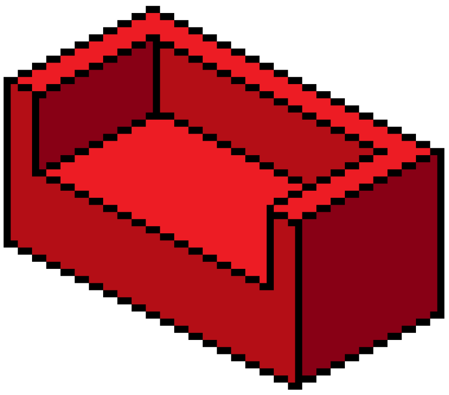
CHRISTIAN CALDERA

COMFY

COMFY

CHRISTIAN CALDERA



A Conference Management Framework

April 2013 – version 1.0

ABSTRACT

Scientific work has to be documented and is usually submitted to a conference to be published as a paper. Prior to publishing, the paper has to be reviewed, which is a cumbersome task for the reviewers as well as for the chair. The chair has to assign the reviewers to the papers which they read and rate. Computer based programs - so called conference management systems - are used to help reviewers and chairs managing the submissions. These systems maintain the submissions, enforce access rights and guarantee a certain amount of anonymity between reviewer and author. This thesis presents a different approach to such systems. The use of an API instead of a closed system makes it much easier for external programs to harvest and process the data within the framework. Because of this API it is also easy to customize the system to the users needs. An example implementation of this new approach shows the advantages proving that this new approach is feasible.

ZUSAMMENFASSUNG

Wissenschaftliches Arbeiten wird dokumentiert und bei Konferenzen als sogenanntes Paper veröffentlicht. Bevor so ein Paper publiziert werden kann, muss es noch von Gutachtern rezensiert werden. Diese Aufgabe ist sowohl für die Gutachter als auch den Vorsitzenden der Konferenz eine mühselige Arbeit. Der Vorsitzende muss die Rezensenten zuordnen. Diese müssen wiederum das Paper lesen und eine Rezension schreiben. Computerprogramme, sogenannte Konferenzmanagementsysteme, werden benutzt um den Vorsitzenden und den Gutachtern zu helfen die Einreichungen zu verwalten. Diese Programme verwalten die Paper, gewährleisten Zugriffsrechte und garantieren einen gewissen Grad der Anonymität zwischen Gutachtern und Autoren. Diese Arbeit präsentiert eine neue Herangehensweise an so ein System. Um die Kommunikation mit dem Framework einfacher zu machen wird eine API angeboten anstatt eines geschlossenen Systems. Diese API erleichtert das Anbinden externer Programme um Daten zu erfassen und sie zu verarbeiten. Der User kann aufgrund dieser API das System auch individuell auf seine Bedürfnisse anpassen. Eine Beispielimplementierung zeigt die Vorteile und Realisierbarkeit des neuen Systems.

PUBLICATIONS

"COMFy - A Conference Management Framework" was accepted as a paper on the 17th International Conference on Electronic Publishing the Digital Information Networks (ELPUB 2013[1]). [2]

1 http://www.bth.se/com/elpub2013.nsf

ACKNOWLEDGMENTS

This project would not have been possible without the support of many people. I would like to thank my supervisor René Berndt who supported and guided me throughout the project. Further I like to thank all members of the Institute of Computer Graphics and Knowledge Visualization as well as the Fraunhofer Austria Research GmbH, especially Christoph Schinko, Harald Grabner, Martin Schröttner and Stefanie Behnke. I would also like to express my highest gratitude to my family, in particular my parents and grandparents for supporting me throughout my study. Last I want to thank all my friends for helping and supporting me throughout the study, especially Andrea Jäger and Melanie Fuchs for mental support through the study and the project.

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am …………………………… ……………………………………………….. (Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…………………………… date ……………………………………………….. (signature)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

1 INTRODUCTION

1.1 OVERVIEW

In the scientific world it is common to present the research of one or multiple persons as a written scientific work. Authors publish their work in journals, books or conferences. Before such an academic work is published in such a media it has to be reviewed as a so called paper. The acceptance of a paper is based on reviews written by experts in the field the paper is written. This process is called peer reviewing.

The peer reviewing process originated from the 18 century. In 1731 the Royal Society of Edinburgh published the peer-reviewed *Medical Essays and Observations*. These essays were given to people who were well versed in their field. The purpose behind this idea was not like today to check if the paper contains scientific progress but to spread the work and ideas. From this time the peer-reviewing process approached step-by-step to our current format. [1]

With the introduction of the computer and the internet the reviewing process developed to a new format. The computer now manages submissions and users under the supervision of the chair. Tedious work like assigning reviewers to hundreds of papers can also be taken by the computer. Reviewers and authors can access now the conference at anywhere at any given time.

The Publishing Research Consortium[1] analysed four of the most common peer reviewing types[23]. These types are:

Single-blind peer review: In this reviewing process the authors don't know who is reviewing their paper. The reviewer on the other hand knows the identity of the author. This practice is done to grant the reviewer protection to give a honest review about the paper. This process is currently the most common in the different journals (see Figure 1).

Double-blind peer review: If the reviewer also don't know about the identity of the author the review process is called Double-blind peer review. Journal editors and authors think this is the most effective and also their preferred choice (see Figure 2).

Open peer review: If the reviewer and the author knows each others identity the process is called Open peer reviewing.

1 The Publishing Research Consortium is a group of associations and publishers, which supports global research into scholarly communication in order to enable evidence-based discussion. - http://www.publishingresearch.net

Post-publication review: These reviews are submitted after the publication of a paper. These papers were reviewed with one of the previous mentioned formats. The Post-publication reviews are considered in Figure 2 as an additional assistance process rather than a reviewing process on its own. They add further information or critiques to such a paper. [23]

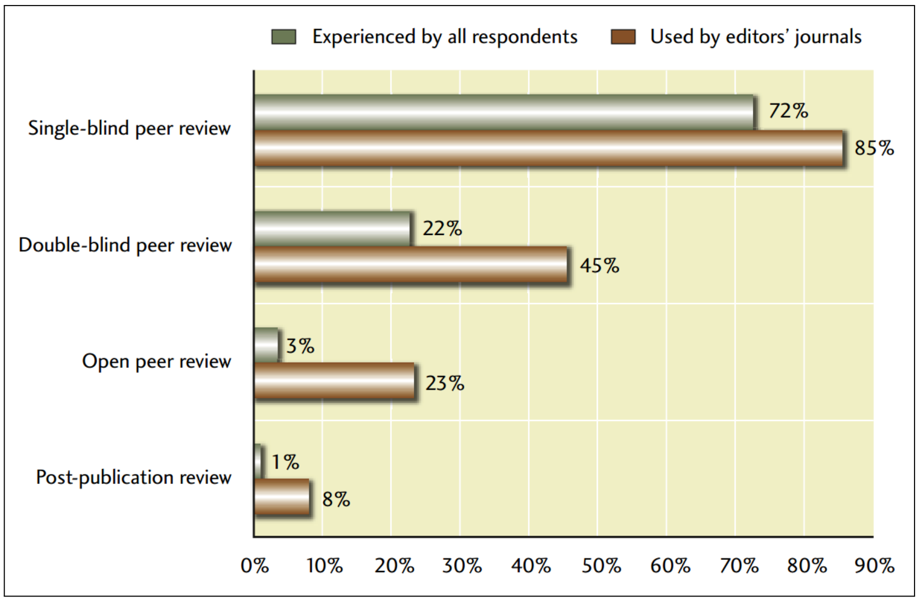


Figure 1: This figure shows the different types of peer reviewing techniques and how often they are experienced by authors and journal editors (Image source: [23])

But the process of peer reviewing is not unquestioned within the scientific community, e.g. Richard Smith presented lots of critiques about peer reviewing in his article "Peer review: a flawed process at the heart of science and journals"[19]. It is slow, expensive, inconsistent, biased and may also be abused. Mayur Amin from Elsevier[2] summarizes the current situation very well:

"Peer review is not perfect, but its the best we have."

So many large publisher and organization uses peer-reviewing as their main process for filtering the best papers amongst all submissions. These papers will then get published in their journals or conference reports. One of these organizations, Eurographics[3], uses the peer reviewing process since their first workshop in the year 1980.

In the year 1999 they developed their own online reviewing system. Their first prototype was called Managing Conference Proceedings (MCP). This system had a user and submission managment. Users

2 Peer Review at the APE (Academic Publishing in Europe) Conference, Berlin, January 2011
3 an European Association for Computer Graphics - http://www.eg.org

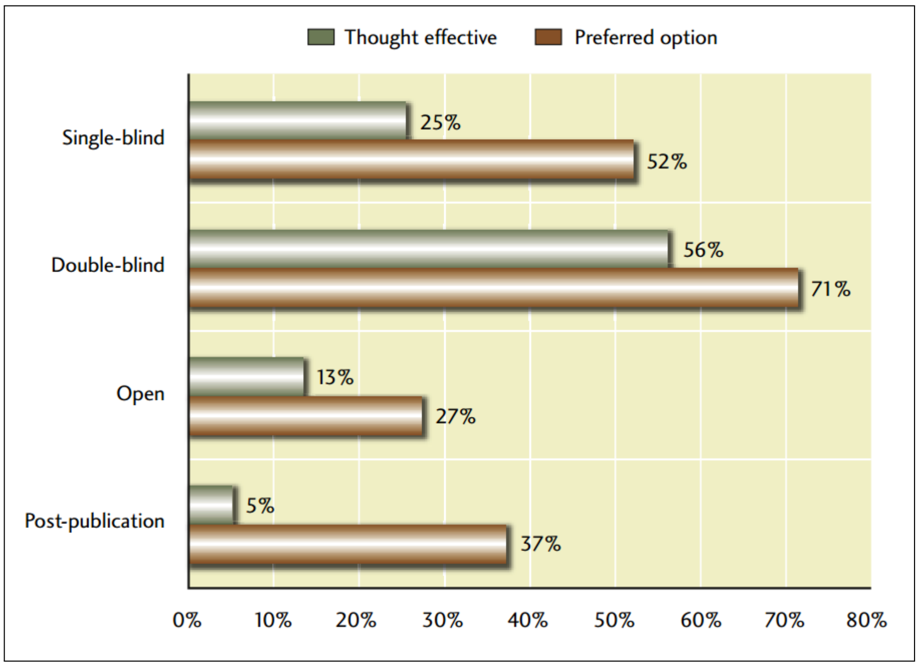


Figure 2: This figure shows the different types of peer reviewing techniques, how they are thought to be effective and whats the prefered choice by the participants (Image source: [23])

could create an account and upload their paper. After assigning the reviewers, they could write a review about the paper and submit it. It was first tested and used in the Eurographics 2000 conference. [6]

Based on the MCP prototype an improved version was developed. This system was called SRM - Submission and Review Management. Like the MCP it is based on a hyperwave information server[4] to manage content and create views. Over the time this system got bigger and got extensions until its current form having over 11.000 members and 25 conferences were managed in the year 2012.[5]

1.2 CONFERENCE EXAMPLE

To get a better understanding of conference management systems in general this section will give an example conference based on the information flow of the SRM system which can be seen in Figure 3.

At first the five different kind of actors of SRM are explained. It is possible that a user can be multiple roles within a conference in the system:

- The chair or a group of chairs are the head of the conference. They sets the rules of the conference, defines deadlines, notifies other users if a deadline is approaching and adds the IPC mem-

4 http://www.hyperwave.com/
5 https://srm.eg.org/

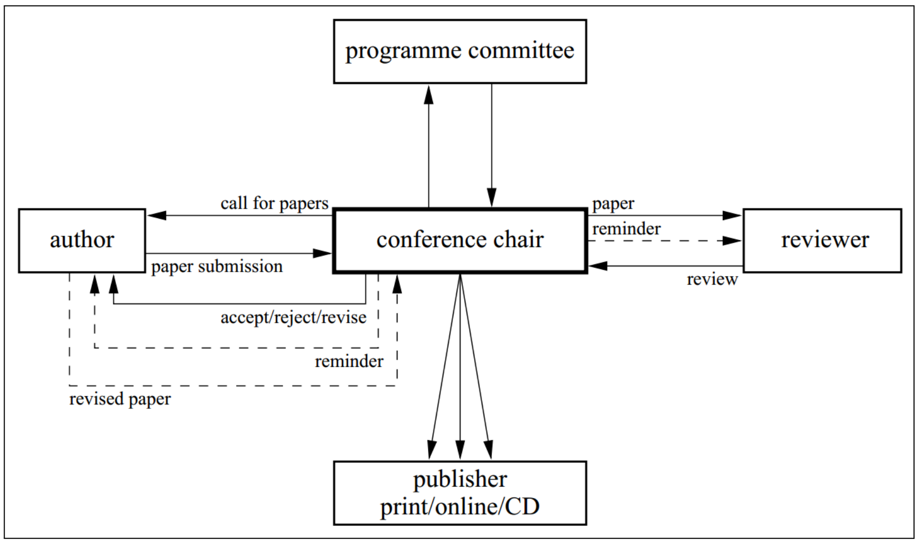


Figure 3: This figure shows information flow of the different actors in the Submission Review System (Image source: [26])

bers to the conference. The most important part of their role is to assign the reviewers to the papers.

- The International Program Committee (IPC) are special users in the system seen as experts in the field of the conference. They voluntary review papers in the conference and support the chair with their suggestion of acceptance. In the system the IPC members are added to the conference from the chair. Then define in which exact areas they are experts within the conference. Further they specify which papers they can review. When they are added as reviewer to a paper they have to review it and find further reviewers. In the end they help the chair accepting and declining the papers.

- The author in the system uploads the paper and additional media files. Further he adds other authors to the paper, sets the preferences of the submission and fill out the additional metadata fields like the title or abstract. If the submission needs revisions or the paper gets accepted the author also uploads the revised paper or the camera ready copy.

- Reviewers have the access rights on the paper they are assigned to. After reading this paper they fill out a reviewing form about their opinion on the paper.

- The publisher takes in the end all camera ready copy (CRC) papers and publishes them in their journal or another medium.

Once a conference starts, the chair sets the deadlines and defines the areas of expertises the conferences focuses in. This special fields may be for example the ACM computing classification[6] (see Figure 4) or any other classification which fits for the conference. When this is done the chair adds users as IPCs to the conference. These IPC users may then define which knowledge they have in the defined areas of expertise. In SRM there are currently 4 possible answers to define the knowledge of a user: *"Expert"*, *"Knowledgeable"*, *"Passing"*, *"No Knowledge"*. This answer will then have an impact on the assignment for the reviewing process. When this setup is done the chair calls for papers. This call promotes the conference by advertising on the website or sending emails to a mailing list. Authors may then create a submission, upload the paper, set additional authors, fill up additional metadata fields like title or abstract. Further they select up to 5 topics from the area of expertise. The chair may also set an abstract deadline. This deadline defines by what date the submission has to be created and the abstract and title has to be set.

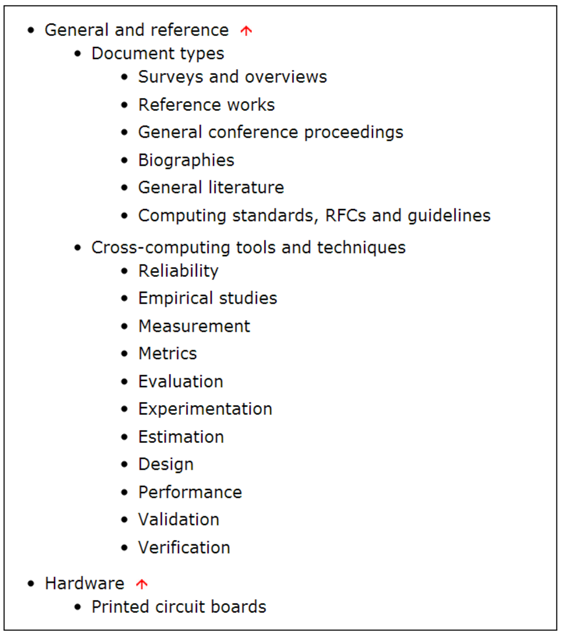

- General and reference ↑
  - Document types
    - Surveys and overviews
    - Reference works
    - General conference proceedings
    - Biographies
    - General literature
    - Computing standards, RFCs and guidelines
  - Cross-computing tools and techniques
    - Reliability
    - Empirical studies
    - Measurement
    - Metrics
    - Evaluation
    - Experimentation
    - Estimation
    - Design
    - Performance
    - Validation
    - Verification
- Hardware ↑
  - Printed circuit boards

Figure 4: This figure shows an extract of the current ACM classification. The current classification can be seen on [3]

When the submission deadline has passed the IPC members may bid on these papers. This bidding process works as follows: The user is presented with all submitted abstracts and titles. According to this metadata he has to decide whether he *"Want to Review"*, *"Could Review"* is *"Not Competent* or has a *"Conflicts of interest"*. According to this selection and the previously set knowledge the IPC member is invited as a reviewer. Further he is encouraged to add further users to the paper as reviewers. After the reviewer has finished his review he

6 A classification for structuring the areas of expertise in Computer science - http://www.acm.org

can discuss with the other reviewers of that paper using a discussion forum.

When the review deadline has passed, the chair has to decide based on the reviews which papers will get accepted, which are declined and which papers need a further review phase. If a paper needs further review the user will see this decision and upload a modified paper. Then the review process will go in another cycle. When a paper is accepted, the user uploads a camera-ready copy (CRC) version of his paper. Then this paper goes into print/production.

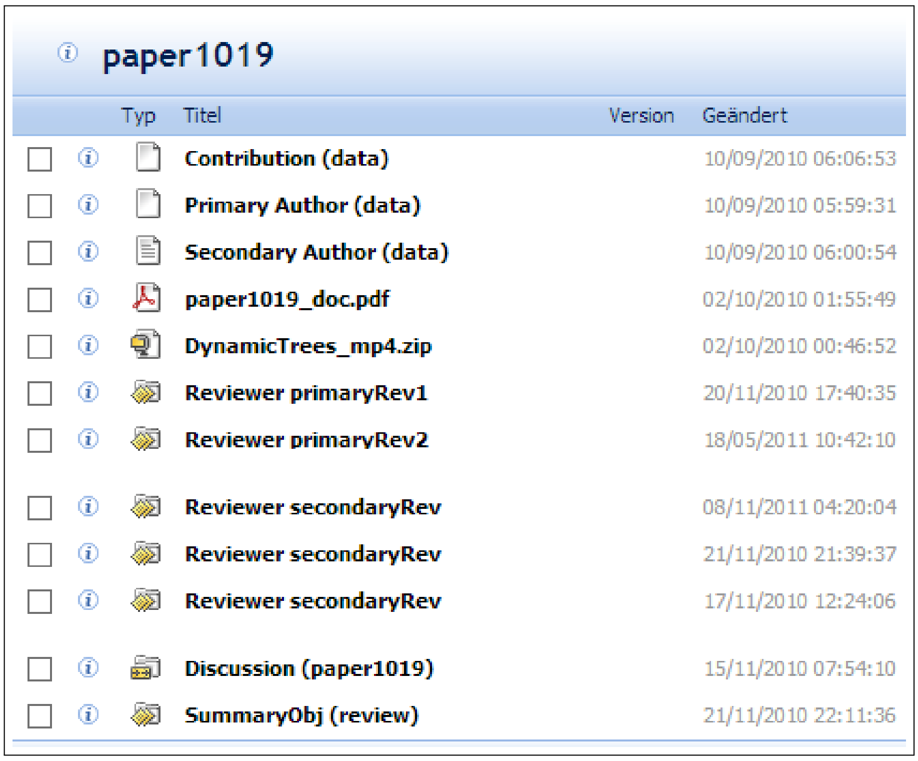
1.3 MOTIVATION

As mentioned MCP and its successor SRM are based on the Hyperwave information server. The Hyperwave information server stores its data unlike relational databases. It creates object and collections which are stored again in collections. Attached to these collections and objects are attributes. These collections, objects and attributes contains the data. To understand this terms better it might be easier to compare it to a filesystem. The collections are similar to folders, objects are like files in these folders and the attributes are like the attributes of the files and folders. For example the created date of a file.

