TESTING IN AGILE

HELEN VOROBEI

OCTOBER, 2017

INTRODUCTION

Helen Vorobei

Quality Architect,
Testing Competency Center Expert
10 years in testing

E-mail: Helen_Vorobei@epam.com

AGENDA OF THE TRAINING

1. Agile Values: what they mean and how work in reality
2. Before Sprint activities: what should be done and popular issues
3. In Sprint activities : what should be done and popular issues
4. After Sprint activities : what should be done and popular issues


AGILE VALUES

AGILE VALUES

Individuals and interactions

over processes and tools

Working software

over comprehensive documentation

Customer collaboration

over contract negotiation

Responding to change

over following a plan

AGILE VALUES AND REALITY

AGILE VALUE

- Individuals and Interactions Over Processes and Tools

WHAT IT MEANS

- Small team, collocated, PO co located with the team
- Cross functional, help each other
- Team is empowered to make decisions

WHAT HAPPENS IN REAL WORLD

- Team is distributed, time zone difference can be 10 hours
- PO from customer side and hard to reach for the team
- Team includes fill-time allocated developers and testers, but designers, dev-ops, automation in other teams

AGILE VALUES AND REALITY

AGILE VALUE

- Working product Over comprehensive requirements

WHAT IS MEANS

- Ready (i.e. Tested) product at the end of sprint

WHAT HAPPENS IN REAL WORLD

- Developers write code till the last moment of the sprint
- Testing can’t be completed in sprint
- Some testing types are performed out of sprints (regression, integration, etc.)

AGILE VALUES AND REALITY

AGILE VALUE

- Customer collaboration Over contract negotiation

WHAT IS MEANS

- Team defines what it will commit to deliver at the end of the sprint
- Requirements evolve, but timescales are fixed

WHAT HAPPENS IN REAL WORLD

- Customer presses on the team the scope and timeline
- And changes requirements within sprint
- And provides not well defined requirements

AGILE VALUES AND REALITY

AGILE VALUE

- Responding to Change Over Following a Plan

WHAT IS MEANS

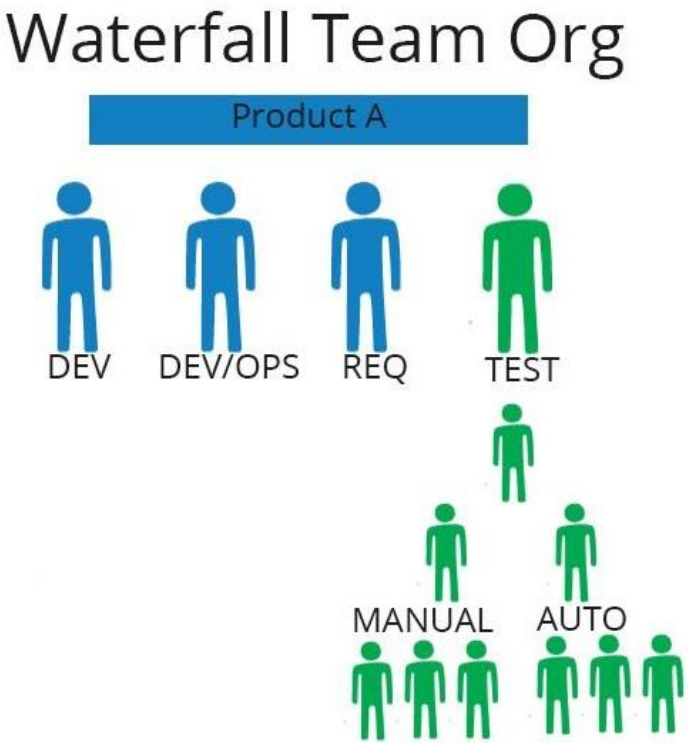
- Sprint Retrospective – learn and improve, how to become more effective
- Welcome changing requirements, even late in development

