Contents

1. Software development life cycle.

The Software Development Life Cycle

This life cycle **outlines** a careful, engineering approach to the development of software. There are many variations on this cycle, some of which are better than others. The following, however, is fairly typical.

Analysis

This step consists of understanding the problem and developing precise specifications (of the **input, output**, and processing). This step may involve interviewing the intended users of the software in order to better understand their needs.

Top-down Design

Do the overall steps first, then the details. (There are rare occasions when a bottom-up approach or some other method might be used.) Decisions on classes and data structures are made at this point. Typically one asks what kind of data the program will have to handle, what one will use to hold this data, and what functions might be needed to process this data. Charts and diagrams are commonly used. For example, one might make drawings of the classes that will be used. If **procedural** programming is used (that is, no user-defined classes), a structure chart might be used to picture the overall decomposition of the problem into functions. An overall description for each function may be written out, perhaps using the "given, task, return" style recommended elsewhere in these Web pages. Rough pseudocode algorithms for the functions may be written out. (Pseudocode is a loosely-defined combination of English and programming language. It allows one to write the algorithm for a function without worrying about minor details of coding.) One may trace the execution of the algorithms by hand or apply mathematical proof techniques to try to **verify** the correctness of the **algorithms**.

Prototyping

A prototype stage might be included at this point. A prototype is a rough working version that the users can try out. In fact, it has been suggested that if you do not deliberately create a prototype, your program will not be quite what you want, so that you will have to begin again anyway and your first attempt becomes a prototype after all. The prototype may leave out a number of features and may even not try to compute correct answers. Part of the goal is to allow the potential users to see the data input screens and output screens, as well as to get a sense for the overall feel of the program.

Coding

Write the complete program out in C++. Too many beginning programmers try to start with this step. For anything **non-trivial** this is a formula for disaster, or at least for wasting a lot of time! This is the step where one worries about all of the details of C++ programming. If good design work has already been done, the coding stage can be relatively easy.

Testing and Debugging

Test as many different cases as possible. With large software, however, it is probably not possible to test all paths through the software. Test boundary **values** (such as grades of 0 and 100 in a grading program) and invalid data (such as grades of -1 and 101 in the same grading program) in particular. Modular testing is helpful (that is, test and **validate** each function separately, often by writing a test program that simply calls this one function). Stubs can be used for subsidiary functions so that the main function can be tested by itself. (A **stub** is a function that contains no code or something trivial like an output statement to say that the function has been called. A stub may also assign some nonsense values to any answers that the function is supposed to produce. Essentially the stub is a function whose real code has not yet been written.) Another good way to check a program is to include output commands at strategic spots to write out intermediate values. The **debugger** can also assist in finding where something is getting a wrong value. Be sure that you know how to use it.

Maintenance

This involves fixing errors that show up after the software is in use, adding new features, changing to faster algorithms, etc. This is where companies tend to waste a lot of time and money, mostly because the previous steps where not done as well as they should have been.

Documentation

This step coincides with all the other steps; it should not be done last. At each stage of the software development cycle, important information is written down. Roughly speaking, documentation can be divided up as follows:

For users:

- tutorial
- reference manual

For the programming team and maintenance programmers:

- internal documentation (the given, task, return style is recommended)
- external documentation
    - specifications
    - pseudocode algorithms
    - various object-oriented design diagrams
    - structure chart
    - data flow diagram
    - record of testing
    - record of maintenance changes

Note that the software life cycle is not really a sequence of steps. It is a cycle in that at any step you may realize that you need to go back to an earlier step due to an error or incomplete information. Do not expect to go through the steps in order and be done. Expect to have to go back to previous steps, to have to make changes to what you have already done, etc.

Note too that in developing large programs, one person would probably not do all of these steps. The first two might be done by a systems analyst. Coding might be done by a team of programmers, where each programmer is responsible for certain classes and/or functions (hence the need for good documentation even at this point to aid in coordination). There may be a separate testing team and a separate maintenance group. There may also be a program librarian when there are a large number of source code and other files involved, specialized people to write documentation, etc.

Note that software users typically expect more than mere correctness of the programs that they use. They also want the software to be easy to use. This can be handled by developing a clear, consistent user interface, by including code to handle bad input data, etc.