When SRM was created it was decided to create collections out of every submission. Within these collections there are objects like the paper, the contribution which contains the metadata of the submission (see Figure 5). This was done to use Hyperwaves special ability to manage the access rights. In case of SRM the author of paper1019 will never be able to see the Reviewer Collections even if he access the directly the collection. Unfortunately this chosen data model doesn't scale very well like the following example shows:

Lets say the chair wants to see all titles and authors of one conference. Hyperwave uses a function called *getChildObjects()* to select the content of a collection. At first the *getChildObjects()* function is used on the conference select all submissions. Then every submission has to be selected to access the title and the author of the submission similar to browsing in a directory. If there are at average 250 submission (see Figure 6) this example takes 251 statements to complete. This use case and similar listings which queries all papers takes up several seconds in the current system because of the huge amount of work Hyperwave has to process.

When the system was designed in the year 1999 the bottleneck was the internet and the hardware. People used their modems to connect to the internet. So they were used to wait several seconds before server responded. And Hyperwave processed the listings faster than the internet could transfer the data. However the internet and hardware got faster over the years but SRM didn't reduce its respond time.



paper1019

Typ	Titel	Version	Geändert
	Contribution (data)		10/09/2010 06:06:53
	Primary Author (data)		10/09/2010 05:59:31
	Secondary Author (data)		10/09/2010 06:00:54
	paper1019_doc.pdf		02/10/2010 01:55:49
	DynamicTrees_mp4.zip		02/10/2010 00:46:52
	Reviewer primaryRev1		20/11/2010 17:40:35
	Reviewer primaryRev2		18/05/2011 10:42:10
	Reviewer secondaryRev		08/11/2011 04:20:04
	Reviewer secondaryRev		21/11/2010 21:39:37
	Reviewer secondaryRev		17/11/2010 12:24:06
	Discussion (paper1019)		15/11/2010 07:54:10
	SummaryObj (review)		21/11/2010 22:11:36

Figure 5: This figure shows the result of the getChildObjects function in SRM on one submission. Contribution is a textfile which contains the metadata of the file like title and abstract. primaryRev1 is again a collections where the attributes of the collection contains the data about the reviewer.

The current bottleneck of the system is the creation of lists in the Hyperwave information server. Responding times of over one second is not acceptable in the current times. As SRM is completely based on the Hyperwave information server it was decided to dispose the old system and start again from scratch. The new system should be based on current technologies, faster but as extensible and modifiable as the current system. Further it should include all features the current system has.

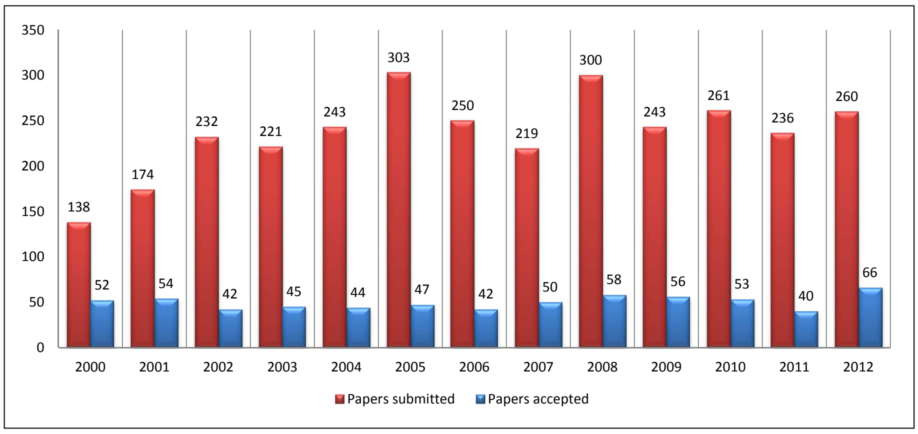


Figure 6: This figure shows the submitted and accepted papers over the last years of the Eurographics Annual Conference

2 RELATED WORK

2.1 OVERVIEW

This chapter will give an overview of other conference management systems. At first it will show some of the most popular and most used current conference management systems then it will present the old SRM system. It will give a small insight on the strengths and weaknesses of these systems. In the end this chapter will summarize all systems, why none of them satisfied the needs of Eurographics and why it was decided to perform a complete redesign.

2.2 EASY CHAIR

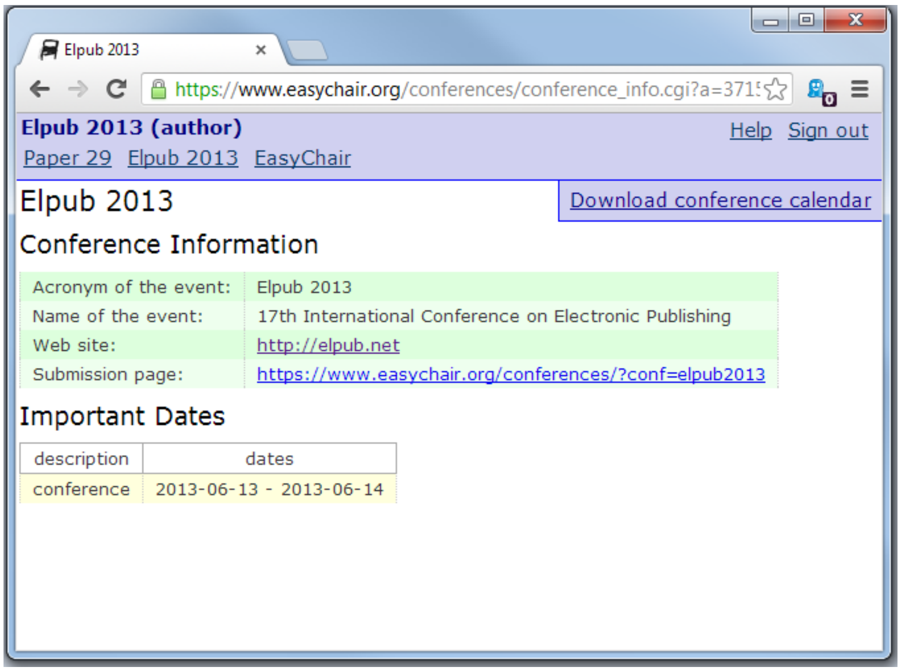
One of the largest conference management systems is Easy Chair(Figure 7). It has hosted nearly 20.000 conferences and is used by over 700.000 users. Easy Chair provides the basic conference management features like paper submission, reviewer assignment and creation of reviews. They also provide a so called rebuttal phase where authors can respond to the reviews. They also claim this management system is so flexible that it has been used already as a project evaluation tool.

One of the strong advantages of Easy Chair is that it is a free product. So it is possible to creates an own conference and manage it, without paying for the service. However they don't provide free support. So if there is the need for help during creation, installation or in managing the conference it won't be free.

Easy Chair supports multiple conference models, from which a chair can choose from. But if non of these models suites the needs of the conference there is no further customization for the chair. So if there is a requirement for the conference which Easy Chair doesn't have, the paid support might implement it for their premium customers. Otherwise it is only possible to use the standard models. [22]

2.3 COMS CONFERENCE MANAGEMENT SYSTEM

Like Easy Chair COMS (Figure 8) is a fairly large management system. They currently have 450.000 page views per month. But unlike Easy Chair their service is not free. There is a one time set up fee. Once this fee is paid, COMS create a website for the needed conference and


Elpub 2013
https://www.easychair.org/conferences/conference_info.cgi?a=371
Elpub 2013 (author)
Help Sign out
Paper 29 Elpub 2013 EasyChair
Download conference calendar

Elpub 2013

Conference Information

Acronym of the event:	Elpub 2013
Name of the event:	17th International Conference on Electronic Publishing
Web site:	http://elpub.net
Submission page:	https://www.easychair.org/conferences/?conf=elpub2013

Important Dates

description	dates
conference	2013-06-13 - 2013-06-14

Figure 7: This figure shows an example page of easychair. It displays the information page of the Elpub2013 conference.

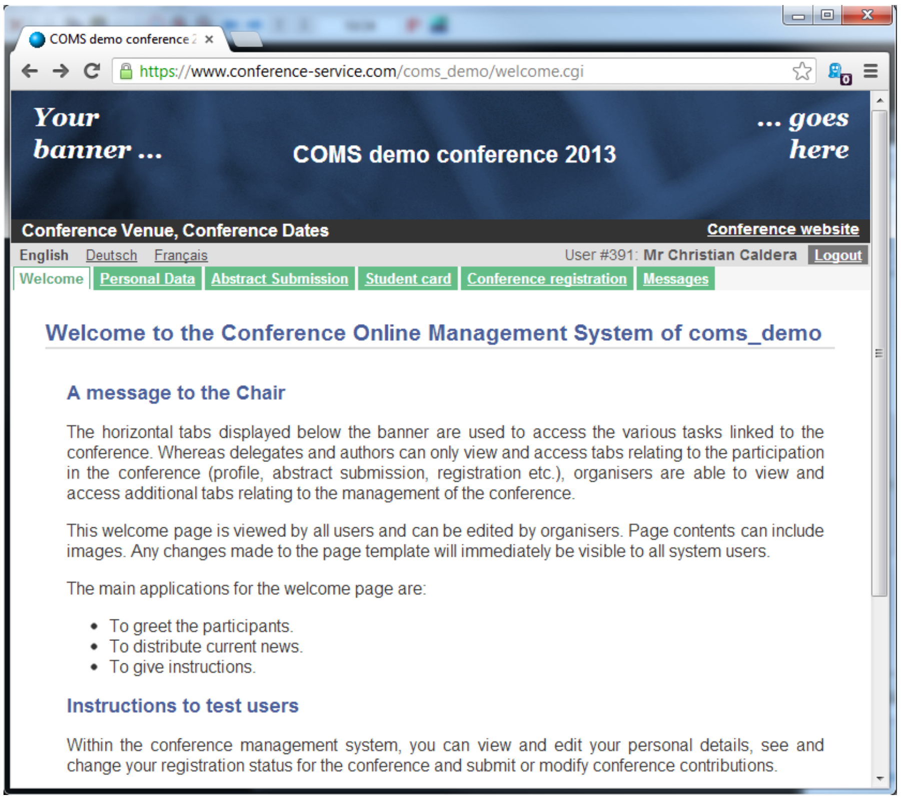
style it the users needs. Further COMS supports 3 different languages (English, French and German) and upon creation of the homepage the chair may choose one of them. When the website is created, there are the standard features of a conference managing system, like creating and reviewing submissions. It is possible to define 9 different review fields, but once they conference is created, there is no possibility no modify these fields. [11]

2.4 OPENCONF

OpenConf is a PHP based conference management tool. They used it for thousands of events in over 100 countries. Like the previos conference management tools, it is possible to upload a paper, assign the papers to reviewers and let it get reviewed.

The standalone feature of OpenConf (Figure 9) is their mobile program. This is an app for various types of smartphones designed to help users in the systems.

To use OpenConf the user has to pay per year per conference. And if the customer wants any customizations for his conference it is possible to contact them, that they implement it for the conference for a fixed price. [10]



Welcome to the Conference Online Management System of coms_demo

A message to the Chair

The horizontal tabs displayed below the banner are used to access the various tasks linked to the conference. Whereas delegates and authors can only view and access tabs relating to the participation in the conference (profile, abstract submission, registration etc.), organisers are able to view and access additional tabs relating to the management of the conference.

This welcome page is viewed by all users and can be edited by organisers. Page contents can include images. Any changes made to the page template will immediately be visible to all system users.

The main applications for the welcome page are:

- To greet the participants.
- To distribute current news.
- To give instructions.

Instructions to test users

Within the conference management system, you can view and edit your personal details, see and change your registration status for the conference and submit or modify conference contributions.

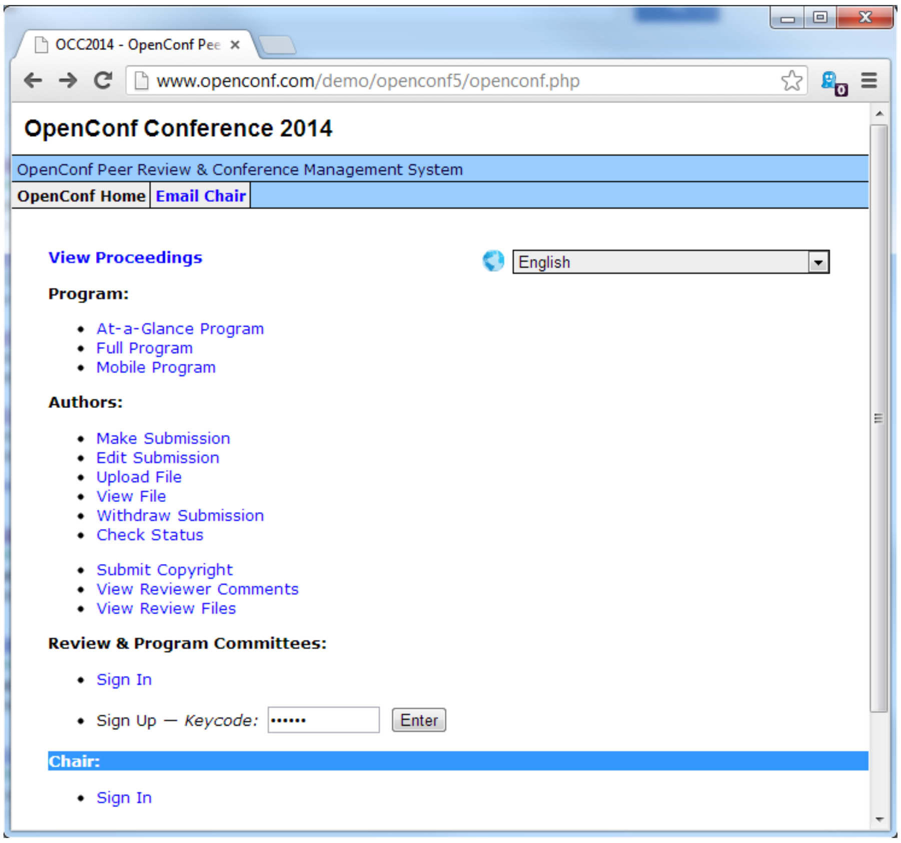
Figure 8: This figure shows the COMS demo conference welcome message after a new account is created.

2.5 CONFIOUS

CONF!OUS The Conference Nous Confious is a conference management tool, where authors can submit their papers, reviewers can bid on them and create reviews. One of the main features of Confious is the dynamic generation of the review form. A chair can setup their own forms and change them to their liking. For example he can add additional textfields, checkboxes or drop-down lists without contacting the administrator of the program and create real dynamic forms. Confious also has a easy to use demo conference. They provide for every role login data to test out their system. Figure 10 shows the demo conference of Confious from the chairs overview about all submitted papers. [12]

2.6 SRM

In the year 1999 the European Association for Computer Graphics (Eurographics) created their first own conference management system. This system was a prototype, with all features for managing



OCC2014 - OpenConf Pee ×

www.openconf.com/demo/openconf5/openconf.php

OpenConf Conference 2014

OpenConf Peer Review & Conference Management System

OpenConf Home | Email Chair

View Proceedings

English

Program:

- At-a-Glance Program
- Full Program
- Mobile Program

Authors:

- Make Submission
- Edit Submission
- Upload File
- View File
- Withdraw Submission
- Check Status

- Submit Copyright
- View Reviewer Comments
- View Review Files

Review & Program Committees:

- Sign In
- Sign Up — *Keycode:* •••••• Enter

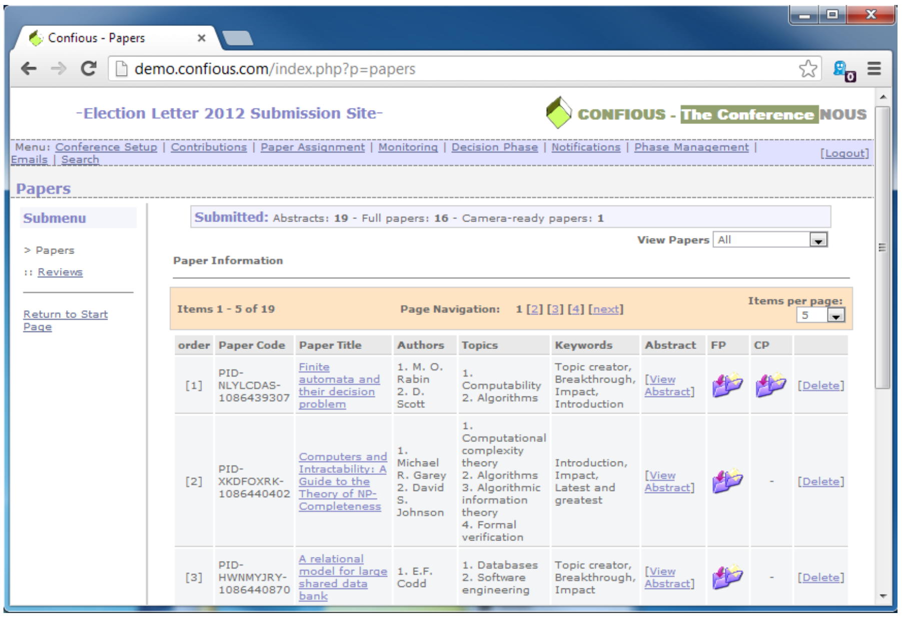
Chair:

- Sign In

Figure 9: This figure shows the Openconf Demo conference 2014. An author can see this screen upon login for his conference and create a new submission.

conferences. The electronic workflow (Figure 11) was created based on the information workflow (Figure 3) and the shown conference example in Section 5.1.

The core part of SRM is the multimedia database managed by the Hyperwave information server. Every actor in the system has access to the data in the server. An author can submit papers with the help of submission tools like templates or authoring guidelines and receive the electronic review. The reviewer on the other hand can receive the paper and submit his review via a review form. The Hyperwave information server manages the access rights so the author won't see the reviewers identity in single- or double-blinded reviewing. The chair can access all data and accept or decline the paper based on the reviewing information. IPCs can access the title and abstract during the specification which paper they want to review. A deamon is always active to remind people of incoming deadlines or for other email notifications. In the last conference phase after the authors submitted their camera ready copy (CRC) paper a publishing tool extracts these papers, and starts the publication process (e.g. reformatting the paper and sending them to print). [26]


Confious - Papers
demo.confious.com/index.php?p=papers
-Election Letter 2012 Submission Site-
CONFIOUS - The Conference NOUS
Menu: Conference Setup | Contributions | Paper Assignment | Monitoring | Decision Phase | Notifications | Phase Management | Emails | Search
[Logout]
Papers
Submenu
> Papers
:: Reviews
Return to Start Page
View Papers All
Paper Information
Items per page: 5

Submitted: Abstracts: 19 - Full papers: 16 - Camera-ready papers: 1

Items 1 - 5 of 19 Page Navigation: 1 [2] [3] [4] [next]

order	Paper Code	Paper Title	Authors	Topics	Keywords	Abstract	FP	CP	
[1]	PID-NLYLCDAS-1086439307	Finite automata and their decision problem	1. M. O. Rabin 2. D. Scott	1. Computability 2. Algorithms	Topic creator, Breakthrough, Impact, Introduction	[View Abstract]	📂	📂	[Delete]
[2]	PID-XKDFOXRK-1086440402	Computers and Intractability: A Guide to the Theory of NP-Completeness	1. Michael R. Garey 2. David S. Johnson	1. Computational complexity theory 2. Algorithms 3. Algorithmic information theory 4. Formal verification	Introduction, Impact, Latest and greatest	[View Abstract]	📂	-	[Delete]
[3]	PID-HWNMYJRY-1086440870	A relational model for large shared data bank	1. E.F. Codd	1. Databases 2. Software engineering	Topic creator, Breakthrough, Impact	[View Abstract]	📂	-	[Delete]

Figure 10: This figure shows the Confious demo conference as a chair. This overview of submissions provides the chair all necessary information about the submitted papers.

