Uses and relevancy of old and new programming languages

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Bachelor thesis

Degree Programme in Computer Applications
Autumn, 2023

Stefanija Morozova

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Computer Applications Abstract

Author Stefanija Morozova Year 2023

Subject Uses and relevancy of old and new programming languages

Supervisors Lasse Seppänen

ABSTRACT

The purpose of the thesis was to analyze the of uses and necessities of programming languages of different generations. The thesis sought to analyze the connection between a programming language's function, research volume and possible industry growth and compared to the industry in use. The research questions were what separates the old languages from the new, how the traits of these languages affect that, and which industries would these languages grow in. Programming language history and Industry growth are also discussed. The thesis was commissioned by the HAMK University of Applied Sciences.

This thesis is practical and focuses on analysing data and basing conclusions from that analysis. First, the thesis defines the scope and timeline of the research, as well as the definitions of each generation. Second, the central concepts related to each generation of programming languages are explained. Four generations in total were defined. The primary research method was applying the theoretical base to data gathered regarding the documentation of that language and the industries in which they would work best. The data was analyzed by the author using EXCEL.

The research demonstrates that neither generation has the upper hand in popularity or relevance, rather each one has one or two well-known and relevant languages which propagate the amount of research is done on that language. The analysis indicates that both old and new programming languages are still in use. Due to the limited scope of the thesis, this research can be reviewed and redone with more accurate data.

Keywords Programming languages, industries, research, documentation, analysis.

Pages 52 pages

Glossary

AI - an area of computer science that deals with giving machines the power of a machine to copy intelligent human behavior. (Britannica Dictionary)

Classification - the act or process of putting people or things into groups based on ways that they are alike. (Britannica Dictionary)

Compiler - a computer program that changes instructions into machine language. (Cambridge University Dictionary)

Computation - the act or process of calculating an answer or amount by using a machine. (Cambridge University Dictionary)

Documentation - the documents, records, etc., that are used to prove something or make something official; written instructions for using a computer or computer program. (Britannica Dictionary)

Function - a process that a computer or a computer program uses to complete a task. (Cambridge University Dictionary)

Interpreter - a computer program that changes the instructions in another program into a form that can be easily understood by a computer. (Cambridge University Dictionary)

OOP – Object-Oriented Programming (Cambridge University, 2017)

Operator - a symbol that does something to a number or quantity in a calculation. (Cambridge University Dictionary)

Parameter - a rule or limit that controls what something is or how something should be done. (Britannica Dictionary)

Protocol - a set of technical rules that control the exchange of information between different computers or computer networks. (Cambridge University Dictionary)

Statement - a line of information in a computer program. (Britannica Dictionary)

Timeline - a written representation of a period of time. (Cambridge University Dictionary)

UI – User Interface - the way in which the information on a screen and instructions on how to use it are arranged on the screen and shown to the user. (Cambridge University Dictionary)

Web-development - the process/technology of creating websites. (Cambridge University Dictionary)

Contents

Table of Figures

1 Introduction

With the growing documentation uses and engineering of programming languages it is difficult to keep up with trends in specific branches of the programming world. Both old and new languages are still being used today. It is difficult to properly understand the relevancy, purposes, and implementations of some languages and how these languages are used and why.

While many new programmers and students are jumping on the newer and more popular programming languages, it is not necessarily set in stone that those are the only ones they should learn. Especially if they plan to stay in an industry which is prone to change. It could also be said in the opposite way that some industries still use old programming languages to function. Do not try to fix something that is not broken, or so the saying goes.

This thesis is relevant and functional to institutions who train programming languages. The thesis will provide conclusions concerning companies who use older programming languages and industries that use newer programming languages, comparisons and definitions for older and newer programming languages, and the quality or quantity of usage that separates the two. Or, depending on the results of this thesis, whether how programming languages are classified should be rethought.

The research questions of this thesis are as follows:

- What are the main differences between “old” programming languages and “new” ones?
- How do these differences affect their usages and research?
- Which industries benefit from these differences and documentations?

2 Programming Languages

2.1 Timeline

This is the main theory-base chapter. The knowledge base presents what is already known about the topic and what kind of similar products, products or processes have been made in the past. The starting point is evidence-based, for example studies, reviews and recommendations are used, but not one's own opinions. Before the languages that will be studied can be properly assessed, the parameters and definitions of them need to be defined. Since data of old and new programming languages will be gathered, what is old and what is new must be defined, and then pick out class/family representatives of each one, to make sure the language groups are varied enough.

The definitions of old and new programming languages differ from source to source. Some sources use timelines, stating that old programming languages are the ones that originated in the 1960s. Others use syntax to define the difference – old ones are simple, new ones are complicated. (Pigott, 2006)

Both definitions have counterarguments to them. There are new languages which are simple and old ones that are complex. In addition, some languages still evolve and even if they were founded/created many years ago, they are still considered “new” in some cases, due to constant development. Some languages are branches of other, more complex, and older languages. (Borkowsk, 2006) (E & ICT Academy, 2018)

Because of this, the thesis must properly define what is considered an “old” programming language and what is considered a “new” programming language. In the limitations of this thesis, programming language classifications used will have “middle ground” and “non-classifiable” groups.

Within the scope of this research, there is no concrete author or research paper from which the generation classification came from. It is likely that the definition of programming language generations was based on the definition of generations as a product – a group of machines that are all at the same stage of development. (Cambridge University, 2017)

Within the scope of research, previous knowledge, and theoretical reviews, programming languages can be defined through generations of development. This paper will primarily focus on 3rd generation languages and 4th generation languages. The following research and work will not include 1st, 2nd or 5th generations. 1st and 2nd are outside of our research scope/are not applicable, as they were made entirely in binary numbers for 1st generation and limited only to assembly languages in 2nd generation. (Bartoníček, 2014, pp. 94-99) 5th generation is not in use widely and requires to be automated and more focused on self-processing actions, like AI and are too young to properly analyze. (Goel, 2010, pp. 3-6)

While some sources try to categorize some representatives of the 3rd generation into the 4th generation, this statement ignores the differences in function, classification, and protocols that 3rd generations have that no longer apply to 4th generation. There are also languages that fall both into 3rd and 4th generation through similarities and/or exceptions, hence the need for a middle ground and non-classifiable group.

However, a timeline for defining both old and new languages must be adhered to. The research cannot go too far back or else there will be too many languages to work with and cannot have the timeline be minimal due to the risk of missing some important languages. The first commercially used compiler language, Fortran, was made and used all the way in 1957. (Backus J. , 1978) For the end of the timeline, this year (2023) cannot be considered. However, 2022 also cannot be considered. This is because the timeline would fall into a prime number of 67, so the closest divisible number below it must be used, which is 66.

With that, 1955 will be defined as the beginning of our timeline, as about 2 years before Fortran, the first 3rd generation language in the scope of the research, and 2021 as the end of the timeline, to give enough time to see when the 3rd and 4th generations started developing, for how long were both used, when the 4th generation started overcoming the 3rd generation and what the state of the 4th generation was a few years later. (Kahanwal, 2013, pp. 6-11) (Brookshear & Brylow, 2020, pp. 320-389) (Sammet, 1972, pp. 601-610)

This timeline will also give enough time to define when middle-ground and non-classifiable languages started forming and what their role was.

2.2 Classification

Now that the timeline of the languages has been defined, the languages must be classified and the scope of classification must be defined . Due to the many differences and similarities between 3rd and 4th generation languages, they must be accurately classified in a way that allows us to neatly put them in each category, or in the very least find similar categories of function. Both generations also have a focus in different industries and specializations. The functions and capabilities of both are also vastly different in some cases and similar in others. As such, the scope of each group will be first defined.

3rd generation programming languages are (3GL) – high level programming languages originally made as an evolution to second generation machine programming. These languages were primarily made for easy and simple coding syntax for efficiency, debugging, portability and problem-oriented coding. Since the original goal of the generation was to make high-level language instructions that could be translated into machine language instructions, most 3GL can be sorted into Compilers, Processors, Translators and Interpreters. (Kahanwal, 2013, pp. 6-11) (Brookshear & Brylow, 2020, pp. 320-389)

This is mostly due to how they execute code and as such have more than one characteristic but follow mostly the same pattern – they take the written code, either interpret or translate it into a sequence of one or more subroutines or into another source code, and then into another language, compile human-readable code into computer-readable instructions or process it using another program. Notable examples are FORTRAN, COBOL, Pascal, C and BASIC. (Brookshear & Brylow, 2020, pp. 320-389) (Chowdhary, 2020, pp. 1-6) (Goel, 2010, pp. 3-6)

4th generation programming languages (4GL) – are defined as the next step in both evolutions and advancement in programming languages, as a step up from 3GLs. Some 4GLs were originally made and developed on the basis of 3GLs, like how C influenced so many languages and how many 4GLs were based on it. The basis of many other 4GLs is also valuable to note, as many technical representatives of 4GLs can be considered branches from 3GLs. This is also valid factor in questioning which languages are 3GLs, which are 4GLs and which ones are hybrids of the two, as there have been previous correspondents in past publications and findings, who were unsure of

what generation of languages they were using. (Alzahrani, 2020, pp. 178-185) (Goel, 2010, pp. 3-6) (Sinha & Sinha, 2021, pp. 1-16)

A notable issue is that since there are so many classes of this generation, there's a variety of languages and often makes the generation a mixed bag when it comes to industry focus and classifications, which will be spoken about in the next chapter. 4GL has 6 branches of focus: data input, data management, data analysts, data output (including reporting), graphics and user-oriented interfacing concerning the utilization of windows. (Kahanwal, 2013, pp. 6-11) (Alzahrani, 2020, pp. 178-185)

Research and comparative analysis shows 4GLs can be split into 5 classes: programming extensions for operating systems using command interpreters (Interpreters), database management and query uses (Query-Based), productivity and production focused pre-compiler tools (Compilers), data output (Reporters), graph makers and graphics generators (Graphics). Some examples of 4GLs, without classifying, are SQL, Oracle Screen Painter & Forms and Reports, operating system languages, Wolfram Mathematica and FOCUS. (Kahanwal, 2013, pp. 6-11)

Middle-ground generation languages (MGL) – in the case of this paper, are defined as having some qualities of both generations, while being withing the timeline scope. Multiple well-known and still present day used languages are represented by this category, most of the time for their varied and versatile nature while still being simple to use and learn. Notable examples that will be further discussed and studied are Python, Ruby, R, MATLAB and Perl. (Mathworks, 2023) (Python Software Foundation, 2023) (Maeda, 2002) (Christiansen & Torkington, 1998)

Non-classifiable generation languages (NGL) – in the case of this paper, the languages and forms of programming that are in this group are those that are outliers in both 4GLs and 3GLs, with non-classifiable or no common traits between both generations but still falling within the defined timeline. As per definitions, this can include languages focused on website development. This class includes Java, Javascript, .NET, VisualBasic, XML, HTML, CSS and PHP. (Mendez, 2014, pp. 1-4)

2.3 3GLs

2.3.1 Functionality and Purpose

3GLs main functions were primarily prominent in the beginning of the 1970s and up to the 90s until newer generations and generation hybrids popped up. This chapter will look into the functionality and userbase of 3GLs to accurately see which industries used which languages and how the previously mentioned classifications were relevant in such spheres. Afterwards, the subchapter will look into user progressions – which were most commonly used and whether they still stayed relevant. Finally, an overview of the main representatives of the generation will be presented.

3GLs were comprised of protocols and functions that would shorten 2nd generation machine computing. They were considered the first high-level computing languages – assigning English phrases to specific codes and then executing them through interpretation, compilation, translation, or processing. (Brookshear & Brylow, 2020, pp. 320-389) They were primarily procedural languages that allowed coding to be much easier and efficient. (Kahanwal, 2013, pp. 6-11)

Translators and Interpreters were similar to one another in function where code in 3GLs were translated or interpreted as 2GLs. The same could be said about Processors and Compilers – either compiling the code or processing it. Translators and Interpreters focused more on the language of the code and what it equated to in 2nd generation, while Processors and Compilers go and execute/run the code all in one go. Each class has its weakness and strength, be it processing time, debugging, error catching or processor and memory load. (Chowdhary, 2020, pp. 1-6) (Alubady, 2009, ss. 1-7) (Paquet & Mokhov, 2010, pp. 33-37)

As such, the 3GLs were developed with the intention of easing the process of coding – making it easier to create it and work through the program logic, rather than crunching and working through numbers. Its main functional perks were that it took less time to create programs, easier to avoid and catch errors and was easy to develop, learn and understand. It was mainly made as an easier step up from the 2nd generation. (Chowdhary, 2020, pp. 1-7) (Kahanwal, 2013, pp. 6-11)

3GLs used to cover virtually all spheres of development – scientific programming, object-oriented programming, business data processing, visual programming, and system programming. For a while, they were used everywhere, but it can be seen from the main representations of 3GLs that they were primarily scientific and business oriented. (Kahanwal, 2013, pp. 6-11) Fortran's name is a combination of the term Formula Translation. (IBM, 2012) COBOL is an acronym for COmmon Business-Oriented Language. (Paquet & Mokhov, 2010, pp. 7-27) Pascal is based on the ALGOL programming language, meaning Algorithmic Language, and named in honor of the French mathematician and philosopher Blaise Pascal. (Paquet & Mokhov, 2010, pp. 7-27) BASIC stands for Beginner's All-purpose Symbolic Instruction Code. (Paquet & Mokhov, 2010, pp. 7-27) Even C was named based on its predecessors – B and CPL (Computer Programming Language) but still has ties from ALGOL as well. Most 3GLs were based off of later iterations of Fortran, with the exception of COBOL. (Borkowsk, 2006) (E & ICT Academy, 2018)

Most of these languages were used in scientific or business spheres, some more than others, but to the point where each classification had a representative of sorts – Fortran and ALGOL being a most obvious Translator, C and JIT (Just in Time) belonging in Compiler class, LISP and COBOL belonging in Processing and Interpretation being left to BASIC and Pascal. (Paquet & Mokhov, 2010, pp. 7-27)

All-rounders like C and Basic were used in many areas, either for educational purposes, system programming or visual programming. They were most prominent in the IT industry and webpage development cases. C has also been involved in research and scientific development fields, and was even used to create C++, an object-oriented programming language. Basic was often a necessity for programming custom software. (Kahanwal, 2013, pp. 6-11) (Glassey, 1993, pp. 1-6)

LISP and COBOL were prominent in the Business and Finance field, due to predictive programming and AI development and large dataset processing and reports with the ability to process lists and arrays. Pascal, which has a mathematic focus, was also included in this field for more accurate finances and calculations. (Kahanwal, 2013, pp. 6-11) (Flahive, 2019) (Carr & Kizior, 2003, pp. 1-10) (Wirth, 50 Years of Pascal, 2021, pp. 39-41)

C, Fortran, Algol and even in some cases Pascal were involved in areas needing accurate and quick mathematical and statistical results. Notable cases were Fortran and ALGOL in Computational

Physics and Chemistry, C in advanced program and software development and Pascal for Object Oriented Programming. (Kahanwal, 2013, pp. 6-11)

2.3.2 Progression of Usage

As mentioned in the previous subchapter, several 3GLs were used for many things. However, which language was most used still needs to be understood. When they started falling back and which ones are still relevant also must be defined.

While all the users of specific languages cannot be accurately counted, especially with older languages that existed much before the uses of it can be properly documented. Google trends can only take us back so far and even then, uses only the number of times the term for the 3GL was searched, not how proactively it was used or useful. However, archives and indexes directly responsible for documenting popularity of multiple languages are available – the TIOBE index, which calculates the number of lines of code written monthly and ranks it. While the total amount of 3GL users cannot be grasped, how popular and widespread each language representative was by this index can be gauged. TIOBE's index has a timeline starting from June of 2001 to this month of this year (April 2023) and a total of 50 observable places in the ranking, so we'll be basic popularity on that. The languages used will be one representative from each classification – Fortran, C, COBOL, and Pascal, although the variation of Object-oriented – Delphi, will be used.

Fortran's highest peak was observed to be between June 2001, when the index was first made and to March of 2002, with a high position of 10th in the ranking. Afterwards, the language slowly dropped in popularity until it reached its 3rd lowest point in August 2004 before rising widely in popularity in April of 2005. This followed many fluctuations in uses, with low points in February 2008, January 2015 and later July of 2020. As of November 2021, to this month, it has been jumping back and forth in popularity between 10th and 20th and is currently the 20th most used language. (TIOBE Index, n.d.) Its usage position has been in chaos ever since the first documentation, despite a survey in the Mathematica journal by Stephen Wolfram showing that at least 1 in 2 people who had knowledge of programming languages knew the basics of Fortran. (Wolfram, 1990, pp. 14-16)

C's rankings show it is still very widely in use – always either holding the position of 1st or 2nd in the ranking, making it one of the most widely used languages. Its lowest point being in August 2017 and March 2023. (TIOBE Index, n.d.)

COBOL is in a situation of going between gradual increase and gradual decrease – its highest position in popularity was in August 2001, being in the top ten at 8th position, making it used in 1 in 8 programmers. It steadily decreased until June 2005 to go back into the top 10, then decreasing again before rising back up in January 2015. Now, COBOL has been at a position of a hallway point of the top 50 ever since 2020. (TIOBE Index, n.d.)

Pascal/Delphi is in a similar situation as COBOL – its highest point was in October 2001 in 6th place and had risen more in popularity in August 2004. However, it has been in stagnation ever since. (TIOBE Index, n.d.)

2.3.3 Main Representative Features

Fortran was developed in 1954-1957 in IBM (International Business Machine Corporation/ Big Blue) by John Backus. Fortran was the first commercially used programming language in engineering and the scientific field. (IBM, 2012) It has a simple syntax for programs, similar to typing out commands in English, but in most versions' cases, there's a limit to how many lines and characters it can write, as well as the commands it can use. While it supports integer, real, logical, and complex numbers, most of its functions and operators are limited to data and data arrays. (Clarke, 1982, pp. 10-16) (Page, 1988) For an order to run on Fortran, it needs to start with a statement of what it will do (PROGRAM, FUNCTION, SUBROUTINE, MODULE, or BLOCK DATA), followed by USE statements, indicating what array or data will be used, FORMAT and ENTRY parameters, additional statements or subprograms and an end statement. (Adams et al., 1992)

COBOL was developed and released in 1959, despite having many predecessors older than Fortran. It was developed in CODASYL (Conference/Committee on Data Systems Languages) by many scientists. A group consisting of Burrough Corporation's computer scientist Mary K. Hawes, the inventor of FLOW-MATIC Grace Hopper, Jean E. Sammet and Saul Gorn had developed it, funded by the United States Department of Defense with the support of its director of the Data System Research Staff Charles A. Phillips. (Beyer, 2009, pp. 289-297) (Gürer, 2002, pp. 175-180)

Compared to Fortran, is far more flexible and has more syntax to work with, but in turn is much more difficult to manage in terms of program organization. COBOL is also different by using four main statements of what the code will do - IDENTIFICATION DIVISION, ENVIRONMENT DIVISION, DATA DIVISION and PROCEDURE DIVISION. Usually, you can use all four but it is not a necessity. The DIVISION in these cases are similar to function brackets but are entirely limited to the sentence below them – the function of the division is not limited to paragraphs but to which statement ends with ".". (Ferguson, 2018) (Coughlan, 2014)

ALGOL was developed in 1958 by both US and European computer scientists in the Swiss Federal Institute of Technology in Zurich. (Backus J. W., 1959) As previously mentioned, has ties in both Pascal and C, so they are similar in syntax and structure. ALGOL's structure is what many programmers favor today, using indented paragraph spacing and labels like program, begin and end. It also holds a much wider array of functions and statements. (Chowdhary, 2020, pp. 1-6) (Paquet & Mokhov, 2010) (J. W. Backus, 1960, pp. 299–314)

Pascal was designed by Niklaus Wirth with Helmut Weber in 1965 for a dissertation on Euler. C was created by Dennis Ritchie in 1973 for Bell Labs with the intent of constructing utilities running on Unix. (Wirth, 2000, pp. 1-10) Pascal and C have equal parts similarities and differences. In terms of syntax, Pascal is more like ALGOL, using most functions as words, compared to how C uses "&&" and "%" in place of "and" and "mod" in Pascal, for example. In Pascal, semicolons separate individual statements within a compound statement; instead in C, they terminate the statement. Comments are formatted differently, as are implementations and keywords, including control structures - C uses more brackets than Pascal. However, C has more expressions in evaluation and expression than Pascal. (Rall, 1987, pp. 53-69)

Taking the history, functionality and purpose that 3GLs have, an educated guess, that 3GLs had focus in scientific and business computation and use the data they're given as simple values, can be made.

2.4 4GLs

2.4.1 Functionality and Purpose

4GLs main functions were primarily prominent in the beginning of the 1990s and are still widely used even to this day. (Alzahrani, 2020, pp. 178-185) This chapter will look into the functionality and userbase of 4GLs to accurately see which industries used which languages and how the previously mentioned classifications were relevant in such spheres. Afterwards, the subchapter will look into user progressions – which were most commonly used and whether they still stayed relevant. Finally, an overview of the main representatives of the generation will be presented.

