Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Dashboard for secondary monitors with userbased automation and user bahavior

Bc. Tomáš Hodek

Supervisor: Ing. Jiří Šebek
Field of study: Open informatics
Subfield: Software engineering
May 2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Hodek**	Jméno:	**Tomáš**	Osobní číslo:	**456948**
Fakulta/ústav:	**Fakulta elektrotechnická**				
Zadávající katedra/ústav:	**Katedra počítačů**				
Studijní program:	**Otevřená informatika**				
Specializace:	**Softwarové inženýrství**				

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Dashboard pro sekundární monitory s automatizací na základě uživatele a jeho chování

Název diplomové práce anglicky:

Dashboard for secondary monitors with userbased automation and user behavior

Pokyny pro vypracování:

Mnoho lidí dnes využívá sekundární monitory pro různé účely. Velmi často pro přehled
aplikací, které nevyužívají často a spíš jen sledují dění, než aby interagovali s aplikacemi na
sekundárním monitoru.[1] Pomocí agregace aplikací, či jejich veřejných API s kombinací
uživatelských pravidel, sledování uživatele a jeho chování by mělo být možné vytvořit
automatizovaný dashboard obsahující uživatelem zvolené aplikace, s plně automatizovaným
chování, který poskytne dokonalý přehled bez nutnosti ručního zásahu uživatele.[5]
Cílem práce je:
1. Zanalyzovat využití sekundárního monitoru s možnostmi přehledů jednotlivých
aplikací.
a. Navrhnout možnosti využití veřejných API, či přímo jednotlivých aplikací
v dashboardu.
2. Seznámit se s možnostmi sledování uživatele a jeho chování, jako například eye
tracking a jiné metody.[3]
3. Navrhnout možnost předdefinovaní pravidel samotným uživatelem a k nim možnosti
chování automatizace dashboardu, či jednotlivých zobrazovaných aplikací.
4. Naimplementovat dashboard aplikaci integrující vícero aplikacích, s možnosti
definování rozložení aplikací, pravidel automatizace a chování.[2]
a. V rámci implementační části naimplementovat využití eye trackingu pro lepší
automatizaci.
5. Otestovat aplikaci na uživatelích, analyzovat využití a případné nedostatky návrhu
oproti využívání sekundárního monitoru bez této aplikace.

Seznam doporučené literatury:

[1]CERNY, Tomas; a Michael Jeff DONAHOO. Second Screen Engagement of
Event Spectators: [2]Advances in Human-Computer Interaction. Waco, TX 76706,
USA, 2018. Research Article. Computer Science, Baylor University.
[3]DUCHOWSKI, Andrew T. Eye tracking methodology. Theory and practice, 2007, 328.614: 2-3
[4]POOLE, Alex; BALL, Linden J. Eye tracking in HCI and usability research. In: Encyclopedia of human
computer interaction. IGI Global, 2006. p. 211-219.
[5]BADII, Atta, et al. SAM: Dynamic and Social Content Delivery for Second Screen Interaction. In:
Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video.
ACM, 2015. p. 119-124.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jiří Šebek, kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **11.02.2020** Termín odevzdání diplomové práce: **22.05.2020**

Platnost zadání diplomové práce: **30.09.2021**

Ing. Jiří Šebek podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.

Datum převzetí zadání	Podpis studenta

Acknowledgements

I would like to thank my supervisor Ing. Jiří Šebek for providing support during writing and also my classmates for valuable advices and psychic support during those rough times.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 20. May 2020

Abstract

The usage of multiple monitors for increasing effectivity of the work is common for many users nowadays. Part of this thesis is an analysis of the usage of secondary monitors and how the applications are used on them. The goal is to use the results from the analysis for designing an application, which would provide a more organized environment of the secondary monitor and by that, increase the effectiveness. The design aims to offer customizable features such as creating own persistent applications layout, the ability to add custom applications or select from existing ones, and defining rules that would use user context for automatization of the application. The final implementation, done accordingly to design, should be modular, extendable, and include all mentioned features to increase the effectiveness of work and ease the usage of secondary monitors. The overall idea and design were evaluated together with users testing the implemented prototype.

Keywords: dashboard, user-context, hci, multi-monitor, layout, monitor, user-tracking

Supervisor: Ing. Jiří Šebek

Abstrakt

Využívání vícero monitoru pro zefektivnění práce je v dnešní době běžné u mnoha uživatelů. Součástí této práce je analýza využívání sekundárních monitorů a jakým způsobem jsou na nich využívány jednotlivé aplikace. Cílem je využít výsledků analýzy k navržení aplikace, která by umožnila uživateli mít organizovanější prostředí sekundárního monitoru a tím zvýšit efektivitu. Návrh cílí na poskytnutí nastavitelných vlastností jako je vytvoření vlastního persistentního rozložení aplikací, umožnění přidání vlastních aplikací nebo výběr z existujících a definování pravidel, díky kterým bude možné provést automatizaci aplikace s využitím kontextu uživatele. Výsledná implementace aplikace dle návrhu by měla být modulární, rozšířitelná a obsahovat zmíněné vlastnosti za účelem zefektivnění práce a zpříjemnění používání sekundárních monitorů. Celková myšlenka s návrhem byla v rámci práce evaluována s uživateli pomocí prototypu.

Klíčová slova: dashboard, kontext uživatele, uživatel, hci, duálni monitor, sledování uživatele

Překlad názvu: Dashboard pro sekundární monitory s automatizací na základě uživatele a jeho chování

Contents

Figures

Tables

Chapter 1
Introduction

This diploma thesis dives into the problematics of using multiple monitors, specifically the usage of secondary monitors, then more deeply into the usage of applications on secondary monitors and automatization of the whole workflow with secondary monitors. The main factor being taken in mind and connected to the applications often used on secondary monitors is the logical layout of the application and overall organization of the secondary monitor‘s screen.

Those factors will be analyzed, separately in the chapter 2 and as a whole in the chapter 4. The Design and Implementation chapters will dive more deeply into the detailed and specific problems and will be then evaluated in the Testing chapter of the thesis. The goal is to present a solution for an application for secondary monitors that would help user be more efficient and productive during his work. The idea will be tested and evaluated in the Implementation and Testing chapters of this thesis.

1.1 Motivation

It has become a standard to use more than just one monitor for people who are working with computers, nowadays even with tablets or some advanced smartphones. Basically, people are able to connect one or more monitors to any suitable device. Speaking of work spaces, in most companies, employees are usually offered second or even third monitor by standard, coming up with company laptops or a standard workstation. In 2017 the number of people switching to the dual or more monitor setup went from 20 % to 90 % in 2002.[1] And this trend is increasing in almost every field where computers are used. From engineers, designers, gamers, and programmers to science workers, office workers, accountants, lawyers, writers, and many more.

The reason behind those numbers and wide spreading of multi-monitor setup is simple. Users can be more productive, fast, and even feel more confident during their work. Using multiple monitors can provide a better environment for multitasking which leads to increased productivity and faster work.[4] As laptops are getting smaller for comfort and portability, the screen size is getting smaller, therefore user might feel frustrated to work only on their laptops all the time. So it is common to have a second monitor at the workplace to connect the laptop and enhance the workplace by more screen size. Some users have even more monitors, the extreme ones might have up to 6 monitors to have access to all the information they need. Those extreme setups can be often seen at business offices or some monitoring centers, where employees need to have an overview of many things at the same time, like stock exchange prices or critical systems status.

Many studies regarding the usage of multiple monitors have been made. Most of them are in consensus, that the multi monitor setup provides more productivity. Not all studies are having the same results in efficiency increase ration. In one study, participants agreed that working with multiple monitor was easier and has helped them find all the information much faster.[2] In other studies, they found out that the productivity can be increased by up to 50 % with multiple monitor setup. The benefits are not only in productivity but the enjoyment of work while using multiple monitors increases as well. Mainly because it can be tiring and frustrating to have a constrained view over the work and repeatedly switching the windows, which is taking time and effort.[1] the next advantage of using this setup is the reduced number of errors users make during completing their tasks. So the results are better than using a single monitor setup.[3] The results have also shown that there are improvements regardless of the screen size. Meaning that any second monitor can help to boost the performance and effectiveness of work.[6]

The numbers and improvements mentioned above are surely dependent on the usage of multiple monitors. There is an impact on productivity in many small details. One of them is the posture and physical setup of monitors. Some users might prefer having two monitors above each other, maybe third next to them aligned vertically. The others on the other hand can prefer having all monitors next to each other in horizontal alignment. The ability to adjust those monitors, based on the current situations can be helpful. In general, those setups are more subjective but can make a huge impact on productivity.[5] Another detail worth noting is mental attention to the multiple monitors. With the wrong usage of the second or third monitor, the user can be easily distracted and his effectiveness might not be that higher than using only one monitor. In the worst cases, effectivity might decrease. But if the user is using his setup properly, and is mentally ready for comprehending multiple workplaces, the productivity can grow really fast.[3]

As long as the risk of having the additional monitor as a distraction instead of a productivity booster, the motivation for having the correct application setup is high. The partitioning of the monitor, creating an applications layout, can be considered similar to the using the multi monitor approach for increasing effectiveness again. The difference is that now, the partitioning is on the software side. By providing partitioned applications layout with software, more options are available for enhancing the user experience even more.[7] Not only layout is crucial, but even the behavior of those applications based on partitioned layout is important. In the end, the whole behavior of the partitioned monitor layout according to the user and his context can improve his experience and even automate some of the users' work and effort.

1.2 Goals

The goals of this diploma thesis are, at first, to analyze current solutions and usage of secondary monitors. Then come up with a design and solution for an application that would provide a user the ability to customize his partitioned layout for the secondary monitor. Implement a prototype of the application with basic functionality to show the core features and principles of the idea. Analyze and test this prototype with users to get feedback about the usability of this application and future benefits. The analysis itself during the implementation and testing phase will determine the feasibility of the whole idea. One of the key features of the application is the automatization of whole solution and increasing the effectivity of work while using this application.

In the analysis part, the goal is to properly dive into the usage of secondary monitors and how users are working with them. Especially what applications are they using for secondary monitors, how is their work structured and divided between multiple monitors to better understand the user behavior. Then look for existing ideas and solutions that are even closely related, to get inspiration and possible best practices that can be used in this thesis. Lastly and most importantly, analyze the automatization process of the whole application. That should include the features, that can be automatized, the style and behavior of automatization, and as well the sources of data, which would trigger the automatization, meaning the user context data or other data connected to the user connect situation. Part of those data is eye-tracking of the user, which could provide interesting data and additional rules. This source of data should be designed and attempted to be implemented into the prototype.

The goal of the design part is to provide general solution for analyzed features. How would be the application structured, what features it should include, what data is the application using and also how the automatization works. This design should be as general as possible, but for specific key parts, the designing will be aiming towards development in a service-based web application, build onto the Electron framework, which can build a web application for desktop usage. Based on the design the prototype will be implemented. Using web technologies like ReactJS and Electron including NodeJS.

Lastly, the goal of testing is to find out whether this solution makes sense for users and if it is feasible. Additionally provide any analysis for future development and research.

To conclude, the final designed application should be fully customizable dashboard application, providing the user the ability to select his own layout for almost any application, he would like to use. For his selected applications and layout, introduce automatization of the whole application, so users would not have to interact with second monitor as much as he usually would. The result would be increased effectiveness, productivity, and overall happiness of the user.

Chapter 2
Background

2.1 Multiple monitor setup

Multiple monitor setups are common things nowadays and are widely used across all fields working in the office and also in home setups. The setup is consisting of at least two monitors, or screens in case of using a laptop. The user then can separate the work across those multiple screens he has and is able to be more productive. The main reason is to fasten the work and limit the number of switching between application and context. Mainly because it is time-consuming, the user can lose attention, e.g. if he has to switch through multiple applications. The next disadvantage can be the process of switching itself, that it can be just annoying for the user. With using multiple monitors, different applications can be on different screens, and thus users can be more focused on one task without any need to switch between applications and contexts.[9, 4]

There is another helpful tool or technique called snipping, which provide the user the ability to snip the applications to the corners of the side of the screen and organize multiple applications at the screen.[9] Multiple monitors highly increase the performance of the user together with the snipping tool, because the organization of the applications can be done on all of the screens.

For overall multiple monitor usage there are many types of research using different methods. One of those methods is eye-tracking, where researches can track how much is user looking at the other screens and for how long. Therefore it can be distinguished whether users tend to have those monitors for a passive usage, they don't use it that much, or active usage, when users are actively switching their attention between those screens and making any actions on them.

More researches are trying to prove the efficiency of using the multiple monitor setup by doing a usability test. Mostly comparing given tasks on the setup with one screen and with multiple screens. The result is then measured by time, precision, and also the quality of the finished task. Most of these researches have the same results, that efficiency and productivity increases and quality of work as well.[10, 3, 2]

2.2 Rule-based systems

The rule-based systems are referred to as systems, that use defined rules, human crafted or generated, to delegate knowledge and interpret the information in a specific way. It is often used in artificial intelligence, machine learning, or data-mining systems. The basics of ruled-based systems can be seen in almost every program, where logic programming is used. That means, a system that is changing it is behavior based on results from if and else statements can be called a ruled-based program.

There are also systems called knowledge systems or expert systems, where usually the expert systems are more specific versions of knowledge systems. Those systems are a more complex version of a rule-based system. They aim to simulate the decision capabilities of an expert when solving complex problems while using knowledge and experience gained from the expert. The goal is to make a system that has the quality of an expert in decision making.[20] This knowledge is expressed using some formal knowledge representation language and providing a knowledge base for the system.[19]

2.3 Knowledge based systems

Knowledge-based systems uses a knowledge base to solve complex problems. The knowledge base represents real-world facts. Often in the form of ontology or frames, Bayes networks, rules, conceptual graphs, or logical assertions. The knowledge base is one part of those systems. The second part is an inference engine. It applies logical rules to the knowledge base and can deduce new knowledge. In general, there are two modes of inference engine. One is forward chaining which is taking known facts and creates new ones. Second backward chaining is given goals at the beginning and determines the facts, that must be asserted to reach the goals. Knowledge-based systems are often used in artificial intelligence implementations as it is a great solution for the representation of human knowledge for machines.[26, 27]

- Predicate logic

  Easy representation and also updating the knowledge. The predicates are logical functions that return Boolean values, so true or false. They are a generalization of propositional variables, such as variable **r** can stand for *It is raining* and variable **v** for *Adam gets wet*. Then the logical function can stand for *If it is raining, Adam gets wet*, then the result is always Boolean value based on the value of the variables.[21]

- Semantic networks

  Representation using objects with connections between them, called relations. The natural representation is graphs, where objects are the vertices of the graph, and relations are edges between the vertices. The edges are labeled, so the relations are better recognizable. Those relations then holds the knowledge base. The semantic networks are compact and can be complex, but the change of the structure is difficult. The key features are comprehensibility of the structure, availability of organizational principles, such as generalization, instantiation, and aggregation and definition of the access paths between concepts and objects.[22]

- Bayesian networks

  Bayesian networks are a type of a probabilistic graphical model. They can be used for a wide range of problems like prediction, anomaly detection, diagnostic, and other analytic disciplines. In this case they describe the probabilities of the relations inside the network, so it's easier to predict the result of actions.[23]

- Frames

  A frame is a structure that consists of attributes and their values. This structure describes a real-world entity. Frames then divide knowledge into substructures by representing stereotypes situations. The frame consists of slots and slot values, being any type and size. Those slots are called facets, which are various aspects of a slot. Facets are providing the ability to put constraints on the frames, so it is possible to filter exact knowledge out of frames using those facets. In artificial intelligence a frame is also called slot-filter knowledge representation.[24] The frames are usually created from the semantic networks. The main difference is, that they contain slots with any number of facets, that can have any number of values. Then the frames are later used as classes and objects. All the frames are connected and the knowledge representation is structured, so are objects and classes. That means frames can have relations similar to those that are known from object-oriented programming, like decomposition and others.[25]

- Rules

  Rules are the most common representation of knowledge. They consist of condition and action, separated into procedural conditions, if situation then action, or declarative, if assumption then conclusion. The rules representation consists of three parts - the set of rules, working memory, and the recognize-act-cycle. This is the cycle, that is responsible for the evaluation of conditions. First, the agent checks the condition, if it exists, it select corresponding rule and uses it as an action part. The working memory has a description of the current state and can work as a trigger point for other rules. For some conditions, there can be conflicts, called conflict sets. In those cases, the agent needs to select a rule from the set. This step is called conflict resolution. The disadvantage of rules is that they can't have any learning capabilities and they are not storing the results for any future uses, also in many cases, they are not very efficient. But they are highly modular and easy to manipulate with, remove or add or modify.[20, 25]