Since 1999 a lot of modifications were done to the system until its current form (see Figure 12). Most of these ideas started as suggestions of chairs and users on how to improve the program. A lot of these suggestions have been implemented. For example the bidding process was expanded. IPCs are now able to specify the paper they want to review and areas where the IPC is knowledgeable in. This feature helps the chair to create a better matching between the IPCs and the papers.

As mentioned in Section 1.3 the system does not scale when creating large lists. One of these use cases is when the chair loads all submissions and reviews to accept and decline the papers. Florian Sumann evaluated this use case in his Bachelor-thesis[21]. The communication alone takes 5 minutes to load, filter and save and filter the data. In his thesis he describes how he tried to increase this performance by the means of *Asynchronous JavaScript and XML* (AJAX). AJAX is a technology where data is loaded asynchronous after the page is displayed at the client. For further information about AJAX refer to Section 3.2. Before this change, it was only possible to load all the data when the chair wanted to get an overview of the current submissions. Now the system displays the page faster and loads the submission data dynamically when it arrives.

Unfortunately the test results didn't really meet the expectations and the anticipated speed up was absent. The overall loading times grew a bit. It was still partly a success, because the systems workflow is now more efficient. The chair doesn't have to wait several minutes

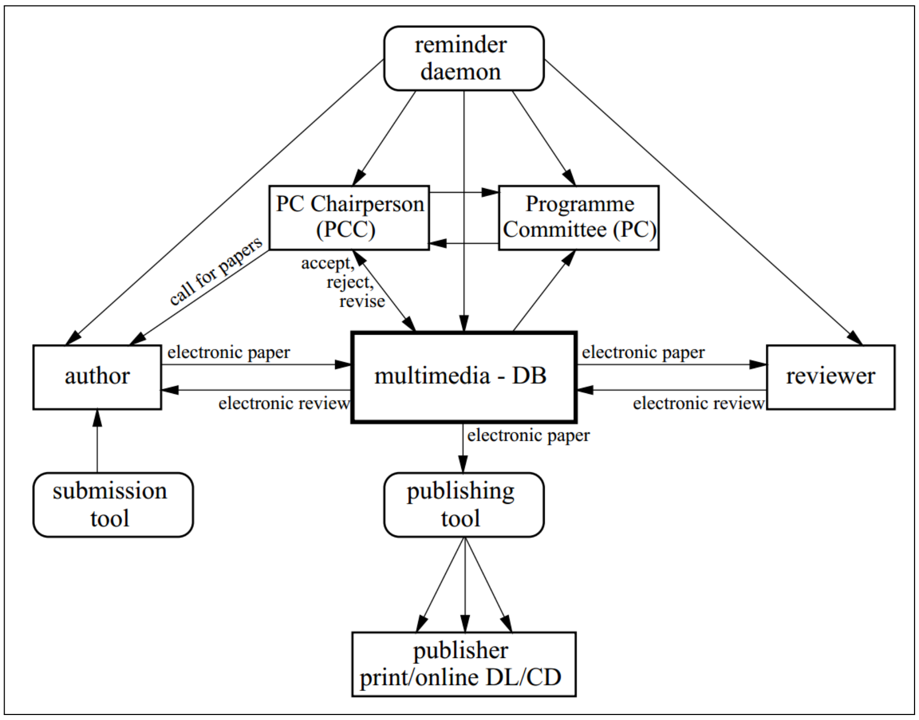


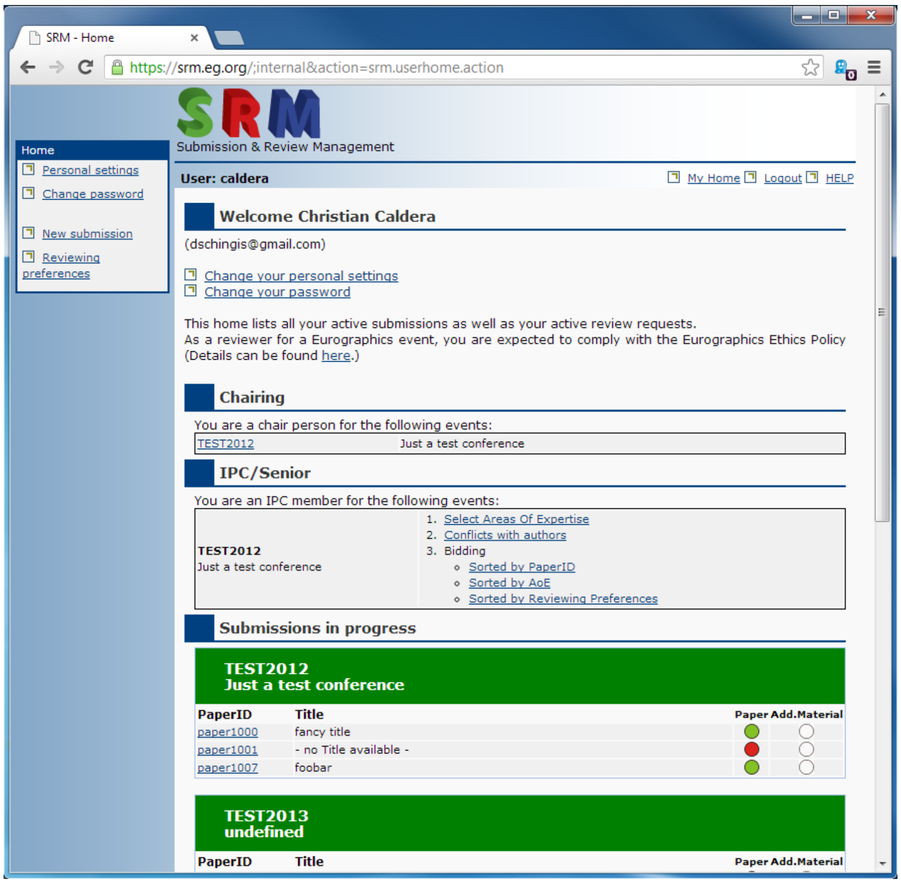
Figure 11: This figure shows the electronic workflow of the Submission Review System (Image source: [26])

for all papers to be loaded and displayed. The request without papers is pretty fast and the first papers are also loaded within seconds. So the chair can start working on the them when the page is still in generation. [21]

2.7 CONCLUSION

As this chapter shows an excerpt of conference management systems. All of them can execute the basic conference management features. They are all modular by either switching configuration modes like easychair or including modules like OpenConf. Most of them also offer some kind of bidding or automatic assignment for the reviewers based on the preferences they gave. So why is there the need for a new management system?

One of the most wished feature for SRM were additional fields in the review and the submission form. The current solution in SRM to satisfy the users are 26 fixed fields in the review form and 6 metadata fields the submission form. But these forms aren't needed in every conference. The new system should have the possibility to include these 26 or less fields. Such a dynamic review forms like Confious were a requirement for the new system. Unfortunately Confious doesn't have dynamic submission forms for additional metadata fields like contribution or benefit.


SRM - Home

https://srm.eg.org/;internal&action=srm.userhome.action

Submission & Review Management

Home
- Personal settings
- Change password
- New submission
- Reviewing preferences

User: caldera My Home Logout HELP

Welcome Christian Caldera

(dschingis@gmail.com)

Change your personal settings
Change your password

This home lists all your active submissions as well as your active review requests.
As a reviewer for a Eurographics event, you are expected to comply with the Eurographics Ethics Policy
(Details can be found here.)

Chairing

You are a chair person for the following events:

TEST2012	Just a test conference

IPC/Senior

You are an IPC member for the following events:

TEST2012 Just a test conference	1. Select Areas Of Expertise 2. Conflicts with authors 3. Bidding ◦ Sorted by PaperID ◦ Sorted by AoE ◦ Sorted by Reviewing Preferences

Submissions in progress

TEST2012
Just a test conference

PaperID	Title	Paper	Add.Material
paper1000	fancy title	🟢	○
paper1001	- no Title available -	🔴	○
paper1007	foobar	🟢	○

TEST2013
undefined

PaperID	Title	Paper	Add.Material

Figure 12: This figure shows SRM home screen where. It gives an overview where the user is chair, ipc and his current submissions.

The second requirement is the bidding system. As explained in Section 1.2 with the current system it is possible to specify the paper and the expertise areas of an IPC. Most of the systems can bid on the papers but very few of them can bid on expertise areas of a conference. Further with the current system a new approach for these bidding system is in development. By means of natural language processing current papers of are IPCs processed and matched to submitted papers in the conference. This matching is done to automate the assignment and bidding process. The desired system should also be able to use this system with an interface.

This interface should also serve as an *application programming interface* (API). This API is intended to make it easier, to add further additional modules to the program when they are needed.

As there are currently no known systems which supports all these features, Eurographic decided to create a new improved SRM system instead of using an existing one and adapt it to their needs.

3 TECHNOLOGIES

3.1 OVERVIEW

This chapter will give an overview over the used technologies and techniques. It should build up the foundation for understanding the next chapters. It will also give an insight why certain technologies where chosen.

3.2 USED TECHNOLOGIES

HTTP "The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers." [4]

HTTP is the current state of the art protocol for loading website to the browser within the World Wide Web. The so called verbs define the action of the user. The current HTTP/1.1 specification defines nine verbs. The two most common used verbs are: GET and POST. The GET verb requests a specified resource on the server. The typical use case for this verb is when the user enters a Uniform Resource Locator (URL) in his browser (e.g: `http://localhost/COMFy/Home`) or clicks on a hyperlink. The browser will translate this URL to a GET request (see Listing 1) and send it to the server. The response of the server is the received website. The POST verb on the other hand sends data to the webserver. The typical usecase is, when a user fills out a webform or uploads a file to the server.

```
GET http://localhost/COMFy/Conference/EG2012 HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0) Gecko/20100101
    Firefox/16.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Listing 1: This listing shows a GET request example. The browser converts the URL to such a request. In this process it appends the further fields.

The additional fields in Listing 1 give the server information about the connected user. For example the local language. This information might be used from the server to create a better user experience. The server might translate the website in the users native language with this information. One important field for the framework is the *Accept* field. This field describes in descending order the representation of the response. In the example the user prefers HTML before XML. [4]

MODEL VIEW CONTROLLER PATTERN The model view controller(MVC) pattern is a pattern which consists of 3 parts. These parts are, as the name suggests a model, a view and a controller. The core idea behind this pattern is to make code reusable and maintainable. This is done by separating the data from the presentation.

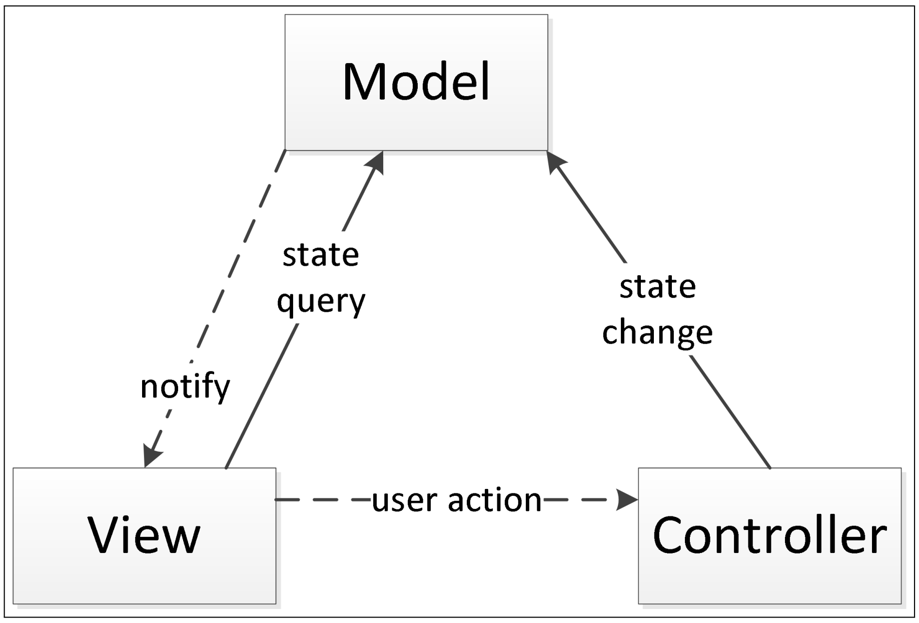


Figure 13: This figure shows shows the typical model view controller workflow. The model contains the data. Once the data changes the model notifies the view which querys the model to get the updated data. The view is seen by the user. When the user modifies data on the view the controller notifies it and changes the data on the model. (picture derived from: [13])

The model is responsible for the data storage, the view is for displaying it to the user and the controller connects the two parts together. Figure 13 shows how these parts are connected. User actions are handed over to the controller over the view. The controller changes the states of the model who notifies the view. The view queries the model and displays the new data. The idea is now if either the model or the view changes the other part doesn't have to be changed. [13]

ASP.NET MVC provides an framework which enforces structured MVC architecture. This leads to a better code design. In return this results to tidier code encapsulation and easier maintenance. This is necessary as the program will be extended and modified in the future. The ASP.net MVC framework with C# as programming language was a requirement of Eurographics for the project. [15]

JAVASCRIPT is a programming language which is mainly used in web browser to modify the Document Object Model (DOM) of a HTML site. For example asynchronous loading of additional data (AJAX) or improving the browsing experience for example with jQuery. Due to its constant development to create better and faster Javascript interpreter it is now also used outside this client-side scripting like PDF documents or desktop widgets. [9]

JSON stands for JavaScript Object Notation. It is a format to save and transport data like XML. JSON based on the Javascript language which has a built in function to parse the data. Objects are encapsulated in curly braces. These braces contains name/value pairs where the value itself can be null, false, true, a string, a number, an array or an inner object. If the value is as a string it is enclosed by two inverted commas, an array is enclosed by square braces and a inner object is enclosed by further curly braces.

JSON is often seen to be more compact than XML. And due to its increasing popularity, additional parsers for nearly every other programming language were developed. Listing 2 and Listing 3 shows the two formats next to each other. They both store the information of a circle. [5]

```xml
<?xml version="1.0" encoding="
    UTF-8" ?>
<circle>
  <radius>1</radius>
  <coordinates>
    <x>2</x>
    <y>3</y>
  </coordinates>
  <Name>Unit circle</Name>
  <IDs>
    <ID>0</ID>
    <ID>1</ID>
  </IDs>
</circle>
```

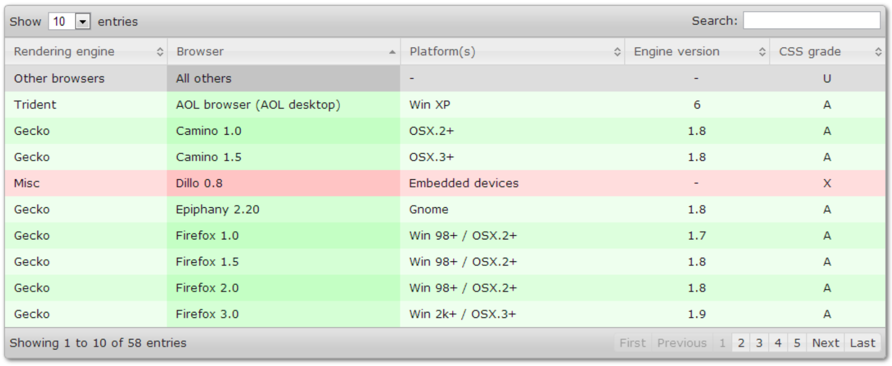
Listing (2) This listing example shows a circle stored in XML format.

