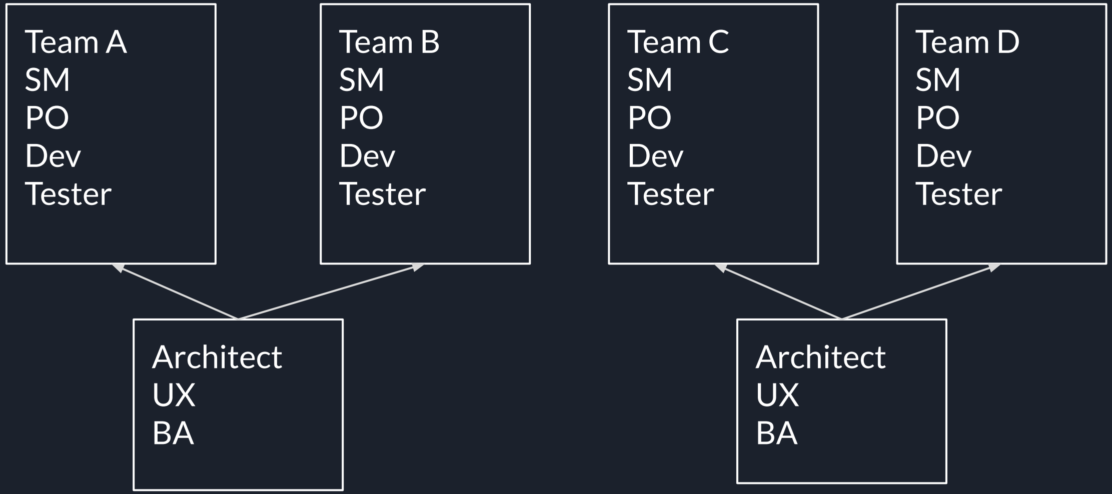
Group 2: As a customer support rep, I want to give a credit to the end user so he can use it in a later purchase.

Goal: list of stories, acceptance flow (MVP) and some alternative flows.

(15 min splitting + 10 min presentation)

Teams

Product Teams



Dev Community of Practice
Architect
Dev

SM Community of Practice
SM

Test Community of Practice
Tester

PO Community of Practice
PO
Business Analysts

Consultant Teams

$$\begin{tikzcd}
\text{Customer} & \text{Pre Sales + Solutions Architect => Proposal} \arrow[l, "1"'] \\
& \text{Sales => Contract and Billing} \arrow[ul, leftrightarrow] \\
& \text{Project Manager => Tracking and Delivery} \arrow[uul, leftrightarrow, "2"]
\end{tikzcd}$$

Business/Software Analyst?
Architect?
UX?

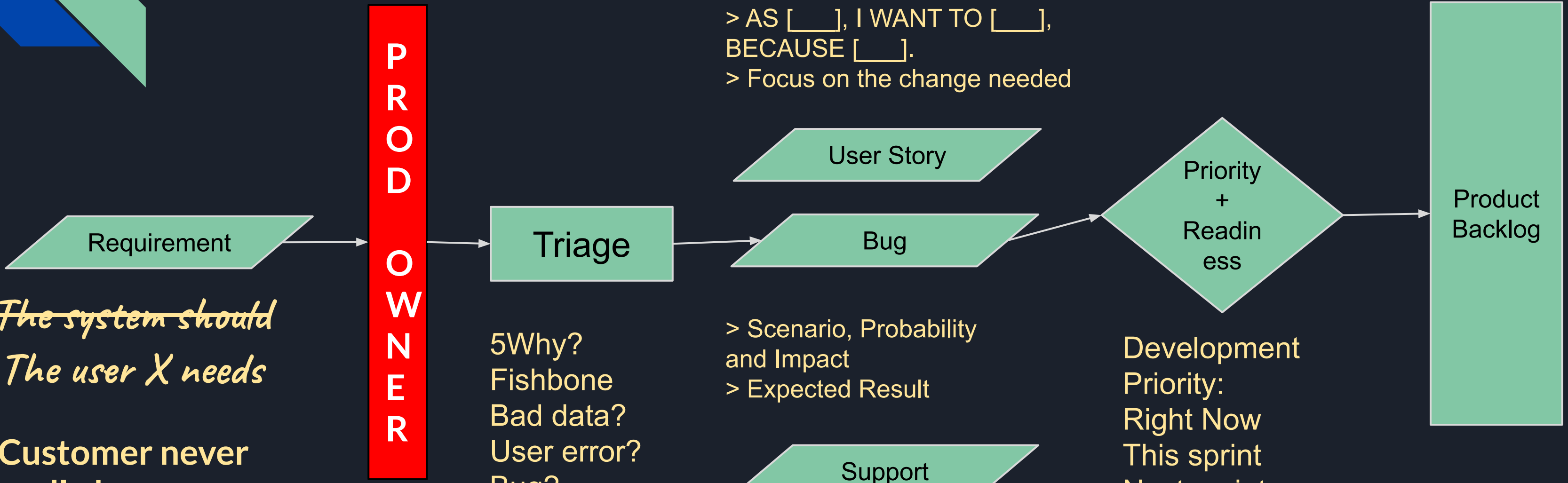
SM	SM	SM
PO	PO	PO
DEV	DEV	DEV
TEST	TEST	TEST

