UpStage Team

Investigation of Programming Languages

UpStage 2015 S2

Table of Contents

[Researching Criteria 2](#_Toc430605684)

[C# 3](#_Toc430605685)

[Java 5](#_Toc430605686)

[Node.js 8](#_Toc430605687)

[Python 9](#_Toc430605688)

[Proof of Concept 12](#_Toc430605689)

[Rationale for choosing C# 14](#_Toc430605693)

# Researching Criteria

For the programming languages:

1. Readability  
   - Is it easy to understand to code?
2. Testability  
   - Are there any useful testing and debugging tools for this language?
3. Learnability  
   - How much knowledge does the team have of this language?  
   - Is it easy to learn?  
   - How much prior knowledge does it require to use this language?
4. Performance  
   - Can it run in real time?  
   - Is it fast enough?

# C#

C# is defined as a modern, type-safe, object-oriented and high-level programming language developed by Microsoft running on top of the .Net Framework (Benton, Cardelli, & Fournet, 2004). It picked the best of other languages like Java, C++ or C.

**Learnability**

Pros

* Provides a number of features to make it easier for a developer to code in this language.
* There is an enterprise (Microsoft) that users can inquire to.
* Language is simplified for some functions such as multithreading.
* Easy to learn if you are familiar with other programming language like Java or C++.
* As UpStage requires real time web functionality, SignalR can be used. SignalR is a library for ASP.NET developers that makes it incredibly simple to add real-time functionality to the application (SignalR, n.d.).

Cons

* There was another disadvantage of C#, which it was not supportable for the platforms other than Windows, but .NET Core became open source in 2014. Roslyn, ASP.NET 5, Entity Framework, etc. these are now all free to use (.NET Blog, 2014). However, if the platform is not Windows, the knowledge of using .NET framework on other platform is required as it is developed for Windows.

**Readability**

Cons

* C# uses the curly brackets like Java does. It is more readable than other languages such as Python that do not use the curly brackets.

Pros

* LINQ in C# makes easier to code but harder to read.

**Testability**

Pros

* .NET provides many testing tools for many purpose. i.e. Telerik Test Studio for a UI testing, TypeMock for unit testing, etc.
* ASP.NET has a useful tools like Elmah (Error Logging Modules and Handlers) or Glimpse (the open source diagnostics platform of the web to inspects web requests to reduce debugging time).

**Performance**

Pros

* Have all the benefits of .NET framework as C# is tied to .NET framework.

Cons

* The size of application can be huge as it contains a lot of things in it. It might affect the speed when the application is running.

**References**

Benton, N., Cardelli, L., & Fournet, C. (2004). Modern concurrency abstractions for C#. *ACM Transactions on Programming Lanugages and Systems, 26*, 769-804.

SignalR. (n.d.). *What is ASP.NET SignalR*. Retrieved August 19, 2015, from <http://signalr.net/>

.NET Blog. (2014). *.NET Core is open source.* Retrieved August 19, 2015, from <http://blogs.msdn.com/b/dotnet/archive/2014/11/12/net-core-is-open-source.aspx>

# Java

Java is a popular and widely supported programming language. It follows the WORA (Write Once, Run Anywhere) philosophy, which means that once compiled, java code will be able to run on any supported platforms without recompiling. For this project, in terms of options for programming the Back-End of UpStage, Java has the following pros and cons.

**Readability:**

Pros:

* As Java, like C#, is a structured language with things such as the {} syntax, it is easier to read compared to scripting languages such as Python or Node.js as the code follows a more defined structure.

Cons:

* Forced/checked exceptions will bloat the code unnecessarily and might not be the best solution.
* Java is a relatively verbose language compared to some scripting languages, such as Python or Node.js, or languages such as C#, which means code that does the same thing is often longer in length in Java than it is in the other languages. This can reduce the readability of the code.

**Testability:**

Pros:

* Forced/checked exceptions means that they will not be left as an afterthought, reducing maintenance costs in the future and making it easier to debug the program. In C#, Python or Node.js, exception handling is not checked and thus programmers may sometimes skip or forget about these.
* Backwards compatibility is an integral part of Java, which will reduce maintenance costs in the future as functions will be less likely to break or stop working between versions or updates. The JVM, or Java Virtual Machine, is fully backwards compatible and can run any code compiled with previous versions of the JDK, in contrast to Python, Node.js and C# which is not backwards compatible past a certain point.
* Java has access to excellent and powerful tools such as Eclipse and Junit, which allows debugging in steps or placing breakpoints when debugging. This will aid programmers in debugging, or otherwise testing, the code. Automated testing tools such as Selenium is also compatible with Java.