```json
{
   "circle": {
      "radius": 1,
      "coordinates": {
         "x": 0,
         "y": 0
      }
   }
   "Name": "Unit circle",
   "ID": [0, 1]
}
```

Listing (3) This listing example shows a circle stored in JSON format.

JQUERY jQuery is a Javascript library which can be used to simplify the manipulation of the Domain Object Model. There are a lot of different widgets, tools and effects which enriches the user experience and simplifies the development. Some examples provided by the jQuery UI[1] are: Dialogs, Menus, Progressbars, Spiners, Tabs. [7]

One example of a tool which is based on the jQuery library are datatables[2]. By providing the data in a defined way, the datatables creates a sortable, searchable table with paging and dynamic amount of entries (see Figure 14).



Show 10 entries Search:

Rendering engine	Browser	Platform(s)	Engine version	CSS grade
Other browsers	All others	-	-	U
Trident	AOL browser (AOL desktop)	Win XP	6	A
Gecko	Camino 1.0	OSX.2+	1.8	A
Gecko	Camino 1.5	OSX.3+	1.8	A
Misc	Dillo 0.8	Embedded devices	-	X
Gecko	Epiphany 2.20	Gnome	1.8	A
Gecko	Firefox 1.0	Win 98+ / OSX.2+	1.7	A
Gecko	Firefox 1.5	Win 98+ / OSX.2+	1.8	A
Gecko	Firefox 2.0	Win 98+ / OSX.2+	1.8	A
Gecko	Firefox 3.0	Win 2k+ / OSX.3+	1.9	A

Showing 1 to 10 of 58 entries

Figure 14: This figure shows a table based on the jQuery library. It features searching, sorting paging and dynamic amount of entries. (Image source:[20])

AJAX stands for *Asynchronous JavaScript and XML*. AJAX is used for sending and retrieving data after the page is loaded and displayed. A classic request sends a HTTP request to the server, the server processes the request and sends the page with the data back. When this data is retrieved at the client the browser renders the page and displays it to the user. This request might take some time where the user is forced to wait. When AJAX is used the server responds immediately with the page. This page is rendered and displayed. After the page finished rendering the browser requests the data from the server. When the data arrives at the client, the browser changes the DOM of the HTML site and injects the data into the site Figure 15. [24]

Some usecases where AJAX is used are:

- The user requests a huge amount of data and wants to start working before the data arrives at the client. This improves the user experience as the user can start working when still some data is missing.

1 http://jqueryui.com
2 http://www.datatables.net

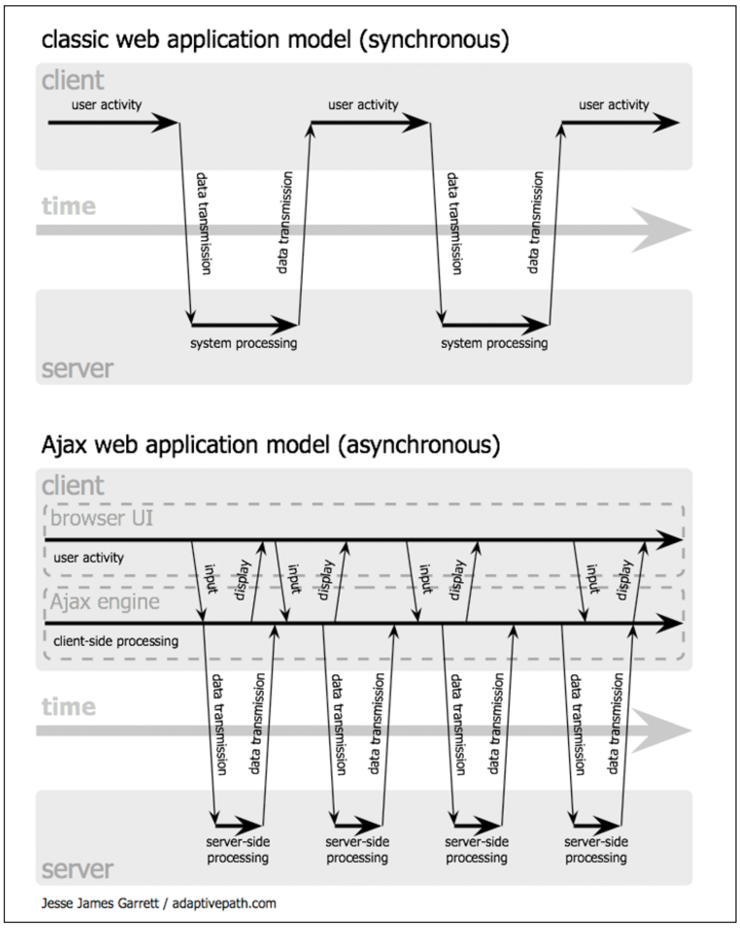


Figure 15: This figure shows the difference between an Ajax and a classical request. With the classical request the client always has to wait for the data. In Ajax the client can dynamically load further data and displays it when the request is complete (Image source: [8])

- The user requests a mainpage of website and is only interested in a link. When the page is displayed the server starts incrementally sending the data but stops when the user leaves the page. This saves processing time at the server.
- When the user changes only some values in a huge form it is advantageous when only the changed data is sent back instead of the whole form.

MICROSOFT SQL SERVER The Microsoft SQL server was the chosen Database management system over MySQL because of two main reasons. The first one is, when developing with Microsoft Visual Studio and ASP.net it will integrate into the project without significant problems.

The second one is the new Filestream feature of Microsoft SQL Server 2008. Prior to this feature there were two possibilities to save huge binary files. Either the the blob (binary large object) files are stored in the database. This will then gain transactional consistency and a reduced complexity when managing these files. But it will result in a bad performance when accessing the data. The second option is that the data are saved as files on the disk and the links to the files are saved in the database management system. This will result in good performance but managing the data, like creating backups, is very complex to do. Also the database can't guarantee transactional consistency on the file system. With the new Filestream feature the database will manage the meta information of the files in the database while saving the blob files on the disc, where the folder is managed by the database. This way it is possible to get the benefits of both worlds. [18]

DEPENDENCY INJECTION & NINJECT Dependency injection (see Figure 16) is a design pattern which resolves dependencies of objects at runtime [17]. An example when such feature is needed is the following: Assuming there is a live system with data in a database and test data. When testing a new feature it is normally required to either switch out the real data in the database or switch the whole database. With the dependency injection pattern it is required to implement the solution against an interface and just switch the implementation. Listing 4 connects one interface of such a repository to the real implementation. There it is further possible to write a test repository, unload the kernel and connect it then to the test implementation. Ninject provides a container for injecting dependencies.

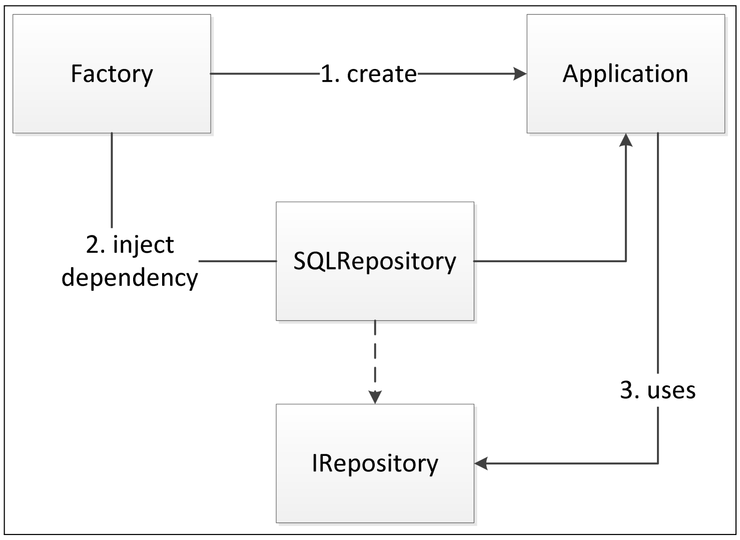


Figure 16: This figure shows how dependency injection works. At first a factory creates an application or an instance of an application. Then it injects the SQLRepository dependency into this application. The application itself uses the IRepository which is an interface of the SQLRepository.

```
   public interface IRepository {
     Conference getConference(int ID);
   }
 4
   public class SQLRepository : IRepository {
     public Conference getConference(int ID) {
       return select * from Conference where id = ID;
     }
 9 }

   public class RepositoryModule : NinjectModule {
     public override void Load() {
         Bind<IRepository>().To<SQLRepository>();
14     }
   }

   public void main() {
     IKernel kernel = new StandardKernel(new RepositoryModule());
19   IRepository repository = kernel.Get<IRepository>();
     repository.getConference(5);
     ...
   }
```

Listing 4: This listing shows a ninject dependency injection. There is a repository interface (Line 1) an implementation of the interface (Line 5). With ninject it is now possible to define a Kernel (Line 18). This kernel glues the Interface with the implementation together (Line 13). After this setup it is possible to talk only to the interface of the repository (Line 19 - 20)

STATEMACHINE A statemachine is a mathematical construct which describes the behaviour of the system. The state machine has a finite amount of states and can switch with state transition between them. [25]

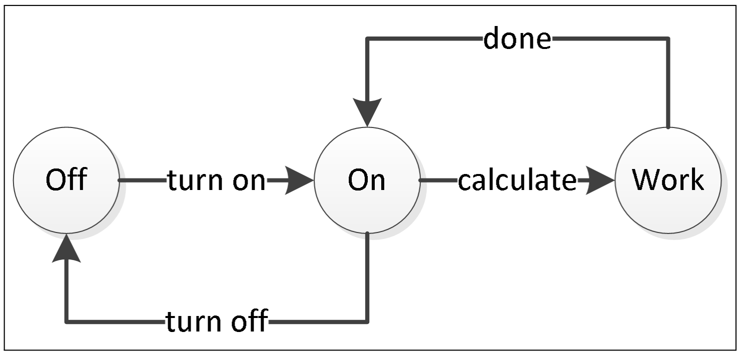


Figure 17: This figure shows a simple state machine which has 3 different states and state transitions to switch between them.

An example of such a machine can be seen in Figure 17. In this example there are 3 different states: Off, On, Work. It can switch between this states. By turning the machine on or off it switches between those two states, Further it is also possible to let the machine calculate. During calculation the state machine is in the state Work. When it is done working the state machine switches back to state on.

4 SYSTEM DESIGN

4.1 OVERVIEW

COMFy can be seen from two different points of view. The first one is the abstract view of the typical conference workflow. This means how the conference gets created, papers are uploaded and papers are reviewed. The second view is the system design and technical details about the implementation. This section will explain the system design in detail, Section 5.1 will show the conference from its different phases.

The new system consists of two parts. First there is COMFy which stands for COnference Management Framework. This framework provides an API to manage a conference. The second part of the system is SRMv2. The majority of users don't want to use the API or read an API description to access the framework. For this purpose SRMv2 was developed. SRMv2 provides the user a graphical user interface (GUI) for using the framework as it is a normal conference management system. At first this chapter will address the framework. Then Section 4.7 will address the API and in the end Section 4.8 will explain the GUI which will be seen by users.

The structure of COMFy is based on different stacks. Each of these layers serves a specific purpose and adds functionality to the underlying layer. The overview of all layers can be seen in Figure 18.

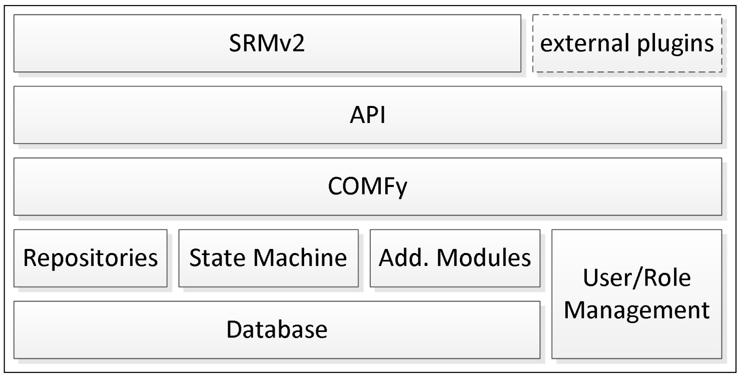


Figure 18: This stackdigaram shows the internal structure of COMFy

The first layer is the relational database. This is where all the data, the papers and additional media files are stored. The second layer is the Domain layer. It consists of two different parts. At first there are the repositories. They query the database and provide the data as a repository to the upper layer. The second part is the state machine. It consists of the business logic for swapping the submissions between the phases. It is also possible to add separate modules on this layer which can be accessed by the framework. As user and role management the generated providers of the framework are used.

The next layer is COMFy and its API. COMFy uses the as a model view controller (MVC) pattern. The controller requests data from the repository in the domain and parses the data into its own model if necessary. This model is then passed to the view in the SRMv2 system. If the user is interested in the data instead of the SRM system, COMFy will prepare the modeldata as JSON or XML.

4.2 USER/ROLE MANAGEMENT

A reason why the Microsoft SQL server was used is that the MVC framework can generate an out of the box user management system. This system generates register and login forms and security features like salted and hashed passwords. It also has some further features like counting the amount of failed password login attempts or when the password was last changed. So if these security features are needed in any time in the future, they are already implemented and only need to be enforced. For example a user may only attempt to login 5 times, until he gets a temporary ban in the system. The database behind the generated system is documented in Figure 19.

A further feature which is used in COMFy is the roles management. The checks if someone has the permission to certain actions are always performed against the Role provider. So if a user performs a certain action for which part it needs a certain role, these action gets rejected an the user is redirected to the associated main screen if the user doesn't have the permission.

The disadvantage of the automated database is the profile table. When using the built in ProfileBase and its functions the performance decreases drastically. This is because the ASP.net framework doesn't know which profile information the user wants to save. They are parsed at compile time from the web.config file. So the whole profile information of a user is stored in one string. They get parsed there with delimiter and information how long an entry is for each profile. And each profile call for each profile field for every person is one connection to the database.

When using this practice with over 11.000 users like SRMv1 currently has it is impossible to get decent access times to the user table. Therefore it was decided to discard the built in profile functionality and create a own. The new one is the profile table, as seen in Figure 20.

The applications table, which is used when creating further application on the same userbase, is currently not used in COMFy.

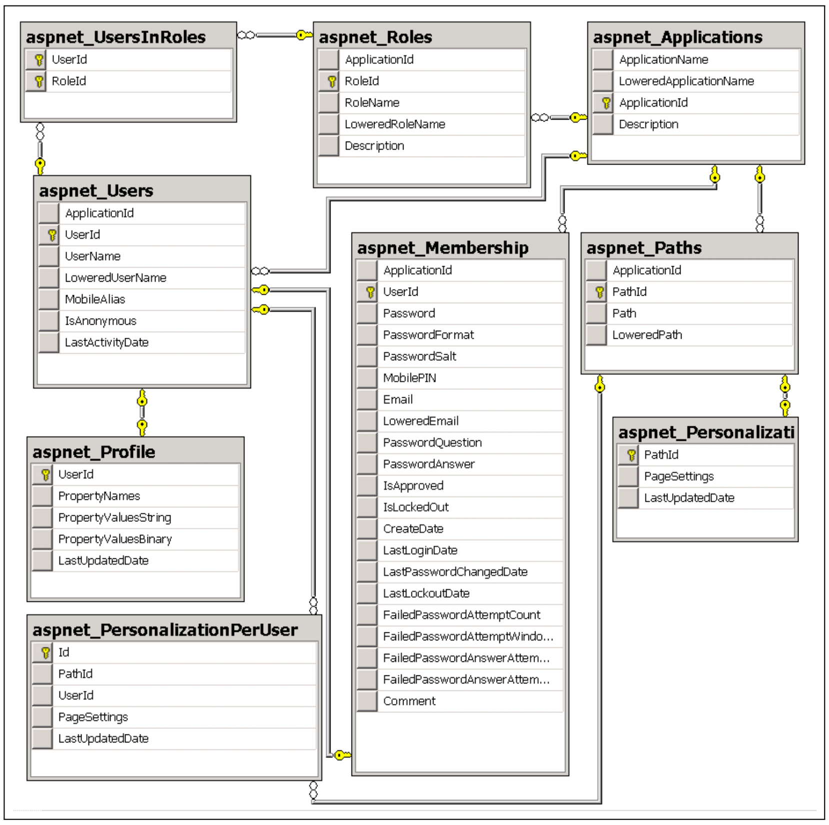


Figure 19: This figure shows the the first part of COMFys database. These tables are generated by the ASP.net framework. The framework also provides the functionality to interact with these tables. Like registering which creates a new user or handling the roles in the system.

4.3 DATABASE SCHEMA

The papers and the zip files are stored in the AdditionalSubmission-Content table. The particular feature of this table is the column ContentData. This column has the previously mentioned filestream attribute (Section 3.2). With this attribute the files are stored in folders and managed by the database instead of being in the database itself.

The second table which contains the filestream attribute is the Files table. It is used for storing additional files wich are not papers or paper related material within COMFy. Currently it is only used for linking files for a conference. In the future it might be used for global files too therefore it needs an additional conference2files table within the system. Its current main purpose is to store Latex and Word templates for the conferences.

To use the filesteam attribute to its full advantage, it is necessary to circumvent the database management system. Otherwise, there wouldn't be a speed up in the access times. When the new line is inserted in the table the database creates a dummy file from which the path is selected. Once the routine has the path, it can inject the file into the dummy file through SqlFileStream. After the file is written to the disk the md5 hashvalue is calculated on the stored file. This hash value is saved in MD5Hash and presented to the user after the upload. This way a user can ensure, that his file is stored on the server without communication errors.

When creating, a conference the chair can create additional fields for each of the submissions. These additional fields is stored in the AddConfFields table. When a user wants to submit a new paper the fields from the AddConfFields are parsed and presented to the user. Upon submitting the input of the user to these fields are stored in the AdditionalSubmissionFields.

The conference chairs, IPC members and authors are stored in their respective tables. Whereas the authors have an additional field called ordering. This ordering represents the order of appearance in the paper.

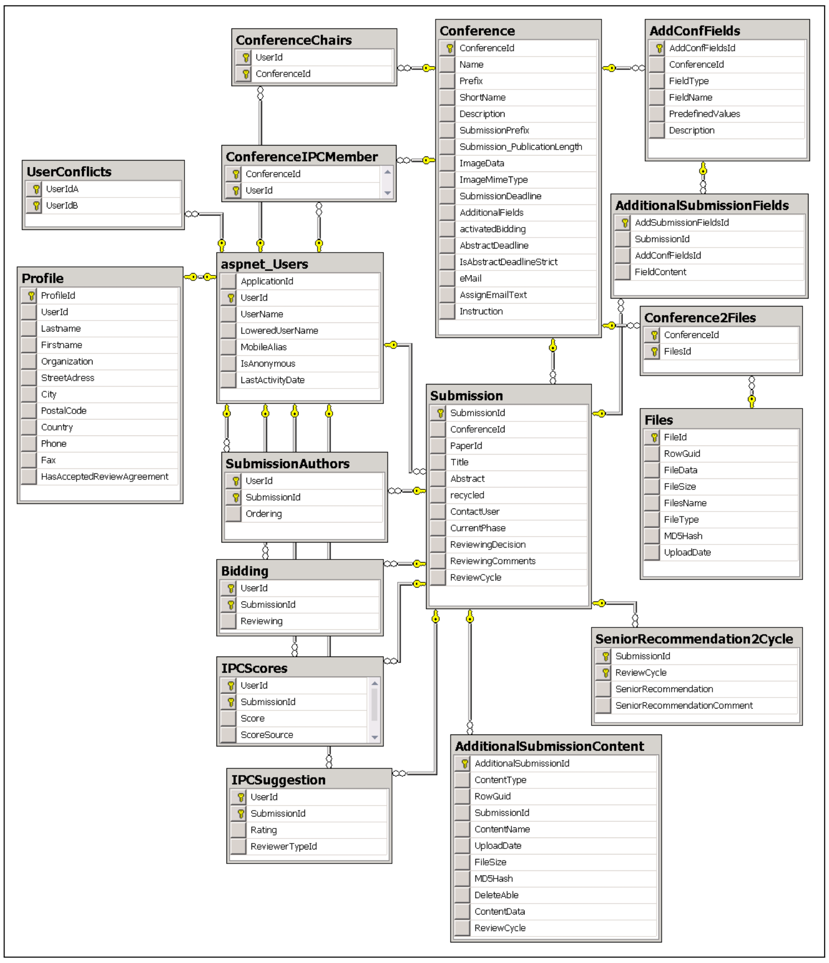


Figure 20: This figure shows the second part of the COMFy database. It displays how the conferences, submissions and profiles interact with each other and how they are connected to the generated database.

The Bidding table (Figure 20) contains specification which IPC user is able to review which submission. The IPCscores can be injected from an external source. The Preferences2User table contains the knowledge of the IPC users to the areas of expertise which are defined in the conference. These tables are used in the to create a suggestion which reviewer is assigned to which submission as in Section 5.4.2 described. The injected IPCscores is a simple addition to the aoeRating to let chair define weighting of these scores.

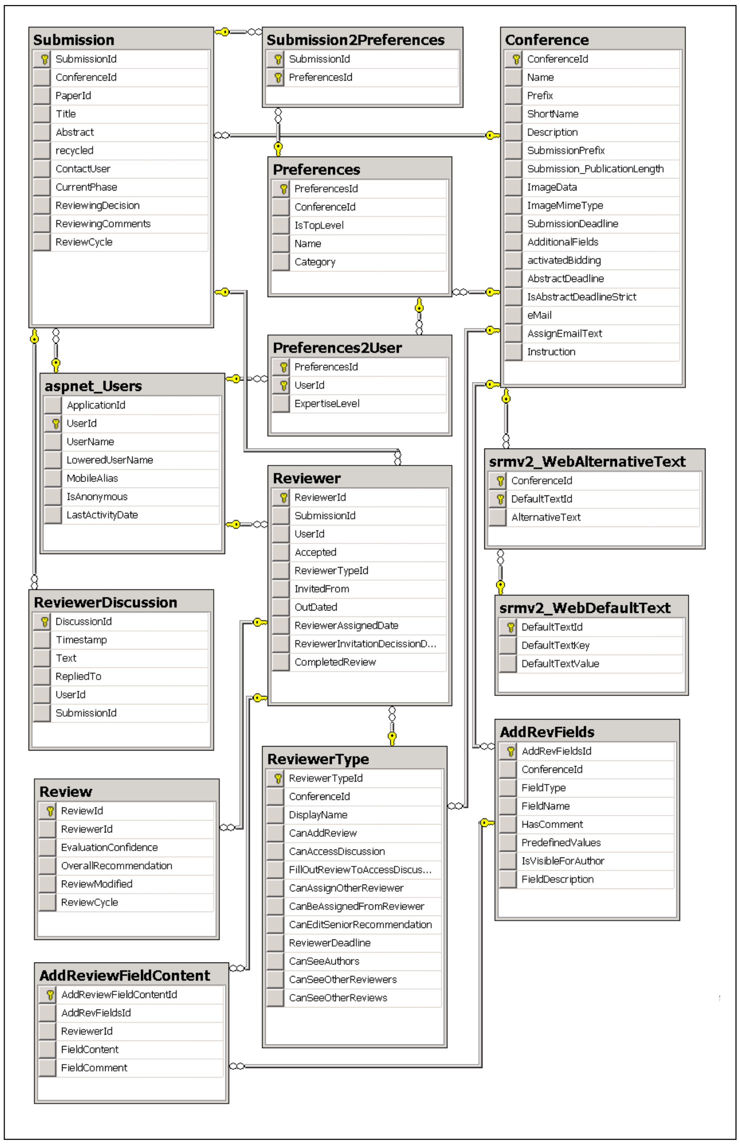


Figure 21: This figure shows the third part of the COMFy database. It displays how the preferences interact with the conference and the reviewers interact with the submissions.

Figure 21 contains the database entries for the areas of expertise (preferences) and the reviewing fields. The mandatory fields for every review are stored in the review table. Upon completion of these fields the reviewer has access to the reviewer discussion. In the discussion it is also saved whom a reviewer replied to, in order to create a folder like structure in the discussion.

Before assigning a reviewer it is necessary to define the review types in the conference. This defines the specific access rights a reviewer has or needs to accomplish. For example, if the FillOutReviewToAccessDiscussion flag is set, the reviewer needs to fill out the review form before he can access the discussion forum. Each reviewer must be associated with a reviewer type.

The additional fields for the reviewing form are like the Additional-Submission fields stored in an separate table called the AddRevFields. They are created by the chair when he configures the conference. When a reviewer is presented the standard review fields, the additional fields are added to the form. The values which are entered by the reviewer are then stored in the AddReviewFieldContent.

The preferences table is filled from the chair after the creation of a new conference. The mapping of the preferences to the submissions and to the users are stored in the Submission2Preferences and Preferences2User fields. Whereas the Preference2User fields also have an expertise level entry in order to store how much expertise a user has in a specific area of expertise.

A further functionality of COMFy is the dynamic web content. In the srmv2_WebDefaultText are key-value pairs of text, which contain certain parts of content of the website. Like the description of what a user should enter at the submisson form. This functionality was created to offer the possibility to replace this text with an own. This text is then stored in the srmv2_WebAlternativeText table.

4.4 REPOSITORIES

> The usual way to enforce separation between the domain model and the persistence system is to define repositories (see Figure 22). These are object representations of the underlying database. Rather than working directly with the database, the domain model calls the methods defined by the repository, which in turn makes calls to the database to store and retrieve the model data. This allows us to isolate the model from the implementation of the persistence.[16]

The domain layer defines this separation by declaring five repositories. These repositories are explained below:

Conference The conference repository is used for modifying the conference as a whole. It contains the functionality like managing the chairs and IPC members, swapping all submissions or adding additional submission fields which are needed during submission.

Submission The submissions repository is used when changes only affect one submission. Like adding or deleting additional authors, ad-

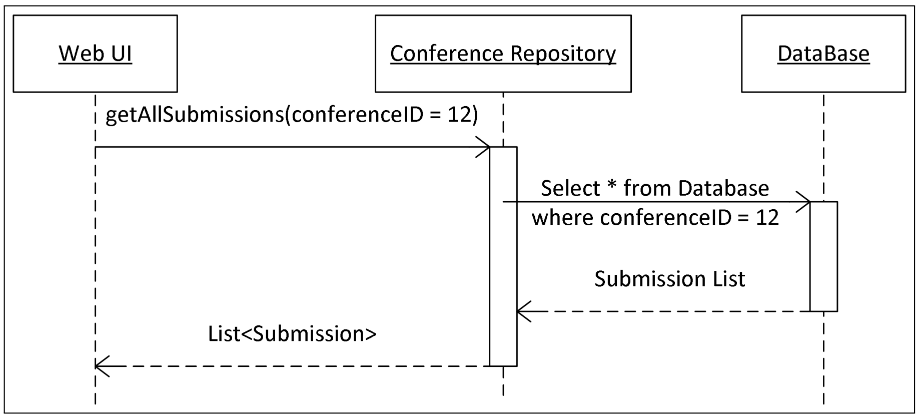


Figure 22: This figure shows the usage of the repository pattern based on an example in the COMFy framework.

ditional content like pdf and zip files, or adding or deleting preferences for the submission

Preference The preference repository is used when changes occur to the areas of expertise, like parsing new areas of expertise. It is also used when getting the areas of expertise for specific users.

Review The review repository contains all functionality about the reviewing. Like creating or modifying a review or creating new reviewer. It is also used for the discussion after the reviewer has created his review.

Profile The profile repository is used when querying the profiles of users. The general case for using this repository is for changing the users profile information. Another example when the profile repository is when for adding authors, reviewers, IPCs or chairs to a submission or a conference.

Nearly all communication between the web UI and the database is done through these five repository. There are some occasions where the repositories are not used. One exception to this rule is when the Roleprovider or the Membershipprovider is accessed. These two providers are generated by the ASP.net framework. The Roleprovider is for security reasons to check if a user is allowed to access certain areas. The Membershipprovider handles password resets or creating and managing a session.

4.5 STATEMACHINE

As mentioned in Section 3.2 COMFy has a state machine to switch the submissions between the different states of the conference. The advantage of this system is that at any given time it is possible to check in which phase a submission currently is. The state of each submission is stored in the database in the CurrentPhase field. A further advantage of this system is that the transition of one phase to another can be handled. For example during the transition of the submission phase to reviewing phase all submitted files will be locked. So if the submission needs a further submission phase the author can't delete his old data. As the state machine is designed in a modular way, it is also simple to add further phases and phase transition.

4.6 COMFY

This is the biggest part of the program. It is designed using the MVC pattern (Section 3.2) and its main purpose is to query the repositories of the domain process. This data is processed in the controller and then provided to the user. It also works the other way around, where it takes the user data, processes it for the repositories and save it there. The other main part of the controller is to ensure that no one can access confidential data or data which a person is not supposed to see. For example the authors of a submission may only be seen by the other authors of the paper and the chair but not the reviewers of this paper, for the double blind reviewing.

COMFy currently has four different controllers.

Home Controller The home controller is responsible for the main screen a user gets to see after he logs into the system. He gets an overview of all of his submissions, reviews and on which conferences he is chair and IPC member.

Account Controller The account controller is used for account functions. The user can change his profile data or reset the password if he has forgotten it. Additionally this controller is used for registering and logging into the system.

Error Controller If the system creates an error or the user requests confidential data he may not access, he gets redirected to the error controller. There he will get notified if he did something wrong or why the system couldn't handle his request.

Conference Controller The conference controller handles every request which concerns a conference or submissions of the conference. This means every request which isn't covered by the previous controller is handled by the conference controller.

4.7 COMFYS API

Occasionally scientist who submitted a paper to a conference, were reviewer or chairs are not satisfied with the functionality, provided informations or other parts of the management system. Some of these scientists also have ideas how to improve the system. To make it easier for them, they might also be willingly to improve the system on their own. The conference providers often don't implement these features because it costs money and the mentioned scientist can't implement the features because providers don't want to disclose their system. COMFy tries to circumvent this problem by providing the user an API.

COMFy was designed as a framework with a complete GUI. Most of the users will use the HTML frontend. If they are satisfied with it, the framework will behave like a normal conference management system. The difference is, when a user is not satisfied with some parts of the framework. If this is the case, every call of COMFy can be used as an API call. This means, it is possible to get the data, when calling an URL parsed in XML or JSON. The user might use this data to create his own representation via multiple requests or just rearrange the layout or even create a widget which checks the server if there are new reviews available.

The big advantage of such an API however is when combined with the dynamic review and submission fields. COMFy provides an evaluation on the rating for every review on each submission. If the chair created special fields in the reviewform or submissionform it is only possible to see these fields when accessing the submission or the review. With the API it is possible to load these and create a own table when accepting or declining the submissions. The advantage is, these tasks are possible without contacting the administrator and without modifying the framework.

4.7.1 *Usage of the API*

COMFy provides three ways of displaying the data. The first possibility is using the HTML. The framework will always output HTML if not specified differently. The other two options are either JSON (Section 3.2) or XML.

There are currently two different ways how to access the data to get XML or JSON in return. The first is to add a data parameter to the request and set this parameter to either JSON or XML. For example the request *http://localhost/COMfy/Conference/Test2013?data=xml*. creates the following XML output Listing 5.

The request *http://localhost/COMfy/Conference/Test2013?data=json* creates the following JSON output Listing 6.

4.7.2 *API classification*

The COMFy API calls can be categorized into four different logical classes.