Task	SM	PO	ARC	DEV	TESTER	UX
Create PBI	R	A	R	R	R	R
Set PBI Priority (WHEN)	C	R	C	C	C	C
Ready: Set Estimation (EFFORT)	C	I	C	R	R	R
Ready: User Need Stable (WHAT)	C	R	R/C	C	C	R
Ready: Tech/Test Design Stable (HOW)	C	I	R	R	R	C
Ready: Acceptance Criteria Defined	C	R	C	C	R	C
Sprint Planning	A	R	R	R	R	R
Create PBI Tasks	A	R	R	R	R	R
Back/Forth with Customer	C	R	C	C	C	R
Daily	A	C	C	R	R	R
Develop, Refactor	I	I	A	R	C	I
Identify Tests	C	C	C	R	R	R
Classify Tests (Acceptance, Alternative)	C	C	C	C	R	C
Execute Tests (VERIFY)	C	C	C	R unit/C	R int,regr/A	C
Sprint Review	A	R/A	I	R	R	C
Retrospective	A	C	C	R	R	R
Allow Deploy to Production	C	A	C	C	A	A

R = Responsible / Executes / Participates Actively
A = Accountable (also Approver) / Ensures Execution
C = Consulted / Helps
I = Informed

Development Cycle

DC - Entry



~~The system should~~

The user X needs

Customer never really knows what he wants

5Why?
Fishbone
Bad data?
User error?
Bug?

> AS [___], I WANT TO [___], BECAUSE [___].
> Focus on the change needed

> Scenario, Probability and Impact
> Expected Result

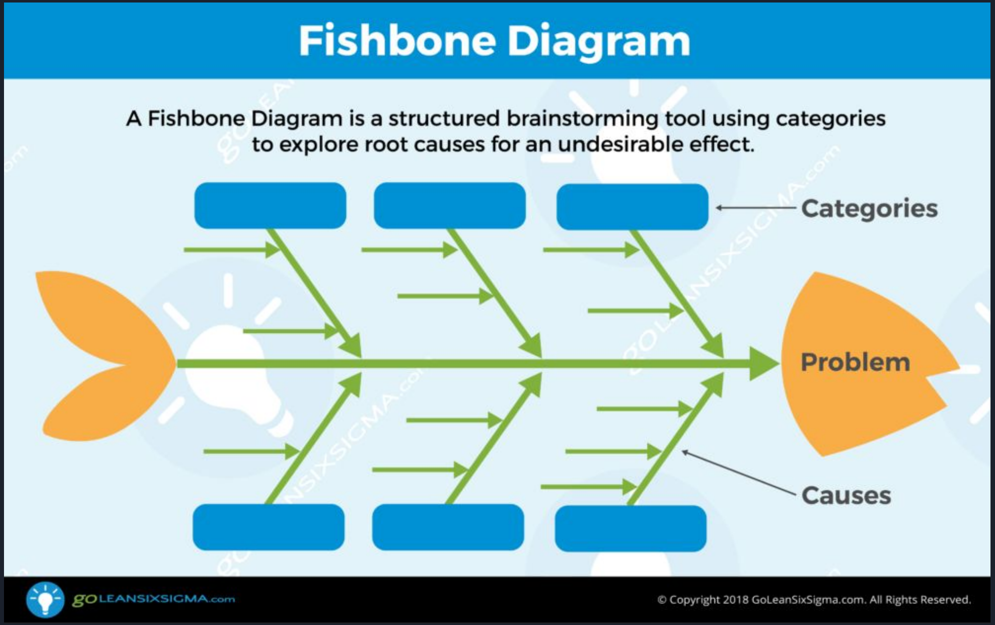
> Data manipulation
> Reprocess messages / documents
> Review of audit logs

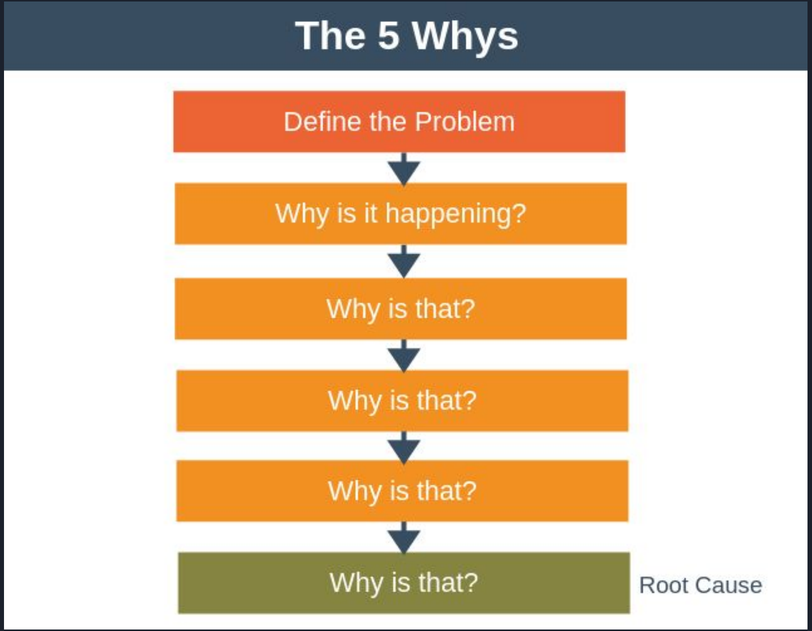
Development Priority:
Right Now
This sprint
Next sprint
To backlog