Compared to 3GLs, 4GLs were much more advanced both in what they could do and be used for, resulting in more active learning and usage of 4GLs. (Ketler & Smith, 1992) There has been prominent confusion in which languages had belonged to 3GLs and which to 4GLs which has led to identification of middle ground/ hybrid languages. The classification and understanding of this generation will be solely by what the official definitions are.

4GLs are a step up from 3GLs but rather than focusing on computing data, the 4Gls focus more on managing data. This includes visualizations and logistics, system management and even application generators. (Kahanwal, 2013, pp. 6-11) There are many versatile languages in this classification which makes it difficult to pinpoint the exact industries these languages excel at, as there is a majority that are based on database usage and data management and an equal amount focused in general use. (Alzahrani, 2020, ss. 178-185)

As previously mentioned, 4GLs have a focus in anything concerning data, graphic makers and generators, productivity and programming using system commands. From a glance, it can be seen that many 4GL languages would be used in business and or corporate oriented environments, as many languages focus on managing data and, using graphic generators, visualize the patterns of that data. (MacDonell, 1993, pp. 1-14) However, with system management and software development being a focus in this as well, it would be wise to include that software development companies would also focus on using 4GLs. (Kahanwal, 2013, ss. 6-11) (Sammet, 1972, pp. 601-610)

Query Based, Reporters and Graphics can be used in almost every industry, as most languages in this classification focus on managing data in databases, output them into reports and visualize said reports. (Heering & Mernik, 2002) Compilers and Interpreters that are used in operating software code and even software development may be used more in IT, but for companies like Oracle who have a diverse product line of software, they can be used virtually everywhere. (Dale et al., 2021) Microsoft and Linux are on any computer, and customer service for these operating systems makes it a necessity for each computer to work with. (Economides & Katsamakas, 2006, pp. 208-218)

It could be said these are necessities for corporate function or simply making evidence-based conclusions. For example, for things like end of year reports.

Query Based languages like SQL PL (Structured Query Language Procedural Language) and FOCUS, the successor of RAMIS, Random Access Management Information System, and the first 4GL, are both used in database management and queries concerning it, either through MySQL or through Excel. (TIBCO, 2022, pp. 1-6) (Information Builders, 1990, p. 12)

No industry since its release has been without Excel, mainly because it was an easily manageable interface and included chart building, data sorting, cleaning, and management. MySQL is largely used in a large variety of sectors – information and technology sectors, financial services, marketing and advertising, telecommunications, even hospital and healthcare. (Gration, 2022)

Reporters consisting of tools like Oracle Reports, are also a staple in many industries and Oracle itself is a widely spread family of software tools. (Oracle, 2020, pp. 1-18)

Wolfram Mathematica is used in functional programming, rule-based programming, plot and graphics creation, image processing, calculus, data analysis, science, engineering, financial engineering and even testing and debugging. This made it widely used in mathematical sciences, engineering, business and social science. (Wolfram, 1990, pp. 14-16)

With all this in mind, industries of 4GLs cannot be accurately narrow down all the as they are too widespread through too many industries, especially in the current age where data management and reports are in every company.

2.4.2 Progression of usage

In the case of 4GLs, due to their wide use through the software they are implemented in, the progression of use of FOCUS, SQL- based languages and W.Mathematica used software and systems – Excel, MySQL and Wolfram Alpha – will be calculated. Operating system languages unfortunately cannot be analyzed in terms of usage popularity, likely due to the fact we'd be calculating the popularity of the actual software, which is not included in this paper.

FOCUS is largely part of Excel, which is arguably the most popular data focused software to this day, and to this day, is still steadily growing. In the span between its release year, 1985, and a decade later, in 1996, the number of users had grown to 30 million users. By 2022, this number has risen to over 1 billion – 1 in 8 people across the entire globe. (Gration, 2022)

SQL's most popularly used database system is MySQL. MySQL is used most of all in three sectors – finance, government industries and IT. Enlyft, a real-time data proprietary platform, estimates the number of companies that use MySQL to be 263,223, with the most common company – over 62 thousand companies- having anywhere between 10 to 50 employees. Even the number of companies with over 10 thousand employees are just shy of 5 thousand. (Enlyft, 2023)

Mathematica's most known software is Wolfram Alpha. Wolfram, as a company, has more than just Wolfram Alpha and has a variety of other types of software. (Wolfram, 2023) Because this is an open-source software, the basic version of which is available to all, the number of users cannot accurately be guessed and as there is little metadata with such information, it can only be estimated to be between Excel's massive number and MySQL's notable size.

2.4.3 Main Representative Features

At the current moment, there is little to no easy way to find code documentation of FOCUS specifically for programming. While many manuals of new releases are available, very few show the code functions. As such the main features of how code can be used in Excel but formatted like SQL will be analyzed. FOCUS, at earliest, had manuals trademarked by Information Builders released in 1994. (Information Builders, 1994)

SQL was, like Fortran, originally developed in IBM by Donald D. Chamberlin and Raymond. F. Boyce in the 1970s but after moving laboratories in 1973, it was released in 1974. (Chamberlin & Boyce, 1974, pp. 249-264)

FOCUS has a similar code syntax to COBOL, in which everything is written in a block or a single line of code and, if the code ends in function, should show the result when executed. FOCUS works best using data that is already provided, and has a versatile array of functions, parameters, and abilities. (Information Builders, 2019) SQL works best with databases but unlike FOCUS, it is primarily made for managing the database and its records, rather than manipulating the internal values. (Smirnova & Tezuysal, 2022)

Mathematica was developed by Stephen Wolfram and released in 1988. Mathematica is written and functions differently. As it is more varied and extended in its uses, it is more focused on software development and program execution. Using Mathematica individual instructions in the code can be assignment statements, iteration statements (loops), conditional (if), input or output statements, functions, or other programming constructs. (Wolfram, 1990, pp. 14-16) (Mangano, 2010)

Taking this and the functionality and purpose that 4GLs have, an educated guess can be made that 4GLs had focus specifically in data based analysis, the visualization of the values given and using the data they’re given as objects with more parameters that ca be grouped and analyzed.

2.5 MGLs

2.5.1 Functionality and Purpose

MGLs, as previously stated, are languages that have both 3GL and 4GL traits. This chapter will look into the functionality and userbase of MGLs to accurately see which industries used which languages and how the previously mentioned classifications were relevant in such spheres. Afterwards, the subchapter will look into user progressions – which were most commonly used and whether they still stayed relevant. Finally, an overview of the main representatives of the generation will be presented.

As mentioned, MGLs are a mix of both previously explained generations. They include advanced languages that can both compute, manage systems or create applications and fall into a timeframe of programming language development, where 3GLs were becoming less prominent and 4GLs were coming more into use. (MacDonell, 1993, pp. 1-14) Some are argued over their proper position in the classifications – languages like Python, Ruby and Perl were argued to be 3GLs due to their frame of usage focus in computing and creation, while MATLAB and R were argued to have belonged to 4GLs, as MATLAB was primarily focused on creating graphics and R is used in data manipulation and reporting. However, multiple other sources had stated that the same languages should have been in reversed classifications, with Python, Ruby and Perl belonging in 4GL and MATLAB and R being in 3GL. (Alzahrani, 2020, pp. 178-185) (Brookshear & Brylow, 2020, pp. 320-330) (Sammet, 1972, pp. 601-610)

Overall, the conflicting classifications are not the only factors as to why these languages are in MGL. Most of these languages have specialization in several categories of both 3GL and 4GL, making them prominent in the same sectors of use and industries. However, with 4GLs being used in many sectors due to reporting and software development, the uses and purposes cannot be narrowed down. (Oracle, 2020)

MGLs are a combination of 3GLs and 4GLs – thus, MGLs are languages that can be used in scientific fields due to accurate and simple usage for computation and calculation and can be used for software and application development. (Kahanwal, 2013, pp. 6-11) Due to having multifaceted capabilities, MGLs are, in the current era, one of the most prominent and widely used language

groups for their flexibility and ease. Like with 4GLs, it would be difficult to pinpoint exactly the industries these languages would be prominent in, as they are widely used in many fields, but it is undeniable that they are a core part of computer science, web development and code-based programming. (Alzahrani, 2020, pp. 178-185)

2.5.2 Progression of Use

Languages that represent MGLs are, in most cases, still widely used today. The usage of the languages is documented properly, both in the TIOBE index and other forms of usage measurements.

Python was originally a language that was only in the top 20 of when it was first introduced in the 90s, which on its own was high. Over the years, Python has slowly been climbing the popularity ladder and is now one of the most popular programming languages, tied with Java and iterations of C in most polls. Its lowest point was from February to November of 2003. (TIOBE Index, n.d.)

Ruby has been slowly falling out of popularity over the years. After a low point in August of 2004, Ruby slowly rose in popularity until it reached 3rd place in popularity in 2008. Afterwards, Ruby experienced a steady decline before staying in a state of fluctuation. As of March 2018, it has been in steady decline, currently only within the top 20 of the chart. (TIOBE Index, n.d.)

R, as a programming language, was not well known until well into the late 2010s, picking popularity up only recently in 2015 and experiencing fluctuations until reaching a peak in 2020 August. As of then, it has been in steady but chaotic decline but still maintaining its position in the top 20. (TIOBE Index, n.d.)

Perl's chart data shows that its peak in popularity has long since passed, having peaked in May 2005 in third place but then faced a strong and steady decline in usage. The lowest usage was in November 2022, falling out of the top 20. (TIOBE Index, n.d.)

MATLAB, prior to its popularity in the late 2010s, had spikes in usage in May 2005, December 2007, and November 2011. Afterwards, its popularity slowly rose to a peak at 10th place in December 2018. As of then, has consistently rose and fallen within the top 20. (TIOBE Index, n.d.)

2.5.3 Main Representative Features

Both Python and Perl were influenced by C, but Ruby was influenced by Python and Perl, while MATLAB (MATrix LABoratory) was not influenced by any of them. R was influenced by Lisp. (E & ICT Academy, 2018)

Python was first developed in CWI (Centrum Wiskunde & Informatica) by Guido van Rossum in the late 1980s but was not officially released and used until 1991. (Venners, 2003) It is regarded as a general-purpose programming language. Its most defining syntax feature is the use of indentation and support of multiple programming paradigms. Like the rest of the languages, its typed in the English language. (Kuhlman, 2013, pp. 10-15) Compared to 3GLs, which often had an issue of code line limits, the number of logical lines a Python program can use, and store is not documented to have a limit. The code structure is like C's but differences in syntax can include traits such as commenting symbol differences, with Python using the hash (#), logical line joining using the backlash (\) and Python's stricter indentation rules. There are also differences in data type processing and the interpretation of some symbols. (Python Software Foundation, 2023)

Perl was made in 1987 by Larry Wall. In comparison, it is a bit more different and limited than C and Python in its functions and use. (Wall et al., 2000, pp. 1-43)Like C and Python, it is made to create comprehensible code easily but rather than being an all-rounder like Python, Perl's main characteristics are that its practical and quick, and stable for computation. It is best known for its text processing – its style is very informal and is good for small and quick programs, making it less likely to crash upon failures. Its variables are considered "scalar" – they hold a single string or number and rather than simply being declared, they require the dollar sign ($) to declare it a variable. Despite this, it can still use and declare arrays and lists with the at symbol (@). Like Python and C, the code does require indentations. However, as Perl primarily focuses on quick and small programs, but has more focus on hash functions. (Christiansen & Torkington, 1998) (The Perl Foundation, 2023)

Ruby will seem like Python and Perl, but that's not the only influence. Notable influences from the 3GLs are Ada, BASIC and Lisp. It was developed by Yukihiro "Matz" Matsumoto in 1995 with the intention of being a more developed programming language in regard to Perl. (Maeda, 2002) Some parts are nitpicked from the five and implemented into Ruby. Unlike its predecessors, Ruby

is focused in object-oriented programming entirely with traits allowing for inheritance and class-stability. (Sieger, 2006) The syntax is largely similar to Python and Perl but unlike them, Ruby accesses methods using metaprogramming. (Thomas et al., 2001)

MATLAB is entirely made with a unique language makeup. It is also unique in the form that it is the oldest language of this group, invented by Cleve Moler and took 10 years to finish, the final product being showcased in 1979. (Chonacky & Winch, 2005, pp. 9-10) Like Python, it has many branches of functionality and uses but unlike Python, its script works best in computation and visualization. MATLAB's system, as software, can have libraries written in Perl or Java but not the other way around. Its language syntax uses English like the rest, but the syntax itself is completely different than the rest of the languages of this group, being more variable and text focus, the latter primarily for calling forth visual factors like binary images. Its unique syntax and variable use can be credited for its many branches of data and image processing. (Mathworks, 2023)

Finally, R, like its predecessor Lisp, is used entirely in statistical computing. R was developed by two professors at the University of Aukland, Ross Ihaka, and Robert Gentleman, and first appeared in 1993. (Ihaka, 2022) It is a language used for data mining and analysis with the help of vectors, arrays, data frames, lists and other forms of data structures. (Giorgi et al., 2022) Like Perl, it uses metaprogramming and is accessible through many other languages like Python. Its functions could be contributed to the same fields as MATLAB. However, the syntax is much simpler and does not rely on full-worded commands to visualize the data its processing. (Grogan, 2018)

Taking this and the functionality and purpose that MGLs have, an educated guess can be made that MGLs had no specific focus and that their main and winning feature was being a collective of jack-of-all-trades languages. These languages are easy to port, have many packages and can be easily learned.

2.6 NGLs

2.6.1 Functionality and Purpose

NGLs are languages that arose in the same timeline as 3GLs and 4GLs, but its traits, uses and focuses were not applicable to either generation. This chapter will look into the functionality and userbase of MGLs to accurately see which industries used which languages and how the previously mentioned classifications were relevant in such spheres. Afterwards, the subchapter will look into user progressions – which were most commonly used and whether they still stayed relevant. Finally, an overview of the main representatives of the generation will be presented.

NGLs are, as previously stated, languages that had appeared and used within our timeline scope of 60 years, but their uses and purposes were not classifiable into 3GLs, 4GLs or MGLs. This meant these languages, while technically capable of performing functions like previous classifications, were not used specifically in the previously discussed branches. This is because a majority of these languages were and still are used in UI development, web services and development and in some cases application development. (Mendez, 2014, pp. 1-38)

HTML (Hyper Text Markup Language) was primarily used to make and display online websites and JavaScript or PHP (PHP: Hypertext Preprocessor) is added to make them interactive, usually embedded right into the HTML. (Berners-Lee, 1993, pp. 1-5) (Lerdorf, 2002) (Champeon, 2001)

CSS (Cascading Style Sheets) is often used for a more interactive user interface and design, often responsible for several webpages at once. (World Wide Web Consortium, 2010) XML (eXtensible Markup Language) and .NET allow data and files to be sent and properly interacted with through webpages. (World Wide Web Consortium, 1998, pp. 1-5) Visual Basic, derived from BASIC, can allow for rapid application development (RAD) of graphical user interface (GUI) applications, often having been used to make forms. (Microsoft, 2021) Java, while more focused on the development of applications, is also used to develop web applications, run and connect web and application servers and different types of databases. (Paquet & Mokhov, 2010, p. 18) Most of these languages are one way or another involved in the development and upgrading of web services or pages.

2.6.2 Progression of Use

Most of these languages are still used and popular today, often being used or made by companies who wish to advertise online or internet connected applications like games, web applications and social media.

Java, by the TIOBE index, has never fallen out of the top 5, with its lowest position being only recently this year in February 2023 at 4th place. While its popularity has slowly dwindled since 2001, where all code was at least a fourth consisting of Java, it is still keeping up with being used by many programmers. (TIOBE Index, n.d.)

Visual Basic originally was not widely used, having only fluctuated within the second half of the top 50. However, a sudden spike in use in February 2020 has given it popularity, with its current highest position being in May 2023 at 6th place. (TIOBE Index, n.d.)

JavaScript takes up the place behind it, despite having a chaotic track record in popularity and usage that's always spiking or dropping, has consistently managed to keep its place in the top 20, with its lowest position being in October 2014 at 12th place and highest in 6th place in February 2019. (TIOBE Index, n.d.)

PHP follows behind the aforementioned two, although the language no longer sits at its peak of popularity. Its highest peak was in November 2004 and highest popularity was in March 2010 at 3rd place until it experienced a steady decline in usage from that point. Its lowest rating was in June 2022, but its usage is still steadily dropping. (TIOBE Index, n.d.)

XML, HTML, CSS and .NET were unavailable in the TIOBE Index, but due to their focus in webpages, they are likely still widely popular and in demand.

2.6.3 Main Representative Features

Due to the nature of some of these languages, the syntax and functions of these languages can be relative similar, with a major difference in functionality and additional features like reactive variables.

HTML and PHP have relatively similar syntax and coding composition. HTML was released in 1993 by Tim-Berners Lee and PHP was created the same year but released in 1995 by a team led by Rasmus Lerdorf. (Lerdorf, 2002) HTML functions by first declaring the document type, then the HTML element, the meta information, the title for the html page and finally the HTML's body, which has the divisions and information that should be shown on the page. HTML is designed to display data with focus on how it looks. (Berners-Lee, 1993)

PHP works the same way, without even changing the declared document type.

The biggest difference comes in where the PHP part is placed and how. All the previously mentioned elements of a HTML page are separated and defined by start tag names (< >) and end tag names (</>), with what goes between those two tags being shown on the webpage. PHP's tag name functions differently, having started with (<?php), following content and ending with (?>), all in one element but there can be several PHP elements in one file. The way code functions within the tagline is also different. Keyword functions, classes and even variables can be placed inside it and are separated by a semicolon, which PHP has for dynamic page content, modification for entries within a connected database and controls user-access. In a way, HTML and PHP are often used hand in hand. (Berners-Lee, 1993) (Lerdorf, 2002)

Java and JavaScript are different cases. To anyone not familiar with their differences would assume they're used together but they're not. Java first arrived in 1995, developed by Sun Microsystems and designed by James Gosling. (Sun Developer Network, 1995) JavaScript was released later in the same year by a different team and developer – Brendan Eich of Netscape and later reverse engineered by Microsoft. (Netscape Communications Corporation, 1995)

Java is used for both application development and webservices and server control. It is platform-independent, object-oriented, interpreted and compiled and a general-purpose programming language. It is derived from C and C++, made to be case sensitive and able to run anywhere. To have something displayed on a console or webpage, it needs to have a class declared, where that class functions and only then what that class does. Java usually functions entirely in its own file, as it is also in charge of developing applications. (Gosling & McGilton, 1996)

JavaScript is much less robust than Java, prototype-based, interpreted only and most importantly a scripting language. It is mostly responsible for the code of many web-browsers. JavaScript has a bit shorter code process and is written in one line without paragraph indents. Since the code is directly connected to the HTML, it must first define where it works, then declare itself either as a GET or a SET function on an object, declare where the output will be and then the actual output that will be shown. The script itself is used and types in an HTML file, usually within an element like a button or paragraph. (Champeon, 2001)

CSS, unlike previous languages, is entirely focused on the aesthetics and design of the HTML/webpage, as it was developed in 1996 by W3C (World Wide Web Consortium). (World Wide Web Consortium, 2010) Its script is a bit similar to PHP but instead of taglines, CSS uses brackets to declare the traits of the element in question ({}) and each trait of a specific element is separated by a semicolon. Advanced CSS elements can include animations, transitions and text effects, but most are declarations of the traits of an element. Unlike PHP, CSS has no declarations for functions. (Mendez, 2014, pp. 109-118)

XML was developed in the same way as CSS and is also syntactically like HTML, however it does not have the same parameters and defined elements. XML was designed to carry data with a focus on what said data means. The taglines in XML are not predefined like in HTML, but it is extensible without many changes if any necessary, making it case sensitive. (Mendez, 2014, pp. 61-65)

.NET and Visual Basic belong to the same code family, in the way that Visual Basic is implemented on .NET Framework. The code used in Visual Basic is completely different than the ones used in HTML or Java but retains some similarities. Like C, Visual Basic uses paragraph indents and line changes to define the end or beginning of a function. Like Java, Visual Basic must first be defined in a Module, then the function/output must be written into a Sub-Module of the main class. Visual Basic was designed and developed by Microsoft, appearing only recently in 2001. (Dollard, 2018)

An educated guess can be made that NGLs had focus specifically in web development and web-design. Even Java, which could be technically considered an MGL as its multipurpose, because of its patent company, Oracle, having made several sub-applications specifically for making queries and forms and its most popular use was the creation of web-applications.

3 Demand and relevance of programming languages

3.1 Methods

In this chapter the methods and analysis of documentation will be discussed, demand and relevance of the programming languages and generations. This chapter will include the quantitative analysis of each language's documentation and demand in the job market and how that reflects on their generations.

In the current thesis status, the methods are a case study and statistical analysis of several indicators of popularity of the programming languages and their classifications. The statistical analysis will be made using gathered records and data on both programming language documentation and programming languages demand.

The point of gathering amounts and records of documentation is to accurately gauge the "supply" of information that programmers and researchers create and distribute. This will include documentation that either studies or uses the language in question. This can be a way of gauging the popularity and widespread knowledge of these languages.

Documentation will consist of both official publications as well as research publications. Official publications consist of official website FAQs, installation guides, system admin manuals, SDKs and APIs and content on the official site of the language. Research publications will consist of published books, articles, conferences, book chapters, reports and other thesis publications.

