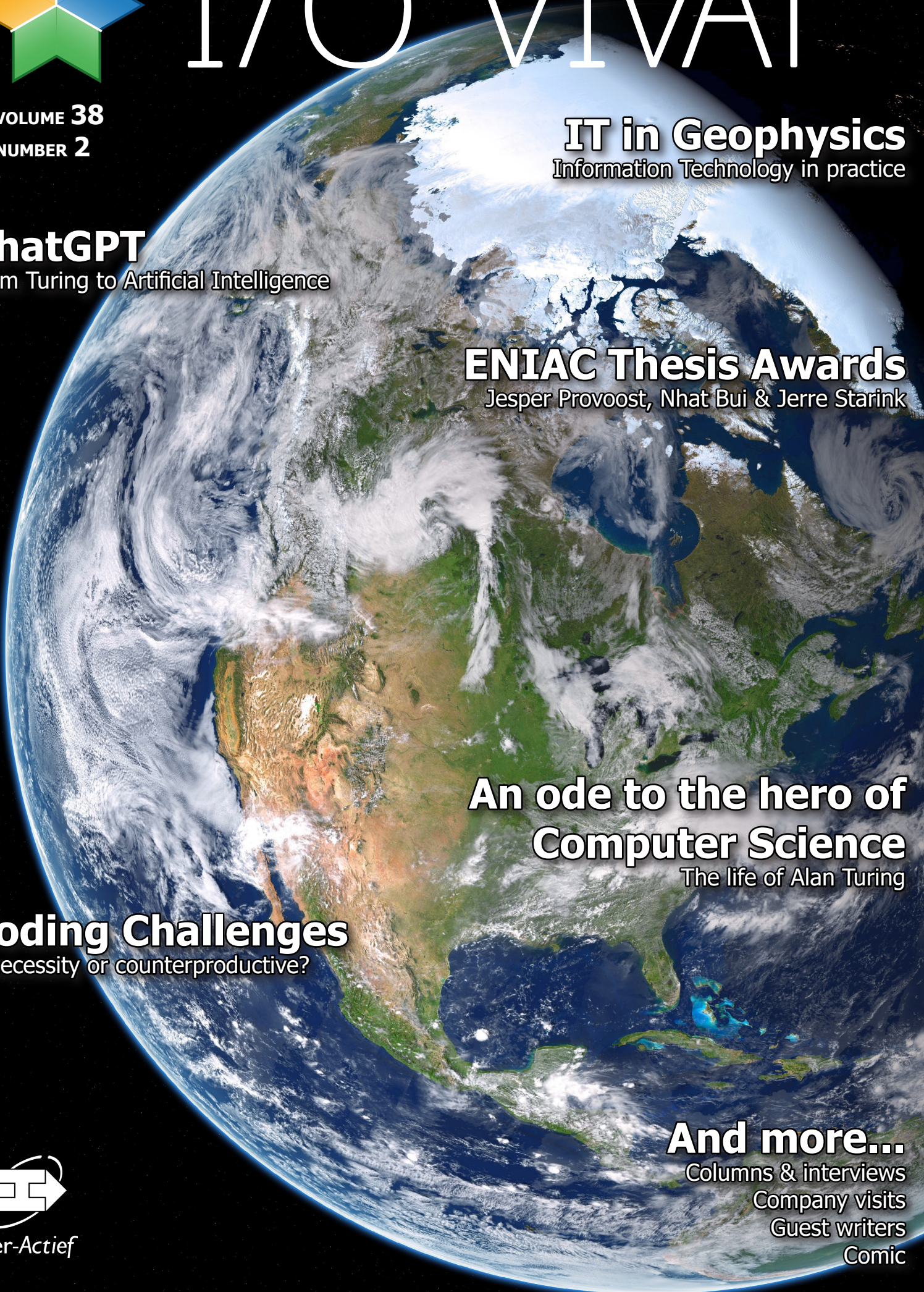
I/O VIVAT

VOLUME 38
NUMBER 2



IT in Geophysics
Information Technology in practice

ChatGPT
From Turing to Artificial Intelligence

ENIAC Thesis Awards
Jesper Provoost, Nhat Bui & Jerre Starink

An ode to the hero of Computer Science
The life of Alan Turing

Coding Challenges
A necessity or counterproductive?

And more...
Columns & interviews
Company visits
Guest writers
Comic

Inter-Actief

//Colofon

Volume 38, number 2,
March 2023
ISSN: 1389-0468

I/O Vivat is the popular scientific magazine of I.C.T.S.V. Inter-*Actief*, the study association for Technical Computer Science, Business Information Technology and the corresponding Master's programmes at the University of Twente.
I/O vivat is published four times a year (1900 copies).

// Chief Editors
Jelle Maas
Ruben Groot Roessink

// Editors
Erjan Steenbergen, Wout Velthuis, Florian Mansvelder, Joris Kuipers, Daan Wensink, Sander Teune, Tristan van Beurden, Hanna Gardebroek, Xanti Luki Lizanzu, Filip Karkalasev.

// Guest Writers
Oliver Davies, ENIAC Board, Vadim Zaytsev, Maarten Meijer, Duru Kocak, Martijn Sikking, Victor Dibbets, Jesper Provoost, Nhat Bui, Jerre Starink,

// Special thanks
Caspar Schutijser, Irma Veldman, Martin Krans.

For questions, comments or suggestions, I/O Vivat can be reached via e-mail at iovivat@inter-actief.net, by phone via 053-489 3756 or by mail:
Study association Inter-*Actief*
PO box 217, 7500AE Enschede

// Printing
Drukkerij van den Bosch & Fikkert

I/O VIVAT

//Editorial

Wait... What? A second I/O Vivat in such a short timeframe? Yes, indeed that is correct. With the full support of the Inter-*Actief* and ENIAC boards a new chapter in the history of the I/O Vivat is started.

First of all, let us say many thanks to the previous I/O Vivat committee and especially to the previous *Chief Editor* and *Executive Editor* who have been at the helm for many years. Also thanks for your support these last few months, especially in understanding the layouting part of the I/O Vivat and your efforts in finishing I/O Vivat 38.1, which has been lying on the shelf for too long.

That being said, we decided to try our best as Chief Editors of the I/O Vivat after receiving many messages inquiring about the dwindling number of publications of the I/O Vivat over many years. These positive questions really made us believe in the power of the I/O Vivat to reach a diverse public, ranging from new students that just started last year, up to former students that have been in the field for 40 years.

To achieve this, it is our (very ambitious) intention to publish the I/O Vivat four times a year once again. Clear and structured deadlines, following the calendar of the University of Twente, have been set for those delivering content to the I/O Vivat and we are going to include more columns from life at Inter-*Actief* as well.

However, to reach our ambitious goals we need your help in getting enough content. You can help in the following ways:

- Do you have an interesting company that would like to reach 1900 students and alumni? Reach out to extern@inter-actief.net for inquiring about the collaboration possibilities with the I/O Vivat (company visits, semi-scientific publications, advertisements and more is all possible).
- Would you like to write an article about an interesting project, research, publication or anything else? Please reach out to iovivat@inter-actief.net.

We thank you for you reading the I/O Vivat!

Jelle Maas
Ruben Groot Roessink
Chief Editors I/O Vivat

//Contents 38.2

38.2 I/O VIVAT //Contents

Evading AV using code injection

By: Jerre Starink
Nominee ENIAC Thesis Award 2021-2022

One of the main driving forces of many cyber security incidents is the use of malware. Last year alone, the AVTest Institute already recorded almost 100,000,000 new malware soaring through the internet, and in 2021 during the COVID-19 pandemic, this number went up as high as 150,000,000.

These kinds of statistics beg the question: How does this happen? Why is malware such a problem? Did we not solve this already with anti-virus software? Typically, it is fairly obvious that our system was infected from the moment it ended up on our computer. For example, you may have clicked a link, downloaded some file, opened it, and suddenly all your files are deleted and replaced with an encrypted version of them. It is then not rocket science to come up with the idea that the file that you just downloaded is probably malware. The actions are very visible and direct (i.e., a mass deletion of files), and the timing is also very telling (i.e., right after opening the file). However, things become more difficult when we also try to defend ourselves against Advanced Persistent Threats (APTs). APTs are threat actors that try to stay unnoticed for as long as possible, and thus, the malware that they use to infiltrate systems is specifically crafted to stay undetected by anti-virus. And one of the techniques that many APTs use to avoid detection is a technique known as **code injection**.

Code injection

The general idea of code injection is to somehow copy machine code instructions from one program into another, and then trick the other application into executing it. This way, the other application starts doing something it was not originally programmed to do. By extension, if a malicious program copies its malicious code into a legitimate application (such as a text editor or web browser), it is not the original malware itself that exhibits the malicious behavior, but rather the application that was previously considered to be benign. This has some interesting implications for a defender's perspective. All of a sudden, the task of defending a computer system against malware becomes significantly more difficult, as it means we cannot limit our vision to just unknown programs that we downloaded from the web. With code injection, *any* program can start exhibiting malicious behavior, including the ones that are developed by trusted vendors such as Microsoft and Google. As such, the *entire* system needs to be monitored, which can be quite computationally expensive. And it gets worse. Even if we are able to monitor everything and notice that our text editor starts doing something bad, should we kill the text editor? It may temporarily stop the malicious behavior, but we do not really tackle the problem at its core and remove the malware the injection came from. Besides, malware can decide to inject itself again into some other program a second time, and killing off all programs on a computer does not sound like a favorable choice either.

So what does such a code injection look like? It may sound like a complicated process, but it can be quite straightforward. In fact, most people actually use code injection on a daily basis without realizing it. For example, if you are using a web browser, chances are that at some point, you installed some kind of plugin or extension (e.g., an ad blocker). This is actually a form of code injection, as it is deliberately injecting additional functionality into your browser by loading some extra code that runs on every page load. Similarly, most operating systems also have built-in features that allow for similar constructions, where additional code or entire modules are loaded into other processes to extend the functionality of the operating system itself. For instance, Windows defines a function called **WriteProcessMemory**, which allows you to directly alter the memory (and thus also code) of any process running on the system. Furthermore, if you are using anti-virus software, chances are that it is registering itself as a **system module** (similar to a plugin) such that small monitoring routines are injected into your running applications. Both examples illustrate precisely why it is a problem and why we cannot fully abolish code injection either. There are many different methods of injecting code into another application, and the line between legitimate and illegitimate code injection is significantly blurred.

Malware analysis

In our research, we set out to get a more fundamental understanding of code injection usage in malware. To do so, we collected 17 of the most commonly used techniques, and test them to see if they are still working on a typical Windows 10 machine. We then continued by comparing every technique to each other and identified reoccurring features and characteristics. We quickly realized that there are two main camps that a specific technique can fall into. The first category comprises techniques that are **actively** interacting with the target process, while the second category includes methods that leverage features from the operating system itself and **passively** wait for the injection to happen instead. The **WriteProcessMemory** example can be considered an active technique, while system modules are types of code injections that fall into the passive category instead.

> "Malware developers are constantly trying to circumvent the radars of anti-virus"

Interestingly, during our research, we found that most malware analysis tools check for the presence of active techniques. This is likely because these are very straightforward, well-known, and widely adopted in hacker communities. Furthermore, they are easy to detect, because they translate to well-known functions that can be monitored. On the other hand, passive techniques are often left out, because it is significantly more difficult to determine whether the operating system itself is performing some operation indicative of code injection, as opposed to the malware explicitly calling some function to initiate the injection process.

Conclusions

To see how much of a problem this really is, we continued with a prevalence assessment by analyzing 3,000 real-world malware samples from the years 2017 to 2020. We found that about 11.15% of these samples used some form of code injection, of which a growing trend used a passive technique (from 40% to 48%). This indicates that malware developers indeed have realized that passive techniques are difficult to detect, and thus are an attractive option for them to further delay getting caught. Anti-virus developers and malware analysts should therefore be aware that this is a method that really should be taken into account when analyzing a program for malicious behavior.

Luckily, there is a lot of work put into this field of malware behavioral analysis, and currently, I am also continuing this project as a Ph.D. candidate at the UT as well. However, what can the average Joe do in the meantime? Of course, one obvious answer is to get into the habit of not trusting everything that you download blindly, but the best thing to do is to accept that these kinds of things *just happen*. Malware developers are constantly trying to circumvent the radars of anti-virus and have become exceedingly efficient at it. Therefore, keep your anti-virus software up-to-date, regularly scan your device for malware, and make sure you have back-ups of your important files in case things go wrong.

Figure 2: Hackers collective Anonymous uses malware to target governments

About Jerre

Jerre is originally from a small farmer's village in the north of The Netherlands. When he was young, he quickly found a passion for mathematics, computers, and programming, and this prompted him to pursue a bachelor's and master's degree in Computer Science and Cyber Security in Eschede.

Currently, Jerre is doing research on the analysis of malicious software as a Ph.D. student at the University of Twente. In particular, his main interests lie in improving the automated extraction of malicious behaviors exhibited by computer viruses. He uses an approach that combines methods found in the fields of software reverse engineering, formal software verification, and compiler theory, and he intends to bring malware analysis to a more automated process that is both more scalable and manageable this way.

Jerre strongly believes in open source and open science, where researchers and developers can easily gain access to results of previously conducted research, and are welcome to contribute their own ideas and implementations to further improve the project.

References

Sources used to write this article can be found in Jerre's thesis, publised at: https://essay.utwente.nl/88617

Coding Challenges

A necessity or counterproductive?

By: Joris Kuiper
Editor I/O Vivat

Google, Facebook, Amazon, Microsoft and many other technology companies use them during their application process. Coding challenges, algorithmic programming tasks that try to reveal how good a software engineer is at finding efficient and scalable solutions.

During the technical interview process, an applicant is asked to solve a problem on a whiteboard on a computer, while the interviewer examines the work of the applicant. When the candidate performs well during the interview and coding challenges, a job offer can be expected soon after.

However, nowadays more and more questions are being raised whether or not these coding challenges are actually effective in finding the best developers for the job position. Candidates question the relevance of these challenges, experience high levels of stress and anxiety. Although companies believe that these technical interviews offer a "reasonably consistent evaluation of problem-solving ability, communication skills, and preparedness", it is also believed that the current process fails in identifying good developers but the process is good in identifying who can perform well in an academic environment.

In this four part series I will try and shine a light on the practice of coding challenges in technical interviews. In this part I will investigate the effects that coding challenges have on developers going through a technical interview. In the next part I will look at the benefits companies experience with coding challenges and if it is actually effective. The third part will cover the design of these challenges and how to prepare for them, and finally in the fourth part of this series, I am going to look to the future of technical interviews and how they can be improved.

The history

It is not clear when and who started with coding challenges during technical interviews, however the consensus is that it started in the 90s when testing code was expensive and pen and paper, chalkboards and eventually whiteboards were used to test code. Tech companies like Microsoft started with brain teasers during job interviews to assess the skills of a potential employee. Interviewees were asked to solve simple logic puzzles and were subsequently scored on their solutions.

Gradually this evolved into proper coding challenges where applicants would be asked to solve more complex problems either on a whiteboard or on a computer. These problems would range from reversing a binary tree to perfectly shuffling a deck of cards. Entire books have been written on these exercises and courses have been created where aspiring developers can learn how to tackle these challenges.

The experience

How these interviews are set up or which types of questions that are being asked usually differ between tech companies. However, the experience from the interviewee's point of view is usually not that different. The potential developer is put under a timed and supervised pressure to solve a difficult problem on a whiteboard or on a computer. Once the candidate is done, their solution is then scored based on their performance and the efficiency of their solution.

Unfortunately, these types of technical interviews can create a stressful experience for the candidate. The conditions in which the participant is placed closely resembles the Trier Social Test. The Trier Social Test is a test used by psychologists designed for the sole purpose of inducing stress in a candidate and seeing how they respond to the experienced stress.

In this test the participant is asked to present a prepared interview-style presentation about a given subject while performing mental arithmetic in front of a panel that does not offer any feedback. The aim of the test is to measure acute stress under laboratory conditions. When placing this in the context of a technical interview, it is clear to see that companies are not looking for which candidate is the best in solving a problem but who is the best in coping in a stressful environment.

