Vitruvius's Continued Impact on Software Engineering Though the Concepts in 'De Architectura'

Nicholas Vail-Stein

Abstract: Objective: The Ancient Roman author Vitruvius and his work *De Architectura* continue to hold their value, especially in the modern field of software engineering. The goal of this paper is to highlight Vitruvius's educational philosophies and how many of his examples translate to software engineering and other modern fields. Results: His works have a few shortcomings, but overall tend to do a very good job translating to software engineering and give good insight into the educational process in doing so. Conclusion: There is much for a software engineer to learn from the works of Vitruvius for their own educational process.

***Index Terms*—Architect, Coding, Computer Science, Field, Program, Software Engineering**

Introduction

The ancient Roman times give a very good insight to human intelligence and cooperation. One man of such rational thought was Marcus Vitruvius Pollio, an engineer and author, whose works capture the basis of Roman technological and philosophical advancement. He looked past the notion of mastering a single study and instead implied that knowledge of all relevant studies would lead to much greater creations that could incorporate many ideas. His only remaining work, *De Architectura*, or *Ten Books on Architecture*, is a detailed example of his philosophy on what makes an architect great [1]. This story has much greater implications than just architecture, as its examples play into many structured and varied fields in today's world, especially in fields such as software engineering.

Architecture as an example complex fields

Vitruvius believed that the studies of an architect should be much greater than that of just architecture. He believed that there were many fields that would assist architects as leaders, builders, and people and that these improvements would relate back to their architecture work. This philosophy didn't only focus on just strengthening the architects abilities either, Vitruvius also believed that It was extremely important for an architect to understand the context of their project such as the history or technology that would have an impact on it. To add even more Vitruvius even believed a great architect would be versed in such far off fields like the arts. He greatly believed that all fields interacted in many ways and he gave many examples of times in architecture where being knowledgeable could end up making a project into something that was truly great.

Fitting this template to software engineering

Vitruvius believed his model was very good at capturing the entire educational process for not only architecture, but for all scholars. This can be seen when he remarks "I promise and expect that in these volumes I shall undoubtedly show myself of very considerable importance not only to builders but also to all scholars" [1]. His devotion to not only create a work that would assist architects but to be a general set of guidelines that are adaptable to many high level fields is why this model fits so well to software engineering. Aside from his examples almost everything he notes can be almost directly related to software engineering.

The relation between all studies

The improvement of self that modern schooling focuses a lot of time on is in many ways related to what

Vitruvius meant when he stated “As for philosophy, it makes an architect high-minded and not self-assuming, but rather renders him courteous, just, and honest without avariciousness” [1]. Vitruvius’s idea that philosophy gives people virtue and that virtue creates better workers resonates with today's educational system. Many schools are attempting to create very safe and open work spaces for students from a young age and encourage everyone to be honest in everything they do. Punishments in schools are also strictly enforced for those who do not hold virtue to a high regard. In the workforce being honest and accepting repercussions for mistakes is often more honorable than trying to push off issues to others or hide them.

In fitting software engineering to Vitruvius’s model we need to note some of the other studies that a software engineer must understand. There are many courses that a Software Engineer that will take besides the courses in the craft, and even some specialized courses teach more about workflow than just project design. Every software engineer will also have gone through many general education or ‘core’ courses before even having chosen a college major. These courses give a software engineer both a strong background in the courses content, but also teach good learning and personal habits.

Aside from gaining a strong core understanding of subjects software engineers often have to work on projects that are very close to another field and require a deep understanding in that fields work. This is especially true when the software is designed to be used as a resource in something such as the medical field where “Software engineers are needed for the development and implementation of algorithms used in a variety of ways to aid clinicians in patient care. Virtual reality systems are valuable in diagnostics and teaching” [20]. This is just one example of the many fields that software engineers are needed in to continue industrial automation.

History

One of the major studies that Vitruvius stressed the importance of was history. At first glance history and architecture have very little to do with each other but Vitruvius gave a great example about how history has interacted with history when he wrote; “For instance, suppose him (the architect) to set up the marble statues of women in long robes, called Caryatides, to take the place of columns… Hence the architects of the time designed for public buildings statues of these women, placed so as to carry a load, in order that the sin and the punishment of the people of Caryae might be known and handed down even to posterity" [1] This is referring to the statues of women as pillars, holding the weight of the roof, or symbolically a burden. The women in question are those of a certain Greek war who symbolically still carry the burden of what their state had done to the Greeks. Vitruvius points out how such a task would have to be explained to those in very precise terms to get approval from those funding the building to carry out such a feat of architecture.

