Software Design: Project Phase II

Marko Schütz-Schmuck

March 25, 2019

Phase II should be the second (or higher) iteration of an iterative process involving domain description, requirements prescription, and, most importantly, software design.

Expectations

At the end of phase II the model should have become much deeper. Some initially hidden concepts should have appeared by now, user stories can be told in language that seems immediately plausible to all stakeholders (client, prospective users, developers, ...).

User stories now act as the starting points to justify high-level goals and objectives of the project. The high-level goals and objectives are further broken down, showing how components contribute or obstruct the goal or objective.

Phase II covers some of the requirement. Which ones? The ones that offer the team and the clients the greatest return on investment. This can be gauged by the relationship between the estimated effort for implementation and the estimated value for the client.

Your project now "hangs together" well: the source code reflects the model, the model explains and uses the concepts from the ubiquitous language, user stories can be illustrated by walking through the model, etc.

Of course, the project is supposed to use a repository for all its artifacts. It should also include documentation of exceptional insights (breakthroughs). Profound changes based on deep insight help explain how concepts are currently understood.

Always try complementing informal descriptions, prescriptions, etc. with formal descriptions, prescriptions, etc. Remember

"Natural language is imprecise; formal language is inaccurate." [1]

It's good to involve many different "senses" in building the mental model: diagrams appeal to the visual/spatial "sense". Use them to complement textual presentations (formal or informal).

There should now be a more complete understanding of the concerns that need to be addressed in the individual layers (in a layered architecture, or in other sub-structures if a different architecture is used).

Modules are now populated with key functionality that represents true value to the user and/or client.

Persistence should work for at least some of the implemented features.

There should be no source code in the implementation that is not accompanied by tests.

There needs to be some front-end to interact with the work-in-progress. This can be textual, graphical, browser-based, ... Whichever comes most easily.

Jointly reflect upon the roles of the team members: What are the strengths of each team member? In which way does the member contribute most of the time? For example, concerning modeling ideas, one member might be the one to frequently generate modeling ideas and the other might be very good at questioning, criticizing, and reviewing these ideas. Identifying (and hopefully agreeing on) the strengths of each team member can improve the collaboration within the team.

References

[1] Bruce Mills. *Practical Formal Software Engineering: Wanting the Software You Get*. Cambridge University Press, Leiden, 2009.