Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Further development of a Web Application based on Principles of Software Engineering

Bachelor Thesis

in Computer Science

by

Dinh Minh Nguyen

First Examiner: Prof. Dr. rer. nat. Kurt Schneider
Second Examiner: Dr. rer. nat. Jil Ann-Christin Klünder
Supervisor: M. Sc. Jianwei Shi

Hannover, 22 August 2022

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Bachelor Thesis selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 22 August 2022

Dinh Minh Nguyen

Zusammenfassung

SynchroPC ist eine Webanwendung, die entwickelt wurde, um effiziente Agenden für wissenschaftliche Programmausschüsse zu erstellen. Der Algorithmus, der zum Erstellen von Agenden verwendet wurde, funktioniert jedoch nicht optimal und ist für die meisten Benutzer schwer zu verstehen. AuSSerdem gibt es Schwachstellen im System, die behoben werden müssen, und neue Anforderungen kommen von den Kunden. Ziel dieser Arbeit ist es, die Webanwendung weiterzuentwickeln, um den Algorithmus erklärbarer zu machen, die noch im System vorhandenen Fehler zu beheben und neue Funktionen hinzuzufügen, um die Qualität des Systems zu verbessern.
Um dieses Ziel zu erreichen, werden Django- und Boostrap-Frameworks zusammen mit der jQuery-Bibliothek verwendet, um die Anwendung weiterzuentwickeln. Die verbesserte Version von SynchroPC bietet einen neuen Algorithmus zum Erstellen von Agenden, der einfacher zu verstehen ist und bessere Agenden erstellt, sowie neue Tools, um die Verwendung von SynchroPC flexibler und komfortabler zu gestalten.
Um die Qualität des neuen Algorithmus zu testen, wurden eine Reihe von Tests durchgeführt, um die Ergebnisse des alten und des neuen Algorithmus zu vergleichen. Die Ergebnisse der Prüfungen werden protokolliert und ausgewertet.

Abstract

SynchroPC is a web application developed to create efficient agendas for scientific program committees. However, the algorithm that has been used to create agendas is not working optimally and is hard to understand for most users. There are also flaws in the system that need to be resolved, and new requirements are coming up from the customers. This thesis aims to develop the web application further to make the algorithm more explainable, fix the errors that are still in the system and add new features to improve the system's quality.
To achieve this goal, Django and Boostrap frameworks, along with the jQuery library, are used to develop the application further. The improved version of SynchroPC provides a new algorithm for creating agendas, which is easier to understand and create better agendas, and new tools to make the usage of SynchroPC more flexible and comfortable.
In order to test the quality of the new algorithm, a number of tests have been conducted to compare the results from the old and new algorithms. The results from the tests are recorded and evaluated.

Contents

Chapter 1

Introduction

1.1 Motivation

The peer review process plays a pivotal role in the world of researchers and scholarly publishing. According to Kathryn S McKinley [1], peer reviewing seeks to evaluate research using qualified experts, who have the scientific training, experience, and skills to judge the merit of the work - they are also known as program committees. However, many program committee meetings are unproductive and a waste of time. In reality, holding a practical and purposeful meeting can be pretty challenging, and many problems can occur in the process. That is why meetings must be well planned and appropriately chaired.
SynchroPC is a web application developed to create an influential agenda for program committees, synchronize individual agendas, and create a meeting environment for all participants worldwide. This web application's simple yet effective workflow is the key that led to its success at the International IEEE Conference on Requirements Engineering. However, it was developed quickly, so problems still lie within it. Along with new requirements, there is much work to bring this web application to its fullest potential.

1.2 Goals Definition

This thesis aims to improve synchroPC further based on the principles of Software Engineering. During the development process, the algorithm used to create the agenda is optimized and more understandable so that users can create agendas without difficulties and misunderstandings. Furthermore, new front-end functionalities for account activation and administration are also being considered to enhance user experience with these processes. Along with improvements to ready-to-use features, new Security and Usability requirements are also identified and collected using Requirement Engineering before they are implemented and migrated into the web application.

Throughout this thesis, the Principles of Software Engineering should be strictly followed. Requirement Engineering is used to collect stakeholders' requirements, ensuring that solutions to every problem are reasonable, complete, and practical. The whole project is managed using agile practices to guarantee the quality of work and end product with time-effectiveness. Testing is also developed as an essential part of development, ensuring that every step of development is in the correct direction and estimating the project's progress. The result of this thesis will be evaluated carefully using acceptance tests, which are discussed at the beginning of the project. The implementation chapter will elaborate on the details of how these principles are applied.

1.3 Structure of the Thesis

The content of the thesis is structured as follows. In this first chapter of the thesis, motivation and goals are discussed, as well as the structure. Chapter 2 defines the fundamentals of the principles of Software Engineering that were used to achieve the goals of this thesis and also provides necessary knowledge of additional concepts and tools. Essential requirements are described in the third chapter, as well as the stakeholders of this product. Chapter 4 goes into details about the further implementation of the web application on the back-end and front-end, as well as the testing process. Chapter 5 focuses on the process and results of the evaluation for this web application, ensuring that this project's goals are fulfilled. Finally, chapter 6 summarizes and concludes the thesis and mentions some ideas and improvements for the future development of the web application.

Chapter 2

Fundamentals

In this chapter, the fundamental knowledge required for the further development of this project is declared. Section 2.1 goes into a rough description of synchroPC and what it does. Section 2.2 presents what agenda is and its role in synchroPC. Section 2.3 discusses the Principles of Software Engineering and their practices. Furthermore, finally, the library and tools used during the development are introduced in the last subsection of this chapter.

2.1 synchroPC

synchroPC is a software solution developed to hold program committee meetings at international conferences. Its primary function is to create an agenda that all conference members can follow to evaluate scientific papers effectively and deliver accurate decisions about these papers, whether they are accepted, rejected, or suspended. Information from EasyChair, a free web-based conference management software system used, is also importable to the web application. Users can work right away without importing it one by one manually.
This web application has three prominent user roles: PC-Chair, Member, Conflict of Interest - COI. PC-Chair is the person who holds and has the highest control over a conference. As a PCChair, a user can invite another E-Mails member to participate in the already created agenda. However, if a member is involved as a COI, this user will not be allowed to evaluate that paper. After the discussion, a PC-Chair can mark a paper as accepted, rejected, or suspended and move on to the following paper.

2.2 Principles of Software Engineering

According to Schneider [2], the Principles of Software Engineering are the principles that should be implemented by both traditional and agile

development approaches. As a part of this projects requirement, agile methods are applied to this project. Therefore, the Principles of Software Engineering are implemented using best practices for the agile development process. This section is focused on principles and practices that are essential and used for the project.

2.2.1 Working systematically

When it comes to a professional software project, everything needs to be prepared beforehand. Any software system is created using entities and relationships, and these parts and their dependencies must be discussed before implementing the desired system.

Incremental Development Model

The Incremental Development Model is fitted perfectly with our project, as requirements tend to change during development time. Over the traditional waterfall model, the cost of modifications is significantly reduced. Moreover, the product can be gradually developed and deployed, facilitating regular customer feedback. This way, customers and the development team can easily track the progress of the whole project so that they can make adjustments or improvements in the next increment. Despite some disadvantages from the managing perspective, such as invisible progress or degraded system structure, the Incremental Development Model is still a great fit, especially for this small project.

Agile Software Development

As mentioned in Chapter 1, there is much work to bring this web application to its fullest potential. However, the further development of the web application is in a short period, which is four months. That means people should spend more time on this project, primarily developing and testing. In addition, it is expected that customer requirements will change in the future. Even for customers, actual requirements are sometimes unclear until they develop experience with the system. Further development process should also embrace these changes in the near future. In order to deal with these challenges, agile methods are applied to the project.
According to Sommerville [3], agile methods are incremental development methods in which the increments are small, and typically, new releases of the system are created and made available to customers every two or three weeks. Based on the Manifesto for Agile Software Development [4],agile methods have become popular due to their advantages over traditional, plan-based development models. They allow the software to be delivered quickly, make it easier to adapt to changes and avoid unnecessary bureaucracy by using informal communications. In addition, customers are closely

involved in the development process, and consequently, the system can be evaluated frequently during each iteration. With those benefits, agile methods have been particularly successful for custom system development within an organization, where there is a clear commitment from the customer to become involved in the development process and where there are few external stakeholders and regulations that affect the software, as stated by Sommerville [3].

Extreme Programming

There are many practices that were built from agile methods, but the most effective practices that are implemented in this project is a set of Extreme Programming practices. According to Beck [5], the approach was developed by pushing recognized good practices, such as iterative development, to extreme levels. However, not all of the practices can be used. These must be applied based on the suitability of management practices and the development team's culture. In conformity with the project, several Extreme Programming practices are listed and explained below in Figure 2.1, according to Sommerville [3]:

Principle or practice	Description
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Incremental planning	Requirements are recorded on "story cards," and the stories to be included in a release are determined by the time available and their relative priority. The developers break these stories into development "tasks." See Figures 3.5 and 3.6.
On-site customer	A representative of the end-user of the system (the Customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.
Refactoring	All developers are expected to refactor the code continuously as soon as potential code improvements are found. This keeps the code simple and maintainable.
Simple design	Enough design is carried out to meet the current requirements and no more.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Sustainable pace	Large amounts of overtime are not considered acceptable, as the net effect is often to reduce code quality and medium-term productivity.
Test first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.