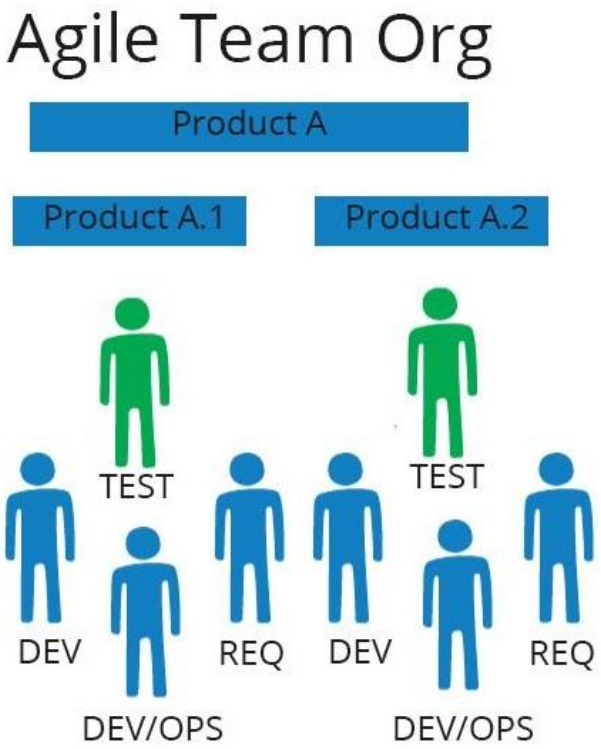
WHAT HAPPENS IN REAL WORLD

- Team hesitates to say openly about problems
- To much pressure to deliver to have time for retrospectives!
- We have retrospectives, but nothing changes!
- Requirement are being changed continuously because not ready before development starts


SPRINT
Superdry
Superdry

HOW TESTER ROLE CHANGES IN AGILE





HOW TESTER ROLE CHANGES IN AGILE

Traditional approach

Testers **detect** the **differences** between **existing** and **required** conditions

Agile approach

prevents

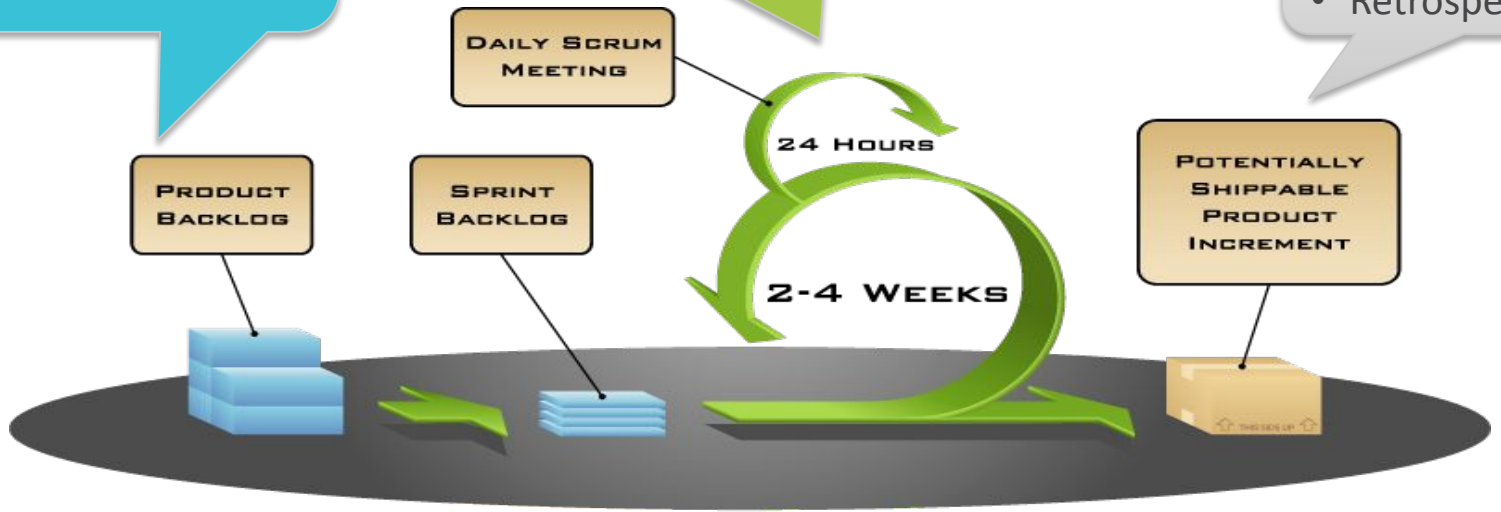
Testers ~~**detect**~~ the **differences** between **existing** and **required** conditions

WHERE TESTERS CAN REALLY INFLUENCE QUALITY?