Cons:

* Although programmers are forced to catch exceptions, they are not forced to handle the exceptions, and may instead mask them by an empty exception handling, which will make the code harder to debug or maintain.

**Learnability:**

Pros:

* An extensive library of modules and APIs lessens the chance of having to “Reinvent the Wheel”.
* Familiarity, as Java is the main language taught to AuT students, and as such upskilling time for new members can be reduced.

Cons:

* A vast library means it can be difficult to find the proper libraries and/or tools, which costs time. An inferior option may be used in place of a better option if not enough research is done in choosing which libraries to use, as some libraries are known to be of worse quality than their equivalents.

**Performance:**

Pros:

* There are several libraries, such as Atmosphere and Spring Framework, which aids java in achieving asynchronous communication, which is a requirement for the project.
* As Java depends on its JVM to run the code, its performance is relatively stable on all platforms, whereas C# might run slower on non-windows platforms, and Python and Node.js are dynamically typed, which will slow them down on run-time.

Cons:

* Java applications may have some performance spikes, such as when a class is loaded when it is first referenced, or when the garbage collection happens, which might result in non-asynchronous communication for a short period of time. As such, java’s real-time performance is not consistent.

**Extra:**

Pros:

* Widely supported on most platforms, including computers and mobile devices, like Python and Node.js, but unlike C# which is primarily made to work on windows platforms.

Cons:

* Java is relatively heavy-weight compared to scripting languages like Python and Node.js.

# Node.js

Server side technology - (Replacing Twisted)

**Readability:**

+ should be easy for anyone with JavaScript knowledge

**Testability:**

+ It is testable with the use of assertion testing ([https://nodejs.org/api/assert.html](https://www.facebook.com/l.php?u=https%3A%2F%2Fnodejs.org%2Fapi%2Fassert.html&h=gAQHEi3Vi&s=1))

eg.

* assert.fail(actual, expected, message, operator)
* assert(value[, message]), assert.ok(value[, message])
* assert.equal(actual, expected[, message])

- JavaScript has a lack of features to assess modules or the code quality

 + Four types of tools needed for successful Node.js testing:

* A Testing Framework (Mocha, Vows, Intern)
* An Assertion Library (Chai, Assert)
* Stubs (Sinon)
* Module Control (Mockery, Rewire)

(<http://fredkschott.com/post/2014/05/nodejs-testing-essentials/>)

(<https://github.com/mochajs/mocha>)

**Learnability:**

+ NodeJS apps are written in JavaScript so it’s easy to learn if you are familiar with JavaScript

-+ It still has a long way to mature as it has a little online support in developing. There are social networks for Q and A also video sites for tutorials.

(<http://nodetuts.com/>    - for nodeJS video tutorial)

**Extendability:**

+ For server development and web development

- NodeJS structure is not mature enough to identify any packages that can be trusted

(<https://www.quora.com/What-are-the-disadvantages-of-using-Node-js>)

**Performance:**

+ Uses V8 developed by Google for use in Chrome and it compiles and executes at lighting speeds due to V8 compiles JavaScript into native machine code (Sychay, 2014)

+ Non-blocking I/O

+ Asynchronous programming: Ability to handle thousands of concurrent connections with minimal overhead on a single process.

- doesn't support multi thread

(<http://www.bayt.com/en/specialties/q/53925/what-is-node-js-what-are-its-advantages-and-disadvantages-how-it-can-be-leant/>)

# Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

**Pros**

* Python is a high level dynamically typed language. It is easy to pick up for new programmer and easy to use. Because it is a high level language, Python is typically shorter than other programming language like Java.
* If people are used to Python, it actually more efficient in developing stage.
* Python is easy integration with and extensibility using C and Java.
* For IDE, NetBeans 8.0.2 started supporting Python which is a bit easier for AUT students (We all familiar with NetBeans)
* For current UpStage, the code is written in Python and we need to maintain the current UpStage. We must learn Python. If we use Python for our new UpStage, we don't need waste extra time for learn other new programming languages.
* Debugging Python programs with PyDev is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on.