The documentation will later be grouped together by generation to visualize the difference in research and papers written about these languages. The groups will show which language generation was most studied over which years.

The point of gathering amounts and records of job offers is to accurately gauge the "demand" of professionals and necessity of said languages. This can be a way of gauging the use and implementation of these languages. However, records concerning job offers, programmer positions and solid numbers concerning employment rates that date prior to the 2000's are far and few in between. As such the growth not in employment but in the spheres of application and

industries where each generation specialized in will be analyzed. 3GLs having branches of scientific, government and financial focus, 4GLs being more corporate and education focused, NGLs being used in web development and design, MGL used in a mix of scientific, educational and computer programming.

Demand will consist of job offers that require or prefer specific languages in each year of our previously mentioned timeline. The demand records will later be grouped by generation to see the necessity trends in these generations.

Due to the sheer number of publications and lack of documentation of the online job market, our research, to be feasible, must be limited. As such, the publications will be limited down to the HAMK Finna International E-materials and ACM (Association for Computing Machinery) Digital Library. Since HAMK Finna will also include ACM in its search, the search query will be tweaked to not include those from ACM. Finna itself will include more than one database in its searches but as ACM is the oldest and most prevalent database on computer science publications, it will be outlined as some databases do not include/have publications before a certain point. The job market documentation will be made based on each language where possible. With the large amounts of years in the timeline, the timeline records of 1955 to 2021 will be limited by a section of six years, with a total of 11 sections.

All records of each generation will be saved and stored in each generation's individual Excel file, categorized by language sheets.

3.2 Documentation

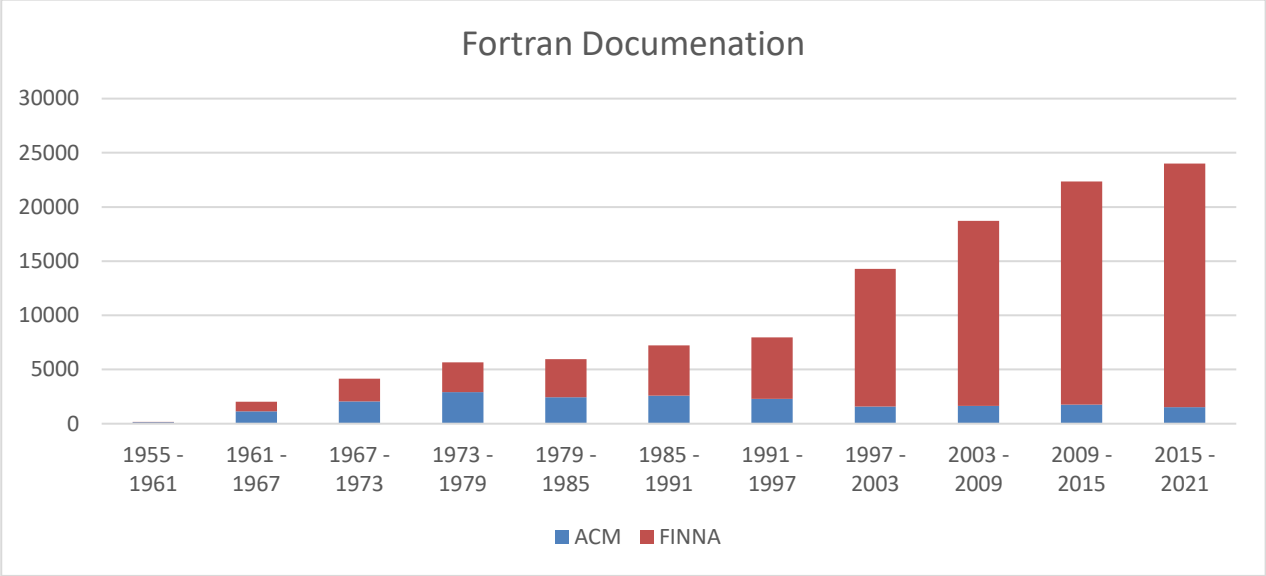
3.2.1 3GLs

This subchapter will show the finding and analysis of each generation's language documentation trends and research. Each part will look into the gathered statistics of each language and then compare all of them, making conclusions on how actively each language was researched and written about as both a sign of topicality and necessity. The charts and tables are within their respective Excel sheets, assigned as stacked columns for both ACM and FINNA values.

Starting with 3GLs, these languages have had documentation since 1955, the beginning of our timeline and since most of the languages in this generation came out of the timeframe between 1955 and 1961, despite some coming a bit later.

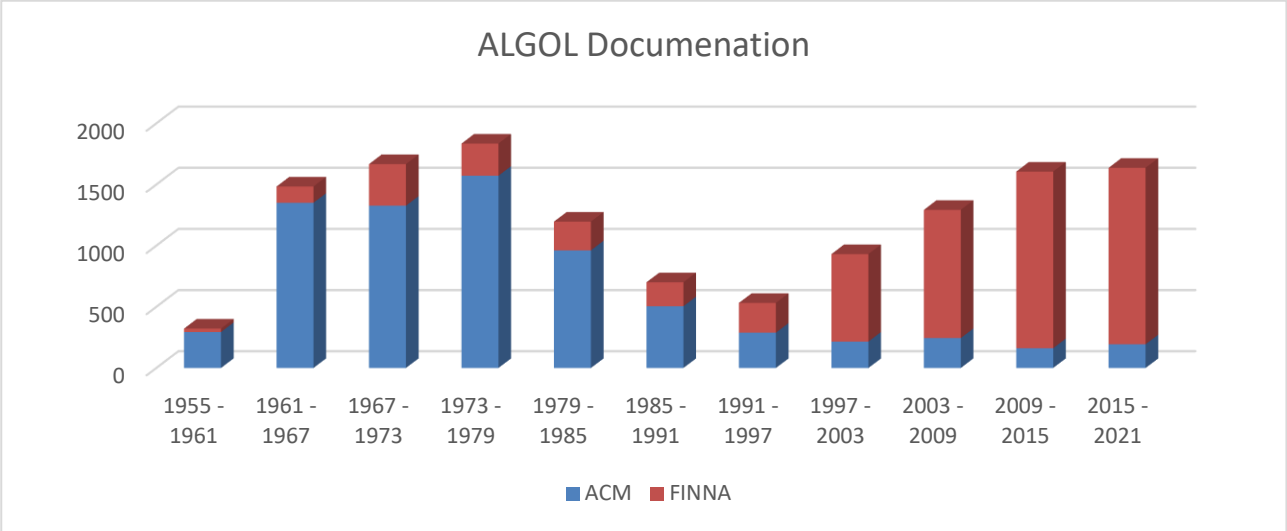
Starting with Fortran, in Figure 1, despite a slow rise in the first half, the language retains its necessity and implementation throughout the timeframe, having a sudden rush in publications between 1997 and 2003. Up until 1985, ACM and other databases published a more or less equal amount, but later as newer databases and journals were founded, ACM's publications and discussions moderately decreased.

Figure 1. Fortran Documentation



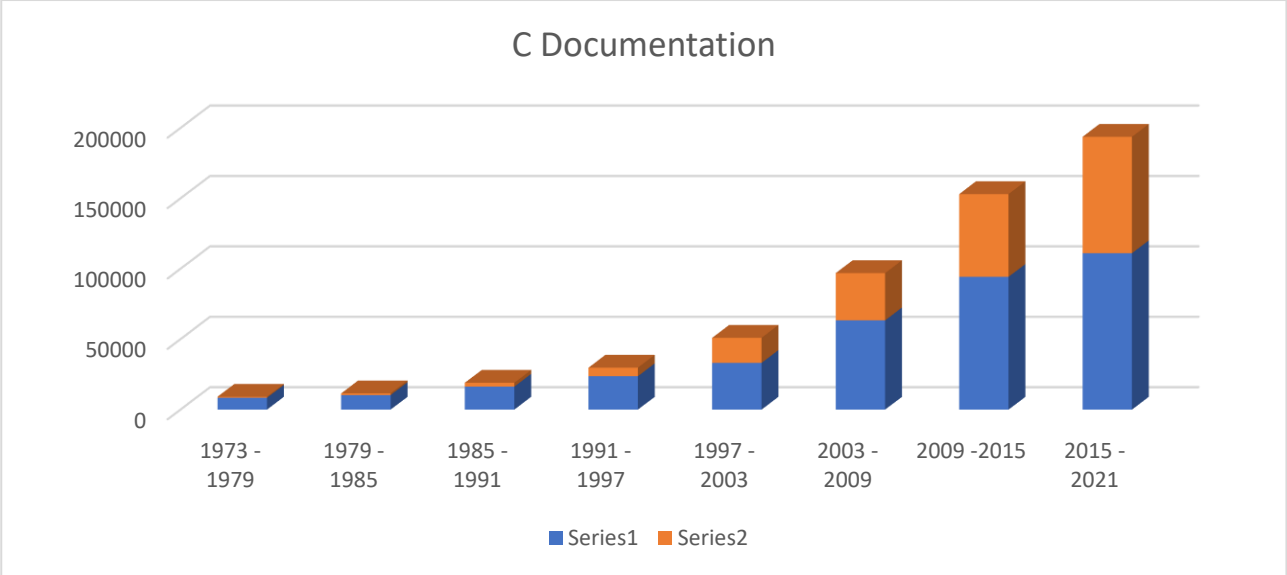
ALGOL, the second language of this generation, shows a wave-like pattern in publications and favored databases in Figure 2. As ALGOL was released in 1958, it only had about 3 years in the first bracket before following the next by skyrocketing in publication from ACM. Other databases had only begun to implement and write about it. ACM continued to pour out papers and publications about it in steady and high numbers until the beginning of the 1980s, when interest and uses quickly dropped. The loss in traction continued until the end of the 1990s and beginning of 2000s, when interest in ALGOL was renewed and more journals and databases started publishing en-masse, contrary to ACM which slowly reduced its output steadily. The resulting data gives us a show of rise in popularity, then a fall in use and a sudden comeback from other publishers and researchers.

Figure 2. ALGOL Documentation



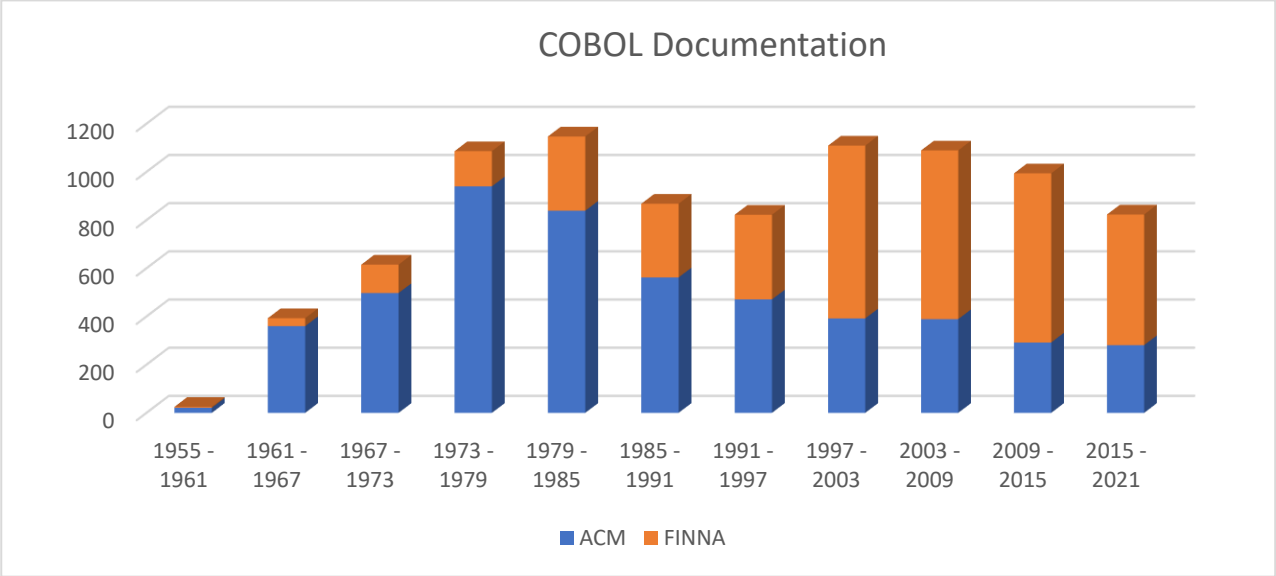
C, as the next language in the roster, is trickier to accurately analyze. This is due to difficulties that had come about in the research process with contradicting research results. Through this in Figure 3, C is still very popular and topical among both ACM and other sources, publishing out more or less the same amount of research by 2021. Originally, ACM had a long-standing monopoly but more journals and databases were funded and made. This made more sources pile into C, which to this day is still very relevant, both by research and popularity, as written in the theoretical part.

Figure 3. C Documentation



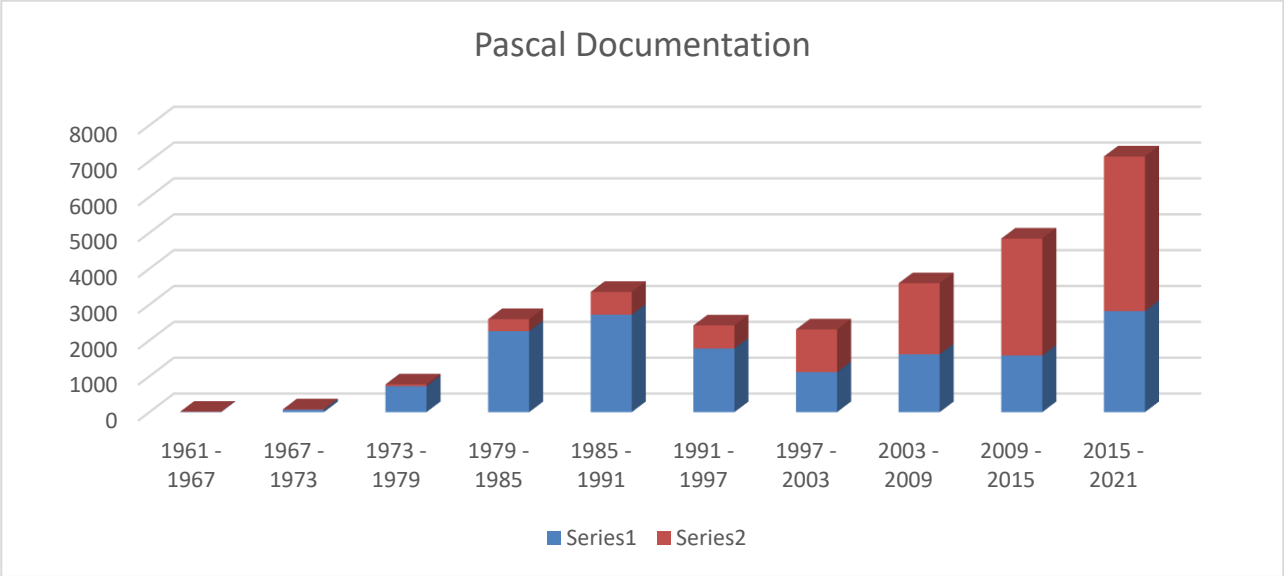
Fourth in our lineup is COBOL in Figure 4. The structure COBOL has in popularity is similar to ALGOL, but unlike ALGOL, it can be seen that COBOL is slowly decreasing in written works in its "second wave". COBOL had originally gotten strong traction in its first half from ACM, but the database slowly decreased its publishing from 1979-1985. However, the language itself rebounded and rose back into relevance thanks to other publishers in 1997-2003. Currently the research on COBOL has decreased but is steady.

Figure 4. COBOL Documentation



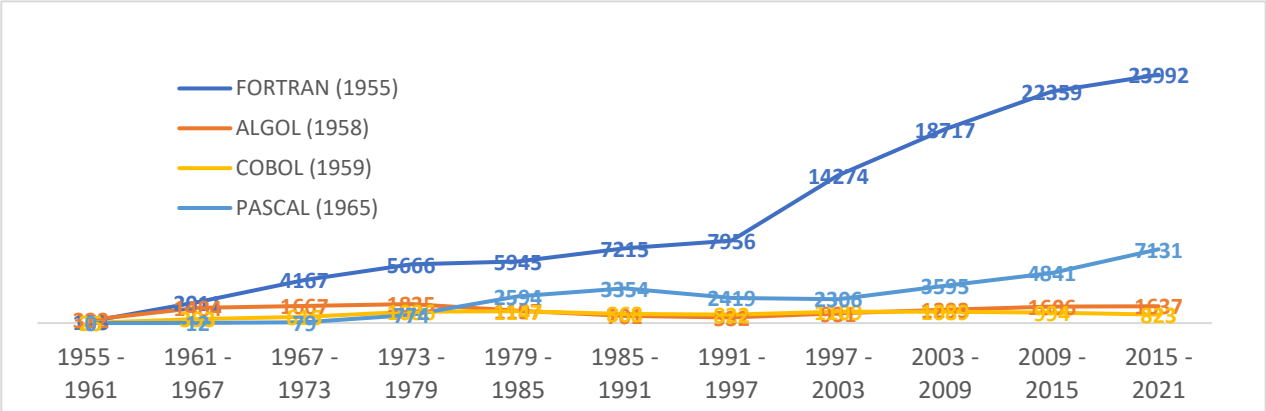
The last language analyzed will be Pascal. Pascal has a mix of the two previous versions of charts seen prior. Originally as seen in Figure 5, Pascal was not well researched for the first decade of its appearance but later was more thoroughly studied up until the beginning of the 1990s. The language experienced a slow decline before rising back to popularity, additionally beating its previous record, in 2003. Ever since then, Pascal has had rapid growth in interest and implementation, like other languages in its group due to newer publishers.

Figure 5. Pascal Documentation



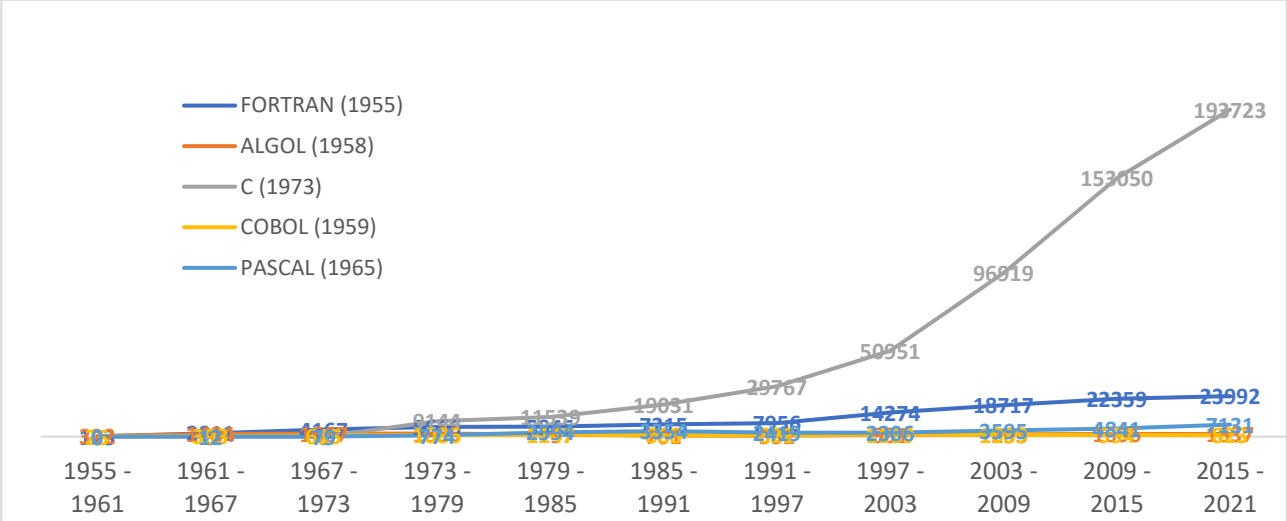
The results of all the languages will be analyzed and their growth to one another will be compared. As seen in Figure 6, that FORTRAN (dark blue) and Pascal (light blue) are still rising consistently.

Figure 6 3GLs Total Documentation



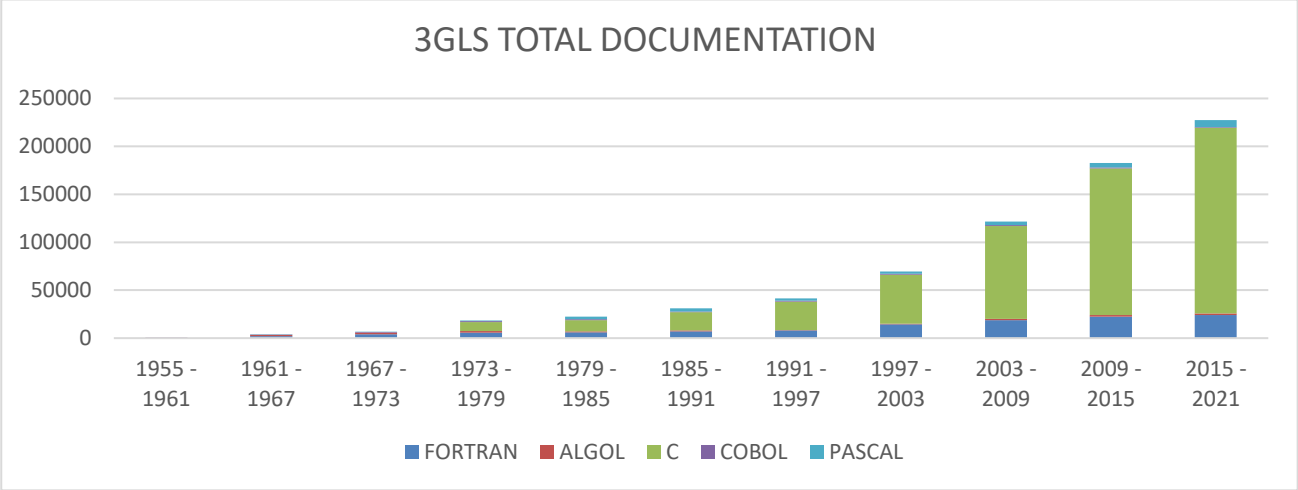
C (grey) is also rising in use, not only comparative in popularity, but in consistent growth of issued papers over periods of time. COBOL (dark blue) and ALGOL (orange) are consistently keeping within their previous trends of growth and research development. So far, C is visibly in the lead as a priority in research in 3GLs. Comparing the graph with C, Figure 6, and the one without C, Figure 7, it can be more accurately seen that Fortran is gaining more popularity.

