Stop Flipping a Coin before Release: Bring Your Requirements closer to Code

Kristina Egorova and Anna Dmitrieva (Grudina)

Deutsche Bank Technology Center

The BA Achievement Award is organised by

Achievement summary

What was delivered and what was the result?

Facing the complexity of the Business Analysts' team making concurrent changes to one system, we have come up with the solution to link functional requirements and their implementation in code. With this solution in hand, we significantly simplified the analysis and development processes and saved efforts and time. Specifically, we have delivered 6 complex projects with approximately 20 changes in them. From an analysis perspective, documentation and peer review of requirements took 1-2 hours instead of 3-4 due to standardized formatting of the requirements. And from a development perspective, requirements were implemented in 1-2 hours instead of the usual 3-4 hours spent on requirements interpretation and discussions, locating the right place in the code and aligning the incoming requests, such that there are no contradictions. In addition, we were able to detect approximately 5 defects caused by concurrent change requests – overall, we saved about 400 hours of analysts and development work.

Achievement details

Please explain the details of the initiative and the approach that was taken.

Problem Statement

Business Analysts are problem solvers who transform business needs into change requests and requirements specifications. At the same time, Business Analysts work with Development teams to explore the solution options, clarify the business scenarios or use cases, and communicate the solution to Business stakeholders.

Thus, a typical software development project can be seen as a coin with two sides to it. On the one side, we have business stakeholders who define the goals. On the other side, we have a development team, working on solutions tailored to support the business goal.

As Business Analysts usually communicate with both the Business and Development teams, it is implied that BAs can easily "flip a coin" in two directions: from Business to Development to translate the business need into a functional change, and from Development to Business to communicate the system limitations or clarify the conflicting requirements.

However, when it comes to financial technology in the banks, the projects are very complex in different dimensions. First, a single business need can be supported by

several integrated systems, which evolved over time and were gradually enhanced to meet market demands and regulations. Secondly, if the system is big, there can be a team of several analysts where each analyst is flipping his/her own coin – thus the analysts need to communicate among each other and be aware of the competing changes.

Lastly, mature systems are often comprised of several modules, having different programming languages or frameworks. Therefore, from the developers' point of view, it may be very challenging to match the change request to the actual lines of code and understand what exactly needs to be changed. Moreover, if there are several changes requested by different analysts and there is no strict convention for requirements specification, it is hardly possible to align all of the requests.

The matching of functional specifications and code process consumes efforts and time and can risk timely delivery. Specifically, until a developer has found the code he/she needs to change, it is unknown whether the change request is based on up-to-date functional specification, or there was a gap in functional specification and the entire solution requires revision and re-approval from Business stakeholders. If a revision is needed, the Analyst is flipping a coin back to the business side, and then back to the Development team. In addition to this, if a developer occasionally applies changes to a wrong place and it is discovered only at the testing stage, then such change is rolled back, and the matching process starts again.

Solution

As our BA team is co-located with the Development team, we came up with the idea to bring the requirements closer to code in order to automate the requirement-code matching and thus simplify the change delivery process and reduce the associated complexity risks.

We collaboratively designed a plugin for Intellij Idea, which automatically links functional specification items with the code. For example, the plugin anchors the mapping document to the corresponding mapper, parses the pre-formatted attributes and displays the requirement code as the comment in code. The plugin brings developers the comfort of a modern development studio: it highlights new data elements, clearly shows the differences between requirements' versions, and displays occasionally or intentionally removed items. Moreover, the plugin analyses and shows the dependencies between data elements, so if there are conflicting change requests, this can be immediately raised to the BA team.

Results

Once the Development team prepared the plugin, Business Analysts had to verify and clean up all the requirements, as the plugin demands pre-formatted documentation. Due to such demand, all automated requirements were standardized and BA team now follows the rules of preparing the requirements. With rules in place, all the requirements fully cover the possible cases, are easy to peer review and re-use, as there are no different requirements for the same functionality.

This data clean-up and automation exercise allowed the BA and the development team to assess the platform functionality, discover the similar parts and reduce the system complexity such that similar requirements and functionality are re-used for the various functional flows.

With the plugin enabled, we have successfully delivered several critical business requests without flipping a coin right before the release – thanks to the early mistake discovery and automatically highlighted changes.

Key achievement
Why do you think this initiative should receive the award?

We believe that our approach has several apparent benefits.

First, it enables early mistake discovery and prevention such that the BA team is equipped with an automated layer of requirements verification. For example, if more than one BA is making changes and the requirements are contradicting each other, the plugin shows it even before the requirements are fully implemented and can be tested.

Secondly, the plugin reduces the probability of a developer making a change in the wrong place and consequently reduces the chances to revert the wrong changes and develop them again. Thirdly, the plugin reduces the probability of misinterpretation of requirements, as the requirements are standardized and any difference between them is shown in a developer-friendly format.