Before sprint:
- Grooming
- 3 Amigo sessions
- Planning

In sprint:
- Test as soon as feature ready
- Collaborate with developers
- Move all testing types in sprint
- Define Testing Pyramid
- Control Definition of Done

At the end of the sprint:
- Retrospective



BEFORE SPRINT ACTIVITIES

CLARIFYING REQUIREMENTS – OPTION 1: GROOMING/ SPRINT REFINEMENT/SCRUM GUIDE

When	Before a sprint, even 1 week before
Input	User Stories created and described by PO
Who participates	All team, dev and test, PO
What testers do before grooming	Analyze user stories from the point of: • What information is missing and will prevent us from testing? • What is strange /not logical/contradicts other requirements? • Do I know how to test this? • What is missing to test this? (e.g. data)
On grooming	Ask questions by user stories Clarify PO answers Plan if needed additional discussions (e.g. with architecture)

CLARIFYING REQUIREMENTS OPTION 2 - 3 AMIGOS SESSIONS

Define 3 Amigos from each team to discuss user stories:

- **Business Analysis** or Product Owner -What problem are we trying to solve?
- **Developer** -How might we build a solution to solve that problem?
- **Tester** -What about this, what could possibly happen?

When it works better:

- Pre-grooming sessions
- Discuss high-level requirements
- When team capacity is very limited and can’t invite the whole team

WHAT CAN GO WRONG?

WHAT CAN HAPPEN?	WHAT CAN WE DO?
• Not enough stories exist/have details in the backlog before grooming • PO does not know what the purpose of the story or details • PO is not ready to answer to questions • PO does not come on grooming	• Plan grooming in advance • Prepare and share questions with PO before session • Push to “move out of sprint” not clear stories on planning
• Team works with several POs and they contradict each other • PO changes opinion a bit later • Requirements are changed on the fly	• Store PO’s answers in common source • Have answers recorded • Measure and communicated impact of changes

PLANNING

When	The 1-th day of a sprint
Input	User stories with clarified requirements (after grooming)
Who participates	All team, dev and test, PO
What testers do	• Estimate testing for each story • Discuss if testing seems not proportional to development/ too complex • Propose technical debt stories • Create tasks
Output	Sprint backlog: estimated stories that team commits to deliver at the end of sprint

ESTIMATES: WHAT IF

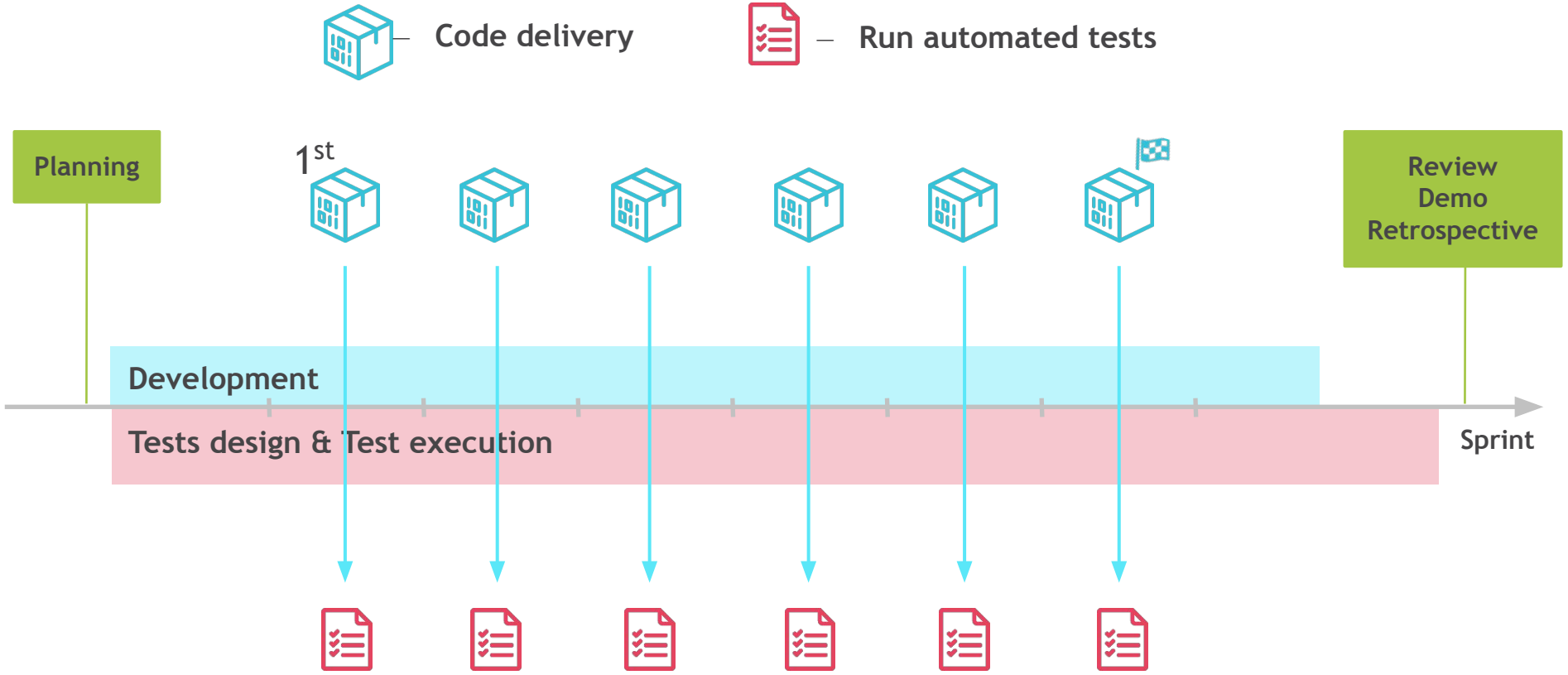
WHAT CAN HAPPEN?	WHAT CAN WE DO?
• Team cannot estimate the user story	• Check that user story satisfies INVEST criteria • Clarify requirements again • Break story on smaller pieces • Consult with architect or any other experts
• Team members have great differences between estimates	• Discuss what are included in estimates by team members • Ask to explain estimates
• Customer presses for lower estimates	• Provide detailed estimates • Explain the risks of estimates reduction – there won't be any stories for delivery at the end of sprint, because team can not complete all tasks, quality risks – will be a lot of defects


