July 2022

Ben Mullan

Extended Project Qualification

What are the Most Effective Means of Addressing the Software Developer Shortage in the UK?

Overview

Introduction

The UK has an endemic digital deficit. It is estimated that whilst half the countries' labourers lack "*basic*" digital literacy, there will be a need for 1.5 Million workers with "*advanced*" computer skills in the next two years[1]. Before the height of the pandemic, the UK had an advertised ~5000 software developer vacancies in April alone. One year later, this figure had become ~10,000[2]. Though the UK is not alone in facing this growing shortage, it lags far behind the US, China, and India, in terms of absolute numbers of developers, and in recent years, has been resorting to hiring from this overseas talent pool, given the shortage of developers at home[3]. As this paper will elucidate, this model is unsustainable for the UK and its economy.

As an *indispensable* means of facilitating forthcoming technological development and progression, an educated, skilled, and innovative workforce of software analysts, systems' architects, and programmers, are required. The UK – currently a global frontrunner in specialist training, engineering, and innovation – *has a duty* to maintain this economic advantage, and lead progression into the age of the *MetaVerse*[4], *Cloud* Computing, and other forms of ubiquitous computer systems.

Why exactly the UK has this shortage however, may not be entirely obvious. Indeed, the last century has witnessed a great many technological breakthroughs occur on British soil – from Alan Turing's propositions on computational logic, to the Sinclair, Acorn, and BBC Micro Computers in the 80's, and more recently the now internationally-renowned Raspberry Pi foundation, with their CreditCard-sized Computers. However, *Brexit* and its restrictions on the mobility of labour (i.e. the removal of freedom of movement for workers coming to the UK), has resulted in many EU software developers leaving, who were previously the backbone of the development workforce.

To make matters worse, the present context of a global pandemic has also placed a greatly increased demand on digital services, and consequently on the developers responsible for these systems. Shortcomings in businesses' IT infrastructure were *exposed* during the lockdowns, and exacerbated due to the number of workers who left the IT sector during this time. May of 2021 found 58% of British and Irish IT staff planning to change roles in the next 6 to 12 months[5], predominantly on account of the "burn out" they experienced during the workload of the lockdowns. Adverts for Information Technology employment vacancies in the United Kingdom are exceeded only by those for healthcare, with a fifth of all vacancies in 2019 being in the technology sector[3]. The problem, however, is *chronic*; not only has the shortage existed for some time, but present factors – such as a lack of resources to involve the younger generation – mean that the deficiency is predicted to continue for several decades. In fact, the first *mention* of the United Kingdom in the *International Informatics Olympiad* Leaderboard[6] – an established global ranking of a nation's young programming talent – was at 122nd place. China and the USA were incidentally each mentioned 4 times therebefore. The UK needs to *up its game*.

Software Development is one of many Services which are vitally important to the British Economy, as they tend to be high-value exports[7].

"Our services export share is around 12% of GDP, compared to about 8% in Germany and France, 4% in the United States and 3% in Japan. In absolute terms, the UK is the second-largest exporter of services in the world behind the United States."[7]

As is substantiated by these data, services contribute to economic growth, accounting for nearly three quarters of economic output (GDP) in most developed nations[7]. As the trend towards digitisation grows, the importance of services will only rise – the UK must retain and enhance its advantage. This will enable it to increase its productivity and thereby improve living standards, as higher growth can be shared across the population[8]. Without the labour force to retain and build our competitiveness, UK living standards may even fall, since we will not have the developers to meet our own needs within the UK, never mind the ability to export developers' skills.

This paper will argue that there is no *single* solution to the national shortage of software developers. Rather, a combination of short- and long-term partial solutions must be implemented in tandem, in order to alleviate what is a fundamentally continuous problem, vital to the future of this nation, and those with which it trades. Specifically, this paper explores several *economic* solutions - including raising developers' salaries, recruiting developers from abroad, and teaching programming skills in schools – as well as *technical* solutions; Low- and No-Code Programming, the effects of "*Legacy Systems*", and the growing role of Artificial Intelligence in software development processes.

(Incidentally, some sources and sections herein occasionally refer to "IT Professionals" as opposed to "Software Developers" specifically, but since the latter is merely a subset of the former, the same trends and principles are applicable and observable.)