2.4 Event-driven applications

Event-driven applications based on event-driven architecture use events for disseminating the information to all interested parts of the system. Those parts process the event, evaluate the information, and perform action. This action can invoke a service, save data, trigger another process or any other functionality hooked onto this event.[11] The events may be generated by the user or the program itself or even some other system that might be connected to the network or the application.[12]

The event-driven architecture orchestrates behavior around the production, detection, and consumption of events and reactions on events consumed and processed. It can consist of creators and event consumers as well at the same time. Creators are the sources of the events and consumers are processing events made by creators and then reacting to it. The consumers consists of event handlers and event listeners. In some cases event listeners can pass the events to the event handlers, as they can work just as a listener and middleware for the events. So listeners can be seen as a middleware for handlers, or better said consumers of those events.[13]

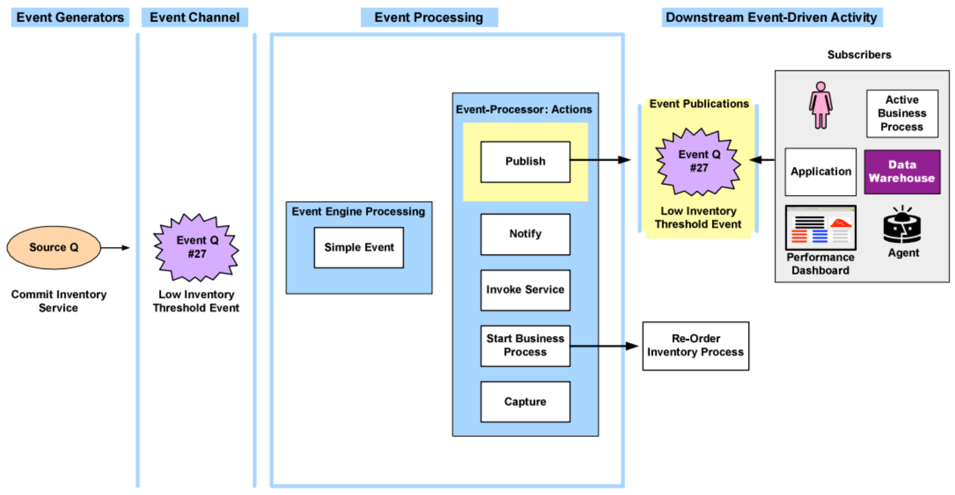


Figure 2.1: Simple event process example (taken from [11])

2.4.1 Event sourcing

By the definition mentioned in [14] (by Fowler 2005b): *"Event Sourcing ensures that all changes to application state are stored as a sequence of events. Not just can we query these events, we can also use the event log to reconstruct past states, and as a foundation to automatically adjust the state to cope with retroactive changes."*

The events in the event sourcing pattern are persisted in an event store where can be reached by the system and the store acts as a source of truth. The store is publishing those events, so consumers can be notified and handle those events.[15]

The benefits of event sourcing are:[15]

- Performance and simplicity of persisting data runtime. Meaning the ability to keep the entire state of an application in-memory, or persist it to the storage media. That would be used if the system crashes or there is a need to shut down, the data are used to restore the previous state of the system.
- The events are immutable, so using only append operation will provide consistent event store.
- The events are simple object only describing an action that occurred. They are not directly updating any data store.
- Event sourcing can help to prevent concurrent updates and causing conflicts or deadlocks in the system. That's because of not directly updating the data store. But the model has to be still designed to protect the potential inconsistency of the data.
- Testing of error scenarios or application states that may occur or may have not. The state machine can be built upon the possible events and their behavior inside the system. This can be used for reestablishing states that caused problems or construct states that have not occurred yet and test them.

This pattern is usable in many cases, especially when the consistency of data is crucial. It is widely used in cloud computing systems using microservices. Also it can be used for systems, where the information about the purpose or reason for the change is needed and stored. Next can be systems where dividing the request for a change and change of data needs to be separated, so the event handlers can work like a middleware. The next use case can be needed for an audit log for every data change made, because the event store is simply representing it from a design.[18]

2.4.2 Command Query Responsibility Segregation

In short CQRS is a pattern often used in combination with event sourcing. It is a design pattern that reads and write operations over data. Based on the object-oriented design principle, there are two categories of methods, one called the command category and query category. The command part of this pattern changes the system state but does not return any data. On the other side, query returns the system state, or data, but does not change anything.[14]

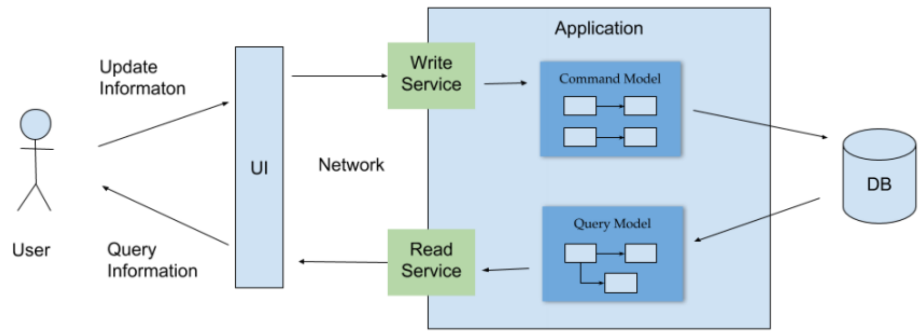


Figure 2.2: System architecture using CSQR (taken from [13])

2.4.3 Events

An event is an identifiable occurrence that happens inside or outside the system. The event can signify a problem, opportunity, threshold, or a deviation. In general events usually keep information that is valuable and significant for the system.

Events usually consists of the event header and the event body. The header contains information about the occurrence, such as the source of the event, some identification, timestamp, and it might contain the condition for occurrence as well. Event body contains the information itself, which should be passed. It is a description of what happened that this particular event was created. It can contain the additional data, which are the result of the event or just a pointer to them, so event handlers can reach those data and process them.[12]

The events should be fully described, so there is always a deterministic result or behavior of the system, after the event handler process the event. Also the event has to be immutable, relevant to the domain, and usually happens in the past.[14] To ensure the events are understandable by every consumer, ontology, lexicon or at least some event rules should be used as a specification for events.

2.4.4 Event-driven programming

Event-driven programming is used often when working with I/O operations, input, or output operations. Especially when there is a need for using concurrency. The event-driven programming is used instead of threads in those cases because it can provide more simplicity and better managing of the concurrency tasks. Even-based programs are then more stable and robust.[16] The paradigm uses event listeners that detect events and then pass them to the event handlers. In theory event-driven programming can be used in any programming language. [17]

2.5 Eye-tracking

Eye-tracking is a technique that tracks an individual's eye movement. This movement can be measured and evaluated as where is the user looking at any given time and also the sequence of his eye movement, from where to where was user shifting is attention.[28] There are two types of eye movement tracking, one that measures the position of the eye relative to the head and second that measures the orientation of the eye in the space, in other words point-of-regard. The device for eye movement tracking is usualSoly called eye tracker.[29]

The most accurate eye-trackers use high-performance cameras, that can detect near-infrared spectrum light and can benefit from active illumination in the infrared spectrum. They are highly accurate trackers, but user needs special equipment for those. It is usually a dedicated camera, that works in a light room with infrared light sensor, that can track the iris precisely. Also there is always a need for image processing software, that will process the eye-tracker data.[28]

2.5.1 Measurement methodologies

In general, there are four categories of measurement methodologies, that were or still are being used for eye-tracking tasks. Almost every one of them requires special equipment, software, proper calibration, and setup. The first eye-tracking methodologies were introduced in the 1950s and they are still being enhanced and simplified for better usage in the real world, and not just scientific researches.

Electro-OculoGraphy (EOG)

Widely used method in the 1970s. The methodology relies on the measurement of the skin's electric potential differences. These differences are measured by electrodes placed around the eyes. It requires the subject to be steady and is not very accurate for point of regard measurements, because the measurement is relative to the head position. So for POR tracking, some sort of head tracker must be used or the subject needs to be steady, but the accuracy would be dropping.

Scleral Contact Lens/Search Coil

It is one of the most precise methodologies, it requires attaching a mechanical or optical reference object to the contact lens, which is then worn directly on the eye as normal contact lenses. One of the method is using the wire coil, other can be reflecting phosphors or line diagrams. Those attachments are then registered using an electromagnetic field or other magneto-optical configurations. Although this method is the most precise, it is not very used, because of the intrusive method of inserting contact lenses into the eye.

Photo-OculoGraphy/Video-OculoGraphy (POG/VOG)

Techniques in this category usually do not provide a point of regard measurements, although they could in specific scenarios. This category includes a wide variety of eye movement recordings and measurements of the movement of the eyes under different circumstances, like pupil shape, the position of the limbus, and corneal reflections of the light.

Video-Based Combined Pupil/Corneal Reflection

For measuring the point of regard, either the head must be fixed so the position of the eye is relative to the head and point of regard coincide or more features must be measured to calculate the correct relative position of the eye to the position and movement of the head. Two of those features, to provide measurement without fixing the head, are corneal reflection and the pupil center. Corneal reflection means a reflection of the light source, often infra-red, from the eye.[31]

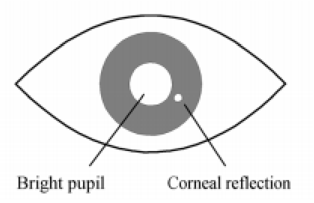


Figure 2.3: Bright pupil and corneal reflection as seen from infra red camera (taken from [28])

Corneal reflection of the light source, also called Purkinje reflection, is measured relative to the location of the pupil center. There are four reflections formed, due to the construction of the eye. The video-based trackers are usually using the first one. After calibration procedures, eye trackers are able to measure the viewer's point of regard on a surface where are the calibration points. So in case of a computer, they would be displayed on a monitor, and calibration would be done against those. So the point of regard would be on the monitor display. The point is found by trigonometric computations, where the distance between bright pupil and corneal reflection is measured and then used to calculate the point of regard.

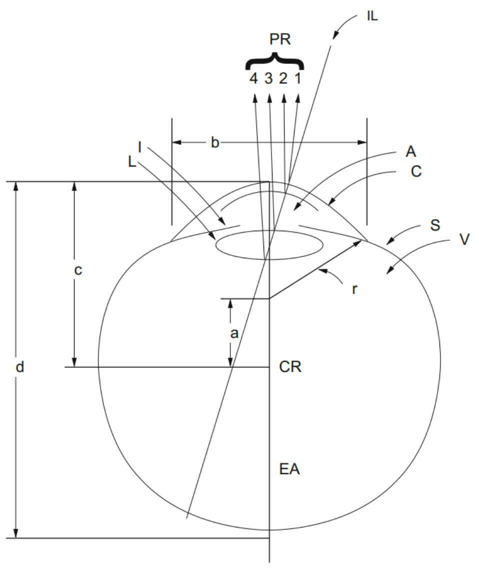


Figure 2.4: The reflections of the eye. (taken from [29])

2.5.2 Usage of eye-tracking

As technology evolves and high-resolution cameras are more accessible, the eye-tracking research and usage in practice is increasing.[32] Also the possibility of real-time eye tracking with current machines is very common as it can be run on normal computers with average hardware resources and also an almost average camera. Even software that computes the tracking of eyes does not need the infrared cameras, as the resolution of the web cameras is getting better and better.

Regarding using the eye-tracking technology, there are some other factors, that can be tracked. Human vision has two parts of the area that can be seen. First, with high resolution, is foveal vision. It has about 2 degrees of the visual field where humans can see sharply. The other part is peripheral vision, where humans can see blurry objects, unable to read or distinguish if he previously did not see them. Based on this fact, there can be tracked two things. Fixations and saccades. Fixation is, when the eye is "resting" on something specific. It takes up to a half of a second, but for that time, the eye is fixed onto one point. The saccade last only about up to one-tenth of a second a during that phase, human is effectively blind. Eye-tracking is more focused on fixation phases as at that phase, it can be distinguished where is the user looking and for how long.[30]

Human computer interaction

In human-computer interaction is eye-tracking widely used for purposes of testing. The users are tracked while they are using the tested applications. While users are being tested and using the application, the tracking software is gathering data and often creating a heat map. The result is the overall heat map of the user's eye movement across the application. That can be then evaluated by the authors if the user was behaving as intended.

Some new apps are trying to experiment with controlling the interface, or the whole app using eye-tracking techniques. So users might scroll with their eyes, or stop the video, when they are not looking.[30]

2.5.3 Webcam eye-tracking

The webcam usually does not have infrared light to scan the reflection from the pupil and the corneal reflection. Webcams have only a visual spectrum, with limitations of resolution as many webcams do not have that high resolution, because for common use it is not needed. The next drawback could be the latency or the frame rate, which can be resolved by lowering the resolution, but then the image of the eye can be one blurry picture. Also webcams are highly dependent on the light, so good light conditions are important. So the process of eye-tracking with webcam is often in the following steps:

- Detect location of the face.
- Detect location of eyes within the face.
- Detect orientation of left and right eye.
- Map the orientation of the eyes onto the screen coordinate system.

The kay part for mentioned steps to work is the calibration, which is done the same way as with standard eye-trackers but here it could be crucial for the result.[31]

2.6 HCI - Human computer interaction

Human-computer interaction, or in short HCI, is a field of study focusing on the interaction of humans with computers. HCI studies the design of the applications and hardware elements of computers, to be as user friendly as possible. The main reason why are these researches increasing is the fact, that more and more people are using computers and are in need to use them on every day basis. Therefore it is important to make usage of computers and related technologies as accessible as possible.[35, 36]

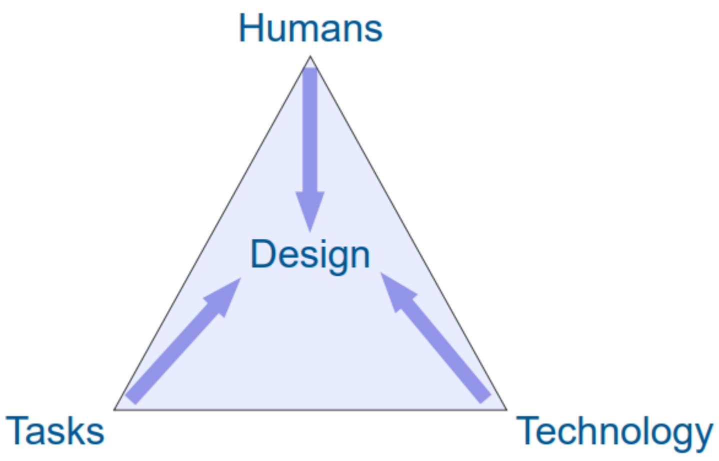


Figure 2.5: HCI elements. (taken from [33])

2.6.1 User interface - UI

The user interface provides the user the ability to communicate with the computer. It can be perceived as a bridge that connects the computer with the user. The study behind this bridge is backed by the HCI field. As more people are using computers, the more crucial is the development of UI for any programs that people work with.

The main focus of UI development is to provide user-accessible, easy to use, and effective interface. The interface must be also intuitive, so the users do not have to read manuals and undergo any lectures that would teach them how to use those programs. Another important parts of UI are esthetics, emotions that users have during using the program, and also the experience that would user get. Those are also key factors when developing a good user interface.[34]

In general user interface does not have to mean the software program or an application graphics. By UI we mean also mechanical things, buttons, and so on, which can be interacted with by the user. In software development, the UI specification is GUI, a graphical user interface. The world, graphical is often missed, when the context of a software application development is clear. So sometimes it can be the same thing. In this thesis, the short UI will be used, as the context is strict to the software application.

The graphical user interface, next just user interface, can be divided into two parts. The static user interface and the dynamic, or adaptive, user interface.[33]

Static UI

Static user interfaces are almost immutable when it comes to the layout of the elements in the application. The only part, that is changing is the data and content. The main advantage of the static UI is that it can be easily remembered. Thus the user can use the application more by memory then recognition of the elements, which is increasing his effectiveness and fastening the task he is doing. Also it is much more easier and comfortable the more he uses the application. For those advantages to work, the UI must be well designed, that there are no annoying elements or distractions. Because if there were any, users would be more unhappy with using the application for a long time.

