Syntax and Semantic

Learning Tool

Thomas Hutchinson

May 2018

Computing Science

Supervisor: Dr Lindsay Marshall

Word Count: 999,999,999,999.21

Abstract

This paper wishes to explore and build upon already available Syntax and Semantic Learning tools currently available on the market (such as Stack Overflow) to create an application. This application will then provide a better learning experience to the end user allowing them to find and pick up required syntax and semantics for any required language.

The motivation behind this Dissertation and application was an attempt to apply some of the advances in the pedagogy of programming/scripting languages to a real learning tool. The new application was meant to be exploration of what can be possible when the pedagogical theory is applied

Declaration

*“I declare that this document consists wholly of my own work except where explicitly stated”*

*Thomas Hutchinson*

Acknowledgements

I would like to thank my own lazy arse for being bothered to write this stupid paper.

**Table of Contents**

[1. Introduction 6](#_Toc510016076)

[1.1. Aims and Objectives 6](#_Toc510016077)

[1.2. Table of Figures 7](#_Toc510016078)

[2. Background Research 8](#_Toc510016079)

[2.1. Market Research 8](#_Toc510016080)

[2.2. Audience Research 9](#_Toc510016081)

[Survey Rationale 9](#_Toc510016082)

[Survey Results 9](#_Toc510016083)

[2.3. Language Research 13](#_Toc510016084)

[2.4. Academic Research 14](#_Toc510016085)

[Language Pedagogy 14](#_Toc510016086)

[Crowdsourcing and Gamification. 15](#_Toc510016087)

[2.5. Implementation Technology 17](#_Toc510016088)

[2.6. Evaluation Techniques 18](#_Toc510016089)

[3. System Design 19](#_Toc510016090)

[3.1. Requirements 19](#_Toc510016091)

[3.2. High Level Design 20](#_Toc510016092)

[3.3. Fun Design Things 21](#_Toc510016093)

[4. Implementation 22](#_Toc510016094)

[4.1. Implementing the Upvoting System 22](#_Toc510016095)

[4.2. Implementing the Post Tagging System 23](#_Toc510016096)

[4.3. Implementing the Real Time Search and Sort System 24](#_Toc510016097)

[5. Testing 25](#_Toc510016098)

[5.1. Unit Testing 25](#_Toc510016099)

[5.2. System Testing 26](#_Toc510016100)

[6. Evaluation 27](#_Toc510016101)

[6.1. Audience Survey Results 27](#_Toc510016102)

[6.2. Focus Group Testing 28](#_Toc510016103)

[6.3. Project Requirements 29](#_Toc510016104)

[6.4. App Reviews 30](#_Toc510016105)

[7. Conclusion 31](#_Toc510016106)

[7.1. Overview 31](#_Toc510016107)

[7.2. Aims and Objectives 32](#_Toc510016108)

[7.3. Summary 33](#_Toc510016109)

[7.4. Future Improvements 34](#_Toc510016110)

[8. References 35](#_Toc510016111)

[9. Appendices 36](#_Toc510016112)

[9.1. Appendix A: 1st User Survey Results 36](#_Toc510016113)

[9.2. Appendix B: A Load of Crap 37](#_Toc510016114)

# Introduction

This section provides a general overview of the Dissertation covering the Dissertation structure itself as well as the general aims and objectives.

## Aims and Objectives

**Aim:** To create a tool that can be used by both intermediate and expert level programmers to aid in their acquisition and learning of new [programming/scripting] languages syntax and semantics.

1. Research existing products currently available on the market and how my tool can offer a service that is different from competitors while still allowing its users to learn new syntax effectively.
   1. Efficiency will be marked by the user being able to find their required information with only a single search and to have found a suitable post within 5 minutes*If a suitable post exists*
2. Research the Pedagogy of [programming/scripting] languages to ensure the tool teaches its information in the most effective way possible.
   1. Effective learning will be defined as a user being able to acquire that knowledge then apply it in a project/ work environment.
3. Conduct a review into the user perception/usage of the currently available tools as well as my hypothetical tool.
4. Develop said tool using the collected information and research to influence design decisions and the overall user experience.
5. Evaluate the performance of the tool in relation to the research from the previous objectives once completed by measuring various statistics such as user activity and user retention.