2. Comparison of two programming languages.

Java and assembler

Java- object oriented high level programming language. It’s usage is extremely widespread due to it’s simplicity, clean sintax and quite fast compiling speeds. It’s used in various computer programs, mobile applications and even some servers.

Assembler is extremely different. It’s not object oriented and it’s not a high level programming language. It’s the opposite – it’s a low level programming language used for machines and not software. It’s specifically designed for controlling individual parts of a circuit such as registers or multiplexers. Since the language is low level, it’s not written by using words, it’s coded by using bits and short commands. It’s used in machine programming because it’s extremely fast.

3. Operating systems (e.g. Android vs. iOS).

(Comparison of Operating Systems in terms of: 1. Speed 2. Compatibility

3. Lower Hardware Requirements 4. Search and Organization

5. Safety and Security 6. Interface and Desktop 7. Taskbar/Start menu)

Android and iOS are both operating systems used in mobile devices. However, while android is used in various models such as Samsung, Google, Huawei, Motorola and many more, iOS is limited to Apple’s products. This causes some inconveniences in app development as apps have to be separately created for each OS. One app will most likely work for multiple android versions and as a result in many phone models, however since apple uses iOS, the apps have to specifically modified in order to function.

Windows and Linux –

Windows is a operating system, used in most user computers because its interface is really comfortable to use and get around in. Meanwhile Linux requires knowledge in IT and rarely supports a comfortable and accessible interface. Windows has high compatibility with all apps as it is extremely widespread. The system itself has a huge amount of built-in features such as a fast and efficient file explorer and a quite decent antivirus app. On the other hand, Linux is a lot more barebones. It does not have such built in features, however it is a lot faster and has lower hardware requirements. It is mainly used in servers or supercomputers due to its high speeds. Additionally, Linux is safer and more secure as it works in its own enclosed space. While windows users can connect to the internet and accidentally download viruses through the usage of the internet.

4. The latest methods and practices for protecting against viruses, spyware, and hacking.

Paying/rewarding hackers – elaborate

Antiviruses – scanning your pc regularly

Not visiting suspicious websites

Keeping the operating system up to date as it receives regular security updates

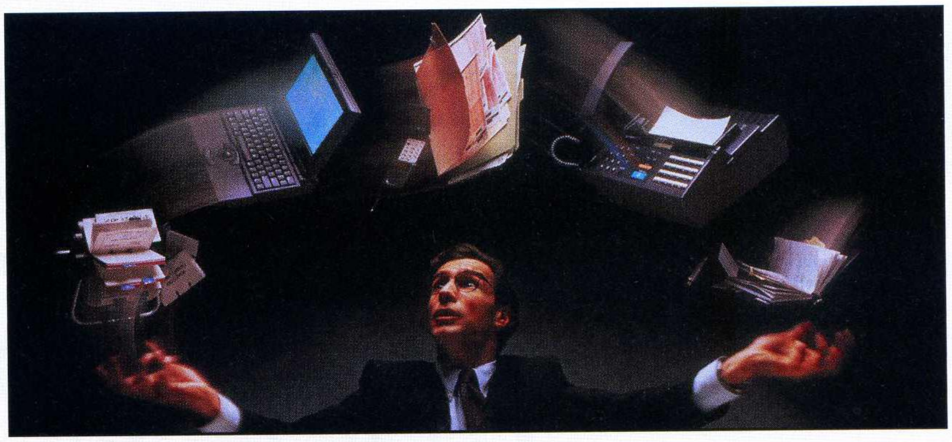
Piracy

5. White hat hackers.

The term "white hat" in Internet slang refers to an ethical computer hacker, or a computer security expert, who specializes in penetration testing and in other testing methodologies that ensures the security of an organization's information systems. Ethical hacking is a term meant to imply a broader category than just penetration testing. Contrasted with black hat, a malicious hacker, the name comes from Western films, where heroic and antagonistic cowboys might traditionally wear a white and a black hat respectively. While a white hat hacker hacks under good intentions with permission, and a black hat hacker has malicious intent, there is a third kind known as a grey hat hacker who hacks with good intentions without permission.

