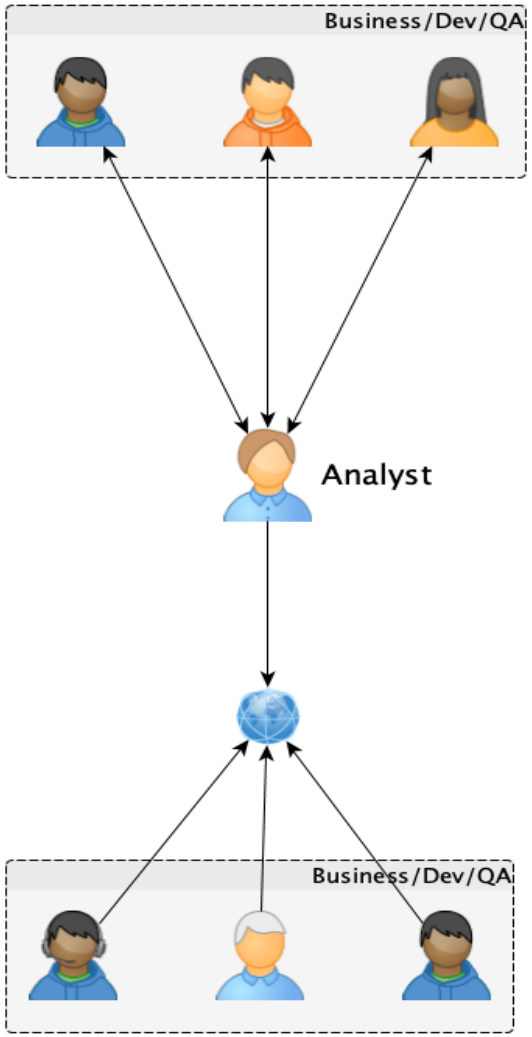
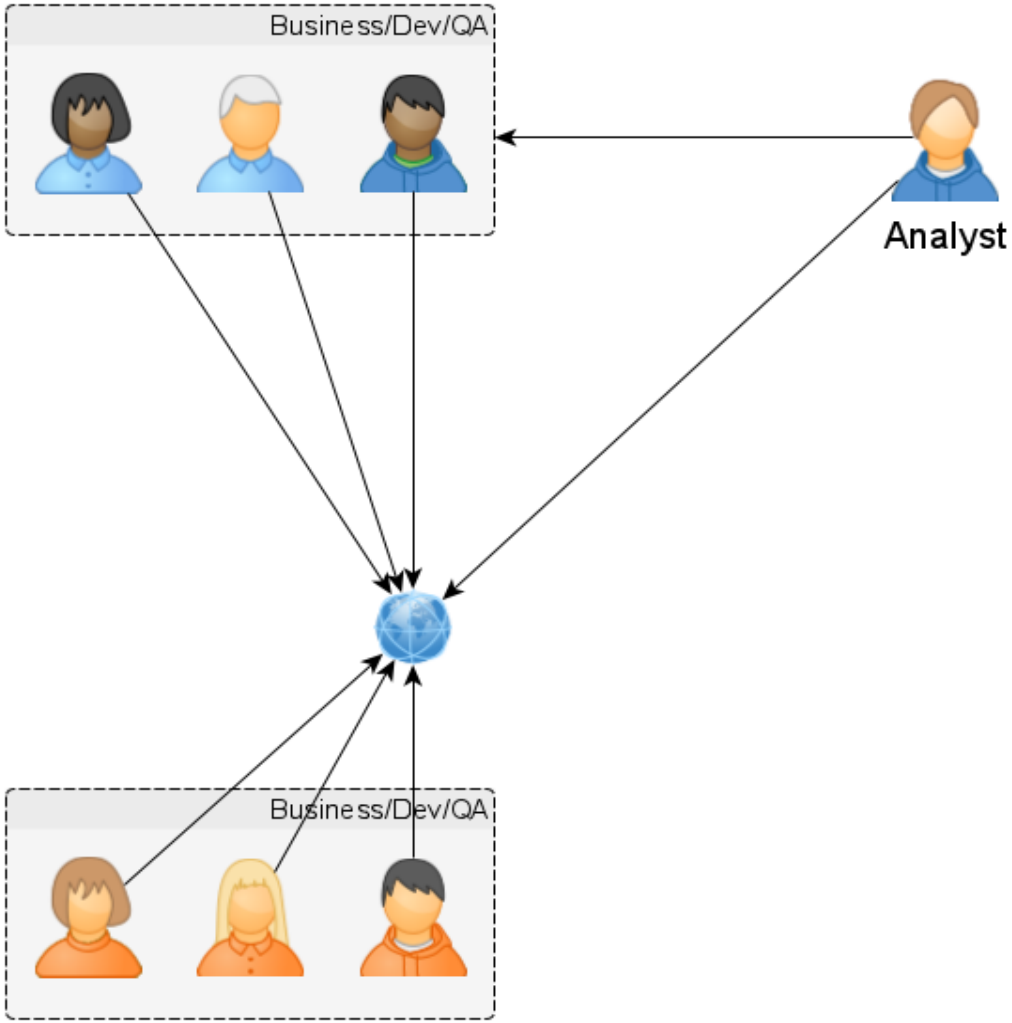
Concept of Analysis