```xml
<Conference xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http
    ://www.w3.org/2001/XMLSchema-instance">
  <ConferenceId>24</ConferenceId>
  <ShortName>Test2013</ShortName>
  <Name>TestConference for SRMv2</Name>
  <Prefix>paper</Prefix>
  <Description>
  This is the first conference handled with the new COMfy/SRMv2 system.
  </Description>
  <Submission_PublicationLength>8</Submission_PublicationLength>
  <SubmissionDeadline>2012-12-27T10:27:00</SubmissionDeadline>
  <activatedBidding>false</activatedBidding>
  <pDeadline>2013-01-27T10:27:00</pDeadline>
  <sDeadline>2013-01-27T10:27:00</sDeadline>
  <tDeadline>2013-01-27T10:27:00</tDeadline>
</Conference>
```

Listing 5: This listing shows a XML formatted response for a request sent to the COMFy API.

- The UserHome API calls are for calls which are used for retrieving information about the conference and submission of the user within the system. These UserHome API calls can be seen in Table 1

- The account API calls are used for everything related to the account and the system, like registering a new account, logging into the system or changing the profile settings. Some of the Account API calls can be seen in Table 2.

- The conference API calls are used for everything regarding the whole conference. For example viewing or editing the conference, managing the IPC members or manage the fields in one conference. These conference calls are primarily used by the conference chair and the administrator as normal users have no rights to modify anything regarding the conference. Some of the conference calls can be seen in Table 3.

- The submissions API calls are a subcategory of the conference calls. The submission identifier itself is only unique within one conference. This means it needs the conference identifier to access the correct submission. This identifier is set by the system and consists of the defined prefix of the conference and a number which starts at 1000 and increases with every submission. Because of this decision, the identifier is always short and it is possible to see the conference by the means of the URL schema. The submission API calls are used for everything concerning a submission. These include viewing and editing, assigning reviewers or other authors. Some of these API calls can be seen in Table 4.

```
{
   "ConferenceId": 24,
   "ShortName": "Test2013",
   "Name": "TestConference for SRMv2",
   "Prefix": "paper",
   "ImageData": null,
   "Description": "This is the first conference handled with the new
       COMfy/SRMv2 system.",
   "Submission_PublicationLength": 8,
   "ImageMimeType": null,
   "SubmissionDeadline": "/Date(1356600420000)/",
   "AdditionalFields": null,
   "activatedBidding": false,
   "pDeadline": "/Date(1359278820000)/",
   "sDeadline": "/Date(1359278820000)/",
   "tDeadline": "/Date(1359278820000)/",
   "eMail": null