Most Priority
Most Ready

Least Priority
Least Ready

PBI does NOT need to be 100% detailed. It just needs to be STABLE.


Fishbone Diagram
A Fishbone Diagram is a structured brainstorming tool using categories to explore root causes for an undesirable effect.
Categories
Problem
Causes
goLEANSIXSIGMA.com
© Copyright 2018 GoLeanSixSigma.com. All Rights Reserved.

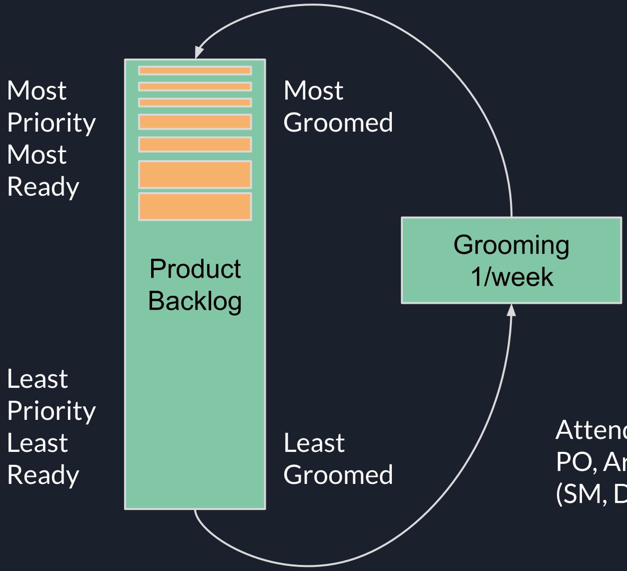

The 5 Whys
Define the Problem
Why is it happening?
Why is that?
Why is that?
Why is that?
Why is that?
Root Cause

“El sistema debería permitir buscar solo los productos activos”

“El logo de free shipping debe aparecer al costado del título del producto”

“El listado de ordenes no funciona” > “porque la fecha estimada de entrega no aparece”

DC - Grooming



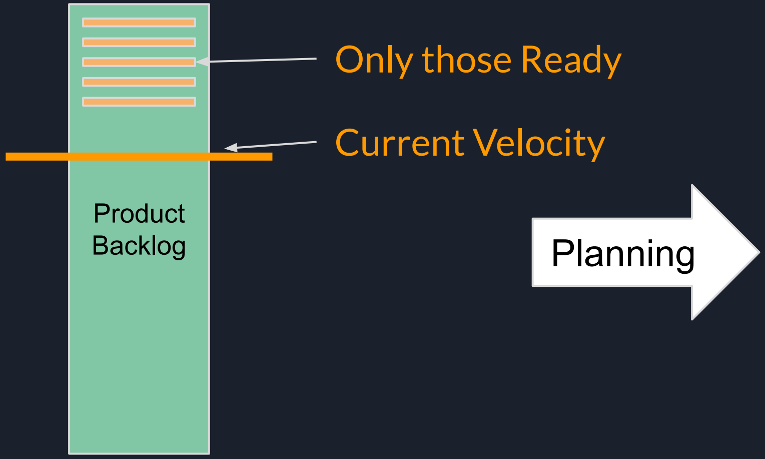
Output:

- Stable User Need
- INVEST Stories
- Draft Tech Design
- Draft Test Design
- Draft estimation
- Backlog Priority Reviewed

Attendees:
PO, Architect, Test Lead
(SM, Dev, Tester)