6. Employment trends in the IT sector.

Giganomics: And what don't you do for a living? by Judith Woods



If you ever meet John Lees at a party, he might talk about being a career coach, mention that he writes books on business or drop in the fact that he's also a part-time Anglican priest. 'I've got a classic portfolio career,' says Lees, 50. 'The advantage is that by working for a variety of employers, no one has complete power over you to switch work on or off. The drawback is that I have an immensely complicated diary.'

These days, many of us are juggling one-off projects, short-term contracts and assorted consultancies in a bid to survive. Job security used to be a given. But growing numbers of professionals are reinventing themselves by setting up as portfolio workers in a new employment phenomenon dubbed giganomics. Instead of jobs for life, they rely on a series of 'gigs'.

Former *Vanity Fair* and *New Yorker* editor Tina Brown, who coined the term, writes: 'No one I know has a job any more. They've got gigs: a bunch of free-floating projects, consultancies and bits and pieces.'

Tina Brown paints a bleak picture of freelancers' lives, burdened with all the 'anxieties, uncertainties and indignities of gig work', grafting three times as hard for the same money as a salaried employee, without any of the benefits, such as sick and holiday pay or a pension.

Nick, 37, a graphic designer based in London, can attest to the stress felt by portfolio workers. 'I was made redundant two years ago and went freelance,' he says. 'I hated it, because I am terrible at selling myself and I'm not laid-back enough to live with the insecurity of not knowing where I'll be in six months. I managed OK, and I earned as much as I had done previously, but there was a price to pay in terms of sleepless nights.'

Suzy Walton, a former senior civil servant and mother of four, with a background in central government, including the Ministry of Defence, has taken up a series of non-executive directorships. A portfolio career has proved a lucrative alternative to corporate life.

'I sit on the boards of a military organisation Combat Stress, which looks after veterans with post-traumatic stress disorder, the Internet Watch Foundation and Birmingham Children's Hospital, and a few others,' says Walton, 45.

Walton admits that none of these roles generates a substantive salary on its own – a FTSE 250 company might pay about £30,000 a year to a non-executive board member – but when combined, her directorships provide a good income. Just as importantly, she enjoys the challenges. 'It's hard to keep up to speed with the issues in each, but I enjoy doing that. A portfolio career isn't for the fainthearted; there's a zero-tolerance attitude to being late or missing a commitment. But it's a fantastic lifestyle.'

Anyone with this pick-and-mix approach to work needs to be excellent at time management. The upside is the freedom to pick and choose work, and to do it at a time that suits. Cary Cooper, Professor of Organisational Psychology and Health at Lancaster University, says it's a classic swings-and-roundabouts scenario. 'The good news is that you're supposed to have control over what work you do. The bad news is that you feel you can't say no to anything,' he says. 'You should also be able to have a better work–life balance. But the people who employ you expect you to be on call whenever they want you.'

The creative industries such as advertising, graphic design and the media already rely heavily on freelancers, as does IT. Many more companies will need portfolio workers in future. 'There's going to be much more multiple part-time working,' says Professor Cooper. 'Organisations are getting rid of staff, but they will buy back some of them on a portfolio basis.'

7. A piece of news in IT.

Cybertruck

8. Application of virtual reality in at least 3 fields.

Education – Helps students experience historical events or visualize subjects such as math, physics or chemistry without the dangers of some substances, geography – allowing the users to see famous world attractions without leaving their homes or schools.

Leisure – There are a plethora of video games, created for virtual reality. These games are not only fun, but also encourage the players to exercise. For example the video game "Beat saber" is a really fun way to exercise while listening to music and having fun. Some allow players to experience what it would feel like to have the ability to fly.

Art – Some virtual reality devices come attached with controllers. That detect hand movements. Some programs allow the user to draw 3d paintings or sculptures in the virtual world. The advantages are that the person can spread their creativity, share it with friends and correct mistakes, what wouldn't be possible in the real world.