Pavel Ozerov

Specification vs Living Documentation





Disadvantages of Specification

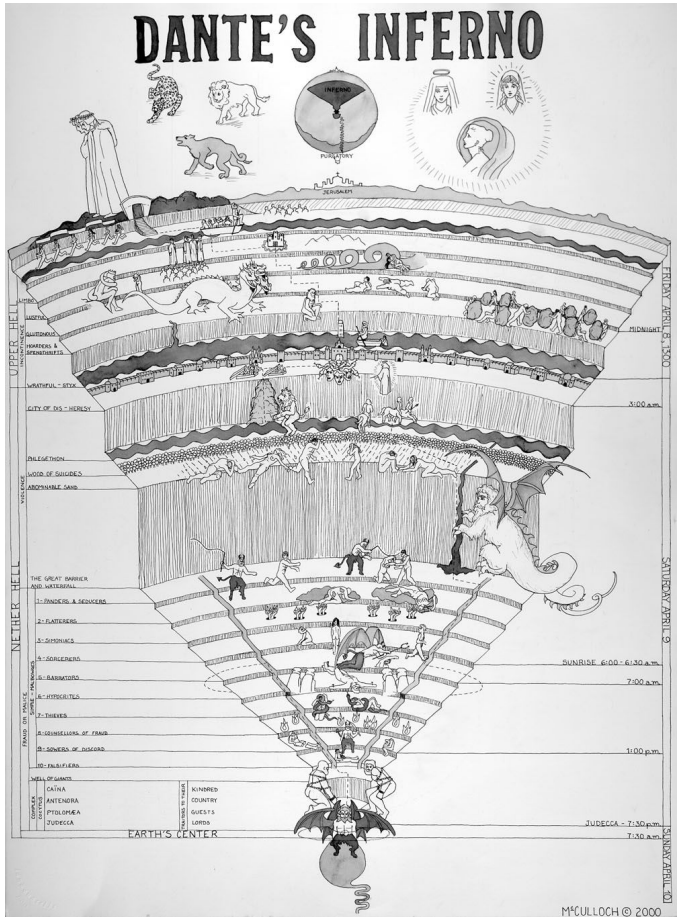
- Analyst is a bottleneck
- Slow speed of changes
- Two different queues that are always full of tasks. The one has TODO things, which can only be handled step-by-step. Another one with things-to-be-changed based on a late feedback from team

Advantages of Living Documentation

- Instead of making changes by oneself, analyst collects all the needed information, structures it and creates a shared place, where all involved people can work
- Everybody is responsible for the documentation and knowledge sharing

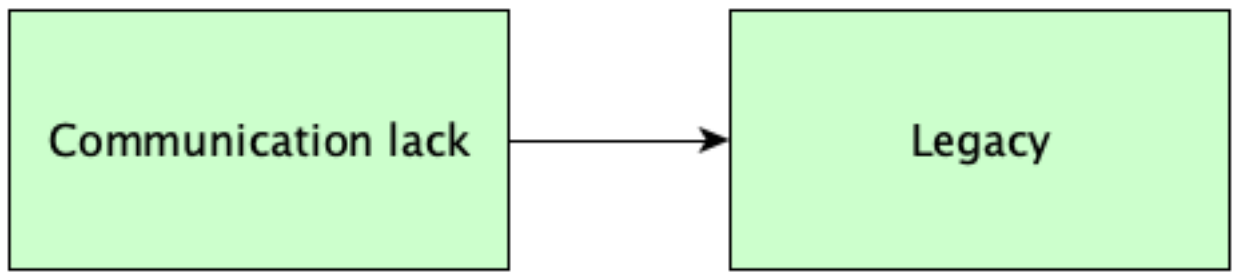
Why BDD is a problem solver?

