ProceeDings

A web-based word processor automating the production of conference proceedings

Gabriel Wurzer[1], Bob Martens[2], Thomas Grasl[3]
[1,2] Vienna University of Technology [3] SWAP
[1,2] http://www.tuwien.ac.at [3] http://www.swap-zt.com
[1,2] {gabriel.wurzer|b.martens}@tuwien.ac.at [3] tg@swap-zt.com

In this paper an online editing system for eCAADe papers is presented, which is also the technology behind this volume. On the occasion of the eCAADe 1999 conference in Liverpool, a novel layout for the proceedings was developed. In the course of forthcoming annual conferences, this became the distinctive "look and feel" for eCAADe papers. Due to the complexity, professional typesetting was required for and the authors were disconnected from the publication and layout stage. This paper elaborates on the development and implementation of a web-based tool, which takes care of the typesetting and delegates this activity to the authors. Neither software installation is required, nor specific training must be completed in advance. On top of this the degree of homogeneity can be raised significantly, thus supporting the editors in charge to concentrate on the task of harmonising the publication content.

Keywords: *Word Processing, Proceedings Preparation, Cloud Computing*

OVERVIEW

To the annoyance of both authors and editors, word processing packages are still in their infancy when it comes to guarding a publication template against modification and improper use. Modification, i.e. inserting or altering template styles, is a 'feature' that is often seen when copying and pasting between documents or from the web. Improper use of a template, on the other hand, is a failure to meet the semantics of the entered content itself - think, for example, of entering references incorrectly. On top of this, the authorship is in charge of the final layout and obliged to stay within the given page-limit. Until now, the workflow suffered from a sharp cut between the word-processing-stage and the loosely coupled layout software stage.

During the past three years, we have been developing a web-based word processor that copes with the mentioned problems: It enforces eCAADe's own conference proceedings style and makes use of structured content with added semantics. The resulting contributions can then be automatically compiled into a proceedings book, or exported into $\LaTeX$ for further editing.

We will start with a brief outline of our solution, but then immediately come to our main contribution: The development process behind such a massive web application. Decisions taken in that phase are often far from obvious, which is why we would like to share some of our insights with soul mates who also want to bring their app to the web or into the cloud. We will conclude with a brief statistical overview of some of the data collected during the submission process. This contribution targets hence the wider eCAADe2014 conference theme of "Fusion" in the sense of a collaboration tool aiming at data integration.

BACKGROUND AND RELATED EFFORTS

When speaking about the editing of conference proceedings, managing the time-line is the dominant issue. In the given area of CAAD the (printed) proceedings ought to be finished by the beginning of the conference; a post-conference publication is no option, neither is preponing the deadline for submission. This means that within a timeframe of approximately 12 weeks (or so) roughly up to 150 papers with a significant amount of figures/tables/images need to be edited. Therefore, it makes sense to delegate a justifiable amount of work to the authors supporting the homogeneity of each individual contribution. An electronic publishing environment should be able to secure unintended violations of implemented rules/restrictions. It should be mentioned that data entry via a structured web mask is mandatory. The papers/contributions are to be centrally stored and perpetual backup routines must be implemented.

A first thought which instantly comes up is that a gaggle of editors must have been tackling these issues over and over. Possibly there exist different views on the level of consistency and accuracy to be achieved. It could, however, also be that some of the work involved is regarded as unavoidable and divinely ordained, be it alone the persistent use of upper and lower case in the title and headings. Indeed, wide-ranging reflective publications are numerous [such as Russey et al 1995; Fredriksson 2001] and even a series of conferences has been dedicated to this topical area [such as King 2000; www.elpub.net].

When searching for similar tools predominantly a variety of tools can be retrieved, allowing to handle a collection of already created PDF-files. ProceeDings focusses on the contrary on the production of the publication entry itself, carefully respecting the predefined publication guidelines. As a matter of principle, content management systems (for websites) would allow to settle a similar task within a system of distributed roles. However, this would in most cases require a training, whereas ProceeDings claims to be almost self-explanatory.