## Table of Figures

|  |  |  |
| --- | --- | --- |
| Figure Number | Description | Page |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Background Research

## Market Research

## Audience Research

### Survey Rationale

Another important part of determining which direction my system should go in is by asking my target demographic a few questions relating to their current tool usage and learning habits. I can achieve this through focus group testing but also an audience survey. This survey would be targeted at both student/” junior” developers and senior developers alike to gauge opinion about my possible tool. I primarily want to find out how developers learn new syntax and how/what tools they currently use to solve problems. Below is a list of possible questions for the survey alongside a short purpose of asking that question.

### Survey Results

After hosting the survey online for one month a total of 16 answered it. While this response number is not brilliant for forming conclusions about my hypothesis. Using it in conjunction with my academic research as well as the follow up survey after the completion of the app will help to give more reliable data about the validity and correctness of my hypothesis.



Another point of note is the distribution of occupations. Due to the limited reach of the survey it was primarily completely by Full-Time Students. While this isn’t necessarily a negative point it does mean that my conclusions must be adjusted to account for the primary demographic of my application being students.



The survey also revealed that Stack Overflow is by far the most popular language learning tool amongst the student demographic. This means that the market research should primarily focus on what makes Stack Overflow so popular with this demographic.

On top of this it will also be worthwhile to explore why the other tools do not hold as much favour with this selection of students. As you can see from the above graph Stack Overflow was marked as a 7 in usefulness by most students who answer this section. That is directly compared to an Article based site like Dev.to which received a high selection of 1 responses. This data will be useful in future conclusions regarding the effectiveness and popularity of certain applications over others.



Another interesting response is trend towards tool dependency. Out of the 16 people who answered the survey 62.5% of the respondents voted 4 or above claiming that losing access to their preferred tool would hold a detrimental effect on their normal workflow. Once again this is useful information for later as it shows that users may not be learning or retaining information which requires them to make multiple visits to their tools. On the other hand it could just show that students are required to develop in wide range of languages which means they are in constant need of additional learning materials.



Regarding the layout and development of the app one surprise was the number of respondents who prefer a solution-based search system. The respondents answer clearly show that it is greatly preferred when a tutorial is given with a specific real-world example. When it comes to developing the application, it will be important to attempt to implement this idea and to allow the linking of tutorials and specific examples. This could be a link to the idea that high level theory is difficult to learn by itself so showing the application of that theory makes it easier for a wider range of programmers to learn and apply that concept. Therefore, for the tool to be successful with this user base it needs to rely on linking theory to concreate examples.



Yet another interesting point of note is the number of respondents who rely on Stack Overflow posts to acquire and learn new languages. Once again this speaks volumes about Stack Overflow’s ability to provide tutorial material to its users who on average return whenever they need to learn a new language. The second highest response for this question with 13 answers is “taught by a teacher/expert”. So, the best way or at least the most favoured way to learn is by example or by expertise. This will be an important element when it comes to my application as It needs to allow the experts of a language to fully demonstrate and create tutorials for specific topics. By looking at these survey results it can be concluded that the respondents prefer learning from experts and being given examples of the theory in motion.

## Language Research

## Academic Research

### Language Pedagogy

Teaching languages can be a difficult and complex topic to discuss because every single learner will be subject to different parameters and situations. A learner’s previous knowledge and understanding of programming paradigms and concepts has been found to play a big role in their aptitude for understanding new language. Leon Winslow concluded that an expert can create various mental models and implement generic programming concepts easier than novices. He also states that a novice may only become an expert through years and year of practice and refinement *[Winslow 1996]*. It then stands to reason after these years of training and practice they have an understanding and skill that allows them to acquire these languages faster.

There is another aspect to this practice though. The simple act of constant practice isn’t the only element that increases one’s aptitude for language acquisition. Semantics also plays a huge role in determine a “programmer’s general skill/ability” found Allan G. Bateson *(Allan G. Bateson, Ralph A. Alexander, Martin D. Murphy 1987)*. A programmer’s understanding of core fundamentals and topics will aid them far more than specific syntax knowledge when understanding new languages. Bateson also concluded that the greatest gauge of a programmer’s general aptitude is their semantical knowledge over their syntactical knowledge.