A recent study performed at the NC State University found that, when a candidate is tasked with solving a problem with an interviewer present, almost two thirds of the participants were not able to complete the task in the given time. When they were asked to solve the problem in a private setting without the interviewer present, only one third of the participants failed in solving the challenges in the same amount of time. The same study also found that more than half of the participants were able to find a solution in less than ten minutes when asked to solve it without someone else present supervising their work.

Relevance

An often voiced concern about the technical interviews process by developers is that the presented coding challenges by interviewers are not relevant to the actual job or work. In a study that looked at the sentiment of developers towards the technical interview process, one developer mentioned that the solution they made for the problem in the given time-frame became more representative of their experience instead of their actual experience. The developer felt that their CV and what they had learned over the years became totally irrelevant when they had to do coding challenges at a job interview.

On the other hand, proponents of the technical interview said in the same study that understanding fundamental algorithms is, in their opinion, more favorable than understanding the myriad of modern frameworks. Although both sides of this discussion can be found in forum threads online, most of the time you will find the number of times stated that the current technical interview process is not relevant outweighs the number of times stated that coding challenges were useful and relevant for their eventual work.

Conclusion

Coding challenges during the technical interview process have become a cornerstone of the solicitation process in the field of software engineering. Unfortunately, it has turned out to be a procedure to reliably induce stress in the candidates in a manner that very closely mimics the setup of the Trier Social Test. Companies are no longer testing the quality of the interviewee but they are testing how well the candidate can perform in an acute stress situation. Furthermore, the relevance of these coding challenges are questioned more and more. These observations are clearly indicating that for candidates of technical interviews, the current practices are lacking or faulty and should be changed.

This is part one of a four part series in which I try to investigate the world of coding challenges during technical interviews. In the next part I will look at the process from the point of view of the companies.

References

https://pubmed.ncbi.nlm.nih.gov/28229114/https://doi.org/10.1109/VLHCC.2019.8818836

https://doi.org/10.1109/VLHCC.2019.8818836

https://dl.acm.org/doi/10.1145/3368089.3409712

https://betterprogramming.pub/a-history-of-coding-interviews-23b5e8f9c92f

https://www.goodreads.com/book/show/25707092-cracking-the-coding-interview

https://dev.to/lynnetye/engineering-whiteboard-interviews-yay-or-nay-3hko

https://doi.org/10.7717/peerj-cs.173

SIDN

Interview with Caspar Schutijser

By: Joris Kuiper & Wout Velthuis
Editors I/O Vivat

Caspar Schutijser
Research Engineer at SIDN Labs

With 'common' users only experiencing the internet through a browser, the backbones of the internet are often overlooked. Every time someone visits a .nl domain, servers of SIDN are involved, even though most people don't realize that. In order to shine a light on this important organization for the 'Dutch part' of the internet, we interviewed Caspar Schutijser, who works as a research engineer at SIDN Labs.

Could you tell us a bit about SIDN?
The reason why SIDN exists is already covered in the name Stichting Internet Domeinregistratie Nederland *[in English: Foundation Internet Domain Registration Netherlands]* (a foundation was the original legal form of SIDN). Recently, all activities for .nl have been transferred to a new private limited company called SIDN B.V. We maintain the .nl Top Level Domain (TLD), which means that we manage a massive database with all the .nl domains and publish it through DNS.

People can register a .nl domain through a registrar into our database. We ourselves do not do the registration, but that is done through registrars. The registrars register a domain with us and then we generate a zone-file in the DNS domain and we ensure that this file is put on our nameservers such that if someone wants to use a .nl domain, they can do a DNS lookup and then access the service as intended.

Why is it not possible for consumers to buy a domain directly from SIDN?
We believe that everyone who registers a .nl domain name should be able to choose a service provider that meets their needs. We issue .nl domain names through a large network of registrars, all with different specialisms, prices and services. Services such as hosting and web design, which we don't provide. The size and diversity of the registrar community is one of the main reasons why .nl is such a large and successful domain.

What are some difficulties you encounter in maintaining the .nl TLD?
There are increasingly more laws for digital infrastructure and critical infrastructure. Every time a new law is introduced, we have to act on it. We are also heavily involved in giving feedback on new laws and regulations. In Europe, there is an organization, CENTR, which has a lot of country level TLDs from inside and outside Europe. They have working groups and together they keep an eye on developments related to TLD operators.

On a technical level, there are a lot of challenges. For example, how to deal with DDoS attacks, or new recommendations for DNS are released through a new RFC. These new recommendations talk about how certain aspects of DNS should work and these recommendations have to be implemented by our infrastructure team.

Another challenge we have is the recruitment of new employees. The current demand for IT personnel is a problem for many companies, but we do have a lot of open vacancies to fill. This remains a challenge.

Your website states that you are one of the best TLD maintainers in the world. What differentiates you from the other TLD maintainers?
An important aspect is that we have one of the most reliable and safe top-level domains. In particular, we aim to reduce the amount of abuse in our TLD. An example of that is our effort to take down fake web shops. Those are shops that do not deliver the ordered products, or shops that sell counterfeit products.

Furthermore, our infrastructure is very robust and our team is technically very knowledgeable. I do not want to say that this is bad everywhere else though, but it is something I am proud of.

Besides that, many people enjoy the fact that we are a non-profit organization. The money that remains after all our costs are covered are used for projects that are beneficial for the internet.

SIDN has different divisions, namely SIDN Labs and SIDN Fund. Can you tell us a bit more about them?
At SIDN, we currently have more than a hundred employees. SIDN Labs is the research department of SIDN. For maintaining a .nl domain we ask a few euros a year. We do not need all the revenue we earn to maintain the .nl domain nameservers, instead we use the remainder of our re venue in a way that benefits the Netherlands and the internet community as a whole.

Because SIDN is a not a for-profit organization, we want to use the remaining income to do good. Part of our revenue goes to the SIDN Fund. This separate fund is free to spend this funding as they see fit, but all with the aim of making the internet stronger and more accessible or using the internet in an innovative way. If anyone has a good idea that overlaps with a certain theme and the internet, they can submit a project idea and receive funding.

You did your graduation project at SIDN Labs into IoT devices. IoT is also one of your research fields currently. What are the biggest challenges you encountered while researching IoT and especially the security of IoT devices?
That is a good question. Although we have shifted our focus away from IoT a bit recently, the biggest challenge, that of security, on those devices remains. Security on IoT devices is often very underwhelming. Default passwords can easily be found online, which enables others to easily make botnets, software is often not patched, or no patches are available or the devices have insecure default settings.

Then why did SIDN Labs shift its focus away from IoT devices?
With only a team of ~11, we unfortunately cannot work on all the subjects at the same time, and we realized that we should focus a bit more about the future of the internet. We decided to leave the SPIN project, the most important software that we made with regards to IoT, to the community. This was easy as the project has always been open source. This way, we can focus more on the future of the internet.

Could you tell us more about 'future internet' and which kind of research you do into this?
Broadly speaking, we ask ourselves what we, as a society, want from the internet in the future? Because, whether we like it or not, the internet is playing an increasingly bigger role in our lives with more digitalisation and more IoT devices. The internet is working pretty well but it is also not perfect. What we at SIDN Labs would like to have is more transparency on the internet. We are thinking about new mechanisms that create more transparency on the internet. We try to build and implement new prototypes and demos to show how it could be.

> "Call us and you will be helped"

What kind of trends do you see in the future of the internet and IoT devices?
I mentioned an important subject earlier, namely transparency, which really grabs the interest of more and more people on the internet. Additionally, if you look at the fundamentals of the internet then you see some protocols that have existed since the 1970s, for example TCP, BGP, etc. We have had them for tens of years and they have been working pretty well so far. Because they work so well, people think that it will be difficult to change the internet.

You previously mentioned the SPIN project, can you elaborate?
SPIN is one of the projects we made to counter the lacking security on IoT devices. SPIN is simply said, a piece of software that you can attach to your home router and the software keeps track of the IoT devices on your network. With SPIN, you can see what your IoT devices do on your network. Furthermore you can use SPIN to block certain network traffic. For example, if you see that your Smart TV is connecting to Facebook and you do not want that, you can block the traffic between Facebook and that device.

The idea for SPIN was created around 2016 when there were a lot of DDoS attacks with a botnet called Mirai. This botnet existed from a lot of hacked IoT devices and we realized that it would be a big problem if a botnet like Mirai would decide to attack the .nl nameservers, for example. That is why we decided to look into what we can do to improve security from IoT devices.

What kind of graduation projects do you offer at SIDN (or SIDN Labs)?
We have students graduating on the future of the internet, and we had a student that looked into a new internet protocol. The latter researched how to implement this new proposed internet protocol and especially at how anycast can be implemented in this new architecture. We also have students graduate in domain name security, i.e. fake webshops, and on the SPIN project.

We offer a broad range of graduation projects and internships. At SIDN Labs, we have been doing this for a long time, since research is part of our DNA and we have a lot of interaction with the academic field. Some of my colleagues have part time appointments at universities. We are also involved in some courses at the UT, namely in the master courses Advanced Networking and Security Services for IoT (SSI). Besides that we also offer guest lectures on a regular basis.

Graduating at SIDN

Interested at graduating at SIDN? Send an e-mail to sidnlabs@sidn.nl. Include research ideas and why your research would benefit from collaboration with us. Also add your CV.

If you spark our interest, we invite your for a conversation on your (and our) interests!

IT in Geophysics

Classifying Earth from above with Machine Learning

By: Erjan Steenbergen
Editor I/O Vivat

Humanity has found a way to analyze almost every aspect of the earth using maps. We have been able to visualize climate change, de/reforestation, city growth, etc. using satellite images. How are these maps created and how can we show all these important aspects of the earth so well? In this article, we will dabble into the world of remote sensing and how machine learning algorithms help us classify geothermic images.

Remote sensing is exclusively possible because of electromagnetic radiation. Electromagnetic radiation is made up of electrically charged particles called photons, which are the fundamental units of light. These photons can be exhibited as both a wave and a particle. The wavelength of a photon is related to its energy through the equation $E=hc/lambda$, where E is the energy of the photon, h is the Planck constant, c is the speed of light, and $lambda$ is the wavelength of the photon. This equation shows that the shorter the wavelength, the higher the energy of the photon.

Absorption and scattering are two important phenomena that can be described when identifying a photon as a particle. Absorption occurs when a photon is absorbed by an atom or molecule and its energy is transferred to the atom or molecule. Scattering occurs when a photon is deflected by an atom or molecule and changes direction.

Due to scattering we can see colors, as some wavelengths are visible for the human eye. For example, leaves are usually green, since they scatter photons with a wavelength that represents green. The visible light spectrum is only a very small part of all the possible wavelengths identified on earth.

These phenomena are used in remote sensing to create maps from satellite images. This process involves collecting images of the Earth's surface from satellites or other aerial platforms and then using these images to create maps. The images are typically taken in different wavelengths of the electromagnetic spectrum so called 'bands', such as visible light or infrared, and can provide a wide range of information about the Earth's surface, including its composition, topography, and land cover. For example, if a researcher wishes to analyze vegetation growth, the best way to separate the trees from other objects is to use the near-infrared and red bands. Healthy vegetation usually has a high reflectance in the near-infrared band and low reflectance in the red band.

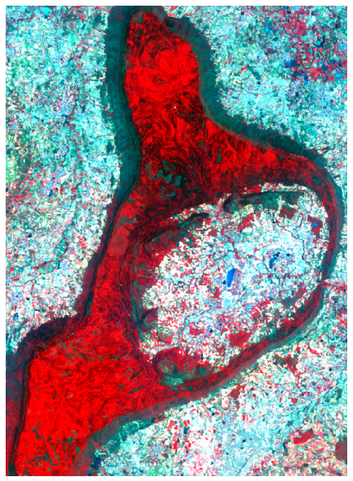


Figure 1: Aerial picture of Phu Wiang area in Thailand showing vegetation as red

This is what is shown in Figure 1. It looks really interesting, but why is this done and how does this go from a very detailed image, to a very simplistic, easy to understand map? Luckily, machine learning algorithms known as "Image Analysis Methods" help us classify these images to single-colored pixels.

The process of classifying an image does take some human effort. A user first 'trains' the algorithm by classifying small areas of a certain class. For example in Figure 1, we could have two classes; forest and no forest. An important question to ask is: how is it possible to differentiate forests with other vegetation?

Different land covers sometimes do reflect the same color, but we have various means of differentiating them, such as texture, shape, size, pattern, etc. In our context, forests have a more grainy texture, whereas other vegetation, such as fields, have a smooth texture. Therefore, we will classify grainy, red-ish areas as 'forest' and other areas as 'no forest'. After the initial classification is finished, an algorithm will be chosen to further classify this image. Figure 2 shows the final product.

Figure 2 is classified using an algorithm called Random Forest (RF). No, this name does not come from being able to classify forests with it, but because it splits the data into subsets and uses the 'majority vote' of multiple decision trees to classify pixels. Compared to a single decision tree algorithm, it greatly reduces its overfitting problem by doing this. Its namepart 'random' stems from the fact that the decision trees are created randomly. Another positive of RF, is that its parameters are comparatively found to be very robust.

On the other hand, RF-classification may have difficulties handling very large datasets. This is not a problem for Support Vector Machines (SVMs), which is another machine learning algorithm. They can handle large datasets very efficiently. Additionally, it can handle very small datasets with good accuracy as well. Just like RF, SVM has a small chance of overfitting, as it uses a hyperplane in a feature space, ignoring outliers. Although this lightweight algorithm works very efficiently, it cannot reach an accuracy as high as other popular algorithms in remote sensing.

Another very accurate machine learning algorithm used in remote sensing is Convolutional Neural Networks (CNNs). As opposed to previous shallow algorithms, CNNs deep learning allows it to have a better and more powerful generalization ability than other algorithms. Sadly, due to the limited amount of training data in the field, CNN is not able to live up to its potential. Additionally, compared to others, it has a very high computational cost and thus has a very low time efficiency as of now.

These algorithms all have something in common, which is the fact that they are pixel-based, meaning that they classify each pixel as a certain class. Another method of classifying remote sensing images is Object Based Image Analysis (OBIA). OBIA is a fairly new found method that involves grouping pixels into "objects" based on their physical properties, rather than analyzing individual pixels in isolation. These objects can be individual buildings, trees, or other features on the ground.

One advantage of OBIA is that it can more accurately capture the shape, size, and context of features on the ground, since it takes into account the relationships between pixels within an object. This can be particularly useful for detecting and mapping features that are difficult to discern at the pixel level, such as small buildings or trees. Another advantage of OBIA is that it can more effectively incorporate additional sources of information, such as topographic data or prior knowledge about the landscape, to improve the accuracy of the analysis.

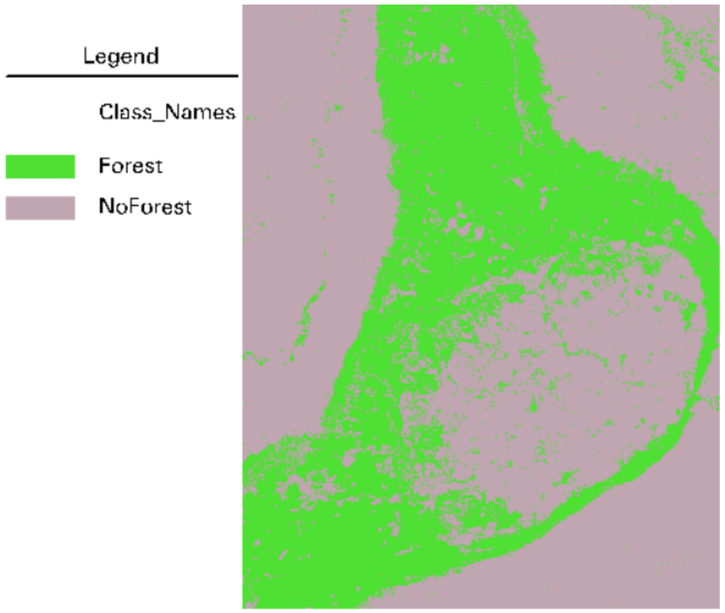


Figure 2: A map of Phu Wiang after using Random Forest Classification

However, there are also some disadvantages to OBIA. One disadvantage is that it can be more computationally intensive than pixel-based algorithms, since it involves analyzing and grouping large numbers of pixels into objects. Additionally, OBIA can be more sensitive to changes in the resolution or quality of the imagery, since it relies on the relationships between pixels within an object.