A solution like ProceeDings does not make the guild of editors jobless as there is still a need for taking care of the content of the publication as a whole. Especially the setup of coherent sessions and their composition in (parallel) sequences is rather demanding.

PROBLEM: FORMATTING

It is not the first time that the eCAADe has created a web-based platform for its community: The Cumulative Index on CAD (Martens and Turk 2000), for example, was founded as paper archive for education and research in architecture and urban planning. We have targeted the same community with our web-based word processor, however, it is more than likely that we will find us serving a greater range of fields in the future.

There were two main drivers that led to the development of the tool:

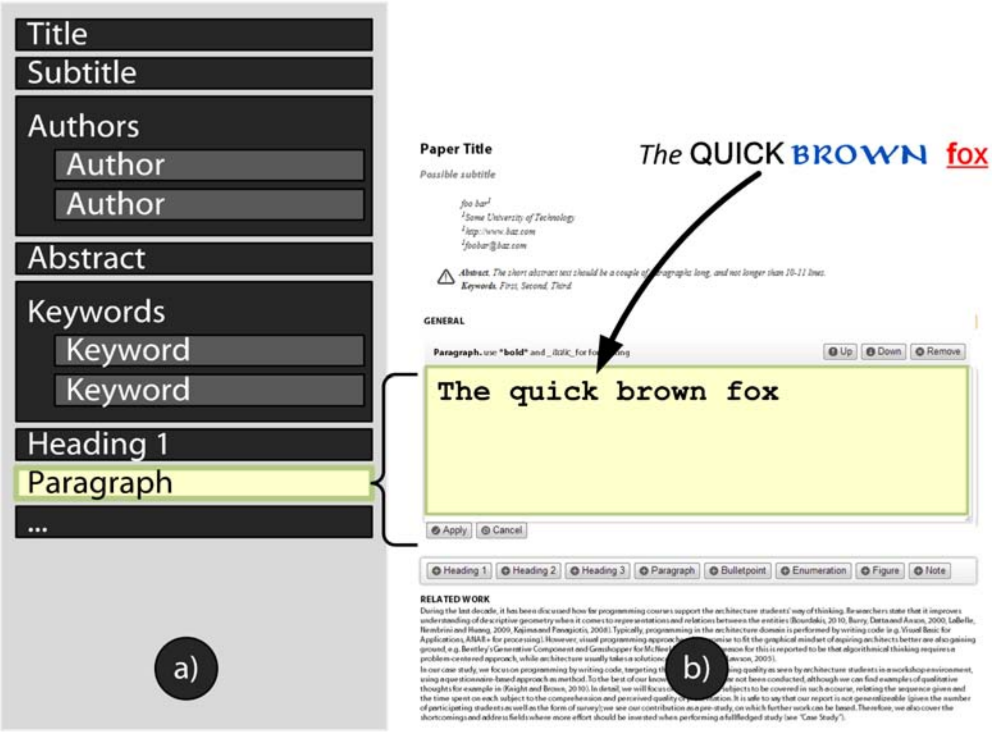
- We were unsure what to do about Microsoft Word's persistent tendency of breaking the formatting of a document. Locking the template had only mixed success, and likewise did Springer's Manuscript template (which uses Visual Basic for Application Macros for taming Word to some extent, but: The user has to enable Macros and copy-and-pasting from the web automatically allocates new paragraph styles...).

- We felt that a general-purpose word processor is ill-suited for the conference's needs, as the user enters text but the program does not enforce semantic and structural rules (see next section) that lie at the heart of every paper. LaTeX does (or better: can do, if instructed) all of this, however, it is not common in the field and requires a certain degree of a learning effort. Having a larger audience switching from Word to LaTeX was, as can be imagined, also not feasible.

SOLUTION: SYNTACTICALLY DUMB, SEMANTICALLY RICH

Microsoft Word's problems in keeping formatting intact is the result of mixing text semantics (heading, paragraph, bullet list,...) and styling (font family, size,...). Copy-and-pasting brings out the worst of both worlds, in the sense that the text is pasted using the current paragraph format, which is automatically updated to reflect the pasted text's styling. Discarding styling information altogether and pasting only the text would be the obvious solution, however, there is no way of forcing this behaviour without user intervention, e.g. by storing settings together with the template.