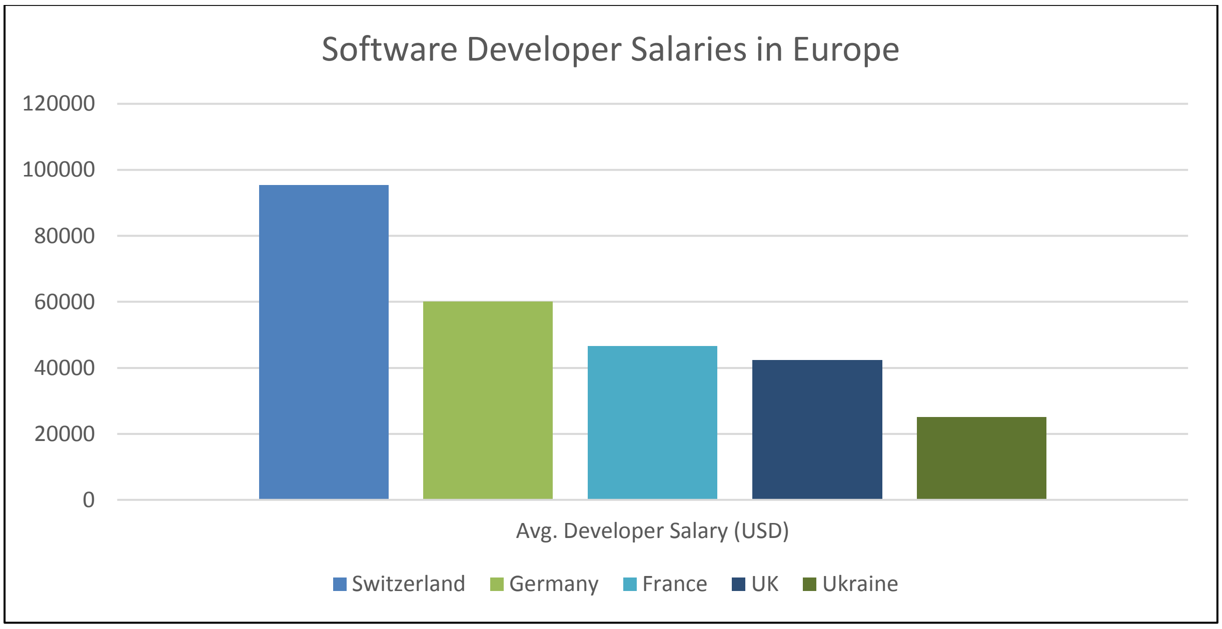
Exploring Potential Solutions

Hiring Developers from Abroad

In the UK (and, for that matter, most developed nations) there are labour shortages across numerous specialised sectors which require several years of preparatory training. A lack of software developers (one of the professions impacted hereby) in the UK for instance, means that 59% of *global* companies currently hire developers from abroad, in order to “optimize software development costs”[9].

The COVID-19 pandemic is *partially* responsible for this[10], but circumstantial factors such as *Brexit* and the UK’s aging population, have encouraged an estimated 200,000[11] workers from the European Union to return to the continent, and, an increasing proportion of the *remaining* workforce (particularly self-employed men over 50[12]) to retire, or withdraw from the labour force. The cumulative effect of this is that employers struggle to recruit developers from other sectors, and the *multi-sectoral* shortage will further justify the demands of the current IT workforce for higher wages.

These increased salaries, however, have the knock-on effect of increasing a firm’s costs. Since most companies operate to *maximise* their profits[13], they will attempt to reduce expenditure by *lowering* the wages of their employees. But since the workers in the UK are beginning to expect *higher* salaries – for the aforementioned reason – organisations will look abroad, to hire *cheaper* developers. This trend is known as “*off-shoring*”.



[Source 14: “*Software Developer* Salaries around the world”]

One particularly popular off-shoring location is the Ukraine, where - as can be seen from the figure above - the average salary of a software developer is 41% lower than that of a developer in the UK. Its relative proximity to the UK also makes it an *attractive* recruitment destination, since *Tinbergen’s*

Gravity Theory states that countries tend to trade most with nations which are closest to them[15]. In fact, "the IT outsourcing industry in Ukraine currently employs *over 200,000* tech specialists"[8].

Certainly, on a global scale, there is an increasing trend towards exporting IT services. A Price Waterhouse Cooper report in 2020 showed that "Globally, the fastest growing sector since 2005 has been computer and information services, driven predominantly by the emerging markets. Its share of global service exports has grown from 7% to 10% in the last 15 years"[16]. Therefore, as a consequence of the demonstrably growing demand, using a growing quantity of foreign workers in the sector appears to be an inevitability.

Due, however, to the typically short-term nature of contracts with oversees developers, there are some concerns with the quality of output. The employee will not be familiar with the firm's culture or standards, and, as they are hired on a temporary contract with plenty of alternative work available elsewhere, they will be less committed to providing a high-quality service. In actuality, the concentration of skilled labour in the Ukraine is so high, that is has become a source of competitive advantage; the nation has specialised in an industry in which they can now satisfy demand more efficiently than competitors, due to lower costs[17]. This is known as "Ricardo's Comparative Advantage" in economics, and it ultimately translates into higher profits for the UK firms who make use of the cheaper offshore workers. This affords these British companies money to reinvest into further training and innovation – thereby helping to subsidise the desired increase in the supply of workers, and, improve the UK's efficiency.

Whilst outsourcing to the Ukraine and similar nations such as Poland and Slovakia with skilled, young populations, might solve *some* of the UK's shortages in the immediate future, such specialisation also has costs. British firms may previously have considered relocating to the Ukraine, thus depriving the UK of jobs. The determining factor in the extent to which British firms relocate, may well be the quality and dependability of Ukrainian developers. Qualitative examinations to date infer that the Ukraine ranked highly, with "good" levels of English, technical expertise, and familiarity with new and emerging technologies[8]. Yet, with only 200,000 workers in the Ukraine and the ravages of war, outsourcing will no longer satisfy the UK's 1.5 million[1] vacancies.