Figure 7. 3GLs Documentation 2



The same can be said for the volume of research in Figure 8, with C having a majority.

Figure 8. 3GLs Documentation Volume



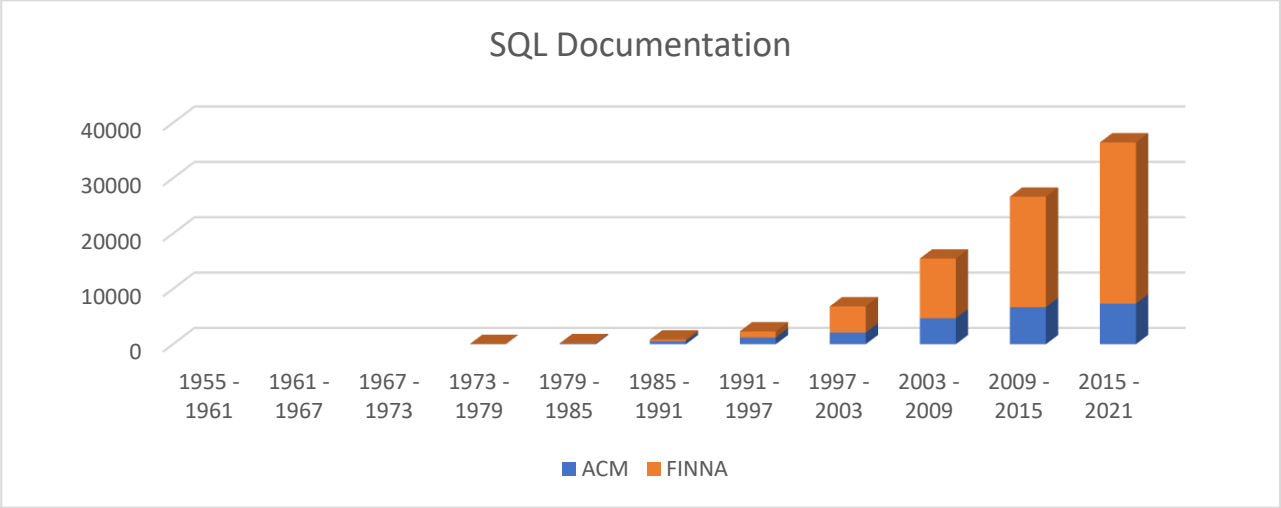
With this, the documentation of 3GLs has been analyzed and visualized.

3.2.2 4GLs

Moving onto 4GLs, these languages have sporadic documentation, as the only 3 languages represent and each debuts in a different decade. Nonetheless, it will include prior dates as a sign that they are technically of a different generation, despite some coming a bit later.

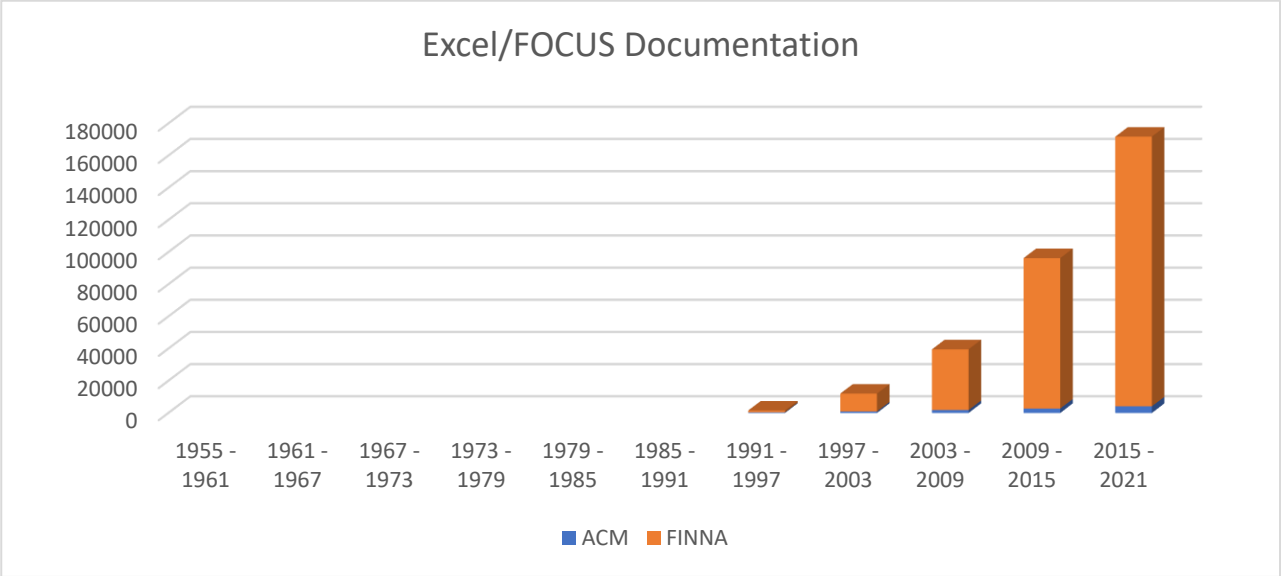
Starting with the oldest, SQL using Figure 9. SQL did not get traction in research and use until the beginning of the 1990s and only continued to rapidly grow in the 2000s, compared to the other two languages. Despite currently being at an all-time high, it is mostly with the use of other databases, libraries, and publishers. However, ACM is slowly beginning to write more about it as well.

Figure 9 SQL Documentation



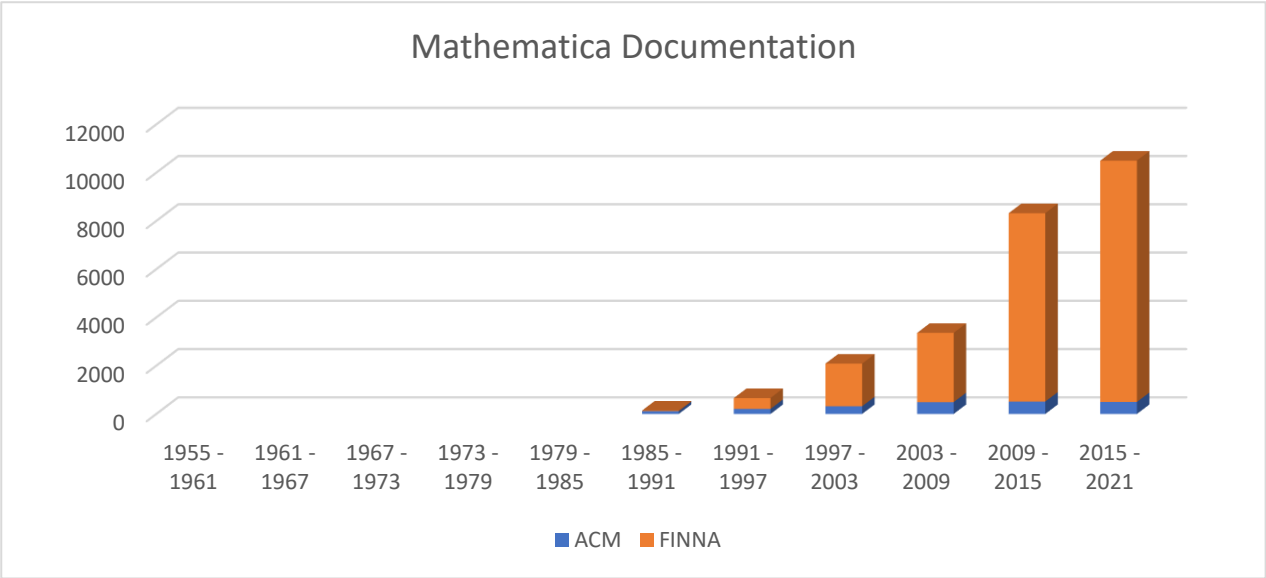
Next, FOCUS and, by its technical extent, Excel. Among its other representatives, Excel/FOCUS is the youngest of these but regardless is just as relevant. As seen in Figure 10, being released only in 1994, it did not get any relevant research done until the 2000s, when it was widely used among other Microsoft products and other dataprocessing softwares. The skew in data between ACM and other libraries is visible and many other databases have a majority of pubications.

Figure 10 FOCUS Documentation



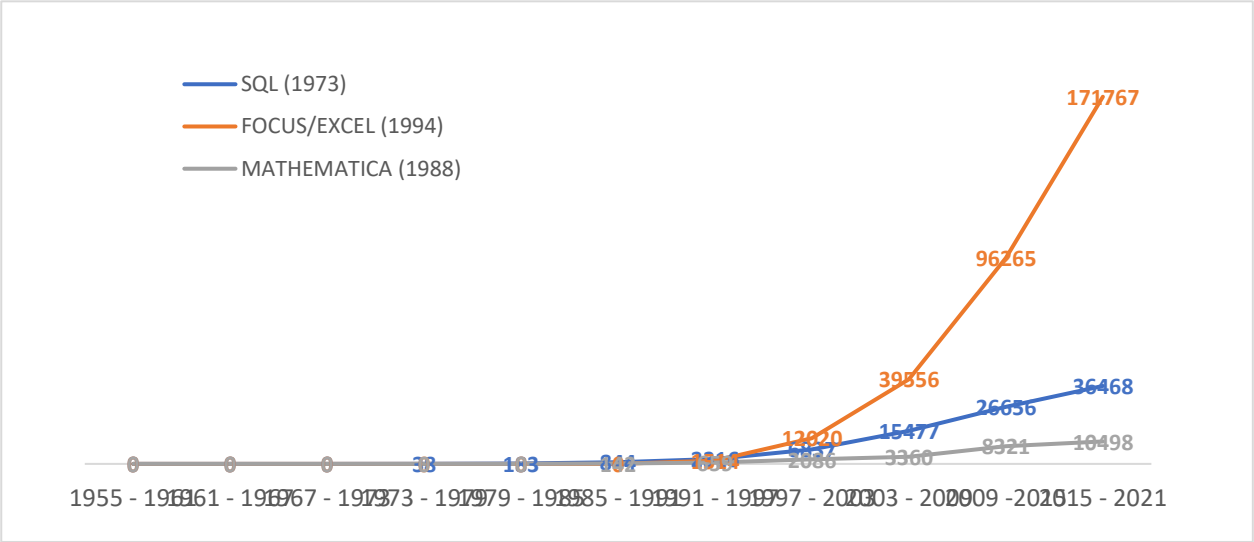
Wolfram Mathematica is the last of 4GLs which will be looked over using Figure 11. Similar to SQL and FOCUS, a lot of the documentation from it comes from a variety of publishers and databases compared to ACM, which shows very few mentions of the language. Mathematica is also shown to have started out small but experienced rapid growth in research and development in the early 2000s and to this day continues to be a topic of interest, as per the data of the chart.

Figure 11 Mathematica Documentation



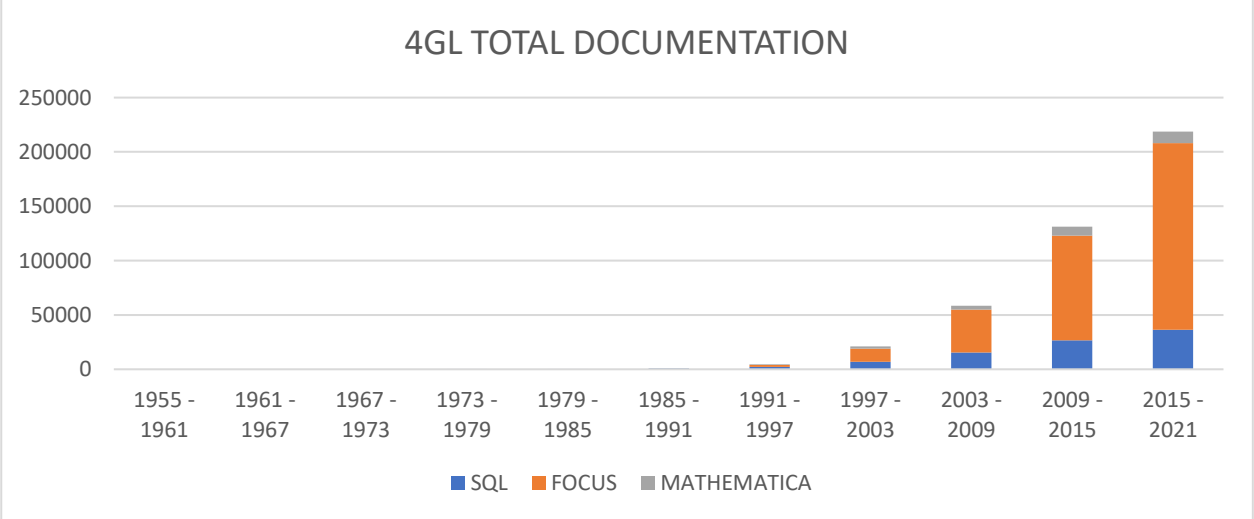
Finally, the comparison of the data of the three languages in Figure 12.

Figure 12 4GLs Total Documentation



Excel and Focus are the two that experienced the most rapid growth compared to prior years, very well rising more and more in each bracket. While SQL is also steadily rising, future prospects may show either spontaneous growth or stabilization. Mathematica has had a jump in relevance in the late 2000s and early 2010s and is slowly catching up.

Figure 13 4GLs Documentation Volume



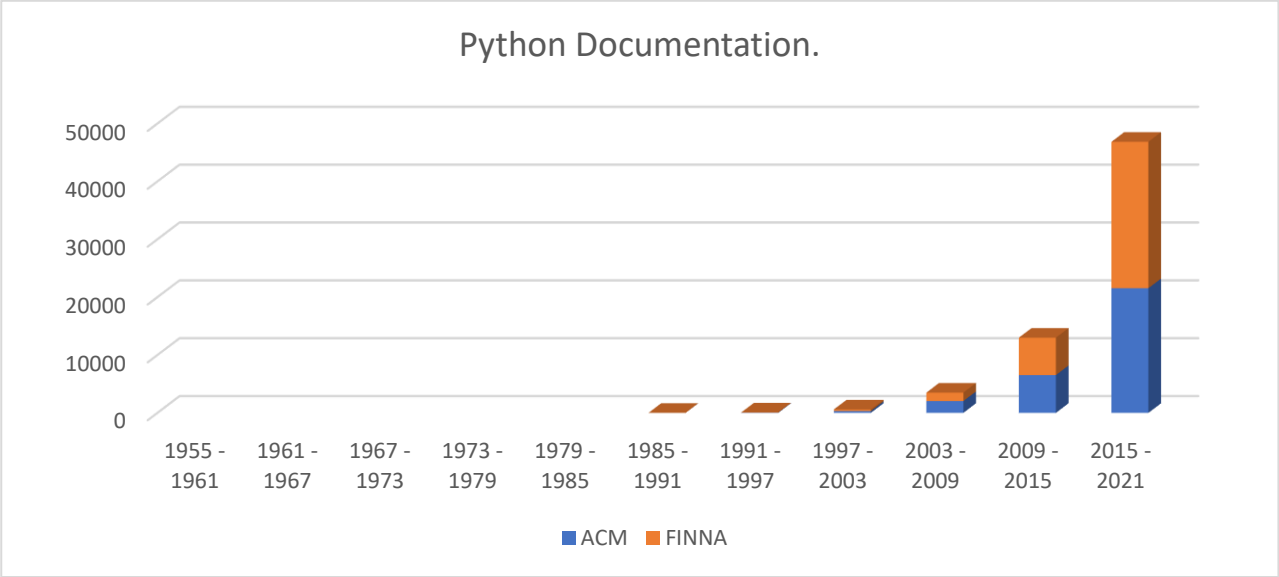
In Figure 13, it is visible that Focus and Excel have a mass advantage.

3.2.3 MGLs

Changing to MGLs, the classification that was described as a mixed bag. MGLs have a different issue that often came up in the data gathering of the documentation and the programming language R was the one whose data was most affected by it. Due to this, the languages included are Python, Perl, MATLAB and Ruby. The reason for this classification is because there are contradictions between which languages belong to which generation. As such in this group there are the languages that are consistent with both 3GL and 4GL traits.

Starting with Python, As seen in Figure 14, until the early 2000s, very little documentation and research was done. In the mid-2000s there was a sudden growth in interest. So far, the language has been researched thoroughly and widely. In addition to that the previous decade shows a rapid output of documentation research and publications. Python shows that the difference between ACM and other publishers is not that different, with ACM having less publications over time than new and more quantitative databases.

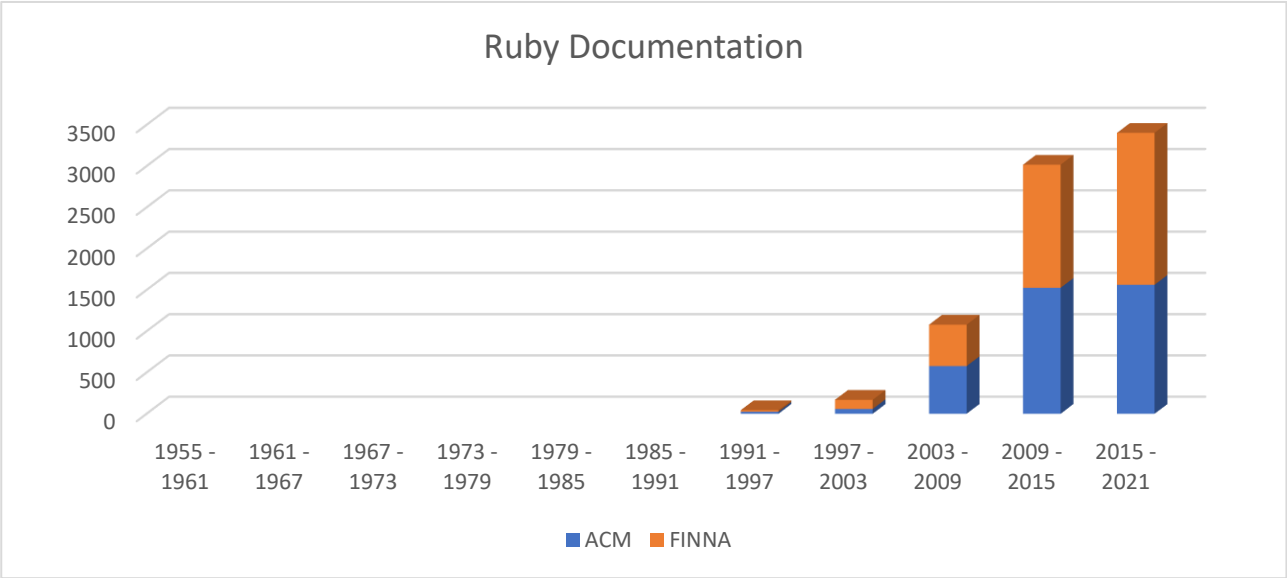
Figure 14 Python Documentation



The next language in question is Ruby. In Figure 15, the gathered data there was not much interest in the beginning. However, its popularity rose significantly in the beginning of the 2000s and continues to steadily publish a large amount of publications and papers. While in hindsight this is a relatively new language it shows that it is popular and that it is used and researched still to this

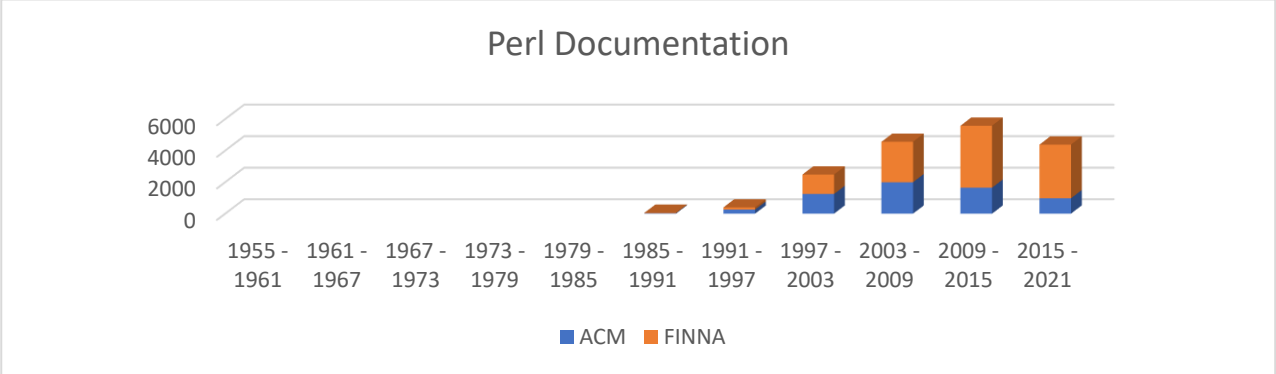
day. The discrepancy between ACM and Finna in documentation shows that they are somewhat evenly matched.