History has had some very large impacts on the software engineering world, especially within itself. Oftentimes elements of a program needs to be easily recognized for simplicity. These elements, or Icons, do not change over very large courses of time and knowing of and implementing these elements can be critical for the use of a product. “Icons are also a language, of the visual sort. In the early days of the graphical user interface, the desktop metaphor was heavily relied upon to communicate new ideas about computing. This was essential to its success, as the nascent GUI made a newbie out of everyone” [18]. This notion that icons, and what they represent can make it easier for users to navigate software has been largely utilised since its conception. Icons are found in extremely large quantities from hardware icons such as the power icon, to the floppy disk being a popular way to represent the option to save.

A floppy disk save icon

There is also a lot of coding language and code bases that are now considered to be ‘legacy’ or outdated. That does not mean they are necessarily useless to learn because there will always be jobs that require knowledge in these coding languages and the more primitive coding languages often times assist students in theoretical concepts of more advanced programming languages. “MIT uses an excellent book which teaches the mathematical model of how computer languages and algorithms are structured. It's very theoretical and complete. Any student taking that course will gain a complete understanding of how computer programming works, and be able to implement the entire process from the source code to machine code generation” [4]. An educational process like described allows students to grasp the fundamentals of coding to ease into more foreign and complex topics from

the ground up. This can give students a better understanding of the underlying structure in the code they use.

Aside from these aspects there is also a lot that software engineers can take in from history regardless of the history of software itself such as in the case Vitruvius mentioned. This is very common in video games which often times use historic events to invoke a sense of realism within the player. The limits of the realism in games is often attributed to both hardware limitations and believability. This is why some better examples of video games with underlying historical significance are viewed such that “Game scholars and historians alike argue that games can teach about history because, as simulations, they allow participants to play within the bounds of known history" [15] Without a great amount of knowledge in history or without enough to work alongside history specialists a software engineer would be absolutely clueless in creating games that invoke such a sense of realism or immerse people in a past time.

The Library of Alexandria as portrayed in Assassins Creed Origins

Global understanding

Vitruvius’s view of history was not just viewing events of historical significance to include homage to past events. It could be looked at as having a good sense of cultural understanding to appeal to, or avoid repulsing, those of that culture. In his earlier example where the Caryatides were depicted as holding up a building could never be used in their homeland of Caryae because it would be culturally inappropriate to depict their own people in that way. This seems obvious but if for example using Caryatid women as pillars became very popular and widespread than those who did not know the significance might unfortunately make plans for a culturally inappropriate building plan.

In Software engineering there is a whole new level to appealing to cultures as software is very often times multicultural in nature. This can be a double edged blade for many software engineers as they potentially have a much larger audience but also have to be very careful not to be offensive to any culture. One simple slip up can tarnish not only a software engineer but the company they are working for. In one case such as this the Patriots name was tarnished through an automated tweet congratulating a twitter member with a racist username prompting the company to respond. “On Thursday, the Patriots deleted the offending tweet and – via Twitter – issued the following apology: ‘We apologize for the regrettable tweet that went out from our account. Our filtering system failed & we will be more vigilant in the future’” [28]. Software engineers must always be on the lookout to prevent culturally inappropriate content from being attributed to themselves and their employers to prevent scenarios such as this.

Despite shortcomings appealing to this global audience can be extremely successful if approached in a proper way. An example of a company appealing to a global market through understanding of the differing cultures and politics is Google Maps. An article by The Economist looked into how Google Maps worked towards this which stated; “This global audience has prompted Google to produce several local versions of maps for the 200 countries it covers. As well as being written in different languages, the maps also conform to local laws" [13]. The local laws that Google was conforming to where border regulations, in which they decided to display differing disputed borders dependant on what country the map was being accessed from. This shows the attention that companies must put into their products that aren't functional, just for the purpose of being aligned to a culture's beliefs and politics.

Localising software for differing markets is not the only way in which it can have global appeal. Other software may seek to be beyond culture and politics. A good example of creating a system that can appeal to many cultures without having to localise anything would be Nike, who has created a customised shoe service known as NIKEiD, which according to Forbes is “... an online service that allows customers to create their own gear by customizing color, design and performance features to get their gear exactly as they want it” [12]. This shoe customization service allows customers to design their own shoes to fit their tastes, and in turn, culture.

Philosophy

Another seemingly unrelated field that Vitruvius recommended be looked into by the architect is Philosophy. He believed that it would benefit everyone positively by putting them in the correct mindset to work well. The knowledge of philosophy would, according to Vitruvius, “Let him not be grasping nor have his mind preoccupied with the idea of receiving perquisites, but let him with dignity keep up his position by cherishing a good reputation" [1]. Vitruvius believes that studying philosophy

worked to content people so that they could focus less on other aspects of life such as making money and really hone in on their craft. At this time philosophy and deep thought also took place of the sciences, in which physics is a very important aspect for any architect to understand.