TO REGISTER IN JIRA

DC - Sprint Planning



Sprint Backlog

User Story 1
X hours =
R + D + TD + T + Doc + Perf

Research, Development, Test Design, Testing, Documentation, Performance Test

Priority = User Priority + Dev Priority + IsPrerequisite

User Story 2

User Story 3

User Story 4

User Story 5

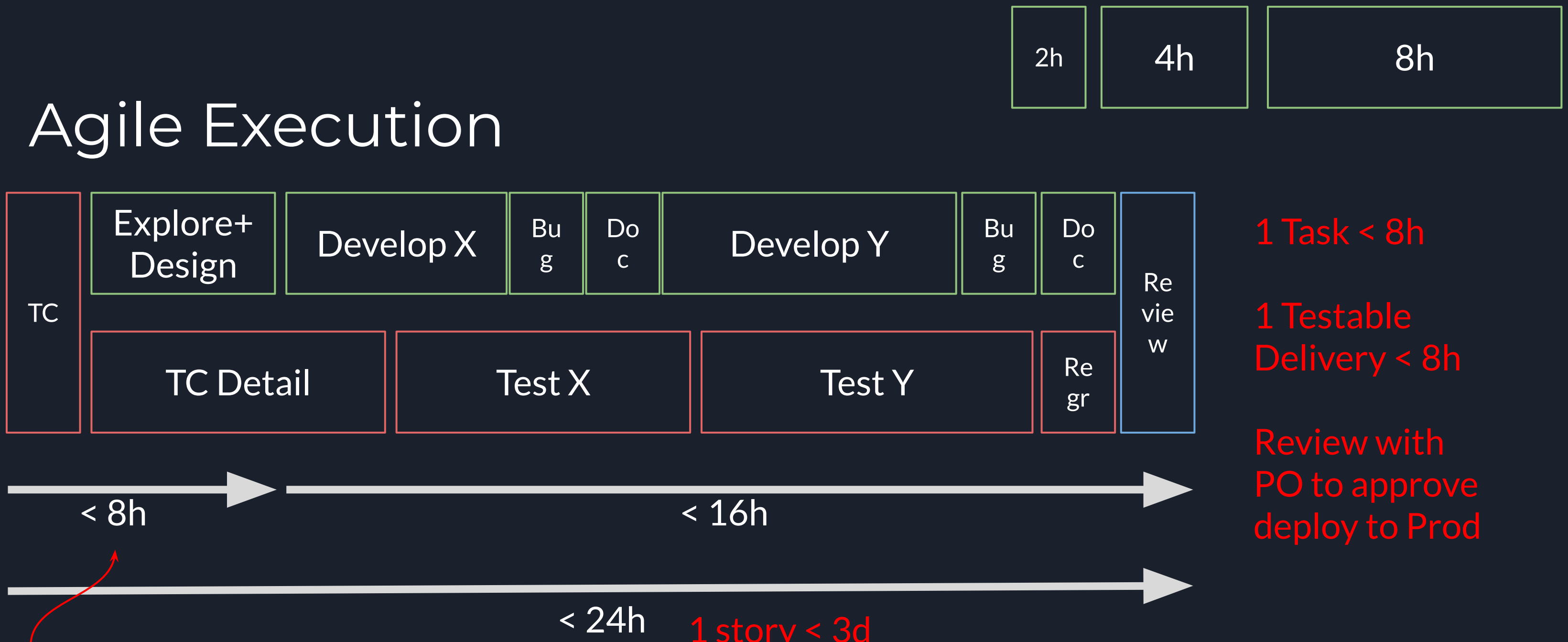
User Priority vs Development Priority

- Sometimes the same, sometimes not.
- User Priority: His own priority based on his usage pattern, daily tasks, and value scale.
- Development Priority: Pareto based, Many vs one user, probability based, impact on acceptance flows, roadmap importance

User Priority
High
Medium
Low

Dev Priority
P0 Showstopper / Critical (only for hotfix, downtime)
P1 High Impact on Acceptance / Must have
P2 Med Impact on Acceptance / High impact on Alternate / Need to have
P3 Low Impact on Acceptance / Med impact on Alternate / Nice to have
P4 Low impact on Alternate / Nice to have

Agile Execution

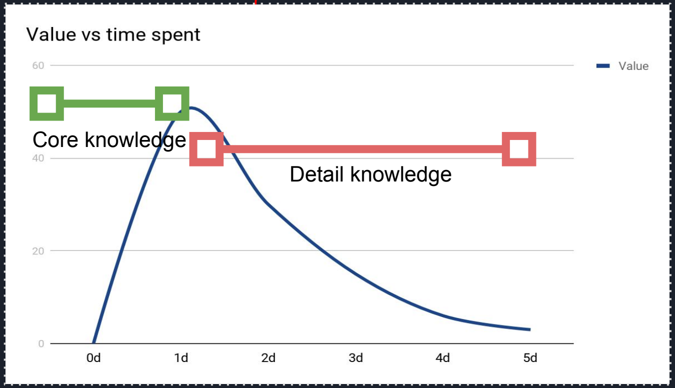


1 Task < 8h

1 Testable Delivery < 8h

Review with PO to approve deploy to Prod

1 story < 3d



New Idea? Improve? Delete?
Functional: Talk to the PO
Technical: Talk to the Architect
> Discuss, Agree, Prioritize ...

Bug?
Tester: Reproduce
> Prioritize ...