The conclusion is that hiring from abroad offers far less potential than it did previously. It will continue to fill some gaps, particularly when EU nationals are offered remote working contracts, but a UK supply of labour is required for both quality and practical reasons. For example, firms must pay tax on any IT equipment that they send out to remote workers abroad, and then repay when the kit is sent back to be repaired or replaced[11].

It is clear that British voters and politicians can be a part of the solution. The British government has laterally introduced the H*omes for Ukrainians* scheme. At the time of writing, 138,000 households in the United Kingdom had signed up to host a Ukrainian family. A positive, unintended consequence, hereof may be that more software developers are forced to live in the UK, helping to assuage some of our developer shortages. However, this fails to alleviate the global problem, which has knock-on effects for the UK.

To make matters worse, restrictions on the free movement of labour – brought on by Brexit – are deterring many potential foreign workers, and even causing some existing employees to leave:

> *"Britain is also reckoning with Brexit-related immigration changes. That is why in many countries, industries, such as hospitality, that rely on foreigners, face the most acute shortages. Politicians must be clear that closed borders will come with a painful price tag - or change tack."*[16]

To combat this, *Freedom of Movement could* be reinstated for workers in industries of priority - such as software development. The UK government has, in fact, instated an "Exceptional Talent Visa", which "allows exceptionally-talented people from around the world to come and work in the tech industry here". These are, however, little-known and hard to acquire[18]. Because the UK has historically been the second most competitive service exporter globally, without a more flexible immigration regime, it is in grave danger of losing its comparative advantage[19] to the detriment of its export revenue.

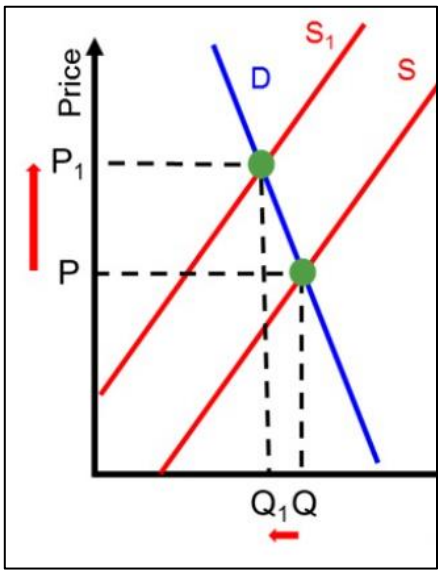
The characteristically distant nature of offshore development means that the largest pragmatic issue is commonly the difficulty in communication. Complex software systems tend to have correspondingly complex requirements, which must be clearly communicated to, and understood by, all parties involved in the developmental processes. *Gopal* writes that "the lack of proximity hampers a client's ability to monitor its vendors and coordinate development activities closely"[20]. Despite this, the International Data Corporation (IDC) forecasts the size of the Software Development Industry in India – another prominent outsourcing location – to soon be worth 8.2 Billion USD[21], given its current growth rate of 15.9% each year.

It is evident that the convenience and affordability of *offshoring* provides an inexpensive and relatively effortless "*quick fix*" to some of the immediate demand for custom IT systems; some businesses reduce their costs by as much as 70%[9] through their use of offshore development. Its validity as a sustainable and long-term solution however, is dubious because of the outlined concerns over quality, communications, and knock-on effects to the UK's economy; there is a danger that workers here will become complacent and acceptant that software development is best done elsewhere.

Raising Software Developers' Salaries

As a means of retaining existing workers, and providing financial incentive for the purposes of attracting new development staff, allocating higher wages to the profession does have the potential to compensate for *some* of the one million workers removed from the workforce by the COVID-19 pandemic[10].

This argument is supported by the economic principle of *Labour Market Analysis*; professions that require specialist training over a number of years are deemed to have an *inelastic supply* – one which cannot be easily made to increase, due predominantly to the duration of training that workers must undergo beforehand.



[Source 22: "*The Labour Market*"]

Initially, the supply of software developers in the UK is at quantity "*Q*". At this point, the supply of workers meets firms' demand for IT staff, giving employees a wage of "*P*". Due in part to the aforementioned effects of Brexit and the outbreak of COVID-19, many workers have left the sector, and, in so doing, effectively shifted the remaining supply of developers from "*S*" to "*S1*". For this reason, the quantity of developers shrinks from "*Q*" to "*Q1*", but the *wage rates* rise by significantly more than the quantity *falls*, thereby quantifying this *inelastic demand*.

Simply put, a higher advertised salary is required to actually attract new recruits, and even retain existing workers, because the rarity increases the justifiably-demandable wages, especially in the context of a national shortage of workers in this sector.

The UK government states that "roles requiring digital skills pay 29% (£8,300 per annum) over those roles that do not"[23]. It might be difficult to rationalise paying software developers *even more* than their current salaries, when the list of professions in greater need of financial input (e.g. Teachers, Junior Doctors, and The Police Service) is quite so unjustifiably long.