Figure 2.1: Some of the Extreme programming practices

2.2.2 Take requirements into account effectively

The idea of how a system works and controlled is described through requirements. This is the reason why requirements have to be collected as

the first step of development process using **Requirements Engineering** technique. According to Metzner and Verlag [6], in requirements engineering are software's requirements determined, analyzed, described and modeled as technical solution. It is an iterative process that interleaves with each increments of development process. The components of requirements engineering and their dependencies are described in Figure 2.2, as presented by Sommerville [3].

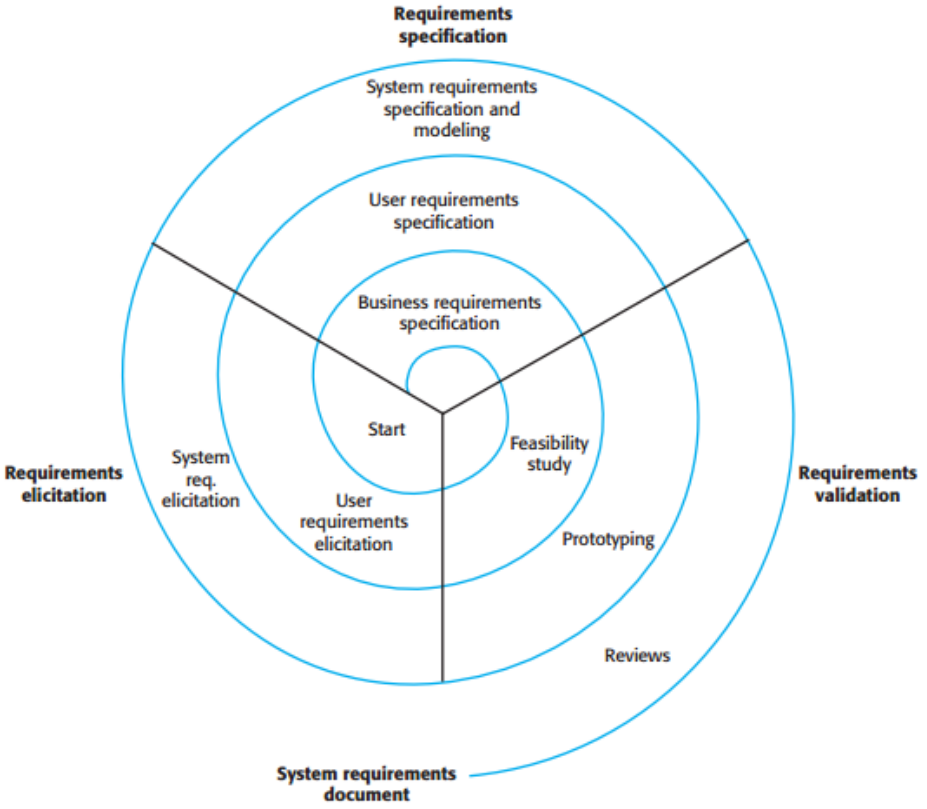

Figure 2.2: A sprial view of the requirements engineering process

Requirements Engineering consists of five stages, which are: *Elicitation*, *Interpretation*, *Negotiation*, *Documentation*, and *Validation/Verification*.

- **Elicitation**
  In the requirements elicitation, customer's requirements will be gathered through communications between software engineers and stakeholders in the requirements elicitation. Firstly, stakeholders must be identified as individuals, groups, companies, or associations impacted by the software project results. Next, the system's environment should be examined, consisting of old systems, interfaces, or documentation. Last but not least, communication between interested parties through

interviews, workshops, or surveys is established in order to obtain requirements. Moreover, finally, the collected requirements are refined into more technical and understandable requirements, which will be used in the next step.

- **Interpretation**
  From elicitation stage, the refined requirements are interpreted, sorted, and ordered from the elicitation stage based on category and their significance to the project. Afterward, they are detailed and concretized into formal requirements.

- **Negotiation**
  Using formal requirements, dependencies and conflicts between requirements are addressed. Inconsistencies are taken into account, and the solutions to them are discussed to find a compromise that pleases all parties involved in the software project.

- **Documentation**
  According to Sommerville [3], requirements documents are essential when systems are outsourced for development, when different teams develop different parts of the system, and when a detailed analysis of the requirements is mandatory. The software requirements gathered from the previous step should be fixed in formal documentation, which can be used by both customers and developers of the system. However, it is essential to note that requirements can be changed during the software development process, and these changes should also be provided in the documentation.

- **Validation/Verification**
  Validation and Verification are the last steps of the requirement engineering process. Both of them are especially important in order to prevent extensive rework and retest costs in the future. Requirement Validation checks if the provided requirements are what the stakeholders want from the software. On the other hand, Requirement Verification is the process of checking if the software requirements are matched with the required documentation in the previous step.

2.2.3 Programming understandably

Understandability is essential for changing, maintaining, and further developing software. Along with new requirements, new codes are also added to the system, which degrades the software structure over time. That leads to the fact that changing, maintaining, and further developing will become more expensive. For that reason, good code understandability will

make it much easier to work with the system and prevent errors during the development process.
In order to adopt this principle, Code and Commit Conventions are applied in the whole project. The Implementation chapter will discuss how these concepts are applied and implemented.

2.2.4 Uphold Testing and Quality

Developers always want that their software works as expected. Nevertheless, in reality, this is not always the case. People make mistakes, and developers are also people - they are good in their expertise but not perfect. Even when the software works, we can only say it works at one point. Moreover, is a working software good enough? According to Ousterhout [7], if development only focuses on getting the software working as quickly as possible, it is inevitable that it lacks a good vision for the future. As a result, adding more codes in the future will increase the system's complexity and raise the cost of the development process. These are the reasons why software quality should always be taken into account for any software project. This section will discuss essential techniques that are utilized to uphold the quality of this project.

Quality Model

In order to evaluate quality of a software project, the quality model proves to be useful. Based on ISO/IEC 25010 [8], there are, in total, eight quality characteristics in the general quality model, which is used when evaluating the properties of a software product. They are as described in Figure 2.3.

Figure 2.3: The eight aspects of Software Quality

However, it depends on the customers and which aspects are essential for the product. Therefore, not all of the characteristics are used to construct the quality model for this project, as some are either not required by the customers or may conflict with other aspects, which are more important to consider. In this project's scope, the following quality characteristics are put into practice, with the importance in listed order: Security and Usability.

- **Security**
  According to ISO/IEC 25010 [8], Security is the degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization. As much information about the program committee and scientific research is kept in our system, and the web application must be secured to guarantee the integrity of the paper evaluation process and protect personal information.
  System vulnerabilities may arise because of requirements, design, or implementation problems, or they may stem from human, social, or organizational failings, based on Sommerville [3].
  The Requirement and Implementation section will discuss how this characteristic is enforced.

- **Usability**
  Usability refers to the satisfaction degree when a specified user uses the product to achieve their goals. The software has good usability when users are appeased with the effectiveness and efficiency which they experience. It is typically an important goal of quality, and there are a lot of tools, libraries, and frameworks that have been developed in order to achieve that goal. However, getting lost in technical possibilities is easy, and people forget what the application was intended for. In order to avoid that, fundamental knowledge about users and their tasks must be taken seriously. According to Schneider [9], the following quality aspects are distinguished for the adaptation of software to humans:

  – **Utility**
    First of all, the required function must have existed in the software. Which functionalities should be available for each type of user must be found out during the requirement engineering process. Otherwise, the application will not help its users much with their tasks.
  – **Correctness**
    An application should not just provide functionalities, all of them must also work as expected.
  – **Usability**
    It is the degree of how useful and corrects the provided functions are in reality to its users, and this aspect is more critical than Utility and Correctness.

  Utility and Usability are sometimes contradictory: a simple application can be helpful but will not help much. On the other hand, complicated software can provide all the users need, but such a

complicated is more challenging for its users to learn and operate by themself. In this project, the application should provide only essential functionalities so users will not feel overwhelmed.

Testing

he quality of a software product cannot be kept over time without a good set of test cases. According to Spillner and Linz [10], testing is the program's execution to prove the effect of errors, determine the current quality degree, increase trust in the program, and prevent the cost of error in the future. In the scope of this project, Unit-Testing has been used in order to uphold the quality of the system.
According to Schneider [9], a Unit Test is characterized by the level in the system hierarchy. Modules are tested; the modules or units are usually classes or simple packages in object-oriented languages. Integration and system tests do not fall into this category. Testing should be done from the bottom up. Modules should be thoroughly tested before they are put together in the system structure, and the integration and system test begins. All module tests must run correctly before the modules are integrated, but that does not mean that about 100% statement coverage or perfect requirement coverage would be achieved.
Along with Unit-Testing, several tests are designed to test the quality of the Magic Algorithm. In Chapter 5, the details of the tests and its result will be discussed.

Chapter 3

Requirements

In this chapter, requirements which need to be fulfilled are taken into account. In sections 3.1 and 3.2, all functional and non-functional requirements taken during the development process are listed and briefly explained.

3.1 Functional Requirements

- [**R01**] *The web application should be further developed using Django, along with jQuery and Boostraps*
  With built-in administration and protection for common security issues, Django is a perfect framework for this project. With two powerful Javascript-Framework, Django enables rapid development with a feature-rich user interface.

- [**R02**] *The algorithm for creating agenda should be refined and configured so that it is clear and understandable for all users*
  The algorithm for creating agenda is complex and still has flaws, which leaves much space for improvement.

- [**R03**] *Front-end functions for account activation and administration should be modified or implemented*
  The design of the web pages for account activation and administration still has some problems. In order to increase user experience, these problems have to be removed.

- [**R04**] *PC-Chairs should be able to make agenda public or private, in order to prepare the agenda efficiently*
  Currently, all users can see an agenda once a PC-Chair creates it. This initialled agenda is not yet edited by the PC-Chair and requires more work before it can be published. All members must only see the agenda once it is ready.