Pasting "only text" does do away with superflu-



Paper Title

Possible subtitle

foo bar[1]
[1]Some University of Technology
[1]http://www.baz.com
[1]foobar@baz.com

⚠ **Abstract.** *The short abstract text should be a couple of paragraphs long, and not longer than 10-11 lines.*
Keywords. *First, Second, Third*

GENERAL

Paragraph. use **"bold"** and _italic_ for fo[illegible]ing

```
The quick brown fox
```

RELATED WORK

During the last decade, it has been discussed how far programming courses support the architecture students' way of thinking. Researchers state that it improves understanding of descriptive geometry when it comes to representations and relations between the entities (Boutdakis, 2010, Burry, Datta and Anson, 2000, LaBelle, Nembrini and Huang, 2009, Kajima and Panagiotis, 2008). Typically, programming in the architecture domain is performed by writing code (e.g. Visual Basic for Applications, ANAR+ for processing). However, visual programming approaches [illegible] promise to fit the graphical mindset of aspiring architects better are also gaining ground, e.g. Bentley's Generative Component and Grasshopper for McNeel [illegible] reason for this is reported to be that algorithmical thinking requires a problem-centered approach, while architecture usually takes a solution[illegible] (Lawson, 2005).

In our case study, we focus on programming by writing code, targeting th[illegible] ing quality as seen by architecture students in a workshop environment, using a questionnaire-based approach as method. To the best of our know[illegible] ar not been conducted, although we can find examples of qualitative thoughts for example in (Knight and Brown, 2010). In detail, we will focus o[illegible] subjects to be covered in such a course, relating the sequence given and the time spent on each subject to the comprehension and perceived quality of [illegible]ntation. It is safe to say that our report is not generalizable (given the number of participating students as well as the form of survey); we see our contribution as a pre-study, on which further work can be based. Therefore, we also cover the shortcomings and address fields where more effort should be invested when performing a fullfledged study (see "Case Study").

Figure 1
(a) Contributions as set of paragraphs which are context-aware, they know their place within the text body (example: paragraphs cannot be moved higher than the "keywords" section) and also the editing options according to their type (example: figures offer an upload). (b) Editing happens via plain text entry and markup, which enables the tool to discard superfluous formatting as in the given example.

ous styling, however, we completely loose text semantics - the paragraph format, bold, italic, and so on. In our web editor, we have therefore devised a two-fold editing strategy that enables us both.

Semantically rich. We regard each contribution as a set of paragraphs containing plain text (see Figure 1a). Paragraphs are context-aware: For example, a heading cannot be moved further than the list of keywords. One may not add an abstract block in the body of the paper, or anything else but references in the references section.

Syntactically dumb. Paragraphs are edited - one at a time - simply by clicking them. In the simplest case of a "paragraph" format, the application then shows a text box where one can enter or paste plain text (see Figure 1b). For paragraph types that acquire structured information, such as the list of authors or a reference entry, we present not one but many plain text boxes in which the user types in the needed information. Formatting beyond the paragraph level is done using a wikitext-like syntax, for example *bold* or /italic/. What wikitext markup is available in each paragraph ultimately depends on its type: For example, headings offer nothing but plain text, while paragraphs furthermore enable **bold**, *italics* and $equations$ done in this manner.

TECHNICAL OUTLINE

Our tool is composed of four technical layers:

- **Editor Front-end**: We chose to implement a "Web 1.0" application which is based on classical request/response cycles, utilizing PHP (served by nginx/php5-fpm) for generating the HTML presentation (styled by CSS). Optional functionalities such as the real-time preview of formulae are handled via JavaScript libraries (jQuery, MathJax) which degrade gracefully in case of missing browser support. As a matter of fact, the application displays even on outdated browser versions (we tested e.g. Internet Explorer 6 and the text browser Lynx [1], which dates back to 1992).