JIRA

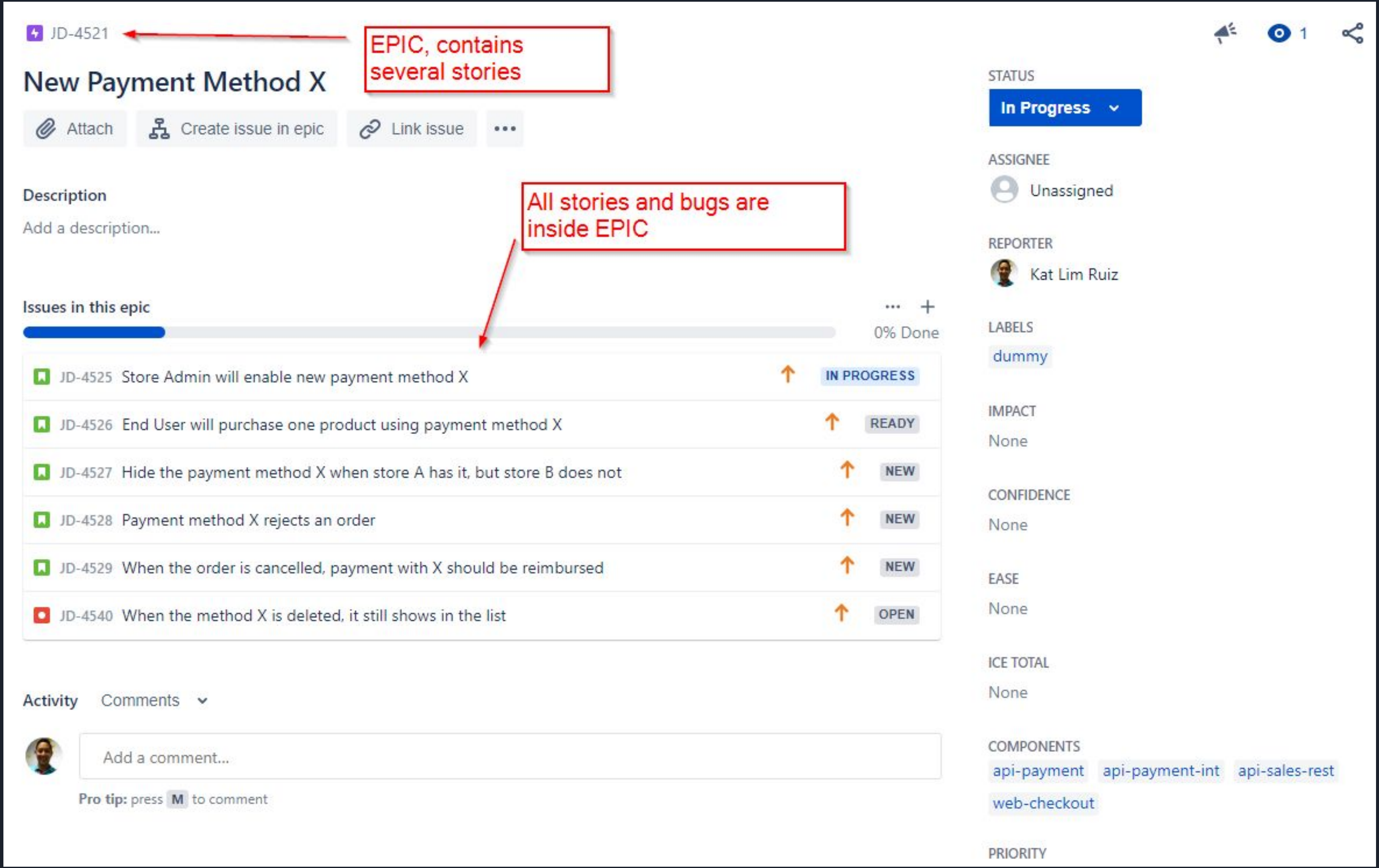
- Epic ← Level 0
  - Story
  - Bug
- Story ← Level 1
  - Subtask
  - Test ← Level 2
- Bug
  - Subtask
  - Test
- Support

Sprint

- Story
- Bug
- Support

Epic spans multiple sprints.

Story or Bug does not need an Epic.



JD-4521

EPIC, contains several stories

New Payment Method X

Attach · Create issue in epic · Link issue · •••

Description

Add a description...