Both Bateson’s and Winslow’s work combined help paint a more detailed picture of how one learns a programming language and learns it effectively. A novice will simply focus on surface features such as control structures and simple iteration while an expert will dive deeper and focus on more complex aspects of a language much faster *(Winslow 199)]*. This is because of their ability to understand complex features such as object orientation and they can manipulate that to solve problems much faster and to a much higher standard. Their code will consist of less errors, be more efficient and reach specifications much easier. A new language shouldn’t affect an expect as much as a novice because they understand and can apply their knowledge regardless of the environment.

Regarding my own application these papers have helped to direct my own tool in the direction of providing syntactical assistance while rooting that assistance in semantic information. By this I mean, users will be rewarded for thoroughly explaining code snippets and referring them to generic programming structures and concepts rather than just writing the syntax with no explanation. This benefit both experts and novices as the experts get the code syntax they require, and novices can expand their knowledge of programming semantics. This over time will help develop them into better programmers by improving their semantical knowledge rather than just their syntactical knowledge. Which in turn improves their overall programming aptitude.

### Crowdsourcing and Gamification.

For the syntax/semantic tool to be of use to anyone it is going to require a large amount of data. This data will have to be created covering a different topic and be sourced in a short time-frame. That is why crowdsourcing will be an indispensable technique in helping to build up the resource base and to sustain its growth. By gathering a community based around the application and encouraging accurate content creation the potential of the application can be realised.

What is crowdsourcing though and how can it be used effectively to gather a large quantity of data in a short time span (1 to 2 months)? Well David Geiger summarised crowdsourcing into 4 major categories: Crowd Rating, Crowd Solving, Crowd Processing and Crowd Creation. These categories where devised on the type and complexity of contributions required from the user as well as how those contributions were used and evaluated *(Geiger 2011)*. After analysing the requirements of my system, I devised that the best styles of crowdsourcing for me would be the Crowd Creation and Crowd Rating.

These two styles would be separated across the two primary user groups. The posters and the searchers. Users may travel between these groups freely but when completing a task, they will be a part of either one group or the others. The posters will follow a Crowd Creation crowdsourcing. This involves the users being given the freedom to create certain content within the specification of the site *(Geiger 2011)*. Using a form that requires them to enter certain information but then allows them to create whatever relevant text-based content they want they will have the freedom to provide high quality content. Searchers can then make use of a Crowd Rating system to assign value to these posts and ensure high quality content is rewarded. Geiger himself commented on this relationship between creation and rating saying that a larger audience of creators requires a larger audience of people rating *(Geiger 2011)*. This is because after a certain point and a large enough community managing in-coming content becomes impossible and a larger crowd is required to keep on top of it.

Unfortunately, crowdsourcing is useless without a crowd and for my system I would require a constant healthy stream of new content meaning the tool not only needs a crowd but a consistent one at that. Rather than amassing a crowd it would instead be better for the system if a community was constructed. Communities often develop around a mutual point of discussion or common goal and a healthy community remains as bonds and trust are developed *(Zwass 2010)*. By encouraging user interaction through the voting system and comments sections a community will develop around the application. This in turn encourages further user involvement and increases the chances of the user’s continued use of the application. Which is what the application requires to provide up to date syntactical resources to the searching users.

On the topic of community retention. Keeping a community focused and attached to a specific application or concept can be the most difficult aspect of any crowdsourced project. It is the natural way of humans to start of enthusiastic about topics and activities and then gradually over time become board or disinterested. This then leads to them leaving the platform and migrating somewhere else. This would be categorised as a steady increase in users then over the span of several weeks and months the user count slowly stops increasing and then starts decreasing. This then induces a death spiral in users as the content begins to stagnate and more users leave because of the other users leaving. This then concludes in the platform only being left with a skeleton of its former self and it can be nearly impossible to revive a platform after that. See the History of Myspace as a perfect example of this. A more interesting platform appeared, and a clear majority of the users quickly migrated. *(Wikipedia-Myspace 2017)*