- **Editor Back-End**: Instead of a classical database, we store our documents as files. The reason behind this is that load tests showed database performance to drop when memory is constrained, whereas files-based access remains fairly stable even if the server is under heavy load. Furthermore, we utilize a memory database (memcached) for fast access to back-end data and session management.

- **Printing**: Documents are converted into $\LaTeX$ for printing. The actual generation of a PDF is a very costly task, and therefore, we have utilized a flexible "printer queue" (utilizing Beanstalk for PHP) where multiple other servers could help in generating a print preview of the currently edited document. On the other hand, having too few resources to serve all print requests will not overload the server, since all clients are queued and only some are served at a time.

- **Book Compilation and Metadata creation**: The final compilation looks at a spreadsheet stating (1.) in which volume is (2.) which session, containing (3.) what contributions. It then re-renders all PDFs in that order, giving the correct page numbers and also assembling the list of contents in the process. The resulting PDF are the proceedings, but not the final product: We still have to do an extraction of metadata (e.g. authors, title, references) and creation of metadata for indexing, in for example CUMINCAD, automatically.

All these parts work together so that authors can collaborate in producing a consistently formatted proceedings that saves the editor valuable time which can now go into the actual editing process . Scientifically though, these points are hardly of concern. Indeed, what we are more interested in is *how* the community works with our tool, and *when* this work is delivered. An analysis of this is brought in the next few sections.

LESSONS LEARNED

As eCAADe 2014 was the first time that a web-based editing tool was used, we coined it ProceeDings^{Beta}. Due to the constant correspondence with the authors and monitoring the layout of the whole proceedings ourselves, we are able to give some insights on what worked and what did not.

What worked

Inserting well-behaved text. Pasting paragraphs from the clipboard *as plain text* and formatting that with markup worked like a charm. The only exception were inline equations, which presented an own problem (chars that are known to Microsoft Word but not to Unicode, WingDings and the like).

Warnings. Most authors took a great deal of pride in having their paper *warning-free*: For example, the system would complain about too short or too long abstracts, and people strongly responded to that. That (in the end) the system was so liberal as to allow submission of every paper, regardless of warnings, was not of concern. The people simply wanted to help getting their work right, and this is one of the most assuring things which we encountered during review of all the final submissions. Some exceptions (which we have to deal with, of course) were the references, where the system would sometimes complain wrongly about a URL being wrong, when in fact they had only given an additional [accessed 2nd June 2014]. We are delighted by this ability to judge the system by ones personal experience in scientific writing - some authors also left use *NOTE* paragraphs explaining what they want to appear in the paper and why they could not accomplish - and thus this is one of the points which we build on for the next version, exposing e.g. more tools that we as editors had in figure positioning and paragraph indentation ("dont-indent" my paragraph, please).

Selecting the right kind of paragraph for the job. For people not concerned with structured text, terms such as "heading 1, 2, 3", "authored book" and "edited book" might not be very descriptive. So it shook us when we realized that people actually understand far more about paragraph formats than we had anticipated: After all, we had not given out a template in LaTeX instead of the web-based system because of fear that this would be not understandable. The impression that we got is that that, even though LaTeX might be too much to stomach for the whole community, the general concepts are clear for everyone - and this includes structured markup such as the ones mentioned.

What did not work

A paper is not an image gallery. It would be a lie to say that most authors added figures *accompanying the text*. In fact, it was the other way round: The text would accompany the figures! Given the very limited abilities of LaTeX in positioning the graphics, this was as much of a pain for us as in layouting as has been for the authors in editing. The future perspective on this is clear: (1.) Constrain the use of images to cases where they are visualizing the text (every image needs to be cross-referenced, as mentioned in the User Guide), (2.) constrain the number of images to a minimum, we think of 5-8 at the moment and (3.) give more options for positioning the images in the text, accompanied by a clear description over how LaTeX will attempt to position them.