Dynamic UI

Dynamic user interface is on the other hand not immutable. It can change its elements as well as layout and data with content. The changes might be affected by context, user preferences, or some effects from outside the application.

The very similar design is the adaptive user interface. That is more focused on adapting the user interface through the time and trying to achieve the most suitable interface for the user. It can adapt to one user specifically, be tracking user context, user data, and also his preferences, or in general to all users based on global user usage data collected over time.

The advantage of those systems is the behavior that can be achieved through the adaptation, it could be exactly the one that user needs and it could be perfectly tailored to his need or needs of a group of users. The example can be adaptive menus, that can change based on usage of the tools from the menu, as well as the layout of the menu can be changed based on user-specific usage.

The disadvantage of adaptive and dynamic user interfaces is the cost and effort that needs to be put into the development. With wrongly designed adaptive UI, the user can be quickly annoyed and stop using the application, because he does not like the UI after the adaptations.

2.7 User context

The user context can be understood in many ways. In software development the user context consists of relevant information about the user, better his environment and situation. There are many aspects when creating a user context and what is in the user context and what should be in a global context of the application or the overall situation.[41]

The information inside the user context can more general, like the estimated location, could be as a place of interest or the type of situation the user is in, like working, traveling, or other. Then there is more concrete information, like temperature, used applications, noise around the user, or light conditions. There can be many more and it depends mostly on the device the software is installed and permissions the user gives to the software.[42]

Persisting and processing user context can be challenging and the challenge can the representation of the user context. The store for the context should be in specific format, that can be easily processed, manipulate, and can also have some relations or other features useful for work with the user context. That information can be really complex, so it is necessary to have a robust format, which is also readable by humans and machines as well. There have been introduced some basic requirements for such a format:[41]

- Structured

  Information should be structured, so it is effective with search but also with delete, add, or update operations.

- Interchangeable

  The context should be able to move to other applications and should not be dependent on a specific application.

- Composable/Decomposable

  Distribution of the data across multiple storages should be possible.

- Uniform

  For all use cases and applications, the context format should be all the same.

- Extensible

  The ability to add more information is a must for format storing user context information.

- Standardized

  Format should be standardized for the communication between different sources, so both of these sources know the standard.

Most of the existing formats are based on the Resource Description Framework (RDF), which is a standardized framework using XML for storing and passing information over the web and HTTP.

2.7.1 User context tracking

Tracking of the user context is widely used in mobile phone applications because of the presence of multiple sensors, which can access multiple user information. In mobile phone software such tracker applications are called context-awareness applications, because they are connected to the user context almost directly.[38]

Another tracking of user context can be seen everywhere on the internet, where Google and other companies are tracking our searches and then provide more relevant and personalized advertisements, products, or results from our search. This tracking is then used by companies, to see if their ads are working well and people are buying their products.[37, 39]

By user tracking it is also possible to see how users are evolving. Those data from the context can be stored and evaluated over some periods of time. Applications can then adapt based on those evaluations or authors can change their approach and make some changes. The software can also learn from those evaluations and change the behavior for some specific tasks.[40]

The more interesting user context tracking for this thesis is tracking of the local context. Or even better real-time context of the user. The information about what is user doing at the moment, what application is he using, where is he sitting, and also where is he looking can be used for automatization of processes in the applications or adapting the user interface. Also if some applications need tracking to be live, because their behavior can be based on in.[38]

Chapter 3
Related work

There are many related papers and researches when it comes to multiple monitors. Mostly from years after 2000, when multiple monitors started to rise even for personal usage. Those researches are focusing only on the usage of secondary monitors in terms of efficiency.[5] Trying to prove, that it is better to use them and that the productivity of the user increases. As mentioned in chapter Motivation, they were successful in proving that.

Nowadays, there is a huge support of multiple monitor setup from operating systems itself. But they lack the user-friendly layout organization when it comes to working with secondary monitor. Therefore snipping tools were introduced, which helps organize applications on the screen by snipping them to the corners and dividing the screen into the sections. For macOS there need to be third party applications installed, but Windows and Linux are supporting this feature natively.

The examples of those applications can be KDE Window manager for Linux, macOS Spaces, Tiles, Magnet, and Spectacle. All of those are providing almost the same functionality, dividing the screen layout, so the user can have a more organized applications layout on all monitors. Unfortunately there is no integration of the applications or even persisting the layout setup in most of these. And also the customization of the layout if often quite limited.

With focusing on application integration, there are more possibilities. The one category are applications which are integrating many web apps and providing user to have all those tools at one place. They are using web technologies to provide the whole integrated application, additionally implement notifications and small additional features. The user only selects which services he would like to use from the list of available services. The difference in those is that they can show and use only one service at the time. So user, have to interact with the application every time he wants to switch to different service. Despite this limitation, these applications provide an enhancement in the organization of the services user likes to use. The examples for these applications could be Station[45], Franz[46], and Ferdi[47]. And there are many other similar applications.

The drawback of the above-mentioned applications could be their performance. As they behave like web browsers and they are not always optimized that well. Overall I found no application with a similar complex purpose as is the aim of this thesis. Existing applications are facing only the problems mentioned above.

Chapter 4

Analysis

This chapter of the thesis includes the overall analysis of the problem, the existing and related solutions and their approach, and techniques mentioned in the background chapter. In addition, part of the analysis is a personal research which was made for this thesis using a questionnaire and specific group of users. The goal is to provide a better understanding of the problem and better starting point for the design part. The possible technologies and principles, that can be used can be mentioned as well.

4.1 Multi monitor usage

Following the motivation part of this thesis, multi monitor usage is becoming almost a standard for most of the office workers and also users at home. The results of multiple researches claims to prove the increased efficiency when using multiple monitor setup. The next step of analyzing the usage of multiple monitors is aiming at the efficiency of multiple monitor control and app usage itself as well as the organization of the tasks across the monitors. It is essential for users to use multiple monitors in a way, that they are not being distracted from work, which would affect the overall efficiency in a negative way. So the key part for achieving the most efficient work space with multiple monitors is the appropriate configuration and application layout. For every user this may be different, and it is up to every individual what setup is least distracting and most supportive for their work.

4.2 User research - multiple monitor usage

The user research to support the analysis of the multiple monitor usage was done using a questionnaire, which would lead to a better understanding of user behavior and the most common usage in a particular targeted groups.

The participants were all from IT companies and mostly programmers and people working in the same field. The total amount of participants was 46. The results of this research should show how are users currently using secondary monitors or in general, multiple monitor setups and whether they see any disadvantages or a space for the improvements. This research has not included any information about the current thesis and its idea, so participants are not influenced and it is possible to analyze the real current situations.

The questions were following:

1. How many monitors are you using in total? Response was a number and laptop screen counted as a monitor.
2. What screen setup are you currently using? If you are switching between different setups, please feel free to write down those and reasons why.
   - Next each other
   - Above each other
   - Other
3. If you are using laptops, are you using laptop monitor as your main monitor?
   - Yes
   - No
   - I a not using laptop
4. How often do you switch between main and secondary screens?
   - Often - every couple minutes
   - Not so often - every 15 minutes
   - Sometimes - every 30 minuts
   - Rarely - every hour
   - Almost never - every couple hours
5. Please, shortly describe purpose of your secondary screen. Open question.
6. What apps are you using on your secondary screen? Open question.
7. How much is your secondary screen organised? 1 - it is a mess, 10 - perfectionist dreams Response was a number from 1 to 10.
8. What OS are you running?
   - Windows
   - MacOS
   - Linux
9. Do you use any apps for managing apps/windows on your screens? If so, which? Open question.
10. What are the biggest disadvantages of a secondary screen for you? If you see any, what would you change in using secondary screens? Open question.

Almost half of these questions are open. The reason is to collect various thoughts from users about using multiple monitors. Also it will help future analysis about existing ideas or show missed questions that should be tackled in the analysis process.

4.2.1 Results

1. How many monitors are you using in total?

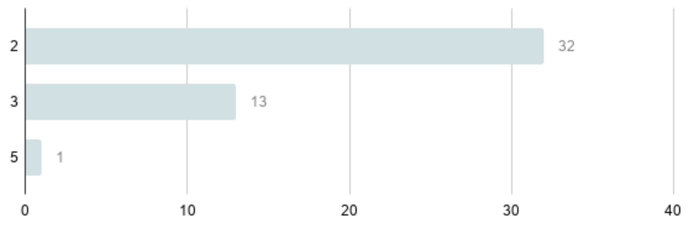


Figure 4.1: Number of monitors results. Vertical axis is number of monitors, horizontal number of answers.

The target groups were people with multiple monitors, so every respondent has two and more monitors. Where the monitor counts as a laptop screen, so there is no need to differentiate between desktop and laptop users. The majority uses two monitors, a total of 32 respondents. Thirteen of them use 3 monitors and one of them uses 5 monitors.

2. What screen setup are you currently using? If you are switching between different setups, please feel free to write down those and reasons why.

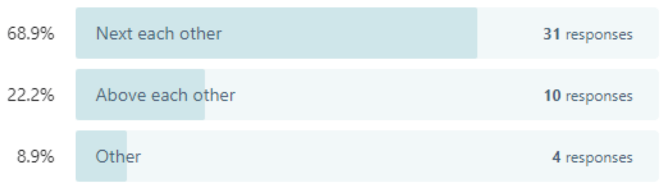


Figure 4.2: Question 2 responses.

The majority of respondents have monitors next to each other. Four respondents who selected the "Other" option also have some variety of next to each other setup, mostly two next to each other and one under them.

3. If you are using laptops, are you using laptop monitor as your main monitor?

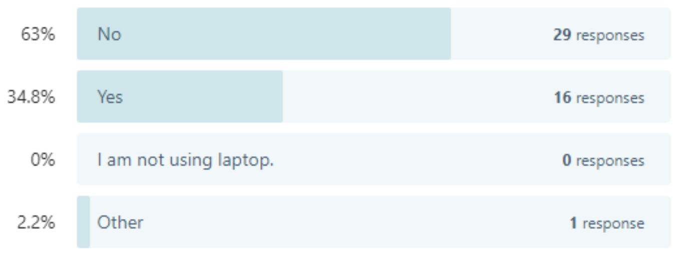


Figure 4.3: Question 3 responses.

Most users tend to have their laptop as a secondary monitor and work with the external one as the main one. That can be the case for bots main setups, above each other, and also next to each other. But for example, it's easier to track if the user is paying attention to the second monitor or not if we would use laptops web camera.

4. How often do you switch between main and secondary screens?

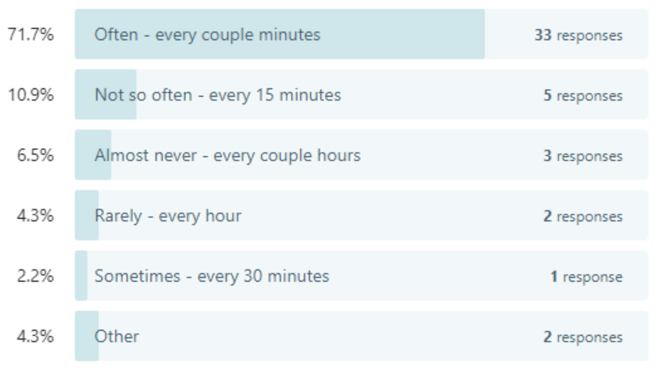


Figure 4.4: Question 4 responses.

More than two-thirds of users tend to use the secondary monitor very actively. Only a small percentage of them are using a secondary monitor passively their work. That means either their application layout is not ideal and some of those switches are unnecessary or they might have some terminal or other features that require some sort of interaction, set on the secondary monitor.

5. Please, shortly describe purpose of your secondary screen.

All respondents almost agreed and were in consensus that the secondary screen is a place for not that much-used application. Some of them are using it for a web browser or actual application they are working on. Many of the users, almost 3/4 of them are having chats, emails, and some other information-gathering applications on the secondary monitor. Also it can be a place for applications that can be potentially disturbing.

Users with 3 monitors usually have their third one as more passive ones and using it for music applications, emails, chats, and less used applications. But all of them agreed that it is mainly for increasing their effectiveness during work.

6. What apps are you using on your secondary screen?

Many users responded with browsers, chats, and email applications. The other are very often terminals or IDEs. Followed by applications like dev tools and adobe products (illustrator, photoshop, etc.).

7. How much is your secondary screen organized? 1 - it's a mess, 10 - perfectionist dreams

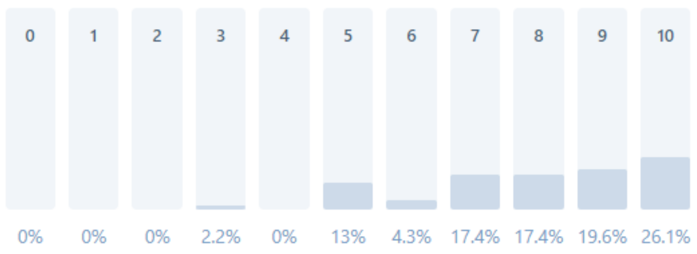


Figure 4.5: Question 7 responses.

Respondents secondary monitors are usually organized. The main reason for that is probably the fact, that they are professionals and use their setups all the time. So being organized is also the key to effectiveness and productivity. But not all of them are having perfectly organized secondary monitor layouts.

8. What OS are you running?

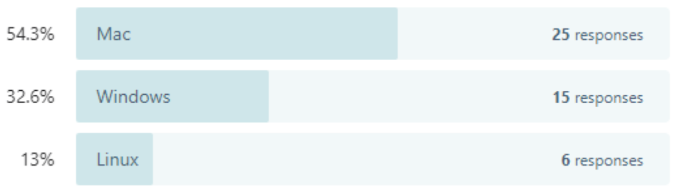


Figure 4.6: Question 8 responses.

The MacOS is a bit more popular in fields like IT, but regarding multiple monitor setup it has weaker support than Windows or Linux, as it does not have integrated snipping tools like Windows and Linux does. That is the reason why many Mac users have third party applications for snipping and setting up the layout on their monitors.

9. Do you use any apps for managing apps/windows on your screens? If so, which?

Half of the respondents are using no application for secondary monitor management. This means they have to manually organize their applications or use native snipping tools integrated for example in windows. Other users, mainly mac OS users are using tools like Magnet, Mac OS Spaces, Spectacle, KDE default, and Tiles. Those applications are only for setting up the layout of your applications by snipping them to the corners or using short cuts. None of them is adding any more features, and users are not able to save those layouts persistently and change them according to context for example.

10. What are the biggest disadvantages of a secondary screen for you? If you see any, what would you change in using secondary screens?

About 20 % of respondents see the disadvantages in the physical setup of their monitors and fact, that for long term use of the secondary monitor, user can have bad posture and hurt his neck or back. Few of those respondents also mentioned portability. Sometimes they need to move to meetings or just sit on a couch or even work from home and they are not able to move their setup with them. Also difficult cable management and monitor positions were mentioned in responses.

This leads to another issue that was mentioned. With bad portability comes issues with workflow and setting up the layout of the applications. Respondents are having trouble setting up the layout properly every time they disconnect from monitors or even come home, where they might have a different setup and also different context of work.

The last issue that was mentioned a few times, in like 10 % of responses, was bad optimization from operating systems, resource demand, and also a hard organization of the applications. Also one respondent mentioned that he or she sees disadvantages in applications that require a manual click or some interaction for just showing some news and that the notifications are not enough for getting the whole information. That makes the need for an extra unnecessary interaction with the secondary monitor.

4.2.2 Evaluation of the research

Based on the result of the questionnaire we can surely say, that focusing the research towards multiple monitor setup does make sense, as the majority of people are using it. And many will join them as technology grows and those setups are being more and more available. Supporting fact is, that even programmers and IT workers, who should be the ideal users of complicated setups, are having some issues with using secondary monitors and still find some missing features and space for improvements. As it was mentioned before, the goal of this thesis is to provide a solution for better productivity and usage of secondary monitors, also their automatization. Simply, make just users life much more easier while using multiple monitor setup. Regarding the results, users might appreciate the mentioned features that is being designed in this thesis. Such as the layout selection and set up, which can be persistent and the user might choose one regarding the context, or even better, it might change itself based on context change. Or automatization, so the user is not forced to switch between monitors so much with his controls and ease his movement over the workplace.

4.3 Organization and user data

The result from the questionnaire and also results from the researches mentioned above are clear. Organization is the next key to be more efficient and productive while using the multi monitor setup. This supports the idea, that providing customizable layout for the user might help him to achieve a better workplace, where he would feel better.