Tower of Babel (DNA of Legacy)



If something is **not structured**, it's probably **not documented**, which means that it **does not exist**

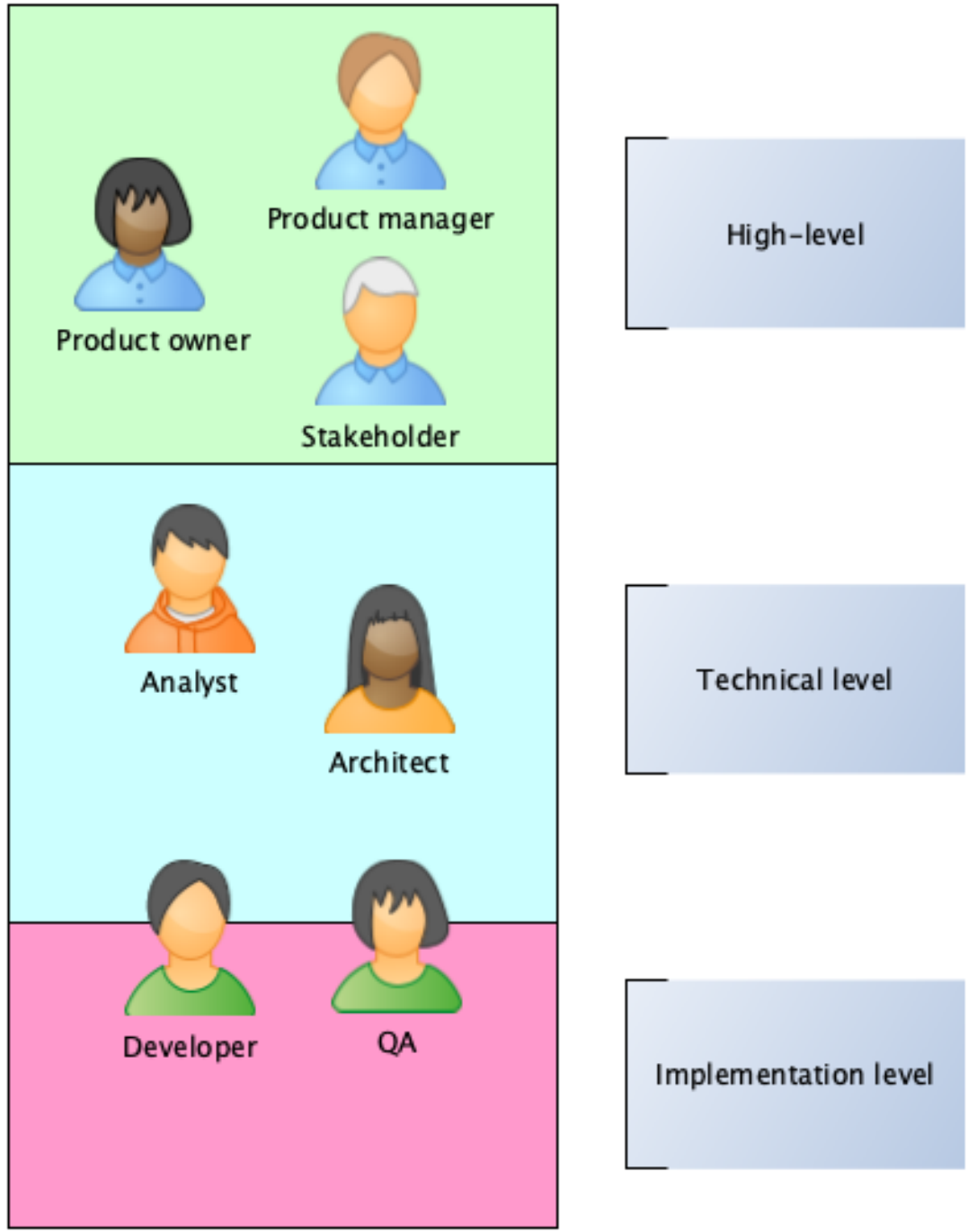
What is the root cause of Legacy?