Figure positioning. Figure positioning deserves some more attention: LaTeX will position a figure either "right here" in the text or let it "float":

- "Right Here" means that images will be one column wide. If there is not enough space vertically in this column, LaTeX will shift them to the next column (or even the next page, 1st column), leaving an ugly hole in the text. Authors need to know that they need to close these holes by inserting a "here" figure where it has enough space, at best in the middle of a column, surrounded by lots of text.

- "Floating" means that LaTeX will put a figure either at the top or at the bottom of *the next opportunity*. That again means: When a figure is "beyond the top" of the page, it will insert it at the bottom of this page or at the top of the next page. When the layouter has already crossed the "the bottom" of the page, it will insert it on "the top of the next page". In practise, this means that all floating pictures need to be defined well before their insertion position, so that the layout algorithm has them at hand when going through the text. Arguably, this is quite counter-intuitive. However, this is nothing that we can work around, as LaTeX is built like that.

The mentioned points are even more enervating when preparing for different kinds of output media: For example, an eBook has the requirement that figures always appear exactly in the spot where they are mentioned, i.e. "here". For a printed proceedings, however, we may additionally use "floating" figures, which may need to be defined before the spot where they are actually referenced. Essentially this is the divide between *structure* (as in eBooks) and *layout* (as in printed proceedings). It will be our task to think about ways in which we can bridge this gap, providing more options for positioning (e.g. figure "on an own page", which was used during editing) and also for vertical spacing before and after figures.

Formulae. Formulae were the main cause why a paper did not print properly. This is no coincidence, but the result of two diametrically opposed policies on dealing with erroneous input: The web-based formula viewer (MathJax) would simply ignore all offending markup and display what it could make sense of without complaining, while the printing algorithm (LaTeX) would immediately stop and report an error.

Errors produced by LaTeX are handled by ProceeDings such that it shows an error page. It does, however, neither know what happened nor where the error lies (i.e. no parsing of error text, yet). Therefore, we display a generic error page that gives some hints over what could have gone wrong, but is no use when it comes to hunting down specific errors in a formula - leading to a lot of despair and troubleshooting via mail. In further detail, our analysis shows that there were three separate cases that led the printing algorithm to fail:

- **"Unicode" formula input into web editor**: What most people do is paste a formula from Word into a paragraph. Technically, this is no formula, but merely a set of characters that are hopefully Unicode - and the system will appropriately save them as normal text. In some cases, however, characters that Word pastes are simply not Unicode - they are symbols in Windows encoding! So this process can fail terribly, if we have no clue what this symbol is (remark: this is likely, as we have one programmer (currently writing this) against the rest of the Microsoft world). A better way would be to input a formula markup, presented in the next bullet point.

- **"Formula" input into web editor**: With the help of surrounding #, an author can insert an inline formula (the other option would be to make a standalone formula paragraph, which does not need that). Some authors have taken this hurdle, but kept pasting Unicode into them. This can go well - the formula is layouted in the formula font instead of the text font - but this can also fail (when the character is not defined in the formula font).

- **Formula defined in ASCIIMath but LaTeX is too dumb to digest it**: For those authors that did embrace the formula syntax (technically: ASCIIMath syntax, an easy way to enter even the most complicated formulae), there were two further hurdles: Either the formula was not entered correctly or the the converter to LaTeX simply produced erroneous results; the web editor would always produce a result, in the sense of "I am happy with what you enter; I will typeset what I understand and skip the rest"; but LaTeX would not work this way - it would crash! So this went definitively wrong, such things should never happen.

Summarizing, we should definitely parse the error text given by LaTeX and show the author *what* the error is, and *where* it lies in the paper. Another lesson learned: Do not use the conversion from ASCIIMath to LaTeX, use the image produced by the web-based formula viewer, at least for non-inline equations. In that way, what you see would truly be what you get.