9. The risks of artificial intelligence.

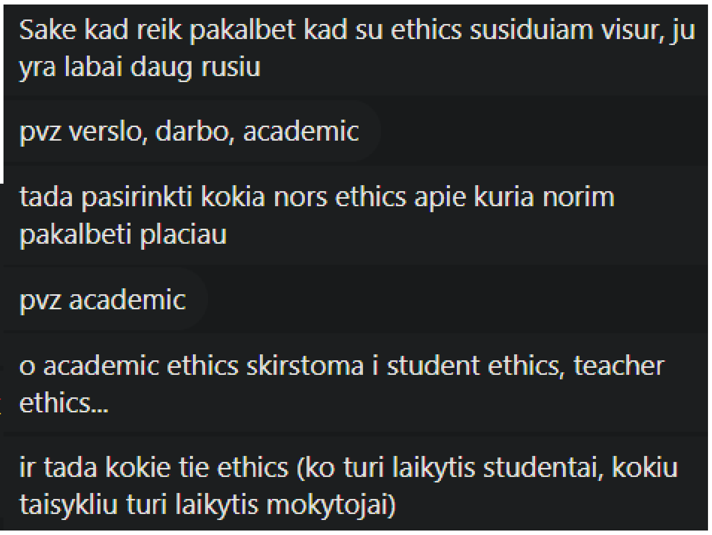
May become smarter than us

Processes a lot faster than humans

If given too much information, may lash out or start making wrong and unethical decisions

May turn against humanity

10. Ethics (business/ academic,etc).



Sake kad reik pakalbet kad su ethics susiduiam visur, ju yra labai daug rusiu

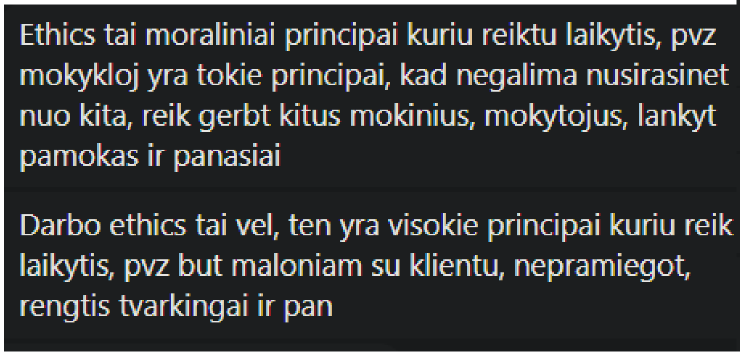
pvz verslo, darbo, academic

tada pasirinkti kokia nors ethics apie kuria norim pakalbeti placiau

pvz academic

o academic ethics skirstoma i student ethics, teacher ethics...