Figure 15 Ruby Documentation



Moving onto Perl. Despite being around for longer than its peers, it does not have the same data fluctuations as Python or Ruby. Rather than in the early 2000s, Perl started picking up popularity in the late 1990s as seen in Figure 16. This continued continued to grow until the mid 2000 tens and it is slowly losing progress in usage. Unlike its previous representatives the proportion of ACM to FINNA is not as equal, but also not to the point one has a total monopoly.

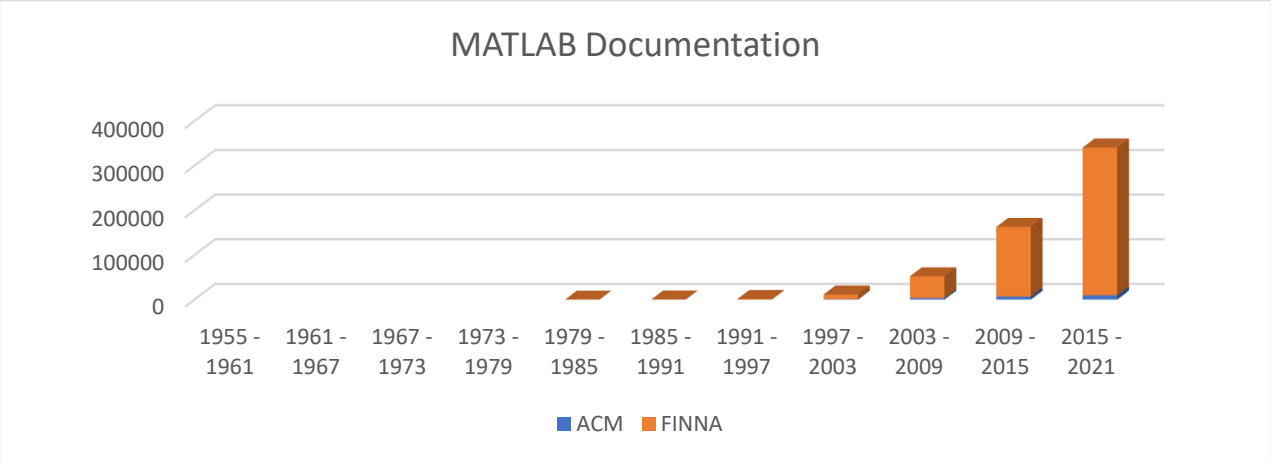
Figure 16 Perl Documentation



Lastly, MATLAB. MATLAB, despite its massive popularity in the early 2010s, in its starting year and the years following it did not have much popularity until the late 2000s, as seen in Figure 17, with

prior years publishing so little in comparison to its later stages of popularity. It can be seen by the chart that MATLAB's popularity and relevancy is still on the rise, albeit the publications come from many sources. ACM hadn't had any publications concerning the language until it started to rise in popularity.

Figure 17 MATLAB Documentation



Finally, a comparison between all four mentioned programming languages will be made. From the data seen on the graph, all four languages had different popularity statistics and are currently in different stages of their own development. Figure 18 has all the MGLs, and the difference in documentation is sizeable, with MATLAB taking the lead by a landslide, and it has been consistently growing in popularity since the late 2000s.

Figure 18 MGLs Total Documentation 1

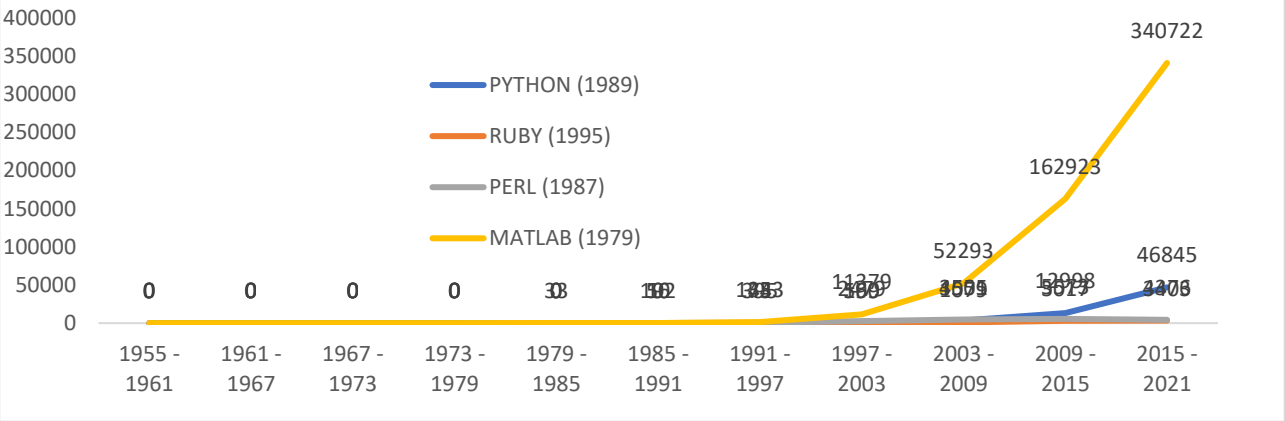
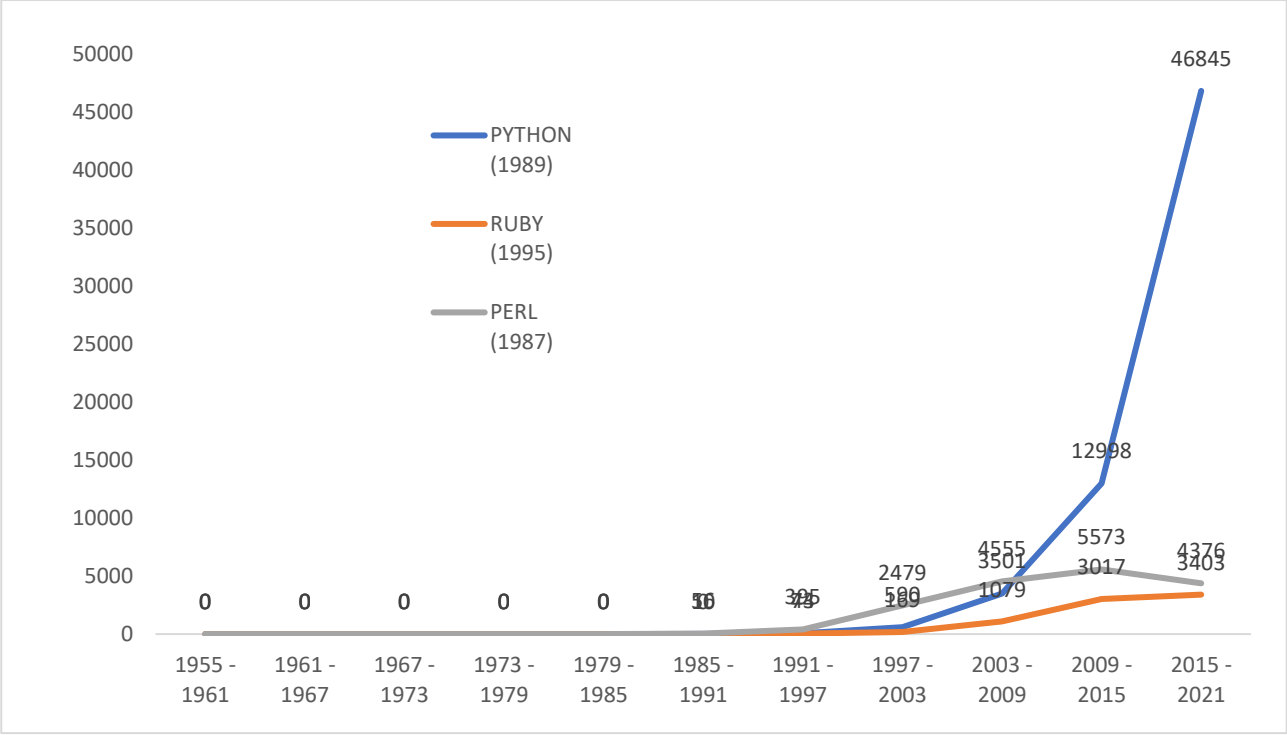


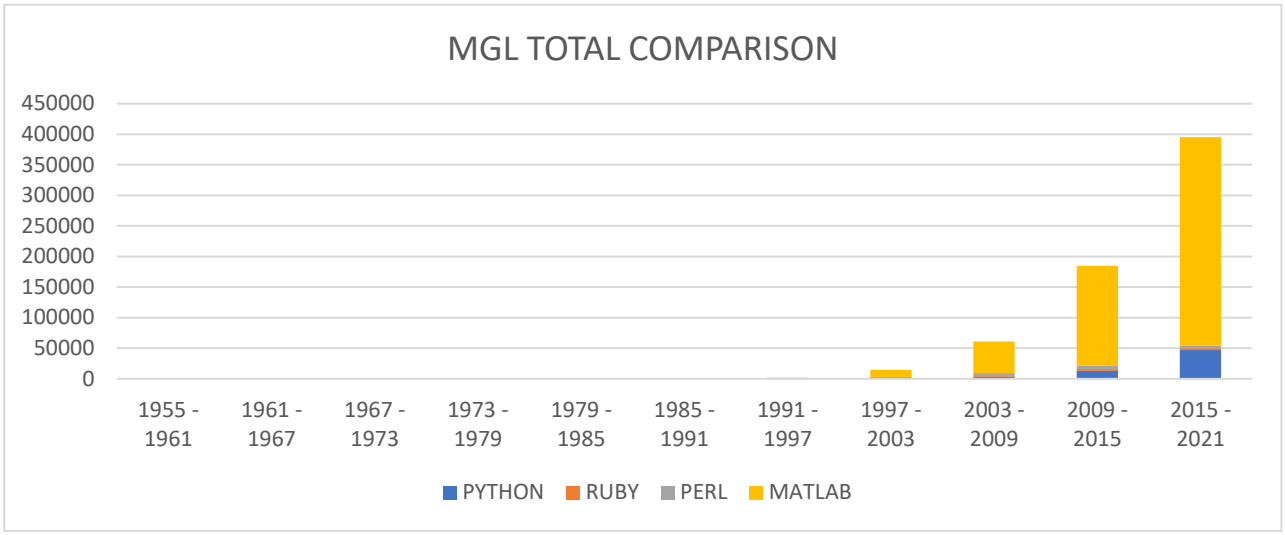
Figure 19 has left out MATLAB. This is due to a difficulty to accurately see the difference between Python, Perl, and Ruby. Figure 19 shows that, without MATLAB, Python takes the lead both in growth and numbers. Ruby so far is only recently getting traction, having only risen to similar levels that Perl had only recently in the past six years. Python, which officially was released ten years after MATLAB, and had a massive influx of documentation in the same time period, has a bigger advantage than Ruby and Perl. While Ruby is mostly stabilizing, Perl is slowly losing relevance.

Figure 19 MGLs Total Documentation 2



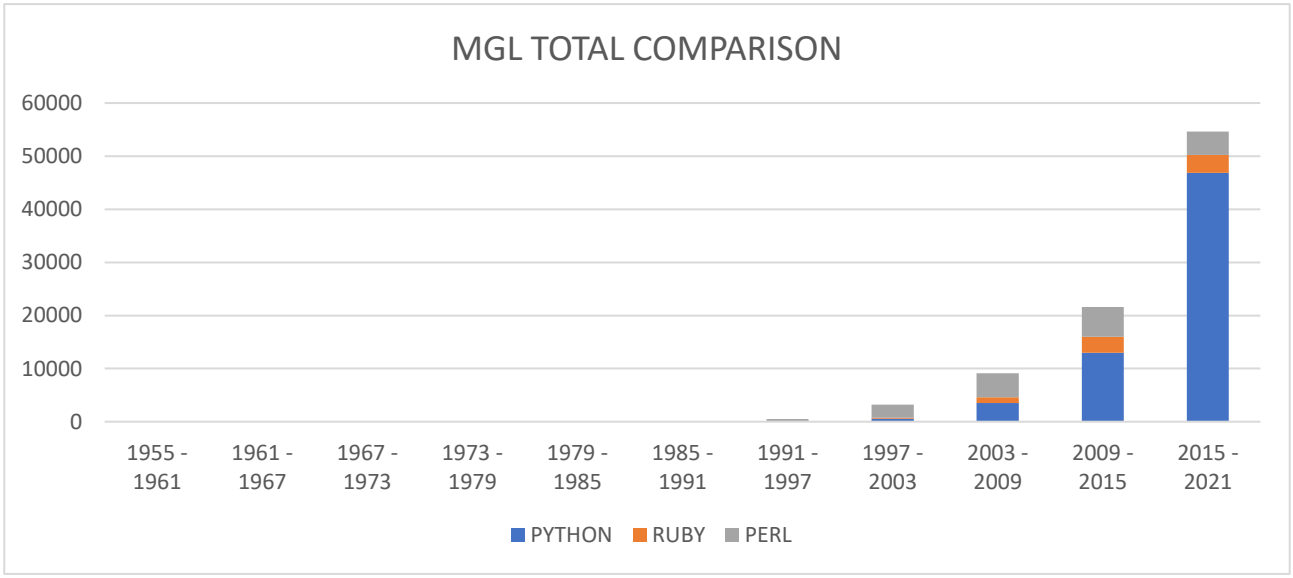
In Figure 20, the volume of research and documentation is far more visible. MATLAB takes up a staggering majority of publications in each section, with Python being visible only starting in the early 2000s. Compared to the two, Ruby and Perl are barely visible with comparatively small numbers. MATLAB has an advantage in both being released earlier than all of the previous languages by a whole decade.

Figure 20 MGLs Documentation Volume 1



Whereas without MATLAB, in the late 1990s, Perl had more relevancy than both Ruby and Python. While it later lost relevancy in the late 2010s, Python had skyrocketed in popularity, relevancy and documentation.

Figure 21 MGLs Documentation Volume 2



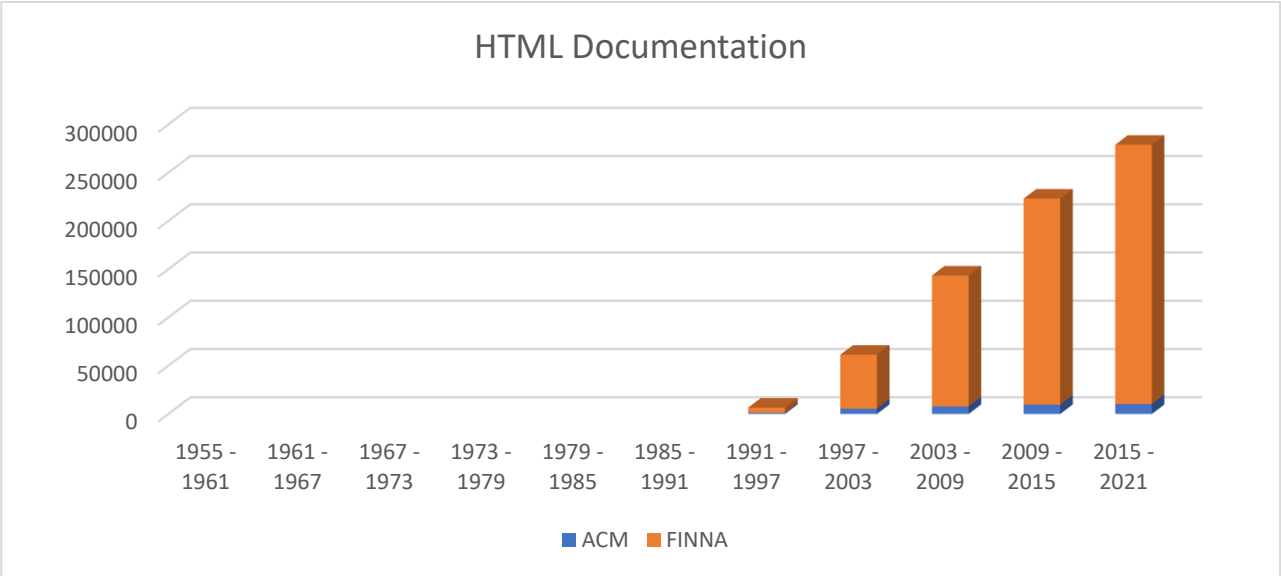
The data of documentation of MGLs has been analyzed, visualized, and presented.

3.2.4 NGLs

Lastly, the NGLs will be investigated. Many of these representatives will likely have similar documentation or similar popularity patterns. This is because most of their popularity came from the necessity of having programming languages for the development of the World Wide Web. The reason why many of these languages are not included in either MGLs, 3GLs or 4GL is because, despite having come up in use within the same timeline, they are not applicable to the other traits of the other three generations. This makes it difficult to classify them in each one.

Using Figure 22, it can be seen on the chart that the publications concerning HTML are steadily rising in numbers and in its research. There has been significant in the mid-1990s, which later led to significant and steady rise with each new section having broken through the last one into the next bracket milestone. While ACM has not been publishing as many as other publishers, the publications that have been outside of ACM are significantly more contributing to the research of HTML. Like Mathematica, it is currently on the rise in research and in publications. This is likely due to the significant Relevance that HTML has on web development the World Wide Web and the creation of websites.

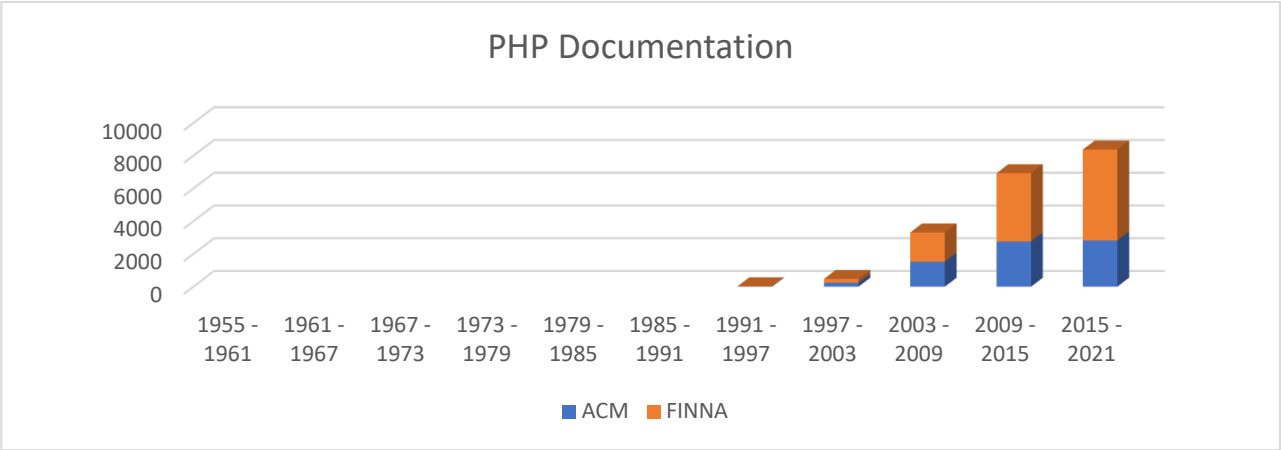
Figure 22 HTML Documentation



PHP will now be investigated. As seen in Figure 23, while the amount of publications is not as high as HTML it is still under significant rise within the same time period. Contrary to HTML it can be

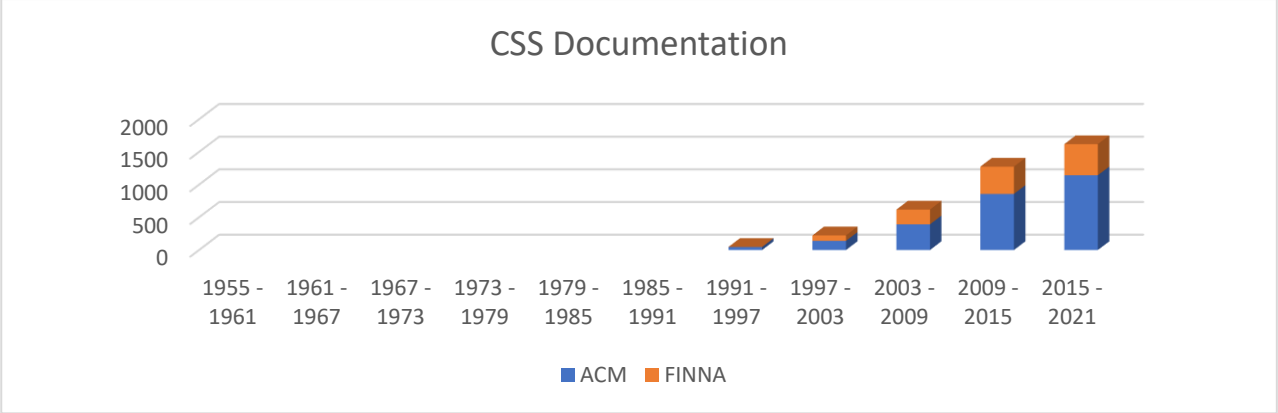
seen that HP has more publications from ACM and there is a significant amount of other publishers who are also publishing about it.