What did work, but most authors say it didn't

References were a source of constant dispute between the editing team and the authors. As must be said, we had lack of support in importing from EndNote or BibTex, which is a shame fully taken. As a result, the authors had to re-enter every reference again, which was frustrating. However, the process of having to review every reference again *according to eCAADe's needs* has proven very effective in assuring quality. In fact, it is clearly one of the conferences where quality of references is of the highest standards, since we enforced semantic rules rather than only taking "the data" that authors would provide (for example, journal articles require a volume and a page) which is far beyond what people would normally give us. Furthermore, it allowed authors having different citation management systems to produce one homogeneous reference section.

A BRIEF LOOK AT ABSTRACT SUBMISSION

ProceeDings was used for the editing of full papers, while OpenConf has been employed for abstract upload (Word or PDF). Thus, strictly speaking, this part should not appear at all in our paper, as it is something we are not concerned with. However, it might nevertheless put some contrast on our later analysis of the the final submission (see next section).

When looking at the number of submits to the abstract submission site, we can note that these are roughly four times of abstracts handed in. Further analysing when the submits happened, we can see that typical eCAADe submitter is:

- **Well-behaved**: The peak of the initial abstract submissions is *eight days before the initial abstract submission deadline* on 3rd February (there was an extended submission deadline on February 10th, which led people to pause for two days before resuming a steady stream of submissions).
- **Occupied during the week**: Most submissions happen *Mondays and Fridays between 12 und 16 hours UTC*.
- **Expecting an extended abstract submission**: There are *as much submission in the extended deadline period as before*; this means that people really assume that there will be an extended abstract submission deadline, and use this to correct their paper.

These three points are of course a very subjective interpretation - the analysis of the final submission (see next section) tries to put some more scrutiny in so as to further narrow down what a conference organizer can expect, on a statistical basis.

FULL PAPER SUBMISSION

The following is a statistical overview of the full papers submitted for eCAADe 2014 (sidenote: 164 paper were initially accepted, 148 were present at the time of the extended submission and 127 remained in the final proceedings; either the authors failed to complete their work or withdrew their paper after submitting; the following data is a snapshot as per 18th June, two days after the extended deadline).

If ever there was a proof for a tendency to procrastination in academia it is shown in figure 2. The graph displays when authors started uploading their papers (positive y-axis) and when they finally submitted the paper (negative y-axis). The timeline along the x-axis starts at the beginning of April, which is when the first invitation to log-in to the system was sent out, passes the initial deadline (D) on the 9th of June and ends just after the extended deadline (ED) on the 16th of June. Work on the papers did not start to increase until about two weeks before the deadline, with a sizable portion not starting until after the

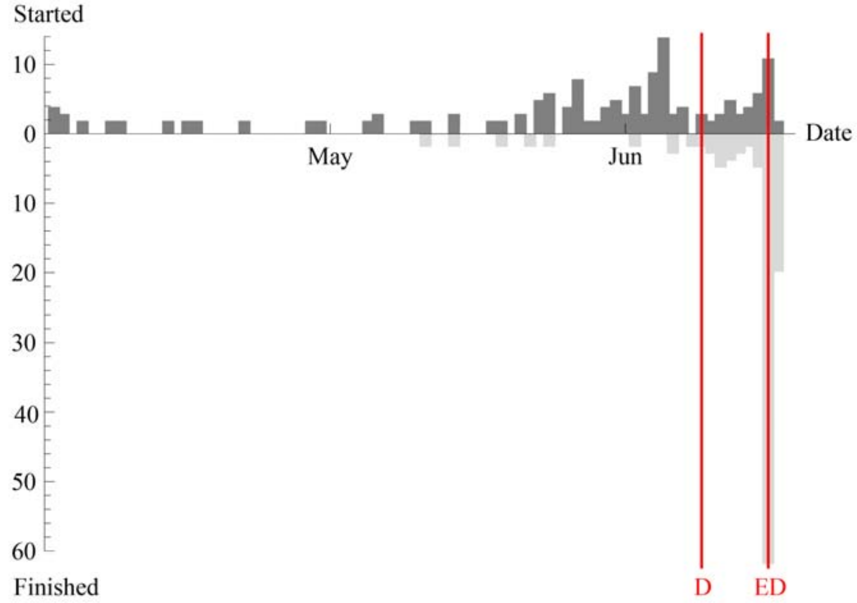


Figure 2
Timeline showing the dates the authors started and finished uploading their data. The initial deadline (D) and the extended deadline (ED) are overlaid.