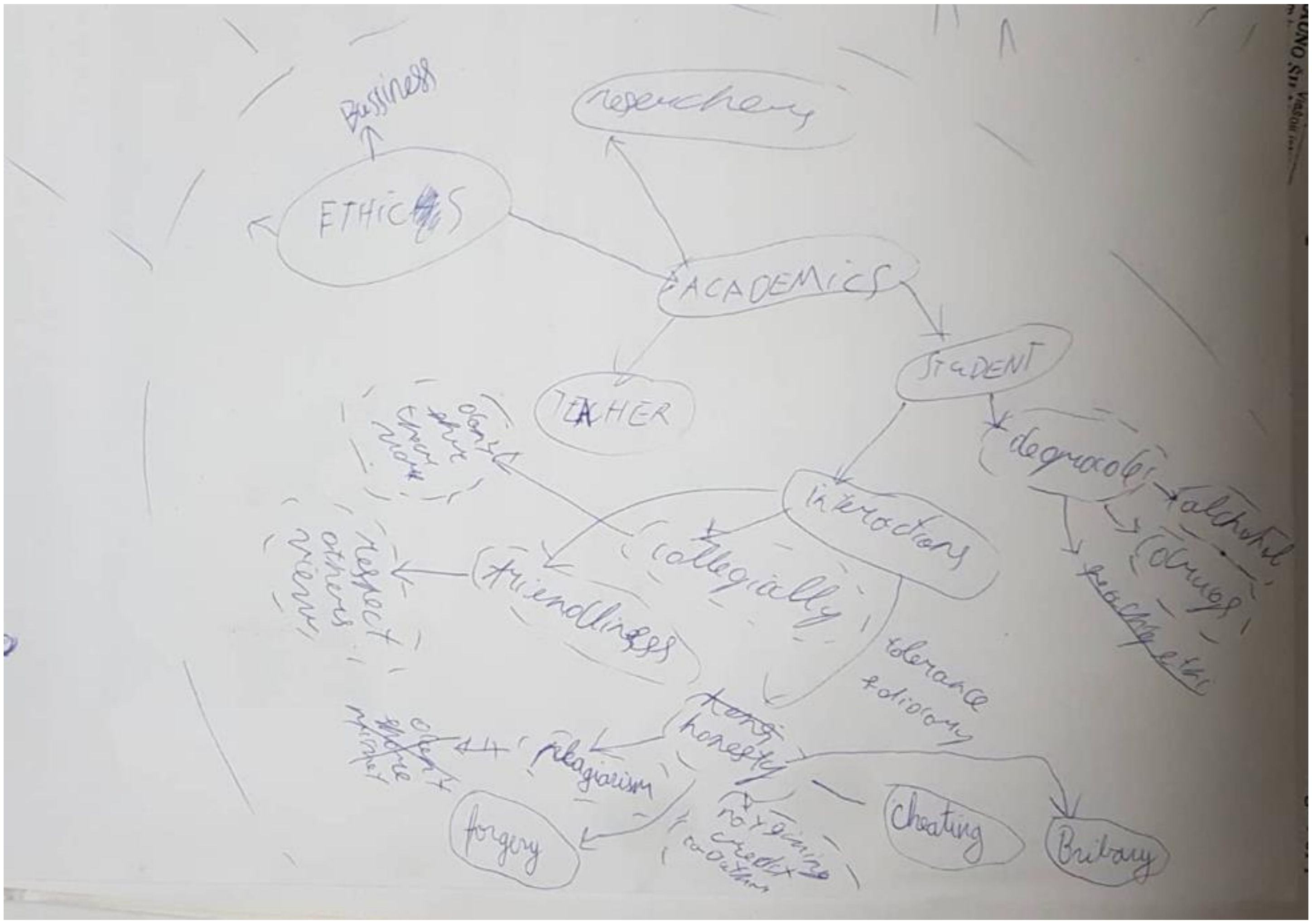
ir tada kokie tie ethics (ko turi laikytis studentai, kokiu taisykliu turi laikytis mokytojai)



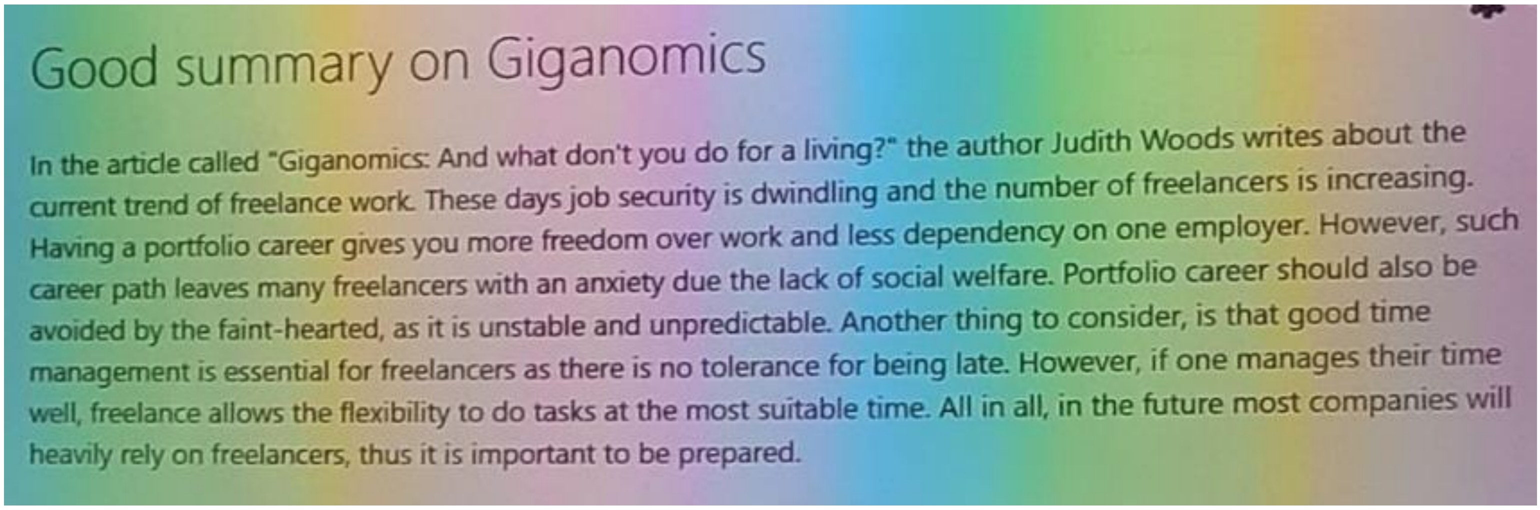
Ethics tai moraliniai principai kuriu reiktu laikytis, pvz mokykloj yra tokie principai, kad negalima nusirasinet nuo kita, reik gerbt kitus mokinius, mokytojus, lankyt pamokas ir panasiai