Having a good layout, that suits users, seems to be increasing efficiency by a lot. Native operation systems are also supporting this thought, as they are starting to natively provide the possibility to set the layout of your applications on the screen.[48] The Windows have it natively as a snipping tool, where the user drags the application to the side or corners and application snips to that part of the screen. Similar applications are also on macOS, but the user has to download them from third-party providers as those tools are not integrated. On Linux systems, this functionality depends on the distribution, some of them have it natively or users are able to download additional extensions that would give them the functionality.

The snipping tools have one missing feature and that is the persistence of the layout. At least for most of the tools mentioned by users and found, there is no option to save the current layout of the applications. Every time user moves with the layout, turn off the computer or very often just unplug the monitor and then plug it again, the layout is lost. Then the user has to set up the layout again manually with applications he wants. Although it does not usually take long and it is mostly intuitive, it can give some frustration to the user, especially when the user is changing his environment a lot, so he has to do it many times per day. He might even abandon the organization and snipping and get used to the application anarchy or just partial organization.

4.3.1 Organization applications for messaging

As an organization application we can consider the applications mentioned in the chapter Related work, like Station, Franz, or a fork of Franz, Ferdi. Those applications work on a WebView principle, they provide the interface for the application itself and then the view of the application user wants to show. The behavior is similar to the usual internet browser but the difference is in the setup and persistence. Those applications provide persistent setup and can be even moved to another system and still keeps the user setup. The main advantage is that the user can have all his most used web applications in one place, so it helps him to be more organized. The potential drawback of those applications is the usage of RAM. The apps are not displayed all at the time, but can have some services running for notifications for example. With those, the resource need might be increasing.

An interesting feature of apps like Ferdi or Station is the option to add custom service. Applications are written in JavaScript and providing an option to write a custom service using a predefined template. It's a complete standalone module with its own 'package.json' file that describes the configuration of the module. Then there are two files, each exporting one function that accepts their provided module, having the API for customizing the application. There are possibilities like enhancing notification, work with external API for fetching, or having a custom URL. They are not providing the rendering customization, so at the end, the application always renders provided URL by the user, but with customized behavior.

4.3.2 User data

User data, including user context are the essential part for this thesis as they are the base of the following rule-based approach and automatization. In this case user data consists of user context and data the user provides to the application as config files or some other inputs. Together they create a user context. As mentioned in the Background chapter, there are two parts, storing the context and tracking or retrieving the user context data. There are many ways of storing and retrieving those data, but it has to be kept in mind, that this application will probably run on a desktop computer or laptop, instead of a web application or so.

As those data are the core of the rules and automatization, the quality and usage of them is crucial. The advantage, in this case, might be the environment in which the application would be running. Operating systems can be, with some of the user approvals, very opened for sharing user context information. In the best-case scenario it could provide and track the whole user behavior inside the system. That includes the application usage, time in applications, some basic settings, and even others. Then there is the access to the camera, microphone and other peripheries connected to the system. In overall, the user context can be full of information and really complex on personal computers.

With the data gathered and stored there is also an ability to retrieve user behavior as a part of the user context. That can be done by analyzing some of the retrieved data. From the analysis, some behavioral models can be made and stored together with user context as a user context data. The reason for this could be seen when it comes to rules and automatization, which can be based on user behavior. Mostly because of the fact that it is just more informational as it is more complex.

The tracking of the user itself can also provide useful information. Regarding the eye-tracking, the application is able to detect whether the user is paying attention or not, or if he is even currently being present at the computer. With advanced eye-tracking there might be the possibility to even detect where exactly on the screen is the user looking. With multiple monitor setup and information about all running applications, we can detect on which monitor and which application is the user looking at the moment. That would provide valuable information for future automatization and ruled based behavior. On the other hand, the computation of the eye-tracking can be very resource-consuming and also the precision of the tracking using for example a web camera is not very good, which can lead to false-positive results.

4.4 Rules and automatization

Before designing the rule-based part and follow up automatization, it is necessary to determine the scope. The reason is, that without a scope, we can start from some if/else statements to a fully knowledge-based system, that uses machine learning and artificial intelligence for improving the rules and making better automatization decisions. Therefore first it's necessary to narrow down the scope in order to design the application properly.

To better understand the complexity of the rules, here are be some basic examples:

- When some music player is present in the application, the app will mute the player when another sound is played in the system. Other sounds should be defined, like YouTube video or a movie.
- Eye-tracking can detect whether the user is focusing on the work or not and what app is he currently using. So when the user is in focus mode, the app would mute notifications and do not make any too visually noisy changes. After the user switch focus to the application, those notifications would appear and the app would unmute itself.
- The application can change contexts based on what is user doing or where is the user currently staying. For example, if he is in the office, the office context would be chosen. If he is at home, context switch, so for example social media are present.
- Specific rules based on used application behavior. Like special notifications when combined with user data.
- Automatic scroll-based on the attention of the user.

Many more rules can be applied. Those use cases can be simple, but also quite complex in terms of combining the data. But the decision model, at least for this thesis, will stay simple and the results would be exact values or true/false conditions. For this purpose, the knowledge-based system is unnecessarily complex. On the other hand, it cannot be solved using only hardcoded if statements, so other approaches will have to be used. The other issue is the diversity of the rules. There can be some basic rules, that would follow up the same pattern but most of them are different from their inner structure and behavior. But overall the rules can be applied like a pipeline or a middleware between the layout module and user context. As it can just basically manipulate the incoming data from user context and then sending the result of the applied rule to the application.

The automatization is connected to the rules. The rules more or less define automatization processes and which one should be triggered. Then automatization is done in the global application or the applications used in the layout. Thus the whole layout or the applications inside it would need to hold the automatization processes and those would be triggered once some rule is applied that would trigger those automatizations. Simply said, the automatization is a result of the applied rule.

In overall, rules and automatization can be seen as two parts, front-end and back-end, where rules, the back-end part, would provide an interface, that would trigger the automatization on the front-end part.

4.5 Analysis summarization

The analysis of different parts of the systems shows the key features, that have to be taken into consideration during the design of the application. For the system to be innovative and providing users the most efficiency increase possibility, only the best parts of each analysis need to be used. It can be considered that the application will consist of four main parts or modules. The first one is the layout module, second is the application module, third is the rules module, and fourth is the user module. Each of these is an essential part of the system.

First part, the layout module will consist of the layout management. It will let the user choose the desired layout, he will have the possibility to create his own and store it for later use. In connection to the layout, there is an application module, app module in short. This module will provide applications that can be used inside the layout. Possibly, it can be any integrated application, or a link to the website which would be shown. Those two modules will need to be connected because the applications will be directly integrated into the layout. The purpose of the user module is to track, retrieve, and store user data. It can be connected to any outer API or service, to get and store more data. Those data are then passed to the application as a part of the user context. As mentioned above, between the user module and applications, there will be the rules module. Once the user context changed, the pipeline would be triggered. Data will be passed from the user module to the rules module, where correct rules will be found and applied. Then the rules module will notify applications with new data and requests for automatization, which will be part of the layout module or even the app module and applications.

So the key features of the system will be be:

- Layout customisation, persistence and adaptability.
- Various applications for use, possibility to provide a link to the web page, or choose an integrated application. Even create a custom application.
- Define rules and behavior as an automatization result from the applied rules. Automatization of the behavior of layout and applications.
- Select tracking data, that would be stored inside user context to either enhance automatization, as rules can get more complex or ease tracking of the user and choose less automatization.

Chapter 5
Design

In this chapter I will go through the design of the application architecture and individual modules architecture. Designing is based on the results from the Analysis chapter. Also the whole design is supposed to be in a general view, so it can be used with any implementation and any programming language suitable to reflect the desired architecture.

For the overall architecture of the application, the scalability and modularity is very important. Every module in the application should be easily replaceable for a better one, or with just a different implementation. Also the scalability is important, because the application might grow as more features, trackers, rules, and applications are being added to it. If those two are considered during the design, it will open up a space for future solutions, enhancements, and also potential research regarding this topic.

Following the main concept of customizability and automatization, which are the core features of this project, the final application should work with generic rules and interfaces. As well as the ability to add own apps or rules by the user. With that, the user would have a free hand to choose his own way of using the application. Combined with the properties mentioned above, the application will be composed of five modules. Those are the layout module, dashboard (the rendering module), application module (app module in short), rules module, and user module. Those modules were mentioned also in the analysis where I described possibilities. The final functionality will be described here as well as the connections between them and their architecture.

The figure 5.1 shows the basic model of the application structure. Green modules can be considered as a back-end part, the dashboard is a fronted part which is communicating with the user, and layout manager and store are the front-end module, that takes care of the layout functionality.

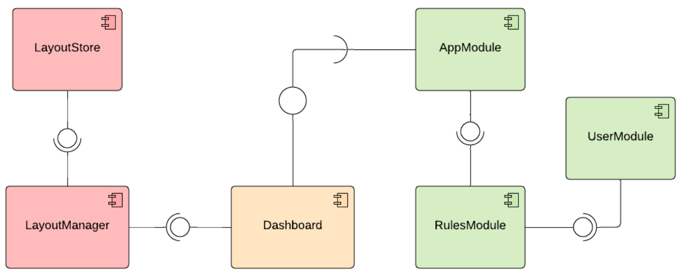


Figure 5.1: Base structure of the applicaiton.

5.1 Dashboard

The base of the application is the Dashboard. It's basically just a rendering process, which is accessing to the layout manager, where it can create or get currently used layout and then render the layout to the user interface. The other connection is by providing the interface for the app module, which is implementing this interface for each application and that is then accessible from the dashboard and able to be rendered in a given layout.

5.2 Layout module

The layout module can be considered as manager and store. The manager is handling the custom layout creation, connections, and work with the storage of layouts. Store is shown as a different component of the module as it can be fetched from different sources and could have some sharing logic in the future. Meaning sharing custom layouts with other users and other functionalities. The key feature of the layout module is to provide full customization of the whole dashboard for the end-user. Basically, the user can set up any layout he wants or choose some predefined layouts.

The layout itself consists of so-called nodes. The whole layout from a rendering point of view can be seen as a tree structure. It is assembled from row and columns, making it a similar structure to the grid system in CSS methodologies or a table representation from tables as we know them. The root node of the tree structure then can be a single simple row as a start. This row then contains at least one column. Therefore each column can contain either nothing, making it the leaf node holding an application, or another row, so it can go deeper into the layout tree.

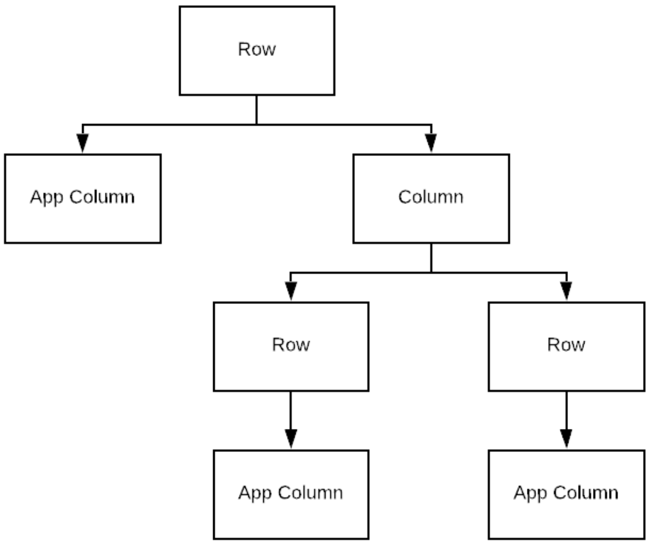


Figure 5.2: Example layout tree structure.

The tree diagram showed in figure 5.2 is representing a single three app layout structure with one app on the left side of the dashboard and then two apps on the right side above each other as shown in figure 5.3. It could be possibly simplified and the two rows having just app column leaves could be omitted. But as mentioned above, those rows helps to keep consistency and provide more flexibility in any case of changes that need to be made to the layout itself.

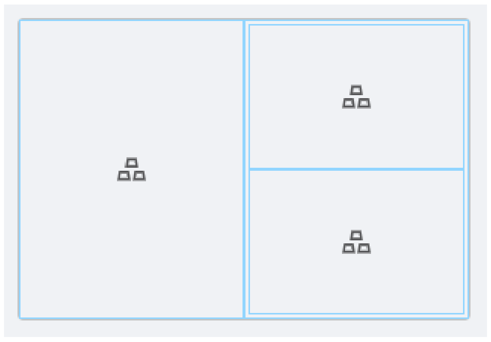


Figure 5.3: Example layout.

The leaves of the tree then contain the final app provided by the app module. Dashboard will simply mount the right apps to the leaf columns while rendering the layout.

5.2.1 Layout representation

The structure of the layout definition needs to be easy to read and render as well for writing and changing the layout. The representation of the layout will be stored in JSON format because it is easy to write and read and also representing actual hash map or normal object, which can be used in any other technology.

One way, which is more focused on rendering, is to copy the structure as rendered HTML code as mentioned above. That will result in the tree-based structure with nested nodes. This structure is good for reading and rendering as it copies the HTML structure of the code, meaning firstly it has `rows` and they contain `columns` such as a `grid` system. That is the best solution for rendering these layouts.

Example of nested structure:

Listing 5.1: Layout tree representation in JSON

```
"layout_one": {
    "rows": [
      {
        "key": "row_1",
        "columns": [
          {
            "key": "1_1",
            "span": 12
          },
          {
            "key": "1_2",
            "span": 12
          }
        ]
      },
      {
        "key": "row_2",
        "columns": [
          {
            "key": "2_1",
            "span": 12
          },
          {
            "key": "2_2",
            "span": 12
          }
        ]
      }
    ]
  }
```

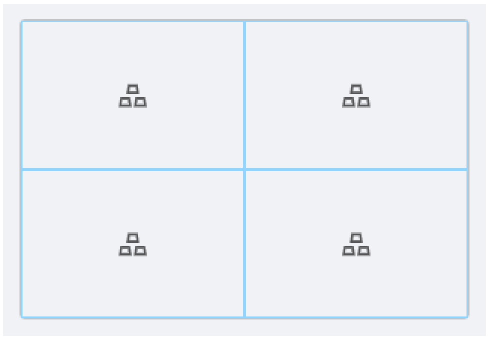


Figure 5.4: Result of the JSON example from listing 5.1

This structure is readable and it is easy to imagine what the layout looks like, so it is also easy to write manually. But once it comes to dynamically creating the layout, it is hard to manipulate with nodes, and adding or removing a single node means a big overhead. Firstly, the correct row needs to be found and extended by a new column, or if adding a new row, a correct column needs to be found and then row can be added. This is unnecessarily complex and it can be simplified.

A better solution is using a hash map to store all the nodes in one object. Each `node` would have its own unique key making it accessible without going through the whole layout definition. The structure will be defined using properties of objects stored with those node keys like in listing 5.2.

Listing 5.2: Layout node representation

```
"node_1":
    {
        "row": true,
        "level": 0,
        "children": ["node_2"],
        "parent": [],
    },
```

Each node will have at least 4 main properties. First is an indication if the node is a row or a column. Represented by `row` Boolean attribute. Second is the `level` for easier manipulation. Third is an array `children` of nested nodes. And fourth is the `parent` attribute so it is easier to move through the nested nodes both ways. Top to down and down to top. Fifth, not shown attribute, is `span` attribute which is representing a width of the column, meaning it is required only for column nodes.

For each node that has `"level": 0` it is considered that it is a child of a root node. Better said, whole layout container, which renders the whole layout. So basically the parent of those level zero nodes is a root `row` node, but it is not specified inside layout definition.

Listing 5.3: Layout representation using nodes

```
"layout_one_hash_map": {
    "node_1": {
        "row": true,
        "level": 0,
        "children": ["node_3", "node_4"],
        "parent": [],
    },
    "node_2": {
        "row": true,
        "level": 0,
        "children": ["node_5", "node_6"],
        "parent": [],
    },
    "node_3": {
        "row": false,
        "level": 1,
        "children": [],
        "parent": "node_1",
        "span": 12,
    },
    "node_4": {
        "row": false,
        "level": 1,
        "children": [],
        "parent": "node_1",
        "span": 12,
    },
    "node_5": {
        "row": false,
        "level": 1,
        "children": [],
        "parent": "node_2",
        "span": 12,
    },
    "node_6": {
        "row": false,
        "level": 1,
        "children": [],
        "parent": "node_2",
        "span": 12,
    },
},
```

Code from listing 5.3 results in the same render as previously mentioned code. There are six nodes, two rows, each with two columns. Some empty values can be omitted to save space and make those definitions more readable if necessary. As it is shown, for column nodes there is `span` attribute giving those nodes width. Each row consists of 24 blocks. Therefore the total span value of columns inside a row node have to be 24. It counts for each row, so nested row can also have 24 blocks even if it is contained in the column with `"span":12`.