**Cons**

* Python do not has brackets for functions, if the code is hundreds or thousands lines, python will a bit a hard to read without proper comments.
* Code is hard to read, easy to lose.
* Python is a high level language, it runs slower.
* Python 3 is not backwards compatible. (The current UpStage is using Python 2.7.3)

**Readability**

Pros:

* Python is using Tab to separate the functions. The code in Python is clean and short.

Cons:

* Python is a dynamically typed language. We do not need to declare the type of variables that make the code become hard to read without proper comments.

**Testability**

Pros:

* By using IDE debugging tools like PyDev, Python is easy to test.

**Learnability**

Pros:

* Python is a high level language and dynamically typed language, it is easy to learn and easy to use.

**Performance**

Pros:

* Python can used in Linux/Windows/IOS Mac X. Python can be used in most popular platforms.

Cons:

* Python is a scripting language, it runs slower than other structure languages. (e.g. Java, C#)

# Proof of Concept

## Description

* Part 1: two pair programmers with the task to create a simple chat app in Java or C#.
* Part 2: pairs exchange their prototypes to extend by adding an avatar button that adds or removes an image in real-time

## Criteria

1. Ease of Use  
   - Is it easy to code?
2. Readability  
   - Is it easy to understand the code written by other members?
3. Extendibility  
   - Is it easy to extend the code?
4. Consistency  
   - Always fast?
5. Real-time  
   - Dies it work well when there are more than two clients at the same time?

## The simple application must

* Have a chat function.

## The extended application must

* Have a button that shows an image.
* Have a button that removes the image.

## Findings

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Prototype Criteria: | | | | |
| Language | **Ease of Use** | **Readability** | **Extendibility** | **Consistency** | **Real-time** |
| C# | There wasn't much use of C# in the prototype but SignalR (the API for chat) was fairly easy to use. The pair programmers involved with chat were familiar with SignalR so it was easily implemented. The second pair completed the extension in a short period mainly because JavaScript was the main language involved. | The steps to complete the prototypes were documented and made it easier to understand what had been done. However when extending the code it was difficult to understand coding. In C# the unfamiliar layout made it hard to read. | Adding the avatar button was easily done | Was always fast | yes |
| Java | Most of part 1 consisted of researching for a suitable API for the chat function (Atmosphere was used). Chat was easily developed but after atmosphere was used it became difficult to use the code. | In Java it was the Atmosphere's code that was difficult to read. | The code was difficult to extend due to the complex coding in the Atmosphere API (used for chat). | Was always fast | The chat prototype was real-time but extension (i.e. adding avatar button) was not. In the end it was not solved. |

# Rationale for choosing C#

Before the project team starts developing the new UpStage, we need to decide which programming language to use for developing it. The existing UpStage has been developed with Python. The clients have said that the team does not need to stick with Python for the new UpStage. They will be satisfied as long as the team completes this project and delivers a high quality product. All team members are familiar with Java as BCIS students in AUT are taught programming in Java, one of them is used to C# through her work experience and some of them knows a little bit of Python as they have been working on the maintenance of the existing UpStage. All members might want to use the programming language they are most familiar with, but what we need to consider is whether it is the best programming language that is suitable to generate the required product with good quality.

Each team member has chosen one programming language and researched on it to find out its advantages and disadvantages based on several criteria we have decided, so that we can decide which language is suitable for the new UpStage. The researched languages are Java, C#, Python and Node.js. We have also made simple prototypes to see how easy they are to implement and if they support real-time with multiple users. Each of these has its pros and cons, but we want to discuss why we should use C# and how it will work for UpStage.

The criteria that we have decided on are Readability, Testability, Learnability and Performance. To ensure that the code will be easily maintainable by future members of the UpStage team, the language chosen must be readable. It should also be compatible with popular or good testing tools to aid in testing. Ideally, the language is easily learnable as well, to minimize potential upskilling time of current and future members. Finally, it has to be able to support asynchronous communication and not take up too many resources, as being real-time is a criterion for the final product. However, as UpStage only has roughly 10 to 30 audience at a time for a given stage, this is not a high priority criterion. Instead, more emphasis is placed on testability, as being easily testable will help ensure that the project stays manageable and successful in the long run.