As in other fields, there is more than one viable algorithm to set your heart on. It is not as simple as picking the best algorithm, as it does not exist. Every algorithm has its advantages and disadvantages. The art of choosing the most applicable algorithm relies on the dataset available, or just trying every algorithm there is. The choice is up to you.

References

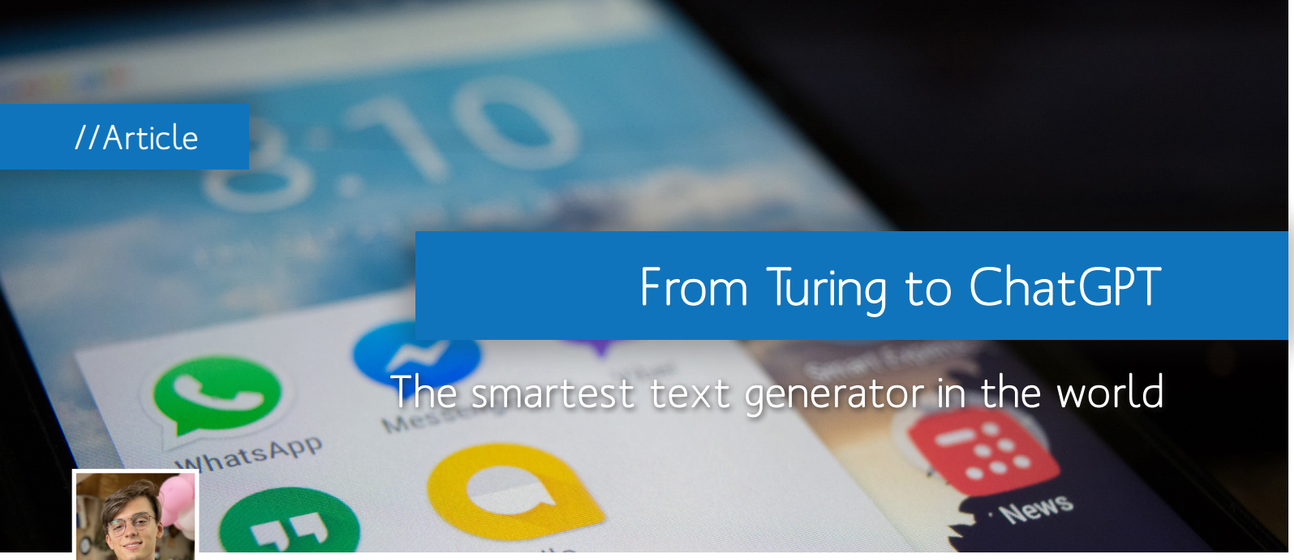
https://www.tandfonline.com/doi/full/10.1080/01431161.2018.1433343

https://arxiv.org/abs/1508.00092

https://www.tandfonline.com/doi/full/10.1080/20964471.2019.1657720

https://link.springer.com/chapter/10.1007/978-3-540-77058-9

https://www.sciencedirect.com/science/article/pii/S0924271613002220



From Turing to ChatGPT

The smartest text generator in the world

By: Filip Karkalasev
Editor I/O Vivat

The quest of building Artificial Intelligence, or intelligent machinery, is in my opinion very interesting, because it brings us as humanity closer to the question of what it means to be intelligent. But what does intelligent machinery, or the more suitable term, "intelligent computers", mean?

Intelligent Computers

In order to answer this question, I am going to make the parallel between the human computer of the early 20th century and the modern digital computer. "The idea behind the digital computer is that it can perform any operation that a human computer can", is the description of the digital computer as given by Alan Turing in "Computing Machinery and Intelligence" (1950). Using this definition, our next question would be: "What could a human computer do according to Turing?" I assume it would have been calculations like: "compute the Taylor polynomial of sin(x-1)" or similar repetitive arithmetic. Anyway, the nature of the tasks given to a human computer was repetitive, boring and arithmetic. So, if the digital computer was designed to mimic the human computer, the nature of the tasks given to the digital computer will also be repetitive, boring and arithmetic.

Limits of Human Computers

So, now I ask again: What does Artificial Intelligence mean? What is the goal? I am going to answer this question by answering the question: "What does an "intelligent human computer" mean?

Well, let me first state what I do not think an "intelligent human computer" is: we cannot expect the human to go beyond the limits of its natural processing and storage abilities in order to calculate ever more bigger things, like the Taylor polynomial of sin(x-1) + cos(x^2) – tan(x^3), faster and faster. At a certain point, the human will have reached his maximum calculating performance. However, what would aid us as the instructor, will be quite straightforward: that we could go to higher levels of abstraction with our instructions, as we educate our computer. For example, instead of dividing our specific need into small chunks, one of these chunks being: "compute the Taylor polynomial of sin(x-1)", we could simply instruct it to "find a function that has the same trajectory as sin(x-1)". The computer would now need to be able to know that a Taylor polynomial would be a suitable way to calculate the answer to the above question. So, summarising, making a human computer more intelligent would mean that we could give him/her ever more abstract instructions, keeping in mind that the nature of these tasks is repetitive and boring, which would require us to educate him/her. Similarly, this could hold for the digital computer. Now that we have an understanding of what it is we want to do with our intelligent computer, our task is to find a way to educate it in order to suit our needs.

Abstraction on abstraction

If the next question that arose in you was similar to: "What are the possibilities of these ever more abstract instructions?

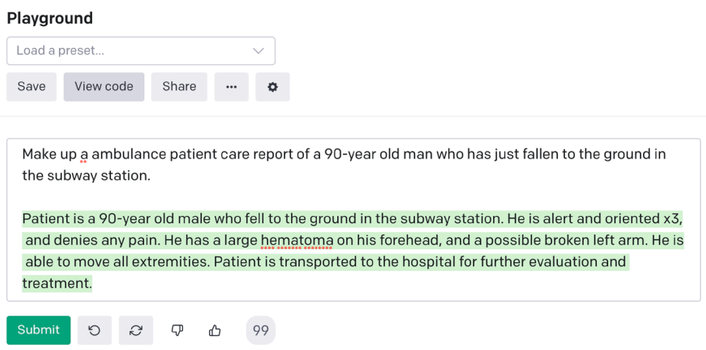


Figure 1: Example of chat capability of ChatGPT

Where does it end?", I am with you. But from the definition above, it seems unlikely that an instruction would be: "paint a masterly, realistic landscape", because this task does not seem very repetitive, boring or arithmetic.

If the goal of AI can be phrased as: "attempting to go ever levels of abstraction higher in instructing the computer", then OpenAI's ChatGPT, which I am going to talk more about in the following part of this article, does a very good job at it.

ChatGPT

Some time ago, one of my friends told me about this company, founded by Elon Musk, that researches and produces language models, which has to do with probability. Their latest version, (Chat)GPT-3, is, if you ask me, a novelty. At a low level, I would not be able to explain what GPT-3 does, other than that it predicts next words, trying to continue the input text you give it. At a high level, it can generate very realistic English and code. Furthermore, it is very easy to use: you open up the so-called "Playground", which is just an input prompt, and select your desired length and randomness (and some other parameters you can play around with). For example, if you ask ChatGPT to write a report for a patient, it returns a realistic report as shown in Figure 1.

Writing Code

In addition, ChatGPT can produce pieces of code in a lot of different languages. For example, it can seamlessly produce the following code from the instruction "Write a piece of code to compute the first 100 prime numbers".

```
"""
Write a piece of code to compute the first 100 prime numbers.
"""

def is_prime(n):
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    for i in range(3, int(n**0.5)+1, 2):
        if n % i == 0:
            return False
    return True

def first_n_primes(n):
    primes = []
    i = 2
    while len(primes) < n:
        if is_prime(i):
            primes.append(i)
        i += 1
    return primes

print(first_n_primes(100))
```

Submit

Figure 2: Example of 'code' capability of ChatGPT

Impressive, to say the least. First off, the model was able to interpret the assignment. Secondly, it was able to produce an astonishingly realistic narrative piece of text or code. So, all in all, ChatGPT seems to be a great leap forward in natural language processing and its ability to generate realistic sentence structures is a powerful tool for future research and development.

Conclusion

Going back to the goal of AI posed earlier, is it of any use that we can generate realistic text and code? Is it not the goal for the computer to understand our instructions, not necessarily to imagine and create itself? But then on the other hand, how could we ever know the computer understands our abstract instructions if it will not be able to respond adequately?

Alan Turing

Having read some of the papers of Alan Turing, I admire his attempt to explain his often difficult ideas in simple words, including the idea of the imitation game (or Turing test). Although the goal of this thought experiment is still not 100% clear to me, it is quite clear that he was ahead of his time in terms of thinking about AI and what it could do: one of the things that stands out to me most was that he wrote something along the lines of "Building AI will not necessarily require faster and more powerful computers than there are now (referring to the 1950's), it is all in the lines of code."

To explain this idea, he makes the analogy to a writer writing a story: it is not the paper that needs to be altered, but the meaning of the words. If that idea is true, it makes coding into sort of an artform that is basically limitless and has a huge impact on the future of humanity.

Alan Turing's life ended tragically, however he left behind ideas that are interesting to explore for everyone interested in the fundamental questions regarding computers and AI, especially given that he was at the frontier of these fields.

A more in-depth biography of Alan Turing is given in the artcle 'The Disgraced Father'.

//Column

From the Chairman

In with the new, in with the old

By: Oliver Davies
Chairman 44th Board of Inter-Actief

It feels like our change GMM was just last week. It is weird then that, at the time of writing, I have been part of the Inter-*Actief* board for over three months. It's safe to say that time has flown by, and while I have been enjoying every minute of my board year, there is enough for us still to do. We made promises at the GMM that we fully intend to keep, and with the chaos at the start of a board year slowly coming to an end, these months are the best time for us to make them come true.

This year, as you may or may not have heard at the end of the association song, we intend to "zoom out". You may be wondering what exactly we mean by a phrase, which seems to have more to do with cameras than a policy plan. For us, zooming out is a great way to look at the association's bigger picture. We want to look further than the multiple social activities that we are so known for. Part of this is placing more focus on serious and educational activities, but we also want to become a place more appealing to international students, master students and university staff.

Besides just looking at the bigger picture, we want to take a look at the physical image of the association, which makes "Zoom out" an even more fun play on words in my opinion. As you may have already seen, we have already started this by reconsidering the design of the Inter-*Actief* office. It is often pretty messy, but in my opinion, we have made it a lot more spacious and relaxed. Besides the office, we also want to give the beloved Inter-*Actief* site a fresh lick of paint, to keep up with the modern times!

This year already, we have had some fresh, new activities. This year, a sailing weekend was finally organised! Unfortunately, the weather was too bad to sail, but the participants had a fun weekend. We also held a board game evening in the Mbasement to introduce master students to each other and the association, which was low-key, but well-received. Seeing the participants' enthusiasm for new activities is very rewarding, and motivating for our ideas for the rest of the year!

As you can see, as a board we are always so focussed on new things that we can do. But thinking about my highlights of the year so far, lots of them have been old traditions that have returned post-COVID. We have seen the first Beagle in five years, the first Christmas dinner in three years and the Inter-*Actief* top 100 is finally physical again. Even looking forward, there is enough coming up that has taken its inspiration from pre-COVID times. The International Business Course is on track to take place again for the first time since 2019, and we are planning to do a promotional round past university staff like previous boards used to. To me, this just goes to show that not only new things can add value to a year, but renewing old ideas can be pretty cool too!

All in all, it has been a busy, but rewarding few months so far. I've had some awesome experiences and joined fun activities which perfectly summarize the identity of Inter-*Actief*. With all our plans and upcoming activities, I can't wait to see what's in store for us for the rest of the year!

Figure 1: aXi Beagle

About Oliver

22 years ago, Oliver was born in Sydney, Australia. At 5 years of age, he moved to Amsterdam, where he quickly learned Dutch and settled in.

Here, he developed an interest in mathematics and IT, so decided to start following the double degree in Applied Mathematics and Technical Computer Science at the University of Twente in 2018. Four years and one dropped AM degree later, he is now the chairman of Inter-*Actief*.

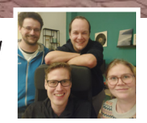
In his free time, you can mostly find him at Inter-*Actief* activities or with his fraternity BAGGER.

//Column

From the ENIAC Board

Getting to know the (candidate) board

By: ENIAC (candidate) Board



With a new year, comes a new board. At the time of writing (January 4th) ENIAC has a candidate board that it would like to introducee. At the time of publication the candidate board will, most likely, be the new ENIAC board. The candidate board is composed of the following members:

Egbert Dijkstra - Chairman

Heya! I'm Egbert Dijkstra (25) and part of the ENIAC candidate board as chairman! I started my studies in 2015 to finish them in Q3 2022. In the meantime I didn't do nothing! I was part of the Inter-*Actief* board in 2018-2019, as officer of External Affairs, was part of the Board of Advisory of SNiC, a national computer science and artificial intelligence congress. As part of various committees I partook in the organization of a couple notable events, like the Inter-*Actief* Top 100 and the first edition of the Summer Sounds. Now that I'm working, I'd like to stay in touch with alumni and students alike!

Koen Braham - Secretary

Hi! My name is Koen Braham (29) joining the candidate board as candidate secretary. I graduated back in 2018 for my master Computer Science following the Wireless and Sensor Systems specialization. Unfortunately, this track is no longer offered. That makes me one of the last WiSe students. The specialization offered flexibility to pick and match interesting courses from cyber security and electro engineering.

Since my graduation I attended ENIACs activities to catch up with my friends and help students on the well-known speeddates. For the new board I would like to keep the activities going and make sure the speeddates stay as a success.

Kevin Alberts - Treasurer

Hey everyone, my name is Kevin Alberts (28), and I aim to become the next treasurer of ENIAC. In 2013 I was in the same cohort as Josje when I started studying computer science, and I finalized my bachelor last year after maybe a bit too many years of studying and activism. I've been a board member of IAPC back in 2015/2016 (also as treasurer) and I participated in a variety of committees at Inter-*Actief*, among which the SysCom, WWW-committee and SocCie. After finishing my bachelor I am currently enjoying a healthy combination of working, helping out the technical committees at Inter-*Actief*, and occasionally spending some time in the good old Abscint.

Josje van 't Padje - General Board Member

Hi all, I am Josje van 't Padje (29). I am in the current board of ENIAC as a General Board Member and I am in the Candidate Board again as a General Member. I started my bachelor Technical Computer Science (back then it was still called Technische Informatica) in 2013. I have been the Officer of Educational Affairs of Inter-*Actief* in the year of 2015-2016. After my bachelors I continued with the master Computer Science, track Data Science which I finished in 2021. I really enjoy being a board member of this association in addition to my job, therefore I like to continue as a board member for ENIAC.

About ENIAC

ENIAC is the alumni association for the studies Business & IT and Technical Computer Science and corresponding masters. Additionally, ENIAC represents the interests of the previous study Telematics. ENIAC was founded to stimulate contact between alumni as well as between alumni and current students. In order to do so, ENIAC organizes several activities (see below) that are both open to members of ENIAC and Inter-*Actief*.