5.3 App module

The app module contains all integrated applications. They are implementing the interface, so the dashboard module can consume those and integrate them into the layout. Customization is also the primary goal for this module as well as for others. That means, there should be a generic solution for integrating applications into the dashboard, so any suitable application can be used. There is the interface from the dashboard rendering side, which can be implemented and used with almost every other app. So looking for possible solutions on how to make the integration of other applications more easier and user-friendly results in those ideas.

The first option for application integration is to use public APIs of 3rd party applications. In some cases those APIs are publicly accessible, in other cases, you need to authorize using some token or private keys. For the APIs with the private key or token, the implementation of the application should contain those tokens or private keys and use them securely for accessing the API. Some of API providers also provide the component or some widgets for the connection to their applications. Therefore it is easier to implement those, as the only necessary thing is to connect those components or widgets to the dashboard interface, so it could be rendered. Simply said, the dashboard would just render the API call result into the selected layout and that is all. All other interactions with the application would go through the same API. The example might be some server-side rendered application or just a provided component with basic data and features.

The second option is the most simple one. Some services do not have to have public APIs and are accessible only on their web pages. So the solution for this type of service is using the generic implementation of the interface. This implementation takes only the URL of services user wants and render the whole service as a browser would. This might be the most common case as it is easiest to use. The drawback of this is, that there is no implementation of application or custom rules for those services as the dashboard application won't have access to those because of the CORS policies and prevention from CSRF or XSS attacks.[49] Most likely all those integrated applications using their URL will be just about rendering whole web applications, or other on provided URL, into the correct layout using the generic implementation of the dashboard interface. This solution would require using the embedded browser or in the most simple way, usage of iframe if we use some web technology as an implementation for the dashboard app. Most of the new applications are blocking iframes, so there is a need to use some kind of embedded browser, which can be provided for example by the Electron framework.

The next option is using custom components. The only rule for those is to implement the dashboard interface, but otherwise there are almost no restrictions. So if the user would like to have its own application integrated into the dashboard, the easiest solution for him is to provide an implementation of the interface and it's up to him, if he provides any custom rules and rule behavior for his implementation.

The first and last solutions seem to be similar or almost the same. The main difference is, that in the first option, we are talking about integration of third party applications that have the rules for the usage and also can have their components or widgets prepared. For the last option, a good example would be implementing a home service connected to the users API for watching movies he is storing inside NAS he has in his own network. Or another example can be implementing a custom player, that would connect to various services like Spotify or Apple Music and play music from them.

5.3.1 App interface

So in general all the integrated applications must implement the interface. Then the dashboard will just get those implementations and use the interface to render them. The interface will is quite simple, as in general the only method that is needed is render method for the component returned by the application. Therefore the interface will store just a component and an URL if there is no component provided, so the web page is rendered.

Listing 5.4: App interface example

```
interface AppInterface () {
    string URL;
    Component component = GenericComponent(URL);
}
```

If no component is set, the generic component is provided with a defined URL. So at least one of those values is mandatory to implement. Behind the component can stand whole application, which exposing the main component, or a page, that would be rendered.

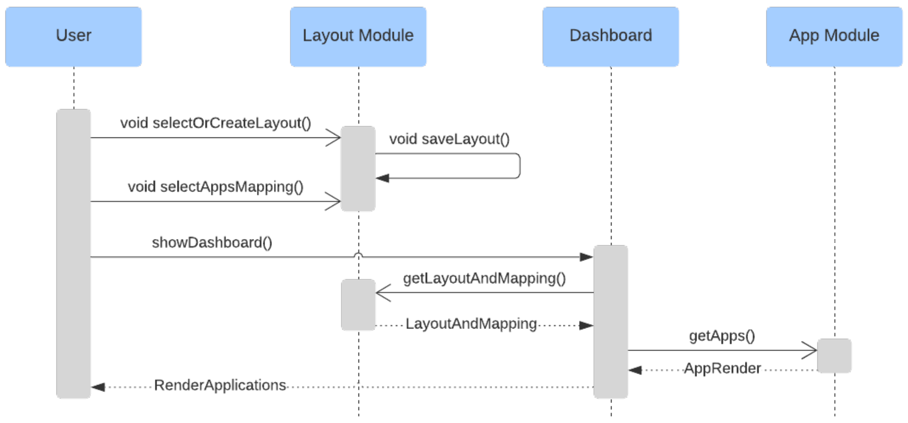


Figure 5.5: App module sequence diagram.

5.4 User module

The user module serves as a controller of the user data, user context, and is also responsible for distributing the context and its changes to the system. It consists of three main parts and be referred to as the user context, as it is basically controller for the whole user context. That includes tracker data, user configuration, and also the environment. The three parts are user context controller, store, and provider.

The user context controller is a main part of the module. It delegates the flow of the data from trackers to the store and then to the provider as well. References to trackers are also kept in the controller, as they can be turned off or on, which is a controller functionality. The tracker could be in another standalone module, but for this use case, it is better to keep then connected, yet separated logically, and control them from the user context controller, so it is easier to keep the correct data.

Next is the user context store. It is simply a store for all the user data, retrieved by trackers, configuration data, or an environment data. The store is represented by a simple store controller that can have a database, or a file system connection inside. But in the design it will be considered as one, because the store controller is just managing operations with the data inside the database, file system, or other. The added functionality to the store is a returned value from save or add operations, which is telling the if the data were changed or not. For example some trackers might produce data every second, so it is effective to know, if the result from the tracker is the same or different as it was before, which can be decided when storing the new data to the store.

The last part is the user context provider. The part that is responsible for emitting the changes of the store to the applications. The provider is based on an event-driven paradigm mentioned in the Background chapter. On every change or a trigger from a user context controller it generates an event and sends it to the system. Then there will be event listeners, that would consume the events with the provided data from the user context. Every event has its own key, mostly the name of the source, from where they were retrieved, and the data itself.

As shown in figure 5.6 the user context controller is responsible for the data flow. It triggers the trackers and listen to the results given by the trackers. Once it got data from trackers and add operations is triggered and data are stored into the context store. The store returns the information about if the data were changed. If they were, the context controller takes the data and passes them to the context provider. Context provider generates an event, fills it with the key and data, and emits the event to the system, or application.

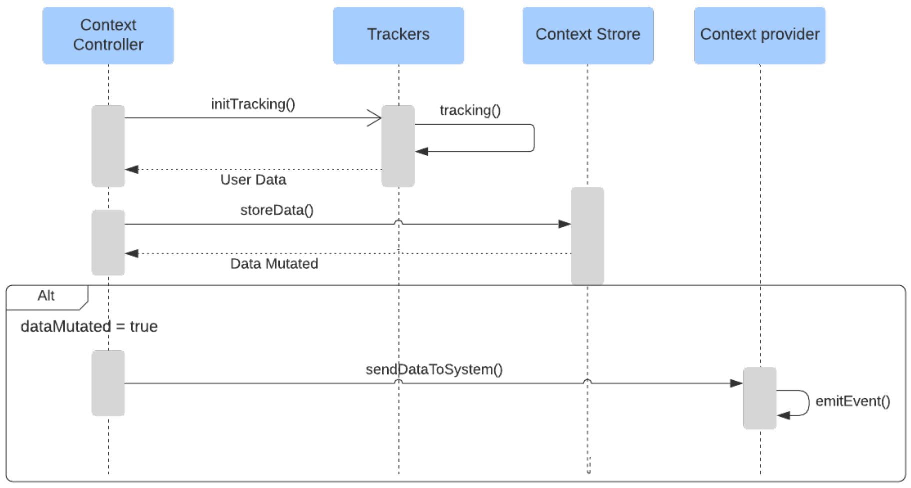


Figure 5.6: User context module sequence diagram.

5.5 Rules module

The rules module, as it was mentioned in the Analysis is sort of a middleware between the user context and application. Its purpose is to take provided data from the user context, apply rules to those data, and then pass the results of the applied rules to the application.

The rules inside the rules module are stored as a pipeline functions. Those functions have to parameters as the input. One is a key of the data from the context module and the second is the object with data. On the output, there is also the pair value of key and object with data, which might be changed by applied rule.

The rules module also has event listeners, that listens for the emitted events from the context store. As well as another provider, or more a notification emitter, which would send the notification to the applications, that specific rule was applied and an automatization or any other task must be executed.

The definition of the rules has a default rule, which just passes the data to the application without any mutation. This rule is for cases where no rule is requested by an application. Then there might be some other default rules, which can be generic for the whole application, like changing the light or context of the application. And the last type of definition are custom rules, that can be provided together with the custom application integration. The author of the custom application is able to write a custom rule, as a separate function, which has the same template as defined rules. Accepts pair *(key, value)* and return also pair *(key, value)*. Thus the user is able to create a custom rule for his custom app, that can return an object he desires. Meaning he is able to control the whole process of retrieving the data from user context and applying them to his application.

The usage of rules is specified together with registering an application. For every application there can be any number of rules, that would be used on the data. The configuration is made by specifying the user context source and adding a corresponding rule to that type of data.

Listing 5.5: Rules configuration example

```
"rules": {
    "active_app": "news-example-app/ExampleAppRule.js",
    "user_attention": null
}
```

The example in listing 5.5 specifies exactly two user context inputs for the application. The first one is `active_app` which is information about the currently used applications in the system. Second is `user_attention`. For the first one, the custom rule `ExampleAppRule.js` function is used. So the data from the `active_app` tracker goes through this function and then to the application. For `user_attention` there is no rule used, so data from this source are being just forwarded to the application. Without the specification in the `rules` configuration, the data would not be event sent to the application, that is why a `null` rule can be specified as well when the raw data are needed or the rule does not have to be applied on the back-end process.

At the end of the process data are emitted in the same principle as from the user context. But the final implementation might be different because the data are being forwarded to the applications running on the front-end part, so this part is implementation-dependent.

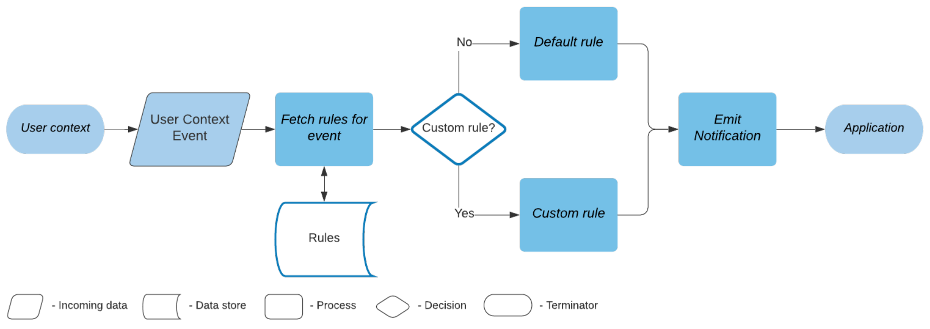


Figure 5.7: Rules module data flow.

5.6 Design summarization

Every part of the application was described and designed separately above. Those separated modules are designed in a way, that they can be replaceable and thus the application is modular as much as possible. Also the scalability is provided by this design, as every module can work in a standalone process and use different channels for data exchange, designed as event-based communication.

The overall behavior of the application is shown in the following diagram. Each module has its own behavior and their inner functionality and behavior were mentioned above, so this diagram is simplified to a module-level scope.

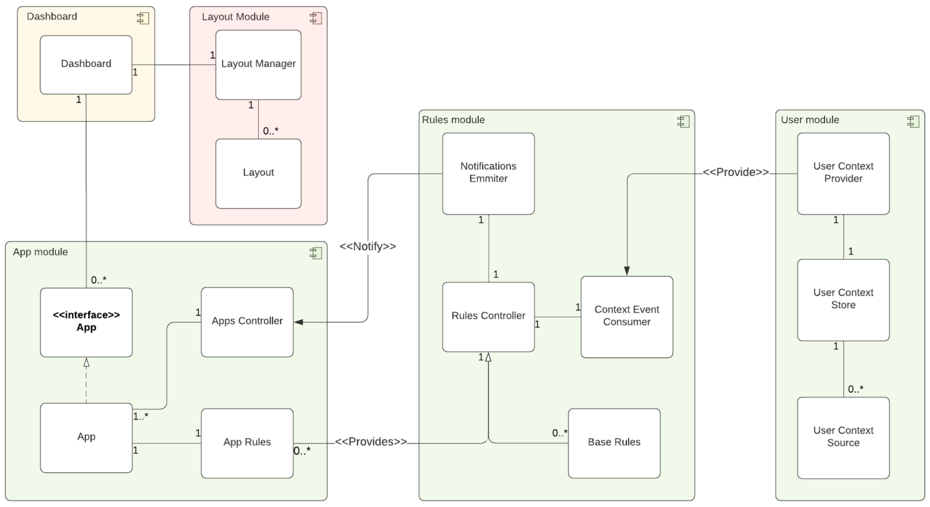


Figure 5.8: Application structure.

Chapter 6

Implementation

This chapter is describing the implementation of the prototype application which is following the designed architecture and representing the key features. The application is implemented in ReactJS using the Electron framework to work as a native application, so the user is able to run it without the web browser natively in the system as any other application.

The Electron framework provides a possibility to build cross-platform desktop applications using web technologies like HTML, CSS, and JavaScript. It is open-sourced and the application can be build to most of the platforms like Windows, Mac OS, and Linux. Also with correct configuration it can be also built as a web application as well. The framework provides two processes, that are used in the application. The main process contains the Electron instance and also integrates NodeJS and has access to all NodeJS features. The renderer process is visible for the user and basically is run by the Electron framework using chromium. Simply said, it behaves as a simple web browser. The Electron then renders the web page inside the renderer process using `BrowserWindow`.

Listing 6.1: Creating BrowserWindow example

```
const { app, BrowserWindow } = require('electron')

function createWindow () {
  // Create the~browser window.
  let win = new BrowserWindow({
    width: 800,
    height: 600,
    webPreferences: {
      nodeIntegration: true
    }
  })
  // and load the~index.html of the~app.
  win.loadFile('index.html')
}

app.whenReady().then(createWindow)
```

I have chosen these technologies mainly because of the personal experience with web application development and also because of the easily achievable modularity of the application. Web frameworks like ReactJS are component-based and that provides space for many improvements in the future. The integrated applications are in many cases web services, so the integrations are easier. This implementation also provides easier integration of custom applications and components as it was mentioned in the design. Those components are just written like any web component, which is easy to learn and implement then reproduce this feature in any other desktop technology, like in Java or C++ frameworks.

6.1 Requirements

The final built application can be installed like any other application in the system, so the requirements are dependent on the build for the operating systems. Then the usual hardware requirements should be met the same as for Google Chrome browser as the Electron framework is based on chromium using the V8 engine, same as Google Chrome and other web browsers.

For this thesis the application was not built for an installation as the built process for Electron is platform-specific and the configuration is quite complex. Also as the app is using experimental features, like eye-tracking, it is not very safe to not have control of the running processes in the background, that could be consuming many resources as it is not optimized for the production environment. The final requirements for this implementation will be mentioned together with a manual in the Installation chapter.

6.2 Project structure

The project structure combines a React web application structure for the rendering part and Electron usage for the main process part. The logical structure then copies the designed architecture from the Design chapter.

The modules that run in the main process, that includes trackers, rules module, user context module, and Electron itself are logically separated in the project. The app module is separated as well from renderer and main process, because of the access for developers and logically, it is a separate module providing the applications. Then in the React part of the application is structured as a usual web application, including the layout module, which is strongly front-end module. I will describe the structure more deeply from the React part to the main Electron part of the code.

6.2.1 Renderer process structure

The rendered process structure is basically a ReactJS application, which is being rendered by Electron. It consists of usual `index.js` and `App.jsx` as main files. The `index.js` is rendering and `App.js` inside the DOM and then other pages and components are being hooked to `App.js`.