- **[R05]** *PC-Chairs should be able to undo their agenda editing*
  Although the agenda can be easily edited using drag and drop, it is possible to make mistakes while editing. In order to prevent that, undo feature should be available for all PC-Chairs.

- **[R06]** *PC-Chairs should be able to change agenda decision*
  After an agenda is discussed and decisions are made, there is no turning back. PC-Chairs may click on the wrong decision, which should possibly be corrected.

- **[R07]** *PC-Chairs should be able to reschedule suspended paper and all members see it on status page*
  A discussion of a paper can be suspended by PC-Chairs, in order to discuss it at a better time. They should be able to resume the discussion by the time they decide.

3.2 Non-functional Requirements

- **[NR01]** *The Principle of Software Engineering should be endorsed over the project*
  Every step and every process in the project should be strictly followed planed and principles.

- **[NR02]** *Agile Practices should be applied during the development process*
  The Agile Method is the most popular in the software engineering world. It suits the best for this project, as requirements can be changed during the development process.

- **[NR03]** *The application should have a secured authentication system*
  With Django Framework, security authentication should be enabled.

- **[NR04]** *The system should provide accessibility and visibility based on the role of the user that is using it*
  In order to keep transparent of the paper evaluation process, important information should only be visible to authenticated users of the system

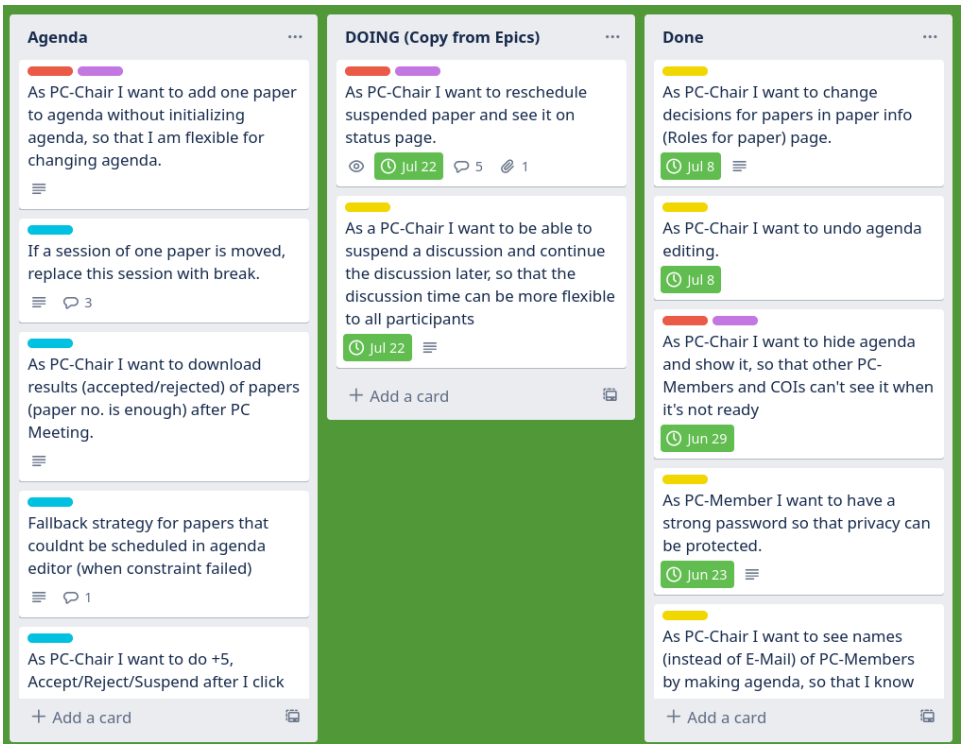

Figure 3.1: A part of the Trello Board that contains all tickets for SynchroPC project

Chapter 4

Implementation

In this chapter, the detailed implementation of the web application is discussed. Section 4.1 briefly explains how the web application works. Section 4.2 describes the back-end of the web application. After that, the front-end improvements will be considered in section 4.3. Following that, section 4.4 describes the testing process. Furthermore, finally, some consideration thoughts about the results and feedback are clarified in section 4.5.

4.1 Workflow

This web application has three types of users: PC-Chairs, PC-Members, and Conflict of Interest, also known as CoI.

- **PC-Chairs**
  A PC-Chair is a leader of the program committee, and he has to effectively plan an agenda and send it out to other members of the committee in sufficient time so that all members of the program committee can attend the meeting thoughtfully. He is also the want who moderates the whole meeting and represents the committee in deciding whether a paper will be accepted, rejected, or suspended.
- **PC-Members**
  A PC-Member is a part of the program committee, and he takes part in the meeting, along with other members, to discuss the fate of a paper. However, it is not a particular members right to decide but the PC-Chairs.
- **CoIs**
  A CoI is also a PC-Member, but he is involved in one or more papers that will be discussed in the meeting. That is why he can not participate in the evaluation process of a paper he is involved in, as his opinions can be biased.

In the beginning, a PC-Chair, who is authenticated in the system and verified by an admin of the system, wants to host a program committee. In order to do that, he can invite other PC-Members, using their E-Mail adresses to create user accounts for them and links to reset passwords so that they can use their accounts privately. After that, he sends the reset password links to the interested PC-Members so that they can log in to the web application. Alternatively, he can import data directly from the easyChair web application and use the imported E-Mail-Addresses to create a reset passwords link. For the papers that will be discussed, PC-Chair can add them manually or import them from easyChair. After the accounts and papers are ready, he can create an agenda based on that. The agenda is created using an algorithm from the web application, and other members can also see it. Next, the PC-Chair will edit the agenda to pass to others' schedules. After finishing the editing, he can start to moderate the meeting by discussing the first paper on the agenda. During this process, he can extend the discussion time, decide the paper's fate and move on to the following paper. He can also provide public notification and online meeting links so that others can see what he has to say or where they can take part in the meeting online. The current state of the meeting is visible to other PC-Members and PC-Chairs. PC-Members, however, are not allowed to see PC-Chairs' decisions. After going through discussions of all papers, all program committee members can log out and log in in the future when another meeting is hosted. The sequence of actions before, during, and after each session of synchroPC based on roles is shown in Figures 4.1 and 4.2.