Ultimately therefore, simply *injecting* money into an already wealthy sector, in a vague attempt to provide a short-term *boost* to what is an inherently *long-term* problem, is not a reasonable solution.

Reforming Computer Science Education

Computer Programming is now on the national curriculum in England[24]. It is an increasingly integrated component of all classrooms and lessons, even across a range of non-computing subjects. Despite these offerings however, the UK Government report that only 11% of students choose to take GCSE Computer Science in secondary schools[25], and a considerably lower proportion actually pursue the subject post-16.

A prominent problem with Computer Science Education in Primary and Secondary Schools at present, is the lack of incentives for IT professionals to become teachers. Since a qualified software developer can command a considerably higher salary working in industry than would be possible as a teacher, it is difficult to recruit inspiring and knowledgeable teaching staff to deliver lessons. The Royal Society's 2017 report*, "After the Reboot: Computing Education in UK Schools"* found computing provision to be "patchy and fragile", and identified concerns around poor teacher recruitment, a shrinking workforce, and teachers' readiness to implement new curricula[25]. In fact, only 1 in 3 ICT teachers were found to have qualifications in their subjects, and pupils described the lesson experience as "repetitive and boring"[26]. Several concluding recommendations were put forward by the report, including:

"Higher education providers need to promote careers in computing education to a wide range of students.";

"Industry and academia should support and encourage braided careers for staff who want to teach as well as work in another setting."[25]

The latter is of particular relevance, as attracting skilled and qualified workers from industry to take up part-time teaching positions – in addition to their existing industrial employment – has huge potential to allow students to speak to and learn from professionals in the real working world. Such accessible and experienced staff are commonplace in – for instance – most *science* departments.

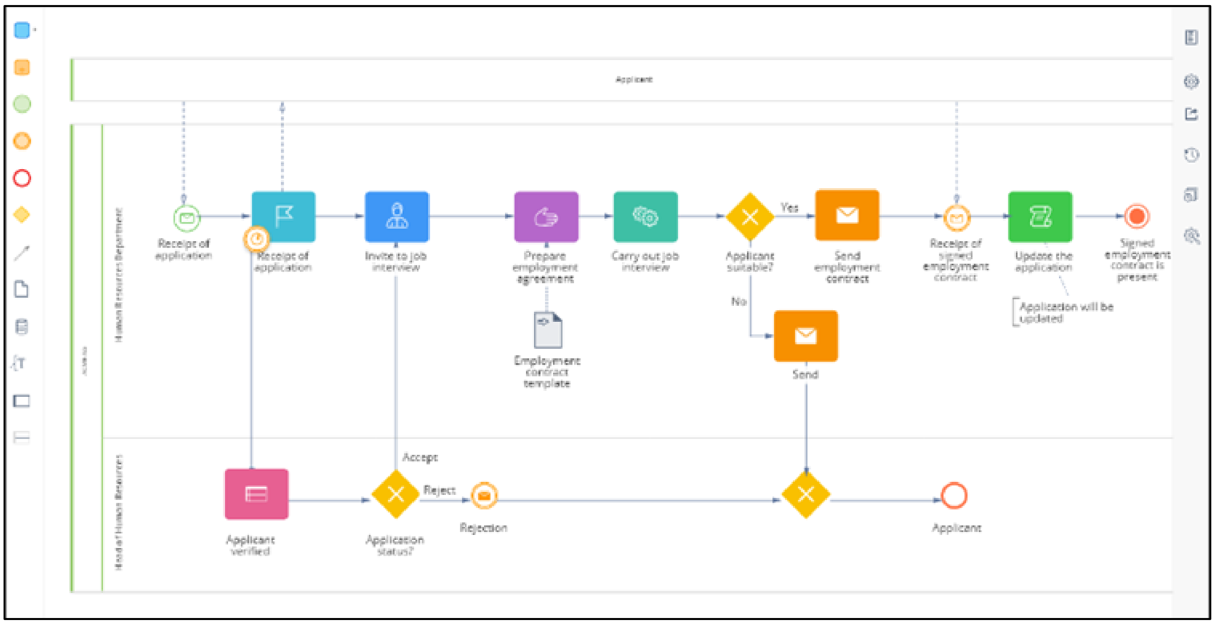
For many pupils, the most immediate problem is simply a lack of motivation and exposure to Programming as an academic and hobbyist field. Many subcultures regard "*coding"* as an activity (or even obsession) exclusively for a select number of socially-awkward nerd-like youths. Such views, if widely-adopted, are able to hinder the progress of the nation in delivering an adequate number of software developers in forthcoming generations. Therefore, it is important to identify strategies which can be implemented to regain the interest and enthusiasm of students, during their time in the educational systems. A study undertaken by the Raspberry Pi Foundation[24] (2021) evaluated their effectiveness of a selection of educational techniques for teaching Computer Science in primary and secondary schools in the UK. The report suggested that more *involved* teaching methods and applications of computing education – such as "physical computing" (e.g. building robots) and "project-based learning" – are better able to "increase intrinsic motivation", and encourage learners to "take ownership and pride in their work"[24]. Thus, teachers need to take a more practical approach in presenting content for computing lessons, as this is better able to fuel and stimulate tomorrow's programmers.