}
```

Listing 6: This listing shows a JSON formatted response for a request sent to the COMFy API.

These tables are only a small overview of all current API calls. A listing of all calls can be found in the Appendix A.

4.7.3 *API example*

As the decision making for hundreds of papers is a tedious task, conferences might have multiple chairs to create the decisions if a paper is accepted or not. In order to allow the chairs to work parallel, COMFy uses the same technique as the old SRM framework. This procedure will be an example how the API can be used.

At first the chairs load all submissions with the current reviews and the current decisions. Then a chair can set a decision by opening the decision form. During opening an AJAX call looks up if there were any changes to the current decision. If there were changes the Domain Object Model and the review form are updated. The chair can decide if he is already satisfied with the decision, if the review form was updated. If he is not, he can set his own decision or close the dialog and start with the next paper.

Once for all papers a decision has been made, the chair can swap all submission to their respective phase. Submissions which need major or minor revisions, are moved to the submission phase, submissions which need further reviewing are moved to the reviewing phase and submissions which are accepted are moved to the CRC phase.

API	Call Type	Description
Home/	GET	information of conferences/-submissions of the user
Home/mySubmissions	GET	submissions where the user is author
Home/myReviews	GET	submission where the user is reviewer
Home/myConferences	GET	conferences where the user is chair
Home/myIPC	GET	conferences where the user is IPC

Table 1: This table shows all home API calls. These calls are for getting the information on the own submissions, chairings, reviewings and conferences where the user is IPC.

4.8 srmv2

As COMFy is an API it encapsulate the full business logic of a conference system. But it does not provide the user a step by step guide, which API calls are possible or should be used when. It would also not be very comfortable for a user to search in over 100 API calls or read the description of them. For this reason SRMv2 was created. It should help the users use COMFy as conference management system.

In Figure 24 all identified use cases of the system are shown. These use cases consists of multiple steps which the API expects to happen in order but can't enforce it on the API layer. The next chapter will illustrate an example of ordered calls using the assign reviewer use case.

4.8.1 *Example use case*

This section will break down one of the use cases of Figure 24 into simple step by step guide. This will try to give an insight why COMFy needs a SRMv2 system and why the API alone is not sufficient. The example will be the *Assign Reviewer* from the chairs perspective.

At first the chair has to select the paper where he wants to add the reviewer. Then the chair has to select the user who will be the reviewer. As there are over 11.000 users in the old system there should also be the possibility to search for the user before selecting him. Once the user is chosen, the chair has to select the role of the reviewer. He might choose between his previously defined reviewer types (see

API Call	Call Type	Description
Account/LogOn	POST	logging into the system
Account/Profile	GET	reading the profile information
Account/Profile	POST	saving new profile information
Account/Register	POST	registering with the system

Table 2: This table shows a small excerpt from all account API calls. They are used for everything in which concerns the account, like registering, loging into the system or changing the profile information.

Section 5.2). Then it is possible to edit the email from a template the user will get when he is invited to review the paper. Once this is done, the chair has to confirm the assignment and send it. (see Figure 25)

All this steps are possible with the API but SRMv2 helps the user to do these necessary steps. As the API is in REST principle (see 4.7) these steps can also be done in one step by confirming the assignment if the user, the Email template, the role and the paper are known. So for a frontend developer it is also possible to change these steps in any way he likes. This order was chosen to help people switch from the SRMv1 system to the new system as the order there was the same.

4.8.2 *Implementation*

SRMv2 can be seen as the view in the model-view-controller in COMFy. It provides the systems procedures in a convenient format for the user. The homescreen which can be seen in figure Figure 26. These tiles guides the user to the associated sub section. These four sections are:

SUBMISSIONS: The submissions section provides the user with all his submitted papers. He gets an overview which phase his papers are and all conferences where he submitted a paper. From there it is possible to navigate to his papers and edit the submission. For example to change the metainformation or upload a revised paper. An example submission screen can be seen on Figure 27.

REVIEWS: This section provides the user with all papers, where he is assigned as a reviewer. The user might accept or decline a review assignment or access a submission from this section. At the submission the reviewer can submit or edit his review.

API Call	Call Type	Description
Conference/EG2012	GET	retrieves information of the Conference EG2012
Conference/EG2012/ EditConference	POST	saves modified conference
Conference/EG2012/ ShowSubmissions	GET	get all submission of EG2012
Conference/EG2012/ manageFieldsForConference	GET	overview of the additional submission fields
Conference/EG2012/ deleteFieldForConference	POST	deletes one of the additional submission fields
Conference/EG2012/ manageIPCMember	GET	overview of IPC members
Conference/EG2012/ addIPCToConference	POST	adds a new IPC member

Table 3: This table shows some conference API calls. These calls are used for managing the conference or displaying information about the conference.

IPC MEMBER: On the IPC member section the user can see all conferences where he is IPC. From this section he can access the conferences bid for the different papers or specify his areas of expertises.

CHAIR: There the user has an overview of all conferences where he is chair. From there he can access the different conferences and manage them.

API Call	Call Type	Description
Conference/EG2012/ Submission/Show/ paper1000	GET	information of submission Paper1000
Conference/EG2012/ Submission/paper1000/ editSubmission	POST	saves the new information
Conference/EG2012/ Submission/paper1000/ assignReviewer	POST	assigns a reviewer
Conference/EG2012/ Submission/paper1000/ removeReviewer	POST	deletes reviewer
Conference/EG2012/ Submission/paper1000/ reviewerDiscussion	GET	access the discussion forum

Table 4: This table shows a small excerpt from the submission API calls. These calls are a subcategory of the conference calls. They are used for all calls which concerns only one submission within one conference like editing the submission, uploading a paper or assigning a reviewer.

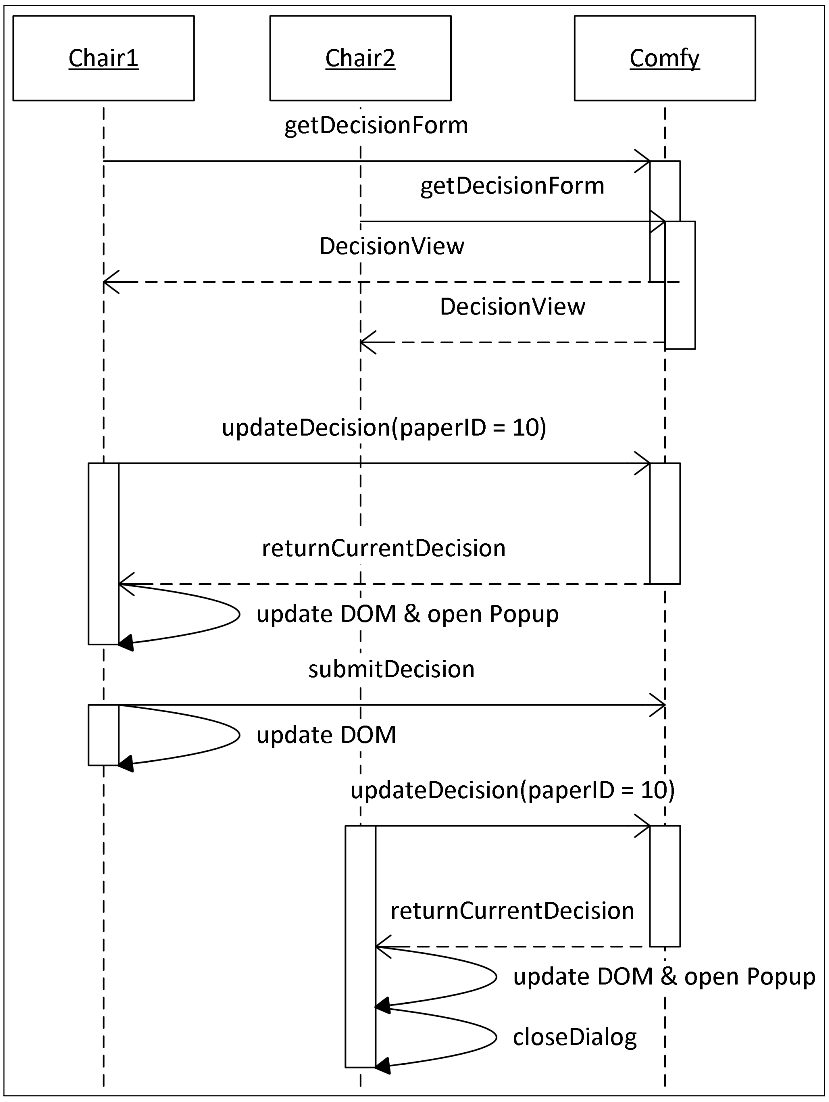


Figure 23: This sequence diagramm shows how two different chairs can work on the same submissions overview formular and how the current decision of the same paper is updated by the use of Ajax to avoid multiple input

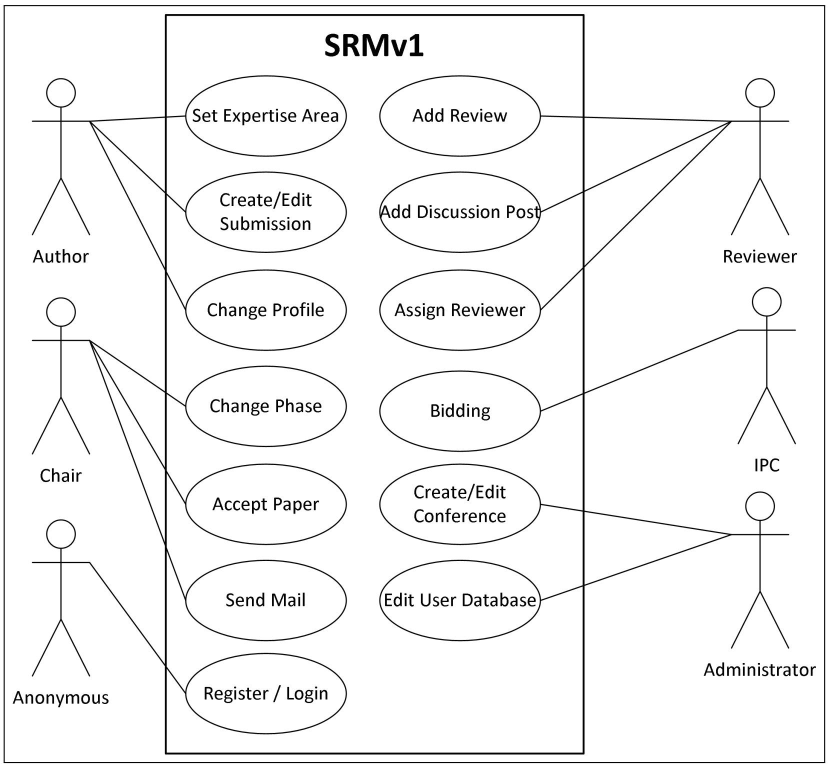


Figure 24: This figure shows all found actors and use cases in the old SRM system

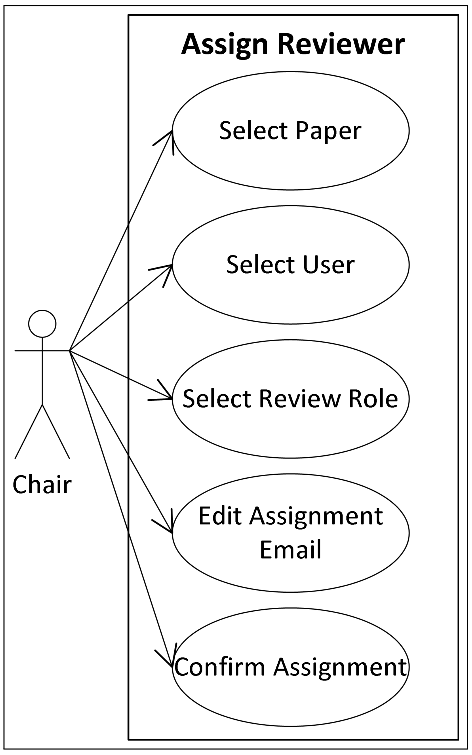
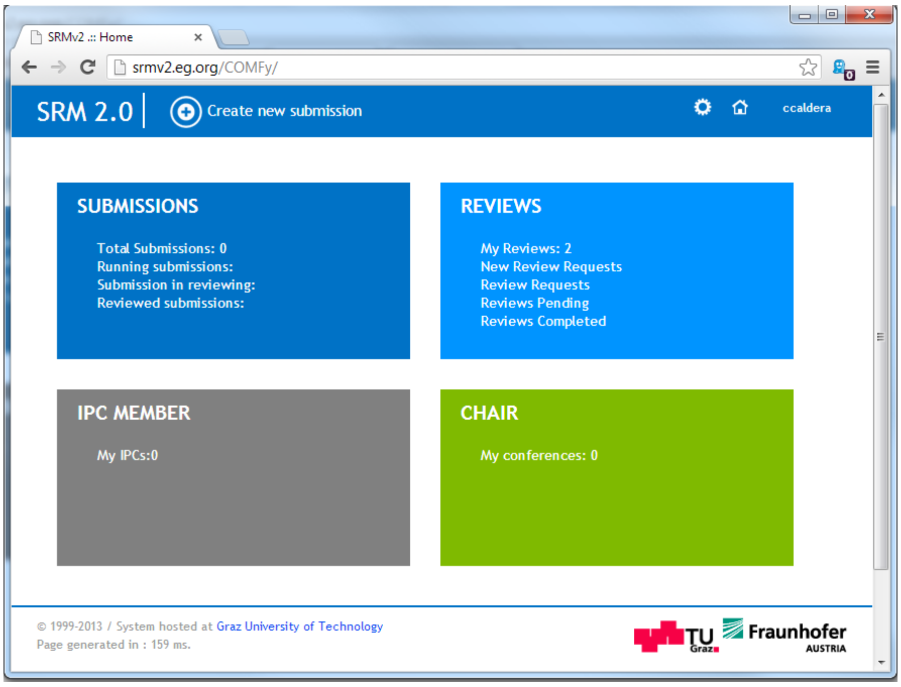


Figure 25: This figure shows the steps necessary for assiging a new reviewer from the perspective of a chair.



SUBMISSIONS

Total Submissions: 0
Running submissions:
Submission in reviewing:
Reviewed submissions:

REVIEWS

My Reviews: 2
New Review Requests
Review Requests
Reviews Pending
Reviews Completed

IPC MEMBER

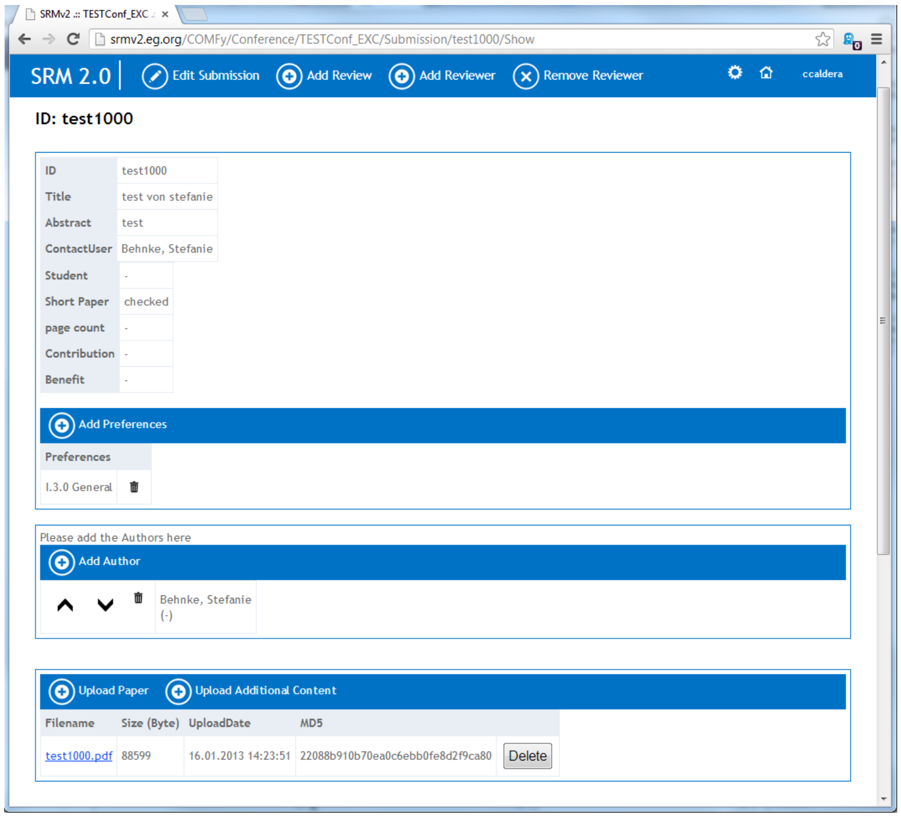
My IPCs:0

CHAIR

My conferences: 0

© 1999-2013 / System hosted at Graz University of Technology
Page generated in : 159 ms.

Figure 26: This figure shows the home screen of SRMv2. Each of the tiles guides the user to the associated section.



ID: test1000

ID	test1000
Title	test von stefanie
Abstract	test
ContactUser	Behnke, Stefanie
Student	-
Short Paper	checked
page count	-
Contribution	-
Benefit	-

Preferences
I.3.0 General

Please add the Authors here

Behnke, Stefanie
(-)

Filename	Size (Byte)	UploadDate	MD5
test1000.pdf	88599	16.01.2013 14:23:51	22088b910b70ea0c6ebb0fe8d2f9ca80

Figure 27: This figure shows a submission screen of SRMv2. From there it is possible to see the metadata, the preferences of the paper, and the paper. If the user is allowed to see the authors they are also displayed on this page.

5 COMFYS WORKFLOW

5.1 OVERVIEW

The previous chapter gave an in depth insight of the design of the technical side of COMFy. This chapter will cover how such a conference runs in the system. Section 5.2 will give an overview, to get a rough idea of how the parts are connected in COMFy and how they interact. Section 5.3 and subsequent sections will cover each part in detail. It will show how some of the internal parts interact with each other and it will explain why some design decisions were made like they are.

5.2 SUMMARIZED WORKFLOW

Through the duration of a conference the submissions pass through specific phases which are shown in Figure 28

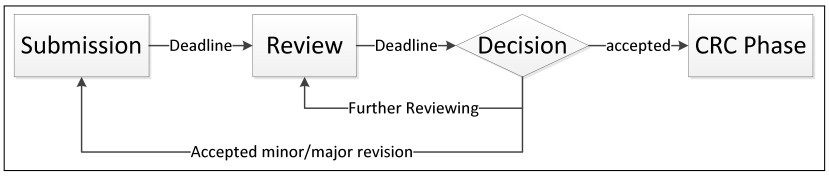


Figure 28: This figure shows the current phases of COMFy.

The first phase is called the submission phase. When the conference is created, the submitted papers will start in this phase. This is also the only phase where the author has still full control over the paper, so he can delete it and upload a new improved version. The system also supports additional zip-compressed files for the submission which might improve the paper with 3D-Models, movies or presentations. Furthermore authors can add other authors to the submission, as well as decide which of the authors should be the person to contact.

When the deadline of the submission phase has passed, COMFy will check every paper for its completeness. If a submission has metadata and an uploaded PDF document it is considered as complete. This submission will then enter the review phase. In this phase the submission won't accept any changes to the submissions by the authors, which means that they can't delete this paper, the zip-files or change the metadata or authors anymore. The chair of the conference still can change the submissions if he needs to. COMFy currently has a sub-phase called bidding in the reviewing phase. In this bidding phase IPC members of the conference can specify which papers they would like to review and which they are not competent enough. Once the bidding is done, a chair can invite reviewers for the submission. These reviewers can then either accept this invitation or decline it. Once they have accepted it, the primary and secondary reviewers can start inviting tertiary reviewers. The system also allows them to view the paper and the additional content of the submission and write a review. When the reviewer has written a full review they get to see a small forum like structure where they can discuss and justify their review.

When the deadline for the reviewers is reached, the chair can decide what will happen next with the submission. It can be accepted, declined, accepted with minor or major revisions or that it might need further reviewing. The authors will be contacted with the chairs decision about the paper. When it was accepted the paper will get to the camera ready copy (CRC) phase. In this phase the authors can upload the CRC version of his paper. When the submission was accepted with minor/major revisions, the submission will enter the submission phase again. There the author can upload a revised paper and content. The old paper and content is still locked for the user. Then the submission again gets into the reviewing phase. In some cases the submission wasn't reviewed within the deadline. Then the chair puts the submission back to the review phase where he can invite other reviewers, or give the original reviewers more time.

5.3 SUBMISSION

5.3.1 *Setting up the conference*

Before anyone can even create a submission, an administrator of COMFy has to create a new event. Upon creation of this event, the administrator has to set a short unique identifier which will then identify this event. The administrator must also add the first responsible chair to this event. After this the chair has full control over this event.

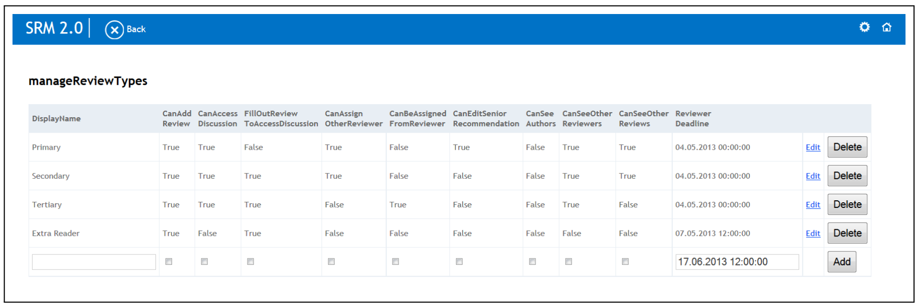
The chair can then modify the conference to his requirements. First of all he might add further chairs to the event, to help him manage the conference. He can also add IPC members to the conference, change the event details and add the areas of expertise for the specific event. This areas will be used by the IPC users and the authors.

The chair might also create additional submission fields like textboxes, dropdown lists or checkboxes which an author has to fill up during his submission. But not also the author can have these special fields, the chair might also create additional review fields a reviewer has to fill up during his review. Further the chair has to create the different reviewer types or use the default ones (Figure 29). There are currently eleven different access settings a reviewer can have:

Display name:	Decides how the reviewer type is called in this conference.
Can add review:	The reviewer can add a reviews to the the submission. These reviews can be seen by the chair and influence his decision if the paper will be accepted.
Can access discussion:	Decides if the reviewer can see the discussion after the deadline reviewing deadline. There the reviewers and the chair can discuss the submission and eliminate uncertainties.
l out review to access discussion:	The reviewer has to fill out his review before he can access the discussion. This prevents the reviewer to get influenced by other reviewers. A primary reviewer, who does not need to write his own review might not need this limitation.
Can assign other reviewer:	Means that this reviewer type can add further reviewers to the submission. These reviewers are informed about the invitation and can accept or decline the request.
Can be assigned from reviewer:	These review types can be added from the reviewer. This prevents a main reviewer to add further main reviewer to a submission.
Can edit senior recommendation:	This field decides if the reviewer can edit the recommendation if the papers is accepted or not. This recommendation will be seen by the chair.
Can see authors:	This field decides if the reviewer does a single- or double blinded reviewing.
Can see other reviewers:	These review types can see the real names of other reviewers. Especially in the discussion where it is either anonymously or the reviewers can see each other.
Can see other reviews:	Decides if a reviewer type can see the other reviews. The primary reviewer can give his recommendation based on the reviews of the other users.
Reviewer deadline:	Decides when a reviewer can't add or edit his own review any more.

5.3.2 *Submission for authors*

Once this conference is created, a user of the system is allowed to create a submission for this event. Upon creation of the submission a user has to fill out the basic fields for a conference and the additional fields the chair set for this specific event. Of course it is possible for the user to modify the submission afterwards. Furthermore the author can add other authors to its submission, define the areas of expertise for the submission and upload zip and pdf files. These added



manageReviewTypes

DisplayName	CanAdd Review	CanAccess Discussion	FillOutReview ToAccessDiscussion	CanAssign OtherReviewer	CanBeAssigned FromReviewer	CanEditSenior Recommendation	CanSee Authors	CanSeeOther Reviewers	CanSeeOther Reviews	Reviewer Deadline		
Primary	True	True	False	True	False	True	False	True	True	04.05.2013 00:00:00	Edit	Delete
Secondary	True	True	True	True	False	False	False	True	True	04.05.2013 00:00:00	Edit	Delete
Tertiary	True	True	True	False	True	False	False	True	False	04.05.2013 00:00:00	Edit	Delete
Extra Reader	True	False	True	False	False	False	False	False	False	07.05.2013 12:00:00	Edit	Delete
	☐	☐	☐	☐	☐	☐	☐	☐	☐	17.06.2013 12:00:00		Add

Figure 29: This figure shows the managment console for the reviewing types. Here it is possible to add, modify and delete different types of reviewers which are used for adding a reviewer.

authors get the same user rights as the creator of the submission. Further it is possible to sort and order the authors in accordance with the submitted paper.

5.3.3 *Submission for IPC*

During the submission or reviewing phase a chair can activate the bidding process. When the bidding is activated every IPC member must complete the following 3 steps.

Set areas of expertise At first the IPC has to set the expertise level in the areas which are defined by the chair. He can choose from either being *expert*, *has knowledge*, *passing* or *no knowledge* in every specific area.

set conflicts To avoid that a person will review a paper when he knows the author, the reviewer gets a list of all authors from all submissions. This conflict will remain for all future events for these two persons.

bidding When the IPC member has completed the previous steps, he can bid for the papers. During bidding the user has three possible ways to sort these papers to get a better overall view of the papers. Either he can sort it after the paper ID or by the areas of expertises or by his preferences of the expertise level. He can then specify for each paper whether he would like to review it, if he could review it or if he is not competent enough to review it.

5.3.4 *Submission for chair*

During the submission phase a chair can keep the overview of all submissions and which IPC already completed his bidding.

5.4 reviewing

5.4.1 *Pre reviewing*

Once the deadline is expired, the authors can't modify or change their submissions and papers anymore. The chair is still able to upload or modify the submission or papers if he wishes to do it. But before the reviewing can start, the chair has to prepare it to customize it to his desires. The chair still can add or modify his own review fields, which reviewers have to fill up for the review.

Then the chair activates the the reviewing phase. By doing so COMFy transfers all submissions into the reviewing phase with its state machine. To learn more about COMFys state machine see Section 3.2. To simplify this process for the chair COMFy uses an automatic suggestion system to make the decision for the chair easier. During this transition all papers of all submissions, which get to the reviewing, will be hard locked. This means, that when the deadline is expended for minor or major revisions and the submission returns to the submission phase see Figure 28, the author still can't delete his old submission. This guarantees that the previous submitted files wont be lost.

5.4.2 *Assigning reviewers*

The next step is the assignment of the reviewers. COMFy uses a specific algorithm to create a good matching for the reviewers. Before this matching is taken live it will be presented to the chair as a suggestion. The chair can modify the distribution or restart it to create a new matching. In order to create this matching 3 different values are taken into account (Equation (1)).

$$IPCsuggestion = aoeRating + biddingRating + external\ IPC\ score \tag{1}$$

The first value is created out of the areas of expertise of the paper and the IPC. The formula can be seen in Equation (2).

$$aoeRating = 25 \cdot \frac{4 \cdot \#\{expMatch\} + 2 \cdot \#\{knowMatch\} + \#\{passMatch\}}{\#\{Area\quad of\quad expertises\quad of\quad Paper\}} \tag{2}$$

The expMatch is the amount of areas of expertise which the paper has and the IPC member is expert in. KnowMatch is the amount where the IPC member has knowledge in and passMatch where the IPC has only passing knowledge. This formula guarantees a value between 0 and 100. So if a IPC is expert in every area, then knowMatch and passMatch will be zero and expMatch will cancel out with the amount of area of expertise.

The second value comes from bidding. If a user wants to review a paper he will get 100 points, if he could review a paper he gets 80 points. These values and the formula have been tried and tested with the old SRM system and are proven to generate good results.

$$\mathtt{biddingRating} = \begin{cases} 100 & \mathtt{Want\ Review} \\ 70 & \mathtt{Could\ Review} \\ 0 & \mathtt{no\ Expertise} \end{cases} \tag{3}$$

COMFy however has a third value, the external IPC score (Equation (4)). This value will be used in the future. A program will get all IPC members and search for their papers in old conferences. This program will then process the papers with natural language processing. These processed papers are then matched to the natural language process papers in the conference. So every IPC gets matching points to the papers in the conference. The external IPC score then serve for a weighted assignment for reviewing.

$$\mathtt{external\ IPC\ score} = \begin{cases} 0 & \mathtt{default} \\ x & \mathtt{external\ Injected} \end{cases} \tag{4}$$

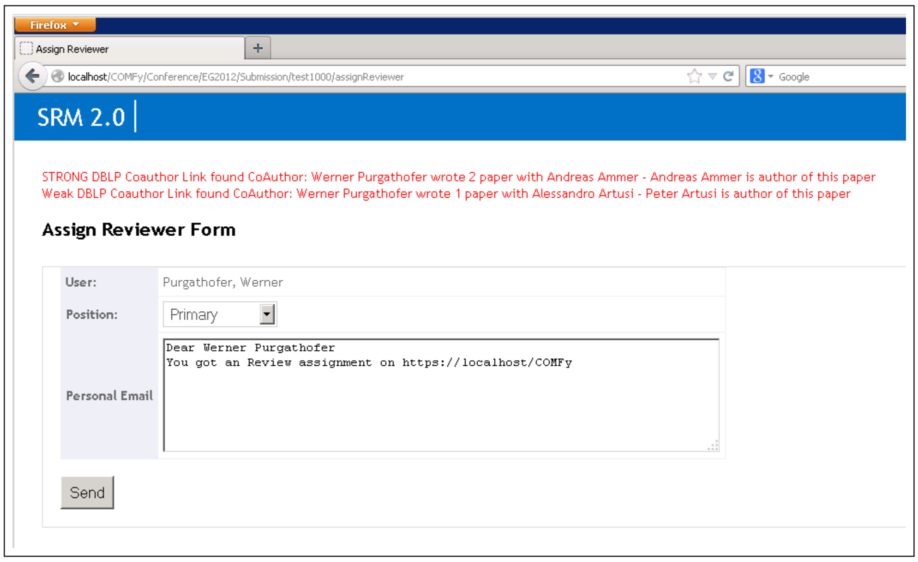
The assignment of the papers to the IPCs is a modified stable matching problem which is a np-complete problem[14]. In COMFy the best matched reviewer is assigned to a paper until he is assigned to more than the average workload of reviews then the second best matched reviewer is assigned.

Once this suggestion is created, the chair modify the result. Once he is satisfied with the suggestion he can take the assignments live. This means that all IPCs gets a request to review the paper in assigned reviewer role. It is also possible that the chair manually invites users to review the paper.

5.4.3 *Reviewing phase*

Once a user or IPC member gets a review invitation, he will get notified that there are new review requests. If the request gets accepted, the reviewer gets the authorization to view the submission and download the paper. When the reviewer finished reading the paper, he can add a review to the submission.

If the review role allows to add further reviewers. It is possible for them to add them in this phase. For these invited user the reviewing phase starts again at Section 5.4.3. Before the chair or the reviewer who assigns the new reviewer finished the assignment, the assigned reviewer and the authors are checked if there is a coauthorship between these users. This is done by using the Digital Bibliography & Library Project (DBLP) API interface. If such a coauthorship is found, the current user is warned about the authors (see Figure 30). This warning might be ignored from the user, if the coauthorship is not correct.[2].


Firefox
Assign Reviewer
localhost/COMFy/Conference/EG2012/Submission/test1000/assignReviewer
Google
SRM 2.0

STRONG DBLP Coauthor Link found CoAuthor: Werner Purgathofer wrote 2 paper with Andreas Ammer - Andreas Ammer is author of this paper
Weak DBLP Coauthor Link found CoAuthor: Werner Purgathofer wrote 1 paper with Alessandro Artusi - Peter Artusi is author of this paper

Assign Reviewer Form

User:	Purgathofer, Werner
Position:	Primary
Personal Email	Dear Werner Purgathofer You got an Review assignment on https://localhost/COMFy

Send

Figure 30: This figure shows a warning of a found DBLP coauthorship between two authors and the assigned reviewer from the chairs perspective.

During the review phase the chair can again overview all reviews, how much reviews every submission has, or view the discussion. He can also intervene if something doesn't go according to plan within one of the submissions.