Activity Calendar

Regional Drink (Enschede City Center) - March 10th

Regional Drink (Utrecht) - April 21st

Graduation Speeddates (1/2) - May 24th

Large Activity Texel - June 17th

Regional Drink (Enschede Campus) - June 27th

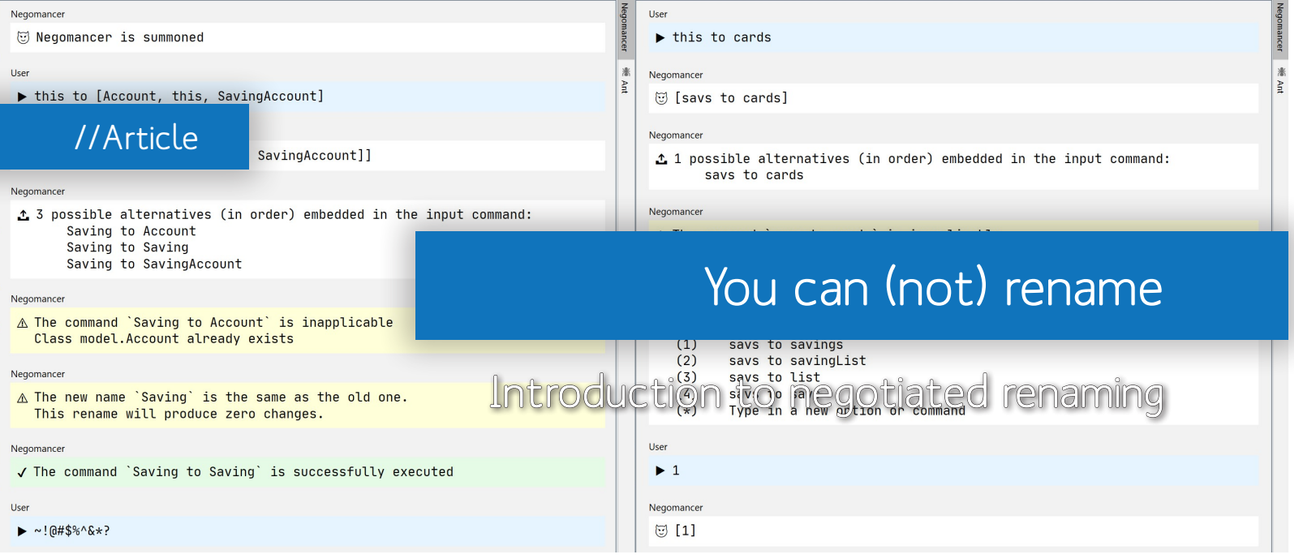
Family Activity - September 16th

Online activity - October 13th

Graduation Speeddates (2/2) - October 18th

Regional Drink (Leiden) - November 17th

Budget GMM with activity - December 16th



//Article

You can (not) rename

Introduction to negotiated renaming

By: Nhat Bui
Nominee ENIAC Thesis Award 2021-2022

On average, 70% of any codebase is identifiers. Hence, one of the most used techniques of code refactoring is renaming. Renaming is the process of globally and systematically changing the name of an identifier within the code to better represent its purpose. Symbol renaming is widely supported in most code editors and integrated development environments (IDEs). However, most, if not all, of the current renaming flows are basic and linear, or with a low level of negotiation. Negotiate is the act of trying to achieve a compromise between the transformation engine and its user, giving the user more flexibility and the engine more adaptability.

In this project, we experimented on introducing proper negotiation into the refactoring flow, starting with the rename operation, by formulating a negotiation scheme and expressing it through a custom domain-specific language (DSL).

Background

A living (natural and software) language will (almost) surely evolve from its initial form along with its grammar to reflect the needs of its users. A grammar evolution can be viewed as a model transformation, which can be further expressed by a sequence of transformation operations. In most cases, the transformation process is linear in the sense that it provides zero degrees of negotiation, which will report an error and halt whenever a transformation command is assessed to be inapplicable or vacuous (i.e., a transformation leads to zero changes.).

On the other hand, code refactoring is the process of purposefully restructuring a piece of existing code, deliberately and systematically altering its internal structure to improve its quality, while maintaining its external behavior. Underneath, code refactoring is done by performing a chain of transformation operations on the code elements with the same principles as grammar transformation. Here, we are particularly interested in the rename operation. Renaming in short is the process of globally and systematically changing the name of a non-terminal symbol within the code to better represent its purpose.

Extensive studies have been conducted to investigate the nature of identifier renaming and why developers rename, showing that the majority of developers want and need more assistance with naming identifiers. While the current approaches have some back and forth with the user, presenting the renaming opportunities and suggestions, the negotiation is only introduced after the initial renaming operation is performed. This leaves the actual rename operation initiated on the initial target identifier with little to no negotiation. Our work aims to complement existing methods, rather than replacing them, by introducing proper negotiation directly into the transformation process itself.

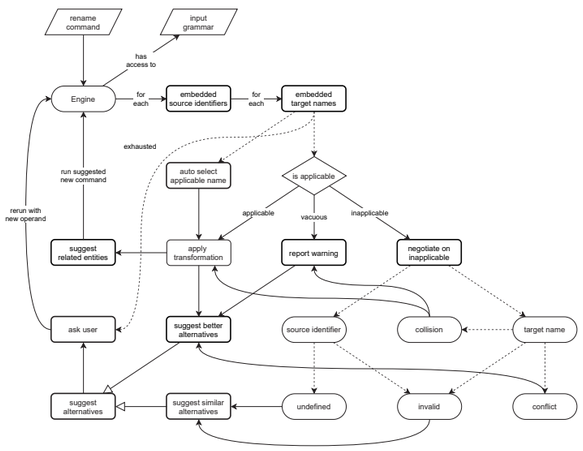


Figure 1: Proposed negotiation scheme for rename refactoring

Negotiation strategies

We dissected the construction of a rename operation and formulated a negotiation scheme to fit it, based on the previous works of Zaytsev (2012, 2014). More formally, a rename command can be written as: *rename: a → b*

We found out that there are two negotiable main arguments in a rename command, namely the source identifier *a* and the new target name *b*. Under certain conditions of defined-ness, validity, and equality of these two arguments, a rename command can be: applicable, vacuous, or inapplicable. For each scenario, we examined and curated a combination of strategies to introduce negotiation into the renaming flow.

For applicable rename commands, these strategies are already acceptable. However, this is where other research on automatic (re)naming recommenders and identifier name quality can be applied to provide suggestions on naming improvements and opportunities. Regarding vacuous rename commands, although they will result in zero changes to the source code, failing the entire command sequence when facing one does not help. Thus, it is more logical to accept them and move on to the next command.

Rename commands are inapplicable because of its arguments. For the source identifier, it can be undefined or invalid, while for the target name, it can be invalid, a collision, or a conflict. All these cases can benefit from some suggestions. However, unlike others, a collision does not cause a fatal error that halts the transformation, which can still be considered acceptable with an extra warning.

Overall, we defined two categories of negotiation strategies: active and passive. *Passive negotiation strategies* do not require user intervention while the command is being processed. This includes strategies that can be handled internally by the engine and those that can be embedded directly in the command:

Embedded alternatives: Allows embedding different alternatives in the command. The engine will go through each one in the defined order and stop at the first acceptable command.

Auto select applicable name: The user explicitly requests the engine to select any applicable target name for the command.

Vacuous acceptance: Produces a vacuous warning and moves on to the next command.

Collision acceptance: Produces a collision warning and still applies this transformation if it is acceptable by the language.

Conversely, *active negotiation strategies* require *user inquiry*. While the command is processed, if the engine encounters any issue, it will prompt for user input. The degree of actively asking the user is limited by the inquiry level to control the tradeoff between automation and interactivity. This includes:

Suggestions: Present to the user a list of applicable suggestions.

Manual negotiation: Ask the user for manual input to recover the command. Alternatively, the user can also provide a new command to replace the current one.

These strategies combined to form a negotiation scheme (Figure 1) that enables negotiation in the renaming flow. This scheme is not only useful for the rename operation but also for any operation that contains names that can apply similar name negotiating strategies to recover from some cases of inapplicability.

Negomancy - A language for negotiation

Negomancy was created to express negotiation for renaming. Contrasting to a general-purpose language, a DSL is much simpler to learn and use due to its more limited scope, focusing only on a specific domain, hence terms can be expressed at the abstraction level of the problem domain.

The main design philosophy of *Negomancy* syntax is concise and intuitive. Statements in *Negomancy* can be either a command to request the engine to perform some action or a reply to respond to the engine's prompts. Each statement is a combination of case insensitive keywords and string arguments separated by whitespaces.

The negotiation features of *Negomancy* are closely based on the strategies in the established negotiation scheme above. At the syntax level, some negotiation strategies are not available, namely the *vacuous acceptance* and *collision acceptance* strategies. In addition, *Negomancy* also provides some extra negotiation for case styling the names using the same principle as the *embedded alternatives* strategy.

Negomancy is a special language that depends on and leverages the extensibility of the implementing platform to perform checks and execute the refactoring transformation. To demonstrate that a language like *Negomancy* is feasible, we built *Negomancer*, a plugin that serves as a proof-of-concept of negotiated refactoring built on top of the *IntelliJ Platform* written in the *Kotlin*. *Negomancer* partially adapts the proposed DSL *Negomancy* along with its features and the negotiation abilities. *Negomancer* is presented as toolwindow (see header image), a collapsible side panel that the user can access on demand, or it will automatically open when a rename action is triggered from the source code.

To evaluate the expressiveness of *Negomancy* and ensure that it faithfully represents the domain of interest, we have performed an ontological analysis. The ontological analysis was done by mapping and comparing the proposed DSL and an independently designed ontology of the domain of negotiation on the rename refactoring operation. The results of this evaluation showed that for most parts *Negomancy* expresses the domain with completeness and clarity. The main issue remaining is the lack of a complete (and preferably concise) way to represent a specific identifier, which deserves its own research and falls beyond the scope of this project.

References

Sources used during this research can be found in the thesis as published by Nhat and co-authors. The thesis is available at https://essay.utwente.nl/88515.

Zaytsev, 2012, https://doi.org/10.1145/2467307.2467313

Zaytsev, 2014, https://doi.org/10.5381/jot.2014.13.3.a1

//Article

Chatting up I/O Vivat

How GPT is Changing the Game for Articles

By: Joris Kuiper
Editor I/O Vivat

The use of artificial intelligence in content creation has been on the rise in recent years, and one of the most popular tools in this field is ChatGPT made by OpenAI. I was curious about how this technology could be used to create articles for the I/O Vivat. So, I decided to put ChatGPT to the test and ask it to write this very article.

Writing an article for the I/O Vivat magazine using ChatGPT is a powerful way to generate high-quality content quickly and easily. The I/O Vivat is a magazine created by the study association Inter-*Actief*, aimed at providing insights and news to its members, alumni, and companies associated with the association. The magazine focuses on topics related to the world of Computer Science that are of interest to its members, and ChatGPT is a state-of-the-art language generation model that can write text in a wide range of styles and formats, making it a valuable tool for creating content for the I/O Vivat.

The first step in using ChatGPT to write an article for the I/O Vivat is to choose a topic that is relevant to the association, its members, alumni, and associated companies. This could be a recent development in the field of Computer Science, a new technology that is gaining popularity, or a research paper that has been published in a prestigious journal. The topic should be something that would be of interest to the readers of the I/O Vivat, and should provide valuable insights and information.

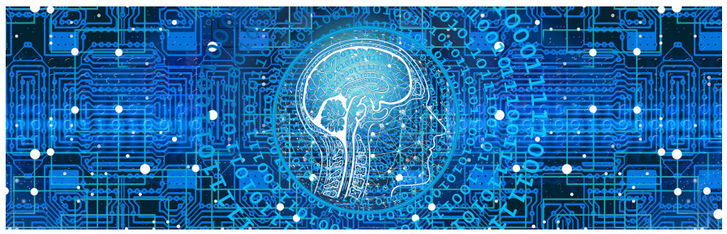
Once the topic is chosen, the next step is to provide ChatGPT with some background information and a few examples of the type of text you would like it to generate. This can be done by inputting a few sentences or paragraphs about the topic, and then providing some examples of similar articles or pieces of writing from previous issues of I/O Vivat magazine or other technical publications within the association's industry. ChatGPT uses these inputs to understand the style, tone, and content of the desired output, and can then generate text that matches these requirements.

One way to generate an article is by doing an interactive session, in which the model will generate text and you will prompt with it new text to keep going, in this way you can guide and direct the flow of the text, correcting any mistakes or adjusting the direction if needed. This approach allows you to have more control over the text generated by ChatGPT, and to make sure that it is in line with the requirements of I/O Vivat.

Another alternative is to set up a specific prompt, which you can use to generate a complete article in one go, though this will require some more previous work and research to make sure you can set the prompt to the desired tone and style of the magazine, and to the topic and specific points you want the article to address. This approach can save time, as it allows you to generate an article with a single command, but it requires more careful planning and preparation.

As ChatGPT generates text, it is important to review and edit the output to ensure that it meets the standards of the I/O Vivat and that the text is free of errors. This may involve correcting grammar and spelling mistakes, rephrasing sentences, and adding or removing information as needed. Also, it's important to make sure that the text is in line with the association's mission and values, providing relevant insights and value to its members, alumni and companies.

When the article is complete, it can be submitted to the I/O Vivat redaction for review and possible publication in a future edition of the magazine. The use of ChatGPT can help streamline the writing process, allowing you to produce high-quality content quickly and easily. Furthermore, this approach can also provide a way to have fresh perspective and ideas on the topic, since the model can use its vast knowledge base to come up with new angles and insights on the matter.



It is important to keep in mind that while ChatGPT is a powerful tool, it is still a machine learning model that can make mistakes and may require human editing and oversight. However, by providing it with clear instructions and editing the output carefully, you can ensure that the text generated by ChatGPT is of the highest quality and suitable for publication in the I/O Vivat. Additionally, it is also important to consider ethical implications of using AI generated content, such as ensuring that it is clearly labeled as generated by a machine, and that it does not infringe on any copyright laws.

One of the benefits of using ChatGPT for writing articles for the I/O Vivat is that it allows for more consistent and frequent content creation. The magazine can be updated more frequently with new articles, without having to rely solely on the availability of human writers. Additionally, ChatGPT can also assist with research for the articles, as it has access to a vast amount of information on various topics, making it an ideal tool for finding relevant data and statistics to include in the article.

Another benefit of using ChatGPT for writing articles is that it allows for a more diverse range of perspectives and writing styles. With the model's ability to understand and emulate different writing styles, it can be used to write articles in different styles, such as a formal research paper or a more casual news article. This can help to appeal to a wider range of readers, and make the magazine more engaging and interesting.

While using ChatGPT to write articles for the I/O Vivat magazine can be a powerful way to generate high-quality content quickly and easily, there are also some downsides to consider. One of the main concerns is the possibility of errors or inaccuracies in the text generated by the model. As with any AI-generated content, it is important to review and edit the output to ensure that it is free of errors and that the information is accurate. This can be time-consuming and may require additional resources to ensure that the content is of high quality.

Another downside is the risk of bias in the generated text. As ChatGPT is trained on a large dataset of text, it may inadvertently include biases that are present in the training data. This can result in text that is not inclusive or representative of all perspectives, and may not be suitable for publication in a magazine like the I/O Vivat. It is important to monitor the generated text for any potential biases and to make necessary adjustments to ensure that it is inclusive and representative.

Additionally, the use of ChatGPT can also raise ethical concerns regarding authorship and the potential for AI-generated content to be passed off as human-written. It is important for the I/O Vivat to clearly label any articles generated by ChatGPT as such, and to be transparent about the role of the technology in the content creation process. This can help to ensure that readers are aware that the content has been generated by a machine, and can make informed decisions about the value of the information presented.

In conclusion, using ChatGPT to write articles for the I/O Vivat magazine is a powerful way to generate high-quality content quickly and easily. By providing the model with background information and examples of the desired output, and by carefully crafting a prompt, you can ensure that the text it generates is suitable for publication in the magazine. With human editing and oversight, the use of this tool can save a lot of time and effort in the writing process, while providing fresh perspectives and insights on various topics in the field of Computer Science. It is important to keep in mind the ethical implications of using AI generated content and the need for clear labeling and compliance with copyright laws.

OpenAI

References

The author generated this text with GPT-3, OpenAI's large-scale language-generation model. Upon generating draft language, the author reviewed, edited, and revised the language to their own liking and takes ultimate responsibility for the content of this publication.

OpenAI's ChatGPT is available at https://chat.openai.com

About ChatGPT

According to its creators:

ChatGPT: Optimizing Language Models for Dialogue

ChatGPT is a model which interacts in a conversational way. The dialogue format makes it possible for ChatGPT to answer followup questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests. ChatGPT is a sibling model to InstructGPT, which is trained to follow an instruction in a prompt and provide a detailed response.

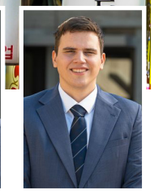
About OpenAI

OpenAI's mission is to ensure that artificial general intelligence (AGI)—by which we mean highly autonomous systems that outperform humans at most economically valuable work—benefits all of humanity.

We will attempt to directly build safe and beneficial AGI, but will also consider our mission fulfilled if our work aids others to achieve this outcome.