Philosophy at its core is very important to a profession such as software engineering. The ability to critically think is crucial to the problem solving nature of the field and with a background in philosophy a software engineer may gain an edge in their field. According to McNeese State University “Philosophy teaches a number of skills that are valuable in a variety of professions. The hallmark of philosophy education is critical thinking and inductive reasoning.” [29] Critical thinking and inductive reasoning are also very important parts of software engineering and getting another perspective on it may make a big difference.

Virtue is also important in the world of software engineering, especially when people's privacy and wellbeing are involved. It is not uncommon that a company will face intense backlash over poor ethics. In recent news a huge social media company, Facebook, has taken extreme losses over poor privacy strategies that ended in a data breach of user information that many claim should not have been stored in the first place. A letter from the European Consumer Organization to facebook states that “It also seems that Facebook did not adequately protect its users’ data. Firstly, by carelessly giving access to third parties to its users’ data. Secondly, by failing to take all the necessary measures to correct the situation once the abuse came to its knowledge, back in 2015" [24]. Complains by a consumer organization to a company can heavily tarnish their reputation. The huge backlash on facebook due to their actions is proof that consumers in a modern world still value the virtue of those who build the products they use.

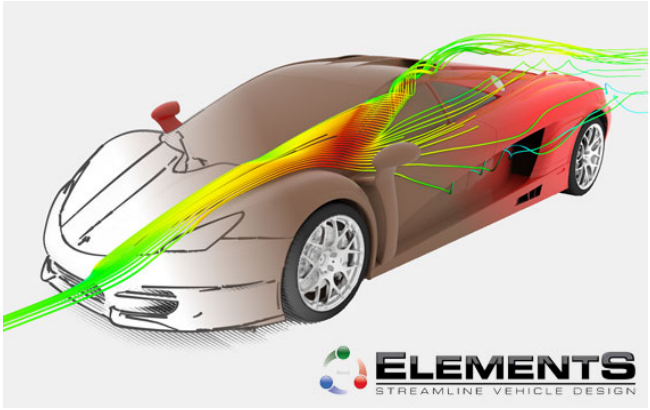
Another positive benefit of philosophy is that it might help people overcome or prevent mood disorders like depression or anxiety. While not all mood disorders can be overcome some cases are caused subconsciously when people begin to think negatively about certain aspects of their lives. Philosophy aims to bring back logical thought to overcome pessimism in cases like this, which could result in enhanced mood. “Additionally, philosophy demonstrates that problems often have multiple solutions, and teaches its students to approach problems from a number of different perspectives” [29] Learning to solve real life issues from different perspectives may be another useful tool in overcoming adversity being mentally healthy.

Physics

Vitruvius's definition of philosophy also expands to the field now known as Physics. Vitruvius stated that “Furthermore philosophy treats of physics where a more careful knowledge is required because of the problems which come under this head are numerous and of very different kinds..." [1]. In the excerpt Vitruvius is mentioning that the greek physics have a distinct place in philosophy and must be approached very carefully by an architect. Now the concepts of physics and philosophy are distinct fields both of which having great importance in many modern studies.

Having a vast knowledge of physics is a very important aspect of software engineering. There are many uses for applied physics in today’s technological world. The easiest place to find this is in video games which go to extreme measures to simulate physics and make their game feel more believable. In a book by Ian Millington on the topic of Video Game Physics engines he notes that “Physics is a hot topic in computer games. No self-respecting action game can get by without a good physics engine, and the trend is rapidly spreading to other genres, including strategy games and puzzles" [10]. This book goes into great details on how true to life physics can be implemented into a virtual world and manipulated to make gameplay appealing. Just how an architect needs to understand the physics behind the building they make in a real world a game designer must understand the physics of the game engine they use to create believable scenarios.

There is also more industrial uses for physics in software engineering. Very often products must undergo physics tests that can not take place in the real world and must be accurately replicated by a computer simulation. One example of this is automotive aerodynamic testing software which aims to test how well a car’s design is without producing costly parts and using wind tunnels. “ELEMENTS offers a cost-effective solution that combines the best of vehicle design simulation practices, all within an easy to use GUI that requires no third party software” [28]. This type of physics based software is extremely prevalent in the modern world as it can bring costs down substantially for car manufacturers and is an efficient way to test new designs and push industry standards.