5.5 DECISION

As seen in Figure 28 the decision is not really a phase like submission or review phase as it doesn't have a deadline. But the decision processes is one of the core process in a conference. There the chair has to decide what happens with every submitted paper. The chair typically starts to work on the papers after the review deadline. There he decides whether a paper gets accepted or declined. Some of the papers might also need a minor or major revision, or if they require a further review cycle. The submitted reviews are assisting the chair in his decision. It is also possible for the chair to ask the reviewers in a discussion form for further opinions on one paper. The reviewers, which review roles has access to this discussion forum, can visit this forum and answer the question or exchange opinions with other reviewers. For every submission there is such a specific forum. They have a directory like structure so it is possible to answer previous postings or create a new entry post.

5.6 CRC

The CRC (camera ready copy) is the final phase. There the authors can upload their final document which contains their names and the complete references. Again during the CRC phase the chair can check, which of the submissions already have been uploaded. Once the CRC papers are submitted they can be printed in the conference proceedings.

6 DISCUSSION

This chapter will give an overview of the thesis and the project. It compares COMFy with other conference systems out there and especially with the old SRM system. It will show where it needs polishing in order to improve the user experience for the targeted user community.

6.1 COMPARISON OF THE RESULTS

Upon creation of this document the framework wasn't used by a conference. Therefore it still has to show how it behaves and perform under real conditions. The test results however are promising. In order to create a possible real environment, all users of the old system were exported from the old system and imported into COMFy. Once all 11.000 users were imported to the system some performance checks were done on the user data table like searching users. Whereas the old SRM system always took up several seconds to finish the request, COMFy consistently finished the requests in under 500 milliseconds. The reason for this huge difference is the underlying storage layer of SRM. As said in Section 1.1, SRM is based on Hyperwave.

6.2 MODULARITY

A major challenge in current applications is to enhance the user experience by responding to user requests and implement them into the system. This section should show how easy it is, to modify COMFy or when someone wants to add features to it.

6.2.1 *Modifying the database*

All database tables are mapped via LINQ to SQL in entity classes in the domain. The general rule is, if there is avalue to be added to the database table it is enough to add the associated value in the entity.

There are two exceptions to this rule. The first is when modifying the dynamic additional submissions fields or additional review fields table. These tables are selected by outer joins and LINQ still has some problems with integers, null values and outer joins. So it is necessary if adding values to these two tables to also add the parsing in either the SQLSubmissions- or the SQLReviewRepository outer join function. The second exception is the profile and submission table. These two tables are not directly connected to the table itself, instead they are connected to a view. The reason for the profile to be connected to a view is the e-mail field. As this field is managed by the ASP.net framework and to obtain data consistency it is selected in the view and handled by the remaining framework as a profile field. So it is possible to edit the e-mail address with either the membership provider or the profile repository. The reason for the submission to be connected to the view is its submission short name. This short name is used in the whole framework as an identifier. Although it doesn't exist in the database. The short name is a concatenation of the prefix of the conference and the paperID of the paper itself. This paperID is unique within the conference but not over multiple conferences. So only the conference and the paperID generate a unique identifier for the submission. If a value is added to the submission or the profile table, the value should also be added to the related view.

When modifying or deleting a value from the database it is enough to modify or delete the associated value in the entity field and check and update all references

6.2.2 *Modifying the repositories*

The repositories in the domain are basically a collection of select, insert, update and delete commands in LINQ to SQL. So it queries the database and returns the value to the controller.

Adding a new command is very simple as it only needs a further entry in the interface and the implementation. Then the new command is available in every controller the interface is included.

When modifying or deleting an existing repository method, it only needs to be checked if the function itself works after it has been modified and if all references still fully functional. Upon deleting all references should be deleted as well.

6.2.3 *Modifying the WebUI*

Most of the time there are changes to the conference controller, because this module is the core of the application.

Before modifying the controller, some basic knowledge how the related action is connected to the controller is required. The URI schema of COMFy for conferences follows a simple principle. If a action is related to a conference the following URL is used: `.../Conference/{confShortName}/action`

If the action relates to a submission this URL is used: `.../Conference/{confShortName}/Submission/{submission}/action`

The action defines which method is called during a request of the user. All current actions can be looked up in the Appendix A. So if there is the need for a new action or view or API call, just add a new action to the controller in the appropriate region. It is simple to find the right action with this schema. Then the controller parses the {confShortName} and the {submission} from the request in their respective strings.

Then all actions follow a simple step by step guide.

- At first the user is checked if he is authorized in the system.
- If he isn't authorized he is redirected to the login page. If he is, the needed data is fetched from the repositories.
- Then it will be checked if the user is allowed to perform the requested task. This is done after the fetching because for the checks data from the database is needed.
- If the user is not allowed he will get redirected to an action which he is allowed to see. If he is allowed, the task like saving, fetching more data or deletion is performed.
- After the task is done the user gets his corresponding answer from the server.

6.3 FUTURE WORK

One of the major points which will greatly increase the user experience, is to use the programs scores feature. The idea is, that once all papers are submitted, a program will scan them with natural language processing. The same program can then scan over some papers of every IPC members and then create matchings between the publications of the IPCs and the papers of the authors. Then the IPC of the processed paper can be assigned to review the paper according to the previously found matching. COMFy's API currently supports both cases. Either mentioned program can ingest only matching values of an IPC member to a submission. So it takes the matching value into account. During the suggestion creation process, this program can ingest the suggestion itself. This way it will skip the suggestion creation process. By using the first case it is also possible to modify the weight value of the matching values as the internal program can give any values from 0 to 200. If the matching values contain higher values they are automatically weighted higher and therefore more likely to be used during assignment. Once this process is fully functional the areas of expertise and the whole bidding system will then be deprecated as a program searches for areas instead of the user to add himself to his paper. The authors don't need to add preferences to their submissions as well as the chair, who doesn't need to input any preferences. The most time however will be saved for the IPC members, as they don't need to answer which areas they are expert in as well as they don't need to bid on every paper, if they want to review it or not.

Until now no conference was using the system. In the small test conferences which were managed with the system there were no major difficulties. But as practice shows, real world systems often behave different than test systems. And once some small conferences are organized with the new framework it will turn out were some parts needs improvement. Then, although performance was kept in mind during creation of COMFy there might be some cases which need further tweaking. As opposed to COMFy in SRMv1 every possibility was taken into consideration to get a further speed up. So there might also be some places where COMFy needs tweaks to further increase the speed of the requests.

If a users queries COMFy often for one result it is also possible to create a further API call to offer this data in one request. This might be necessary if these request requires much server capacity.

6.4 CONCLUSION

In this paper a new conference management system was introduced. The main feature which distinguishes COMFy from other conference systems is its API functionality (Section 4.7). Every requested page can also be treated as an API call. This provides the user with a lot of flexibility if he wants to configure the interface by himself. If the user only wants to use the system in its standard configuration it is possible to use SRMv2 (Section 4.8).

A further feature of the system are its dynamic reviewing and submission fields. This feature fits very well with the API. As someone can adjust the conference to his own needs without contacting the administrator by usage of the dynamic fields and its API. It is even possible to use COMFy for other workflows where user generated content has to be reviewed like books, designs or journals.

COMFy still has to prove itself under real life conditions. It is the question if and how users will accept a new system and how much users will use the API. Currently the first small conference is held on the new system. There it will show how users are accepting the new system. The current feedback is however primarily positive and the acceptance for the new system looks great.

BIBLIOGRAPHY

[1] Benos, Dale J. ; Bashari, Edlira ; Chaves, Jose M. ; Gaggar, Amit ; Kapoor, Niren ; LaFrance, Martin ; Mans, Robert ; Mayhew, David ; McGowan, Sara ; Polter, Abigail ; Qadri, Yawar ; Sarfare, Shanta ; Schultz, Kevin ; Splittgerber, Ryan ; Stephenson, Jason ; Tower, Cristy ; Walton, R. G. ; Zotov, Alexander: The ups and downs of peer review. In: *Advances in Physiology Education* 31 (2007), Juni, Nr. 2, S. 145–152. – URL http://dx.doi.org/10.1152/advan.00104.2006

[2] Christian, Caldera ; Berndt, René ; Fellner, Dieter W.: *COMFy - A Conference Management Framework*. – 2013 in press

[3] Computing Machinery, Inc. The Association for: *CCS 2012 - Table of Contents*. April 2013. – URL http://dl.acm.org/ccs_flat.cfm

[4] Connolly, Dan: *Hypertext Transfer Protocol – HTTP/1.1*. September 2004. – URL http://www.w3.org/Protocols/rfc2616/rfc2616.html

[5] Crockford, Douglas: The application/json Media Type for JavaScript Object Notation (JSON) / IETF. 7 2006 (4627). – RFC

[6] Fellner, Dieter W. ; Zens, Marco: *Managing Conference Proceedings TUBSCG-2001-02*. 2001

[7] Foundation, The jQuery: *jQuery*. 2013. – URL http://jquery.com. – [Online accessed 19-February-2013]

[8] Garret, Jesse J.: *Ajax: A New Approach to Web Applications*. Februar 2005. – URL http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

[9] Goodman, D.: *JavaScript bible*. Hungry Minds, 2001 (Bible Series). – URL http://books.google.com.au/books?id=46MwAQAAMAAJ. – ISBN 9780764533426

[10] Group, Zakon: *OpenConf*. Januar 2013. – URL http://www.openconf.com

[11] Mandl, Marianne: *Conference Management System (COMS)*. Dezember 2012. – URL http://www.conference-service.com

[12] Papagelis, Manos ; Prof. Plexousakis, Dimitris: *Confious*. Dezember 2012. – URL http://www.confious.com

[13] Programmeerkunde, Laboratorium V. ; Goderis, Sofie ; Goderis, C S.: *On the Separation of User Interface Concerns A Programmer's Perspective on the Modularisation of User Interface Code Proefschrift voorgelegd voor het behalen van de graad van Doctor in de Wetenschappen*. 2007

[14] Ronn, Eytan: NP-complete stable matching problems. In: *J. Algorithms* 11 (1990), Mai, Nr. 2, S. 285–304. – URL http://dx.doi.org/10.1016/0196-6774(90)90007-2. – ISSN 0196-6774

[15] Sanderson, Steven: *Pro ASP.NET MVC 2 Framework*. Second Edition. Apress, 2010. – ISBN 9781430228868

[16] Sanderson, Steven: *Pro ASP.NET MVC 3 Framework*. Third Edition. Apress, 2011. – ISBN 9781430234043

[17] Schwarz, Niko ; Lungu, Mircea ; Nierstrasz, Oscar: Seuss: Decoupling responsibilities from static methods for fine-grained configurability. In: *Journal of Object Technology* 11 (2012), Nr. 1, S. 1–23

[18] Sebastion, Jacob: *An Introduction to SQL Server FileStream*. August 2009. – URL http://www.simple-talk.com/sql/learn-sql-server/an-introduction-to-sql-server-filestream

[19] Smith, Richard: Peer review: a flawed process at the heart of science and journals. In: *JRSM* 99 (2006)

[20] SpryMedia: *DataTables*. 2013. – URL http://www.datatables.net. – [Online accessed 18-February-2013]

[21] Sumann, Florian: *AJAX4SRM - Asynchroner Datentransfer für Submission & Review Management*, Technische Universität Graz, Diplomarbeit, 2011

[22] Voronkov, Andrei: *Easy Chair Conference System*. Dezember 2012. – URL http://www.easychair.org

[23] Ware, Mark: Peer review: benefits, perceptions and alternatives / Publishing Research Consortium. Publishing Research Consortium, 2008 (PRC Summary Papers 4). – Forschungsbericht. – URL http://www.publishingresearch.net/PeerReview.htm

[24] Wenz, Christian: *JavaScript und AJAX*. 7., aktualisierte Auflage. Bonn : Galileo Computing, 2007. – URL http://www.galileocomputing.de/openbook/javascript_ajax/

[25] Wright, David R.: *Finite State Machines*. 2005

[26] Zens, Marco: *Creation, Management and Publication of Digital Documents using Standard Components on the Internet*, Technische Universität Braunschweig, Dissertation, 2004

A APPENDICES

This section will give an overview of all API calls which currently exists in COMFy

Account::LogOn

The get function for the Login of the program

Account::LogOn

the post function to log into the program
model (SRMv2.WebUI.Models.LogOnModel): contains the username and the password
returnUrl (System.String): to this url he will get redirected when the user is logged in

Account::LogOff

The get function to log out of the program

Account::ViewUserDb

The usertables for the administrator to edit or delete users

Account::ViewUserDb

the post function for the administrator to edit/delete users
pressedButton (System.String): "delete" = delete the user, "edit" = go to the admin edit user profile form
userToAdd (System.String): The userId of the user the admin wants to delete or edit

Account::AdminEditProfile

the get function for the administrator to edit the user
userToAdd (System.String): the userid of the person who

Account::AdminEditProfile

the post function for the administrator to edit the user
model (SRMv2.Domain.Entities.Profile): the userprofile of the user

Account::AjaxAddUser

The get ajax call to add a new user to anywhere (add reviewer, add chair, add author) in the program. This function returns a userlist. It is based on the jquery plugin DataTables (http://datatables.net/)
param (SRMv2.WebUI.Models.jQueryDataTableParamModel): The parameters for the datatable, see their homepage for further information

Account::Register

the get function for a new user to register with the system

Account::Register

the post function for the user to register with the system
model (SRMv2.WebUI.Models.RegisterModel): the profile of the user who should be added

Account::CaptchaImage

Returns a captcha immage for the user to enter during registering. Code from: http://www.stefanprodan.eu/2012/01/user-friendly-captcha-for-asp-net-mvc/
prefix (System.String): If the captcha is needed anywhere else, it can get an additional prefix to save it on the serversession
noisy (System.Boolean): if noise should be added

Account::checkEmail

email check from http://haacked.com/archive/2007/08/21/i-knew-how-to-validate-an-email-address-until-i.aspx
eMail (System.String):

Account::ChangePassword

The get function to change a userpassword

Account::ChangePassword

the post function to change a userpassword
model (SRMv2.WebUI.Models.ChangePasswordModel): the new and old passwords

Account::ChangePasswordSuccess

the view function to show the user, that his password was changed

Account::ShowUser

This function takes the userId and queries DBLP and Mendeley APIs and return the Coauthors and the papers writtten by this person
userId (System.String): The userId of the person

Account::RecoverPassword

the get function to recover a password if the parameters are given, this get function also gets called to activate the new password
password (System.String): the new password to activate it
email (System.String): the email person of the user which password should be activated

Account::RecoverPassword

the post function to recover a pasword
model (SRMv2.WebUI.Models.RecoverPasswordModel): contains the email where the new password will be sent

Account::AcceptReviewAgreement

the get function to accept the review agreement. This should be accepted before a user can review papers

Account::AcceptReviewAgreement

the post function for the review agreement. Once agreed, he may review papers
model (SRMv2.WebUI.Models.AcceptReviewAgreement): The model contains the userId and if he accepts the reviewagreement

Account::AdminLogin

The get function for the admin to login as other user. The admin needs his adminusername, password and the username of the user he want to login to

Account::AdminLogin

The post function for the admin to login as other user.
model (SRMv2.WebUI.Models.AdminLogOnModel): contains the adminusername, password and the username where the admin wants to login to
returnUrl (System.String): to this url he will get redirected when the user is logged in

Account::Profile

The get function for the user to change and watch his profile

Account::Profile

The post function for a user to change his profile
model (SRMv2.Domain.Entities.Profile): The new profile model of the user

Conference::ShowSubmissions

Shows all submissions for a specific conference.
confShortName (System.String): The unique conference ShortName

Conference::Show

Shows all the data of one specific submission
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId

Conference::getAuthorsOrdered

returns the authors for a submission ordered in an ordered list
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId

Conference::addSubmission

the get function for adding a new submission to a conference
confShortName (System.String): The unique conference ShortName

Conference::addSubmission

the post function for adding a new submission
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.EditSubmissionModel): The model for the submission without the additional fields
collection (System.Web.Mvc.FormCollection): As we can't create beautiful models for the dynamic fields, we have to parse them from the FormCollection At first the additional fields are selected from the database and then the data is parsed from the collection

Conference::editSubmission

the get function for editing a submission
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId

Conference::editSubmission

the post function for editing a submission
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.EditSubmissionModel): The submission model without the additional fields
collection (System.Web.Mvc.FormCollection): As we can't create beautiful models for the dynamic fields, we have to parse them from the FormCollection At first the additional fields are selected from the database and then the data is parsed from the collection

Conference::deleteUndeleteSubmissions

show all submission (even the submissions in the recycle bin) to delete/undelete them - Submissions are never deleted during a conference phase only recycled
confShortName (System.String): The unique conference ShortName

Conference::deleteUndeleteSubmissions

The post function to delete/undelete submissions - only accessible for the chair and the admin
confShortName (System.String): The unique conference ShortName
collection (System.Web.Mvc.FormCollection): The checkboxes for the submissions - if they are checked they are deleted

Conference::swapPhaseSubmission

The get function to swap one Submission into another phase
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId

Conference::swapPhaseSubmission

the post function to swap one submission into another phase
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId
model (SRMv2.WebUI.Models.PhaseSwapModel): the phase model, which contains the current phase and the swapcommand

Conference::parseFieldListForAddSubmission

private parser which parses the fields when adding a new submission
conferenceId (System.Int32): the conferenceId for which conference the fields are parsed

Conference::saveAdditionalFieldsFromFormCollection

parses and saves the data from the formcollection see add/editsubmission for further details
collection (System.Web.Mvc.FormCollection): the formcollection which we got from add/edit submission
submissionId (System.Int32): the submissionId for which submission the fields are parsed

Conference::addPreferencesToSubmission

the get function for adding preferences to a submission
confShortName (System.String): The unique conference ShortName

submission (System.String): The ShortName for the submission - generated from conference prefix + paperId

Conference::addPreferencesToSubmission

the post function for adding preferences to a submissions, at first all previous preferences are deleted then the checked are added
model (SRMv2.WebUI.Models.ListPreferenceModel): which preferences are checked
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId

Conference::DeletePreferenceFromSubmission

deletes one specific preference from a submission
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId
preferenceId (System.String): the preferenceid which should get deleted from the submissiojn

Conference::addAuthorToSubmission

a function which returns all authors, currently not used - see ajax.adduser
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId
searchValue (System.String): a searchvalue to search for specific users

Conference::addAuthorToSubmission

the post function to add a author to a submission
model (SRMv2.WebUI.Models.AddAuthorToSubmissionModel): the shortname from the submission from the model is needed
confShortName (System.String): The unique conference ShortName
userToAdd (System.String): The Guid of the user which should be added
searchValue (System.String): a searchvalue, currently not needed, only if ajax.add

Conference::MoveAuthorOrder

post function used for moving the authors in their order of appearance

confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId
Author (System.String): The Guid of the person which should be moved