Basic `react-router-dom` is used for navigating inside the application. Similar to web application, pages are routed using URL paths, like */setup-layout* path, that leads to the layout setup page. The overall state of the application is stored inside the React context. The context is then provided to the whole application so it is accessible from every module.

Listing 6.2: App.jsx file.

```
const App = () =>
    (
        <UserContextProvider>
            <Router>
                <BaseLayout>
                    <Routes/>
                </BaseLayout>
            </Router>
        </UserContextProvider>
    );
```

Providing the context is then resolved by propagating the context data to the children of the parent components, which is the `UserContextProvider` as can be seen in the above code snippet.

Listing 6.3: User context return

```
return (
        <UserContext.Provider value={value} {...props}>
            {children}
        </UserContext.Provider>
    )
```

The key part of the code in the renderer is the Layout module, which takes care of creating, selecting, and rendering layouts. In the implementation, the layouts are stored just inside the local storage of the application, which behaves in the same way as in the usual browser. In future implementations, this will be probably replaced by a database or any other persistent and more manageable storage. But for the purpose of this application, having local storage as a database is enough. The local storage also acts as a source of truth for the user context, where user context is being updated together with local storage and whenever it needs new data, the application reloads or just start, it fetches the data from local storage and distribute them across the application.

The layout module itself provides the selection of a layout. The user can pick an existing layout or create his own by simply adding new rows or columns.

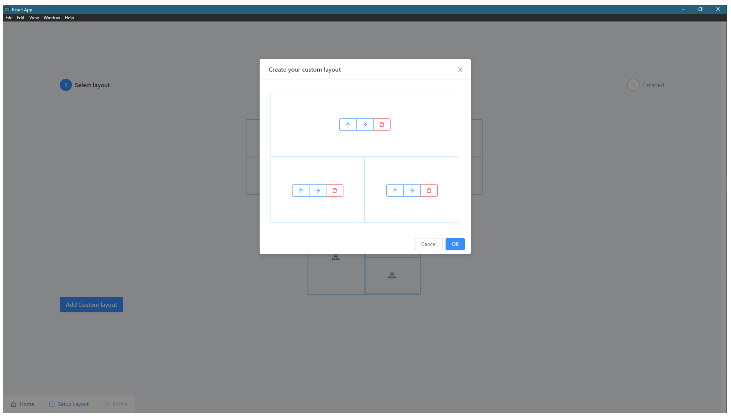


Figure 6.1: Custom layout creation.

The layouts are stored as a JSON format mentioned in the design part. Then rendered using a generic component, which is responsible for rendering all the layouts called `LayoutContainer`. It is a generic component, that takes four attributes. The layout, name of the layout column, which is a component that will be rendered inside the leaf nodes and custom styles. The column to be rendered is a generic component provided by a module, that needs to render the layout. For selection, those are the `PreviewContainerContent components`. For rendering the final layout with the application, it is an `AppModule` component, that takes care of rendering correct application.

Listing 6.4: Layout container

```
const LayoutContainer = (
    {layout, name, renderColumn, customStyles}
) => {
    const styles = `layout ${customStyles}`;
    const rootKeys = layout && Object.keys(layout)
        .filter(key => layout[key].level === 0);

    const renderColumns = (key, children, span) => (
        <Col className="col" span={span} key={key}>
            {children.length
            ? renderLayout(children)
            : renderColumn(key)}
        </Col>
    );

    const renderRows = (key, children) => (
        <Row
            className="row"
            key={key}
        >
            {renderLayout(children)}
        </Row>
    );

    const render = {
        false: renderColumns,
        true: renderRows,
    };

    const renderLayout = (keysToRender) => (
        keysToRender.map(key => {
            const {
                row = false,
                children = [],
                span = 24
            } = layout[key];
            return render[row](key, children, span)
        });
    );

    return (
        <div className={styles} key={name}>
            {layout && renderLayout(rootKeys)}
        </div>
    )
};
```

The rendered `AppModule` component is responsible for rendering service. The component gets the configuration of the service, retrieves the source of the main class implementing the App interface, and gets the instance of the application. The instance is then stored as a state of the `AppModule` component and passed to the `AppComponent` that renders the final application component and also is responsible for fetching the data from rules notifications.

Listing 6.5: AppModule component

```
const AppModule = ({app}) => {
    const [appInstance, setAppInstance] = useState(null);

    if (!app) {
        return <div>No app selected</div>
    }

    const {src, key} = app;

    const requireCustomApp = (file) => (
        require(`../../apps/store/${file}`)
    )

    if (!appInstance) {
        if (src) {
            const customApp = requireCustomApp(src);
            setAppInstance(new customApp.default(null));
        } else {
            setAppInstance(new AppInterface(app));
        }
    }


    return appInstance
            ? <AppComponent
                instance={appInstance}
                appKey={key}
              />
            : <div>loading...</div>;
};
```

The AppComponent is using Reacts `useMemo` for storing the listeners for the notifications. That prevents multiplying the listeners as to components re-render upon the data change. Those re-renders then provide the ability to adapt and make automatization or any changes of context inside them.

Listing 6.6: AppComponebt

```
const AppComponent = ({instance, appKey}) => {
    const [data, setData] = useState({});

    const processData = (inputData) => {
        const {key, value} = inputData;
        const newData = {...data};
        newData[key] = value;
        setData(newData);
    }

    useMemo(
        () => (
            NotificationEventListener(appKey, processData)
        )
    ,[]);

    return instance.component(data);
}
```

The listeners are listening for the events, or notifications, from the rules notification provider, same as mentioned in the design. Because the Rules module is running inside the main process, there is a need for inter-process communication. Electron provides a solution for this called IPC, inter-process communication, and it has two providers. The `ipcMain` is an Event Emitter and listener module for the main process, and `ipcRenderer` is for the renderer process. It can listen on any specified event specified by a string key value.

Listing 6.7: ipcRenderer listener

```
const NotificationEventListener = (appKey, callback) => {
    ipcRenderer.on(appKey, (event, data) => {
        callback(data);
    })
}
```

To summarize this flow, it can be looked from top to bottom as follow. Layout container renders the layout and for leaf nodes it renders the `AppModule` component, that is responsible for requiring the application. The `AppModule` then creates an instance of the application and passes it to the `AppComponent` that hooks the listeners, keeps the data of the application, and renders the application component. The application component is then rendered in the `LayoutContainer` for the user.

6.2.2 Main process structure

The main process is where the Electron runs. The `main.js` file is called on the start of the Electron app, it creates the window and then renders the application inside it.

Once the window is created, all other modules can be initialized. The main module, which is the core, is the `UserContext` module. It is a static class, in JavaScript represented by an object, that has the `init()` function, that initializes the trackers and then `emitData` function, which is used to delegate data across the modules. In the module, in JS it is represented by a `.js` file, also instances to the `UserContextStore` and `UserContextProvider` are kept.

Listing 6.8: UserContext module controller

```
export const store = new UserContextStore();
export const provider = new UserContextProvider();
const trackers = {};

const UserContext = {

    init: () => {
        for (let tracker in Trackers) {
            const trackerInstance = new Trackers[tracker]();
            const {name} = trackerInstance;
            trackers[name] = (trackerInstance);
            provider.initSubject(name);
        }
    },


    emitData: (key, value) => {
        const saved = store.save(key, value);

        if (saved) {
            provider.emitEvent(key, value);
        }
    },

}

export default UserContext;
```

The trackers are implemented by using a class `Tracker` that has a connection to the `UserContext`. The class provides the `emit` method of the `UserContext` so all trackers that extend this class have access to the `emit` function and can call it.

Listing 6.9: Tracker class

```
export default class Tracker {
    constructor() {
        this.emmit = UserContext.emitData;
    }
}
```

The `emit` function stores the data inside the `UserContextStore`. In this implementation, no database is used, the same as for the layout store in the renderer process. There is the only need for storing the latest values because the module needs to know if those data were changed or nothing in the context has changed. Thus function `save` return a Boolean value representing the change of the data. Then if data were changed, a provider is called and he emits the event with the data and key of the tracker.

The connection between the Rules module and Context provider is done using the RxJS framework. The context provider is keeping `Subjects` for every event source it has. Then through those subjects, events are emitted. Those subjects are then accessible in the application and every module that calls `.subscribe` on the `Subject` is going to be notified every time an event is emitted.

So at last a Rules module is initialized and during the initialization, one of the first steps is to set up the subscriptions for the subjects provided by the user context module. For every app, there can be any number of subscriptions based on how many trackers, or type of data, are required. For every tracked data, a subscription is created and stored. The subscription has a `next` method that is called every time an event occurs. It passes the value from the event and a callback is called. The callback then calls appropriate rule function, which is decided on the configuration if the application has a custom rule or a default rule should be used.

Listing 6.10: Subject subrscription

```
const ruleFunction = rules[tracker]
    ? this.requireCustomRule(rules[tracker]).default
    : DefaultRuleParser;

this.contextListeners[appKey][tracker] = subject
    .subscribe({
      next: (value) =>
      this.rulePipeline(ruleFunction(tracker, value), appKey)
    })
```

In the code from listing 6.10 a `ruleFunction` method is called, that apply the rule and the result of the rule is then passed to the `runPipeline` method. This method is using the `RuleNotificationService` to send a notification event to the renderer process. This service has a static method that sends a type of an IPC message to the renderer process. The drawback of `ipcMain` is, that it cannot initiate the communication and is designed only for replying to the request from the renderer process. Because of that, the `webContents.send` method, that is accessible on the instance of the `BrowserWindow` is called. This method triggers the event, that can be listened to by using `ipcRenderer` as mentioned above.

Listing 6.11: Sending data to renderer process

```
export default class RuleNotificationService {
    static setWindow (window) {
        mainWindow = window;
    }

    static sendNotification (data, appKey) {
        mainWindow.webContents.send(appKey, data);
    }
}
```

6.2.3 App module structure

The third part of the code infrastructure is the App module. It contains a store with all the custom applications, the `AppInterface`, used for application integration, and a config file, for configuring the paths to the applications. The generic component for rendering all the applications that use only URL is also stored in this module, called `WebViewContainer`.

The app interface is simple class following the design shown in listing 6.12.

Listing 6.12: AppInterface

```
export class AppInterface {
    constructor(url = null) {
        this.url = url;
        this.component = (_) => (
            <WebViewContainer app={{url: this.url}}/>;
        )
    }

    get getComponent() {
        return this.component;
    }
}
```

The applications just extend this class and set the `component` variable or `url` variable. The `component` variable should be a function that returns the component, by default it returns the `WebViewContainer` component using set `url` variable. So at least one of them must be specified.

The generic component `WebViewContainer` is responsible for rendering the application, or services, that are required only by defining URL. Rendering the specified URL inside the renderer process is made by using the `react-electron-web-view` library. It uses the Electron tag `<webview>` which is based on Chromium's `webview`.

Listing 6.13: WebViewContainer

```
const WebViewContainer = ({app}) => {
    const {url} = app;

    return (
        <ElectronWebView
            style={{width: "100%", height: "100%"}}
            className="web-view"
            autosize
            src={url}
            allowpopups
        />
    )
}
```

There is also the configuration JSON file for the integration of applications. It works as a place to register the users custom applications. Then all modules, that need to load custom functions, like the application itself or rules, are looking into this configuration. The following example shows a `News example app` that has a custom component in `src` and uses rules for two trackers and for one of them it provides custom rule function.

Listing 6.14: App configuration

```
"news-example-app": {
  "name": "News Example App",
  "src": "news-example-app/NewsExampleApp.js",
  "rules": {
    "active_app": "news-example-app/ExampleAppRule.js",
    "user_attention": null
  }
}
```

6.2.4 Eye-tracking and custom app implementation

In the prototype, one of the goals was to experiment with the eye-tracking implementation as the source of the user context and follow up automatization based on this input. That is using the laptop web camera to keep track of the user and his eye movement. The ideal result would be the information, where exactly is the user currently looking, and which application is he paying attention to. Then those data can be retrieved and based on the context and the application he is looking at, an adaptation or automatization can be made. The example use case can be the following. Application with news headlines and some basic description is scrolling or sliding through the news constantly, like in public TVs, so users can see more news at the moment. But once the user starts paying attention to the news application and starts reading the headline with the description, the scrolling should be stopped, so he has time to read through the description and does not have to wait until the headline is shown again.

I attempted to implement exactly this use case, using custom news application, that would use slide automatic slideshow and stops whenever the user starts paying attention to the slideshow. Finding the right library for eye-tracking was a challenge. Usually users use prepared software and special eye-tracking cameras, but that was not an option or it would be too complex for this prototype. Also, the resource consumption had to be taken in mind.

The best candidate is the OpenCV library, used in many existing eye-tracking solutions. This library is quite complex, I attempted to use their JavaScript solution, but there was an issue of running this script inside the Electron renderer process, as the compatibility was not ideal. The next possibility and the best one was running the OpenCV library in the main process. This solution would not slow down the rendering process and with the correct setup of the video from the camera, it could work out the best. The video input could be retrieved from the renderer process, using web principles and video input, or it could be handled in the main process using access to the system resources, which requires more complex setup and enabling access. Then the OpenCV library would run as a separate task connected to the main process using an `opencv4nodejs` library integration. The disadvantage of this solution is the OpenCV running on the background, almost separately, and needs to be installed as well. Also, the integration to the Electron is quite complicated and I was not able to sync the node module versions, as it is quite a common problem with Electron, because it has its own versions of node modules.

So the final solution was to use `wegGazer` [50], a JavaScript implementation of gazer tracking using a web camera. This solution is implemented and used for the news slide-show stopping. But it has its limitations, as the computation of the face and eyes position is made from data from the mouse pointer and camera, using two monitors then disrupts the mouse source and tracking is not very accurate. Also retrieving the precise information about where the user is looking is impossible. As the data from this `webGazer` are relative coordinates to the application and they are not very precise. So the only thing left is to track the focus of the user. Once the user stops moving his head and eyes, also cursor, it is possible to say he is paying attention. At this point it is not possible to determine whether the attention is to the news app or somewhere else. But once he starts focusing, the slide-show should be stopped, because he might be reading the headline and description of the article.

The computation of the attention is a simple tracking of average coordinates output from webGazer. When there is a constant output with some allowed deviation for three seconds, user is paying attention. The tracking module triggers this information to the tracker and the information goes through the same flow mentioned above and gets to the news application. Then the slide-show is stopped until the user makes a move and disrupt the incoming data over the allowed deviation. Then the opposite information is distributed and slide-show starts playing again.

Listing 6.15: Attention detection pseudocode

```
int sum = 0;
int count = 0;
int currentAvg = 0;
bool looking = false;

attention_tracking (coord) {
  if (abs(currentAvg - coord) < DEVIATION_CONSTANT) {
    sum += coord;
    count += 1;
    currentAvg = sum/count;
    if (count > 10 && !looking) {
      looking = true;
      emmit('User is paying attention');
    }
  } else {
    if(looking) {
      emmit('User stopped looking');
      looking = false;
    }
    sum, currentAvg = coord;
    count = 1;
  }
}
```

Current implementation runs in the renderer process as a standalone script, triggered by React application. The messages are then sent via `ipcRenderer` to the main process to the corresponding tracker, where the data flow starts as designed.

Another custom application implemented is the Spotify player and an active app tracker. The active app tracker uses `active-win` library for NodeJS, which is using an access to the native code and the system. It returns the currently active window in the system, so it is possible to track what application is the user currently using. With this information I implemented a simple demonstration of rule usage. The Spotify player stops playing whenever the user opens a browser window with YouTube in its title. Then whenever the user use a different application, the player starts playing again. This demonstration uses the implemented features as designed, so it proofs the concept.

6.3 Installation

The application prototype is not built as a package, that could be installed like a native application. The reason for this is the security and ability to track resource consumption because this implementation is not optimized well as it would add complexity above the goals of this thesis, as well as perfectly working eye-tracking solution, which is now not optimized at all.

For running the application, following is needed.