The criteria for the prototypes are Ease of Use, Readability, Extendibility, Performance Consistency and Real-time. Based on the research criteria, as we value the Testability, we decided that Java or C# should be used. The team developed prototypes for both languages in groups. The prototype is a simple chat room application, and after it is done, the other group tries to extend the prototypes with the functionality of displaying or hiding an image. These prototypes must be able to perform in real-time, and we tested this by opening multiple windows of the chatroom and seeing if anything done in one window is reflected in the other window.

C# is a modern, type-safe, high level object oriented language that is developed by Microsoft, running on top of the .Net framework. As UpStage is a web-based application used for online performances, the performers’ performance must be seen by the audience immediately without a page refresh. A technology that can make real-time web applications is required for UpStage and the SignalR library for C# allows this. Of course, other programming languages also have tools that provide real-time function such as Atmosphere for Java or twisted for Python, and Node.js supports real-time by itself.

Java and C#, being structured languages which use certain syntaxes such as curly brackets, make the code more structured and easier to follow. Scripting languages like Python and Node.js do not have this syntax, and thus code written in those languages can be seen as messy or overwhelming by people not familiar with them. This will become more and more relevant as the code base grows. This is an important thing to consider as making the code easily readable will improve the maintainability of the code, and makes it easier to locate the source of bugs. However, Python and Node.js code are shorter due to the lack of syntax, which makes them easier to read in their own rights. As the project team is more experienced in Java and structured languages in general, we feel that the advantages of C# are more applicable than those of Node.js and Python.

When we tried to extend the prototypes, we had difficulty extending the Java prototype as the real-time library we used, Atmosphere, was confusing and hard to read. On the other hand, the code for the C# equivalent which is SignalR, is very short and easy to read and understand. This ties to the ease of use of the two languages; Java and C# is relatively similar in terms of their ease of use, but the real-time libraries we found is vastly different from each other. As SignalR code is more organized and easier to read and implement, we found that C# is overall more easy to use and readable in this project.

Ease of testing is also a big part of the reason we chose C#. C# has access to an excellent IDE with powerful debugging tools in Visual Studio, and the debugger is easy to use. Python has plug-ins for existing IDEs such as PyDev for the eclipse IDE that allow testing of Python code, while successfully testing Node.js code requires getting and learning how to use 4 components, namely the testing framework, an assertion library, stubs and a module control tool. These are unnecessarily complicated, which is something we want to avoid.

Learnability is another factor to consider. Node.js is written in JavaScript, which is a relatively simple language to learn. Some of the papers taught at AUT also use JavaScript, which means it is likely that current and future team members have some knowledge on JavaScript already, and thus reduce the amount of upskilling needed. The front-end of the current UpStage is also written in JavaScript, which further increases its advantages. Team members should already be familiar with Java as it is the main language taught in AUT, so it also requires minimal upskilling. Python is a required learning regardless of the language we decide to use, as the current UpStage uses Python, and it needs to be maintained. However, for future team members, when the current UpStage no longer needs to be maintained, Python will no longer have this advantage. C# is similar object oriented, which is the same as Java, and thus it is relatively easy to learn.

In terms of performance, Node.js is ideal as it is relatively lightweight, unlike Java and C#. It is also faster than those languages, because it does not utilize blocking threads and thus do not have too much downtime. Also, unlike Python which is interpreted during runtime, it does all processing beforehand, thus ensuring that performance is consistent during runtime. While Java can be faster in certain cases, it is prone to inconsistent performance as classes are only loaded when first referenced, or garbage collection can happen at random times, which can cause performance spikes. Node.js also has non-blocking I/O which ensures smooth performance during runtime. C#’s SignalR library handles these and ensures asynchronous communication at all times, even with multiple users, and this is a good thing to have as the clients have expressed that they want a stable performance with varying numbers of visitors. This means that while both Java and C# can perform real-time operations, C# is more stable in terms of performance.

There are other advantages we did not mention here but it is worth to choose with these reasons. As developers ideally want to create small and fast micro-services that are easy to maintain, C# is a good choice for its ease of implementation and testability. It will help the team make a great application that is a good quality and easy to maintain. Node.js can still potentially be used for certain parts of UpStage if needed, particularly parts which require consistently high performance or do not require rigorous testing but C# is the main language that will be used.