



info@netobjectives.com
www.netobjectives.com

275 118th Avenue SE
Suite 115
Bellevue, WA 98005
425-688-1011

Agile Design and Code Reviews

by Alan Shalloway

About This Chapter

I wrote the precursor for this chapter before I became adept at Agile methods. I was using agility at the time, but didn't know that's what it was called. No books on XP or Scrum existed at the time. However, it seemed that current design methods and design reviews were too cumbersome. Not only were they inefficient communication vehicles, they didn't help come up with designs very much. All too often they were simply vehicles for explaining to the group the poor design our architect came up with and why it was too late now to do anything about it.

Since becoming both an early adopter and promoter of agile practices I have realized that many of the poor results I originally wrote this article for will not occur in an agile project in the first place. The short iterations called for in agile methods preclude large up-front designs in the first place. However, since I recognize that many non-agile teams are trying to transition into agile, we have included this chapter in this book as an assist in this transition.

Overview

Design/Code reviews are recommended instruments advocated in nearly all software development methodologies to improve the quality of the software product. Net Objectives has worked with many different companies that have used design/code reviews in their development process. While everyone agrees these are necessary, how they are actually implemented varies greatly. They are often implemented in such a way as to be "heavy" in terms of resource allocation and often result in little, if any, improvement in the overall quality of the design or code. A more subtle, sometimes greater, danger is that when management sees reviews taking place, they may assume that quality designs are also taking place. Unfortunately, reviewing designs does not guarantee quality. Contrary to the assumptions of management, the typical design/code review process often contributes to poorer design and code. The reviews actually become counter-productive because they are not accomplishing their intents but mask the problem, and consume valuable resources as well.

This chapter describes the current typical review process – both what it is trying to accomplish and why it typically fails. It goes on to describe a simple way to perform design and code reviews that is productive, lowers risk and better utilizes a project's resources.

The typical review methodology

Current design reviews typically take place when a designer feels that their design is complete and that she is either ready to start coding or ready to hand it off to someone else

who will implement it. The purpose of the review is to ensure a proper design process has taken place and, more importantly, the design meets the minimal standards of the group and the project. Unfortunately, by the time the review takes place, it is often too late to change the design if an improvement is suggested. How many projects have you been on where the design is completed before it was scheduled to be completed? I would guess very few. This means, that, unless the design is effectively unworkable, when reviewed at this late stage, any better alternatives have little or no time to be explored. They are typically to be incorporated into the next release.

Another contributing factor to the review's failure is that management often sees design reviews as a way of communicating to the group at large. The review is often done to all potentially interested developers. This means many developers who only need a very cursory review at the design will now be exposed to greater detail than necessary because it has now become one of their responsibilities. Unfortunately, it is almost impossible to have a review that is open or interactive in this situation. Instead, the review becomes more of a lecture conveying the design that has been decided on by the designer to the participants.

Another problem stems from few developers having had training in public speaking. They often aren't aware of different learning styles inherent in an audience. Furthermore, unless the designer is using some sort of modeling tool, it will be very difficult for the designer to present a concise description of their design. This often results in the design presentation not conveying the issues and key points of the design effectively,. The result is a presentation that is difficult to understand and therefore an inefficient use of time. Unfortunately, this only increases the intimidation many of the reviewers already feel – further dissuading feedback.

People in general, and engineers in particular, are loathe to admit they don't understand something when it appears that everyone else understands well. It takes a tremendous amount of confidence and/or courage to question what a designer meant and potentially admit one's own shortcomings. If a question or concern occurs to a reviewer, she will often believe the problem is more likely due to her misunderstanding than it is to a poor design. It may be very intimidating to question the designer, especially since she has had a lot of time to think about things and the reviewer is just seeing it now. In this situation, what chance does a reviewer have to make a real contribution? Very little.

It is more likely the person asking the question will appear dumb, so typically the reviewers just nod¹. In any event, they know the more they challenge the design, the more likely their own designs are to be challenged. There is an unspoken agreement amongst developers; do not challenge each other and pretend everyone understands. In any peer reviewed encounter, the psychology of the situation is delicate and often swings into an area virtually devoid of constructive criticism with only negative outcomes engendered from the process.