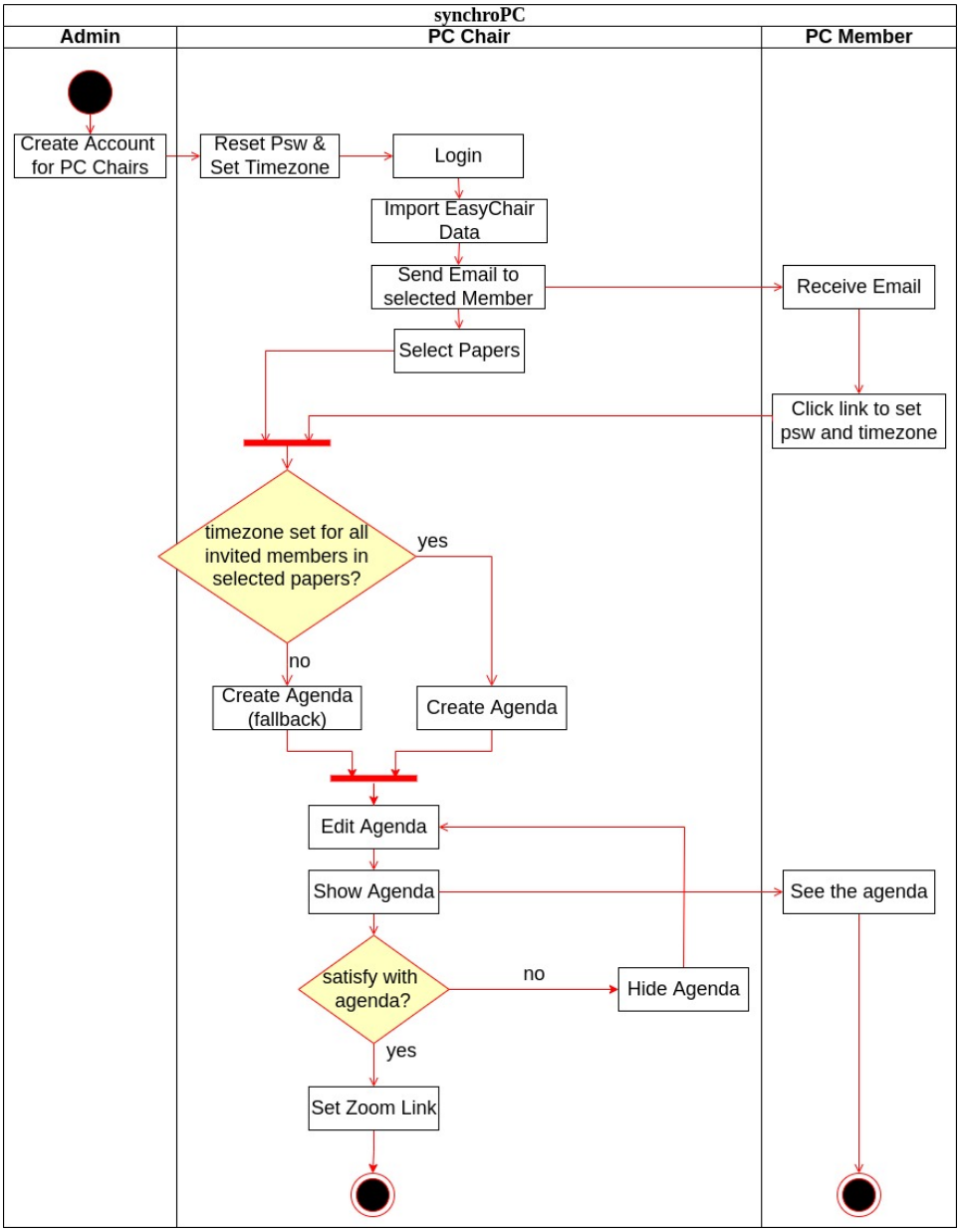

Figure 4.1: Sequence Diagrams - Before Meeting

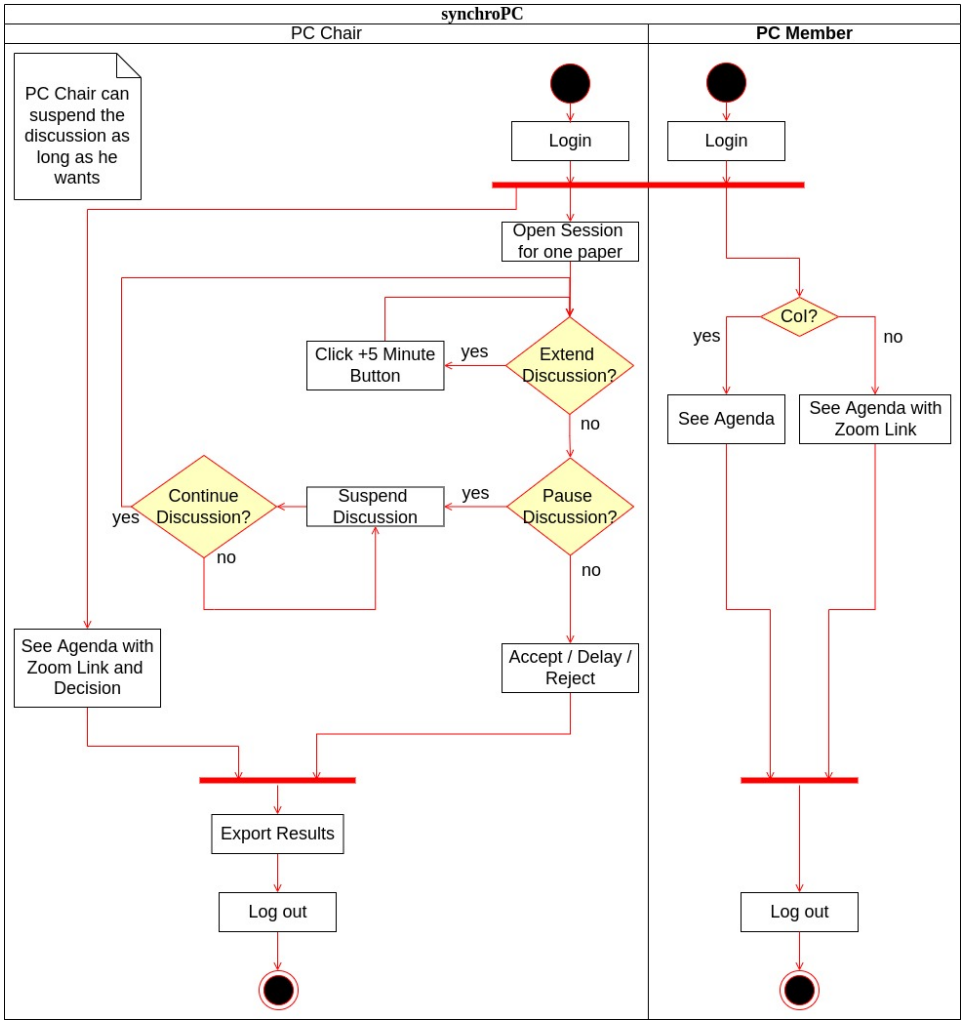

Figure 4.2: Sequence Diagrams - During and After Meeting

4.2 The Magic Algorithm

This section describes the implementation of an algorithm used to create an agenda and its problems and explains the improving approaches and methods.

4.2.1 How it works

Creating an agenda is the most critical feature of synchroPC. In order to create an agenda, every participant in the program committee's available time must be considered. In the case of synchroPC, the participants can be all around the world, which makes the differences in time between participants vary. The ultimate goal is to generate an agenda for reviewing

the scientific papers of choice so that most of the program committee can participate in a fixed range of time.
In order to archive that goal, the current synchroPC has offered an algorithm based on the Greedy Algorithm. The core idea of this algorithm is to sort and arrange the schedule for reviewing paper submissions according to the time zones of eastern-most reviewers and try to configure it later to handle the meeting time for western-most reviewers. Along with the main algorithm, synchroPC also provides an effective tool to edit the agenda and a scoring system for exceptional cases which are hard or impossible for the main algorithm to handle.
As inputs, the algorithm gets all the agenda configurations, such as the paper submissions that will be discussed, the date and time interval that the meeting can take place, the default discussion time, and the number of PC-Chairs for each submission. Before attempting to create an agenda, all of these data will be checked for validation. The data from paper submissions and reviewers can be imported from an excel file, which can be exported from the easyChair Website. Each paper must have at least one reviewer, and also there must be at least one paper submission for the algorithm to work with.
Once the inputs are checked, each selected submission's algorithm will be executed. First of all, the algorithm collects time zone information of all reviewers, and then it calculates the western-most time zone, eastern-most time zone, and the difference between them and the average time zone. Based on those data, the best-suited PC-Chair will be picked and added to the list of reviewers. If the difference in time between reviewers who have the western-most and eastern-most time zones is too big, the submission will be marked as "failed". It will still be added to the agenda, but it will not be prioritized as all reviewers cannot take part unless the range of time for discussion increases. Otherwise, the submission will be marked as an "okay" submission, sorted based on the minimum time zone of its reviewers, and ready for scheduling. After that, the algorithm tries to schedule all OK submissions on the first meeting day based on the agenda configuration.
For each "okay" a scoring system will be used to evaluate the quality of created agenda. This system has two types of scores: Local Score and Possible Score. The local Score is the Score of the current agenda item, and the Possible Score is the maximal Score that the agenda item can have. The higher the Local Score is, the more optimal is an agenda, but it should not exceed the Possible Score. The scoring system is based on the number of reviewers, which includes normal PC-Members and picked PC-Chairs, who have local time between the start and the end of the meeting day. There are three options for picking PC-Chairs: "None", "At least one" and "All Chairs". With the "None" and "All Chairs" options, no or all available PC-Chairs will be picked. For the "At least one" option, the PC-Chair for each submission will be chosen based on time zone: the algorithm will pick

the PC-Chair with the time zone as close as the average time zone of all reviewers. Each reviewer with an acceptable meeting time equals one unit in Local Score. A possible Score is a Score for the best case scenario, where all reviewers have sufficient meeting times.
After the local agenda is calculated, the algorithm will try to add a break before the currently scheduled submission to increase the Local Score of that submission. By adding pauses between discussed submissions, the discussions start time for the western-most reviewers will not be too early so that they can join the meeting in the configured time frame. Both Local Score and Possible Score will be recalculated to ensure that adding a break indeed improves Local Score. If the time slot for that paper after adding a break is acceptable, the agenda will be saved to the database, or else it will be scheduled later on other days. Day 3, or Overflow day, is the last day a submission can be scheduled. If there is insufficient time for all the OK submissions, the rest will be arranged for a second meeting day, using the same algorithm as the first day. After that, if there are still unscheduled submissions, they will be marked as failed and added to the database. All failed submissions will also be scheduled on day 3.

4.2.2 Problems in the algorithm

The algorithm is working, but many logic and implementation issues need attention and improvements. Also, the algorithm does not handle well in some complicated cases. All of these problems are summarized as follows.
First of all, critical problems should be detected and resolved. There are four main problems, which are:

- *The function for evaluating new agenda score is not correct*
  As explained in the previous section, the scores are recalculated every time the algorithm adds a break before an agenda item. The new Local Score must be compared with the old one to guarantee that the new start time of the agenda item is better than before. However, the current compare function only compares the old local score with 0, which is not logically strict enough in this case. The code block below shows how this function is implemented before improvements.

```
1 def is_schedule_acceptable(agenda_score_info):
2     acceptable = 0
3     for reviewer in agenda_score_info.member_times:
4         if reviewer.score > 0:
5             acceptable = acceptable + 1
6     return acceptable > 0
```

Listing 4.1: The agenda score compare function

- *The Push-back strategy is not a good solution to optimize the scores of an agenda*
  In order to improve the Local Score of one agenda item, the algorithm adds a broken item before it without considering the Local Score of the other items behind it. Furthermore, the algorithm sorts all submissions in a way, so submissions with a smaller minimum time zone of its reviewers will be scheduled first. The submissions are sorted without considering the distance between the maximum and minimum time zone, which is an essential criterion for scheduling a submission.
  This approach leads to two problems. First of all, the Local Score for items at the beginning of the agenda looks "perfect" and the Local Score of the items is much worse at the end. Secondly, the time of break item before one agenda item can be so big that it can fit one or more other agenda items.
  Figure 4.3 shows an example of an agenda with a long break. To make all reviewers for paper 18 able to join the discussion, the algorithm delayed the start time for this discussion by 120 minutes.

```
def apply_push_back_strategy(agenda_score_info, day_start,
        time_slot, agenda_item, is_first):
    return_items = []

    max_difference = timedelta(minutes=0)
    for reviewer in agenda_score_info[1]:
        difference = day_start.replace(tzinfo=None)
            - reviewer.local_time.replace(tzinfo=None)
        max_difference = max(difference, max_difference)

    if max_difference > timedelta(minutes=0):
        if not is_first:
            break_item = Agenda(
                submission=None,
                timestart=time_slot,
                duration=max_difference
            )
            return_items.append(break_item)
        moved_item = Agenda.copy(agenda_item)
        moved_item.timestart = time_slot + max_difference
        return_items.append(moved_item)
    else:
        return_items.append(agenda_item)

    return return_items
```