Study Tour Evolve

Singapore & South Korea



By: Martijn Sikking & Victor Dibbets
Study Tour Committee Evolve

Last September, a group of students and teachers set sail to Asia to embark on a study tour. The group spent 3 weeks exploring Singapore and South Korea by visiting companies and universities while also partaking in cultural activities.

Preparation

During these visits participants can experience the working environment, cutting edge technology and get a tour around leading universities and companies. Next to these visits, the participants also have the opportunity to face an entirely new culture and experience the ins and outs of Singapore and South Korea.

Ofcourse, such a trip does not appear out of thin air. The committee which organised study tour Evolve started all the way back in June 2021. From early mornings to try and contact companies in the countries we were visiting to late nights figuring out if all our plans actually fit the budget, a lot had to be done. However, the participants also have to put in a lot of effort. Prior to the study tour, participants have to complete a research project within the theme of the study tour, "Cities of the Future", similar to the research project at the end of the bachelor's programme. This was performed in the third and fourth quartile of the previous academic year. In the summer break, the participants performed a contract research assignment. These assignments are IT-related work assignments at a company or the university. These contract research assignments do not only provide the participant with relevant working experience, but also fund the study tour.

Finally, the participants follow a 3 week course at the start of the new academic year. During this course, they conduct research and analysis on the countries we visit. The cultural differences between the Netherlands and the countries of destination are analyzed, as well as the scientific and professional landscape. Now that all this was done, it was finally time to depart!

Singapore

After sitting in the plane for around 15 hours, with a transfer in Helsinki in between, we arrived in Singapore. Here we stayed in a nice hostel in a district called Little India, with the local 7-Eleven available to supply us with food, even if jetlag keeps us up late at night. On an average day in the study tour, we visit 1 or 2 companies/universities, with a visit being either in the morning or in the afternoon. The evenings were either planned with activities, like visiting the Gardens by the Bay, or participants had some free time to explore the city themselves. Besides activities organized by the committee, many people also came up with things to do themselves. For example, some people went to watch the F1 race that took place while we were visiting Singapore. After 10 days in Singapore, the study tour continued its journey in South Korea.

South Korea

After being interrogated about potential corona symptoms, everyone safely made it through into Korea. After hopping onto a bus we were driven straight to the city of Daejeon. We would be staying there for a couple of nights and we stayed in a beautiful hotel, which was a godsend following the large hostel rooms in Singapore. Amongst other things, we visited KAIST, a university with a beautiful campus where we got to learn about the research they do and projects the university had worked on in the past. We also went to Twinny, a smaller company which makes self-navigating robots, which ended up chasing us! After those couple days in Daejeon, we took an early bus to Seoul, the capital of South Korea. Luckily for us, the day after arriving in Seoul was a holiday, so we could take some time to explore the city. Some headed into the city to do some shopping, while others visited the mountains surrounding Seoul for some hiking.

Figure 1: Participants of Study Tour Evolve

After these free days, it was time to see what Seoul had to offer in terms of company and university visits. We went to Korea University, KINX (the Korea Internet Neutral Exchange) and even a bus that took us all to Incheon (just west of Seoul) for some visits there. Similarly to Singapore, these visits were also focussed on Cities of the Future. The last visit we had was to OP.gg, a company which produces an application to help players in certain Esports.

Of course, there was also still time to have some fun (not that the company visits are not fun obviously). While in Seoul we also went on a ferry cruise over the Han river, and even got up super early in the morning to get to the DMZ, the demilitarized zone of the border between North and South Korea.

Figure 3: Pavilion in Singapore

All in all, we feel the trip was a great success, but if you want to know more about the trip we encourage you to take a look at the travel blog. The blog was written by various participants of the tour and given a unique insight into everything that happened while in Asia! You can find the travel blog at *ictsv.nl/evolve*.

It is of course too late to join Study Tour Evolve, but don't be sad, there will most likely be another Study Tour in 2024! This Study Tour will go to a new destination and have a new theme so it could be a unique experience for you, while also picking up work experience at the same time. If all that is not enough for you, consider joining the next Study Tour committee, information for which will be available soon.

Figure 2: South Korean Temple

Study Tour

The study tour is a trip organized by a committee of Inter-*Actief* where students get to experience the (working) culture in countries outside of Europe. This takes place every 2 years, so the next iteration will take place in September or October of 2024.

Besides students, several teachers join the study tour. They are there to make sure the participants do the work to earn the ECs (study points) they can receive for joining the study tour, as well as join in on the fun.

In general, a study tour visits 1 or 2 countries. Previous iterations of the study tour include:

Study Tour Shift (2018) - South Korea & Japan

Study Tour MISC (2016) - Singapore, Malaysia & Indonesia

Study Tour USB 14.0 (2014) - United States & Brazil

Study Tour Noodle (2012) - South Korea & China

Study Tour Bonsai (2008) - Japan

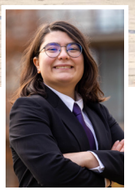
Study Tour Chakra (2005) - India

Study Tour Samba (2003) - Brazil

//Column

Symposium Denarius

Finance & IT - March 14th 2023



By: Duru Kocak

Secretary Symposium Committee Denarius

One of the most prominent traditions of Inter-*Actief* is the yearly Symposium.

Every year Inter-*Actief* organizes a Symposium; a day centered around a chosen domain from the vast world of Computer Science. The goal of the Symposium is to give students a more in-depth look into a specific part of the world of Computer Science, as well as getting them into contact with companies that specialize in the domain, which can give them insight into the different career paths that are at work in the sector.

During the day, several speakers talk about topics relevant to the chosen theme, and students talk to multiple companies that work in the field in between the lectures. Those companies are present to have one-on-one conversations with students, to explain what work they do in the sector, and often about the career opportunities they can offer those interested.

This year the symposium, named Denarius after the Roman silver coin, will cover aspects around the latest technological developments in the sector of finance and cash flow. Symposium Denarius will deal with the role of Computer Science in the world of money and delve into the intersection of finance and technology. How has information technology revolutionized payment methods, stock trading, and cryptocurrency? How is data analytics changing the way financial decisions are made? How can technology be used to enhance risk management and compliance in the financial industry? Just by a few clicks on a smartphone, many financial transactions can now be easily completed online or contactless; what are the security risks, and how are they tackled? We hope to provide a new perspective on the impact of IT on finance and by extension our society. Held on the 14th of March at Schouwburg Hengelo, this symposium will bring together experts and professionals from the finance and IT industries to discuss the role of technology in finance, the future of fintech, and answer all questions.

Speakers from various companies such as MoneyBird and ING will share their insights and expertise on a variety of topics related to finance and IT. Some of the speakers include Bart Leurs, the CITO of Rabobank, Luc Correia Cabrito from Icoinic and, Prof. dr. ir. C.W. Oosterlee from UU. This is a great opportunity for students and professionals alike to learn about the latest developments in these fields, network with industry experts, and gain valuable insights into the future of finance and technology.

As the annual symposium of Inter-*Actief* is drawing closer and preparations are continuing at full speed, the Symposium Committee invites you to join us on the 14th of March.

DENARIUS

38.2 //25

I/O VIVAT

Symposium Denarius

The symposium is organized by 6 Inter-*Actief* members of the association, who have been working for the past year on arranging a location, budget by collaborating with companies, and, most importantly, interesting speakers.

More information can be found on the website of Symposium Denarius: https://symposiumdenarius.nl.

//Column

Dies Natalis

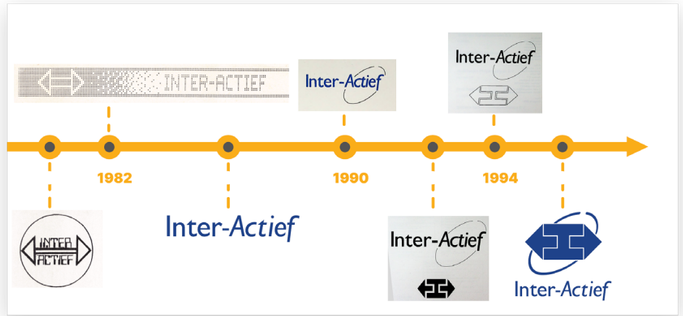
By: Maarten Meijer
Board Representative Dies Committee 2023

During the week of the 12th of March, Inter-*Actief* dedicates their time on their Dies Natalis - or birthday. On this very day in 1981, Inter-*Actief* was founded as a study association, when the study split off from Mathematics. Back then, the association was called S.V.I. Inter-*Actief*, and the board had meetings in the same room as Abacus.

During the formation of the association, several names for the new association were proposed. Among these, the bigger competitors were “MANIAC”, “INTERFACE”, “LITTLE BIT” and “DIGIT”. The prize for the winning suggestion “INTER-ACTIEF” was for Prof. Blaauw, who won a brand-new Rubik’s Cube.

After the name was chosen, the first logo was made, having an “arrow of interaction” as it’s main element. One and a half year later, a second revision was made, while keeping the arrow.

Figure 1: History of Inter-*Actief*’s logos



Some time later however, the arrow got lost, but the font was changed to the one that we still use today! With a couple different revisions of the logo, the renowned swirl was added and the logo transformed into the one that we see around the association all the time.

To celebrate this year’s birthday of Inter-*Actief*, the Dies committee has been busy preparing extraordinary activities.

When it’s your birthday, it cannot be forgotten to go all-out and have some fancy wine-and-dine with a luxurious dinner. Thus, like every year, the Dies dinner will not elude us, as well as the Dies reception beforehand, where we will enjoy some delicious cake. To further get into the birthday ambiance, the Dies committee is also organising an unique, large-scale, real-life game, but I won’t spoil any details for that yet…

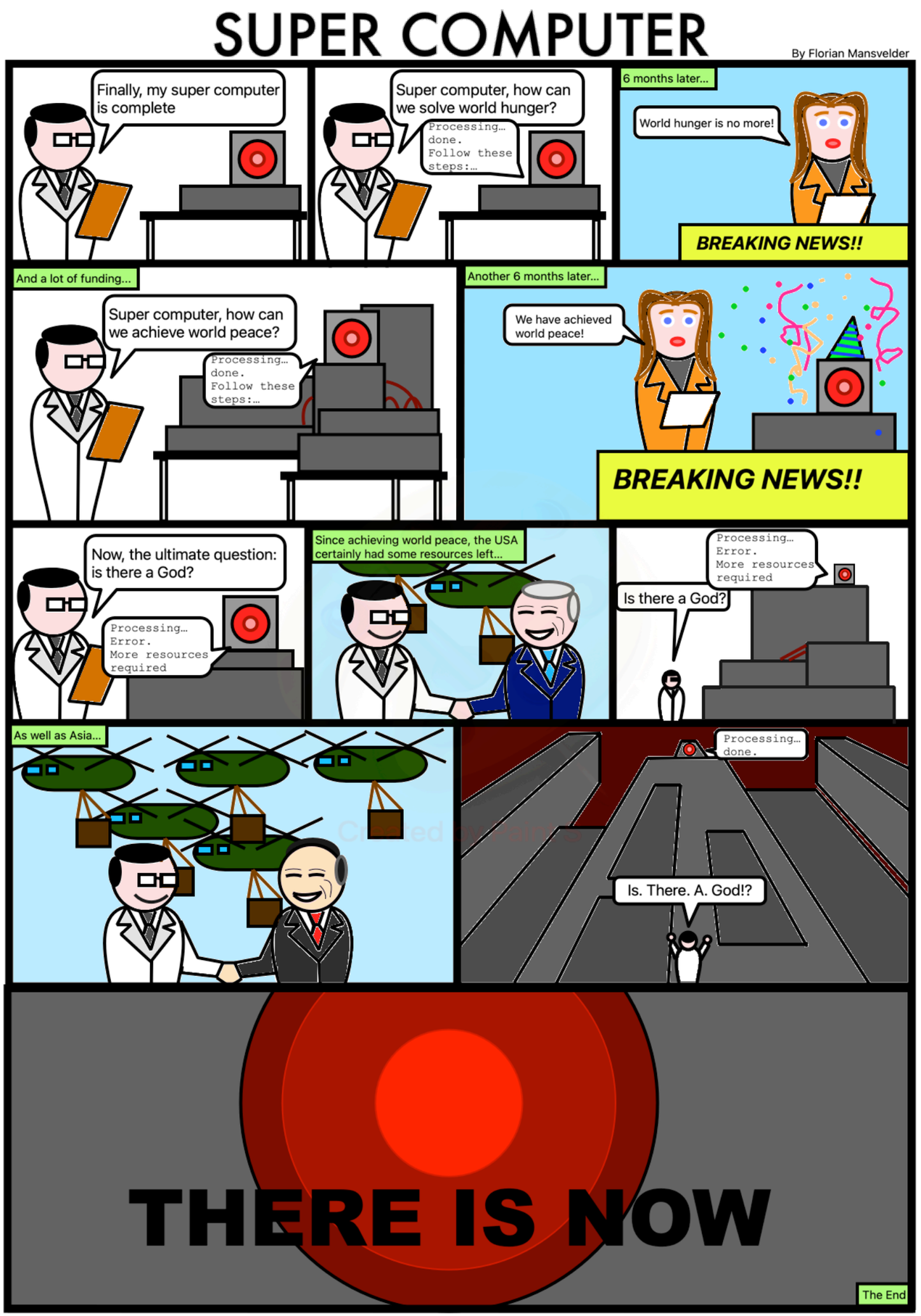
Figure 2: Dies Roadmap 2023

Dies Natalis

Dies Natalis, or ‘Dies’ in short, means ‘date of birth’ in Latin. The term, which originated in the Catholic Church, is commonly used among universities and student associations to signal the birthday of an organisation.

A Dies Natalis is celebrated with many festivities around the birthday of the organisation. Inter-*Actief* celebrates its Dies on March 12th, exactly two months before the University of Twente on May 12th.

The Dies committee happily invites you to join the festivities up and around March 12th 2023.



//Article

The Disgraced Father

An ode to the hero of Computer Science

By: Sander Teune
Editor I/O Vivat

The legend Feynman, the nightmare Laplace

A physics student generally likes to talk about his heroes. It usually does not take long for an enthusiastic member of Arago to completely overwhelm you with facts about Feynman. You'll hear about his eccentric personality and his contributions to the world of Physics (something I myself am completely unfamiliar with, by the way) in all colourful details. However, this phenomenon is not limited to Physics. Most fields of science have their heroes and icons. The average mathematician cannot help but feel a certain sense of hype at the words Euler and Gauss. Even studies like electrical engineering, despite their "practical" approach to the subject matter, deal with dozens of founding fathers of their field, in the form of theories and formulas named, without further inspiration, after their respective inventors. What electrical engineering student doesn't cringe when he hears the word Laplace?

However, this phenomenon that I would like to call the "heroes-of-the-field" seems absent in computer science. I'm sure the average TCS student will have had some history of computer science, but only what could be considered the bare minimum. Then there is the average BIT student, where the history, philosophy and people behind computer science matters even less. This is an incredibly unfortunate phenomenon, precisely because our beautiful field of computer science is rich in many spiritual fathers, mothers and heroes. So, I thought it would be a good gesture to put one of these individuals in the spotlight for my debut article in the I/O Vivat.

Computer science has never existed in a vacuum. Because of our tendency to limit our vision to what we see in front of us on our screen, we sometimes forget the great impact our science and work has on the world around us. There was one person to whom this was emphatically clear, the father of modern computer science: Alan Turing.

The Imitation Game

Some of you will now be thinking "Ah, that guy." Others may be reminded of the movie "The Imitation Game" from 2014. Perhaps there are others to whom the name doesn't mean anything to at all. For those who have seen The Imitation Game: bravo! You can skip to the last few paragraphs from here. For those who haven't seen the Imitation Game: I'm probably going to slightly spoil the movie for you.

June 1912. Europe was a significantly different place than it is today. Amsterdam was still a city without electric lights, Germany was an empire, and everyone was talking about the Titanic, which had sunk about 3 months prior. In this Europe, in London, on June 23, Alan Turing was born.

In Turing's youth, two things were immediately apparent: he had an aptitude for mathematics, other beta subjects and he was extremely athletic. Although he was incredibly good at math, his school didn't really appreciate this, reasoning that school was no use if you only wanted to do beta science. During this time, he soon became friends with Christopher Morcom and together they would spend hours solving mathematical problems. Unfortunately, Christopher died of tuberculosis at the age of 17. After the death of his best friend, Turing began pondering a whole new question that had occurred to him: what is intelligence and consciousness and how was it connected to matter? This got Alan thinking about the brain as a machine, based on mathematical logic: a computer.