Tester goal: provide immediate feedback to developers!
IN SPRINT ACTIVITIES

HOW CAN WE FIND DEFECTS EARLIER OR PREVENT THEM?

- Get clear requirements after grooming/ 3 amigos sessions
- Collaborate with developers:
  - Explain what you are going to test, show your checklists/ test cases
  - Clarify together any questions regarding to user stories
  - Propose “good” development practices to use: unit tests, code review, coding standards, etc.
  - Agreed about continuous deployment to test feature as soon as it is ready, not waste time for waiting new builds
- Don’t postpone “special” testing type till the end of release

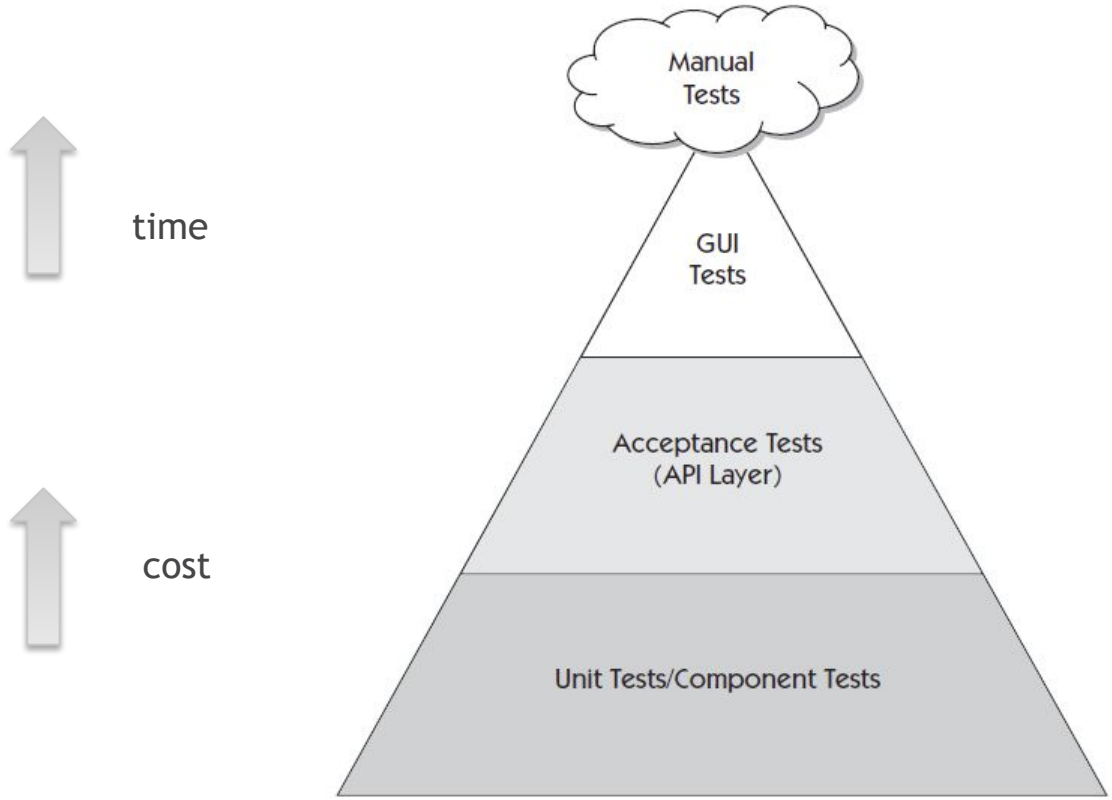
IDEAL TESTING TIMELINE IN SPRINT



TO BUILD IDEAL TESTING TIMELINE WE NEED

- Define Testing Pyramid
- Integrate auto tests into CI/CD pipeline