- Node 10

  (Can be installed from https://nodejs.org/en/download/)

- Yarn package manager

  (Can be installed from https://yarnpkg.com/getting-started/install)

- Python (node-gyp is used for building the application, which requires python installed)

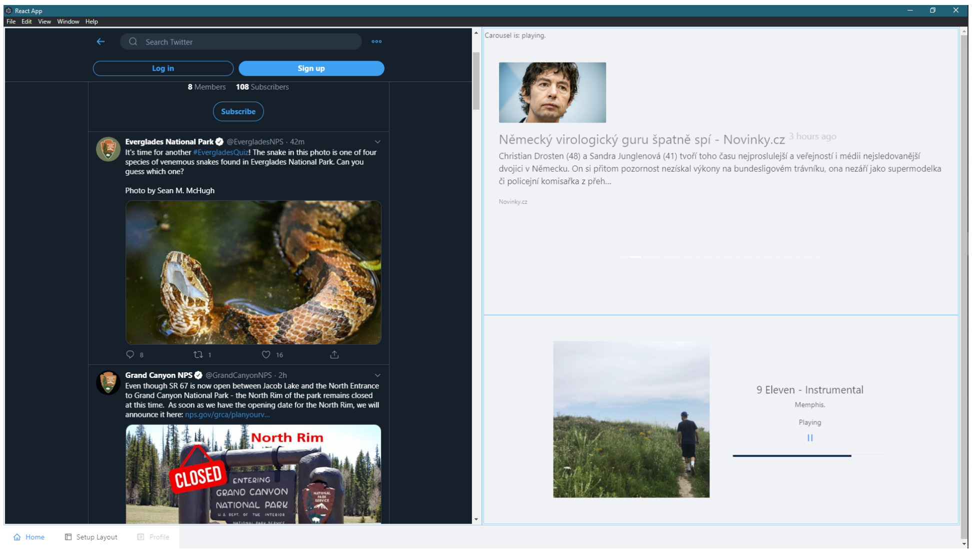
  (Can be installed from https://www.python.org/downloads/)

- For windows, windows-build-tools might be needed, also build-tools for Mac OS

  Installed using `npm install --global windows-build-tools`

  or `yarn global add windows-build-tools`

With those installed, running the `run.sh` script or `yarn run-app` will install, rebuild, and run the application. It starts up the React application first, then Electron application which renders the React app. Killing the script will close the application. Tested on Windows 10, Mac OS might have different behavior for shutting down the run script, read *README.md* file for more information.



8 Members 108 Subscribers

Everglades National Park @EvergladesNPS · 42m
It's time for another #EvergladesQuiz! The snake in this photo is one of four species of venemous snakes found in Everglades National Park. Can you guess which one?

Photo by Sean M. McHugh

8 1 16

Grand Canyon NPS @GrandCanyonNPS · 2h
Even though SR 67 is now open between Jacob Lake and the North Entrance to Grand Canyon National Park - the North Rim of the park remains closed at this time. As soon as we have the opening date for the North Rim, we will announce it here: nps.gov/grca/planyourv...

North Rim
CLOSED
ENTERING
GRAND CANYON
NATIONAL PARK

Carousel is: playing.

Německý virologický guru špatně spí - Novinky.cz 3 hours ago
Christian Drosten (48) a Sandra Junglenová (41) tvoří toho času nejproslulejší a veřejností i médii nejsledovanější dvojici v Německu. On si přitom pozornost nezískal výkony na bundesligovém trávníku, ona nezáří jako supermodelka či policejní komisařka z přeh...

Novinky.cz

9 Eleven - Instrumental
Memphis.
Playing

Figure 6.2: Preview of the application

Chapter 7
Testing and evaluation

In this chapter I will go through a brief testing of the implemented application and more through the evaluation of the results of this thesis and the idea.

For proper testing applications like this one, testing with users is the best solution. It would be preferable to do the testing for a longer time, track their behavior, and let them get used to the application features.[52] Then evaluate the progress they made after some period of time and see if the efficiency increased and the users are keen to use the application. But that would require almost production ready prototype, and overall usability is not in scope of this thesis. Tt is important to show the full potential of the application, so at least the basic services should be integrated and the overall user experience should be at a high level. Next is the installation and optimization of the resource consumption, so the application can be tested in the real environment with other applications and should not cause slowing the system.

To be able to test the application like mentioned above, the implementation must be close to the production product. But that is not the scope of this thesis, as the goal was to test the idea, approach, and feasibility of the solution. With the prototype implemented, it is not even possible to test in a real environment as the application with eye-tracking is very resource consuming. So the testing was made with the potential first target group for the application, which is developers, as they are able to use full potential and functions of the application and they were the target group for the analysis as well.

7.1 User testing

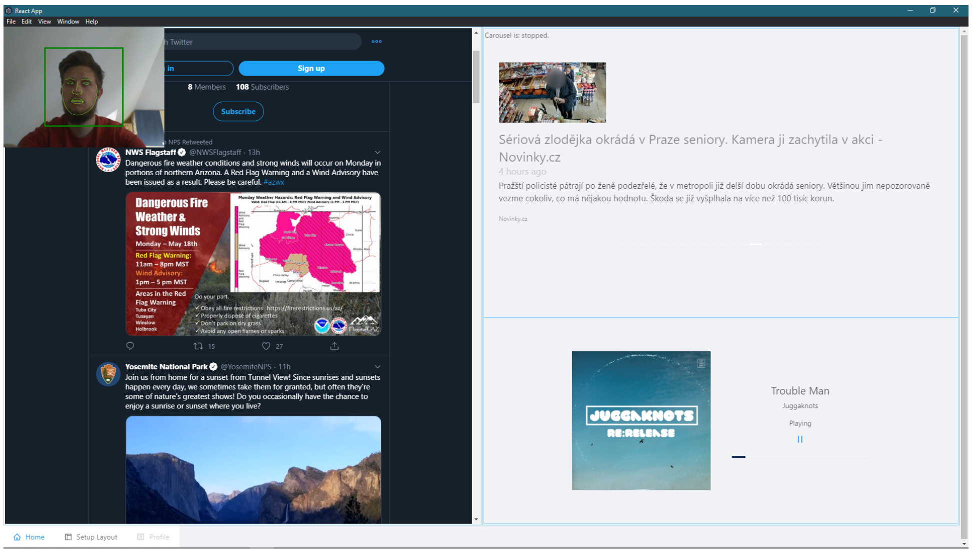
I prepared an easy script for the installation, as mentioned in the installation and two integrated applications. For the developers I tested with, a simple scenario was made that they followed to get the feel and idea of the application features and possibilities. Requirements for the developer was the same as for the installation and also a functional web camera to test the eye-tracking feature. The next but not mandatory was a Spotify account, so the Spotify custom app could be tested. During the testing they were given an explanation of the feature and how it works, so they are aware of the possibilities they have and comprehend the whole concept of the application. Testing was done with four participants, that went through the whole application scenario.[51]

The scenario was following. Firstly, they installed the application and run it.

- With no layout set, the first task was to create a custom layout, the structure of the layout was also defined.
- Next was the selection of the applications. Two integrated applications were selected, to the top-right node the news application, to the right bottom the Spotify application. On the left, any web services or site could be selected via inserting the URL.
- Once they have selected layout, they went to the home page, where the applications were rendered using selected layout.
- First, they tested the Spotify reaction on using the browser and opening the YouTube site, which would stop the playing Spotify song. It served as a demonstration also with an explanation of the possible enhancements of this feature.
- Secondly test of eye-tracking. The eye-tracking is run by right-clicking the News application and start the tracking. Based on their behavior, every time they focus on one spot, the news slide-show should be stopped and then after a user activity resumed again.

After going through these scenarios, a discussion was initiated on three main topics.

- Layout - selection, persisting and its functionality.
- Custom applications - style of implementation and usage.
- Rules and automatization.



NWS Flagstaff @NWSFlagstaff · 13h
Dangerous fire weather conditions and strong winds will occur on Monday in portions of northern Arizona. A Red Flag Warning and a Wind Advisory have been issued as a result. Please be careful. #azwx

Dangerous Fire Weather & Strong Winds
Monday – May 18th
Red Flag Warning: 11am – 8pm MST
Wind Advisory: 1pm – 5 pm MST
Areas in the Red Flag Warning
Tuba City
Tusayan
Winslow
Holbrook
Do your part.
✓ Obey all fire restrictions https://firerestrictions.us/az/
✓ Properly dispose of cigarettes
✓ Don't park on dry grass
✓ Avoid any open flames or sparks

15 27

Yosemite National Park @YosemiteNPS · 11h
Join us from home for a sunset from Tunnel View! Since sunrises and sunsets happen every day, we sometimes take them for granted, but often they're some of nature's greatest shows! Do you occasionally have the chance to enjoy a sunrise or sunset where you live?

Carousel is: stopped.

Sériová zlodějka okrádá v Praze seniory. Kamera ji zachytila v akci - Novinky.cz
4 hours ago
Pražští policisté pátrají po ženě podezřelé, že v metropoli již delší dobu okrádá seniory. Většinou jim nepozorovaně vezme cokoliv, co má nějakou hodnotu. Škoda se již vyšplhala na více než 100 tisíc korun.
Novinky.cz

Trouble Man
Juggaknots
Playing

Figure 7.1: Testing environment with eye-tracking turned on.

7.1.1 Results

Participant 1

The first participant was using the Mac OS system. Installation was smooth and also the layout setup went well. The only thing he mentioned was the user experience which is not very good as the application is just a prototype. Then he had some issues with login to Spotify, as to login opens in the whole application, not just in the node of the layout and it was hard to get back to the home section. Next issue during testing was the eye-tracking, which is being not very accurate and was making some glitches, which made it almost unusable for stopping the slide-show.

Discussion:

1. Layout Setting up the custom layout is a good practice and the participant is using it on his system as well. The real improvement and a key to success is the persistence of the layout and applications, and also the customizability.
2. Custom applications The integrated applications can be useful, but participants would not want to write any, so he prefers already defined ones. Also some sort of configuration for those apps would be preferable, to be able to customize even defined ones.
3. Rules and automatization According to participant it has potential, but needs to be specified well. Also some of the changes might be too distracting. The eye-tracking might be used for switching to focus, so he can start writing once he looks at the application. He sees potential in the rules based on the system context changes, like the working environment, and then switching to the gaming environment.

Participant 2

The second participant was using Linux distribution Ubuntu. The installation went smoothly after the requirements were installed and the application runs correctly. The participant had no issues, only the layout selection was not very intuitive, so I had to help him going through it.

Discussion:

1. Layout The layout selection is a similar idea to work-spaces and virtual desktop, so it could be useful. Also, the persistence is very useful.
2. Custom applications For participants, it added value to the overall application and it seems reasonable. He would need to see more services used as a custom application to try them.
3. Rules and automatization The participant is not a fan of eye-tracking and using a camera for automatization or adaptation, so he was very skeptical about it. Also it not very safe and the precision is not at

the point of using it in the production, even if a better library and framework were used, according to the participant. He would like to have some standby mode, that would save what he missed while he was away from the computer.

Participant 3

The third participant was using Mac OS and had some difficulties while installation, mainly because of the permissions setup in the system. He had no issues going through the scenario and the application was running smoothly until the eye-tracking was started. In his system it was quite unstable and unable to adapt to his movements, so the accuracy was bad.

Discussion:

1. Layout The creation of the custom layout should be more intuitive and provide more freedom for the user. The persistence of the layout comes handy.
2. Custom applications It is the most valuable feature of the application, as he can be using it for almost everything, that is web service-related.
3. Rules and automatization According to the participant, rules and automatization might not be even necessary for the whole application. He personally does not need them, maybe use them as a parental control, when the rules would be reacting based on the user.

Participant 4

The fourth participant was running Windows 10, the installation went smoothly and the application was running without any problems. The participant liked the overall idea, as it would improve the current solutions, that are still too manual and require too much interaction.

Discussion:

1. Layout The participant did not like the application selection, which is a simple prototype. It is a key part of setting up the layout, so in production it should be more intuitive and easy.
2. Custom applications The participant would like to see the configurations of the application when selecting them. Each configuration could be generated as a form, so it is more user friendly and the user can configure each custom app.
3. Rules and automatization The participant cannot really imagine using the rules widely, only for some small tasks. Also, he suggested, that some rules might be predefined for types of users. And then selected as a set of rules according to the user.

7.1.2 Summary

The testing of the application was adapted to the prototype implementation and the main goal of the testing was to gather feedback from potential target user group. The participants of the testing were helpful and provided useful feedback for potential future work as well as tips on what to improve and what to avoid. The most critical part of the testing was testing eye-tracking as it is not working as it should and the calibration with multi-monitor usage and embedded applications is just too unstable. So there was a need to more interact with participants, to give them more details about the ideal workflow of the eye-tracking related features.

In summarization, the concept was approved by all participants, and all the key features were found useful and interesting. That is a confirmation and proof of the overall concept described in this thesis. Another good feedback from the participants was an interest in the future of this project and shown interest in a production solution or any similar application, that would help them be more efficient during their work.

7.2 Personal evaluation

There was another perspective on the testing part in this thesis. It was to prove that the design an overall idea is feasible and can be implemented according to the design, without any changes in structure or functionality. As there was no similar application existing, the concept was challenged and implemented accordingly to the testing. The implementation architecture corresponds to the design and behaves as intended. Also the modularity was achieved, and any part of the implementation can be enhanced or replaced by a better solution without changing the overall structure or other modules.

The only negative result of personal and user testing was found in the eye-tracking implementation. The library *webGazer* was not an ideal choice and as it works quite well when the calibration is done properly (by simply moving around the application a keep eyes on the cursor), it is not useable for common usage, as the precision is small and also the data are hard to process. The testing with users also shown, that they do not trust in eye-tracking usage for automatization and adaptation of the application, as in common there are too many errors, even with high end tracking software and hardware.

Chapter 8
Conclusion

The goal of this thesis was to analyze and design a solution for an application for secondary monitor that would improve user effectivity while using multi monitor setup. Part of this thesis was proving the feasibility of the design by implementing the prototype. The key features of the solution were customizability, ruled-based automatization using user context data, custom app integration, and overall modularity. Furthermore, another goal was to implement the prototype based on design which would prove the feasibility of the solution. The prototype implements the core features and demonstrates the ruled-based behavior based on user context data retrieved from eye-tracking. Finally an evaluation and testing with users was performed.

In chapter 3, I went through the related existing solutions. Information from those solutions were then analysed in the chapter 4. In the analysis, a user research was done in order to understand user behavior and their needs when working with multiple monitor setup. Together with analysis of the problem, the base requirements for the design were set. To use the best practices and cover missing features and disadvantages, the solution provides easy layout customization and its persistence, then the ability to map selected services to the layout and their customizability. The rule-based behavior and automatization should be fully configurable, as well as the user context tracking.

Based on the analysis, the design focused on modularity and scalability. Only with high modularity, the final solution can be easily maintainable and customizable from the user perspective. The scalability is then needed in order to cover potential growth of the solution as more application can be integrated and user context can grow together with rule definitions and automatization patterns. The design presented the architecture with five main modules and covered the needs from the analysis. The designed solution was made as generic as possible to provide various possible ways of implementation, following modern trends and technologies.

The implemented prototype of the designed solution was written in JavaScript using technologies ReactJS and Electron, thus the application can be run on desktop machines and multiple system platforms. The implementation followed the designed architecture and proved its feasibility and also the modularity in the process of adding new features and enhancing the prototype.

The prototype includes an integrated example applications with default custom rules in order to evaluate the design. Implementation of those examples showed the benefits of the design and ability to easily extend the application. The only issues were in the implementation of eye-tracking source, which was described in section 6.2.4. The eye-tracking source of data is in overall a challenging problem, which was confirmed by users in section 7 who did not felt comfortable using it.

Despite concerns regarding eye-tracking, the results from user testing and evaluation of the solution were very positive. In general, users liked the application and the concept. Every feature was reviewed and explained to the users from the chosen group of developers. The most appreciated features were the custom layout creation together with persistence of the application mapping and the custom integrated applications. Some of them found the ruled-based automatization, very useful when used correctly, but were not sure it they would use it themselves.

To conclude the thesis, it was shown that the idea and designed solution is feasible. The implemented prototype successfully proved the architecture design and its key features - modularity and scalability, which means that the current drawbacks and imperfections of the implementation can be easily replaced or improved. The ability of easily extendable applications, context sources and rules was also achieved and shown. Regarding the evaluation from the users, the solution stood up well, all questioned users reacted positively and were keen to the idea of trying the application in production environment as it could improve their work experience.

Chapter 9
Future work

First of all, the overall application can be optimized and built for all platforms, so the installation is easier and the application can be installed by any user. That would mean prepare the built scripts, optimize the code in terms of memory usage and prevent the errors connected to these optimization. For individual platform, there might be different behavior or needs for menus, permissions and other, which need to be taken in mind as well.