So, to improve user retention the application requires something else. An extra system needs to be developed and deployed to attract new users and keep the old, invested, users around. This system could be based around the theory of Gamification. Gamification is the process of adding features and elements of games to non-game tasks. This has been shown to increase user retention as well as increasing the chance of attracting users in the first place *(Luis von Ahn 2008)*. There are several ways of implementing game features and these ways can have very levels of effectiveness based on how they are implemented and for what purpose. Luis spoke about user ranking in his 2008 magazine article. This is the process of taking a user’s input then quantifying that contribution in terms of effort and quality and giving them a score. That score can then be applied to ranks and leader boards *(Luis von Ahn 2008)*. This could be a brilliant feature for my site as it can be applied to both the searchers and the posters.

Posters can be awarded based on the communities’ evaluation of their post. Searchers can then be awarded for providing frequent comments and rating on these posts. The level of reward would have to be based on level of contribution as if users are rewarded for simple tasks such as giving rating they have been shown to give more spurious ratings *(Zwass 2010)*. So, a smaller score could be given when a user rates and a larger score given to high quality posts. This score can then be acquired over the user’s lifespan and as they reach certain milestones they can “rank up”. This is a badge they can display in comment boards as well as their profile page. It can be deduced that a user who has spent a long time acquiring a high rank will be less likely to leave the site.

With the techniques of Crowdsourcing and Gamification combined the syntax tool should be able to gather a large data set of learning resources and then maintain a steady stream of new data as required. Users will be encouraged to join in and stay around because of the community of fellow posters and researchers who can communicate in comment sections. They can then strive together to gain more points by posting and completing tasks that benefit the site such as creating posts and rating effective content. This in turn will then attract new users to the site due to the range of high quality content. If all goes to plan it should be steady upwards spiral of user numbers as the invested community rises thanks to word of mouth and recommendations.

## Implementation Technology

## Evaluation Techniques

# System Design

## Requirements

## High Level Design

## Fun Design Things

# Implementation

## Implementing the Upvoting System

## Implementing the Post Tagging System

## Implementing the Real Time Search and Sort System

# Testing

## Unit Testing

## System Testing

# Evaluation

## Audience Survey Results

## Focus Group Testing

## Project Requirements

## App Reviews

# Conclusion

## Overview

## Aims and Objectives

## Summary

## Future Improvements

# References

* *(Winslow 1996) –* “Programming Pedagogy – A Psychological Overview”, Leon E. Winslow, ACM SIGCSE Bulletin – Volume 28 Issue 3, Published: September 1996, Pages 17 to 22
* *(Allan G. Bateson, Ralph A. Alexander, Martin D. Murphy 1987)* – “Cognitive Processing Differences Between Novice and Expert Computer Programmers”, Allan G, Bateson, Ralph A. Alexander, Martin D. Murphy, International Journal of Man-Machine Studies – Volume 26 Issue 6, Published: June 1987, Pages: 649-660
* *(Wikipedia*-Myspace 2010) – Wikipedia Myspace (2017) Available at: <https://en.wikipedia.org/wiki/Myspace> (Accessed: 12th October 2017)
* *(Luis von Ahn 2008) – Luis von Ahn, Laura Dabbish ‘Communications of the ACM Volume 51 Issue 8’ (August 2008), Pages 58-67*
* *(Zwass 2010) –* “Co-Creation: Toward a Taxonomy and an Integrated Research Perspective”, Vladimir Zwass, International Journal of Electronic Commerce – Volume 15 Issue 1 Published 2010, Pages 11-48
* *(Geiger 2011) –* ‘Crowdsourcing information systems: a systems theory perspective’ *In proceedings of the 22nd Australasian Conference on Information Systems (ACIS 2011)* Sydney, Australia, Sydney: ACIS Pages 0 – 12
* [rigaux.org](http://rigaux.org/language-study/syntax-across-languages.html#VrsMnlMmrllc) (2017) Available at: <http://rigaux.org/language-study/syntax-across-languages.html#VrsMnlMmrllc> (Accessed: 12th October 2017)

# Appendices

## Appendix A: 1st User Survey Results

## Appendix B: A Load of Crap