The elements aerodynamic simulation graphic

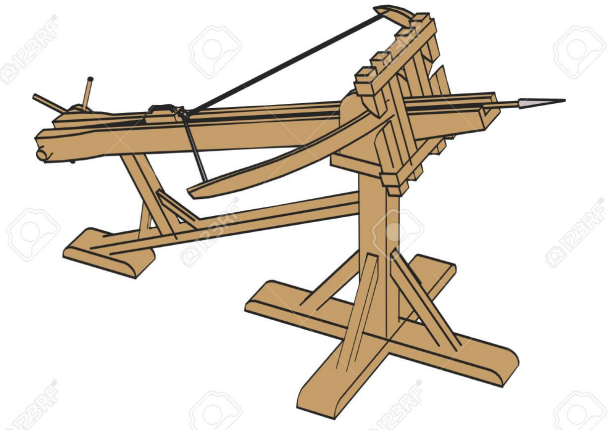
Music

Often times Vitruvius used very specific examples of when one field might overlap another to place perspective on what type of work an architect would end up doing. This can be seen when Vitruvius gives an example in music, and more specifically in theatres. When talking about the technology that goes into a theatre, Vitruvius mentions how “there are the bronze vessels which are placed in niches under the seats in accordance with the musical intervals on mathematical principles” [1]. Without a knowledge in the mathematics of music an architect would have no clue how to place the seats of a theatre to get the proper resonation of the vessels. Through this obscurely specific example Vitruvius points out how much studying an engineer really has to do to become successful in their craft.

Software engineers often have times when their field and another overlap, and the need for professionals with experience in both fields is necessary for the success of a product. This is extremely prevalent in the music industry where there is a rapid growth in software driven products for music production, consumption, and distribution. Most consumption of music is now done online, just how it was done in theatres in ancient roman times, which means that software engineers need to be the ones in charge of most of the industry. In a blog by Andrew Dubber he states that “You’ll read a lot in the news about YouTube, Spotify, Apple, Soundcloud, Microsoft, Google, Samsung, Nokia and other tech giants ‘negotiating with the music industry’ over one thing or another. But that’s not actually what’s happening. Those guys ARE the music industry" [9]. Software engineers who have a passion and a knowledge in music can now make it partly into the music industry through the creation of their software and it doesn't end at distribution services either. There are many software resources created by software engineers with a proficiency in music aimed at music production.

Math

Math was a topic not directly covered by Vitruvius in his examples, but rather an overarching theme in many of these examples. In the section of his work focused on music he heavily referred to the mathematical theory of music as aforementioned and when he stated “Music, also, the architect ought to understand so that he may have knowledge of the canonical and mathematical theory, and besides be able to tune ballistae, catapultae, and scorpiones to the proper key" [1] In this Vitruvius uses music and tension to explain mathematical and physical properties. The way a ballista and catapults are tuned is by turning two cables that hold tension to pull down an arm. To fire the projectile correctly both ropes must be at near equal tensions, and to achieve this both strings may be plucked to hear what sound frequency they make.



A ballista with two tensioned ropes.

It seems odd that this would have anything to do with mathematics, and for the most part the sound frequencies don’t, but the reasoning behind it does. For all of the energy of a ballista to go into the projectile all of the forces must be balanced. As Vitruvius notes about this, “For the arms thrust through those stretched strings must, on being let go, strike their blow together at the same moment; but if they are not in unison, they will prevent the course of projectiles from being straight" [1]. Without a knowledge of physics and mathematics in this case it would be impossible to get a consistent shot with these style weapons.

Software engineering also runs into many different types of mathematics. Some of the most common mathematics that software engineering has to deal with is the concept of Big O. “In computer science, we often encounter algorithms to analyze. The analysis most often asks for the running time of the algorithm, but it can also be memory consumption, stack depth, and other resources, often expressed as a function of the input size" [6]. This

concept of Big O and a software engineers understanding of it are crucial to any type of database as they are used to calculate the slowest potential, fastest potential, and average running time of a program.

There are still countless forms of mathematics that a software engineer would almost certainly need to know, all relating to specific jobs that they may run into. Having a deep fundamental understanding of mathematics so that one can pick up on new forms of it is possibly the best a software engineer can do to keep themselves from getting caught up in math.

THE WORK AND THE THEORY

Vitruvius believed that no craft one only comprised of one task. He made an argument that “It appears, then, that Pytheos made a mistake by not observing that the arts are each composed of two things, the actual work and the theory of it" [1]. This means that an architect would not only need to understand what a building should look like but also the building techniques that will need to be used to craft that building. This can assist in keeping buildings from being impossible to build or up to improper standards.

For software engineering the difference is almost more clear than with architecture. According to Drexel “Computer software engineers apply the principles and techniques of computer science, engineering, and mathematical analysis to the design, development, testing, and evaluation of the software and the systems that enable computers to perform their many applications" [3]. The theory side of software engineering is the knowledge that allows them to develop, test, and evaluate computer systems. It is seen through topics such as data structures, which aim to simplify the process of creating complex software by breaking up their components. It also involves the human interactions of obtaining and working towards accomplishing the requirements of a project.