Listing 4.2: The Push-back Strategy

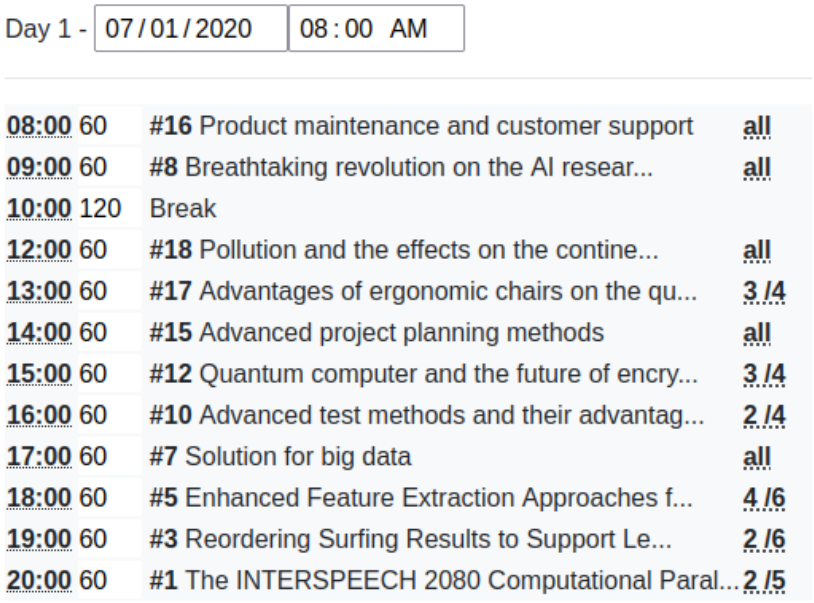

Figure 4.3: An agenda that has an item with a 120-Minutes-Break.

- *The problem with PC-Chairs and their time zones*
  In the algorithm, the number of PC-Chairs for each submission can be zero, one or all PC-Chairs. In the case of one PC-Chair, the algorithm will pick the best PC-Chair, whose time zone has the least distance from the average time zones of all reviewers. It guarantees that the picked PC-Chair is closed to other members of the program committee and hence, increases the agenda scores overall. However, two Chairs will be picked in the case of all PC-Chairs, which makes the Local Score look worse, especially if the Chairs are far from each other.

- *Agenda items with invalid discussion time or start time*
  If the start and end time of the meeting is too close to each other, and there is not enough time to discuss every paper, the algorithm should not create any agenda and suggests that the PC-Chair should increase the discussion time for the agenda. Before improvements, the algorithm can generate an agenda with zero minute discussion time, and more than one agenda items have the same start timem, which should not happen. In this case, some of the submissions should be pushed to the other days, Day 2 and Overflow Day, and the discussion time should remain non-zero.
  The problem with invalid time leads to another problem, that agenda score is not always accurate. The reason is, that the Local Score from submissions with zero minute discussion time should not be included

at all as they are invalid
Figure 4.4 shows what agenda items with invalid start times look like. The algorithm schedules for discussion at night time, even when PC-Chair already specified the discussion end time of the day.
Figure 4.5 shows an example of agenda items with invalid discussion time. Four agenda items have the same start time, at 10 o'clock, and the discussion time is zero.

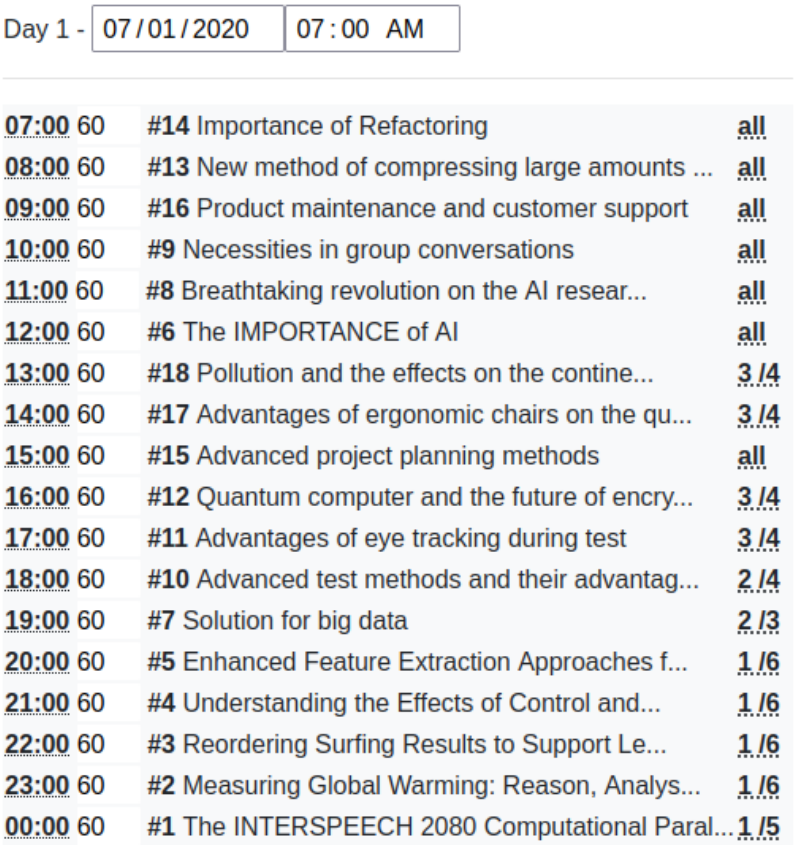

Figure 4.4: An agenda with night time items

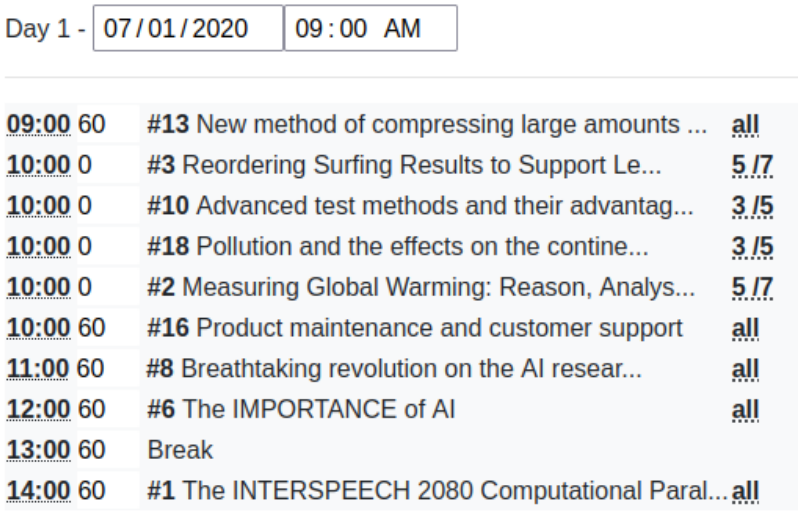

Figure 4.5: An agenda with 0-Minute-Discussion-Time items

Beside the major problems, there are also some smaller problems that are relevant to the Algorithm:

- *Problems of the Unit Tests*
  Some of the old Unit Tests for the Magic Algorithm are not working. Moreover, the new Magic Algorithm should also be tested to guarantee that it works and, in most cases, is better than the old one. For that reason, more Unit Tests for Magic Algorithm should be created.

- *The scoring function uses the scores from invalid agenda items*
  Every agenda item that lies outside the start and end time or has zero discussion time should not be considered, and their score should not be used to calculate the overall agenda scores. This problem can be improved by adding more conditions to check for the calculate score function.

Improvements

The nature of the Magic Algorithm is Greedy Algorithm: it builds the agenda by adding submissions one by one and guarantees that the time slot for each added submission is optimal. With this approach, the time complexity is typically tiny, and it is also easy to implement. However, this approach can only find the optimal local solution, which is not always the optimal global one. On the other hand, finding an optimal schedule will cost a lot of time and memory, which is not a good idea when the number of submissions is considerable. For this reason, a better approach to improve the Magic Algorithm is to optimize the old Magic Algorithm to reach a

better local optimal solution that is as close to the optimal global one as possible. Instead of using a new algorithm, the Magic Algorithm is kept "greedy" with some modifications. That way, the new Magic Algorithm is still fast and creates better agendas, which are as close as possible to the optimal agendas. The improvements in the new Magic Algorithm are listed as follows:

- **Remove the Push-back Strategy**
  Push-Back Strategy can only guarantee to improve the Local Score of one submission. As explained above, it does not care about the submissions below the one it pushes back, so it will not know if it makes the total sum of all Local Score looks worse or better. This strategy is unreliable enough to be used and will be replaced with another approach. The new Algorithm will instead divide the time interval between start and end time into time slots. Each time slot has a duration equal to the default duration that has been set in the Agenda's Configuration. It will then try to schedule discussions into these time slots.