¹ The author is a developer who has learned how to train. In my training roles, I have noticed how I have the ability to make anyone in our audience nod. After explaining a particularly difficult concept, I am sometimes anxious that it was understood. I have long ago learned that if I am desperate enough for assurance, all I have to do is to look at someone directly in the eye for a few seconds. Under that gaze, the student will nod their head, suggesting they, *of course*, understood what was said, while muttering to themselves – 'I have no idea what that meant'. Of course, I have learned that proceeding on the basis of nods alone is a sure way to have a poor training – but it illustrates the point that nods sometimes don't mean understanding.

The bottom line is design reviews that take place in front of the *entire* staff provide little tangible value to the team or the project. True understanding does not take place because the designer has probably not properly prepared for the review and just as importantly the audience is not prepared for the presentation. Given a lack of presentation training, she will have difficulty anticipating the questions she will face. People won't ask questions because of the intimidating factors present.

What are we really trying to accomplish with reviews?

Before continuing with my own proposals for reviews, let's look at what we are trying to accomplish in the first place. Basically, design reviews should:

- ensure a good design has taken place
- verify that the design meets the defined requirements and maintains the scope of the project.
- communicate to those who need to know the design so they can better understand how to use the architecture under review
- give feedback to the designer regarding how well the designed module is doing what it needs to do
- be a vehicle for knowledge transfer, that is, expose less experience designers to good design techniques
- clarify the documentation of the design (questions asked during the review should be documented for others to see)
- be as inexpensive as possible
- keep the team informed as to the status of a particular aspect of the project

This is quite a lot to ask of a single session. Many of these intentions are not necessarily aligned with each other. For example, maintaining the scope of a deliverable or sub-component of the system may not, and indeed, often is not, a mechanism for delivering the 'best' architecture. This balance needs to be understood by the key developers of the project in order to forge a pragmatic balance between design 'purity' and project objectives. It has been my experience that it is usually impossible to accomplish all of the aforementioned goals of a design review with a single review session. Several reviews are necessary. Of course, I am not suggesting several reviews be done in front of the entire design staff. Aside from being expensive, they would not address many of the current challenges. I *am* suggesting different kinds of reviews for different purposes. The total cost of these multiple reviews will typically be about the cost of one large review. However, significantly better results can be achieved.

The 'approach-review' strategy

The best way to ensure a good design will take place is to assist the designer early on in the process. A good way to do this is to have a small group (2-3) of senior developers and a representative of the business domain to assist the designer in her approach to solving the problem. One or two senior developers should be chosen by the designer, but an invitation to the group at large should also be made. One never knows when someone on your own staff has actually either has direct experience in the design or a keen interest in it. Even if such a person doesn't exist, or everyone knows everyone's experience level, the invitation gives an opportunity for someone who is very interested to participate.

The session itself should be a 1-2 hour meeting where different approaches are discussed. Its informality and brain-storming quality, which is accentuated in a small group, allow for an easy exchange of ideas. The varied background of the group will most likely allow for relevant experience to be used than if a sole designer goes off on her own.

By having a mini-review of the approach, the designer can design an initial API for their module sooner. This also accelerates getting feedback from those who will use it.

The ‘mini-design review’

After this initial design assistance, the designer works on her design in detail. She may want assistance from those in the mini-design group during this process. Since they have already established a core understanding of the problem, their assistance will likely be beneficial. In any event, after completing the design – or at least nearing it’s completion, a ‘mini-review’ should be scheduled. This review should include the same initial team as well as one or two additional members to ensure that the users of the designed module will also be represented. The primary purpose of this mini-review is to establish that a proper design was done. Ideas can be challenged more easily in this group because everyone should have a general level of understanding of the problem domain and the small group fosters a collaborative environment. If something must change, it can be accomplished prior to the presentation to the larger staff.

A secondary purpose of the review is to prepare the designer for the presentation to the group as a whole. The reviewers in this mini-session should make suggestions for describing the design to the larger group. Furthermore, some minimal documentation, e.g., modeling diagrams, should be constructed, if they haven’t been built already.² If the designer does not know how to generate these, one of the reviewers should lend a hand. Ideally, the group should have either standard presentation format or policy that can be used repeatable by the team members. This both facilitates knowledge transfer (the designer has learned a new skill) and improves the module’s documentation.⁴

The ‘design review’

Now, the group and the presenter are ready for the ‘normal’ design review. Here, as before, all affected staff should be present. The design will have already been checked for quality, the designer will have had experience explaining it to a small group and at least a minimal set of drawings (models) will have been constructed. This will allow for a concise, understandable presentation. Ironically (and fortunately) there is even more likelihood that, if someone has a concern about the design, she will ask about it than before. Given the higher

² If no one on your staff knows how to do these, give us a call!© Seriously, get a copy of Martin Fowler’s UML Distilled, 2nd Ed. Still the best UML book and not just because it’s also one of the thinnest.

