CLASSROOM TO CUBICLE: HITTING THE GROUND RUNNING IN THE REAL WORLD OF SOFTWARE DEVELOPMENT

Mary Slocum[*]
M.S. Program in Computer Science, Rivier College

Abstract

Computer science classes help students become good programmers. But, does it take more to be successful when starting your first job after graduating? There are other areas that computer science undergraduates need to be familiar with before entering the technical work force. This is intended to provide graduates with an edge: an inside look at the roles, processes, and project workflow to enable students to adjust to the workplace and become productive quickly.

1 Motivation

Students looking to go into a software development career need to be as best prepared as they can be for today's work environment. These days, it is not just about sitting in a cubicle and developing. There are interactions with other types of technical individuals and working with many different and new tools. In addition, developer must collaborate with their fellow developers as development is often done as a team and not independently as an individual. The intention is to cover many of the major tools and processes used by an engineering team, but there are still more. The purpose is to help the student become comfortable with the workplace model and expectations in order to be successful. However, it should be noted that if software engineering tools are not properly integrated, they can become a major blocker for team productivity [1]. So, special care must be taken to pick the set of tools to be used.

2 Roles

A software developer in today's workplace may fill one or several of many different technical roles depending on the work environment. In a large company, there can be testers (Quality Assurance (QA)), technical writers, release engineers, designers, product managers, and program managers to name some. Another technical role may be whether the product needs to localized and utilize translators.

In a smaller company, you may have to wear many hats and be designer, developer, and tester. In some companies, one might have no choice as to the tools used; while in other companies, a person may have a selection of options.

3 Development Tools

In some companies, one might have no choice as to the tools used; while in other companies, a person may have a selection of options.

3.1 Design

Often from Marketing, the developer would receive a Product Requirements Document (PRD) that outlines the high-level requirements for the product release. From this document, it is usually the responsibility of the development team to come up with a corresponding design and functional

ISSN 1559-9388 (online version), ISSN 1559-9396 (CD-ROM version).

specifications. For starters, the development document will contain a high-level approach (or answer) to the PRD requests. This may be an architectural document with flowcharts or Microsoft Visio charts initially. Or, it may contain the suggested Application Programming Interfaces (API) or functions to create.

In the web world, there are designers for usability, appearance/style, and content. The usability designer focuses on how the user will interactive with the website. This experience can vary depending on the expected audience for the website. For instance, the design would be different when designing a website for children instead of for adults. The website for children will probably have different colors and larger buttons, for example. So, rather than designing the menus, the usability designer constructs the navigation experience as to what people can be expected to do on the website. The content design determines the information that will be presented to the user. Whether to use text or images is one example of a content design decision. The design team will usually provide development with wireframes which are web drawings of what the site is intended to look like.

3.2 Integrated Development Environment (IDE)

An integrated development environment is a product that typically consists of an editor, compiler, link or interpreter, and debugger. There are many different development environments now to choose from. There are some available for purchase such as Microsoft's Visual Studio (for C, C++, C#, and Visual Basic) which also has a free version called Visual Studio Express [2]. In addition, there are free software development environments such as Eclipse that is popular and has plug-ins for Perl, Python, and COBOL though it is natively used for Java. Those that you pay for usually have support, automatic updates, documentation, and extra libraries to assist in quicker development time.

3.3 Version Control

After the IDE, the next important tool for the developer is the version control system. This software and backend database stores the source code and keeps track of the changes made by developers in a central location. It is important for the developers and the business to store code in a safe location along with monitoring the changes being made. This gives the developers the ability to track changes for each version or release of a product, revert (or back out of) changes if necessary, and see the particular developer who has made the modifications. In addition, much version control software has the ability to create branches, so some developers can start working on the next release while others work on finishing the release. There can be feature branches or fix branches which can later be merged into the main trunk of the tree of source.

Some popular version control packages are Subversion (SVN) [3], Perforce, Microsoft's Visual SourceSafe, Team Foundation Server to name a few. Some of these have integration for an IDE while others have their own user interface or commands that can be used.

3.4 Code Review

Peer code review is necessary to proactively find issues in the source code. This is also another means for the team to mentor each other on better ways to program. Exercising code reviews can be tricky since there may be cultural or psychological implications to consider. Who does the code review and when need to be determined. For example, a team may agree that code reviews are only necessary when within a certain timeframe of releasing a product such as after a code freeze date. Or, it can be decided to do code reviews every time anyone checks in code into version control no matter when in the release cycle.

Peer code review can be done manually by developers just looking at the code changes (using utility to present code difference using `svn diff`) or using web-based product such as Review Board

[4]. A web-based product is often used so that the code review information is kept in a central location. Such a product will email the specified individuals to notify them to review the code and then they will be required to `Approve` or `Reject` the changes with an explanation. Even though code reviews are time consuming, they are much more efficient than testing [5]. A typical engineer, for example, will find approximately 2 to 4 defects in an hour of unit testing but will find 6 to 10 defects in each hour of review code [4]. Also, Peer code review (PCR) maximizes the learning outcome of the student in writing quality code while removing the extra burden of checking programs written by students [6].