Figure 23 PHP Documentation



Moving on to CSS with Figure 24. Like PHP, it does not have as many documentations, but it is still being relevantly researched. One of the main differences is that CSS has more documentation from ACM than from other publishers to the point where it has 2/3 exactly from ACM. While other publishers are slowly increasing the number of research papers articles and publications concerning CSS, so has ACM. Between the mid-2000s and the mid-2010s the number of publications that they had added to their databases is double than what it was previously.

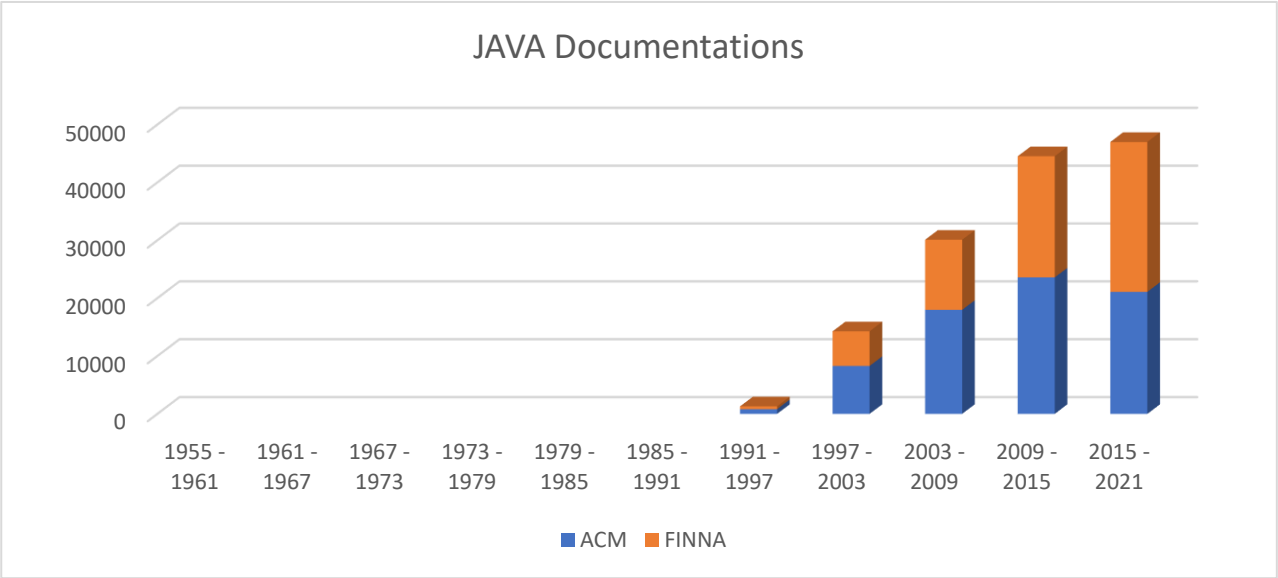
Figure 24 CSS Documentation



Moving on to Java, in Figure 25, the trend of publications seems to have stabilized. Over the sections that can be seen, publications from ACM and other publishers are more or less evenly

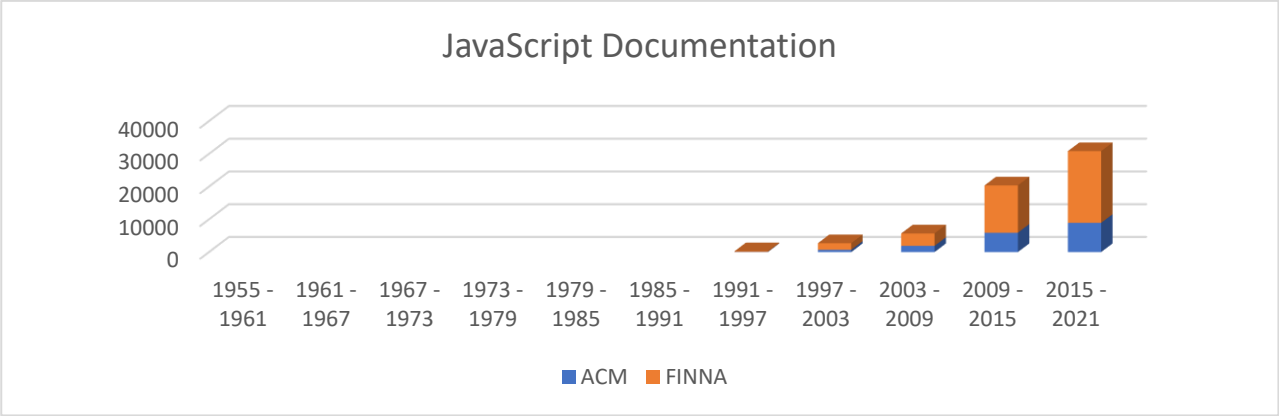
matched with some years having more and other years having less. The significant growth seems to be slowly stabilizing but the numbers are still significantly large in documentations.

Figure 25 JAVA Documentation



Following the documentation of data is JavaScript. Similarly, to the other representatives that were mentioned it is currently on the rise, as seen in Figure 26. However, those made by ACM, while relevant, do not hold as much amount as other publications. The numbers are like what Java had but approximately 1/10 of what Java has.

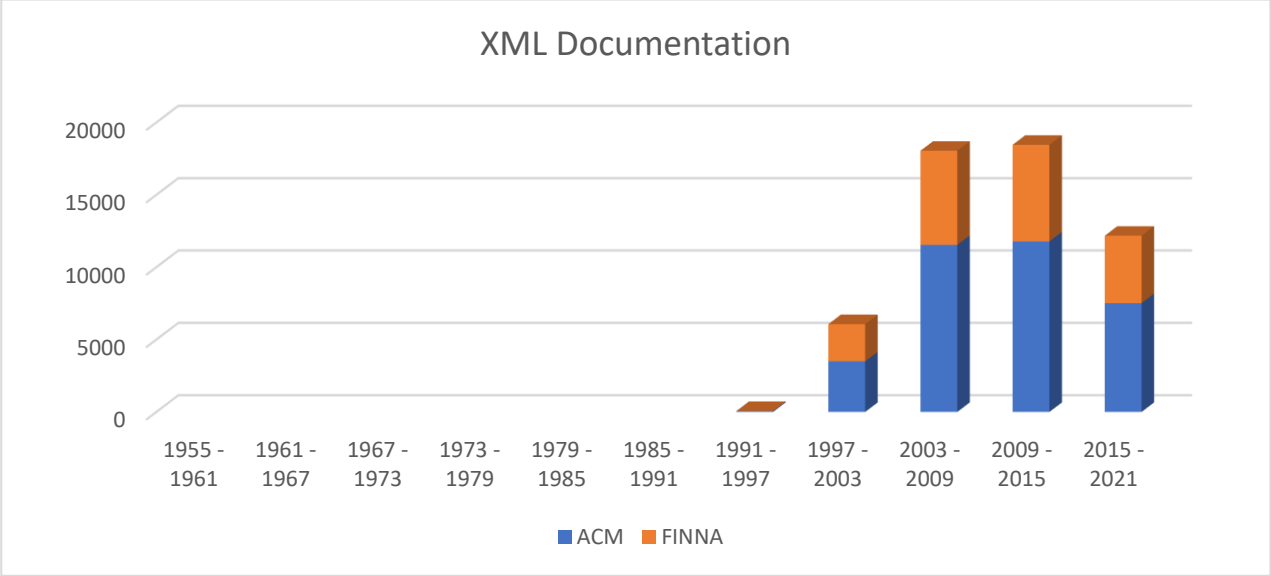
Figure 26 JavaScript Documentation



Studies and research about XML do not hold the same trend as the rest of the languages in this group as seen in Figure 27. It is feasible that in the past decade XML has slowly been losing

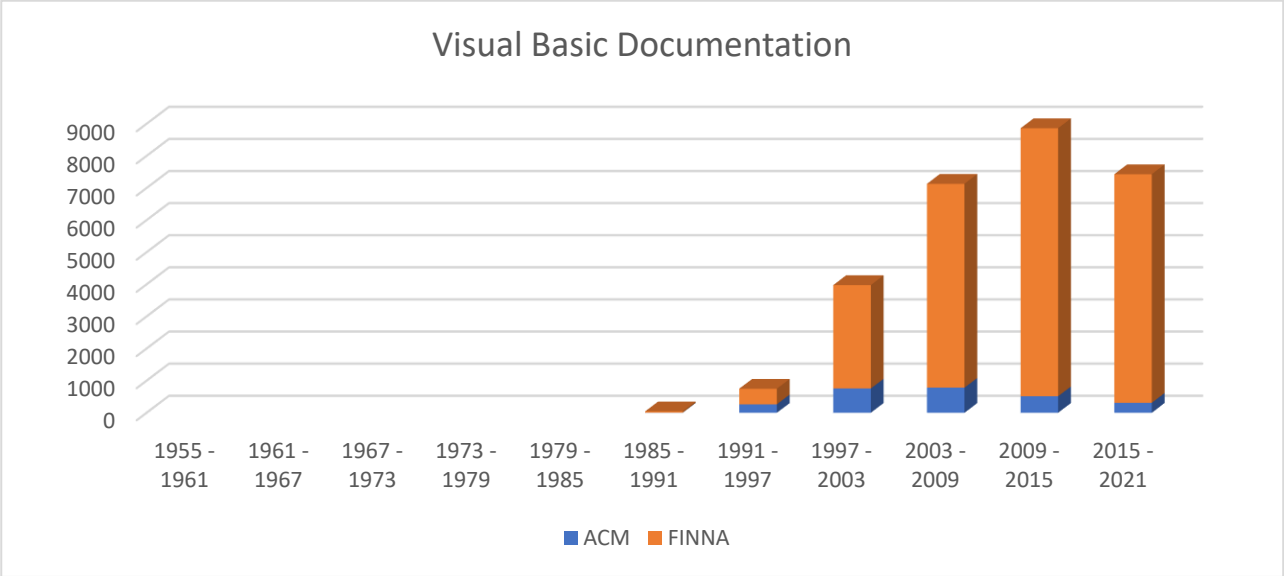
traction and documentation and popularity. Having its peak in the previous first half of the decade, it is now slowly losing traction. However, ACM is still publishing a significant amount throughout the years.

Figure 27 XML Documentation



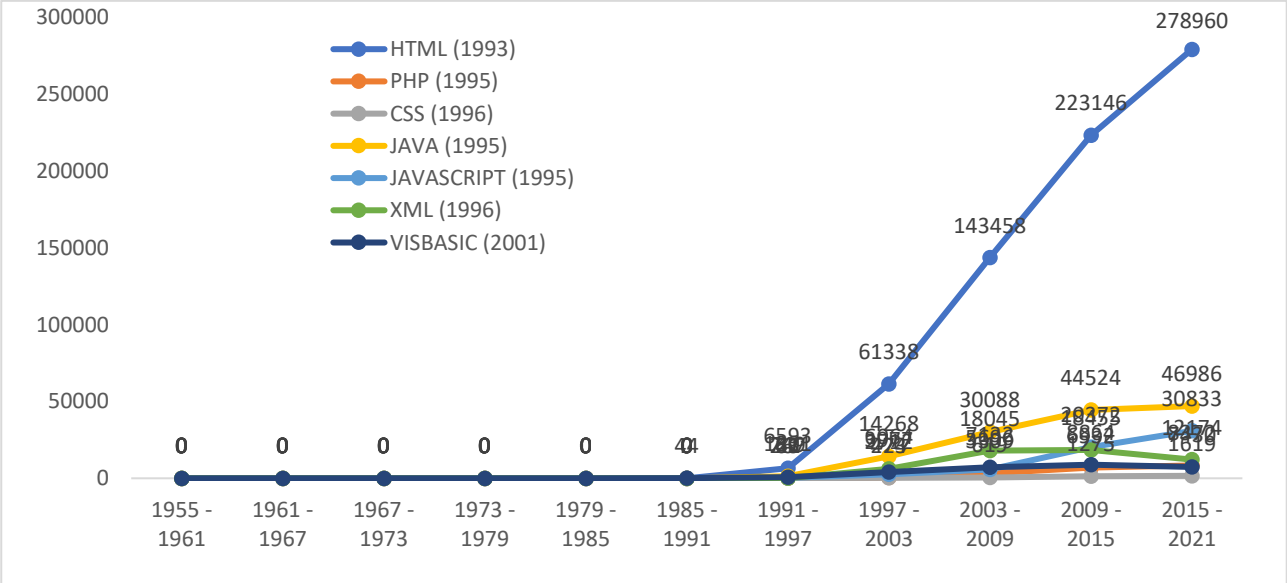
Finally, the last language in this part of research is Visual Basic. Visual Basic is a product of Microsoft, and undoubtedly has more documentation and research outside of ACM, as visible in Figure 28. However, even the interest that ACM had, Visual Basic originally peaked in the early 2000s. With other publishers it peaked in the mid-2010s. Now, the popularity and the irrelevance of it is slowly dropping but it is still relevantly used. The mount of publications is significantly less than the rest of the languages in this group. It is also worth noting that because visual basic came out only in the beginning of this millennium, the publications concerning it prior to its official release are not as prominent or known of.

Figure 28 Visual Basic Documentation



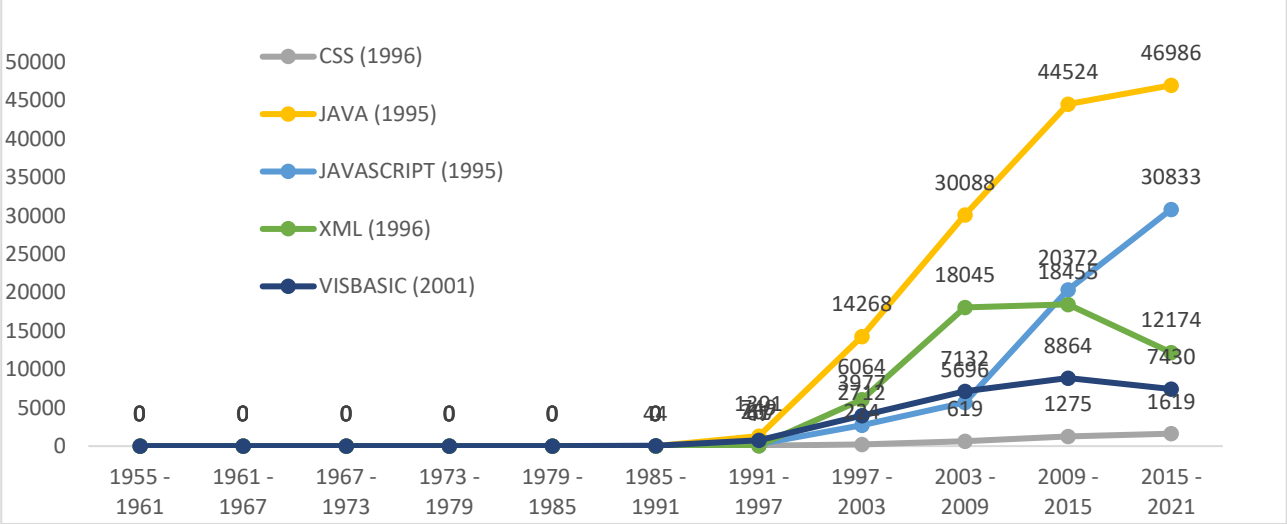
With all the languages now properly analyzed, the comparative statistics of the NGLs begins. As can be seen in Figure 29, HTML is significantly more popular than any other language by an increasingly large amount. Other languages like Java and JavaScript are slowly on the rise whereas others like PHP, Visual Basic, CSS and XML are slowly losing traction or the numbers that they are pushing out are incomparable to HTML. This is likely due to HTML being the backbone of many web services, many web sites and many online providers.

Figure 29 NGLs Total Documentation 1



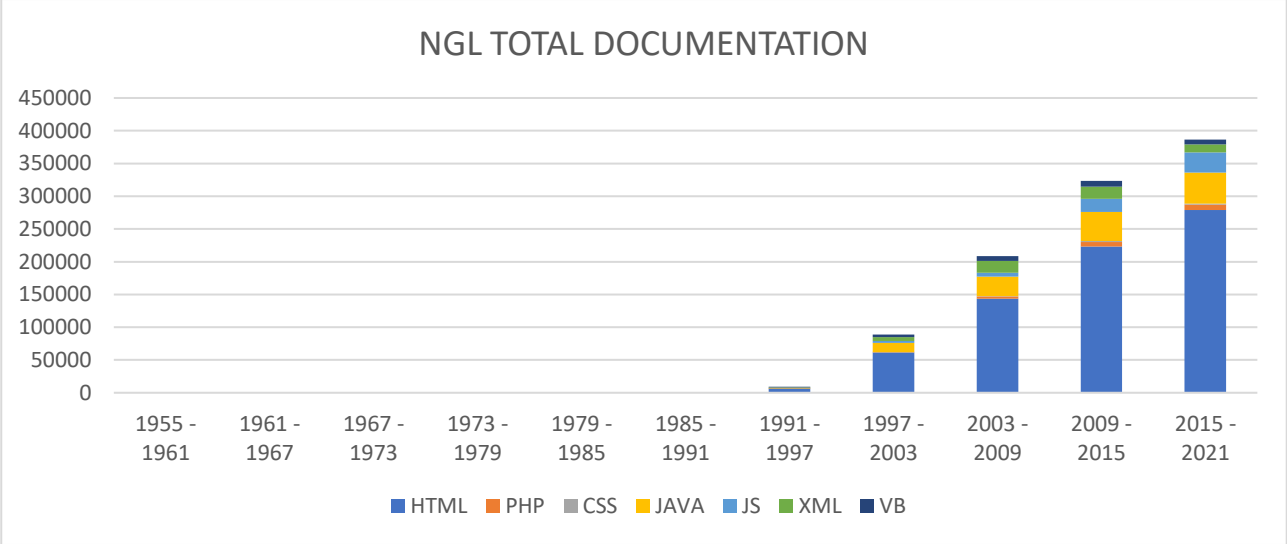
However, if HTML was removed and the languages are simply compared like in Figure 30, the comparisons can be seen much clearer.

Figure 30 NGLs Total Documentation 2



The volume, in Figure 31, in comparison of each language shows that HTML has the most value in regards 2 documentation amount and has historically held that position throughout the timeline, while other languages also have prevalence, it is not as much as HTML.

Figure 31 NGLs Documentation Volume

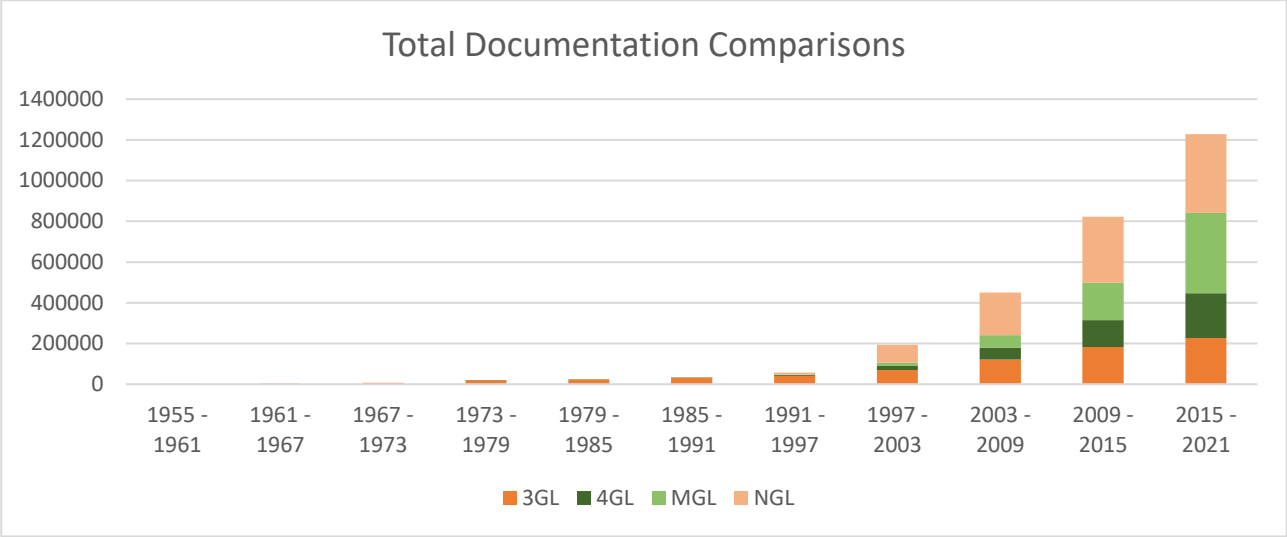


With this, the analysis and documentation of NGLs are finished.

3.2.5 Overall Comparisons

Now, each generation will be compared to each other. By combining the total number of all publications in each generation, the generations can be compared in a broader spectrum in Figure 32.