Alan went on to study mathematics at King's College in Cambridge. After that he studied for a while at Princeton in America, only to return to England, just before the outbreak of World War II in 1939.

Cyberwarfare

Almost immediately after the outbreak of war, Turing went to work for the British Intelligence Service, as a decoder at Bletchley Park. Here he set about decoding messages that Nazi Germany was sending around in order to stay one step ahead of them.

Turing, along with his team, developed several computers capable of translating the Nazis' secret language: the so-called Enigma code. The highlight was when he managed to translate the messages from the German submarines, allowing the British army to be one step ahead of the German navy; something that was crucial for supplying the island country. Later Winston Churchill would state that the only thing that ever had him worried during the war, was the phenomenal German submarine force.

After the war, Turing was awarded a position in the Order of the British Empire, but this was to remain a secret. It was calculated that his actions were so instrumental in the end of the war, that he caused the war to end 2 years earlier. It is no understatement that Turing saved thousands, if not millions of lives with his mathematics and computers.

Turing was a hero after the war. It seemed like nothing could go wrong; he was obviously a hero. However, this all turned out to be short-lived. After World War II, a new era of war began: a cold war. Turin was arrested several years after the war. The reason? He was homosexual.

Disgraced Genius

Turing was found "guilty" of homosexuality. Besides the fact that at this time homosexuality was outlawed in England, people speculated that the Soviet Union might have used this to blackmail him. He was given two options by the judge: either he had to go to jail, or he would be released on the condition that he would be chemically castrated with hormone therapy. Turing chose the latter. This is where his downfall truly reached its high point. The hormone therapy, aimed at making him impotent, had terrible effects on him. In addition to all this, he was fired from all his positions and the rumours that he was being blackmailed by the Soviet Union grew louder. On June 7, at age 41, he was found in his home by his housekeeper. He had died of cyanide poisoning and an investigation determined suicide as the cause of death. A man who contributed so much to computer science and saved thousands of lives died disgraced in his own home.

Artificial Intelligence, Anno 1936

Turing's contributions to computing are also still incredibly relevant in our time. Are we at the forefront of a new paradigm? Looking around us, A.I. is spontaneously everywhere. What for decades was a theoretical topic with only a few practical applications is now more than reality: A.I. art programs such as Dalle-E and the open-source Stable Diffusion seem to be able to generate art that is indistinguishable from the real thing. Software such as ChatGPT gives the impression that you are conversing with a real person. However, the concept of A.I. and certainly talking to a robot is not a new phenomenon. Here, too, Turing was a pioneer.

Creating life and attributing life to non-living objects has always been a fascination of man. Look, for example, at our ancestors who attributed anthropomorphic characteristics and intelligence to mountains, trees and rivers. Therefore, it shouldn't come as a surprise that we have a fascination with artificial intelligence. In 1818, Mary Shelley wrote about Dr Frankenstein, whose scientific experiments enabled him to create life. When he finally saw what he had created, he felt tremendous regret. Had he gone too far? Unlike Mary Shelley, Turing's interest in creating life was a lot more practically oriented: Turing mainly wondered when something could be considered life, intelligence or simply actual thinking. As such, the more advanced computer scientists among you will probably be familiar with the Turing test.

The Turing test was drafted by Turing in 1936 but was elaborated on in his 1950 paper "Computing Machinery and Intelligence." The Turing test can be summed up in the imitation game (not the movie, the thought experiment):

In the imitation game, a man and a woman must engage in conversation without being able to see each other. The man must pretend to be a woman, while the woman tries to prove that the man is not actually a woman. In the middle of the experiment, the man is switched with an artificial intelligence (e.g., a chatbot). If it remains just as difficult to unmask the man (now a robot), the artificial Intelligence has passed the Turing Test. In short, if you cannot say with certainty that you are talking to a robot, according to Turing, the robot is "intelligent."

One of the articles in this I/O Vivat was written by ChatGPT. What makes this article different from one written by a flesh-and-blood human being? Perhaps your answer will be "Nothing." And voila: we can conclude that according to the Turing test, ChatGPT is intelligent, as far as we can approximate.

Turing himself claimed that he expected that by the year 2000 computers would have about 100MB of memory and thus pass the Turing test with all ease. In hindsight, this prediction turned out to not be entirely true, but there is no denying that we are making great strides in 2023. Has Turing's prediction finally come true with software like ChatGPT and developments in Deep learning?

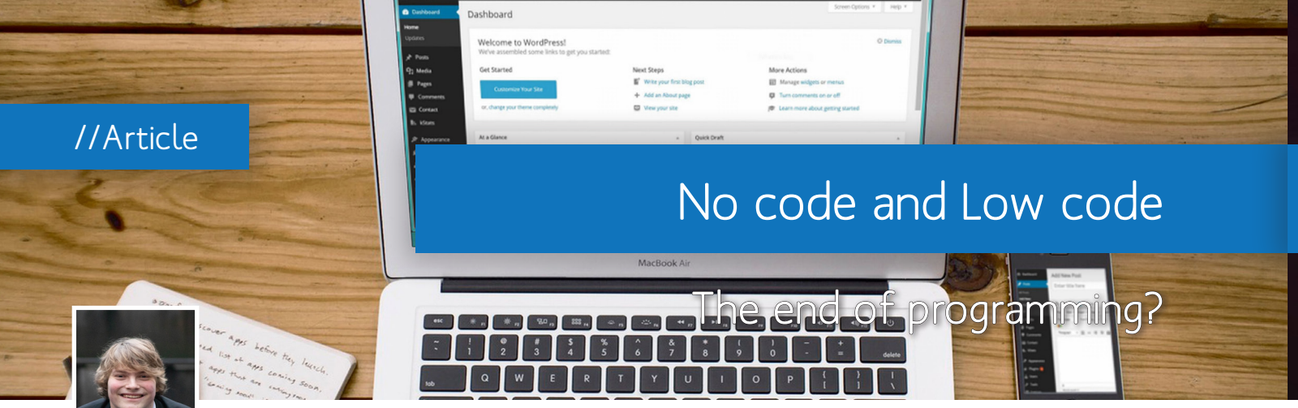
Heritage (of intolerance)

Turing's life has been invaluable to our field of computer science, as well as to the world at large. Not only would World War II have lasted much longer without Turing, but it could also have turned out completely differently. This man, this Alan Turing, is the father of our field. I believe that is something we can be incredibly proud of and inspired by.

The downside is that we see that even a genius like Turing was not above being a victim of the prejudice and small-mindedness of his contemporaries. We are fortunate today that we live in a more tolerant era, but it is still important for us to stay sharp. We must continue to question whether certain judgments are justified. It is important to remember that science is never isolated, but part of the more complex whole. We too have a piece of responsibility during and after our time in university that we carry with us. How do we shape our place in science

"We can only see a short distance ahead, but we can see plenty there that needs to be done"

-Allan Mathison Turing



//Article

No code and Low code

The end of programming?

By: Daan Wensink
Editor I/O Vivat

Writing computer code is the closest a person can come to modern magic. A few words and numbers go into the computer, and somehow the computer knows that the code is supposed to be a website, and everything works. It is often difficult to get into coding on a more professional level. A simple Google search for "how to write computer code" returns over 1.6 Billion results, all from different websites. This has meant that coding is mostly done by a group of people with prior experience with coding.

But recently, a new type of computer code has seen a rise in popularity: so-called "no code" or "low code" programs are programs that allow a user to design a system using a simple and often intuitive user interface, and then have the program generate the actual code for the system. No understanding of programming required. One of the most popular examples of this is WordPress, a product that allows users to create a website in the same kind of way as a PowerPoint presentation, and then have the actual website generated. In more recent years, WordPress has reported that 43% of new websites are created using WordPress.

This opens the door to a new future possibility: no code programs could replace ICT professionals in being the go to solution when a company needs new software. Much the same way as robots have replaced 23% of German manufacturing jobs since 1994, and predictions estimate another 8.5% of worldwide manufacturing jobs replaced by 2030. It is therefore good to investigate what happened with manufacturing jobs, because If we view the rise of no code and low code programs as a parallel to the rise of robots in factories, they could be a good way to predict what will happen to ICT jobs.

Manufacturing jobs

When we think of robots being used in factories, the first thing that comes to mind is often the increased efficiency and longer work hours, which make robots better workers as opposed to humans. Every robot in a manufacturing job replaces roughly 2 humans doing the same job. This means that robots can replace a large amount of humans in these types of 'vulnerable' jobs. But what does this mean for the humans who lose their jobs?

Counter intuitively, we find that robots taking over low quality jobs like manufacturing jobs results in better lives for the people that used to hold these jobs. The total quantity of work remains about the same but the composition of this work changes when robots replace lower paying jobs. In Australia, the introduction of machinery and robots has replaced a lot of jobs like crop and livestock farmers, secretaries and product assemblers, but these jobs have been replaced by jobs in sales, home care (both child and elderly) and manager positions. Which are often less physically hard and better paying jobs than the jobs they held before. However, there has been an increase in unemployment for the population aged 15 to 64 but this is more due to the increase in labor supply by the steadily growing population than the loss of total jobs. A division of hours into jobs shows that job opportunities remain the same or even grow.

What does this mean for computer scientists?

Although predicting the future is impossible, the comparison to the manufacturing jobs can help make an educated guess towards what this means for computer scientists. Some programming jobs will be lost because they are replaced, but new jobs will come to replace them. An obvious one is of course the job of maintaining the no code programs, as the programs still rely on code to function. But another less obvious example is to help compare the programs and tell the clients which one would suit their needs best, as an understanding of code and system design is very much still needed to design good systems. But the most important lesson learned is that jobs will always change and evolve, and therefore, it is good to stay flexible and never stop learning new skills.

References

https://google.com

https://wordpress.com

https://papers.ssrn.com/abstract=3039031

https://resources.oxfordeconomics.com/how-robots-change-the-world

https://onlinelibrary.wiley.com/doi/full/10.1111/1467-8462.12245

//Article

Predicting World Champions

Prediction model usage during sport events

By: Jelle Maas
Editor I/O Vivat

The FIFA World Cup, the quadrennial international soccer tournament, is one of the biggest and most highly anticipated sporting events in the world. With millions of fans tuning in from around the globe, the World Cup is a global phenomenon, attracting the attention of sports fans and non-sports fans alike. Given the global reach and high stakes of the event, it's no surprise that prediction models have become an important tool in forecasting the outcomes of World Cup matches.

The Elo rating system, developed by Arpad Elo, is a widely used prediction model for soccer. It is a method for calculating the relative skill levels of players in zero-sum games, such as chess. A zero-sum game refers to a situation where one player's gain is exactly balanced by the losses of the other player or players. In other words, in a zero-sum game, the total amount of resources in the game remains constant, and any gain by one player results in an equivalent loss by another player. The Elo rating system is based on the idea that a player's performance in a given game is a function of their skill level and the skill level of their opponent. The Elo rating system is used to predict the outcomes of matches and to rank teams and players.

Another popular prediction model for soccer is the FiveThirtyEight model, developed by Nate Silver. This model is a statistical one that uses a combination of data, expert knowledge, and machine learning algorithms to forecast the outcomes of sporting events. The model is named after the number of electors in the United States Electoral College and has been used to predict the outcomes of elections, as well as sporting events.

When comparing the Elo rating system and the FiveThirtyEight model, it is important to note that while both models are used to predict the outcomes of soccer matches, the Elo rating system is based on relative skill levels of players and teams, while the FiveThirtyEight model uses a combination of data, expert knowledge, and machine learning algorithms to make predictions. Both models are widely adopted by soccer fans and analysts and are often used to make informed bets on the outcomes of World Cup matches.

One of the key advantages of prediction models in soccer is their ability to provide objective and unbiased forecasts of the outcomes of matches. This can be particularly useful in situations where there is a high degree of subjectivity in the evaluation of teams and players, such as in soccer where human judgment (by referees) plays a significant role. Prediction models can also be used to identify trends and patterns in the sport, which can be used by teams to make informed decisions about strategies and tactics.

There are, however, limitations to the use of prediction models in soccer. One of the main limitations is that the models are only as good as the data that is used to create them. If the data is incomplete or inaccurate, the model's predictions will also be flawed. Additionally, prediction models do not take into account the impact of external factors, such as injuries, team dynamics, and weather conditions, which have a significant impact on the outcomes of soccer matches.

Despite these limitations, prediction models have become increasingly important in the world of soccer, and are being used by a range of stakeholders, including but not limited to coaches, players, and fans. Coaches, for example, may use prediction models to identify potential opponents and to devise strategies to counter their strengths and weaknesses. Players may use prediction models to assess their own performance and to identify areas for improvement. Fans, meanwhile, may use prediction models to make informed bets on the outcomes of matches or to simply satisfy their curiosity about who is more likely to win.

In conclusion, prediction models have become an important tool in the world of soccer, particularly in the context of major tournaments like the World Cup. Some of these prediction models also have become very sophisticated, such as the past twelve years of predictions by EA SPORTS FIFA which predicted the three World Champions correctly. Besides these predictions the models provide objective and unbiased forecasts of the outcomes of matches, and can be valuable in making informed decisions and identifying trends and patterns in the sport. Whether you are a coach, a player, or a fan, prediction models can provide valuable insights into the world of soccer.

References

https://www.historyhit.com/gaming/arpad-elo-rating-system

https://fivethirtyeight.com

https://www.ea.com/nl-nl/games/fifa/fifa-23/news/ea-sports-fifa-world-cup-22-prediction

Topicus

Interview with Martin & Irma

By: Hanna Gardebroek & Jelle Maas
Editors I/O Vivat

Martin Krans & Irma Veldman
Product Owner & Information Analyst

With the rise of automation, the general state of the world is becoming faster, bigger, and there are more opportunities for everyone. Even though you may not have heard of Topicus, it has become an important player in the market of automation. They have been seamlessly working in the background of many familiar sectors such as Education, Finance, Healthcare and Social Services. Indirectly, they have made changes for mostly every individual in the Netherlands. In the following conversation, we will speak with Martin and Irma to shine a light on the differences that Topicus makes for the betterment of society.

Could you please give an introduction of yourself and Topicus?
Martin: Topicus was founded in 1998. We started as a consulting company, and in 2002 we turned into a software company. I wasn't present at that time, because I joined in 2005 and back then there were only 30 to 35 people working here and now there are 1000(!). Luckily, it is still as informal as it was back then. I worked in the department making software for the financial sector (Finance) for sixteen years and then switched to the department focusing on software for the education sector (Education). Since the founding of Topicus, we have always been busy trying to shake up the market and change everything around us.

Irma: I started as a grad student in 2009 and stayed ever since! I started as an information analyst and have had many different roles. In the first years, I worked on a software product in Dutch healthcare that supports coordination between care providers. After that, I got the chance to focus on organizational changes within Topicus. Now, I work as a team lead on multiple teams in our educational division.

How do you challenge yourself within the company (after many years)?
Martin: We want the people working here to be able to really utilize their own talents, and we do not want to limit people's abilities to develop their own skills. We want your talents, and if you have a good idea you have the freedom to achieve it.

Irma: We want the people working here to be able to really utilize their own talents, and we do not want to limit people's abilities to develop their own skills. We want your talents, and if you have a good idea you have the freedom to achieve it.

Why does Topicus have so many departments (Education, Finance, Healthcare, and Social Services)?
In most cases, we often started very small within a domain and continued to expand it. After expanding, we support large parts of the value chain within a domain, or sometimes even the entire value chain. Our systems are not directly linked, but suppose someone advances from elementary school to high school and both schools use Topicus' system. Conveniently, there is no trouble for the user to switch. The reason that we operate in so many domains is because Topicus can identify both opportunities in new domains as well as overlap with domains we already operate. Initially, this gives us the opportunity to use existing software, code and/or components to start supporting a small part of the value chain in a new domain. In the long run, however, when we start supporting more and more of the value chain in that domain, the software usually evolves into domain-specific platforms, applications, modules, and components.

The advantage that we currently have with the number of departments is that we have a lot of different knowledge within one company and if we want to know something we knock on someone's door and receive the answer in three seconds.