3.5 Unit testing

In many organizations, developers are now becoming responsible to unit test their own code before submitting it to QA. This unit testing is usually to at least exercise the "right" (or "positive") path for the code. QA would then be further responsible to test other routes such as a user putting in invalid data.

Some manual tests are simply just a list of steps in Microsoft Excel for a person to either indicate `Pass` or `Fail` for the test case. Developers can also write code or automated tests for certain situations in order for the manual tests to not have to be done numerous times. These automated tests can also present on the version control server in order for these tests to be regularly run without user intervention (such as running these tests every night on the server). In most organizations, new functionality means a new unit test.

Though the idea of Test-Driven Development (TDD) has been around since the 1960's, it has received its current popularity where the test is created, run, and before code is even written [7]. Then, the test is changed (or refactored) as development is done and the process is repeated.

4 General Engineering Tools

4.1 Project Planning

There are many different development methodologies. However, the two most common ones currently are Waterfall a linear process and Agile/SCRUM which is an iteration process. Waterfall is a sequential process where specific tasks are done in a certain order. So, implementation is not done until design is complete. On the other hand, Agile uses an iteration process using sprints of an agreed upon amount of time where design, implementation, and testing is done per iteration. For each iteration, there are "stories" and tasks per stories. The stories are rated on their relative amount of difficultly called story points.

No matter what the methodology, there are different tools used for tracking the progress of a project (or release), tasks, and resources (people). For Agile/SCRUM, there is a web-based (free) tool called SeeNowDo (at www.SeeNowDo.com) for iterations and tasks management called a storyboard [8]. There are also other Agile Management tools available on the market for consideration as this methodology is quickly catching on for web-based development. In addition, the tool could simply be a card, posted on a wallboard, prioritized by the project manager which is then delegated to a team member to build [9].

Of course, there are also the more traditional tools such as Microsoft Project. Using Microsoft Project, the projects and people can be tracked by dates. In addition, there are enterprise tools such as Rally Dev for Agile development management (ADM). Similar to SeeNowDo, Rally Dev will allow you to define stories and tasks along with tracking time estimates and resources.

4.2 Documentation

For software development, there are many different kinds of documentation that occur other than the design. There is documentation for the project (like the design and program process documentation), and there is documentation for the product, like user guides, and API and Software Development Kit (SDK) documentation.

Along with functional specifications, there may be dataflow or workflow diagrams. Once again, this can be created using Visio or shapes with Microsoft Word. Also, Unified Model Language (UML) can be used to help visualize the system being developed such as Visual Paradigm for UML Community Edition [10].

There is also the user guide that development must work closely with the documentation team. Sometime, development must create an informal document with screenshots in order to show the typical user flow. Also, development might have to review the documentation for accuracy. In addition, there is often online help or a tutorial that must be incorporated into the product that the documentation team has created.

Though project documentation can be created using Microsoft Office such as Word for developers to share, many are now considering doing internal web-based documentation using a Wiki such as Trac or MediaWiki [11]. Most open-source Wikis can be easily hosted on a server which allows all developers to contribute to its content. This Wiki can contain data such as Agile iterations, development processing information (for example, how to set up a development environment), or have file attachments of the documentation created. Lastly, Microsoft SharePoint has become popular in some Windows-based companies as a repository for internal documents for various teams to share.

Lastly, there is documentation of the source code. Often, a team will create development standards that will specify how the code is commented and naming conventions for functions and variables for documentation in the source code itself. In addition, there is a source code documentation generator tool called Doxygen which can be used for C++ [12].

4.3 Bug tracking and Testing

Along with development's unit testing, there must be formal testing done by QA. When QA finds a "bug", then the issue must be entered into a bug tracking system. Once again, there are many tools for this purpose. From this tool, there is usually the ability to create reports or metrics on the fixes in order to determine the rate of the issues coming in versus being fixed. Also, the team would want to track how many critical or blocking issues in order to determine if a release is ready. With each bug, there is also usually a rating for severity or priority along with steps. With the steps, it is important for the development and QA team member to communicate clearly as to the issues and the environment being used in order to be efficient.

QA also will be responsible for complete testing such as load testing (such as using LoadRunner), and performance testing. Load testing is to ensure software systems can scale up while maintaining functionality and performance [13]. For the web, load testing may focus on increasing the number of concurrent hits a website can handle without any speed deprecation. Of course, there is other monitoring that can be done including memory usage.

4.4 Licensing product

A consideration for a product before releasing is whether it requires licensing or not.

Licensing can be done on a per-user basis (either specifics users or any number of concurrent users) or per machine. Either way, the necessary information needed for licenses must be first determined. For example, if you want the license to be machine-based, then you need to determine and collect the necessary hardware information. And, you need to consider if the hardware no longer works, then should there be the ability to transfer the license to another system.

In addition, you might want the license to be subscription based that requires renewal after a certain amount of time. So, a date might be required for the license data. Also, a grace period (for a certain amount of days) may be needed along with invoking the necessary reminders. When a renewal or grace period is past, then the behavior of the product must be determine such as proceeding to reduced functionality mode.