- Execute frequently:
  - Unit tests - after each commit,
  - Smoke tests – after build deployment,
  - Regression tests – nightly (on demand or by schedule)

TESTING PYRAMID



HOW MANY PROJECTS HAVE A “TESTING PHASE” AFTER SPRINTS?

WHAT TESTING TYPES ARE OFTEN OUT OF SPRINT?

- Regression testing
- Integration testing
- Compatibility testing
- Mobile testing
- Performance testing

HOW MOVE THEM IN SPRINT

- Build Testing Pyramid, include auto tests in CI/CD
- Test integration with mocks
- Agree with separate teams about time readiness of 3-d party component
- Create compatibility matrix
- Divide tests with mobile/browser specific or not
- Distribute functional and regression tests between browsers and devices
- …

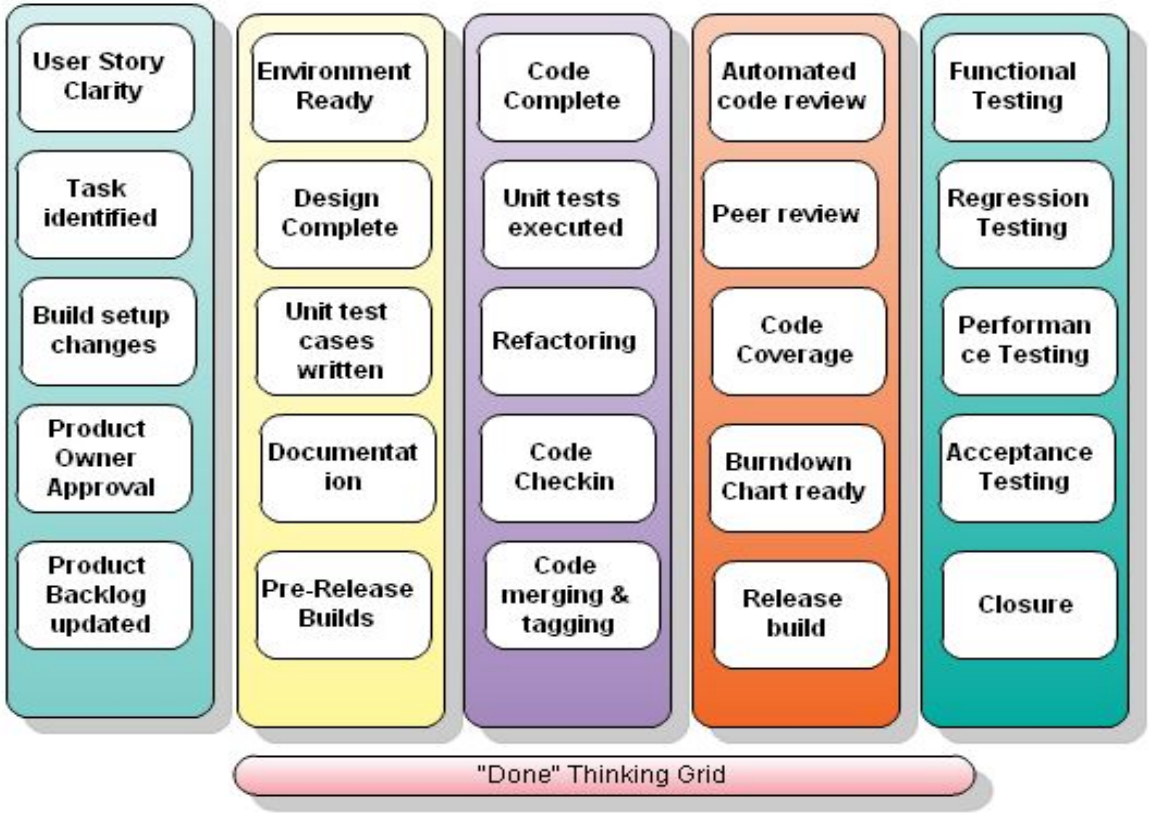
WHAT CAN HAPPEN DURING SPRINT?