The current system leaves much to be desired from the perspective of both students and their teachers. The staff shortage and the lag in educator training are totally at odds with the chronic developer shortage. Recent evidence indicates that initiating engagement at this formative stage *can* have a significant long-term effect on the number of developers. The Chartered institute for IT recorded an unprecedented ~160,000 applicants to Computer Science degrees[27] in 2021. Nevertheless, Apprenticeships and Degrees should be further promoted by the government and by universities.

"Low-Code" and "No-Code" Development

In an attempt to allow a wide range of employees - *without* degrees in software development - to reap the benefits of writing their own programs, "Low-Code" (LCD) and even "No-Code" (NCD) development systems have, in recent years, rapidly risen in popularity[28]. These systems permit "business professionals with no coding experience to build applications"[29] for themselves.

Predictably, the likes of Microsoft, Oracle, and Alibaba have been amongst the first to jump aboard the Low/No Code bandwagon[28]. Their products facilitate "visual programming" by means of *drag-and-drop* component arrangement[53], such as can be seen in the diagram below.



[Source: An example of a *Low-Code Development* (LCD) Platform]

In addition to the ease of use, the primary selling point for this innovative technology is that it can facilitate "*faster development*" of software[29], and thereby deliver applications to market more rapidly. Furthermore, whilst conventional development platforms require an in-depth knowledge of *APIs* (to link different data services together), much of this unnecessary convolution is conveniently hidden from view with LCD platforms, making these systems ideal and even *specialised*, for inter-service data handling and processing. This variety of programming task is commonplace and in high-demand. Some employees in the research carried out by Luo (2021) also mentioned that the NCD systems allowed for a greater level of teamwork and collaboration when working on the same project concurrently, within a group[29].

Despite these attractive benefits, Microsoft themselves admit that experienced developers are still required, if any "*significant customisation*" is needed in a Low-Code application[30]. They also mention that *No-Code* implementations are "*intended to solve only one business challenge*" (i.e. a single process), which is a major limiting factor for the vast majority of instances where a custom solution was warranted in the first place. Luo (2021) concluded that "although using LCD(s) is easier and faster than coding, it still has a steep learning curve"[29] for users. Because of the abstract and complexity-masking nature of LCD, not only are such systems unfamiliar to experienced developers, but they also fail to provide their users with an insight into what underlying code is actually being run in their *own* applications. As a result, users of LCD and NCD products will struggle to develop a more profound understanding of the software they are writing, which in turn hinders their progression to more sophisticated and widely-adopted programming languages and systems, when compared to the conventional approach of simply learning a programming language the first time round.

Woo – Senior Technology Writer for the *Engineering* journal – contends that "the use of such no-code platforms—and their low-code cousins which require some minimal coding knowledge—is on the rise, a trend that represents a step towards a decades-long goal in computer science to automate coding". The overall opportunity for LCD and NCD systems presents an entry point for many inexperienced professionals who are unfamiliar with conventional programming languages, but nevertheless require the ability to produce *custom* business systems, to varying extents. What remains to be seen in an official capacity, however, is how the lack of customisation and sophistication currently characteristic of these systems might fare in the long-term, and how easily users of these systems might be able to progress onto more advanced and versatile developmental platforms when - or if - they saturate or outgrow the abilities of the NCD and LCD products.

It is nevertheless concludable that LCD and NCD systems promulgate *some* beneficial extent of programming knowledge, and thereby lessen the demand for consulting one-time-job developers.

“Legacy” Systems and Derived Demand

A *Legacy System* is defined as an “old”, usually “core” (essential) software system which may have been running in a company for decades in a very reliable fashion, but which does not make use of modern technologies or practices, and which might not “belong to strategic technology goals”[31].

A prime example of a programming language indicative of a Legacy System is *C.O.B.O.L.*[31]*, whereof this is an example:*

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELOWRLD.
PROCEDURE DIVISION.
    DISPLAY "Hello, World!".
    STOP RUN.
```

Here, however, is the equivalent program in a modern Programming language, *JavaScript*:

```
console.log("Hello, World!");
```

As can be seen from this illustrative comparison, “Legacy” systems and their scripts are often perceived to be needlessly cumbersome and indirect. In many ways, the biggest problem is that *because* so many of them are largely undocumented, costly, and “difficult to extract […] business logic” from[31], they are unapproachable for newer and younger developers who are not taught the older languages or methodologies. *COBOL*, for instance, isn’t even *mentioned* in the 20 most used programming languages of 2020[32].

Since 52% of Legacy Systems are said to be “reliable”, and 77% of them “business-critical”, many developers are having a rather difficult time convincing management boards to scrap them and reimplement the systems with modern technologies, which are *potentially* more efficient, easier to manage, and – importantly – maintainable and comprehensible by younger developers. In fact, *90% of the time*, the “lack of resources” for Legacy Systems is what warrants their eventual upgrading[31].

Replacing the old systems would – at first glance – actually have a *negative effect* on the national shortage. Many more jobs would be created, which would in turn culminate in an *increased* scarcity; there would effectively be the same supply, but more demand for the skill.

However, the principle behind replacing Legacy Systems is one example of a factor which creates *derived demand* – and is therefore illustrative of a wider set of stimuli fuelling the forthcoming generations of the software industry. The opportunity created by a demand for the implementation of modern replacements for Legacy Systems creates a *pull factor*, which attracts a new younger generation of IT professionals. This is an economic principle known as derived demand; the industry grows in size, because of the momentum of intake of new workers caused by the availability of employment. In the long term, a greater demand for developers will bring about a greater capacity, and more workers will join the industry to lessen the shortage.

Owing to this principle still being in its infancy, there is little evidence concerning precisely how effective the replacement of Legacy Systems would be as a means of *creating* more IT jobs and attracting new developers. However, this lack of data does not prevent the following conclusion from being drawn: simply *enlarging* the sector will not quench the current shortage. In other words, the supply of software developers must increase *relative* to the demand, instead of commensurately to it.

The Role of AI in Software Development

Though “Automatic Programming” is not a new idea[33], it remains a prevalent one in terms of alleviating some of the workload on software developers. If it *were* possible to simply provide an omniscient supercomputer with a brief and have it autonomously construct a corresponding programme, this *could* potentially satisfy much of the remaining demand for coded business systems.

In the present day, a fully-autonomous system to write entire applications is non-existent. Such a system would perform what is called “Program Synthesis”[28]. There are, however, currently three main products[34] which can sit *alongside* programmers and provide them with Artificially-Intelligent suggestions for the computer code they are writing. Microsoft’s “IntelliCode” and “DeepDev”, as well as GitHub’s “Copilot”, are amongst the first of their kind in assistive and predictive programming tools.

However - the potential for a *monopoly* structure is created by the fact that one single organisation owns all of this intellectual property; Microsoft dominates the market for AI Software Development Tools and is therefore able to dictate the long term price for such products, by creating a knowledge barrier to entry and, creaming off super-normal profits. Having purchased GitHub[35] – the incomparably colossal online database of open-source code – Microsoft has afforded itself the ability to further train and refine its AI programming tools, in a way which is now inaccessible to any other company. If the multinational continues to exert this monopoly power, *and* can facilitate sufficient development of the tools, then such services may be able to take on repetitive programming tasks and ultimately even replace humans altogether, for certain simplistic and systematic software creation processes[36].

Questions have been raised about the *morality* of using the code of thousands of unsuspecting programmers[37] to train a vast AI which in many ways seems to just regurgitate pieces from the programs it has seen before - thereby arguably possessing little more talent than the *copy* and *paste* buttons found on most computers since the 1980’s[38]. However, every one of the 400,000[37] software projects uploaded to the database from which the AI was trained, was provided with the uploader knowing that anybody on the internet would be able to see and use their code anyway. In fact, this can be considered an example of a *positive externality* in production; the action of one developer sharing their code has the potential to benefit society to a far greater extent than the benefit to the individual programmer. It can be justified as creating a social “good”. All this is to say: whilst there is some contention over the morality of the system, it does not warrant the removal of this section from the consideration of solutions.

A more significant concern - with *Copilot* in particular - has been about the security of the automatically-generated code. Since the Copilot service was trained with over 14 terabytes of code[37] written by 56 million users, it is an inevitability that a proportion thereof will be to some extent malicious, or, susceptible to attacks such as *buffer overflows* and *SQL injection*. A group of researchers from New York University claims that as much as *40%* of the output generated by Copilot is "vulnerable" to such security inadequacies[39]. To illustrate why such problems exist, take the following illustration of rapid change into account:

> *When storing passwords in a database (e.g. for the back end of a website), the recommended practice for quite some time, was to simply "hash" (cryptographically summarise) the password using an algorithm such as MD5. A few years later, precomputed hash tables of sufficient size to render MD5 insecure became widespread, and the advice was to prepend a small string called a "salt" onto the password prior to hashing both. Shortly after that, even this was considered unsatisfactorily secure, and nowadays best practice is to pass the string through several layers of hashing algorithms, usually performed by a specialised library.*

In other words, security recommendations change rather quickly - but all the old, less secure code *remains* in the training set, so the AI continues to learn from it and suggest potentially vulnerable script to software developers, who, if not fully aware of what is on their computer screen, may permit the code to enter a production environment and ultimately render entire websites or companies susceptible to being "hacked".

AI is not at present capable of writing entire pieces of software for a given brief provided by a human. The predominant reason for this is because AI lacks "Domain Knowledge"[33]; unlike a human, it is not aware of the intricacies and holistic requirements of a system's environment. Where it comes into its own however, is in *assisting* or even *fully automating*, the repeatable, predictable, and otherwise systematic parts of the software-development process. Therefore, as a solution to quenching wider society's thirst for a large technical workforce, Artificial Intelligence does not independently have the ability to solve this problem, but does increase the productivity of individual developers and their teams.

Conclusions

In the short term, the most effective *immediate* means of increasing the UK's capacity for Software Development, will be to promulgate the use of technical solutions such as those discussed in this paper: Low- and No-Code Development, and AI-based assistive tools.

One clear trend is that AI development tools will *continue* to improve in their ability to assist and automate; each developer will be able to produce more output, so fewer are needed to fulfil the market demand. *Beta Testing* for GitHub's Copilot only started in 2021[39], but as more statistics become available herefor, a wider audience of programmers are likely to adopt these tools as an integral component of their workflow. Moreover, the integration *of* AI *into* [Low and No]-Code development products[28] aids these already powerful and widely-adopted systems in becoming all the more versatile. The fact that LCD and NCD products are especially effective at API-based data manipulation programming is also promising, due to the primacy of this *form* of software development in many modern businesses. In fact, a *Tech Republic* survey from 2021 alleges that almost half the IT firms in Britain already use No- or Low-Code Development.

It is obvious, however, that simply addressing the efficiency and productivity of the current software development workforce, will not suffice to bring about a dependable supply of programmers in future generations. After all: a *Copilot*, is not a *Pilot*. To create a lasting effect, the current cohort of students in the education system must be encouraged to engage *with*, and take career-initiating interests *in*, Electronics and Computer Science. The recommendations given by practised and outreaching organisations such as the *Raspberry Pi Foundation*[24] and *Royal Society*[25] must be put into practice, specifically: facilitating "… careers for staff who want to teach [Computer Science] as well as work in another setting.", and placing greater emphasis on interactive teaching methods such as "physical computing" and "project-based learning". The gov.uk Apprenticeships website witnessed a record 12,500 Technology Apprenticeships taken up in 2022 – a 30% increase from the same time last year. Preliminary evidence such as this demonstrates the potential influence of improvements to IT education, and that the outdated view of a degree being a *prerequisite* for IT jobs is no longer putative.

Pupils at this young age *must* have the opportunity to interact with knowledgeable subject specialists – industry staff must be incentivised to teach in schools (conceivably on a part-time basis), or become otherwise involved in the educational system. Kickstarting this scheme may have to be done with financial enticement or prospects of career progression. *Hands-on* experiences such as Apprenticeships must be more fervently advocated too, as these provide an even more direct pathway to careers in software development.

O*ff-shoring* and *increasing salaries* proved to be unsustainable in the long run, due predominantly to the difficulties of remote quality control, and unjustifiability of the higher wages. The principle of UK-based firms generating higher profits from a cheaper short-term workforce, however, does allow them to reinvest this money into training workers in the UK, and it is evident that a certain extent of off-shoring will continue to occur in the UK, for the foreseeable future.

Similarly, the principle of *enlarging* the sector via *derived demand* is not viable: it is true that more workers would *join* the sector because of an increased number of job opportunities, but this would ultimately exacerbate the shortage; *what's needed* is *more* developers per the current number of jobs, not the other way around. In other words, just making "*software developer*" a more *common* occupation (because of *pull factors* such as the replacement of Legacy Systems) is not effective.

In summary, whilst the implementation of AI, LCD, and NCD will continue to provide a much-needed short-term boost, only by targeting the younger generation of students – through recruiting industry-familiar staff and using proactive teaching methods – can the shortage be ultimately solved.

References