OrderUp (System.String): true if he should be moved up, false if down

Conference::DeleteAuthorFromSubmission

post function used to delete an author form a submission
confShortName (System.String): The unique conference ShortName
author (System.String): The Guid of the person which should be moved
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId

Conference::addContentToSubmission

get function used to add a zip file to a submission
confShortName (System.String): The unique conference ShortName
submission (System.String): ShortName for the submission - generated from conference prefix + paperId

Conference::addContentToSubmission

Post function to add zip file to a submission
model (SRMv2.WebUI.Models.AddContentToSubmission): Currently used only for the submission- and conference short name
content (System.Web.HttpPostedFileBase): the uploaded file of the client

Conference::addPaperToSubmission

get function used to add a pdf file to a submission
confShortName (System.String): The unique conference ShortName
submission (System.String): ShortName for the submission - generated from conference prefix + paperId

Conference::addPaperToSubmission

Post function to add pdf file to a submission
model (SRMv2.WebUI.Models.AddContentToSubmission): Currently used only for the submission- and conference short name
content (System.Web.HttpPostedFileBase): the uploaded file of the client

Conference::DeleteContentFromSubmission

the post function to delete content from the submission (pdf files can only be replaced by uploading a new pdf - chairs/admin overrule this restriction)
confShortName (System.String): The unique conference ShortName
submission (System.String): ShortName for the submission - generated from conference prefix + paperId
fileID (System.Int32): the file id which should be deleted

Conference::GetFile

returns the bytestream when the file should be downloaded
confShortName (System.String): The unique conference ShortName
submission (System.String): ShortName for the submission - generated from conference prefix + paperId
fileID (System.Int32): The requested file ID

Conference::manageFieldsForConference

get function to create/delete the additional fields for the conferences submissions
confShortName (System.String): The unique conference ShortName

Conference::manageFieldsForConference

post for adding a new additional field to the conferences submissions
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.ConferenceFieldsModel): The model from the additional conference Field

Conference::editFieldForConference

This function is used for editing Fields additional submission fields in the conference
confShortName (System.String): The unique conference ShortName
fieldId (System.String): The Id of the field which should be edited

Conference::editFieldForConference

The post function for the editing of a additional submission field of the conference
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.AddConfFieldsModel): Contains the new

data of the additional submission field

Conference::deleteFieldForConference

post function used to delete a additional field for the conferences submissions
confShortName (System.String): The unique conference ShortName
fieldId (System.String): The field id which should be deleted

Conference::manageFiles

This get function Lists all additional Files which are connected to a conference. It is possible to add/delete files like conference templates or other bigger files in this function. All files are stored via Filestream on the filesystem.
confShortName (System.String): The unique conference ShortName

Conference::addFileToConference

This post function is used for adding Files to the conference system. They are stored via Filestream on the filesystem
model (SRMv2.WebUI.Models.FilesManagementModel): The model contains the confShortName
content (System.Web.HttpPostedFileBase): Contains the uploaded file

Conference::GetConferenceFile

This function returns a certain File with the fileID from a conference.
confShortName (System.String): The unique conference ShortName
fileID (System.Int32): The fileId of the file which should be returned

Conference::deleteConferenceFile

This function deletes a file with the fileId from a conference.
confShortName (System.String): The unique conference ShortName
fileID (System.Int32): The fileId of the file which should be deleted

Conference::Index

te get function to show a specific conference
confShortName (System.String): The unique conference ShortName

Conference::addNewConference

get function to create a new conference - only accessible by administrators

Conference::addNewConference

the post function to create a new conference
model (SRMv2.Domain.Entities.Conference): the conferencemodel
image (System.Web.HttpPostedFileBase): the image for the conference

Conference::editConference

the get function to edit the conference - may be used by chairs too
confShortName (System.String): The unique conference ShortName

Conference::editConference

the post function to edit the conferences
confShortName (System.String): The unique conference ShortName
model (SRMv2.Domain.Entities.Conference): the model for the conference
image (System.Web.HttpPostedFileBase): the image for the conference

Conference::Instruction

the post function to edit the conferences
confShortName (System.String): The unique conference ShortName

Conference::DeleteConference

the post function used to delete a conference when its finished - only accessible by administrator
confShortName (System.String): The unique conference ShortName

Conference::addChairToConference

a get function used to add chairs to the conference - replaced by datatables
confShortName (System.String): The unique conference ShortName
searchValue (System.String): used to search for persons

Conference::addChairToConference

the post function to add a chair to the conference
model (SRMv2.WebUI.Models.AddUserToConferenceModel): the model which contains the conferenceShortName
userToAdd (System.String): The Guid of the user, who should be added
searchValue (System.String): an optional searchvalue - replaced by datatables

Conference::viewChairs

the get function to show all chairs and a remove mask for the admin
confShortName (System.String): The unique conference ShortName

Conference::removeChairs

The post function to remove one chair - only accessible by admins
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.ViewChairsModel): contains the model which chairs should be removed

Conference::swapPhase

the get function for a static swap of all submission from one to another phase - currently not used - see autoswapPhase
confShortName (System.String): The unique conference ShortName

Conference::swapPhase

the post function to swap all submissions from an phase to another - see autoswapphase
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.PhaseSwapModel): The model of all submission which should be swapped

Conference::autoSwapPhase

the main get function to swap all submission of a conference - easier as it already tries to decide what should happen to all submissions
confShortName (System.String): The unique conference ShortName

Conference::autoSwapPhase

the main post function to swap all submission of a conference - easier as it already tries to decide what should happen to all submissions
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.AutoPhaseSwapModel): The model of all submission which should be swapped

Conference::manageIPCMember

the get function to get manage the IPCs - used for adding/deleting them
confShortName (System.String): The unique conference ShortName

Conference::addIPCToConference

the get function to add an ipc to a conference - currently not used replaced by datatables
confShortName (System.String): The unique conference ShortName
searchValue (System.String): The search value for persons to be searched

Conference::addIPCToConference

the post function to add an ipc to a conference
model (SRMv2.WebUI.Models.AddUserToConferenceModel): the model which contains the conferenceShortName
userToAdd (System.String): The guid of the user who should be added
searchValue (System.String): The search value for persons to be searched - currently not used replaced by datatables

Conference::deleteIPCFromConference

post function to delete an ipc from a conference
username (System.String): The guid of the user who should be deleted
confShortName (System.String): The unique conference ShortName

Conference::processConflicts

creates a list of all authors of a conference - used to create conflicts between ipc and authors
confShortName (System.String): The unique conference ShortName

Conference::processConflicts

the post function send the conflicts created to the server
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.ListIPCConflictsModel): The model hich containts the conflicts of the user

Conference::biddingById

The get function for bidding by paperId. This function returns the submission of a conference sorted by their paperId
confShortName (System.String): The unique conference ShortName

Conference::biddingById

The post function for bidding by paperId. This function gets called when the user finished his choice on the bidding when they are sorted by the paperId
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.BiddingViewModel): The bidding model which contains the submissions and the users choices

Conference::biddingByAoE

The get function for bidding by areas of expertise. This function returns the amount of submissions one area of expertise one user has. (the submissions themself are loaded per ajax afterwards with BiddingGetPartial)
confShortName (System.String): The unique conference ShortName

Conference::BiddingGetPartial

This functions returns the submissions for biddingbyAoE and biddingbyPref of one specific area of expertise. This function is called by ajax
confShortName (System.String): The unique conference ShortName
itemId (System.String): the preferenceId which papers are returned, "unsorted" the papers who don't have a area of expertise
returnstring (System.String): to save where the request came from (either biddingByAoE or biddingByPref) to konw where to submit

Conference::biddingByAoE

The post function for the bids by Areas of Expertise. The submissions and the user choices are in the BiddingModel
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.BiddingViewModel): The bidding model which contains the submissions and the users choices

Conference::biddingByPref

The get function for bidding by the users preferences. The preferences are sorted where the user is expert, knowledgeable, passing or has no knowledge. Only the amount of submissions of each area are returned. (the submissions themself are loaded per ajax afterwards with BiddingGetPartial)
confShortName (System.String): The unique conference ShortName

Conference::biddingByPref

The post function for bids sorted by user Preferences. The submissions and the user choices are in the Bidding Model
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.BiddingViewModel): The bidding model which contains the submissions and the users choices

Conference::showBiddingResult

The overview function for the chair the see which all IPC member their bids on all submissions and the conflicts a IPC might have with one submission
confShortName (System.String): The unique conference ShortName

Conference::showIPCScores

returns the Scores of IPC members for each submission. These scores will be injected from the outside for the suggestion. This get function is only to see the view if the API is not used.
confShortName (System.String): The unique conference ShortName

Conference::showIPCScores

the post function to inject the IPC scores into the system. These will be used for the suggestions system where IPC are assigned to the papers

confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.ListIPCScoresModel): The model which contains the ipc and the submissions and the associated score

Conference::editSuggestion

With this get function, the chair can modifiy a suggestion of the system, if he is not satisfied with the created.
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId

Conference::editSuggestion

The post function to edit the suggestion. Here the chair posts his modified result. This function can also be used to inject the suggestion from other sources
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId
primary (System.String): The userId who should be primary on the paper
secondary (System.String): The userId who should be secondary on the paper
model (SRMv2.WebUI.Models.EditSuggestionModel): The suggestion Model which contains the tertiary and the suggestion Rating for every person

Conference::showSuggestionsForIPC

The get function to show the the overview of the suggestion for Ipc member where he is primary, secondary and tertiary for each paper
confShortName (System.String): The unique conference ShortName

Conference::takeSuggestionsLive

This get function takes all the suggestions live which are in the suggestion table
confShortName (System.String): The unique conference ShortName

Conference::createNewSuggestionForIPC

The get function to set up for the internal programm to create a new suggestion. If a primary shold get choosen, a secondary and if ter-

tiary how much tertiaries and how many papers a user can maximal have
confShortName (System.String): The unique conference ShortName

Conference::createNewSuggestionForIPC

the post function to calculate the new suggestions. A suggestion for a person is createdy by adding the aoe of a person and the paper (0-100) the bidding of the person (0-100) and the suggestion (userdefined how large it can be). After the calculations are done, the users are assigned to the paper according to the set up in the model
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.NewSuggestionModel): the model which contains the setup for the caluclations (see get function)

Conference::GetImage

returns the Image of a conference
confShortName (System.String): The unique conference ShortName

Conference::setPreferencesForConference

The get function where a user can set his preferences for a conference
confShortName (System.String): The unique conference ShortName

Conference::setPreferencesForConference

The post function where the user can set his preferences for a conference
confShortName (System.String): The unique conference ShortName
formCollection (System.Web.Mvc.FormCollection): Each preference and how the user has voted on it

Conference::parseNewPreferences

The get function to parse the Areas of expertise for a conference
confShortName (System.String): The unique conference ShortName

Conference::parseNewPreferences

The post function to parse the areas of the expertise for a conference. If a line starts with a "*" it gets interpreted as preamble
model (SRMv2.WebUI.Models.ParsePreferencesModel): Contains the new

preferences as one textfield
confShortName (System.String): The unique conference ShortName

Conference::manageReviewerTypes

Overview of the current Types of reviewers in the Programm, from there it is possible to add, delete and edit them
confShortName (System.String): The unique conference ShortName

Conference::editReviewType

The edit form for the Reviewertype with the ID reviewTypeId
confShortName (System.String): The unique conference ShortName
reviewTypeId (System.Int32): the ReviewerTypeID which should be edited

Conference::editReviewType

The post function. When this function is called the model with the new reviewertype is saved
confShortName (System.String): The unique conference ShortName
model (SRMv2.Domain.Entities.ReviewerType): the Model which contains the information about the new edited reviewtype

Conference::deleteReviewerType

deletes the review type with the Id = reviewertypeId
confShortName (System.String): The unique conference ShortName
reviewerTypeId (System.Int32): The Id which should be deleted

Conference::addNewReviewType

Inserts a new ReviewtypeId to the conference
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.ReviewerTypeManagementModel): contains information about the conference which should be added

Conference::assignReviewer

Post function from assigning a user (from the userlist) to create a form to complete the assignment
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId

userToAdd (System.String): The userId of the user who is choosen as reviewer
searchValue (System.String): The search value for persons to be searched - currently not used replaced by datatables

Conference::confirmAssignReviewer

Stores the user as reviewer in table and sends out email to the reviewer that he has a new review to accept
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId
model (SRMv2.WebUI.Models.AddReviewerToSubmissionModel): contains the position, the userID of the reviewer and the message for the Email

Conference::removeReviewer

This get functions parses all reviewers from a submission and shows it to the chair
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId

Conference::removeReviewer

The post function to remove a reviewer from a submission
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId
model (SRMv2.WebUI.Models.RemoveReviewerFromSubmissionModel): The model contains all reviewers which should be removed from the submission

Conference::reviewerDecision

If a reviewer accepts/declines his review assignment, and he clicks on the link which is sent in the email. This function is called.
confShortName (System.String): The unique conference ShortName
submission (System.String): The submission which he accepts or declines
reviewerId (System.Int32): The Id of the assigned reviewer
decision (System.Int32): The decision he chooses (1 = accept, 2 = decline)

Conference::currentReviewStatus

the get overview function for the chair to get all reviews, reviewer and the submissions, here he can also edit the decision if a submission will get accepted or declined
confShortName (System.String): The unique conference ShortName

Conference::currentReviewStatus

the post overview function where the chair adds a review decision (if it get accepted or declined) and the comment to a specific submission
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId
model (SRMv2.WebUI.Models.ReviewStatusModel): the model contains the comment and the decision of a submission

Conference::reviewGetCurrentDecision

the get function to get the real actual decision, if two chairs work at the same time on all submissions gets called with ajax
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId

Conference::addReview

the get function to get the review form, if the user has already entered a part of his review, these values are parsed as well
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId

Conference::addReview

the post function to add a review to a submission
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId
model (SRMv2.WebUI.Models.ReviewModel): the reviewModel contains the standard fields
collection (System.Web.Mvc.FormCollection): used for the fields which are dynamically parsed

Conference::manageReviewFields

the get function for the chair to manage new review fields. The current fields are parsed in the view
confShortName (System.String): The unique conference ShortName

Conference::manageReviewFields

the post function to add a new review field to the conference
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.ReviewFieldsModel): contains the data of the new ReviewField

Conference::editReviewField

The get function for editing a additional review Field in a conference.
confShortName (System.String): The unique conference ShortName
fieldId (System.Int32): The id of the field which should be edited

Conference::editReviewField

The post function of the review field which should be edited
confShortName (System.String): The unique conference ShortName
model (SRMv2.Domain.Entities.AddRevFields): Contains the new data of the edited Review field

Conference::deleteReviewFieldForConference

the post function to delete an additional field from a conference
confShortName (System.String): The unique conference ShortName
fieldId (System.String): the fieldID which should get deleted

Conference::showReview

shows the review of one reviewer for one submission
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId
reviewerId (System.Int32): The reviewer Id the chair wants to see
reviewId (System.Int32): The Id of the review

Conference::reviewerDiscussion

the get function to show all current discussion entries of one submission
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId

Conference::reviewerDiscussion

the post function to add a new entry to a discussion
confShortName (System.String): The unique conference ShortName
submission (System.String): The ShortName for the submission - generated from conference prefix + paperId
model (SRMv2.WebUI.Models.ReviewerDiscussionModel): contains the data for a new Discussionentry

Conference::sendEmailToIPC

The get function to create an email to all IPC members of a conference
confShortName (System.String): The unique conference ShortName

Conference::sendEmailToIPC

The post function to send an email to all IPC member of a conference
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.sendEmailToIPCModel): contains the email template and the ipcs which gets the mail, further the subject and a cclist

Conference::sendEmailToReviwer

The get function to create an email to choosen reviewers of a conference
confShortName (System.String): The unique conference ShortName

Conference::sendEmailToReviwer

The post function to send an email to choosen reviwer of a conference
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.SendEmailReviewerModel): containts all the data - who gets the mail, the email template, the reviewers

Conference::sendEmailToAuthors

This function is creates a template for sending emails to the authors.
confShortName (System.String): The unique conference ShortName

Conference::sendEmailToAuthors

This is the post function for sending emails to the authors. The email will be sent here.
confShortName (System.String): The unique conference ShortName
model (SRMv2.WebUI.Models.SendEmailToAuthorModel): Contains the data for the email (like which authors and the email template what they get)

Conference::AjaxAddUser

The get ajax call to add a new user to anywhere (add reviewer, add chair, add author) in the program. This function returns a userlist. It is based on the jquery plugin DataTables (http://datatables.net/)
param (SRMv2.WebUI.Models.jQueryDataTableParamModel): The parameters for the datatable, see their homepage for further information

Conference::AjaxGetReviewStatus

The ajax call for the chair managing all reviews and deciding if a paper is accepted or not
param (SRMv2.WebUI.Models.jQueryDataTableParamModel): The parameters for the datatable, see their homepage for further information
confShortName (System.String): The unique conference ShortName

Error::General

The general exception, thrown when something went wrong
exception (System.Exception): the exception the user gets presented

Home::Index

the main screen where a user gets all the information of his submissions, reviews and where he is chairs and ipc

Home::mySubmissions

a listing of the users submissions

Home::myConferences

a list where the user is chair

Home::myIPC

a list where the user is ipc

Home::myReviews

a list of the users reviews

Home::Events

a list of all current events

Home::acceptReviewer

the post function for the user to accept a review request
reviewerId (System.String): the users reviewer id
accept (System.String): if he wants to accept or decline it (2 = decline, 1 = accept)
confShortName (System.String): The unique conference ShortName