WHAT CAN HAPPEN?	WHAT CAN WE DO?
• Test environment is down	• If it often happens, set up back-up test environment • If it is caused issues with build deployment => CI/CD, automation tests, unit tests • Discuss with dev lead/ DM/Customer. Total time we lose is the strongest argument
• No test data	• How can we get as production-like data as possible?) (replication, sub setting , **anonymization**, generation)
• Dependences from other teams/systems	• Develop mocks and stubs • Agree with other teams about delivery dates

WHAT CAN HAPPEN DURING SPRINT?

WHAT CAN HAPPEN?	WHAT CAN WE DO?
• Developers deliver scope till the end of the very last day	• Agree day when the first developed stories will be delivered for testing • Push developers to deliver changes smoothly • Agree X days or hours before end of sprint when all stories will be delivered • Reduce team's capacity by the number of points not achieved
• A lot of defects in the functionality	• Unit tests, automation tests in CI/CD pipeline run before build deployment • Test locally • Define bug root cause: environment issues, requirements issues, code issues (unit tests are passed?)

DEFINITION OF DONE



DEFINITION OF DONE FOR USER STORY

- Code completed and checked in
- Code review done
- Unit tests created and passed
- Tests created and passed on all envs
- Bugs fixed and verified

DEFINITION OF DONE FOR SPRINT

- All stories done
- Regression testing done
- Integration testing done
- Performance testing done
- Build documentation prepared
- Etc.


AFTER SPRINT ACTIVITIES

RETROSPECTIVE MEETING: WHAT AND WHY

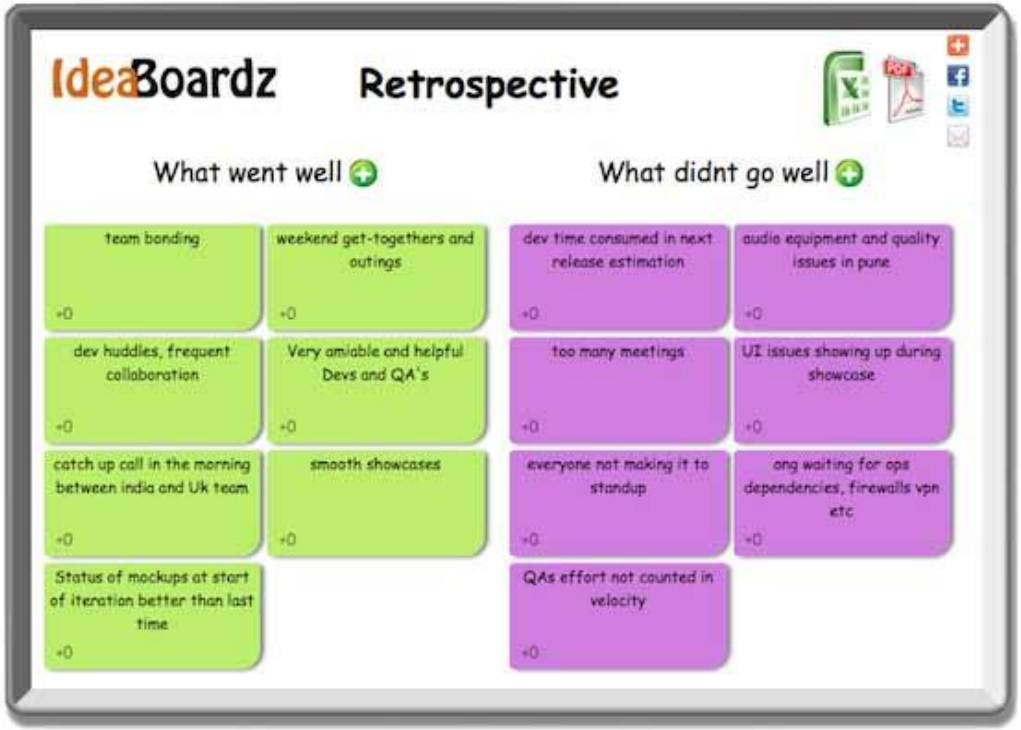
Retrospective is the meeting where the team discusses what **could be changed** that might make the **next sprint more value**

Why retrospective is important?