All stories and bugs are inside EPIC

Issues in this epic

0% Done

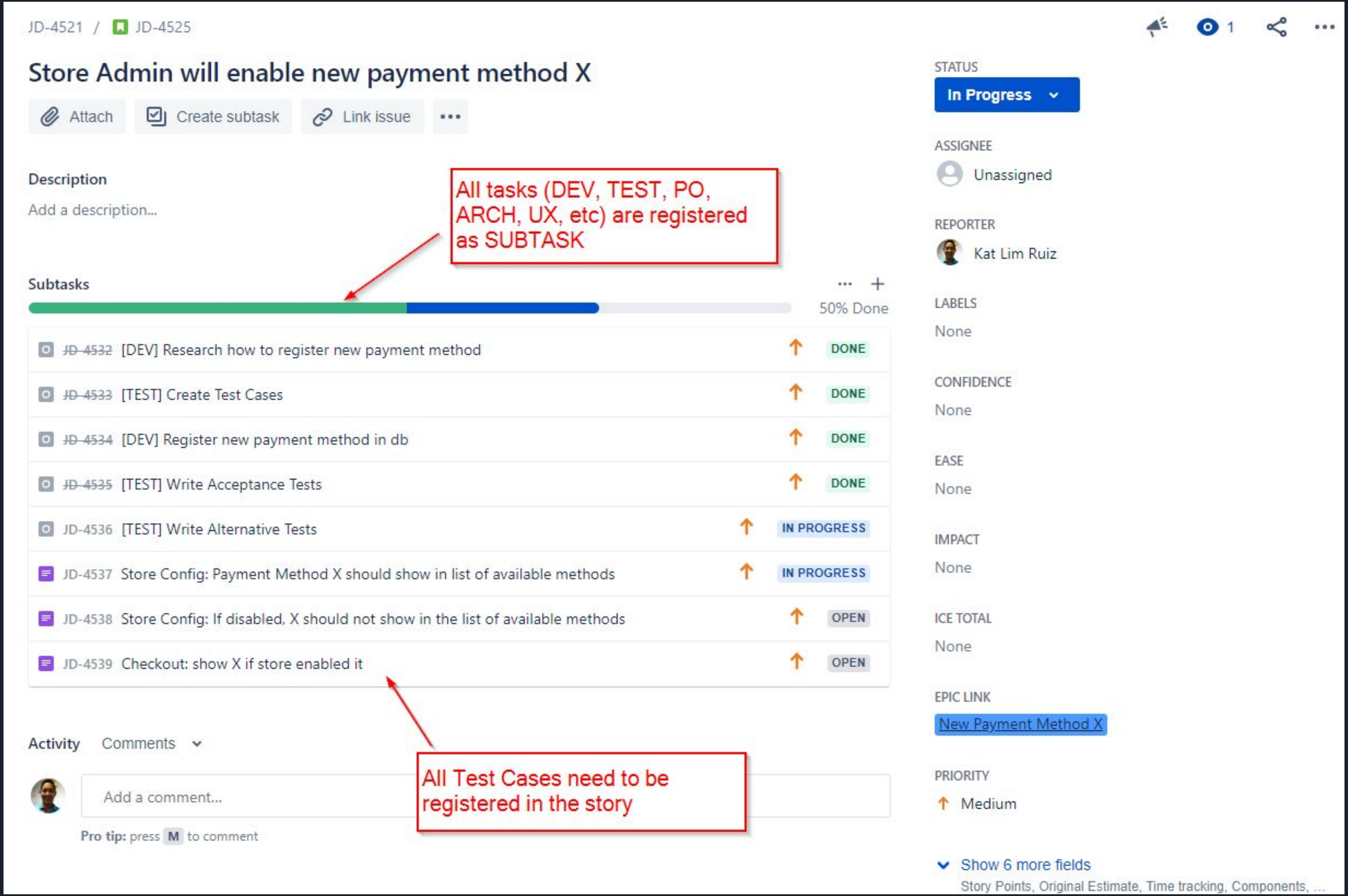
JD-4525	Store Admin will enable new payment method X	↑	IN PROGRESS
JD-4526	End User will purchase one product using payment method X	↑	READY
JD-4527	Hide the payment method X when store A has it, but store B does not	↑	NEW
JD-4528	Payment method X rejects an order	↑	NEW
JD-4529	When the order is cancelled, payment with X should be reimbursed	↑	NEW
JD-4540	When the method X is deleted, it still shows in the list	↑	OPEN

Activity · Comments

Add a comment...

Pro tip: press M to comment

STATUS	In Progress
ASSIGNEE	Unassigned
REPORTER	Kat Lim Ruiz
LABELS	dummy
IMPACT	None
CONFIDENCE	None
EASE	None
ICE TOTAL	None
COMPONENTS	api-payment, api-payment-int, api-sales-rest, web-checkout
PRIORITY	


JD-4521 / JD-4525

Store Admin will enable new payment method X

Description

Add a description...

All tasks (DEV, TEST, PO, ARCH, UX, etc) are registered as SUBTASK

Subtasks

50% Done

~~JD-4532~~	[DEV] Research how to register new payment method	DONE
~~JD-4533~~	[TEST] Create Test Cases	DONE
~~JD-4534~~	[DEV] Register new payment method in db	DONE
~~JD-4535~~	[TEST] Write Acceptance Tests	DONE
JD-4536	[TEST] Write Alternative Tests	IN PROGRESS
JD-4537	Store Config: Payment Method X should show in list of available methods	IN PROGRESS
JD-4538	Store Config: If disabled, X should not show in the list of available methods	OPEN
JD-4539	Checkout: show X if store enabled it	OPEN