⁴ This is one of the areas we disagree with the Extreme Programming camp. XPers would likely say no UML diagramming is needed. However, I feel that a UML diagram at a conceptual level (see the “Create A Conceptual Model Of Your Architecture” chapter) aids communication and is worth the cost of developing it. It is not hard to maintain because it doesn’t contain a lot of detail.

quality of the presentation, more people will understand more important details of the design. This will give the audience confidence to ask questions. This review will typically require one-third to one-half the time required if the mini-design and design-review preparation were not done. This is due to better preparation and because the intent of the review is now different (more of communication, less of questioning the design).

Summary of approach

I am proposing that instead of one design review to your staff as a whole, three different reviews are performed, each serving different purposes. These are:

- mini-review of the approach with 3-4 people besides the designer for 1-2 hours
- mini-design review with 4-5 people besides the designer for 1-2 hours
- full design review with your entire staff for ½ to 1 hour

Evaluating the Approach

Ensuring a good design

The smaller, up-front 'approach review' will steer the designer in the correct direction early in the design process. The team approach will lower the chances of the designer going in the wrong direction and wasting valuable and costly developer time.

Verify design's requirements and scope

The early designs allow for ensuring the designer has, in fact, included the necessary requirements, while also ensuring that she hasn't gone beyond the scope of the project.

Communication between members

The core group that establishes the approach will be communicating ideas early on. The short, concise full review at the end of the process will let the staff at large know what is happening and serve as a training and knowledge transfer vehicle for the group.

Getting Feedback

Both the mini-approach review and mini-design review will give the designer feedback much quicker than the standard approach of waiting until it is already constructed. Also, during the development of the design, the designer can ask this core team if she is providing the proper functionality required.

Knowledge transfer

Again, the up-front 'approach review' will allow people to work together early on. By having presented the review to the small group early, a better explanation of the design and approach will be made to the group at large. This will increase their understanding. Any new or elegant techniques will be clearer and more likely to be understood.

Improved documentation

By making the presentation of the design to a small group first, the designer will learn how to present to the larger group more effectively. The designer is more likely to create documentation towards this end now that she knows it is needed.

Inexpensive

Let's compare the two approaches in pure man-hour terms for a hypothetical situation (note, the numbers I use are fairly typical from my experience) Let's say we have a staff of 15 developers. The typical design review would last about 2 hours and involve everyone. Therefore, we would be spending 30 man-hours in it. My new approach would involve the following:

- mini-review of the approach with 4 people for 2 hours (8 man-hours)
- mini-design review with 5 people for 2 hours (10 man-hours)
- full design review with 15 people for an hour (15 man-hours)

Total time spent is 33 hours, about the same as before. Even if you spent considerably more time in this suggested process, however, it would be worth it because of the likelihood that you are saving significant hours by the designer as well as getting a better design done.

Summary for Design Reviews

Essentially, I am proposing a methodology centered on specialized, either smaller or shorter, reviews that focus on specific outcomes instead of trying to be everything to everybody. The efficiencies gained by this approach more than compensates for the extra sessions. Even if they do not, we are still getting more value for the time spent. Our focus is not on adding new things to do, but rather eliminating those things that aren't producing results.

Code Reviews

Dealing with code reviews is a little simpler. Assuming a design has been performed, the purpose of a code review is simplified. We really shouldn't have to see if someone has implemented the design. The implementer should be trusted to have done that. However, the implementer may not have done this in the most effective fashion. Essentially, in my opinion, the purpose of a code review is to ensure that the coder knows the best way to translate a design into code⁵.

A large forum is not a good way to accomplish this. Aside from being inefficient, it makes it very uncomfortable for the coder to have her work exposed in front of the entire team. This often results in coders delaying these reviews (and thereby, unintentionally, minimizing potential benefits).

Again, a small, preliminary review has obvious advantages. The reviewers shouldn't be so much like evaluators as much as they should be mentors or coaches. In fact, if you have a

⁵ There is another kind of code review which is to determine the quality of someone's coding style. However, this has a different purpose and should be done in a more private manner and by those in the group considered to be the coding experts.

mentoring team in place, your mentors should perform these reviews. This makes the code review process less intimidating as it can be accomplished under the guidelines of training.

Suggestions made by the reviewers (mentors/coaches) are therefore merely another form of training as contrasted with an evaluation of your work.