1. Help to identify team's problems if there are or areas for improvements
2. Better way to solve issues or do improvements
3. Motivate the team: team feels power to change process as they want
4. Allow to say thank you to team members and share best practices within the team
5. Team building

RETROSPECTIVE BOARD

- Write on stickers
  - what was well
  - what need to drop
  - what to improve
- Put on the board
- Clarify all items
- Vote for items that should be improved first



HOW DO RETROSPECTIVE EFFECTIVELY?

- Do not play the blame game
    - Talk about facts, not about people
- Focus not on problems, focus on solution
- Be prepared
    - Analyze sprint activities in advance (All was delivered in time? All was done in time? Everything was done?)
    - Not only raise the problems, propose solutions
- Use facts and numbers
    - For example, delivery was delays on ..days

WHAT IMPORTANT TO CONSIDER?

- Were new features delivered regularly, not in the end?
- Where there blocker/critical issues?
- Did the team have downtime during sprint?
- Burndown: planned and real lines – are they near each other?
- Team velocity: did team do all tasks that planned? Can team do more? How velocity compares to the previous sprint?
- Was all testing done in sprint?
- All scrum ceremonies were done: grooming, planning, daily stand ups, review and demo?
- What was making our work hard?

RETROSPECTIVE: WHAT IF?

WHAT CAN HAPPEN?	WHAT CAN WE DO?
• Team hesitate to say openly about the problems	• Remind the team about the goal of retro: find solution, not guilty • Organize retro in the way when everyone has a chance to say/write his/her opinion
• Too much pressure to deliver, no time for retrospectives!	• First items for discussion on meeting! • Think about action items together
• We have retrospectives, but nothing changes	• During retro team is voting for actions that should be done next sprint and include them into story board • For each action – owner and timeline (even small step in the right direction) • Start retro with reviewing actions that should be done from previous retro

SUMMARY: YOU INFLUENCE QUALITY

In the real world	Your response
• Team is distributed, time zone difference can be 10 hours • PO from customer side and hard to reach for the team • Team includes developers, testes, designers, dev-ops	• Set up all required team meetings in advance/just keep timeslot for possible questions • Use web cams to watch team on the meetings • Agree time slots with PO for the team questions/meetings
• Developers write code till the last moment of the sprint • Testing can't be completed in sprint • Some testing types are performed out of sprints (regression, integration, performance)	• Agree with developers about first/last delivery days • Delivery features iteratively • Collaborate with developers • Do not delivery sprint scope that untested • Try to move all testing types in sprint

SUMMARY: YOU INFLUENCE QUALITY

In the real world	Your response
• Customer presses on the team the scope and timeline • Requirements are changed within sprint • Requirements are not well defined	• Show team velocity and estimates for the work • Explain the risks of estimates reduction • During planning save some team capacity for such urgent requests • Agree to take in sprint instead of some sprint stories
• Team hesitates to say openly about problems • To much pressure to deliver to have time for retrospectives! • We have retrospectives, but nothing changes!	• Remind the team about the goal of retro • Organize retro in the way when everyone has a chance to say/write his/her opinion • During retro team is voting for actions that should be done next sprint • If these actions are too large, divide them to small steps and define which steps will be done next sprint • Start retro with reviewing actions that should be done from previous retro


Any
Questions

HOMEWORK

- Describe your testing process
- Mark if it is Agile or not
- Propose improvements where it is needed Or explain why they don not need

Example can be found in Excel spreadsheet like here:

Testing process	Description	Agile or not	Improvements
Estimating	Testers don't provide estimates on testing, usually 50% of developer's estimates allocated on testing	Not-agile	1. During a couple of sprints check estimates accuracy (estimates/spent time ratio) 2. Collect statistics about testing delays and overtimes by last sprints or going to UAT/ demo without testing completion 3. Discuss during retrospective that testers estimates should be taken into considering during sprint planning