The actual work of software engineering is the coding, which is writing logical computer statements that can perform computer processes. This is what allows a computer to perform the functions and so is how a software engineer can actually build a program. There are some theoretical concepts in coding but typically those should be covered by a computer scientist who specialises in advanced coding theories.

STANDARDS

When talking about what Vitruvius referred to as medicine, or what in modern times would be referred to as building codes, he shows the concerns of what may happen if an architect’s product is built incorrectly. What would now be known as some sort of legal repercussion he referred to as “disputed points for the householders to settle after the works are finished" [1]. Vitruvius understood the need for regulations on work and how understanding these would help an architect. Most of the standards Vitruvius referred to were based in law, which is a pretty standard practice for things such as structures.

Unlike architecture, this concept of legal standards aren’t fully in place yet for software engineering. There still are many standards that a software engineer needs to follow during almost any project that are more options for customers than legal requirements. This means that there are many standards that a software engineer will need to familiarize themselves with to reach project goals. One small example of varying standards in software is in encryption, which is the process of making data unreadable to all but the one sending and the one receiving it.

The problem with having one set of standards on this would be plentiful. Typically stronger encryption is slower and only useful with delicate data. This can be seen through NIST, a non-regulatory agency who creates security standards that aim to make “... high quality, cost-effective security mechanisms, NIST works closely with a broad stakeholder community to identify areas of need and develop standards and guidelines" [23]. The fact that NIST is still developing standards and is a non-regulatory government agency shows how complex the situation with security is. It is not even required of companies to go through these procedures to secure data, it is just some guidelines for software engineers to consider as an approach to proper security.

HOW MUCH AN ARCHITECT SHOULD KNOW OF OTHER FIELDS

Vitruvius often held to the belief that education of the other fields was far more important to an architect than the simple knowledge of what was proper in architecture. Vitruvius encouraged those without the additional skills learned to not even consider themselves architects. He saw how great the human brain was that so many crafts could be memorized and adapted to one’s own field, but did see a limit to this knowledge, primarily in trying to master the fields in which an architect should study. Vitruvius believed that there were many people who achieved near perfection in their crafts, and that they should be learned from, yet not compared to them. Specialists in each field would far surpass all of those who did not specialize in that field, so for someone only wishing to use that field to better themselves to seek perfection would be a waste of time.

The modern schooling system echos this belief with the concept of ‘core’ curriculum which aims to give all students a well rounded education from a young age. One such movement in education that shows this is the

Common Core which is a part of public education that aims to be “Building on the best of existing state standards, the Common Core State Standards provide clear and consistent learning goals to help prepare students for college, career, and life" [23]. This movement trickles its way into higher education by giving all of those who go on a well rounded background in most fields. Along with this Common Core many colleges have well rounded undergraduate programs for the same reasoning.

Beyond curriculum there is also a social aspect to education. What is meant by this is that even if a student is not being taught through a course about another field they may end up interacting with many people in that field and learn of it in that manner. Vitruvius mentions this as well saying “Astronomers likewise have a common ground for discussion with musicians in the harmony of the stars and musical concords in tetrads and triads of the fourth and the fifth..." [1]. He details how scholars of all fields all have a common ground between their knowledge which works to unite them and share the information of their craft.

This is the basic concept behind activities such as field trips, which allow students to take trips to interact with professionals before they are in the workforce themselves. According to a study covered by The Atlantic “But the study also finds that cultural field trips offer students, and in particular, disadvantaged students, an important opportunity to add measurable depth to their education" [16]. It is also important to note that this is just a single interaction with another field. There are plenty of people who will have learned a good deal about other fields from their peers and teachers outside of a formal educational setting.

Tasks outside of expertise

Vitruvius also mentioned that, despite similarities, there are tasks that should still be left to professionals of their own field. He uses the example that “to physicians and musicians the rhythmical beat of the pulse and its metrical movement” [1]. which he used to explain that “if there is a wound to be healed or a sick man to be saved from danger, the musician will not calm for the business will be appropriate to the physician” [1]. In this example Vitruvius gives a clear example as to why having shared knowledge with another field does not make someone know enough of that field to work in it.

This can also be very relevant in software engineering. According to Monmouth University “[Software engineers] apply the principles of engineering and science to the solution of technological problems that can be solved by the use of software controlled devices" [7]. Just because a software engineer studies the principles of science does not mean that they are in any way prepared to conduct a scientific experiment of any kind. It may also be a misconception that software engineers by default work as business people but there are many tasks that a business person would be more fit to do.