Communication lack
Legacy

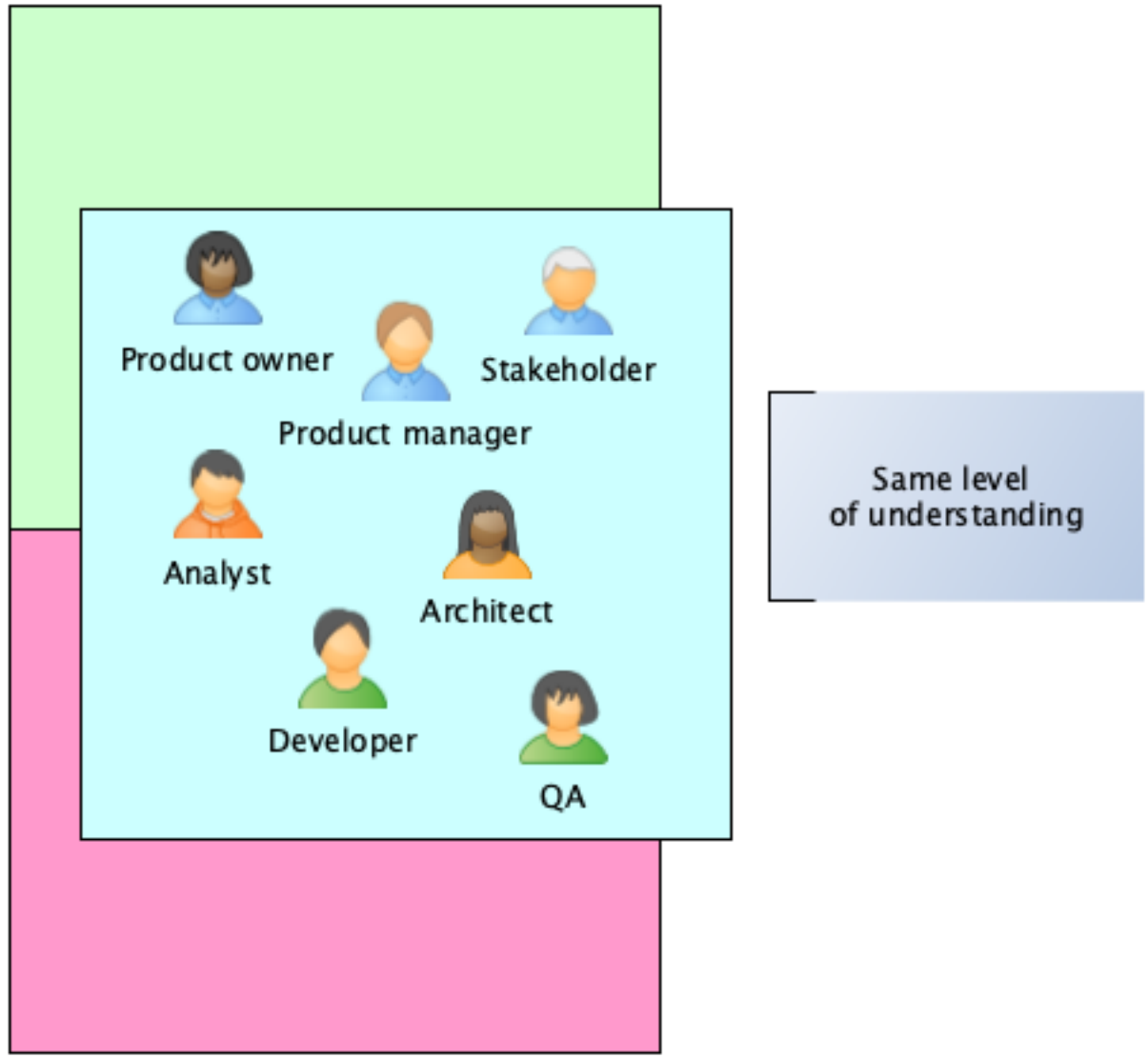
What is the root cause of communication lack?


Different understanding
Communication lack
Legacy

Why people understand things differently?