9.1 Layout and apps

Improvements can be done on any module, the application is modular, so there is no need to improve all parts of the application together. First the layout module can be improved in terms of user experience to provide better feeling and be more intuitive. The layout store then can be stored in more persistent place like a database or for defined layouts, it can be packed in the installation bundle.

The app module has a big potential. The whole module can be separated from the application as standalone service storing the custom apps separately. Another future work might be introducing sub-package system. Meaning every custom application is working as a module with own packages and dependencies. Then it is built and used as another module inside the main app module. This would bring the possibility to add new custom applications to the existing built, at least for the web URL based apps. Regarding user research, some configuration process of the custom apps could be introduced, which would ease the workplace setup.

9.2 Rules, context and automatization

For the back-end part, meaning rules module, user context module and automatization, there is wide range of opportunities. There is even potential to create a knowledge base application, or module, that would have abilities to adapt the rules and behavior, meaning the automatization, based on the user context changes and user reaction to the changes. With this, the whole process of automatization can be improved while using the application and suit the user perfectly as it would be adapted to his needs. But as a start, the first step could be the user interface that would let user configure the rules, tracking data and automatization manually without touches to the code.

With better sources of the user context, improved eye-tracking the rules would have more potential. The data could be used for the custom applications as well, as they can be almost of any type. Data like computer statistic then could be parsed and showed in the custom application to the user, because they might have more value for the user then just a trigger for the rules.

Another interesting improvement of the user context data might be connection via public API. That would add ability to connect IoT devices, phone and other services to the application and provide more user context data or a way to interact with the application.

9.3 User and community

To create more user attractive and growing application, the user profile and user management system could be used. The user then would have the ability to see his configurations and change them in his profile.

Additionally, the custom layouts and custom application might be shared across the community. Every user would have the ability to publish his custom application, or contribute to others. By that, the selection of custom applications might increase and the user community could help with growth of the application. Mentioned sharing could also work for the custom rules and even user context trackers. As the app is modular and scalable it should not take much effort to introduce those sharing possibilities.

Appendix A

Bibliography

[1] MOYERS, STEPHEN. Understanding the Potential of Adaptive User Interfaces [online]. 2018 [cit. 2018-05-09]. Available on: https://speckyboy.com/adaptive-user-interfaces/

[2] PEDDIE, Jon. Jon Peddie Research: Multiple Displays can Increase Productivity by 42% [online]. 2017 [cit. 2020-05-18]. Available on: https://www.jonpeddie.com/press-releases/jon-peddie-research-multiple-displays-can-increase-productivity-by-42/

[3] Dual Monitors Boost Productivity, User Satisfaction [online]. 2011 [cit. 2020-05-18]. Available on: https://www.dell.com/downloads/global/products/monitors/en/dual_monitors_boost_productivity_whitepaper.pdf

[4] BATT, Simon. Do Dual Monitors Improve Productivity? [online]. 2018 [cit. 2020-05-18]. Available on: https://www.maketecheasier.com/do-dual-monitors-improve-productivity/

[5] Jacob M. Truemper, Hong Sheng, Michael G. Hilgers, Richard H. Hall, Morris Kalliny, and Basanta Tandon. 2008. Usability in multiple monitor displays. SIGMIS Database 39, 4 (November 2008), 74–89. DOI:https://doi.org/10.1145/1453794.1453802

[6] How Multiple Monitors Affect Wellbeing [online]. [cit. 2020-05-18]. Available on: https://www.steelcase.com/research/articles/topics/wellbeing/how-multiple-monitors-affect-productivity-and-wellbeing/

[7] OWENS, Justin W., Jennifer TEVES, Bobby NGUYEN, Amanda SMITH, Mandy C. PHELPS a Barbara S. CHAPARRO. Examination of Dual vs. Single Monitor Use during Common Office Tasks. Proceedings of the Human Factors and Ergonomics Society Annual Meeting [online]. 2016, 56(1), 1506-1510 [cit. 2020-05-18]. DOI: 10.1177/1071181312561299. ISSN 1541-9312. Available on: http://journals.sagepub.com/doi/10.1177/1071181312561299

[8] LEE, Sangsu, Hyunjeong KIM, Yong-ki LEE, Minseok SIM a Kun-pyo LEE. Designing of an Effective Monitor Partitioning System with Adjustable Virtual Bezel. KUROSU, Masaaki, ed. Human Centered Design [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, 2011, s. 537-546 [cit. 2020-05-18]. Lecture Notes in Computer Science. DOI: 10.1007/978-3-642-21753-1$_6$0.$ISBN978-3-642-21752-4.Availableon: http://link.springer.com/10.1007/978-3-642-21753-1_60$

[9] HUTCHINGS, Dugald Ralph a John STASKO. Quantifying the Performance Effect of Window Snipping in Multiple-Monitor Environments. BARANAUSKAS, Cécilia, Philippe PALANQUE, Julio ABASCAL a Simone Diniz Junqueira BARBOSA, ed. Human-Computer Interaction – INTERACT 2007 [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, 2007, s. 461-474 [cit. 2020-05-18]. Lecture Notes in Computer Science. DOI: 10.1007/978-3-540-74800-7$_4$2.$ISBN978-3-540-74799-4.Availableon: http://link.springer.com/10.1007/978-3-540-74800-7_42$

[10] TRUEMPER, Jacob M., Hong SHENG, Michael G. HILGERS, Richard H. HALL, Morris KALLINY a Basanta TANDON. Usability in multiple monitor displays. ACM SIGMIS Database: the DATABASE for Advances in Information Systems [online]. 2008, 39(4), 74-89 [cit. 2020-05-18]. DOI: 10.1145/1453794.1453802. ISSN 0095-0033. Available on: https://dl.acm.org/doi/10.1145/1453794.1453802

[11] MICHELSON, Brenda M. Event-Driven Architecture Overview [online]. 2011 [cit. 2020-05-18]. Available on: http://elementallinks.com/el-reports/EventDrivenArchitectureOverview$_ElementalLinks_Feb2011.pdf$

[12] ROUSE, Margaret. [online]. [cit. 2020-05-18]. Available on: https://searchitoperations.techtarget.com/definition/event-driven-application

[13] BARNKOB, Mark; KRUKOW, Jonathan. Event Sourcing and Command Query Responsibility Segregation Reliability Properties. Computer Science University of Aarhus.—2018.—C, 2018, 21-39.

[14] Model Event Sourcing [online]. 2017 [cit. 2020-05-18]. Available on: https://docs.microsoft.com/cs-cz/azure/architecture/patterns/event-sourcing

[15] DABEK, Frank, Nickolai ZELDOVICH, Frans KAASHOEK, David MAZIÈRES a Robert MORRIS. Event-driven programming for robust software. In: Proceedings of the 10th workshop on ACM SIGOPS European workshop: beyond the PC - EW10 [online]. New York, New York, USA: ACM Press, 2002, 2002, s. 186- [cit. 2020-05-18]. DOI: 10.1145/1133373.1133410. Available on: http://portal.acm.org/citation.cfm?doid=1133373.1133410

[16] MEIER, R. Taxonomy of Distributed Event-Based Programming Systems. The Computer Journal [online]. 2005, 48(5), 602-

626 [cit. 2020-05-18]. DOI: 10.1093/comjnl/bxh120. ISSN 0010-4620. Available on: https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/bxh120

[17] Event-driven programming [online]. In: . 2018 [cit. 2020-05-18]. Available on: https://www.computerhope.com/jargon/e/event-driven-prog.htm

[18] FOWLER, Martin. Event Sourcing [online]. 2005 [cit. 2020-05-18]. Available on: https://martinfowler.com/eaaDev/EventSourcing.html

[19] What is Knowledge System [online]. [cit. 2020-05-18]. Available on: https://www.igi-global.com/dictionary/towards-web-unifying-architecture-next/16468

[20] JIŘINA, Marcel. Znalostní systémy [online]. [cit. 2020-05-18]. In: https://courses.fit.cvut.cz/BI-ZNS

[21] ULLMAN, Jeffrey D. Predicate Logic [online]. [cit. 2020-05-18]. Available on: http://infolab.stanford.edu/ ullman/focs/ch14.pdf

[22] PIRNAY-DUMMER, Pablo, Dirk IFENTHALER a Norbert M. SEEL. Semantic Networks. SEEL, Norbert M., ed. Encyclopedia of the Sciences of Learning [online]. Boston, MA: Springer US, 2012, 2012, s. 3025-3029 [cit. 2020-05-18]. DOI: 10.1007/978-1-4419-1428-6$_1933.ISBN978 - 1 - 4419 - 1427 - 9.Availableon : http : //link.springer.com/10.1007/978 - 1 - 4419 - 1428 - 6_1$933

[23] Bayesian networks. Bayes server [online]. [cit. 2020-05-18]. Available on: https://www.bayesserver.com/docs/introduction/bayesian-networks

[24] HAYES, P.J. The Logic of Frames. Readings in Artificial Intelligence [online]. Elsevier, 1981, 1981, s. 451-458 [cit. 2020-05-18]. DOI: 10.1016/B978-0-934613-03-3.50034-9. ISBN 9780934613033. Available on: https://linkinghub.elsevier.com/retrieve/pii/B9780934613033500349

[25] JavaTpoint: Techniques of knowledge representation [online]. [cit. 2020-05-18]. Available on: https://www.javatpoint.com/ai-techniques-of-knowledge-representation

[26] SMITH, Reid G. Knowledge-Based Systems [online]. Schlumberger-Doll Research Old Quarry Road Ridgefield, CT USA 06877, 1985, May 8 [cit. 2020-05-18]. Available on: https://www.reidgsmith.com/Knowledge-Based$_Systems_-Concepts_Techniques_Examples_08 - May - 1985.pdf$

[27] HAYES-ROTH, FREDERICK, WATERMAN, LENAT a DOUGLAS B. Building expert systems [online]. 5. 1983 [cit. 2020-05-18]. Available on: https://archive.org/details/buildingexpertsy00temd/page/38

[28] BALL, Linden J. E Eye Tracking in Hci and Usability Research [online]. 2016 [cit. 2020-05-18]. Available on: https://www.semanticscholar.org/paper/E-Eye-Tracking-in-Hci-and-Usability-Research-Ball/51d4795836f562bc1bdc9375d2b70f560d68b56dextracted

[29] DUCHOWSKI, Andrew T. Eye Tracking Methodology [online]. Cham: Springer International Publishing, 2017 [cit. 2020-05-18]. DOI: 10.1007/978-3-319-57883-5. ISBN 978-3-319-57881-1.

[30] NIELSEN, Jakob a Kara PERNICE. Eyetracking Web Usability. New Riders, 2010. ISBN 9780321714077.

[31] JENSEN, Ole Baunbæk. Webcam-Based Eye Tracking vs. an Eye Tracker. Imotions [online]. [cit. 2020-05-18]. Available on: https://imotions.com/blog/webcam-eye-tracking-vs-an-eye-tracker/

[32] OBAIDELLAH, Unaizah, Mohammed AL HAEK a Peter C.-H. CHENG. A Survey on the Usage of Eye-Tracking in Computer Programming. ACM Computing Surveys [online]. 2018, 51(1), 1-58 [cit. 2020-05-18]. DOI: 10.1145/3145904. ISSN 0360-0300. Available on: https://dl.acm.org/doi/10.1145/3145904

[33] MÍKOVEC, Zdeněk. User Interface Design [online]. [cit. 2020-05-18]. In: https://moodle.fel.cvut.cz/courses/B4M39NUR

[34] RAMACHANDRAN, Krish. Adaptive user interfaces for health care applications. From the developerWorks archives [online]. 2009, 2009, 1-3 [cit. 2018-01-12]. Available on: https://www.ibm.com/developerworks/web/library/wa-uihealth/

[35] HASSENZAHL, Marc a Noam TRACTINSKY. User experience: a research agenda. Behaviour and Information Technology [online]. 2006, 25 (2): 9197 [cit. 2015-12-06]. DOI: 10.1080/01449290500330331. ISSN 0144929x. Available on: http://www.tandfonline.com/doi/abs/10.1080/01449290500330331

[36] Human-Computer Interaction (HCI). Interaction design foundation [online]. [cit. 2020-05-18]. Available on: https://www.interaction-design.org/literature/topics/human-computer-interaction

[37] Google Analytics [online]. [cit. 2020-05-18]. Available on: https://support.google.com/analyticstopic=3544906

[38] BRAGA, Reinaldo Bezerra, Sócrates DE MORAES MEDEIROS DA COSTA, Windson Viana DE CARVALHO, Rossana Maria DE CASTRO ANDRADE a Hervé MARTIN. A Context-Aware Web Content Generator Based on Personal Tracking. DI MARTINO, Sergio, Adriano PERON a Taro TEZUKA, ed. Web and Wireless Geographical Information Systems [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, 2012, s. 134-150 [cit. 2020-05-18]. Lecture Notes in Computer Science. DOI: 10.1007/978-3-642-29247-$7_1 1.ISBN978-3-642-29246-0.Availableon:http://link.springer.com/10.1007/978-3-642-29247-7_1 1$

[39] KOYCHEV, Ivan. Tracking Changing User Interests through Prior-Learning of Context. DE BRA, Paul, Peter BRUSILOVSKY a Ricardo CONEJO, ed. Adaptive Hypermedia and Adaptive Web-Based Systems

[online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, 2002-5-23, s. 223-232 [cit. 2020-05-18]. Lecture Notes in Computer Science. DOI: 10.1007/3-540-47952-X$_24.ISBN978-3-540-43737-6.Availableon: http://link.springer.com/10.1007/3-540-47952-X_24$

[40] BUTOIANU, Valentin, et al. User context and personalized learning: a federation of contextualized attention metadata. Journal of Universal Computer Science, 2010, 16.16: 2252-2271.

[41] KORPIPAA, P., J. MANTYJARVI, J. KELA, H. KERANEN a E.- J. MALM. Managing context information in mobile devices. IEEE Pervasive Computing [online]. 2003, 2(3), 42-51 [cit. 2018-01-13]. DOI: 10.1109/MPRV.2003.1228526. ISSN 1536-1268. Available on: http://ieeexplore.ieee.org/document/1228526

[42] HELD, Albert; BUCHHOLZ, Sven; SCHILL, Alexander. Modeling of context information for pervasive computing applications. Proceedings of SCI, 2002, 167-180.

[43] TAUBERER, Joshua. What Is RDF. In: XML.com [online]. July 26, 2006 [cit. 2018-01-13]. Available on: https://www.xml.com/pub/a/2001/01/24/rdf.html

[44] FERENC, Jakub. Kontextualizace a definice User Experience Designu. Academia [online]. , 2-5 [cit. 2018-01-12]. Available on: https://www.academia.edu/19563748

[45] Station [online]. [cit. 2020-05-18]. Available on: https://getstation.com/features

[46] Franz [online]. [cit. 2020-05-18]. Available on: https://meetfranz.com/

[47] Ferdi [online]. [cit. 2020-05-18]. Available on: https://getferdi.com/

[48] HARATY, Mona, Syavash NOBARANY, Steve DIPAOLA a Brian FISHER. AdWiL. In: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems - CHI EA '09 [online]. New York, New York, USA: ACM Press, 2009, 2009, s. 4177- [cit. 2020-05-18]. DOI: 10.1145/1520340.1520636. ISBN 9781605582474. Available on: http://portal.acm.org/citation.cfm?doid=1520340.1520636

[49] Cross Frame Scripting. OWASP [online]. [cit. 2020-05-19]. Available on: https://owasp.org/www-community/attacks/Cross$_Frame_Scripting$

[50] WebGazer [online]. [cit. 2020-05-19]. Available on: https://webgazer.cs.brown.edu/

[51] NIELSEN, Jakob. How Many Test Users in a Usability Study? [online]. 2012 [cit. 2020-05-19]. Available on: https://www.nngroup.com/articles/how-many-test-users/

[52] Usability Evaluation Methods. Usability.gov [online]. [cit. 2018-05-19]. Dostupné z: https://www.usability.gov/how-to-and-tools/methods/usabilityevaluation/index.html

Appendix B

Listings

Appendix C

List of Abbreviations

HCI	Human-Computer Interaction
UI	User Interface
AUI	Adaptive User Interface
SW	Software
HW	Hardware
OS	Operating System
URL	Uniform resource Locator
API	Application Programming Interface
RAM	Random Access Memory
NAS	Network Attached Storage
CSRF	Cross-site Request Forgery
XSS	Cross-site scripting
CORS	Cross-origin resource sharing
IDE	Integrated Development Environment