**Tooling and Documentation**

**Phillip Thames**

**ITIS 3310**

**6/4/2017**

# Introduction

Tooling and Documentation is a term used to describe a variety of applications used for the development and design of software. Software developers use a variety of tools to aid in the process of building software applications. Most of these tools aim to automate tasks which result in more cohesive and cost-effective software design. During the software design process, developers must document procedures and code to provide a clear understanding of the project’s inner-workings to team members, management and customers. In this paper, I will discuss the necessity of tooling and documentation, the types of each, and the choices developers are faced with when choosing these methods.

# Tooling

## The Need for Tools

Tools have been used for thousands of years to make tasks easier. Software development is no stranger to the use of tools. However, the tools used for software development and design are quite different than the wrenches in your dad’s toolbox. Software tooling includes applications used to make the development of software more streamline and easier. Developers use tools like Maven, Bash, Powershell, Visual Studio, and Eclipse to automate certain tasks such as setting up designs or making documentation from source code. Since software is a forever changing media, the need for automating tasks is a big one. “Engineering Software as a Service” by Armando Fox and David Patterson says that one of software’s biggest problems is cost estimation, so any tool that one can use to lessen their bottom line is usually worth it.

## Types of Tools

Not all tools used for software development are the same. According to Neal Ford in [Tooling and Documentation](https://www.safaribooksonline.com/learning-paths/learning-path-software/9781491987407/9781491901144-video158178), tools are broken into two groups, composable and contextual. Contextual tools are more of a plug and play design. These tools include things like Powershell, Ant, and Maven. These frameworks are great for the developer who is looking for an out of the box solution. They do what they are supposed to do, however, these frameworks are rigid and aren’t very flexible if a developer needs to make changes outside the realm of the framework’s design. On the other hand, there are composable tools to software development. These tools include Bash, Rake, Gant and Emacs. Composable tools are usually more powerful and off more flexibility to the user. Developers use these when it is necessary to make tweeks or extend the nature of the framework. You can think of these two types of tooling as the difference between color-by-numbers and a blank canvas. Color-by-numbers can produce artwork but it leaves no flexibility for the user and on the other hand, a blank canvas is what most artwork starts with but it takes experience and a vision to create the end product.

## Hard Choices

Choosing tools for software development can help or hurt your operation. Obviously the wrong tool for the job doesn’t get the job done and wastes time as well. Neil Ford explains that most developers have a hard time choosing tools because they think of the [cost](https://www.safaribooksonline.com/learning-paths/learning-path-software/9781491987407/9781491901144-video158178) of the tool as a sunk cost instead of maintenance. Developers must understand that not one tool can do everything they need for a software project. They might need to change tools half way through or even clean out their tool box completely. Developers are incouraged to use the easiest tool first and if the need arises for more flexibility, then another tool can be implemented. They must make these decisions based on the cost of the tool and how much time it will save.

# Documentation

## The Need for Documentation

For most anyone who has built a software applications or written code, documentation is the last thing that you want to do with your time. What’s the point? The code works, right? Documentation is necessary for managers, team members, and end users of software applications. Good documentation provides a clear description to others what processes are being taken and allows for good communication within a development team. However, nobody wants to do it, ever. So why not automate?

## Types of Documentation

Software architecture documentation started with types of diagrams to represent what was going on with the project. There were Booch diagrams, Rumbaugh diagrams, and Jacobson diagrams. These diagrams were named after the people who devised them. As [Neil](https://www.safaribooksonline.com/learning-paths/learning-path-software/9781491987407/9781491901144-video158178) explains in Tooling in Documentation, these three got together and created a new documenting solution, the UML diagram. Which stands for the Unified Modeling Language diagram. It is used to create simple diagrams that link to your software design and provide a clear description of the project. Diagrams make an excellent use to explain things to people who might not be so tech-savy and want a simple explaination or view of how the project is going.

## Documentation Choices

How does a developer choose documentation techniques? This is a daunting task. Developer must decide to what extent to document things. Neil suggests that one has to make the decision of when is documenting too much. Who are you documenting for? What is the information they need? Documentation tools exist for all languages so one must have a balance of clairvoyance and wasting time. It is also necessary to keep documentation up to date. If your project or code changes then the documentation will as well. So, file placement and organization is an important practice when keeping track of documentation.

# Summary

Tooling and Documentation can be a god-send for a developer looking to automate tasks and make their job easier. However, hard choices must be made. The correct tools must be chosen and one must not fear change. A developer should always be willing to use what can make their job easier given it is cost effective and does not limit their flexibility. Documentation should always provide clear descriptions to the individuals in need of the services without costing excessive time to the project.

# Q & A

What are the two types of tooling?

1. Powershell and bash
2. Contextual and conceptual
3. Feature and branching

Documentation is necessary for\_\_\_

1. Managers
2. End users
3. Team members
4. All of the above

Why is tooling a difficult discision?

1. Software is ever changing
2. Tooling is thought of as a sunk cost
3. Developers are picky

# References

“Engineering Software as a Service” Armando Fox, David Patterson

[Tooling and Documentation, Neil Ford and Mark Richards](https://www.safaribooksonline.com/learning-paths/learning-path-software/9781491987407/9781491901144-video158178)