- **Change the method for choosing submission to schedule**
  As explained above, the order of okay submissions will decide the order of the whole agenda. It is why ordering is critical, and ordering submissions using only the minimum time zone of its reviewers is not enough. The difference between its reviewers' minimum and maximum time zone also plays a vital role, as it decides how many time slots the submission can be placed on the agenda. The more significant the difference is, the fewer time slots that can satisfy all reviewers.
  Before choosing the next submission to be scheduled, the new algorithm will sort all submissions into two dictionaries for Okay and Bad Submissions, which are for submissions that are possible or impossible for every reviewer to discuss simultaneously. For both dictionaries, the key is the minimum time zone, and the value is a list of tuple from the submission instance and the difference between the minimum and maximum time zone of a submissions reviewers.
  After creating the dictionaries for Okay and Bad Submissions, the Algorithm will try to pick one submission using the following strategy: First, it will try to pick a submission from the Okay Submissions dictionary, in which every reviewer can join the discussion. If no submission can be picked from the Okay Submissions, the Algorithm will search for a submission from the Bad Submissions dictionary, which has the best Local Score in the current time slot. If that is still not possible, an okay submission with the best Local Score will be picked. Then, if the Algorithm cannot still pick a submission, a

submission at the top of the dictionary will be picked. With this strategy, it is guaranteed that an okay submission will always find the best time slot, and the bad submissions will still have a good score.
So how can a submission be picked from a dictionary? At first, from the dictionaries, a group of submissions with a suitable minimum time zone value and maximum difference between the maximum and minimum time zone will be collected. In order to choose the key and values from dictionaries, the new algorithm uses a variable called "index" as a threshold for the minimum time zone. The algorithm only takes submissions with the minimum time zone at least equal to the index if the time slot is sooner than the middle time point of the working day and at most equal to the index otherwise. This method decreases the number of submissions that has to be considered for each time slot. Among the picked submissions, a submission with the highest agenda score for the time slot will be chosen.
After each scheduling step, the time slot will be updated, and the choosing algorithm will run as long as at least one submission is still not on the agenda. If submission for a time slot cannot be chosen from the dictionary of Ok Submissions, it will choose from the Bad Submissions. In the end, both dictionaries should contain only empty list values.
Listing 4.3 shows the Python function, which picks a submission from a dictionary.

```
def choose_next_submission(submissions, index, duration,
        time_slot, day_start, day_end,
        is_midpoint, fit_bypass):
    tz_keys = None
    if not fit_bypass:
        if not is_midpoint:
            tz_keys = [i for i in submissions if
                i >= index and submissions[i]]
        else:
            tz_keys = [i for i in submissions if
                [tz_max for _, tz_max in submissions[i] if
                 tz_max <= index]]
    if not tz_keys:
        tz_keys = [i for i in submissions if
            submissions[i]]
    tz_keys.sort()
    best_choice = None
    best_tz_min = None
    best_tz_dif = -30
    best_point = 0
    for tz_min in tz_keys:
        for submission, tz_max in submissions[tz_min]:
            agenda_item = Agenda(submission, time_slot,
                                 duration)
            points = points_for_agenda(agenda_item)
```

```
26
27              if check_if_submission_fit_timeslot(time_slot,
28                   day_start, day_end, duration, tz_min,
29                   tz_max, fit_bypass):
30                  point = points[0] / points[3]
31                  if point > best_point or (
32                       point == best_point and
33                       tz_max - tz_min >= best_tz_dif):
34                      best_choice = (submission, tz_max)
35                      best_point = point
36                      best_tz_min = tz_min
37                      best_tz_dif = tz_max - tz_min
38      if best_choice:
39          if (best_point > 0 and fit_bypass) or
40              (best_point == 1 and not fit_bypass):
41              submissions[best_tz_min].remove(best_choice)
42              return best_choice[0]
43      return None
```

Listing 4.3: Function for choosing submission for the next time slot

- **Solution for PC-Chairs and their time zones**
  A simple solution for the instability in case there is more than one PC-Chair is that only one PC-Chair should be picked in case they are too far away. The PC-Chair that is closer to other program committee members should be picked. The code snippet for this improvement is shown below.
  Listing 4.4 shows the code for picking chairs in case of the All Chair option.

```
1  # Pick chairs based on time zones
2  def pick_chairs(tz_min, tz_max, tz_avg, chairs):
3      ...
4      if at_least_chairs == AtLeast.ALL:
5          chairs_picked = []
6          least_bad_chair = None
7          least_bad_chair_diff = sys.maxsize
8          for chair in chairs:
9              chair_tz = get_member_timezone(chair)[0]
10                                  .total_seconds()
11             avg_diff = abs(tz_avg - chair_tz)
12
13             if chair_tz < tz_min or chair_tz > tz_max:
14                 if avg_diff < least_bad_chair_diff:
15                     least_bad_chair_diff = avg_diff
16                     least_bad_chair = chair
17             else:
18                 chairs_picked.append(chair)
19     ...
```

Listing 4.4: The code that performs picking chairs for “All Chairs” option

- **Solution for agenda items with invalid discussion time or start time**
  The start time on Overflow Day is adjusted to begin at the correct start time.
  In order to avoid agenda items with invalid start time or duration, the new algorithm will calculate the next time slot every time a submission is scheduled. If the next time slot exceeds the end of the current working day, it will be reassigned so that the next submission will begin on the next working day. For Overflow Day, the time slot can exceed the discussions end time as it is already the last day to be scheduled, even when the time slot is invalid to the agendas configuration. However, the scores of all submissions in this time slot will not be calculated.
  Listing 4.5 shows the code snippet used to calculate the next free time slot.

```
def schedule_submissions_alter(okay_submissions,
        bad_submissions, day1_start, day1_end, day2_start,
        day2_end, duration, spill_logging=True):
    ...
    # Schedule all the submissions
    submission = True
    while submission:
        ...
        time_slot = agenda_item.timestart +
                    agenda_item.duration
        if time_slot + duration > day_end:
            if day_end == day1_end:
                day_start = day2_start
                day_end = day2_end
                time_slot = day_start
        elif day_end == day2_end:
            day_start = datetime.combine(
                            AgendaConfig.get_day_n(2),
                            day1_start.time()
                        )
            day_end = datetime.combine(
                            AgendaConfig.get_day_n(2),
                            day1_end.time()
                        )
            time_slot = day_start
            ...
    ...
```

Listing 4.5: Code snippet for recalculate the time slot for next submission

Figure 4.3 shows an example of an agenda created using the new Magic Algorithm.

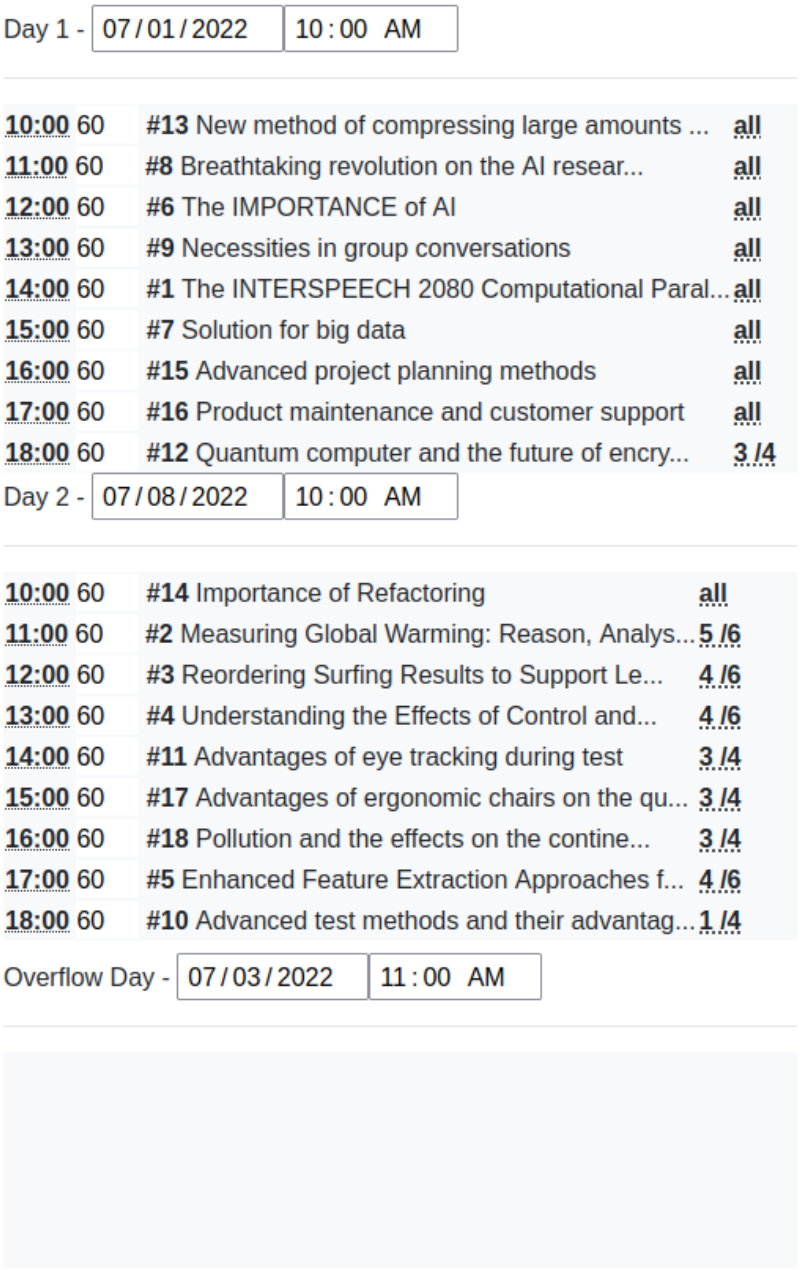

Figure 4.6: A new Magic Algorithm's Agenda

The evaluation of the result from the new Algorithm will be discussed in Evaluation Chapter.

4.3 New Features and Changes

Besides the Magic Algorithm, some tickets in the synchroPC's Trello-Board were implemented during the project. New Back-end and Front-end features will be described in this section.