Darbo ethics tai vel, ten yra visokie principai kuriu reik laikytis, pvz but maloniam su klientu, nepramiegot, rengtis tvarkingai ir pan

Whatever this is:

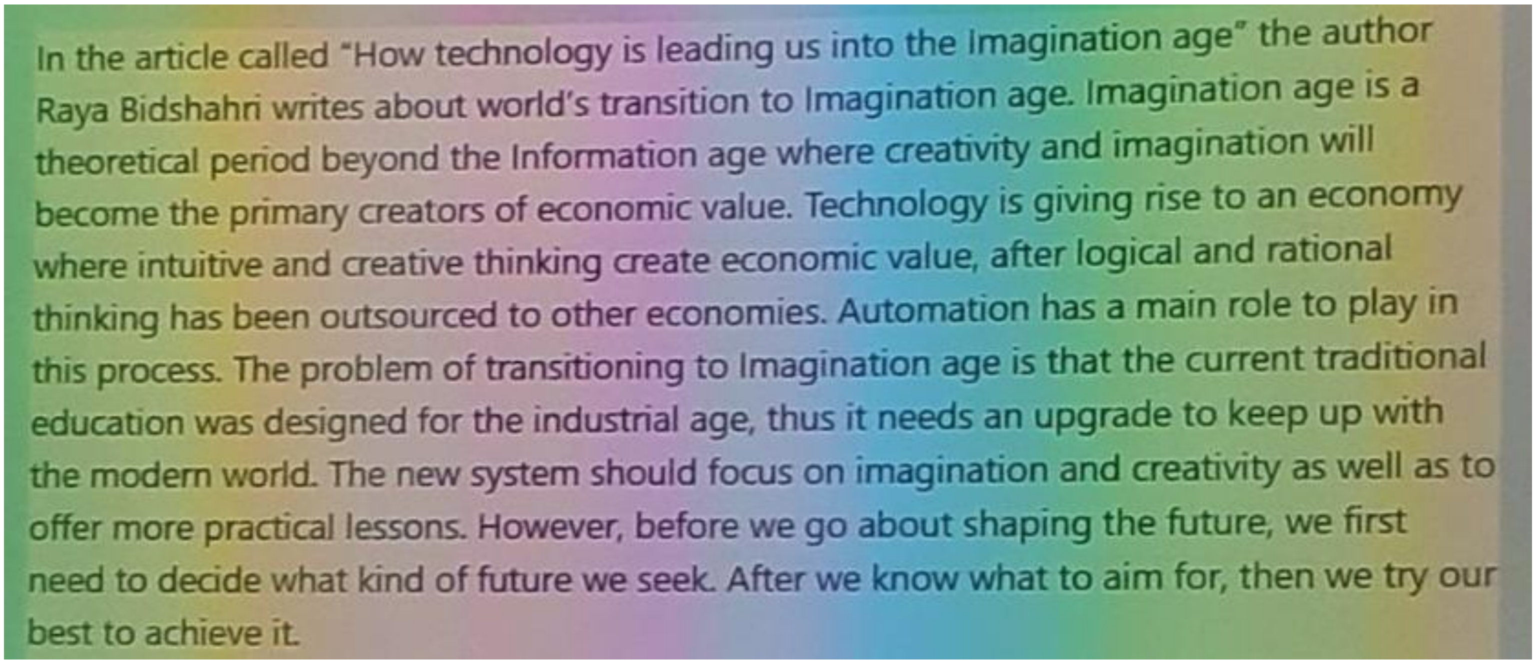


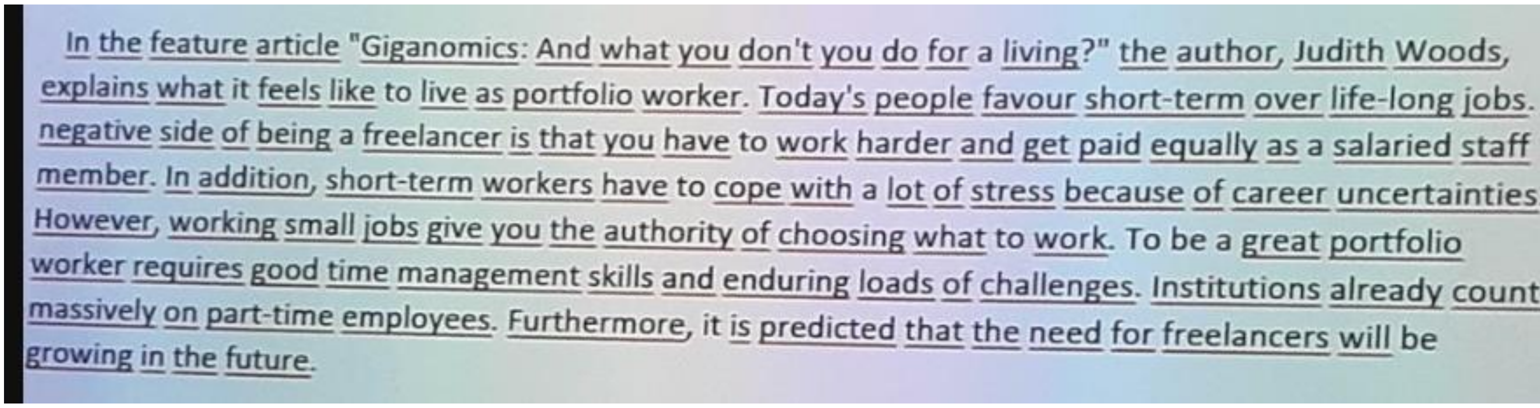
11. Summary



Good summary on Giganomics

In the article called "Giganomics: And what don't you do for a living?" the author Judith Woods writes about the current trend of freelance work. These days job security is dwindling and the number of freelancers is increasing. Having a portfolio career gives you more freedom over work and less dependency on one employer. However, such career path leaves many freelancers with an anxiety due the lack of social welfare. Portfolio career should also be avoided by the faint-hearted, as it is unstable and unpredictable. Another thing to consider, is that good time management is essential for freelancers as there is no tolerance for being late. However, if one manages their time well, freelance allows the flexibility to do tasks at the most suitable time. All in all, in the future most companies will heavily rely on freelancers, thus it is important to be prepared.


In the article called "How technology is leading us into the Imagination age" the author Raya Bidshahri writes about world's transition to Imagination age. Imagination age is a theoretical period beyond the Information age where creativity and imagination will become the primary creators of economic value. Technology is giving rise to an economy where intuitive and creative thinking create economic value, after logical and rational thinking has been outsourced to other economies. Automation has a main role to play in this process. The problem of transitioning to Imagination age is that the current traditional education was designed for the industrial age, thus it needs an upgrade to keep up with the modern world. The new system should focus on imagination and creativity as well as to offer more practical lessons. However, before we go about shaping the future, we first need to decide what kind of future we seek. After we know what to aim for, then we try our best to achieve it.


In the feature article "Giganomics: And what you don't you do for a living?" the author, Judith Woods, explains what it feels like to live as portfolio worker. Today's people favour short-term over life-long jobs. negative side of being a freelancer is that you have to work harder and get paid equally as a salaried staff member. In addition, short-term workers have to cope with a lot of stress because of career uncertainties However, working small jobs give you the authority of choosing what to work. To be a great portfolio worker requires good time management skills and enduring loads of challenges. Institutions already count massively on part-time employees. Furthermore, it is predicted that the need for freelancers will be growing in the future.