Figure 32 Generation Comparison



It can be seen that due to the sheer size of publications, research documentation did not enter 6 digit numbers until the late 1990s, primarily thanks to 3GLs and NGLs. Later, NGLs and MGLs started having more and more research allocated to them, compared to 4GLs and 3GLs, despite said generations having increased popularity as well. From a glance, NGLs and MGLs both attrivute for approximately 30% of all documentation each, while 4GLs and 3GLs only represent 20% each.

3.3 Demand

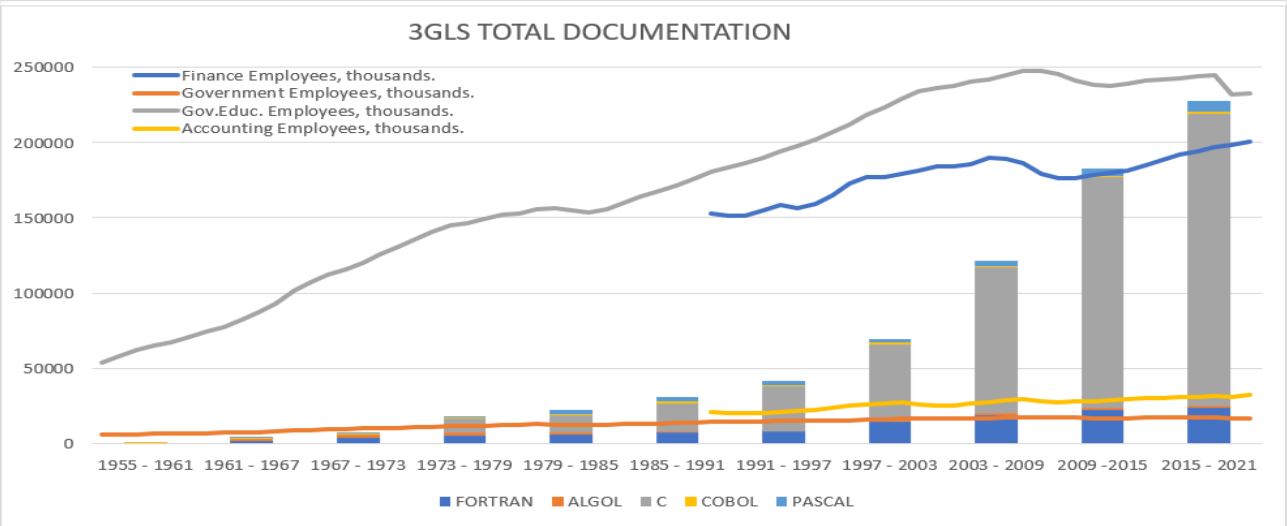
Now that a basis of analysis in research in documentation of each generation is provided, the conclusions to the statistics and data gained concerning industries of each generation can be compared. While each industry and job market cannot be spoken for, using Archival Economic Data from St. Louis Fed's Economic Research Division (ALFRED), vintage data of employment rates in 10 industries is provided, where generations with language focus that fits the requirements of said industries can be adequately and productively used. They have been classified them as such:
3GLs data will be compared in Accounting, Government, Government Education and Finances.
4GLs data will be compared in Information, Private Education and Business Services.
MGLs data will be compared in Computer Systems and Architectural Engineering.
NGLs data will be compared in Computer Infrastructure and Computer Systems.

The employment data to each generations' total documentation will be compared. The numbers of the employment data are saved as thousands.

3.3.1 3GLs

Here are the industries that 3GLs could be useful for and the annual employee statistics and the documentation used as comparison, as visualized in Figure 33.

Figure 33 3GLs Industry Comparison

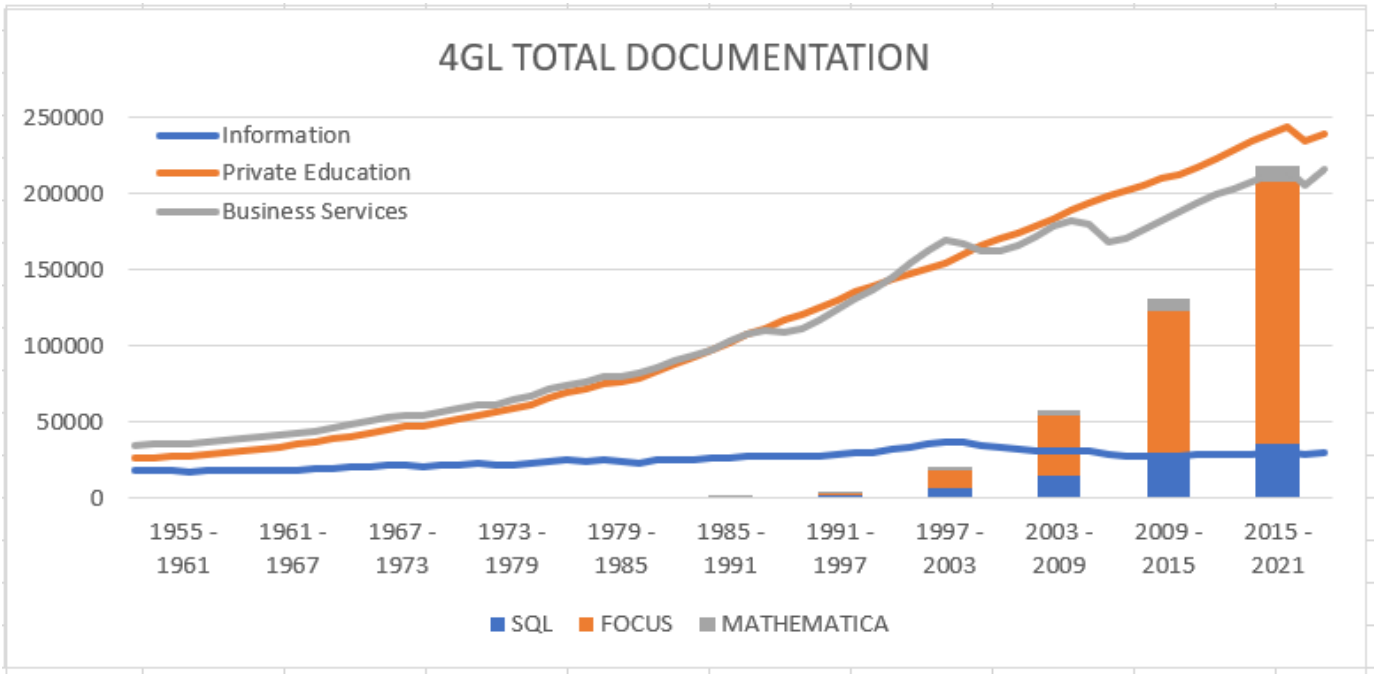


Finance (Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics, 1990) and Accounting (Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics, 1990) employees have records going back only to 1990, whereas Government (Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics, 1939) and Gov.Education (Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics, 1955) go back even further. While Government and Accounting data does not show any remarkable patterns. Finance and Gov.Education show the same trends in rising employees as 3GLs documentation. For Gov.Education, similarly to 3GLs, they experience a rapid rise in the later half of the 1980s, continue to sharply grow until 2007 and 2008 when the economic crisis happened, laying many employees off, only to later stabilize and start rising again.

3.3.2 4GLs

Here are the industries that 4GLs could be useful for and the annual employee statistics and the documentation used as comparison, as seen in Figure 34.

Figure 34 4GLs Industry Comparison



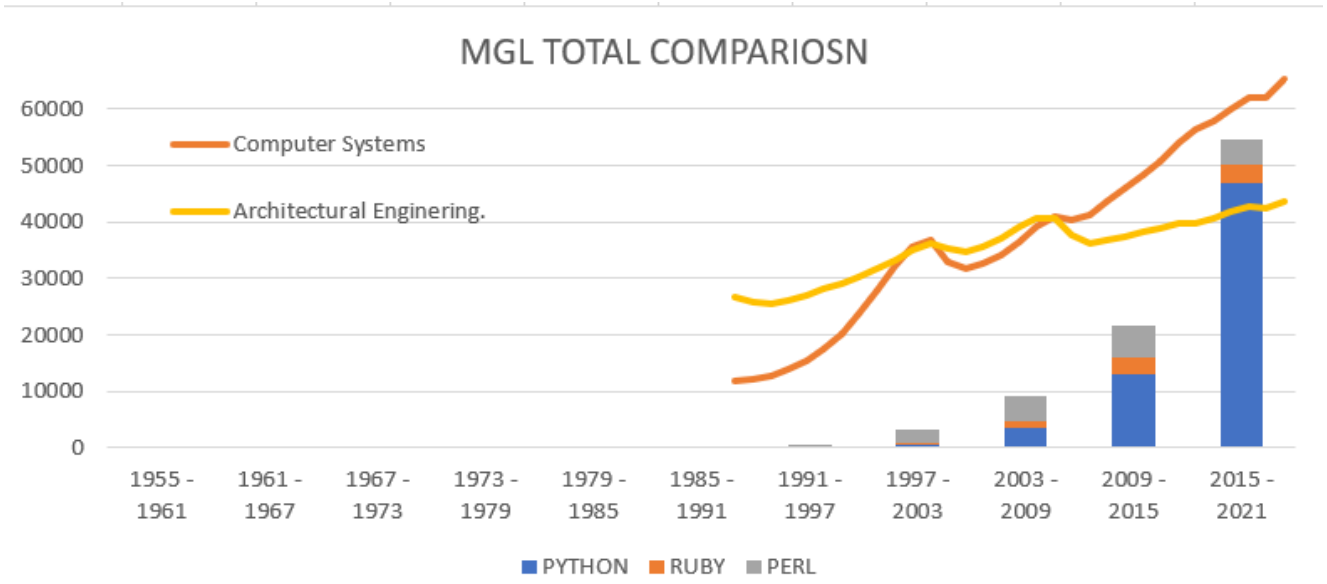
The Information (Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics, 1939) sector, while not as robust as Private Education or Business Services, did experience a rise in employees in the same time as 4GLs experienced a breakthrough in documentation. Private Education (Federal

Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics, 1939) remained mostly unaffected, as it was prior still rising. Business Services (Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics, 1939) followed the trend that 4GLs had, and quite similarly. When documentation started appearing in early 1990s, employment rose significantly until the late 1990's and early 2000s, fluctuated due to a likely shift from SQL to Excel, dropped in the economic crisis of 2008 but later stabilized and rose again and continued to rise up to 2020, fell and started rising again.

3.3.3 MGLs

Here are the industries that MGLs could be useful for and the annual employee statistics and the documentation used as comparison, as seen in Figure 35.

Figure 35 MGLs Industry Comparison

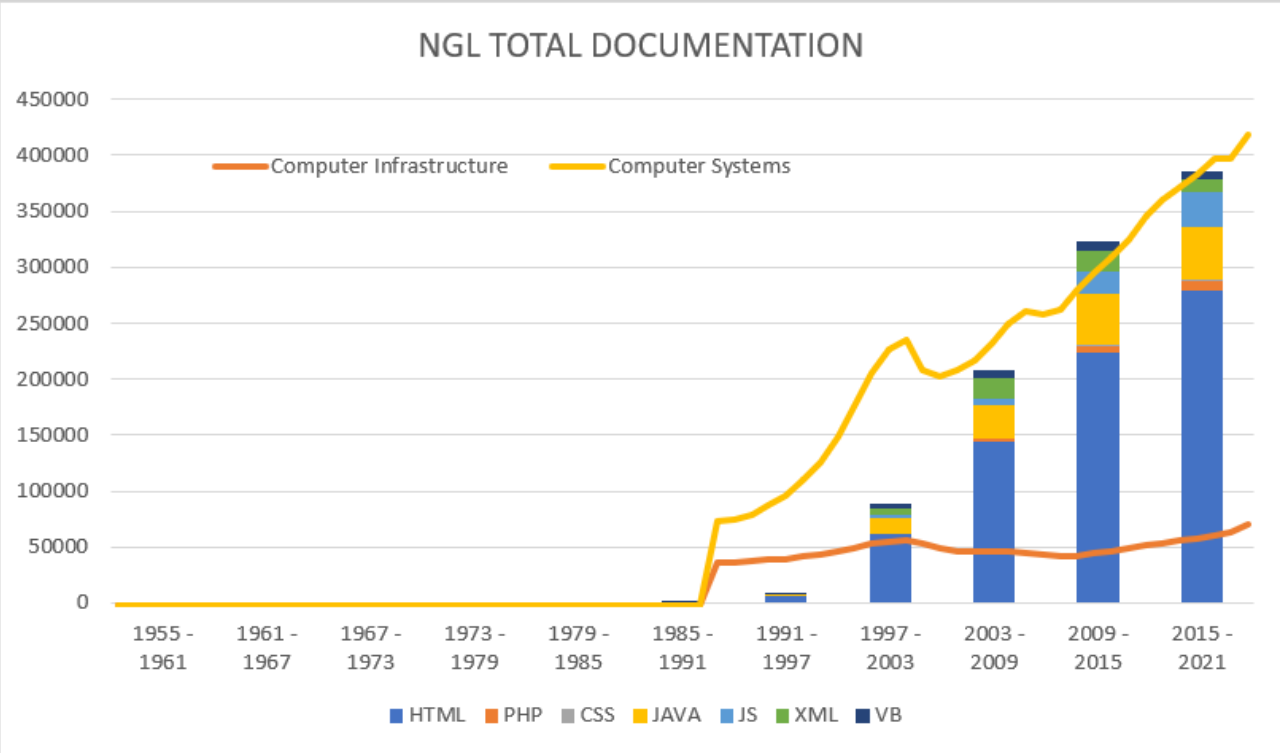


Computer Systems (Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics, 1990) employment rose almost identically in point with MGLs documentation progression and very actively to boot. Architectural Engineering (Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics, 1990) experienced the same fluctuations regarding rise in employment, however, after 2008 and leading up to the 2010s, did not skyrocket as much as Computer Systems.

3.3.4 NGLs

Here are the industries that MGLs could be useful for and the annual employee statistics and the documentation used as comparison, as seen in Figure 36.

Figure 36 NGLs Industry Comparison



Computer Systems (Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics, 1990), like MGLs, follow the growth of NGLs research, even more accurately than MGLs. Computer Infrastructure (Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics, 1990) shows a trend of having rapidly grown when NGLs first came out and employment in those fields has been on a stable rise without massive changes, even with the 2008 crisis. While there was a downturn in the late 2000s to the early 2010s, it shows to be recovering and more people in this sector are being hired again.

4 Results

With the analysis of the documentation data and comparison to the employment data, results concerning the necessity and the uses of the programming languages can be made.

From what can be seen, rather than one specific programming language having utmost necessity and popularity, it is more so that each category has a language which stands out from the rest. Obviously from the industries that the languages could be used in, different languages and different environments have different purposes. Even in those cases, behind those languages are second runners-up who are also widely used in the industries that they are implemented in. As such, there will be one or two specific languages which are a cut above the rest. The languages in question are often a multi-tool type of language that can function within its industry and outside of its industry.

In 3GLs, those languages are C and Fortran, where one is still very relevant and the other is gaining traction. In 4GLs, and those languages are Focus and SQL, both of which are widely used in the areas that they accelerate as well as additional features that help in either managing organizing or visualizing the data that they use.

In MGLs and NGLs it is harder to differ due to the nature of both groups - one is a group that is primarily able to function in several if not many industries and for the multitude of uses and the other is essential for the upkeep, growth, and functionality. Because of that some languages may be more prioritized than others. But in the current case, the most relevant languages in MGLs are MATLAB and Python. For NGLs, those are HTML and Java. But even those languages are not necessarily able to stay like that.

It can be seen from the industry analysis that Fortran and C can still be used and relevantly applied in financial and government industries. For Focus and SQL, they would be essential for business procedures and information sectors. Python And MATLAB while technically able to be implemented in many areas, can excel in computer systems applications and engineering. HTML and Java are essential to computer infrastructure systems and applications.

5 Challenges and Improvement

Some challenges in the thesis include the specifications of certain languages and the classifications of other languages. In the references and source material, there has been a trend of indecision and lack of proper agreement on what some languages are in terms of generations - there are inconsistencies in whether some languages were 3GLs or4GLs. Thus, the research had to expand to MGLs and NGLs to properly fit in the scope of programming languages that came about within this timeline. This is for the sake of an easy-to-understand system, which allows us to properly group languages by focus and by specialization.

Some languages had to be referred to specifically as a branch that language consistently was used in, or a software in which the programming language was most used. One of the bigger examples would be excel and focus. While Excel primarily relies on Focus, Focus has other implementations and uses. Because excel is the most widely used and popular way of using the language, it must be specified that the research focuses on Focus within the Excel software.

The language which had to be specified in its classification was Java. Java was specifically placed in NGLs, because the rights to Java and the patent to it are owned by Oracle. Oracle has implemented Java as a web page and website development programming language, as well as having many programs which allow for quick uses and creations of queries, servers, and other web service matters. Java is also more popular in web development and web application creation than it is in database and data management.

Other problems came in the form of gathering research and being unable to get properly clear data. Programming languages like R, C, Ruby, MATLAB, and Python have consistently shown to have had entries in the search results which were unrelated to both the field and the topic that they were implemented in. Languages like R and C had difficulties since some unrelated papers would get included in the author’s name or the author's name being shortened to a single letter. With C, a still widely used language, filtering worked but not with R. The same issue came up with Ruby as there were some scientists and publishers who had the first name Ruby.

In some cases, like with Ruby and C proper filtering was used to narrow down searches as accurately as feasibly possible. In R, the number of publications in our current field, compared to the fields that were not included, was overwhelmingly confusing and could not be properly checked for each individual publication. Because of that, R was not included in the analysis of MGLs.

MATLAB, Java, and Python had difficulties relating to other meanings concerning their names. MATLAB has the exact same spelling as a town or city in Bangladesh whereas Java is the name of an Island in the same part of the world. Python was originally named after the animal. Because of this some social science, zoology, geography, and unrelated papers were often found in between the papers that had discussed programming languages. Because of this, the possibility of implementing more accurate data is open to consideration.

Another challenge that had come across during data gathering for analysis was the data concerning languages applicable to specific industries and that industry's employment. It is very difficult to find records that go all the way back to 1955, the records that contain classifications by industry, and accurate analysis of those industries and their employment rates. Most government and labor unions websites disclose this but oftentimes there are only records up to a specific point in time or they only go back to the 2000s. There were only a few cases where a dataset was prevalent within the timeline that has been defined.

Even so, these industries do not include the jobs that are included in their industries. Oftentimes, the growth of this industry does not strictly depend on whether a programming language is able to make work within this industry easier. Ideally, data sets that document the workforce in industries where the programming language is actively used. However, in the current time limitations and resources, it is not possible to locate such databases. It was necessary to use conclusions based off industry employment and programming language publication history. The language popularity and usefulness in its specific industry is often speculated on how useful this programming language should be in that industry.

6 Conclusion

In conclusion, the result of this thesis, combined with the theoretical base, documentation, and analysis, can be properly assessed on what has been learned, what can be improved and what this will mean in the future.

To start off, whether the research questions of this thesis have been answered will be analyzed. The differences between old and new programming languages have been found and defined.

The main difference is between old and new programming languages are primarily what was more necessary in the era that they have come about from.

For old languages computation and the usage of data as a value is what separates them from the newer ones. As for the newer ones the usage of data as an object with multiple traits that could be implemented is what is more commonly seen in the languages that are considered new.

Older languages are more compact and can write programs much quicker, whereas newer ones are more case sensitive advanced and are able to store more and execute much more advanced functions.

While there are connections between older languages and newer languages and specific industries, which industry primarily uses this language is entirely dependent on the language itself.

In addition to the knowledge already gained from the research and thesis, new knowledge has been gained. Some languages are oftentimes disputed to be in either one generation or the other, based on the similarities of the of both old and new generations, while some programming languages do not fit in either generation.

While this thesis offers an analysis and a possible knowledge base for future choices and implementations of programming languages, this is not to say that this is the only possible way of going about this. As mentioned in the challenges and improvement, this thesis is not necessarily perfect and is open to future analysis or change in methods for the improvement of the quantitative data.

References

Adams, J. C.;Brainerd, W. S.;Martin, J. T.;Smith, B. T.;& Wagener, J. L. (1992). *FORTRAN 90 Handbook: complete ANSI/ISO Reference.* New York: Intertext Publications : McGraw-Hill Book Co. Noudettu osoitteesta https://micro.ustc.edu.cn/Fortran/Fortran%2090%20Handbook.pdf

Alubady, R. (2009). Programming Languages: Classification, Execution Model, and Errors. *Programming Fundamentals*, 1-7.