Do you see the impact of your products in the day-to-day life of other people?
Martin: Yes, I can definitely see this in my day-to-day life. For instance, last year, my neighbors were moving to a new home. They told me that they were very happy that their mortgage application was handled quickly and smoothly. It turned out that they received a mortgage from one of Topicus' customers, who uses our software.

Irma: Another example is from my perspective as a parent. I worked on the application Parro which was frequently used at the school of my children. Parro is a system which provides communication between parents and elementary school teachers. Parents can mark their children as absent and teachers can easily organize events and update parents with the things occurring in their class. Due to a change in the school board, the school then chose a different application. I have seen what the competitors are doing and I much prefer our system but I might be biased. We usually measure the impact of our products using our social impact factor, or the number of times someone comes in contact with software that is built by Topicus. For example, if you attend primary school and enter our system, then go on to a high school which uses our software again, and then as an adult use our software during a mortgage application, this would mean a social impact factor of three. Currently, Topicus has an overall social impact factor of four. Topicus has a big impact on the everyday life of many citizens even though they may not see it every day because it happens behind the scenes.

In what way do students get in contact with Topicus?
Martin: They usually find us through our website or they know someone who already works for Topicus. It is important to us that students know how to reach us and can easily get in touch with us. Currently, we have a lot of graduate students working here on their bachelor's and master's theses and that is a cool thing because we learn a lot from the students as well. At Topicus you are not just somebody who is here to bring us coffee, you can contribute to the company.

Since many of us have been working in IT for quite some time, it is refreshing to see new students who can bring new perspectives, and it works both ways because we can teach them a thing or two from our own experience.

> "Always busy trying to shake up the market"

We find it important that the graduation assignments fit the person working on them, and if we see that this is not the case we try to find a way to fix this. Nothing is set in stone and there is a lot of freedom here to try to find something that suits you as a student.

What are your future plans for Topicus?
Martin: I have some personal challenges I would like to work on. I really want to develop good software to help support children in the special education segment. These are children who cannot keep up with regular primary education and need additional support and for people who are not in regular primary and high schools. I also want to take our team of product owners to the next level, because over time that team has grown a lot. There are currently many people from all walks of life and it is quite a young team. I have a lot of experience in that position and would love to give them some tips and tricks.

What makes Topicus special?
Irma: You get a lot of books and knowledge thrown at you during university, but at Topicus I really felt that I got the chance to use this knowledge. Not every task is the same as we were taught, but I could really see that the fundamentals of our education are being used in practice here. Practice makes perfect and since I continuously kept using the things I've learned, they stuck with me.

Martin: Topicus already had an excellent infrastructure when COVID hit, so it didn't take much effort to switch from 100% working at the office to working at home. When the lockdowns ended, Topicus decided that working at the office 100% of the time was a choice of the employee which gives employees the freedom to partially work from home as they see fit.

Another fun part of Topicus is your craft beer, what made you decide to start brewing and what does it taste like?
Martin: Our beer is called Gifkikker (poisonous frog) and it is inspired by La Chouffe. It was originally meant as a joke by one of our colleagues who asked one of our directors: "Hey, maybe we should brew our own beer sometime?". Our director just replied with "Why not?" and that was that. We started at a small brewery and it just grew larger and larger. It even turned out to be quite a good exercise for our marketing team on how to promote your products. The beer is delicious and you can even find it in the Vestingbar on campus sometimes!

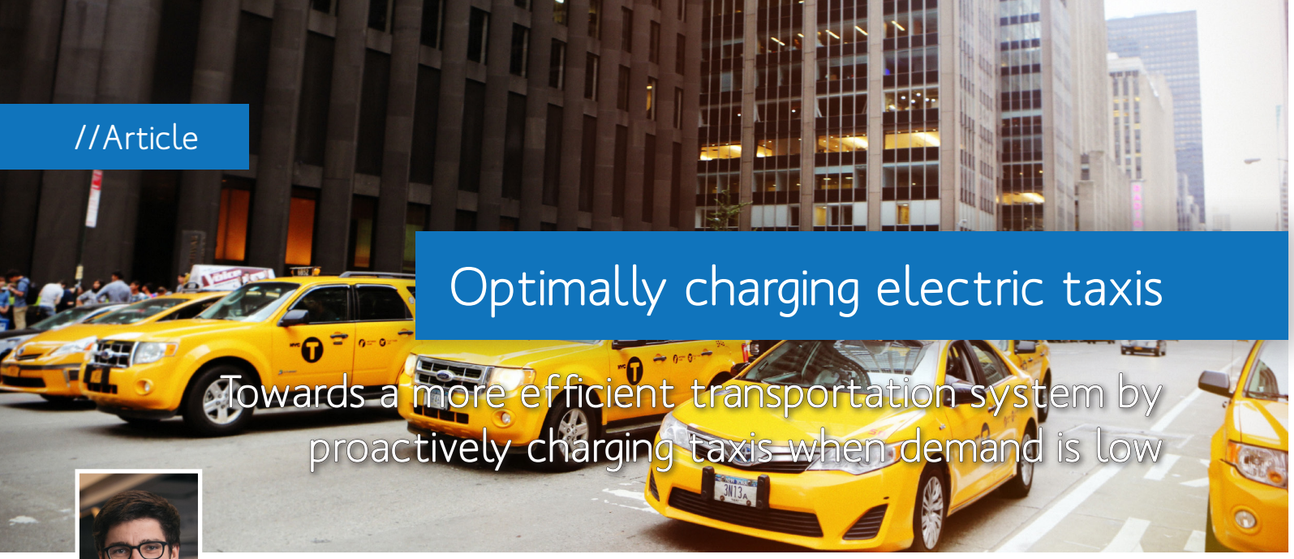
Impact on Society

Self-development while contributing to everyday live? Impact on society. Every day, millions of people use our platforms, including the people in your social environment. Tech solutions that really benefit healthcare, education, finance and the social domain.

Most of this from our 'home town' Deventer, where we are developing our own Topicus Campus. But also from Utrecht, Andelst, Groningen, Wageningen, Gouda and even in Vietnam and Romania. And our ambitions reach even further.

In order to arrive at new insights, we supervise around 100 students (from academic universities as well as universities of applied sciences) annually with their internship or graduation assignment. You get the chance to participate fully in one of our challenging projects immediately. This is where you will learn the fastest from your experienced team members and where you can really contribute to new ideas for challenging ICT projects. As a student at Topicus, you will take your skills to the next level and your efforts will make the lives of millions of Dutch people a little easier. We work closely with various educational institutions and universities and know most about the boundaries of your studies. We also bring you into contact with fellow students so that you are never alone.

Curious? Then check out werkenbijtopicus.nl/studenten for all the stories and the current overview of our (student) vacancies.



//Article

Optimally charging electric taxis

Towards a more efficient transportation system by proactively charging taxis when demand is low

By: Jesper Provoost
Nominee ENIAC Thesis Award 2021-2022

What if we could decrease waiting times for taxis in the city, while at the same time reducing peak loads on the electricity grid? Jesper Provoost is behind a project that aims to make charging for electric taxis more efficient by charging them when there are temporarily no passengers willing to be transported.

Jesper graduated in 2021 as a MSc in Computer Science at the University of Twente. He executed and wrote his Master's thesis at the Integrated Transport Research Lab (ITRL) of KTH Royal Institute of Technology in Stockholm, Sweden. In this interview, Jesper shares insights about the project that he has worked on as a part of his thesis.

What is the thesis about?
A common problem in ridehailing and taxi systems is that the drivers are making the decisions of repositioning and charging in an individual manner, which means that they can decide themselves and don't have the information or the system wide benefits in mind when they make these decisions.

With predictive charging, we can utilize proactive knowledge and optimize for the system wide benefits instead of the individual, "greedy" benefits of the drivers. In this research, I therefore developed an algorithm called ITX (which stands for Idle Time Exploitation) to do this predictive charging. The algorithm bases its decisions on machine learning models which can predict how long an empty taxi will remain without passengers. Based on this information, we can exploit the idle time to charge the vehicles, after which they can be repositioned and ready to serve new passengers. I proved the effectiveness and benefits of the algorithm by running simulations of the real world road network of Manhattan with thousands of vehicles.

What is the main contribution of your work?
Most existing approaches to intelligent charging control work do not combine time, location and duration into a comprehensive control algorithm or are unsuitable for real-time operation in large networks. In my thesis, I therefore present a real-time predictive charging method for electric taxi and ridepooling services. The ITX algorithm predicts the periods where vehicles are idle and exploits these periods to harvest energy. It relies on Graph Convolutional Networks (GCNs) and a linear assignment algorithm to devise an optimal pairing of vehicles and charging stations, maximizing the exploited idle time. My approach works on large-scale graph representations of the road network and enables fine-grained decision-making.

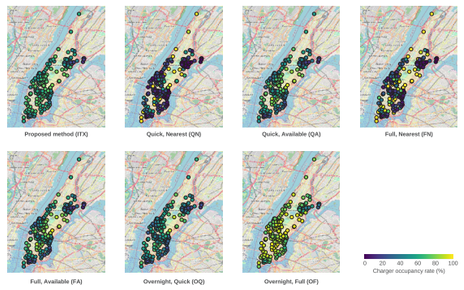


Figure 1: The proposed ITX algorithm compared to other algorithms

Overall, this thesis makes four key contributions, namely:

1) Optimizing decisions regarding charging behaviour, which involves when, where and for how long the vehicles should charge. As of today, there does not exist a method which integrally optimizes the three decision variables by exploiting predicted idle times, simplifying the optimization task into a problem which can be solved in polynomial time. The proposed ITX methodology is shown to reduce operational costs from an operator's perspective while also reducing delays and improving comfort levels of passengers.

2) Providing proactive charging opportunities which facilitate energy harvesting during the day. The resulting energy reserves can be used to satisfy peak demands more smoothly. Additionally, the power levels drawn from the grid are shown to be more stable over time and space, alleviating the burden on electricity infrastructure which paves the way for more sustainable use of the electrical grid.

3) Ease of integration with other operational decisions, i.e., dispatching and repositioning, by letting control processes occur in modular and sequential fashion. This ensures faster runtimes (at the expense of global optimality), but also flexibility and robustness as the ITX method can be used alongside existing dispatching and repositioning frameworks.

4) Real-time and predictive decision-making capabilities at the scale level and granularity of a large real-world city, with 6,000 vehicles and 8,500 roads. Simulation and policy generation are performed on graph representations of a road network, which benefits transferability of the system.

What were the results?
I evaluated the approach through extensive simulation studies of New York City. As a reference, multiple baselines were devised with varying levels of complexity. The results demonstrate that ITX outperforms all baseline methods by at least 5% (equivalent to $70,000 of savings per week for a 6,000 vehicle operation) expressed using a monetary reward function which was modeled to replicate the profitability of a real-world taxi system. Moreover, ITX reduces passenger delays by at least 4.68% compared to baseline methods and increases passenger comfort by facilitating a better spread of customers across the fleet. The results also demonstrate that ITX enables vehicles to harvest energy during the day, stabilizing battery levels and increasing resilience to unexpected surges in demand. Lastly, compared to the best-performing baseline strategy, peak loads are reduced by 17.39% which paves the way for more sustainable use of the electrical grid.

How do you see this being used in the "real world"?
Generally speaking, ITX can be used as an algorithm which provides us with system-wide charging schedules. In practice, this could look like a navigation app for drivers, for instance in a system like Uber or Lyft where drivers can follow the charging path that the system suggests to them. This could resemble Google Maps, where instead of showing the way from A to B we show a route to a charging facility that our system has selected for that particular driver. If you coordinate this centrally with the algorithm that is proposed in my thesis, it will provide large benefits for both the driver and the service operator.

The system is also very suitable for autonomous vehicles. When they are idle and don't carry passengers, they can decide to charge according to the decision made by the algorithm. Obviously with autonomous vehicles it is easier to let the vehicles follow the exact instructions (where, when and for how long to charge), and hence it will be easier to fully utilize the fine granularity with which the ITX algorithm can predict and compute charging schedules.

What is the next step for this research?
I would like to incorporate the interconnection between charging and repositioning into this research. So far, the main strength of the ITX algorithm is that it selects and requests an 'optimal' charging facility for the time that a vehicle is predicted to remain without passengers. However, it does not yet fully take into account whether that charger will be busy in the future and whether the area of the charging facility is attractive in terms of demand. For this reason, it would be best to consolidate the charging and repositioning processes into a single decision-making process. However, the more dimensions you add to the decision-making problem, the more computationally expensive the algorithm becomes. As a consequence, it will be interesting to expand the ITX method with other scalable (meta)heuristic algorithms that can build a bridge between large-scale charging and repositioning processes.

What is the take home message?
Predictive and proactive charging strategies like the one proposed provide many benefits in terms of transport and research efficiency compared to more conventional and egoistical methods. It can result in large benefits for customers but also operators. When using the ITX algorithm, we can cut down on delays while increasing passenger comfort. In the simulations, 95.78% of customers arrive with a delay of less than 5 minutes, which is 6.5% higher than the best-performing baseline. The increased operational efficiency also comes with increased profitability for the operator of the taxi or ridehailing service. For example, in a New York scenario, the algorithm could yield a weekly increase in profits of $70,000 for a fleet of 6,000 vehicles. Additionally, the results suggest that ITX is able to conserve more stable energy levels throughout the fleet, therefore providing better reserves during periods where demand is particularly high. Lower peak loads alleviate the burden on the electrical grid infrastructure, providing benefits to the grid operator while also yielding better energy-efficiency for the overall transport system. In the future I think this will be a very good step towards reaching crucial sustainability goals.

About Jesper

Jesper Provoost (1998) is a researcher in the field of transportation who completed his MSc degree in Computer Science (cum laude) at the University of Twente. During his studies, he spent a year working as a visiting researcher at KTH Royal Institute of Technology in Stockholm, Sweden. Here he worked on predictive real-time repositioning and charging algorithms for ridesharing fleets, which was also the topic of this Master's thesis. Next to his studies, Jesper has previously worked as a part-time Data Scientist at DAT.mobility and as a Teaching Assistant in Data Visualization at the University of Twente.

Currently, Jesper works as a PhD student at the Delft University of Technology. His current research focuses on the development and application of an innovative demand management system in the form of tradable mobility credits. Jesper's interests are located on the intersection between simulation-based optimization, operations research and predictive modeling applied to transportation systems.

Sources used during this research can be found in the thesis as published by Jesper. The thesis is available at https://essay.utwente.nl/88681/.

//Column

Programme Director TCS

Perfectly balanced, as all things should be

By: Vadim Zaytsev
Programme Director Technical Computer Science

Unless you are a certain sports gear producing company, you cannot *just do it*. Any human, initiative or organisation needs a big idea supporting them and driving all their activities. As the programme director, I am often asked what sets our programme apart from others in the field. One aspect that I am particularly proud of is the balance that we strive to achieve in our curriculum.

At the heart of our TCS programme is the concept of RED: Research, Engineering, and Design. Has always been, but I have started to use these three labels more since I have realised how important they are for this programme, both for teachers and students. We believe that only dedication to all these three pillars provides a well-rounded foundation for a successful career in computer science. By focusing on these areas, our students are able to develop a broad set of skills that allows them to approach problems from multiple angles and come up with innovative solutions.

Research is an essential part of any computer science programme, as it allows students to stay up-to-date on the latest developments in the field and to make their own contributions to the field through original research projects. Our programme's emphasis on research materialises in opportunities for students to apply cutting edge methods and tools to real-world problems and to present their findings to others. Those who are interested exclusively in research, are better off studying elsewhere, where their focus is entertained enough. Those who can acknowledge all three while having a preference for research, will find themselves working in R&D departments or becoming PhD or EngD students working on cutting edge projects of industrial significance.

Engineering is the practical application of computer science principles to the development of software and hardware systems. Our programme has always provided students with a strong foundation in engineering principles and practices, covering algorithms, data structures, computer and networking architecture, programming languages, etc. This foundation allows the graduates to excel in a wide range of engineering roles, whether they are building software applications or focusing on hardware. Universities of applied sciences are usually advertised as producing strong engineers, which is true, yet their best and most successful graduates are those with broader interests and numerous hobbies. At TCS in Twente you grow as an engineer while also learning the concepts behind the language constructs and patterns, and get accompanying skills that make you employable.