From the other side of this view there is also the idea that computer scientists and software engineers are interchangeable. There are tasks that both a computer scientist and a software engineer may be able to complete but there is also many tasks that separate them. According to Interesting Engineering “When you become a software engineer your goals will include finding and generating software that you can use on computers" [25]. and “computer science is about answering the question ‘What can be automated?’" [25]. This really highlights the fundamental difference that separates the two fields, and that is that software engineers work much more towards creating software according to a client's requirements while a computer scientist is specialized in creating complex automated processes.

Using knowledge to advance a field

Vitruvius understood that there was a limit to the amount of knowledge one can know, but also acknowledged that there was an ability to master one field and and be highly skillful in others. This can be seen when he states “how can an architect, who has to be skilful in many arts, accomplish not merely the feat - in itself a great marvel - of being deficient in none of them, but also that of surpassing all of those artists who have devoted themselves with unremitting industry to single fields?” [1]. he goes on to explain how an architect can become knowledgeable in other fields because they only need to understand the theory of that field and not actually do the work of the field.

It is not uncommon for someone to master one field and be proficient in another and in the case of software engineering this can be revolutionary. The field of software engineering is still in its developmental phase which leaves much room for software engineers to use their talents to innovate. One such innovator Hideo Kojima had a very large passion for both video games and movies and tried to capture this when he became a videogame designer. “Regarded as one of the top game designers in the industry, Hideo Kojima has taken a decidedly cinematic approach with his games, most notably the Metal Gear Solid games that began on the PS one" [2]. This combining of both cinema and Video games has lead to both him and this games he worked on to success in the industry.

Specialization breeds weakness

Vitruvius believed that those who only specialized in only one field would achieve perfection, but he also

believed as noted before that architecture was not a field on its own. In this case the more specialized an architect the worse off they would be such as noted when Vitruvius stated “For, in the midst of all this great variety of subjects, an individual cannot attain to perfection in each because it is scarcely in his power to take in and comprehend the general theories of them” [1]. For Vitruvius being a specialist was an impressive feat, but much less impressive than being well rounded. He believed that specialization would make an architect weak, but also could understand that there was a place for true masters of their craft.

In a modern world it is easy to see that we still employ specialists for jobs outside of one's scope. When a job needs someone of near perfection in a field it is much simpler to hire someone of said perfection than to train someone for the role. This is why Software engineers hire many specialists and consultants to do outside work for them. This is especially prevalent in larger companies. One such company, Ubisoft, seeks to keep their games historically accurate “... keeps a full-time history grad on staff to collect sources and translate documents, it’s about building a world consistent with, among other time periods, 16th-century Italy" [26]. Vitruvius definitely pointed out the importance of historical understanding but if a software engineer was to focus so deeply into history they would have to lose focus on their primary studies.

THE ACTUAL USES

There are many potential benefits to using this model in the current world but their actual uses may be a lot more narrow. It is very unlikely someone will use all of the knowledge they acquired when in the field, and even less likely that they will know everything they need by the time they get into their field. This does not mean that the knowledge is useless though, there are still many reasons to have learned even the most seemingly useless information.

The most obvious of these reasons is qualification. Most colleges require that many general education courses be taken, regardless of whether the information in these courses will actually help students in their job field at any point. A students ability to pass these courses can show their willingness to take education seriously. Jobs may also look very seriously into the grade point average of students and according to Iowa State University “Many entry-level job postings require applicants to have at least a 3.0 GPA, and there are about half as many job opportunities available to those who have a GPA below 3.0" [19]. This means that if a student did poorly outside of their major their GPA might drop significantly and may be taken to reflect their ability to work hard or lack of motivation.

Another reason is so that they are better at gathering information. There are many seemingly useless courses that students will go through in education purely to make sure they can adapt to out of the ordinary settings and still pick up on relevant information. This is the basis behind educational tactics such as studying abroad. According to USA Study Abroad “By studying abroad, you will experience new perspectives, learn how to navigate different cultures, work with diverse peers, and communicate in other languages" [21]. Learning these skills may prepare a software engineer for when they are put into similar situations in their own work. This may prepare a software engineer for the workforce because they will often be in foreign environments attempting to obtain product requirements.

WHAT FIELD BENEFITS THE MOST

The concepts Vitruvius layed out to make architects more versatile in their studies may be more relevant to software engineers now than it was to architects then. Due to the increase in transportation and the advent of global communications the world has become much more fast paced. This fast pacing requires that people be more adaptable than ever as environments change constantly. Having these changes in society is not a new development, it is just occurring faster now than it would have been in Vitruvius’s time.

Software engineers are often times affected by these constantly shifting cultures and technologies. One very large change that is common is for programming languages to be created, and others to slowly wither away. According to Techrepublic “Software development is a dynamic field, in which new programming languages, frameworks, and technologies may live and die within a few years” [29]. The amount of programming languages a software engineer may be alive to use will far outpace the amount of times an architect has to change their style or medium. This rapid changing means that if a software engineer has to be extremely adaptable and have a good base in theory rather than work.