Product manager
Product owner
Stakeholder
Analyst
Architect
Developer
QA
High-level
Technical level
Implementation level

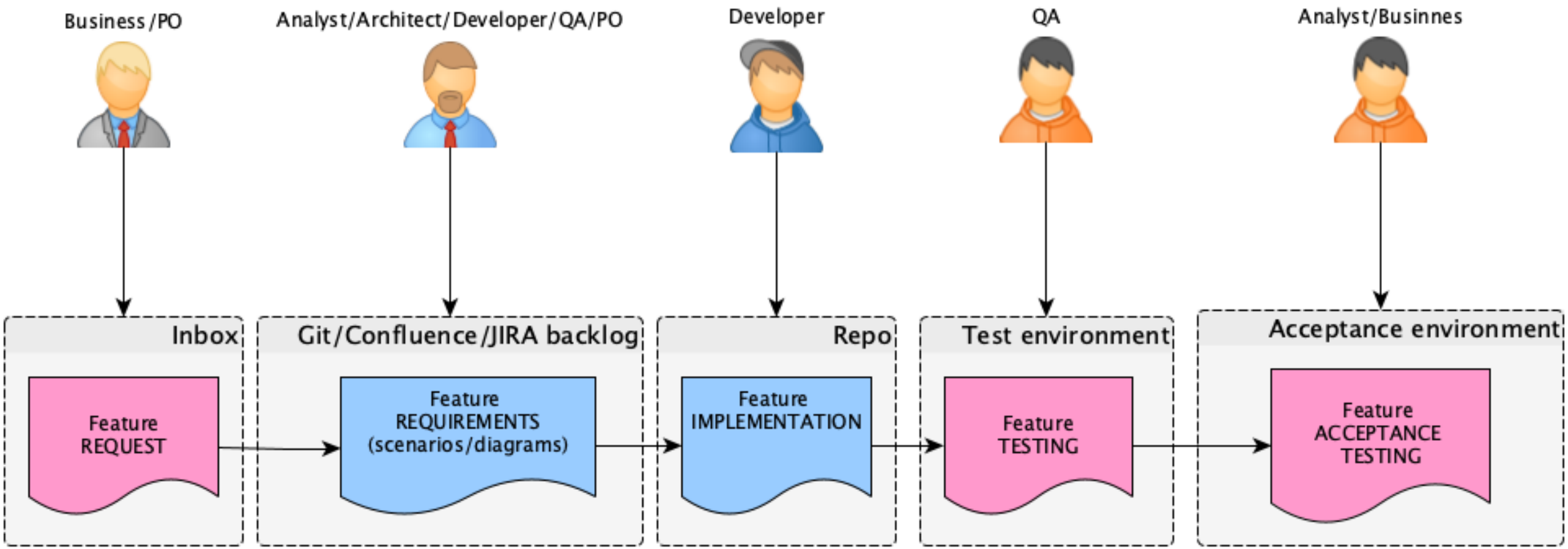
Solution?


Product owner
Stakeholder
Product manager
Analyst
Architect
Developer
QA
Same level
of understanding

BDD essentials

- BDD is all about working together when teams provide only what is valuable and really matters
- Agreement between all the related parties is very similar to a contract
- Allow to keep information constantly up-to-date as a living documentation

But still… how?


Business/PO
Analyst/Architect/Developer/QA/PO
Developer
QA
Analyst/Businnes
Inbox
Feature REQUEST
Git/Confluence/JIRA backlog
Feature REQUIREMENTS (scenarios/diagrams)
Repo
Feature IMPLEMENTATION
Test environment
Feature TESTING
Acceptance environment
Feature ACCEPTANCE TESTING

