-------------------------------------------------------------------------------------

Tom Kilburn wrote the first piece of “software” after building an early computer with his c olleague, Freddie Williams. This program took 52 minutes to find the greatest divisor of 2 to the power of 18 using a series of bits sent to addresses in the machine by simple keywords.

(Yost, 2018)

The term software was coined by John Tukey in 1958 in an article about computer programming following the construction of Fortran in the year prior, a language in which used more human-readable compute lines that would be sequenced into binary that the machine could operate.

(Tootill, 2021)

After the big boom of personal computing, software users got more and more mainstream, catapulting the tech industry forward with popular demand. Many languages came and went, but notably one language made decades ago is still used today—C#.

C# was made by Microsoft’s Anders Hejlsberg, and originally had the name COOL standing for C-like Object Oriented Language, which was a descriptor for the functioning and use-case of a language like C#. Unfortunately, Microsoft lost the COOL moniker due to legal issues.

(Pluralsight, 2019)

-------------------------------------------------------------------------------------

C# 1.0 was developed as a modern and general-purpose object-oriented language. One big advantage it had during initial development was its similarities to Java. That being said, much of the foundation is similar to how it exists today—be it a simpler version. C# came with all sorts of goodies like Classes, loops, conditional structures such as if statements, and variable types like int, float, string, and more. Particularly with Version 2.0 in 2005, support for things like Null values and Iterators got added and significantly improved. These patches and more significantly helped the user accomplish what they needed while cutting down the code they needed to get there.

(erikdietrich, 2020)

-------------------------------------------------------------------------------------

Much of how we used C# back then is still applicable now. Given that C# is an object-oriented language at its roots, it still retains that core philosophy going forward. Now in 2021 the language hasn’t and will not change the simplicity of Object orientation. It’s still one of the driving features to enable inheritance, or classes among objects to give them association—and that’s one of the selling points.

Other than that though, C# has changed a healthy amount from when it was introduced back in 2000. The syntax looks relatively similar, but there have been quite a few changes to cut down on the needed development time in order to accomplish tasks. Additionally, C# is constantly updated to keep trends current and efficient. Support for .NET is especially popular nowadays, as it makes developing your online packages easier than ever.

(Beladiya, 2019)

Multiculturalism is a heavy topic around the world. For good reason. There are many pros and cons to seek out different types of groups and people to work on your project. If you already have a bunch of people you grew up with working on your project, why should you hire people you don’t already know? Well, there are many good reasons for it—especially in software development. Organizations and companies around the globe like NASA and Microsoft make it their goal to create a team of individuals from different backgrounds and experience levels.

(Microsoft, 2021)

(Shea, 2020)

-------------------------------------------------------------------------------------

There are quite a few reasons why you *should* incorporate a more culturally rich perspective on your team. First off, adding more people to your team that may not see eye-to-eye could help your members (and you as well) grow as people. Varying perspective team members could also give more insight into development trends and solutions to incorporate a system that you wouldn’t regularly get with your originally planned team. Having many different people on your team could also mean that you can include a larger target audience, as people from that area of life could give valuable insight into development, and build features that may be extremely useful to your clients.

-------------------------------------------------------------------------------------

With all this valuable insight, adding more diverse staff to your team may also come with some problems.

First off depending on your inclusivity, you may also encounter a language barrier. As Different people speak different languages, there’s no guarantee that everyone you meet speaks the one you do—especially with the world getting more and more diverse by the day.

Similarly, there may be other reasons a particular employee may be unable to perform a certain task, or work on a certain day such as religion, or moral practices.

Additionally, different perspectives can lead to disagreements amongst your staff. These disagreements can vary in their severity, but can often times lead to members feeling unheard, or left out. Another potential problem with hiring more staff is an oversaturation of input. It’s entirely possible that when you’re developing your software team members keep giving input to move towards what they think would be ‘better’ for the program in the long run. If members of your team have differing opinions on the subject, then it’s possible this feedback will not only continue, but past features and iterations will get more verbose feedback as well. This could be a problem especially when your team is on a deadline. After all, in most cases, you will not have unlimited time to work on your project—certain problems need to be weighed in order of importance.

-------------------------------------------------------------------------------------

Although there is no “solution” for anything like religious actions or moral integrity, there are a few problems listed previously that may have reasonable solutions. Let’s take disagreements for example. No matter where you work, you will disagree with something. There are many ways to mitigate potential outcomes of this disagreement, but overall, it will happen. One way to make sure it doesn’t end in less staff is to encourage an open discussion. This could be directly with the individuals in conflict, or it could be separately with supervisors. From there, a resolution to this problem is based almost entirely on compromise. In order to ensure that each and every employee feels respected and cherished, it’s important that you incorporate everyone’s voice in the actions of a workplace—that means no one gets left out. Also, on the topic of feedback saturation, there are a few steps you could take as well. Make it clear to your team exactly what problem they will be addressing, and what deadlines they are expected to work under. When feedback comes up, it is important that the employee understands whether or not the team should action their proposed solution, and why that decision has been reached. This should facilitate a more open and honest work environment that everyone can be comfortable in.

# **References**

Beladiya, K. (2019, December 12). *.NET Trends to Dominate in 2020: Top Frameworks For Web Development*. Retrieved from The One Technologies: https://theonetechnologies.com/blog/post/microsoft-net-core-and-web-trends-to-dominate-in-2020

erikdietrich, B. Y. (2020, April 8). *The History of C#*. Retrieved from Microsoft.net: https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history

Microsoft. (2021, June 6). *Global Diversity and Inclusion*. Retrieved from Microsoft.net: https://www.microsoft.com/en-us/diversity/default.aspx

Pluralsight. (2019, May 15). *Everything you need to know about C#*. Retrieved from Pluralsight.com: https://www.pluralsight.com/blog/software-development/everything-you-need-to-know-about-c-

Shea, G. (2020, October 8). *Diversity & Inclusion*. Retrieved from NASA.gov: https://www.nasa.gov/offices/odeo/diversity-and-inclusion

Tootill, G. (2021, June 6). *The Original Original Program*. Retrieved from CCS: http://www.cs.man.ac.uk/CCS/res/res20.htm#e

Yost, M. (2018, January 25). *A Brief History of Software Development*. Retrieved from Medium.com: https://medium.com/@micahyost/a-brief-history-of-software-development-f67a6e6ddae0