SHORTCOMINGS OF THE MODEL

Obviously no two fields are equal so Vitruvius's works will have to fail at some point describe the job of a software engineer. This failure point comes with the status and name of an architect compared to an engineer. The cause of this difference is primarily because the world of architecture is the same world we walk around in while software can only interact with the world. For example a

building is a tangible object, while an app for a phone only interacts with the object, in this case the phone.

When an architect would create a building they would put their name on it, whether physically or figuratively. If that building was sub-par or failed in any way that architects name and status may fail with it. This is especially relevant in modern times as names in architecture are used for legal purposes, such as the case where a Miami pedestrian bridge collapsed killing and injuring dozens of people. “The lawsuit alleged negligence by FIGG Bridge Engineers and Munilla Construction Management, the two firms involved in the construction of the bridge" [5]. The fact that the architects in this scenario can be taken to court over the failure of their product shows that they are responsible for their product. If a design is flawed because of the architects oversight then all involved may be liable.

In software engineering people are, for the most part, not attributed to their works. In this case if something was to happen in most cases a software engineer can not, or will not be tried for the shortcomings of their work. It is almost accepted in software engineering that failures will happen at some points and rather than attribute these to people or companies they are considered as normal. According to Jim Turley, “Some kinds of mistakes are okay. But you are expected to exercise ‘reasonable care’ in writing your code, testing it, and ensuring that it does what it’s supposed to, while not also doing too much that it’s not supposed to" [8]. Rarely is a product that doesn’t do its job given harsh punishment, and typically at worst is just forced to allow for a refund of the product. Only in a few cases are lawsuits filed against software and they typically target the marketing not being representative of the software rather than its actual code.

Even when software makes a negative impact on the physical world it rarely is attributed to the software or those who made it. This is certainly the case in the recent introduction of self driving cars to public roads. There have been two casualty caused by autonomous vehicles and lawmakers are still hesitant on who should be responsible for the incidents. As the Seattle Times reported “In Washington state, as of now, there’s no clear answer. And there really isn’t much of an answer anywhere else either" [11]. It is not unusual for the laws regarding technology to be slow as this concept of software not being tangible and the lack of existing regulations leaves a lot of unregulated area.

There is also quite a few more fields that a software engineer would need to be aware of that Vitruvius does not mention. This is to be expected as despite the similarities of architecture and software engineering they have many differences. Some notable examples of fields not present in Vitruvius’s paper are management and computer science. The reason management cannot be compared to any of the fields Vitruvius mentions is because structure in business was not so much of a field until more recent times. According to Harvard Business Review “Prior to the industrial revolution, of course, there wasn’t much ‘management’ at all – meaning, anyone other than the owner of an enterprise handling tasks such as coordination, planning, controlling, rewarding, and resource allocation" [27]. The concept of management was not as necessary to study without the modern type of industrialization.

As for computer science, the only reason there is no valid comparison to Vitruvius’s examples is because there is no field that shares so many concepts in architecture as computer science and software engineering do. According to the University of Maine “Both [software engineering] and [computer science] study the use of the digital computer as a tool that makes possible much of modern technology and the overlap between the two fields is significant" [17]. There is no field that architecture overlaps with so much and so there would be no way to use this model to describe computer sciences relation to software engineering properly.

Outro

Vitruvius’s model still holds much relevance today through modern complex fields such as software engineering. Much of what Vitruvius proposed has already been put into the modern education system which only gives additional credit to his claims. The lessons that can be taken out of his work may help students of a study focus on broadening their education to bring them further in their field rather than trying to master their field through study of only that course’s materials. Vitruvius was truly a product of his era and at the height of rational thought.

References

[1] V. Pollio, M. H. Morgan, and H. L. Warren, “The ten books on architecture,” Harvard University Press, 1926.

[2] “IGN: Hideo Kojima biography,” IGN, [Online]. Available: https://web.archive.org/web/20071116080116/http://stars.ign.com/objects/963/963391_biography.html.

[3] “About software engineering,” Drexel. [Online]. Available: http://drexel.edu/ece/about/software-engineering/.

[4] I. Larsen“BASIC-256,” BASIC-256 - Why BASIC? [Online]. Available: http://www.basic256.org/whybasic.

[5] J. Barajas, “More lawsuits filed over the Miami bridge collapse,” PBS, 23-Mar-2018. [Online]. Available: https://www.pbs.org/newshour/nation/more-lawsuits-filed-over-the-miami-bridge-collapse.

[6] A. Chattopadhyay, A. Ellinor, I. Koswara, “Big O notation,” Brilliant Math & Science Wiki. [Online]. Available: https://brilliant.org/wiki/big-o-notation/.