Alzahrani, A. A. (2020). 4GL Code generation: A Systematic Review. *International Journal of Advanced International Computer Science and Applications, 11*(6), 178-185. https://doi.org/10.14569/IJACSA.2020.0110623

Backus, J. (1978). *The History of Fortran 1, 2 and 3* (Osa/vuosik. 8). San Jose: IBM Research Laboratory. Noudettu osoitteesta https://www.softwarepreservation.org/projects/FORTRAN/paper/p165-backus.pdf

Backus, J. W. (1959). The Syntax and Semantics of the Proposed International Algebraic Language of Zürich ACM-GAMM Conference. *Zürich ACM-GAMM Conference.* Zurich: International Business Machines Corp. Noudettu osoitteesta https://www.softwarepreservation.org/projects/ALGOL/paper/Backus-Syntax_and_Semantics_of_Proposed_IAL.pdf

Bartoníček, J. (2014). PROGRAMMING LANGUAGE PARADIGMS & THE MAIN PRINCIPLES OF OBJECT-ORIENTED PROGRAMMING. *CRIS Bulletin*(1), 93-99. https://doi.org/10.2478/cris-2014-0006

Berners-Lee, T. (1993). *Hypertext Markup Language (HTML) - A Representation of Textual Information and MetaInformation for Retrieval and Interchange.* Noudettu osoitteesta https://www.w3.org/MarkUp/draft-ietf-iiir-html-01.txt

Beyer, K. (2009). *Grace Hopper and the Invention of the Information Age.* MIT Press. Noudettu osoitteesta https://www.jstor.org/stable/j.ctt5hhgwc

Borkowsk, S. (2006). *FORTRAN and COBOL Genealogy Tree.* Noudettu osoitteesta https://commons.wikimedia.org/wiki/File:Algol%26Fortran_family-by-Borkowski.svg#/media/File:Algol&Fortran_family-by-Borkowski.svg

Brookshear, J.;& Brylow, D. (2020). *Computer Science: An Overview* (13th p.). Pearson Education.

Cambridge University. (2017). *Cambridge Academic Content Dictionary.* Cambridge University Press and Assessment. Noudettu osoitteesta https://dictionary.cambridge.org/us/dictionary/english/

Canneyt, M. V. (2021). *Reference guide for Free Pascal.*

Carr, D. E.;& Kizior, R. J. (2003). Continued Relevance of COBOL in Business and Academia: Current Situation and Comparison to the Year 2000 Study. *Information Systems Education Journal, 1*(52), 1-10. Noudettu osoitteesta https://www.isedj.org/1/52/ISEDJ.1(52).Carr.pdf

Chamberlin, D. D.;& Boyce, R. F. (1974). SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE. *1974 ACM SIGFIDET Workshop on Data Description, Access and Control* (ss. 249-264). San Jose: ACM. Noudettu osoitteesta https://web.archive.org/web/20070926212100/http://www.almaden.ibm.com/cs/people/chamberlin/sequel-1974.pdf

Champeon, S. (2001). *JavaScript: How Did We Get Here?* O'Reilly. Noudettu osoitteesta https://web.archive.org/web/20160719020828/https://archive.oreilly.com/pub/a/javascript/2001/04/06/js_history.html

Chonacky, N.;& Winch, D. (2005). Reviews of Maple, Mathematica, and Matlab: Coming Soon to a Publication Near You. *Computing in Science & Engineering, 7*(2), 9-10. https://doi.org/10.1109/MCSE.2005.39

Chowdhary, K. (2020). *On the Evolution of Programming Languages.* Jodhpur: Dapartment of Computer Science and Engineering, Jodhpur Institute of Engineering and Technology. Noudettu osoitteesta : https://www.researchgate.net/publication/342733980

Christiansen, T.;& Torkington, N. (1998). *Perl Cookbook.* O'Reilly. Noudettu osoitteesta https://www.geos.ed.ac.uk/~bmg/software/Perl%20Books/OReilly.Perl.Cookbook.pdf

Clarke, A. (1982). USING STANDARD FORTRAN - PAST, PRESENT, AND FUTURE. *Datalink Newspaper*, 10-16. Noudettu osoitteesta https://dl-acm-org.ezproxy.hamk.fi/doi/pdf/10.1145/382116.383051

Coughlan, M. (2014). *Beginning COBOL for programmers* (1 p.). APress. Noudettu osoitteesta https://link.springer.com/book/10.1007/978-1-4302-6254-1

Dale, L.;Jonnalagadda, K.;& Suleman, K. (2021). *Oracle Product Hub User's Guide,.* Oracle. Noudettu osoitteesta https://docs.oracle.com/cd/E26401_01/doc.122/e48837.pdf

Dollard, K. (2018). *Visual Basic in .NET Core 3.0.* Microsoft. Noudettu osoitteesta https://devblogs.microsoft.com/vbteam/visual-basic-in-net-core-3-0/

E & ICT Academy. (2018). C - the mother of all languages. *IITK*. Noudettu osoitteesta https://ict.iitk.ac.in/c-the-mother-of-all-languages/

Economides, N.;& Katsamakas, E. (2006). Linux vs. Windows: A Comparison of Application and Platform Innovation Incentives for Open Source and Proprietary Software Platforms. *The Economics of Open Source Software Development*, ss. 208-218. Noudettu osoitteesta http://neconomides.stern.nyu.edu/networks/Economides_Katsamakas_Linux_vs._Windows.pdf

Enlyft. (2023). *Companies using MySQL.* Enlyft. Noudettu osoitteesta https://enlyft.com/tech/products/mysql

Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics. (1939). *All Employees, Government (CES9000000001).* ALFRED, Archival Economic Data of St. Louis Federal Reserve Bank.

Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics. (1939). *All Employees, Information (CES5000000001).* ALFRED, Archival Economic Data of St. Louis Federal Reserve Bank. Noudettu osoitteesta https://alfred.stlouisfed.org/series?seid=USINFO

Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics. (1939). *All Employees, Private Education And Health Services (USEHS).* ALFRED, Archival Economic Data of St. Louis Federal Reserve Bank. Noudettu osoitteesta https://alfred.stlouisfed.org/series?seid=USEHS

Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics. (1939). *All Employees, Professional and Business Services (USPBS).* ALFRED, Archival Economic Data of St. Louis Federal Reserve Bank. Noudettu osoitteesta https://alfred.stlouisfed.org/series?seid=USPBS

Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics. (1955). *All Employees, Local Government Education (CES9093161101).* ALFRED, Archival Economic Data of St. Louis Federal Reserve Bank. Noudettu osoitteesta https://alfred.stlouisfed.org/series?seid=CES9093161101

Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics. (1990). *All Employees, Accounting, Tax Preparation, Bookkeeping, And Payroll Services (CES6054120001).* ALFRED, Archival Economic Data of St. Louis Federal Reserve Bank. Noudettu osoitteesta https://alfred.stlouisfed.org/series?seid=CES6054120001

Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics. (1990). *All Employees, Architectural, Engineering, And Related Services (CES6054130001).* ALFRED, Archival Economic Data of St. Louis Federal Reserve Bank. Noudettu osoitteesta https://alfred.stlouisfed.org/series?seid=CES6054130001

Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics. (1990). *All Employees, Computer Systems Design and Related Services (CES6054150001).* ALFRED, Archival Economic Data of St. Louis Federal Reserve Bank. Noudettu osoitteesta https://alfred.stlouisfed.org/series?seid=CES6054150001

Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics. (1990). *All Employees, Computing Infrastructure Providers, Data Processing, Web Hosting, And Related Services (CES5051800001).* ALFRED, Archival Economic Data of St. Louis Federal Reserve Bank. Noudettu osoitteesta https://alfred.stlouisfed.org/series?seid=CES5051800001

Federal Reserve Bank of St. Louis; U.S. Bureau of Labor Statistics. (1990). *All Employees, Finance and Insurance (CES5552000001).* ALFRED, Archival Economic Data of St. Louis Federal Reserve Bank. Noudettu osoitteesta https://alfred.stlouisfed.org/series?seid=CES5552000001

Ferguson, A. (2018). *A History of Computer Programming Languages.* Noudettu osoitteesta https://cs.brown.edu/~adf/programming_languages.html

Flahive, P. (2019). *How COBOL Still Powers The Global Economy At 60 Years Old.* Texas. Noudettu osoitteesta https://web.archive.org/web/20190524035248/https://www.tpr.org/post/how-cobol-still-powers-global-economy-60-years-old

Giorgi, F. M.;Ceraolo, C.;& Mercatelli, D. (May 2022). The R Language: An Engine for Bioinformatics and Data Science. *Life (Basel)*. https://doi.org/10.3390/life12050648

Glassey, R. (1993). *Numerical Computation Using C.* Academic Press. https://doi.org/10.1016/C2013-0-10730-1

Goel, A. (2010). *Computer Fundamentals.* Dorling Kindersley.

Gosling, J.;& McGilton, H. (1996). *The Java Language Environment.* Oracle. Noudettu osoitteesta https://www.oracle.com/java/technologies/language-environment.html

Grad, B.;& Westport, C. (2005). RAMIS and NOMAD - National CSS. *Computer History Museum*, (s. 54).

Gration, E. (2022). Microsoft Excel Statistics 2022: Skills Usage and More. *Micro Biz Mag*. Noudettu osoitteesta https://www.microbizmag.co.uk/microsoft-excel-statistics/

Grogan, M. (2018). *Python vs. R for Data Science.* O'Reilly. Noudettu osoitteesta https://tuxdoc.com/download/michael-grogan-python-vs-r-for-data-science-oreilly-media-2018_pdf

Gürer, D. (2002). Pioneering women in computer science. *ACM SIGCSE Bulletin, 34*(2), 175-180. https://doi.org/10.1145/543812.543853

Havenstein, H. (2007). WebFocus-excel Link Strenghthened. *Computerworld*.

Heering, J.;& Mernik, M. (2002). Domain-Specific Languages for Software Engineering - Minitrack Intoduction. *Proceedings of the 35th Hawaii International Conference on System Sciences - 2002. 9.* Big Island: IEEE Computer Society. https://doi.org/10.1109/HICSS.2002.10113

IBM. (2012). *FORTRAN: The Pioneering Programming Language.* IBM. Noudettu osoitteesta https://www.ibm.com/ibm/history/ibm100/us/en/icons/fortran/

Ihaka, R. (2022). *The R project: A Brief History and Thoughts About The Future.* University of Auckland. Noudettu osoitteesta https://www.stat.auckland.ac.nz/~ihaka/downloads/Otago.pdf

Information Builders. (31. March 1986). Information Builders Focus/VAX. *Computerworld, 10*(13), 52. Noudettu osoitteesta https://books.google.fi/books?id=nvgwHu6zOfQC&printsec=frontcover&hl=fi&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

Information Builders. (5. March 1990). February 12, 1990, IBM introduces RISC SYSTEM/6000 FOCUS 4GL was there. *Computer World, 14*(10), 13. Noudettu osoitteesta https://books.google.fi/books?id=cX3dYiZfXe8C&printsec=frontcover&hl=fi&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

Information Builders. (1994). *FOCUS for S/390 CSM Installation Guide.* Information Builders. Noudettu osoitteesta https://ecl.informationbuilders.com/focus/index.jsp?topic=/shell_7709/FOCUS_NewFeatures/source/opener.htm

Information Builders. (2019). *Release 7.7.09 - Overview and Operating Environments.* Information Builders. Noudettu osoitteesta https://ecl.informationbuilders.com/focus/index.jsp?topic=/shell_7709/FOCUS_NewFeatures/source/opener.htm

J. W. Backus, F. L. (May 1960). Report on the algorithmic language ALGOL 60. *Communications of the ACM, 3*(5), 299–314. https://doi.org/10.1145/367236.367262

Kahanwal, D. (October 2013). Abstraction Level Taxonomy of Programming Language Frameworks. *International Journal of Programming Languages and Applications, 3*(4), 6-11. https://doi.org/10.5121/ijpla.2013.3401

Ketler, K.;& Smith, R. D. (1992). Differences Between Third and Fourth Generation Programmers: A Human Generation Programmers. *Information Resources Management Journal, 5*(2), 25-35. https://doi.org/10.4018/irmj.1992040103

Kuhlman, D. (2013). *A Python Book: Beginning Python, Advanced Python, and Python Exercises.* Dave Kuhlman. Noudettu osoitteesta https://www.davekuhlman.org/python_book_01.pdf

Lauckner, K. F.;& Lintner, M. D. (2000). *The Computer Continuum* (2nd p.). New Jersey, USA: Prentice Hall PTR.

Leisch, F. (September 2001). Editorial. *R News: The Newsletter of the R Project, 1*, 40.

Lerdorf, R. (2002). *PHP Pocket Reference.* O'Reilly. https://doi.org/10.5555/572518

MacDonell, D. G. (1993). *Software Development, CASE Tools And 4GLs - A Survey of New Zealand Usage (Part 1).* Department of Information Science University of Otago. Noudettu osoitteesta https://ourarchive.otago.ac.nz/bitstream/handle/10523/928/dp1993-04-online.pdf?sequence=3&isAllowed=y

Maeda, S. (2002). *The Ruby Language FAQ.* Noudettu osoitteesta https://ruby-doc.org/docs/ruby-doc-bundle/FAQ/FAQ.html

Mangano, S. (2010). *Mathematica Cookbook.* O'Reilly. Noudettu osoitteesta https://math.bme.hu/~jtoth/Mma/Mathematica%20Cookbook.pdf

Mathworks. (2023). MATLAB Help center. Noudettu osoitteesta https://www.mathworks.com/help/matlab/index.html

Mendez, M. (2014). *The Missing Link: An Introduction to Web Development and Programming.* Open SUNY Textbooks. Noudettu osoitteesta https://knightscholar.geneseo.edu/cgi/viewcontent.cgi?article=1016&context=oer-ost

Microsoft. (2021). *Visual Basic Language Reference.* Microsoft. Noudettu osoitteesta https://learn.microsoft.com/en-us/dotnet/visual-basic/language-reference/

Netscape Communications Corporation. (1995). NETSCAPE AND SUN ANNOUNCE JAVASCRIPT, THE OPEN, CROSS-PLATFORM OBJECT SCRIPTING LANGUAGE FOR ENTERPRISE NETWORKS AND THE INTERNET. *NETSCAPE Company Press Relations*. Noudettu osoitteesta

https://web.archive.org/web/20070916144913/https://wp.netscape.com/newsref/pr/newsrelease67.html

Oracle. (2020). *Top big data analytics use cases.* Oracle. Noudettu osoitteesta https://www.oracle.com/a/ocom/docs/top-22-use-cases-for-big-data.pdf

Page, C. G. (1988). *Professional Programmer's Guide to Fortran77.* University of Leicester. Noudettu osoitteesta https://www.star.le.ac.uk/~cgp/prof77.pdf

Paquet, J.;& Mokhov, S. A. (2010). *Comparative Studies of Programming Languages.* Montreal: Department of Computer Science and Software Engineering of Concordia University. Noudettu osoitteesta http://arxiv.org/abs/1007.2123

Pigott, D. (2006). *Encyclopedia of Computer Languages.* Australia: Murdoch University. Noudettu osoitteesta http://hopl.murdoch.edu.au/

Python Software Foundation. (2023). *Python/C API Reference Manual.* Noudettu osoitteesta https://docs.python.org/3/reference/index.html

Rall, L. B. (1987). An introduction to the scientific computing language Pascal-SC. *Pergamon Journals, 14*(1), 53-69. https://doi.org/10.1016/0898-1221(87)90181-7

Sammet, J. E. (1972). Programming languages: history and future. *Communications of the ACM, 15*(7), 601-610. https://doi.org/10.1145/361454.361485

Sieger, N. (2006). *RubyConf: History of Ruby.* Noudettu osoitteesta http://blog.nicksieger.com/articles/2006/10/20/rubyconf-history-of-ruby/

Sinha, P.;& Sinha, P. K. (2021). *Computer Fundamentals* (8th p.). BPB Publications. Noudettu osoitteesta https://librarywala.com/books/663283376946-computer-fundamentals-concepts-systems-amp-applications--8th-edition

Smirnova, S.;& Tezuysal, A. (2022). *MySQL Cookbook.* Sebstopol: O'Reilly. Noudettu osoitteesta https://sgp1.vultrobjects.com/books/Codeing/MySQL%20Cookbook%2C%204th%20Edition.pdf

Sun Developer Network. (1995). *The History of Java Technology.* Sun Developer Network. Noudettu osoitteesta https://web.archive.org/web/20100210225651/http://www.java.com/en/javahistory/

The Perl Foundation. (2023). *Perldoc Browser.* Noudettu osoitteesta https://perldoc.perl.org/perl

Thomas, D.;Fowler, C.;& Hunt, A. (2001). *Programming Ruby - Pragmatic Programme's Guide.* Noudettu osoitteesta https://ruby-doc.com/docs/ProgrammingRuby/

TIBCO. (2022). *TIBCO FOCUS for Mainframe Compatibility Information.* TIBCO Software. Noudettu osoitteesta https://ecl.informationbuilders.com/focus/topic/org.eclipse.help.base/doc/focus_mainframe_compatibility_info.pdf

TIOBE Index. (n.d.). *TIOBE - C.* TIOBE Software. Noudettu osoitteesta https://www.tiobe.com/tiobe-index/c/

TIOBE Index. (n.d.). *TIOBE - COBOL.* TIOBE Software. Noudettu osoitteesta https://www.tiobe.com/tiobe-index/cobol/

TIOBE Index. (n.d.). *TIOBE - Delphi/Object Pascal.* TIOBE Software. Noudettu osoitteesta https://www.tiobe.com/tiobe-index/delphi-object-pascal/

TIOBE Index. (n.d.). *TIOBE - FORTRAN.* TIOBE Software. Noudettu osoitteesta https://www.tiobe.com/tiobe-index/fortran/

TIOBE Index. (n.d.). *TIOBE Index - Java.* TIOBE. Noudettu osoitteesta https://www.tiobe.com/tiobe-index/java/

TIOBE Index. (n.d.). *TIOBE Index - JavaScript.* TIOBE. Noudettu osoitteesta https://www.tiobe.com/tiobe-index/javascript/

TIOBE Index. (n.d.). *TIOBE Index - MATLAB.* TIOBE. Noudettu osoitteesta https://www.tiobe.com/tiobe-index/matlab/

TIOBE Index. (n.d.). *TIOBE Index - Perl.* TIOBE. Noudettu osoitteesta https://www.tiobe.com/tiobe-index/perl/

TIOBE Index. (n.d.). *TIOBE Index - PHP.* TIOBE. Noudettu osoitteesta https://www.tiobe.com/tiobe-index/php/

TIOBE Index. (n.d.). *TIOBE Index - Python.* TIOBE. Noudettu osoitteesta https://www.tiobe.com/tiobe-index/python/

TIOBE Index. (n.d.). *TIOBE Index - R.* TIOBE. Noudettu osoitteesta https://www.tiobe.com/tiobe-index/r/

TIOBE Index. (n.d.). *TIOBE Index - Ruby.* TIOBE. Noudettu osoitteesta https://www.tiobe.com/tiobe-index/ruby/

TIOBE Index. (n.d.). *TIOBE Index - Visual Basic.* TIOBE. Noudettu osoitteesta https://www.tiobe.com/tiobe-index/visual-basic/

Venners, B. (2003). The Making of Python: a Conversation with Guido van Rossum. *Artima*.

Wall, L.;Christiansen, T.;& Orwant, J. (2000). *Programming Perl* (3 p.). (L. Mui, Toim.) O'Reilly. Noudettu osoitteesta https://ndl.ethernet.edu.et/bitstream/123456789/26985/1/Larry%20Wall.pdf

Wirth, N. (2000). *The Development of Procedural Programming Languages: Personal Contributions and Perspectives.* Niklaus Wirth. https://doi.org/10.1007/10722581_1

Wirth, N. (March 2021). 50 Years of Pascal. *Communications of the ACM, 64*(3), 39-41. https://doi.org/10.1145/3447525

Wolfram. (2023). *Wolfram Products and Services.* Noudettu osoitteesta https://www.wolfram.com/products/?source=nav

Wolfram, S. (1990). Commentary. *THE MATHEMATICA JOURNAL, 1*(1), 14-16. Noudettu osoitteesta https://content.wolfram.com/uploads/sites/34/2020/07/who-uses-mathematica.pdf

World Wide Web Consortium. (1998). *Extensible Markup Language (XML) 1.0.* World Wide Web Consortium. Noudettu osoitteesta https://www.w3.org/TR/1998/REC-xml-19980210.pdf

World Wide Web Consortium. (2010). *What is CSS?* World Wide Web Consortium. Noudettu osoitteesta https://www.w3.org/standards/webdesign/htmlcss#whatcss

Annex 1: Material management plan

The research material of documentation consists entirely of quantitative data observed by open-source libraries and databases. The documentation material does not contain any sensitive, personal, or restricted data. Thus, it will be stored on the personal computer of the student. Data is stored in an Excel file. Two versions exist – the raw data and the visualized data.

Because none of the information and data is confidential or sensitive, basic security can be applied. Cloud services were not used during this thesis. Personal and/or confidential information was not gathered during this thesis.

The material will be kept for 1 year from the date of approval of the thesis, in case the results or data of the thesis needs to be confirmed or further used.

No privacy policy applies as no data gathered is restricted or personal.

All figures were made based on gathered data by the author. Documentation data was manually gathered by the author. Demand data was provided by The Federal Reserve Bank (FRED) of St. Louis in accordance to their privacy policy:

- You may access and browse our site without disclosing your personal data. The personal information you choose to provide us is voluntary. We do not require you to provide any information about yourself that you do not wish to share.

- As long as you don't engage in a prohibited/restricted use and do adhere to any applicable copyright restrictions, you are free to use FRED for your own non-commercial, educational, and personal uses. FRED allows you to do a lot of things for your own personal use without requesting permission from the Federal Reserve Bank of St. Louis, such as: Include FRED graphs and maps in a private presentation; Create graphs and maps to teach about the data in a classroom setting; Include FRED graphs and maps in your personal blog posts; create individual visualizations of FRED data; View, download, and print FRED content.