Design is an often overlooked aspect of computer science, yet an essential part of creating effective technological solutions that are both user-friendly and developer-friendly. Our programme has always placed a strong emphasis on design, teaching students how to architect new systems, connect existing libraries, and create user experiences that are intuitive and enjoyable. By incorporating design into their work, our graduates are able to create technology that is not only functional, but also aesthetically pleasing and robust enough to be functioning for many years. Graduates are aware of the complexity of the software world and equipped with skills to architect highly nontrivial systems of the future.

By focusing on RED, our programme provides a balanced curriculum that prepares students for a wide range of careers in computer science: whether they are pursuing roles in R&D, aiming to be software developers or software architects, our graduates are well-equipped to succeed in their chosen fields. Goes without saying that we also encourage them to deepen their knowledge even further and choose one of our Master specialisations… Whatever the chosen path might be, I am confident that the balance provided by our RED-centric curriculum has set them up for successful careers.

About Vadim

Dr. Vadim Zaytsev finished Telematics, which now has been absorbed into the CS master as "Internet Science & Technology" specialisation, at UT as his second Master degree. He got his PhD title at the Free University (VU) in Amsterdam. Besides teaching, he has been an active researcher, and published many academic papers on topics such as software engineering and modelling, modernisation and migration of legacy software, transformational and generative techniques, and software languages and grammars, earning the nickname "grammarware". He worked several years in the industry, developing new compilers, some of which were acknowledged by Microsoft Tech awards.

Since 2020, Vadim is back at UT as an associate professor of software evolution. In 2021, he also won the Inter-*Actief* Decentralised Educational Award (IDEA). Vadim has a cat named Viking.

//Article

Linux File Shenanigans

By: Ruben Groot Roessink
Editor I/O Vivat

During my studies, I made a shift from Windows to Linux (Ubuntu) as the Operating System that I use privately. I swore off ever using Windows again, because I fell in love with the simplicity of the Linux file system. Unfortunately, even an ethical hacker (my current job) has to use Windows to write reports and as a target Operating System, because basically every company uses Windows for its day-to-day business. Nonetheless, I would like to share some things that I have learned over the years regarding the Linux file system.

Linux, or actually GNU/Linux (if you ask Richard Stallman), is a family of operating systems based on the Linux kernel, packaged with system software and libraries (mostly from the GNU project). In Linux, similar to other UNIX(-like) operating systems, input/output to and from resources such as documents, hard-drives, modems, keyboards, printers, etc. is handled as simple streams of bytes exposed through the file system.

"Everything is a file" is a common way to describe the philosophy behind UNIX and Linux (although not entirely correct). The statement is true in the sense that Linux does not differ between files and directories (directories are just a special type of file, pointing to other files). And, as mentioned before, devices and such are handled as part of the file system. Of course, this does not mean that processes are part of the file system.

Let us have a closer look. I used my own Kali Linux distribution to create the images in this article.

mkdir

First, we'll make a directory for our tests using the command `mkdir` (make directory) (Figure 1).

```
┌──(kali㉿kali)-[~]
└─$ mkdir vivat

┌──(kali㉿kali)-[~]
└─$ cd vivat

┌──(kali㉿kali)-[~/vivat]
└─$ pwd
/home/kali/vivat
```

Figure 1: Making a new directory

Although this command is quite clear, through numerous encounters with making new directories during penetration tests, I discovered that it has one fatal flaw. (Almost) everytime I have to make a new directory, I find myself having to change directory (`cd`) to the new directory directly after. To combat this (first world) problem, I defined my own `mcd` (make and change directory) as follows:

```
mcd(){ mkdir -p $1 && cd $1; }
```

Executing the command above directly shows the effect as intended, as our current working directory is instanly changed to the newly created directory after executing the command (as can be seen in Figure 2):

```
┌──(kali㉿kali)-[~/vivat]
└─$ mcd vivat_subdir

┌──(kali㉿kali)-[~/vivat/vivat_subdir]
└─$
```

Figure 2: Making and entering directory

touch

To be able to further demonstrate the simplicity of the Linux file system we need to add some dummy files. We can files in several manners, of which I describe three below:

The first command is the `touch` command, which only adds the new file (with the name as specified when issuing the command). While this `touch.txt` file is useful enough for my explanation on the Linux file system, the file does not have any contents.

Another way of creating a new file is redirecting text on the terminal to a file using the angle bracket (`>`). By writing the intended contents of the file to the terminal output (using the `echo` command) and redirecting the output to a file, we can add contents to a newly created file.

It is also possible to redirect output using two angle brackets (`>>`) (which appends to instead of overwrites the file specified).

```
┌──(kali㉿kali)-[~/vivat]
└─$ touch touch.txt

┌──(kali㉿kali)-[~/vivat]
└─$ echo 'Test' > redirection.txt

┌──(kali㉿kali)-[~/vivat]
└─$ la -alah
total 16K
drwxr-xr-x  3 kali kali 4.0K Feb  5 13:15 .
drwxr-xr-x 23 kali kali 4.0K Feb  5 13:14 ..
-rw-r--r--  1 kali kali    5 Feb  5 13:15 redirection.txt
-rw-r--r--  1 kali kali    0 Feb  5 13:14 touch.txt
drwxr-xr-x  2 kali kali 4.0K Jan 29 16:50 vivat_subdir
```

Figure 3: File creation

As an added bonus: an angle bracket the other way around (`<`) redirects the contents of the file to the terminal as input.

There are, of course, many other ways to create files.

ls

Moving on, the `ls` command can be used to *list* (parts of) the filesystem (similar to `dir` on Windows). Almost every time, I use the following flags, for the best readability.

`-l` to display *lengthy* output, including the file permissions

`-a` to includes *all* files in the output of the command, including 'hidden' ones (files starting with a `.` on Linux)

`-h` to display *human-readable* format for file sizes (uses K, M etc. to abbreviate the file size, instead of showing all bytes)

Using the `-l` flag, we list permissions of files (and directories) test file hierarchy. These are displayed using the (example) string `-rw-r--r--`. Admittedly, it took me way too long to understand the permissions string. A similar string is used to display the file permissions on Windows, but much less readable and understandable (and much lengthier).

The first character in the permissions string annotates the *type* of the 'file'. The character `d` is used to specify a special type of 'file', a *directory* (or a file 'containing' other files). Normal files are instead annotated using a dash (`-`). Links to other files (similar to Windows' *shortcut*) are annotated using a lowercase `l`.

The 'type' character is followed by three triads of three characters each. `rw-` and `r--` (twice) respectively in the example. The three characters specify 'read' (`r`), 'write' (`w`) or 'execute' (`x`) permissions for different groups of users. The first triad specifies the permissions of the *user owner* of the file (the first *kali* in the Figure 4). The second triad specifies the permissions of the *group owner* of the file (the second *kali* in Figure 4).

The third triad specifies the permissions of all other users on the file (or directory) specified. A dash (`-`) signals that a permission is turned off, an `r`, `w` or `x` specifies that a permission is turned on. The root user is not affected by file permissions and has all permissions on all files (and directories) in the file hierarchy.

Some example file permission strings are:

`-rw-r--r--` means that only the user owner of the file can read and write to the file, while others can only read the file

`-rw-rw-rw-` means that everyone can read and write to the file

`-rwxr--r--` means that only the user owner can read, write and execute the file. Other users can only read the file

`drwxr-xr-x` describes the default permissions for a new directory. The user owner of the directory can read the directory, write to the directory and read the contents of the directory. The other users can only read the directory and read the contents of the directory (specified with an `x` for directories as directories are never executable).

chmod

```
┌──(kali㉿kali)-[~/vivat]
└─$ ls -lah redirection.txt
-rw------- 1 kali kali 5 Feb  5 13:15 redirection.txt

┌──(kali㉿kali)-[~/vivat]
└─$ chmod 777 redirection.txt

┌──(kali㉿kali)-[~/vivat]
└─$ ls -lah redirection.txt
-rwxrwxrwx 1 kali kali 5 Feb  5 13:15 redirection.txt
```

Figure 5: `chmod` usage

In order to change the permissions of a file, one can use the `chmod` (change mode) command (Figure 5).

The command lets us specify the file permissions in two ways, namely one in which we specify the groups of users and which permissions to add or to remove:

`chmod g+w {file}` to add write permissions for the group owner of the file.

`chmod a+x {file}` (equal to chmod +x {file}) to give all three groups of users the execute permission

`chmod u+x {file}` to give execute permissions to the user owner

`chmod o-rwx {file}` to remove read, write and execute permissions for all other users

It is also possible to specify permissions using their numeric values, completely overwriting the previous values. This command uses the numerical representations of read (4), write (2) and execute (1) permissions to specify the permissions. For example, `chmod 777 {file}` would give read, write and execute permissions to all users, while `chmod 600 {file}` would only give read and write permissions to the user owner of the file.

chown

Another command to change the permissions certain users/groups have on a file is the `chown` command (change ownership) as it changes the user owner and/or the group owner of the file to another user and/or groups (Figure 6). By default, file permissions persist when changing ownership (which only the root user is capable of). The following command and parameters are used to change file ownership: `chown {user owner}:{group owner} {file}`

```
┌──(kali㉿kali)-[~/vivat]
└─$ ls -lah redirection.txt
-rw------- 1 kali kali 5 Feb  5 13:15 redirection.txt

┌──(kali㉿kali)-[~/vivat]
└─$ sudo chown www-data:www-data redirection.txt

┌──(kali㉿kali)-[~/vivat]
└─$ ls -lah redirection.txt
-rw------- 1 www-data www-data 5 Feb  5 13:15 redirection.txt
```

Figure 6: `chown` usage

```
┌──(kali㉿kali)-[~/vivat]
└─$ ls -la
total 16
drwxr-xr-x  3 kali kali 4096 Jan 29 16:55 .
drwxr-xr-x 23 kali kali 4096 Jan 29 16:45 ..
-rw-r--r--  1 kali kali    5 Jan 29 16:55 redirection.txt
-rw-r--r--  1 kali kali    0 Jan 29 16:54 touch.txt
drwxr-xr-x  2 kali kali 4096 Jan 29 16:50 vivat_subdir
```

Figure 4: Example of output of `ls` command (including permission strings)

Special file permissions (SUID, GUID & Sticky Bit)

In addition to the read, write and execute permissions, which are most common, Linux distributions have several 'special' permissions. For example, when looking at the binary executable of the passwd (Figure 7) command, one can see an `s` at the execute permissions of the user owner in the place where one would expect an `x` or a `-`.

This special permissions is called the SUID bit, which signals that this file is always executed with the permissions of the user owner of the file (root in case of the `passwd` binary). This special permissions is useful as this allows other users to change their password, which requires write permissions to the `/etc/passwd` or `/etc/shadow files`. As these files are owned by the root user, the SUID bit allows other users to change their password in the context of the root user. While many legitimate uses exist for the SUID bit, some binaries might also allow a normal user to escalate privileges to the root user, so be careful when setting the SUID bit. Example binaries that allow for privilege escalation in this way can be found on: `https://gtfobins.github.io/#+suid`

The SGID is similar to the SUID bit, but is mainly used on directories. Setting this permission ensures that all sub directories and files inside the directory will get the same group ownership as the main directory (useful for web directories and such).

The final, special, permission is the so-called sticky bit (annotated using a `t` instead of an `x` in the 'other users' group) and only used on directories. This directory permission ensures that all files in the directory can only be deleted or renamed by the file owner (and the root user). This ensures that other users cannot delete files belonging to other users in a shared directory (e.g. the `/tmp` directory on Linux).

```
┌──(kali㉿kali)-[~]
└─$ ls -lah /usr/bin/passwd
-rwsr-xr-x 1 root root 67K Nov 11 09:28 /usr/bin/passwd
```

Figure 7: 'Special' permissions `passwd` executable

Standard I/O streams

When using the terminal (a shell) on Linux, input and output are handled as streams of characters. The following three standard file streams exist:

`stdout` (or standard out) displays output from commands (with file descriptor 1)

`stderr` (or standard error) displays error output from commands (with file descriptor 2)

`stdin` (or standard in) displays input for commands (with file descriptor 0)

Manipulating these streams is quite easy and also very useful. For example, the following command returns a lot of errors (Figure 8).

```
find / -type f -iname "touch.txt"
```

We use the find command to find the touch.txt file that we created earlier in our file hierarchy. We use a slash (`/`) to search in the root directory and all its sub directories. Additionally, we only want to have files (`-type f`) returned (and thus omit directories matching our search from the search results). Using the flag `-iname`, we can specify a (case-insensitive) file name.

```
┌──(kali㉿kali)-[~/vivat]
└─$ find / -type f -iname "touch.txt"
find: '/run/udisks2': Permission denied
find: '/run/lightdm': Permission denied
find: '/run/user/1000/systemd/inaccessible/dir': Permission denied
find: '/run/sudo': Permission denied
find: '/run/openvpn-server': Permission denied
find: '/run/openvpn-client': Permission denied
find: '/run/cryptsetup': Permission denied
find: '/run/credentials/systemd-tmpfiles-setup.service': Permission denied
find: '/run/credentials/systemd-tmpfiles-setup-dev.service': Permission denied
find: '/run/credentials/systemd-sysctl.service': Permission denied
find: '/run/credentials/systemd-sysusers.service': Permission denied
find: '/run/systemd/propagate': Permission denied
find: '/run/systemd/unit-root': Permission denied
find: '/run/systemd/inaccessible/dir': Permission denied
find: '/run/initramfs': Permission denied
```

Figure 8: Execution of command without error redirect

Executing this command the first time, ensures that we receive many errors (Figure 8), because the command executed also tries to read many files in directories which our current user does not have permissions to (*Permission denied*). We can manipulate the standard error stream to write its output to `/dev/null` (an empty void) by specifying the following command:

```
find / -type f -iname "touch.txt" 2>/dev/null
```

This redirects the error output (2) to the `/dev/null` file, just as we saw earlier with the output redirection brackets (`>`) (Figure 9).

```
┌──(kali㉿kali)-[~/vivat]
└─$ find / -type f -iname "touch.txt" 2>/dev/null
/home/kali/vivat/touch.txt
```

Figure 9: Result of file stream manipulation

Conclusion

While I barely scratch the surface of the complexity of the Linux Operating System, with this article, I hope to have given you an insight into several aspects of Computer Science that, at least for me, remained untouched during University.

References

https://en.wikipedia.org/wiki/
- Linux
- GNU Project
- Everything is a file
- Unix

https://boinc.berkeley.edu/wiki/Linux_file_permissions

https://linuxhandbook.com/suid-sgid-sticky-bit/

https://gtfobins.github.io/#+suid

https://developer.ibm.com/tutorials/l-lpic1-103-4/

CHOOSE AMBITION / CHOOSE GROWTH / CHOOSE EXCELLENCE

What are you going to change?

First people, then technology

Even though Nedap is a technology company at heart, our strength is that we always put people first. ‘First people, then technology’ is what we call it. We do so by observing people – and what they need to perform better – before we start developing technology.

We observe. We create. We scale.

At Nedap, design and technology work closely together to meet people’s professional needs. We create elegant solutions that are easy to use and beautiful to look at. These solutions enable interactive shopping experiences from Adidas to H&M, and provide security for major landmarks such as the Eiffel Tower and the Burj Khalifa. We help farmers take care of their livestock and even find you a place to park your car in Amsterdam.

People. Culture. Leadership.

‘First people, then technology’ does not only apply to the markets in which we operate, it also applies to our own organization. We invest in developing talent. At our beautiful campus in Groenlo we challenge ‘Nedappers’ to realize their ideas and ambitions and to achieve more than they could ever imagine. At Nedap, you are challenged to take the lead from day one.

What we find important

At Nedap you get all the space you need, literally and figuratively. But we also expect you to do something with it. We believe in the power of ideas and the energy of your own initiative. And so our entire organisation is designed to do just that.

We work with the relevant people to resolve specific issues. Since we often do this in small, targeted teams, you may sometimes get the impression that you’re working with a small organisation at Nedap. However, the extensive options and resources that you can find only in a large organisation will always be available to you. So, we’re actually big and small at the same time.

Working at Nedap is being part of a flock. You have the power to change your course and to inspire others to follow. We work together on our fantastic campus in Groenlo in various business units and teams, and we share a common ambition. Together we develop technology that helps people to be more productive and successful in their work.

Start your journey: www.nedap.com/students

nedap