[7] “CSSE,” Monmouth University. [Online]. Available: https://www.monmouth.edu/school-of-science/bs-software-engineering.aspx.

[8] “Can you get sued for bad code?,” EEJournal, 29-Apr-2017. [Online]. Available: https://www.eejournal.com/article/20161116-liability/.
[9] A. Dubber, “Technology is the music industry,” Midem, 20-Aug-2014. [Online]. Available: https://blog.midem.com/2014/08/andrex-dubber-technology-music-industry/.
[10] I. Millington, “Game physics engine development,” Morgan Kaufmann Publishers. [Online]. Available: https://epdf.tips/game-physics-engine-development4b496c81c19c8cd5f1afb1efea52665174624.html.
[11] D. Gutman, “Who'll be responsible when self-driving car crashes?,” The Seattle Times, 12-Feb-2018. [Online]. Available: https://www.seattletimes.com/seattle-news/transportation/wholl-be-responsible-when-self-driving-car-crashes/.
[12] M. Harrist, “OracleVoice: supply chain talent shortage: what's an industry to do?,” Forbes, 17-Apr-2018. [Online]. Available: https://www.forbes.com/sites/oracle/2018/04/17/supply-chain-talent-shortage-whats-an-industry-to-do/#38ef0559791d.
[13] “How Google represents disputed borders between countries,” The Economist, 03-Sep-2014. [Online]. Available: https://www.economist.com/blogs/economist-explains/2014/09/economist-explains-1.
[14] “Milestone Adobe CS6 release delivers major innovations for design, web and video pros,” Adobe. [Online]. Available: https://www.adobe.com/aboutadobe/pressroom/pressreleases/201204/042312AdobeCreativeSuite6.html.
[15] A. Shaw “The tyranny of realism: historical accuracy and politics of representation in Assassin’s Creed III,” Loading… The Journal of the Canadian Game Studies Association, [Online]. Available: http://journals.sfu.ca/loading/index.php/loading/article/download/157/185
[16] J. Ryan, “Study: students really do learn stuff on field trips,” The Atlantic, 16-Sep-2013. [Online]. Available: https://www.theatlantic.com/education/archive/2013/09/study-students-really-do-learn-stuff-on-field-trips/279720/.
[17] “Similarities and differences CE/CS,” University of Maine. [Online]. Available: https://ece.umaine.edu/prospective-students/computer-engineering-overview/similarities-and-differences/.
[18] T. Park, “Save icon,” Thomas Park, 22-Mar-2015. [Online]. Available: https://thomaspark.co/2011/04/save-icon/.
[19] “Understanding what employers look for in engineers,” Iowa State University. [Online]. Available: https://www.engineering.iastate.edu/ecs/students/the-employment-process/step-1-understanding-employers-and-the-college-recruiting-process/understanding-what-employers-look-for-in-engineers/.
[20] “What does a software engineer in biomedicine do?,” Top Master's in Healthcare Administration. [Online]. Available: https://www.topmastersinhealthcare.com/faq/what-does-a-software-engineer-in-biomedicine-do/.
[21] “Why study abroad?,” U.S. Department of State, 18-Dec-2017. [Online]. Available: https://studyabroad.state.gov/value-study-abroad/why-study-abroad.
[22] “Why university students need a well-rounded education,” The Globe and Mail, 26-Mar-2017. [Online]. Available: https://www.theglobeandmail.com/news/national/time-to-lead/why-university-students-need-a-well-rounded-education/article4610406/.
[23] “NIST cryptographic standards and guidelines development process,” NIST, March-2016. [Online] Available: https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.7977.*pdf*
[24] “Facebook/ Cambridge Analytica,” BEUC, 27-March-2018. [Online] Available: http://www.beuc.eu/publications/beuc-x-2018-026_facebook_cambridge_analytica.pdf
[25] “Computer science vs. software engineering – how are they different?" Interesting Engineering, 8-February-2018. [Online] Available: https://interestingengineering.com/computer-science-vs-software-engineering-how-are-they-different
[26] M. Osberg, “The Assassin's Creed curriculum: can video games teach us history?,” The Verge, 18-Sep-2014. [Online]. Available: https://www.theverge.com/2014/9/18/6132933/the-assassins-creed-curriculum-can-video-games-teach-us-history.
[27] R. McGrath, “Management’s three eras: a brief history,” Harvard Business Review, 30-July-2018. [Online] Available: https://hbr.org/2014/07/managements-three-eras-a-brief-history
[28] “Elements CFD Software Suite,” Auto Research Center. [Online] Available: http://www.arcindy.com/software.html
[29] “The benefits of studying philosophy,” McNeese State University. [Online] Available: https://www.mcneese.edu/philosophy/the-benefits-of-studying-philosophy