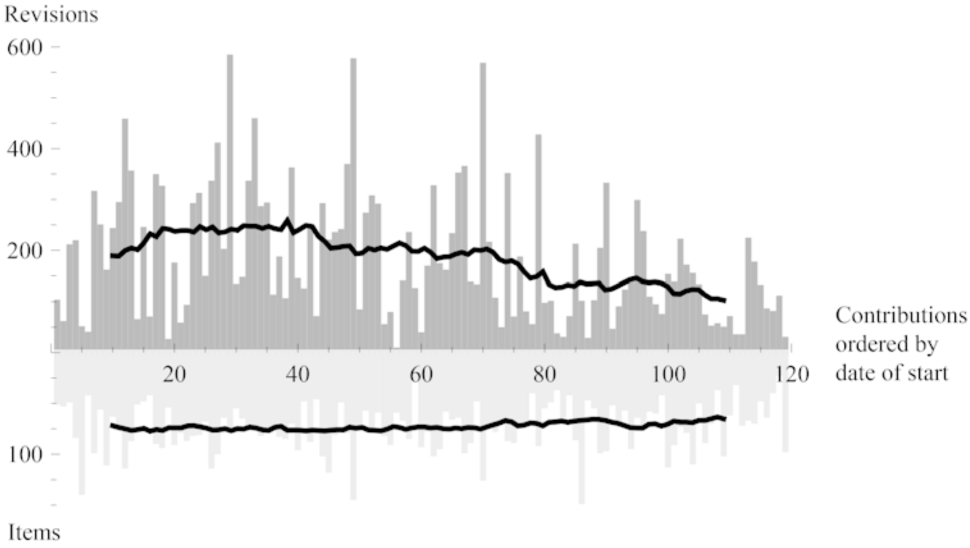


Figure 3
Number of revisions and number of items per paper. Papers are sorted by the order work on them was started. The moving average of each measure is overlaid.

initial deadline had passed. It can also be seen that the email sent out on the 5th of June announcing the deadline extension caused a sudden drop in activity. Procrastination can be seen even more clearly with the paper submissions. Most waited until the extended deadline and some continued to work even beyond the deadline.

However, the graph in figure 3 clearly shows that the individual preferences in time planning have little impact on the length of the paper. Here papers are sorted by the date the authors started to upload their data and displayed along the x-axis. The positive y-axis gives the number of revisions, the number of times the authors made a change like inserting, changing or deleting data, on the negative y-axis the number of items (paragraphs, headers, images, etc.) is displayed as a measure of the overall length. Both measures are overlaid by their moving average over 20 papers. While there is a clear tendency to fewer revisions by papers started later, the overall number of items remains fairly constant.

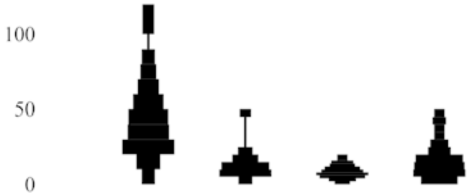


Figure 4
Distribution of content items, headers, graphical items and references (l.t.r.) over all eCAADe 2014 papers.

Figure 4 shows the distribution of items over all papers. The items were grouped into content items (paragraphs, lists, formulas and algorithms), headers, graphical items (images and tables) and references. Papers were then divided into 16 categories according to their keywords and the item distribution was calculated for each category (Figure 5).

CONCLUSION AND OUTLOOK

The implementation of ProceeDings in the framework of eCAADe 2014 has delivered a treasure trove of experience which will be used for further developments. The tool was able to automate the production of eCAADe proceedings, starting from an initial list of accepted publications (coming from OpenConf) and ending with PDFs ready to be delivered to print production.