4.3.1 Reschedule suspended papers

Figures 1 and 2 show how the reschedule function works on the Personal View page. During a session in SynchroPC, a PC-Chair user can suspend a discussion, and the suspended paper will be added to the list on the right side of the page. However, the suspended discussion is disappeared from the agenda and cannot be normally resumed later by the PC-Chairs. In order to solve this, a button is added nearby with the title of the suspended paper on the right side list, indicating that PC-Chair can resume the discussion of this paper by clicking the button. As a result, the discussion for that paper will be added back to the top of the agenda. The timestamp for other discussions will be pushed back to keep the correctness in time of the agenda.

4.3.2 Hide and Show Agenda

Figure 3 describes what the Personal View page looks like for a PC-Chair and a normal member of the Program Committee when a PC-Chair hides or shows an agenda. When a PC-Chair initializes an agenda, it should not be public for other reviewers because presumably, the PC-Chairs must adjust the agenda, for example, by changing the order of the paper discussions or adding breaks between them. Until PC-Chairs are finished with the editing, the agenda should remain private and only visible to PC-Chairs. During the discussion, a PC-Chair can also hide or show the agenda when he wants to edit the current agenda.

4.3.3 Undo agenda editing

Figure 4 shows the position of Undo agenda button on the Edit Agenda page. Using the Undo button, a PC-Chair can undo his editing when he makes a mistake. Each time the PC-Chair clicks the Undo button, the agenda is reversed to its previous state.

4.3.4 Change the decision for a paper

Figure 5 shows how a PC-Chair can change the decision of a paper. On the Roles for Paper page, the PC-Chair can choose a paper and then change its decision using a drop-down list. To confirm the decision, the PC-Chair can click on the Save button, which will be saved to the database.

4.3.5 Add filter to reject weak account's password

It is a problem that a user can use a weak password for his or her account. A password filter guarantees that a password must have a minimum of eight characters and will be rejected if it is a common password.

4.3.6 New Unit Test cases for the Magic Algorithm

With a long development time, the unit tests have been forgotten to be adjusted and could not be run anymore. In order to develop new unit test cases for the Magic Algorithms, all the unit test cases for it have been renewed and can be run normally. The new unit test cases compare the quality of agendas and the runtime from both Algorithms. When an agenda from the new Magic Algorithm is worse than an agenda from the old one, the results will be logged in to a .json file under the folder "db".

4.3.7 Minor changes

Some minor changes have been made to improve the user experience for SynchroPC:

- PC-Chair will now see the names of PC-Members instead of their E-Mails when he initials a new Agenda
- In the Personal View page, the CoI symbols have been made bigger to improve readability.

Chapter 5

Evaluation

This chapter elaborates on the evaluation process to determine if the improvements of synchroPC fulfill the goals of the thesis.

5.1 The Evaluation of Magic Algorithm

How can the new Magic Algorithm be evaluated? The result of new and old Algorithms should be calculated and compared to determine which is better.

5.1.1 Test Planning and Design

According to Schneider [9], the following aspects must be defined for each test case:

Set up

Before the tests can be run, a context must be defined. At this point, all the data of Submissions with its information and CoI assignments, all the Members, their countries, time zones and roles in the Program Committee must be available in the database.
In this project, the submissions and members data will be imported from an Excel file named "Faked Conf Data-v03.xlsx" under the folder "sample-input" from SynchroPC's repository. Also, under the folder "src/DB", there is an instruction to import the countries and time zones information from the SQL Database.
In total, 18 Papers and 12 EasyChair members will be imported.

Input for the tests

As inputs for both old and new Algorithms, a list of submissions, the "from" and "to" dates, start and end time, the default duration for each discussion

and the PC-Chairs option must be provided.
The values of each input field that are used for testing are listed below:

- **The number of PC-Chairs**
  This input determines how PC-Chairs will be picked for each Submission. The number of Chairs can be None, One or All (equal to Two).

- **The number of Submissions that will be scheduled**
  This input decides how the Submissions will be scheduled for the day's start and end time in Agenda Configuration, which affects the time and memory usage of the Algorithm.
  For testing, both algorithms must use the same set of submissions. The number of papers will be used for testing are 8, 12 and 18. Which number of papers are chosen will be declared specifically for each test case. If the number of papers is n, the set of submissions used for the tests will be all submissions with the id number between 1 and n.

- **The discussion time for each day and each agenda item**
  This input affects the number of papers that can be fit in a day. For the tests, the days "07/01/2022" and "07/08/2022" will be used as input for meeting days. Two intervals will be used as start and end times for a new Agenda. The first interval is from 9 to 17 o'clock and the second interval is from 8 to 14 o'clock.
  For each agenda item in the tests, the duration is 45 minutes.

In addition to the inputs, there is another important factor. With the All Chairs option, it must also be considered if the Chairs are geographically close or far away from each other. Two PC-Chairs from imported data will be used for testing: Brianna Knapp and Barbara Bently.

- In the case of two PC-Chairs are close to each other, Brianna Knapp will have the value "Germany" and Barbara Bently will have the value "Ukraine" for the Country attribute.

- In the case of two PC-Chairs far from each other, Brianna Knapp will have the value "Brazil", and Barbara Bently will have the value "China" for the Country attribute.

For None and One Chair option, this factor is redundant; therefore, there will be no test for these particular cases. However, to perform another test, the PC-Chairs setting must be equal for both Algorithms. For other tests, the countries of the two PC-Chairs will be set like in the case of two PC-Chairs are close to each other, which means Brianna Knapp will have the value "Germany" and Barbara Bently will have the value "Ukraine" for the Country attribute.

Output of the tests

As output, an Agenda should be created for each Algorithm, which contains the working schedule for each Submission. The users should see for each Submission the discussion duration, index number, name and the quotient between the number of Reviewers that can take part in the discussion and the total Reviewer amount of each Submission. Then, the quality of the new Magic Algorithm can be evaluated, using the result from the old Algorithm as a threshold.

Evaluation Criterion

The criterion to evaluate the output for both Algorithms are defined in the following order:

- The number of agenda items in both old and new Agendas, which has invalid discussion time.
- The number of agenda items in both old and new Agendas, in which all Reviewers can join.
- The quotient between the sum of Local Score and the sum of Possible Score of the submissions, which are impossible to be discussed by all reviewers.

Number of test cases

There are:

- Three available input values for the number of PC-Chairs
- Three available input values for the number of Submissions that will be used
- Two intervals which is used to initial the start and end time in the Agenda's Configuration
- Two additional test cases for the All Chairs option, when the Chairs are far from each other

In total, there must be at least $3 * 3 * 2 + 2 = 20$ test cases to compare the old and new Magic Algorithms.

5.1.2 Result and Analysis

The complete setup for each test case and each Algorithm's result can be seen in the test result's folder. This section attempts to describe and explain the results to determine the quality of the new Algorithm compared to the old one.

The number of agenda items which have invalid discussion time

Figure A.1 presents the number of invalid agenda items of both algorithms for each test case. In this criteria, the new Algorithm has made better agendas: there is no invalid agenda item in every test output. On the contrary, every Agenda which was created by the old Algorithm has invalid agenda items. The old Algorithm has not checked for the validation of each agenda item, which makes it possible to have invalid entries in the Agenda. This result indicates that the problem from the old Algorithm has been solved in the new one.

The number of agenda items in which every PC-Members can participate in

Figure A.2 presents the number of agenda items in which every PC-Members can join for each test from both algorithms. In case there are eight papers to discuss, the agendas from the old Algorithm seem to have an equal or a slightly more significant amount of perfect agenda items compared to the new one. However, if there are twelve or eighteen papers to discuss, the number of perfect agenda items from the new Algorithm is equal to or more than from the old Algorithm's Agenda. Especially in all test cases with the All Chair option, the agendas from the new Algorithm are always better in this criteria.
What can be seen in these results is that the new Algorithm can deal with a medium or significant amount of papers and handle the case of more than one PC-Chairs better than the old Algorithm. The old Algorithm always tries to maximize the Local Score for an agenda item when it is possible to do so. In order to maximize the Local Score for a small number of agenda items, the other agenda items are invalid in time, or sometimes, there are long breaks between agenda items. On the other hand, the new Algorithm will add no break to the Agenda. It tries to maximize the Local Score of the Agenda by dividing the total time for discussion in a day into time slots and finding the best Submission that suits each time slot. Although the number of perfect agenda items may be less in a few cases, every agenda item from the new Algorithm's agendas is guaranteed to be valid.

The quotient between the sum of Local Score and the sum of Possible Score of agenda items, which are impossible to be discussed by all reviewers

Figure A.3 shows the quotient for each test from both algorithms. In fourteen test cases, the quotient from the old Algorithm's Agenda is equal to or more significant than the one from the new Algorithm's Agenda. In most test cases, the agenda items, which are impossible for all reviewers to join, usually have invalid discussion time in the old Algorithm's Agenda.

They are also closed to each other. The invalid discussion time of an item makes the start time from the following agenda items invalid. As a result, the discussion of an item has no effect or is insignificant to other discussions.

5.2 Discussion

This section describes the problems encountered during the further development of SynchroPC and its current limitations.

5.2.1 The lack of comments in code, PEP-8's Usage and Python code linters

Not many comments had been written in the file of Magic Algorithm, which is one of the reasons why it is hard to understand and explain in the first place. The Magic Algorithm is a complex one with many steps, and it needs explicit, detailed comments in the code file so that other developers can follow and understand it in the future. Moreover, the Style Guide for Python Code, PEP-8, had not been strictly followed. In future works, developers should be aware of these problems and avoid them, not only for the Magic Algorithm but also for other modules.