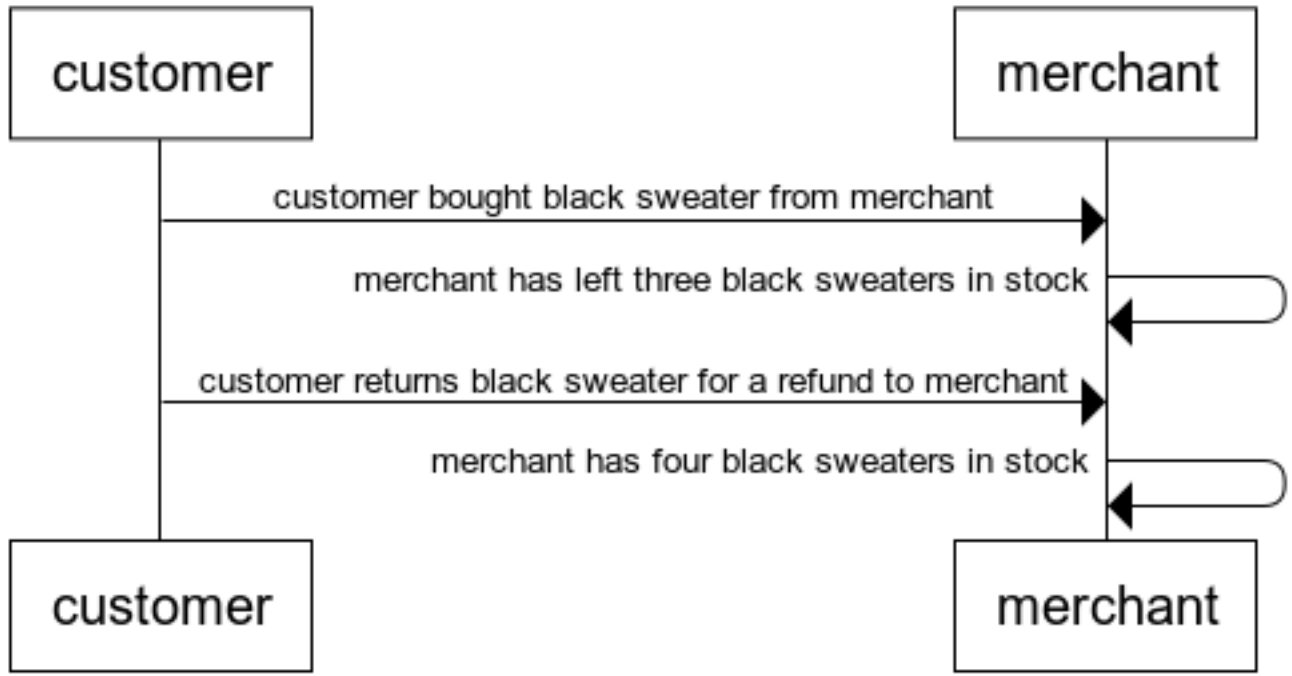
Feature breakdown

- Acceptance criteria based on business requirements
- Sequence diagrams as a basis for scenarios
- Scenarios
- Development
- Testing
- Acceptance testing

Feature essentials

```
Feature: Customer can return goods back to merchant's stock
As a customer
I want to be able to return goods
So that they will be returned back to merchant's stock
And I can get a refund

Acceptance criteria:
- Amount of items added to stock equals to the amount of items customer returns
- In case of return goods must be returned to merchant's stock
- Customer can get a refund
```


customer
merchant
customer bought black sweater from merchant
merchant has left three black sweaters in stock
customer returns black sweater for a refund to merchant
merchant has four black sweaters in stock
customer
merchant

Scenario essentials

```
@Returns
Scenario: Goods returned by customer should go back to merchant's stock
Given customer bought black sweater from merchant
And merchant has left three black sweaters in stock
When customer returns black sweater for a refund to merchant
Then merchant has four black sweaters in stock
```

Writing a good scenario

		AAA
AAA	Given	customer bought black sweater from merchant
AAA	And	merchant has left three black sweaters in stock
AAA	When	customer returns black sweater for a refund to merchant
AAA	Then	merchant has four black sweaters in stock

Checklist

- [ ] **Testability** - expected result and acceptance criteria must be clear on all level of details
- [ ] **Consistency** - style/naming convention should be the same for all the scenarios
- [ ] **Granularity** - level of details should be balanced (not too technical, not very high-level)
- [ ] **Self-sufficiency** - scenarios should be simple and clear outside of a context (AAA)
- [ ] **Doable** - taken TODO tickets should fit into the time frame of development iteration

Keynote

What is BDD?

- BDD is not just the given-when-then syntax, scenarios, user stories, reports or acceptance criteria
- BDD is about seamless collaboration

What is the goal of BDD?

- To teach people from different levels to speak on the same language, understand each other and to work in collaboration on living documentation

What are BDD essentials in a nutshell?

- Output - scenarios
- Outcome - same understanding

Why there is no chance to fail using BDD?

- Because everything is connected. If you want to change business requirement, first of all an item in product backlog has to be created and prioritized. Then it has to be analyzed and new acceptance criteria with sequence diagrams should be clarified. Then scenarios should be updated. And only after that it's possible to change the code and related tests. This is a power of living documentation that is always in up-to-date state