PDSA.com

Solutions for the Real World

PDSA Special Report

Software Process Done Right

The Process of Software Development: Software the Right Way

As developers we all love to code. However, we all have a tendency to want to jump into the coding as quickly as possible. In addition, we all may not be doing the things we know we should be doing. For example, not following industry-standard naming conventions for variables, not writing specification documents, not creating good, reusable components. All of these are things that should be done, but often are not. The reason for this that we typically hear in our consulting engagements is "We are just too busy..." However, when we check back with that same company a few months later, they are complaining that they can't read their source code and they are having problems with the software. User's also complain that the system should have done 'x', but it does 'y', and they now have three other IT projects that they have started. The programmers simply duplicated the code from one project to another instead of creating reusable components, and they now have a maintenance nightmare on their hands.

No matter how busy you are, you must not overlook the importance of the other disciplines of software development such as planning, design, prototyping, architecture, code reviews, and quality assurance. If you have a well thought out plan, you can ensure that the projects you create have a high chance of success, a consistent design pattern, they are maintainable, and have highly reusable components.

Let's take a look at some of the processes you should have in place in your organization.

Planning

Planning is a key element in any project (whether it is a software project or any other kind of project). Planning refers to making sure you have a roadmap for completing a project. This means you know the order in which you will create each piece of the project, as well as how each component will be built. An example of a plan would be the following:

Employee Module

- Meet with Becky in Personnel
- Create a Prototype
- Show Becky Prototype
- Refine Prototype

- Gather Business Rules from Becky
- Write Business Rules into Spec Document
- Meet with Becky to Discuss Spec Document
- Get Becky's Approval
- Design Interfaces for Business Processes
- Begin Coding

While the above is greatly simplified, at least you know the steps that you should be taking for developing the Employee Module for your application. However, it is not enough to plan on just the coding piece, you also need to take into account cost, time for planning, scheduling meetings, scheduling resources to work on the project, making sure they have the right hardware and network infrastructure in place, and a host of other items must be included in your plan.

As you can see, planning involves laying out each step on how you approach the project, how you create each piece of the application, who works on what piece, the time frame in which the components get built, how long it takes to build each component, what tools you need to purchase, how you get paid, and a host of other items that will vary from project to project. Using a good project planning tool, like Microsoft Project, can help you develop a good plan. Do not avoid this all-important first step. This can be the key to making or breaking a project!

Design

Before you start to code, you should design each module from a high-level perspective. This means making a feature list of each component that you will create for the project. You also want to keep in mind any future work you may encounter and ensure that your design has the flexibility to extend any components easily in the future. Create your interfaces (method calls) prior to creating your classes and map those interfaces to the business problem you are trying to solve. The interface names should be human readable with business oriented words that describe the process. Do not use cryptic, programmer oriented names.

Diagrams are a wonderful design tool for laying out business processes. Just a simple workflow of how data flows through an application can describe to a user (and a programmer) in one page what might take 3 pages to describe in writing. Use Microsoft Visio, or other diagramming tools to assist you in developing these diagrams. Keep the diagrams simple, with good labels for each step in the diagram. Other design documents might include a network architecture, server architecture, software architecture, and use-case scenarios.

Prototyping

Along with the drawings and the specification documents that you create, Create a prototype of the project. This prototype includes all screens that you intend to create for the application. The reason for creating a prototype up front is it lets you apply your design and run it by your users without creating the full application. By showing users real screens they can try out, you are able to get them to buy off on features much quicker. It will also prove to them that you really understand what they are looking for in the application. Another benefit of a prototype is users can play with the application and tell you what is missing much quicker than if you just show them mock ups done in PhotoShop or on paper.

Another facet of a prototype would be a proof-of-concept. If you are creating a component that you think can be used in multiple projects, or maybe a component that you don't know can connect to some external entity, then it helps to write some code to try out your theory. This saves a lot of time later because if you find out something cannot be done, then you can look for an alternative design.

Architecture

Architecture refers to your overall software strategy. This strategy includes many elements such as the various reusable components you design, how you put an application together, programmer documentation, how you begin an application, coding standards, and even deployment methodologies. Your architecture says a lot about the professionalism of your shop. One of the first things we ask to see when we go into a new company on a consulting engagement is to see their Architecture Model. We have been shown everything from a simple one page standards document, to nothing, to a full blown, well thought out, architecture. The folks that had a well thought out architecture were the ones that were typically most successful in implementing projects on-time and on-budget.

Code Review

Many programmers dislike code reviews. The reason is most likely due to their lack of planning and standards. However, you should learn to embrace the code review as a learning opportunity. Everyone you meet has something to offer, from the most junior to the most senior programmer. For example, one of the practices we use when consulting with companies, is to take the most junior programmer on the team and have them do a code review (under my supervision) of the most experienced programmers code. The reason for this is the junior programmer will ask questions of the experienced programmer on why they did something the way they did it. The experienced programmer now has the opportunity to explain their techniques to the junior person. By doing this, it forces the experienced programmer to really think about their process, helps him get used to mentoring others, and often reveals a better (simpler) way of doing something. The junior programmer gets the benefit of seeing how the experienced developer approaches a problem, and thus learns something as well.

If you are working by yourself and not in a corporate environment, grab a buddy, or your significant other and have them listen while you explain some code to them. This will get your thought processes going and might reveal a better way to do things in your code.

Quality Assurance

Quality Assurance (QA) refers to the process of checking your code for flaws in design, coding and functionality. It is very difficult for most programmers to QA their own code. This is because you are typically too close to the problem. You might not try out all combinations of keystrokes or data input necessary to check your code for bugs. However, another person who is not familiar with the project will try various things in an attempt to break the code. The rule here is to always have someone else QA your code. If this is not possible, then it is very important to create a set of test plans where you list various tests to try to break the code. By writing down test scenarios, it will force you to focus on what inputs might break your code. Test plans should be done regardless of whether you work alone, or in a group. Test plans should be added to by each person who QA's your application. This means that each person will probably think of new ways to test the application, and will write those methods down into your document.

Summary

Not all programmers have the discipline to follow a standard approach of developing software. We have just outlined many of the steps you should be using on each application you develop. While we have seen successful projects without following a process, more often than not, my team is called into fix those projects that were not successful due to project mismanagement. Let's face it, most applications do not fail because of technical reasons, they typically fail because of project management. When we approach a new project with our clients we outline our complete process for our clients and discuss with them the ramifications of not following each of the steps. Most clients understand and appreciate this process once it is explained to them and how it will save them money over the long run. Develop your set of methods and steps to follow for each application you develop, and you too can have great success at developing software applications that are on-time and on-budget.

If you do not have the time to put all of these processes together, PDSA, Inc. offers our Agile ALM product to provide you with templates of all these documents. You can purchase this Agile ALM complete with all of the resources mentioned in this article and much more. Visit our website at http://www.pdsaagilealm.com for more information.

Contact Information

If you would like to know more about the information in this special report, please contact either Paul D. Sheriff or Michael Krasowski at PDSA.

Paul Sheriff

(615) 675-4632

PSheriff@pdsa.com

Michael Krasowski

(714) 734-9792 x223

Michaelk@pdsa.com

Company Information

PDSA, Inc.
17852 17th Street
Suite 205
Tustin, CA 92780

Tel (714) 734-9792
Fax (714) 734-9793
www.pdsa.com