All Test Cases need to be registered in the story

Activity Comments

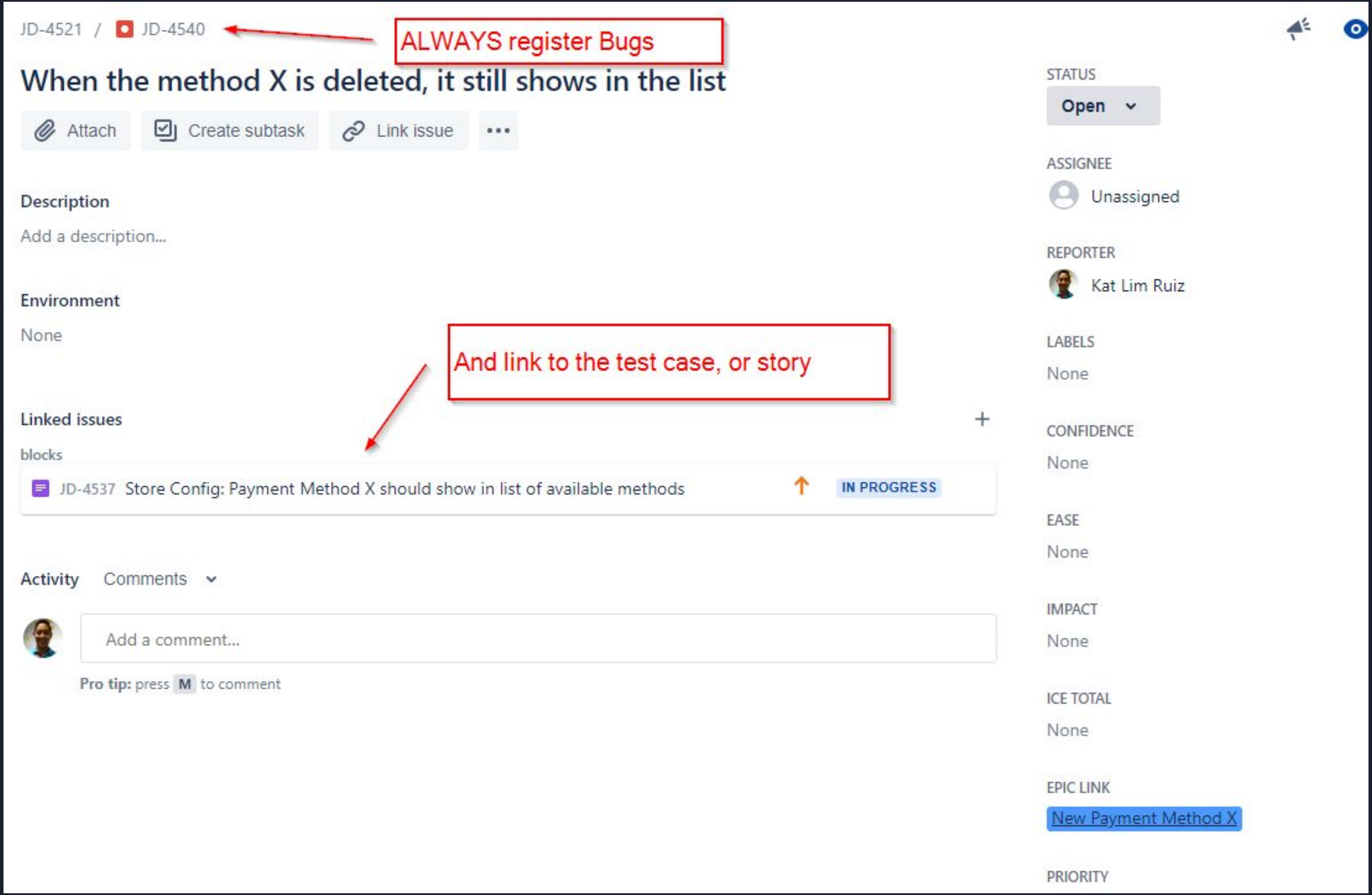
Add a comment...

Pro tip: press M to comment

STATUS	In Progress
ASSIGNEE	Unassigned
REPORTER	Kat Lim Ruiz
LABELS	None
CONFIDENCE	None
EASE	None
IMPACT	None
ICE TOTAL	None
EPIC LINK	New Payment Method X
PRIORITY	Medium

Show 6 more fields

Story Points, Original Estimate, Time tracking, Components, ...


JD-4521 / JD-4540

ALWAYS register Bugs

When the method X is deleted, it still shows in the list

Attach | Create subtask | Link issue | •••

Description

Add a description...

Environment

None

And link to the test case, or story

Linked issues

blocks

JD-4537 Store Config: Payment Method X should show in list of available methods IN PROGRESS

Activity Comments

Add a comment...

Pro tip: press M to comment

STATUS

Open

ASSIGNEE

Unassigned

REPORTER

Kat Lim Ruiz

LABELS

None

CONFIDENCE

None

EASE

None

IMPACT

None

ICE TOTAL

None

EPIC LINK

New Payment Method X

PRIORITY

Mindset

- **THINK ABOUT SCALE (EXECUTION)**
- Scale to work at high performance (high speed all the time)
- Scale to be used by many users at the same time (high concurrency)