Once all of the necessary data is settled on, then there are further details as to transporting this information for validation (such as encryption). However, there is licensing software available that could be used, or the licensing schema can be developed in-house as a project in itself.

4.5 Releasing product

Once the product is complete, there will be a need to come to a conclusion as to how to release it to the public.

For the web, this is the process of deploying it to a live production server. But, first, there are deployments to staging (test) servers that should replicate the production environment. In addition, there are servers for development and QA to work on when creating and testing the product respectively before release.

There are also products that are released on CD or DVD to be sold in stores, or a self-extracting executable to be downloaded from the web for installation. When working on a product to be released on a media, there is a time consideration as the product must be ready weeks before in order for a master CD/DVD to be created and approved for duplication.

A common product to create Windows-based build for installation is Install Shield. An installation build creation is also supplied by some of the IDEs available. The creation of the installation kit is usually done by the Release Engineering team. But, much like development, there are other considerations from just the build. There is the artwork on CD which usually matches the installation dialogs. The determination of one executable with all languages or an executable per language must be discussed. In addition, the question of forced or optional registration may need to be determined. Also, the question of providing an auto-update may be needed. So, there are many other technical elements that go into releasing product.

5 Putting the pieces together in the classroom

In the real world, a team of engineers will work together to create a product. So, the concept for a new class or workshop would be to put these pieces together when working on a project.

The class would consist of half lecture and half hands-on. First, teams would be selected. As in the real world you don’t get to pick who you work with, the teams should also be assigned for this exercise.

Each team would be given a project such as creating a program to play Battleship. The program can be written in whatever language is preferred by the teacher, but we will use C++ as our example. The team members must agree on the C++ development environment (IDE) they will be using (such as Visual Studio Express). From the assignment, they must write documentation as to their proposal to accomplish the assignment. This documentation must include data and work flows. Ideally, this would also include a list of tasks to divide up (and time estimates or milestones).

Next, the team would be asked to create a prototype of proof of concept. At this stage, it could be a basic Win32 console application. When sharing code, the team should be using version control. Of course, this would require that the IT department of the college set up a Subversion repository for the students to use.

Using a free Wiki (such as MediWiki or Trac) each team should be maintaining and modifying their documentation. In addition, the team should try experimenting with Doxygen.

After a successful approved prototype, the design of the final UI would need to be done and coded. At this point, unit testing must be planned and completed. Proper unit testing should ensure that the product is not dead on arrival.

When the project is finished, it will require a Release Engineer to create an installed package which could just be an executable in this case. Depending on the IDE selected, a developer may be able to create a simple executable deliverable. Then, it would be QA's responsibility to test (and development to fix any reported bugs). For this exercise, the bug tracking could be done on the created Wiki or simply using email to communicate the issues. Also, one development team can be the QA for another team for completeness.

References

[1] Maalej, W. Task-First or Context-First? Tool Integration Revisited. *IEEE Computer Society*, November 2009.
[2] Microsoft Visual Studio Express 2010. Express Downloads, 2011. Retrieved April 17, 2011, from http://www.microsoft.com/express/Downloads.
[3] Subversion, Collabnet Enterprise Edition, 2011. Retrieved April 17, 2011, from http://subversion.tigris.org/.
[4] Review Board, 2011. Retrieved April 17, 2011, from http://www.reviewboard.org/.
[5] Watts S. Humphrey. *Introduction to the Personal Software Process*. Addison-Wesley, Pearson Education, Inc. 2002:159-163.
[6] Wang, Y., Li, Y., Collins, M., Liu, P. Process Improvement of Peer Code Review and Analysis of Its Participants. *ACM*, February 2008.
[7] Desai,C.,Janzen,D.,Savage,K. A Survey of Evidence for Test-Driven Development in Academia. *ACM*, Volume 40 Issue 2, June 2008.
[8] SeeNewDo, Digital Taskboard for Distributed Agile Teams, 2011. Retrieved April 17, 2011, from https://seenowdo.com/index.xhtml.
[9] Lievesley, W., and Yee, J. The Role of the Interaction Designer in an Agile Software Development Process. *ACM*, April 2006.
[10] UML CASE tool for software development, 2011. Retrieved April 17, 2011, from http://www.visual-paradigm.com/product/vpuml/.
[11] Welcome to MediaWiki, 2011. Retrieved April 17, 2011, from http://www.mediawiki.org/wiki/MediaWiki.
[12] Doxygen, SourceForge.net, 2011. Retrieved April 17, 2011, from http://sourceforge.net/projects/doxygen/.
[13] Jiang, Z. Automated Analysis of Load Testing Results. *ACM*, July 2010.

[*] **MARY SLOCUM** is a Computer Scientist working on her Masters at Rivier College. She resides in Nashua, NH with her husband and two daughters. Mary is originally from New York where she worked in NYC before relocating to New England. She enjoys running half marathons and has a black belt in karate.