#	Source	Date Accessed
1	The Developer Shortage isn’t going away, so it’s time to start thinking differently https://www.zdnet.com/article/the-developer-shortage-isnt-going-away-so-its-time-to-start-thinking-differently/	23rd Dec. 2021
2	Software Developers in demand https://technation.io/news/tech-hiring-at-its-highest-level-for-five-years/	25 Apr. 2022
3	Tech nation jobs and skills report 2020. https://technation.io/news/uk-tech-jobs-growth-data/	2nd Jan. 2022
4	Prof. Christopher Barnatt Lecture presentation on the “Metaverse” https://www.youtube.com/watch?v=shW8MIbNyuM	29th Nov. 2021
5	Tech workers are preparing to quit. Persuading them to stay won’t be easy. https://www.zdnet.com/article/tech-workers-are-preparing-to-quit-persuading-them-stay-wont-be-easy/	23rd Dec. 2021
6	International Informatics Olympiad (IOI) International Leaderboard https://stats.ioinformatics.org/results/2021	26th Nov. 2021
7	UK trade prospects after Brexit https://www.pwc.co.uk/economic-services/ukeo/ukeo-nov16-trade-prospects-after-brexit.pdf	19th Dec. 2021
8	Buyer’s Guide to IT Outsourcing to Ukraine: Companies, Developers, Rates. https://www.daxx.com/blog/outsourcing-ukraine/guide-outsourcing-ukraine	22nd Dec. 2021
9	Global Offshore Developer Rates https://www.daxx.com/blog/development-trends/average-rates-offshore-developers	25th Apr. 2022
10	COVID-19 cause of 1 Million Missing Workers https://www.bbc.co.uk/news/business-60039923	25th Apr. 2022
11	Brexit’s effects on developer numbers in the UK https://www.gamesindustry.biz/articles/2022-02-22-the-uk-games-industry-is-struggling-with-brexit	25th Apr. 2022
12	UK Government statistics regarding the movements out of work for those aged over 50 years since the start of the Coronavirus Pandemic https://www.ons.gov.uk/employmentandlabourmarket/peopleinwork/employmentandemployeetypes/articles/movementsoutofworkforthoseagedover50yearssincethestartofthecoronaviruspandemic/2022-03-14	27th Apr. 2022
13	Communications of the ACM https://cacm.acm.org/magazines/2021/7/253461-the-2021-software-developer-shortage-is-coming/fulltext	2nd Dec. 2021

14	Software Developer Salary Around the World (2021) https://www.daxx.com/blog/development-trends/it-salaries-software-developer-trends#highest-paying-software-engineers	22nd Dec. 2021
15	Jan Tinbergen's legacy for economic networks: from the gravity model to quantum statistics https://arxiv.org/abs/1304.3252	21st Jan. 2022
16	What to do about a Labour Crunch https://www.economist.com/leaders/2021/05/22/what-to-do-about-a-labour-crunch	23rd Dec. 2021
17	Jones, R.W. and Weder, R. (2018). 200 Years of Ricardian Trade Theory: Challenges of Globalization	28th Dec. 2021
18	Charting A Course For The Future: How London's Startup Scene Can Survive And Thrive In The Age Of Brexit (Institute For Public Policy Research) https://www.ippr.org/files/2018-05/charting-a-course-for-the-future.pdf	21st Dec. 2021
19	Krugman, P.R., Wells, R. and Kathryn Jo Graddy (2014). Essentials of economics. New York, Ny: Worth Publishers	28th Dec. 2021
20	Contracts in Offshore Software Development (JSTOR) https://www.jstor.org/stable/4133977	26th Nov. 2021
21	India Software Market Revenues Forecast to Surpass US$8.2 Billion by End of Year 2021 https://www.idc.com/getdoc.jsp?containerId=prAP48517221	5th Mar. 2022
22	The Labour Market https://www.economicsonline.co.uk/competitive_markets/the_labour_market.html	23rd Dec. 2021
23	UK demand for digital skills https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/807830/No_Longer_Optional_Employer_Demand_for_Digital_Skills.pdf	5th Mar. 2022
24	Teaching Programming in Schools (RasPi Fndtn) https://www.raspberrypi.org/app/uploads/2021/11/Teaching-programming-in-schools-pedagogy-review-Raspberry-Pi-Foundation.pdf	5th Mar. 2022
25	Royal Society "*After the Reboot*" Report https://royalsociety.org/~/media/events/2018/11/computing-education-1-year-on/after-the-reboot-report.pdf	7th Jan. 2022
26	Royal Society: Shut down or Restart? https://royalsociety.org/topics-policy/projects/computing-in-schools/report/	5th Mar. 2022
27	UK Computer Science Degree Applicant Numbers https://www.bcs.org/articles-opinion-and-research/record-numbers-have-applied-for-uk-computer-science-degrees-this-year/	20th Apr. 2022

28	The Rise of No/Low Code Software Development https://www.researchgate.net/publication/342951159_The_Rise_of_NoLow_Code_Software_Development-No_Experience_Needed/fulltext/5f0efaaf299bf1e548b70e2b/The-Rise-of-No-Low-Code-Software-Development-No-Experience-Needed.pdf	5th Mar. 2022
29	Characteristics and Challenges of Low-Code Development https://arxiv.org/pdf/2107.07482.pdf	5th Mar. 2022
30	Microsoft PowerApps^{TM} Low/No Code Platform https://powerapps.microsoft.com/en-us/low-code-platform/	5th Mar. 2022
31	How Do Professionals Perceive Legacy Systems and Software Modernization? https://servicifi.files.wordpress.com/2010/06/icse.pdf	4th Mar. 2022
32	Statistical Data and Programming Languages Analysis https://aip.scitation.org/doi/pdf/10.1063/5.0041762	4th Mar. 2022
33	Automatic Programming: myths and prospects https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.3917&rep=rep1&type=pdf	29th Feb. 2022
34	AI Software Development Tools https://www.infoworld.com/article/3623773/ai-gives-software-development-tools-a-boost.html	29th Feb. 2022
35	Microsoft acquires GitHub https://news.microsoft.com/announcement/microsoft-acquires-github/	3rd Mar. 2022
36	AI is helping to make better software https://www2.deloitte.com/content/dam/insights/us/articles/6342_S4S-AI-in-software/DI_S4S%20AI%20in%20software.pdf	29th Feb. 2022
37	GitHub: 400000 GitHub repositories, 1 billion files, 14 terabytes of code https://hoffa.medium.com/400-000-github-repositories-1-billion-files-14-terabytes-of-code-spaces-or-tabs-7cfe0b5dd7fd	2nd Mar. 2022
38	How Copy and Paste was invented https://www.bbc.co.uk/news/world-us-canada-51567695	29th Feb. 2022
39	Assessing the Security of [GitHub Copilot]'s Code Contributions https://arxiv.org/pdf/2108.09293.pdf	4th Mar. 2022

Word Count: 5389 excluding References and Appendices

(End of Dissertation Document)