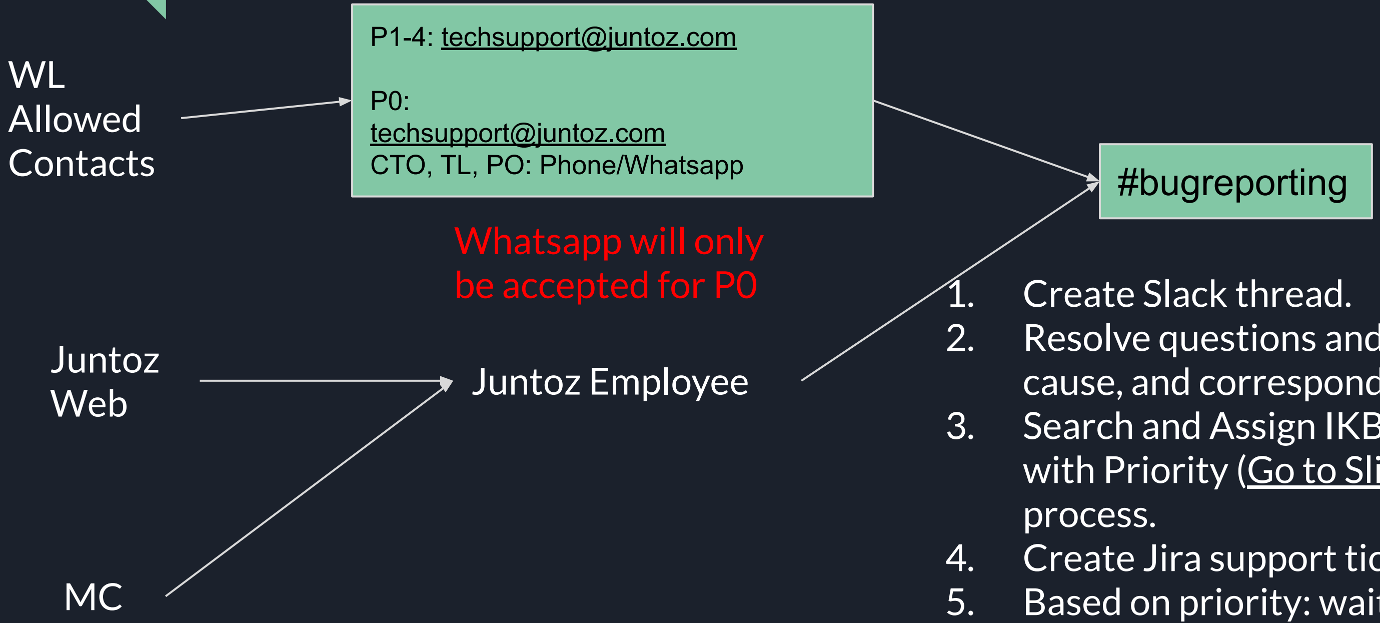
- **THINK ABOUT 80% OF USERS**
- You design for the mass, not for only a few.
- You prioritize for the mass, not for only a few.
- The LESS documentation and training is needed, the better.
- Do not develop something that the user has not asked or agreed. We want to develop something useful. Not something that will not be used or is too complex to use.

- **MAKE A DECISION AS LATEST AS RESPONSIBLY POSSIBLE**
- This also mean **DO NOT DEVELOP OR TEST** something that is not going to immediately be used.
- When designing a feature, do consider the PROBABLE future, not the possible.

- **NEVER MAKE ASSUMPTIONS**
- Always ask, get feedback, get different opinions, ask to the right people.
- First DIVERGE to CONVERGE into a solution.
- Remember: you are the expert to shape the requirement, you know UX more than the user, you know how to deploy a product, you know how to design a software, etc.

Support Cycle

SC - Entry and Execution


WL Allowed Contacts
P1-4: techsupport@juntoz.com
P0: techsupport@juntoz.com
CTO, TL, PO: Phone/Whatsapp
Whatsapp will only be accepted for P0
Juntoz Web
MC
Juntoz Employee
#bugreporting

Tech L1 resolves or escalates
Tech L2 resolves

1. Create Slack thread.
2. Resolve questions and doubts. Get to root cause, and corresponding solution.
3. Search and Assign IKB number which comes with Priority (Go to Slide) and Escalation process.
4. Create Jira support ticket.
5. Based on priority: wait or execute. Reply to #bugreporting.
6. Execute (after queue time).
7. Final Reply to #bugreporting.

SC - Mandate

- Standardized answers to standardized questions/problems
    - Anything outside of standard is escalated.
    - E.g. Registration of new KIB, script that does not fit the standard answer, etc.
    - DO NOT GO BEYOND THE STANDARD, even if you know how.
    - The answer is to ESCALATE and then to STANDARDIZE.
- Support has more priority over development. It is assumed the tasks are quick to execute and non-controversial.
- If what is being asked, falls OUTSIDE of these parameters, ESCALATE.
- IKB = Incident Knowledge Base. Our library of support cases where we document and establish the procedure of each case: causes, possible solutions, procedures.

SC - SLA (TBD)

SLA is for response, not resolution.

P0: Immediate response, asap resolution

P1: 1h max response, asap resolution

P2: 4h max response, 1 work day max resolution

P3: 8h max response, when possible

P4: 16h max response, when possible