5.2.2 Run time of the new Magic Algorithm

Table 5.1 shows how long the new Magic Algorithm will run with different setup and input. Compared to the old Algorithm, the new one is about eight times slower in the case of 50 papers. In order to find optimal Submission for each time slot, the Algorithm has to loop multiple times through all of the submissions in the worst case. However, the run time from the new Magic Algorithm should be acceptable, as the number of papers that have to be scheduled hardly reaches the amount of 50 papers.

Setup	**Old Magic Algorithm's run time (sec)**	**New Magic Algorithm's run time (sec)**
5 Submissions, 10 Reviewers	0.12	0.20
14 Submissions, 25 Reviewers	0.22	1.22
25 Submissions, 50 Reviewers	0.51	2.92
50 Submissions, 100 Reviewers	1.26	8.42

Table 5.1: The runtime in second of both algorithms, in case the discussion time is from 8:00 to 18:00, the PC-Chair option is "one" and the PC-Chairs are in the same place (Germany)

5.2.3 Magic Algorithm Optimization

Overall, the new Algorithm seems to be better than the old one. However, the generated Agenda using the new Algorithm is not guaranteed to be the most optimal Agenda. Other improvement ideas for the Algorithm had not been chosen due to the lack of research time and the Algorithm's complexity.

5.2.4 Further Improvements in Front-end

During this project, a few improvements have been made to the Front-end, but not significantly improved. Most of the web pages in SynchroPC are not responsive, and the layouts for those pages are still not optimal.

Chapter 6

Conclusion

This chapter summarizes the most important topics that were discussed in the thesis. Moreover, it provides an outlook on how synchroPC could be improved in future works

6.1 Summary

SynchroPC is a web application, which is for the coordination and exchange of program committee meetings of scientific conferences. However, the agenda generated for the program committee was not optimal. There are some new Security and Usability requirements from the customers that need to be added to the web application. In the course of this thesis, synchroPC was further developed, following the Principle of Software Engineering.
The main goal of the thesis is to improve the generated agenda's quality, implement new front-end functionalities for account activation and administration, and new features using collected requirements. Django was continually used as the main framework for the server-side of the application. Along with Django, the Boostrap framework and jQuery library has been used to handle the client-side of the application.
Using Requirement Engineering, new non-functional requirements for Security and Usability have been collected and implemented. The Magic Algorithm used to create agendas for the program committee has been further improved and tested. Some bugs in Front-end have been identified and fixed using the tickets in Trello Board.
Overall, the results obtained from the tests for Magic Algorithm indicate that the new Algorithm can generate better agendas within a reasonable amount of time. The output from the Algorithm is not guaranteed to be optimal, but it is close to the optimal agenda.
With the new improvements, synchroPC can continually be used more efficiently and is easier to be maintained and further developed in related future works.

6.2 Outlook

There are still issues on the Front-end side of the web application that future developers should look into. Moreover, some alternative ideas could be used to develop a new Magic Algorithm, such as a Genetic Algorithm, or formulate the current Scheduling problem to turn it into a Constraint Satisfaction Problem and then apply the Backtracking Algorithm to solve it. Whether they can produce a better result than the new developed Magic Algorithm must be answered using further research and experiments.

Appendix A

Evaluation Data

A.1 Test setup and input

Test No.	Brianna's country	Barbara's country	#Chairs	From	To	Start time	End time	#Submissions
1	Germany	Ukraine	0	07/01/2022	07/08/2022	9:00	17:00	8
2	Germany	Ukraine	0	07/01/2022	07/08/2022	9:00	17:00	12
3	Germany	Ukraine	0	07/01/2022	07/08/2022	9:00	17:00	18
4	Germany	Ukraine	0	07/01/2022	07/08/2022	8:00	14:00	8
5	Germany	Ukraine	0	07/01/2022	07/08/2022	8:00	14:00	12
6	Germany	Ukraine	0	07/01/2022	07/08/2022	8:00	14:00	18
7	Germany	Ukraine	1	07/01/2022	07/08/2022	9:00	17:00	8
8	Germany	Ukraine	1	07/01/2022	07/08/2022	9:00	17:00	12
9	Germany	Ukraine	1	07/01/2022	07/08/2022	9:00	17:00	18
10	Germany	Ukraine	1	07/01/2022	07/08/2022	8:00	14:00	8
11	Germany	Ukraine	1	07/01/2022	07/08/2022	8:00	14:00	12
12	Germany	Ukraine	1	07/01/2022	07/08/2022	8:00	14:00	18
13	Germany	Ukraine	2	07/01/2022	07/08/2022	9:00	17:00	8
14	Germany	Ukraine	2	07/01/2022	07/08/2022	9:00	17:00	12
15	Germany	Ukraine	2	07/01/2022	07/08/2022	9:00	17:00	18
16	Germany	Ukraine	2	07/01/2022	07/08/2022	8:00	14:00	8
17	Germany	Ukraine	2	07/01/2022	07/08/2022	8:00	14:00	12
18	Germany	Ukraine	2	07/01/2022	07/08/2022	8:00	14:00	18
19	Brazil	China	2	07/01/2022	07/08/2022	9:00	17:00	18
20	Brazil	China	2	07/01/2022	07/08/2022	8:00	14:00	18

Table A.1: Setup and input for each test case. The number of submissions also indicates which submissions are used for the test. If the number of submissions is n, all the submissions that have the id number from 1 to n will be used for the test.

A.2 Agenda items with invalid duration or start time

Test No.	Old Magic Algorithm	New Magic Algorithm
1	4	0
2	7	0
3	9	0
4	5	0
5	8	0
6	11	0
7	4	0
8	7	0
9	9	0
10	5	0
11	8	0
12	11	0
13	4	0
14	7	0
15	9	0
16	5	0
17	8	0
18	11	0
19	12	0
20	14	0

Table A.2: Number of discussions, which has invalid discussion time or duration

A.3 Agenda items, in which all reviewers can join

Test No.	Old Magic Algorithm	New Magic Algorithm
1	4	4
2	5	5
3	9	9
4	3	2
5	3	3
6	5	5
7	4	4
8	5	5
9	9	9
10	3	2
11	3	3
12	5	5
13	4	2
14	5	5
15	8	9
16	2	2
17	2	3
18	4	5
19	0	6
20	0	1

Table A.3: Number of discussions, in which all reviewers can join

A.4 Agenda items, in which all reviewers can join

Test No.	Old Magic Algorithm	New Magic Algorithm
1	11/19	10/19
2	15/28	15/28
3	19/34	20/34
4	10/22	10/22
5	12/31	15/31
6	14/42	20/42
7	16/24	14/24
8	23/36	22/36
9	29/44	27/44
10	17/29	16/29
11	25/45	26/45
12	27/57	31/57
13	20/28	18/28
14	30/43	27/43
15	38/53	35/53
16	27/40	21/39
17	35/55	32/55
18	43/73	38/73
19	32/69	36/69
20	47/89	40/89

Table A.4: Quotient between the sum of Local Score and the sum of Possible Score of the submissions, which are impossible to be discussed by all reviewers

Appendix B

Contents of the CD

A CD is attached to this bachelor thesis, which consists of the following content:

- The bachelor thesis in digital form (.pdf)
- The most recent version of the source code (as of 22.08.2022)
- Evaluation data. There are 20 sub-folders, each has the name in form "test<test-no>_<chair-option>_<start-time>-<end-time>_<number-of-submissions>"
  - "test-no": The index of the test. It can have the value from 01 to 20
  - "chair-option": The value of chair option in Agenda's Configuration. It can be "none", "one", "all-close" and "all-far".
  - "start-time", "end-time": The day's start and end time of the agenda in Agenda's Configuration.
  - "number-of-submissions": The number of submissions that have been used for the tests. It can be 8, 12 or 18.

Bibliography

[1] K. S. McKinley, *More on improving reviewing quality with double-blind reviewing, external review committees, author response, and in person program committee meetings*, Jun. 2015. [Online]. Available: https://www.microsoft.com/en-us/research/publication/more-on-improving-reviewing-quality-with-double-blind-reviewing-external-review-committees-author-response-and-in-person-program-committee-meetings/.

[2] K. Schneider, *Lecture 1: Einführung*, Grundlagen der Software-Technik, 2019.

[3] I. Sommerville, *Software engineering* (Always learning). Pearson, 2015, ISBN: 9781292096131. [Online]. Available: https://books.google.de/books?id=Cg5irgEACAAJ.

[4] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, *et al.*, "Manifesto for agile software development," 2001. [Online]. Available: http://agilemanifesto.org/.

[5] K. BECK, *Extreme programming. nummer isbn-13: 978-3827321398*, 2003.

[6] A. Metzner, *Software Engineering - kompakt, Hanser eLibrary*. München: Hanser; 2020. DOI: 10.3139/9783446463653.

[7] J. K. Ousterhout, *A philosophy of software design*. Yaknyam Press Palo Alto, 2018, vol. 98.

[8] *Iso 25010*, https://iso25000.com/index.php/en/iso-25000-standards/iso-25010.

[9] K. Schneider, *Abenteuer Softwarequalität : Grundlagen und Verfahren für Qualitätssicherung und Qualitätsmanagement, Abenteuer Software-Qualität, Das Q-Buch*. Heidelberg: dpunkt-Verl.; 2012, Abenteuer Software-Qualität, Das Q-Buch, ISBN: 9783898647847. [Online]. Available: https://www.tib.eu/de/suchen/id/TIBKAT%3A671822039.

[10] A. Spillner and T. Linz, *Basiswissen Softwaretest : Aus- und Weiterbildung zum Certified Tester ; Foundation Level nach ISTQB-Standard*. Heidelberg: dpunkt-Verl.; 2005, ISBN: 3898643581. [Online]. Available: https://www.tib.eu/de/suchen/id/TIBKAT%3A489520014.