During the whole process, the editors still have an important role: As the technicalities of the paper layout are dealt with, they can direct their attention towards the content as such, i.e. focus on editing. In that context, ProceeDings allows to monitor ongoing developments within the community of submitting authors at an early stage, i.e. before the (final) submission and eventually to deliver feedback. Till a relatively late stage, authors can be involved in the publication process and the final outcome can be made available at any time.

The number of opportunities to (unintentionally) overrule the template is shrinking and especially ongoing live experience will accelerate the improvement of the interface. Most importantly, the users can take care of the layout themselves: You-Get-What-You-See (YGWYS) instead of You-Will-Sometimes-See-What You-Will-Get (YWSSWYWG). In this regard, the pre-publishing option (preprints) might gain interest. The extraction of (coherent!) metadata, required for indexing etc. has to be highlighted as well. It is unlikely that ProceeDings will encompass the abstract/paper submission and review stage. Here, a number of well-functioning (open source) environments has been made available for a longer period of time.

ACKNOWLEDGEMENTS

ProceeDings was programmed by Gabriel Wurzer over the course three years. Bob Martens took the tool into the eCAADe council, from which we got a first "go" in getting it so far that it would be ready for eCAADe 2015. Luckily, Emine Thompson gave us the chance to test-drive the system even earlier, as publication platform for the eCAADe 2014. We are extremely greatful for this decision, since it involved the risk that some functionalities would still be beta-quality. The paper importer proved to be

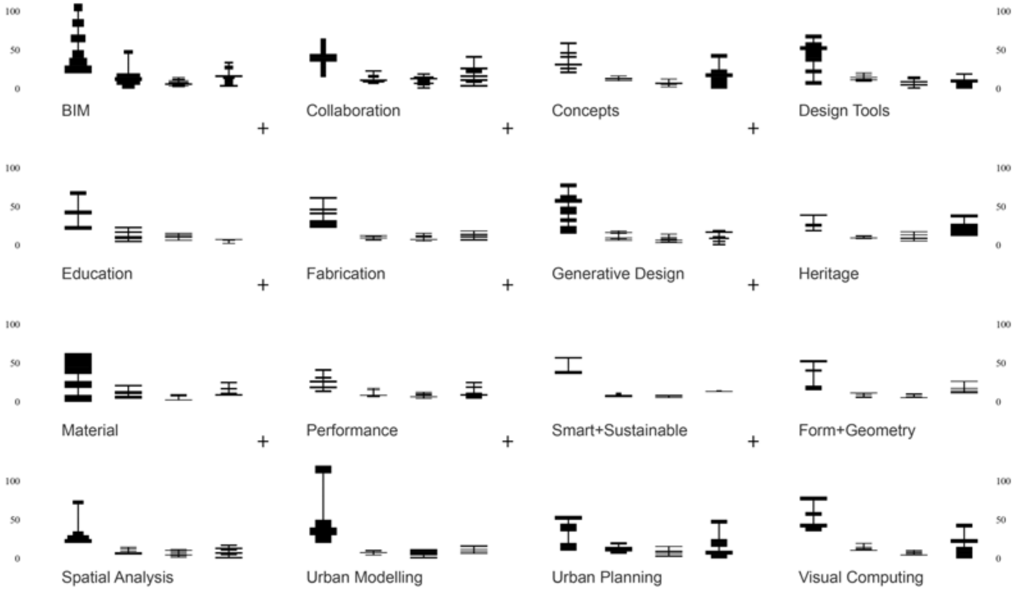


Figure 5
Distribution of content items, headers, graphical items and references per paper category.

such a case, thanks goes to Marie Davidova and Martin Tamke for pointing out multiple issues. Likewise, Thomas Grasl and Rudi Stouffs gave their input on multiple features either missing or totally buggy, as did a multitude of authors through our bug reporting system. Thanks goes as well to Gregor Hartweger, Andrea Wölfer, Rudolf Scheuvens and Georg Penthor: The first three made it possible that we got our own server hosting the system (memory and CPU power: *sufficient*), and the latter one made sure we could relocate our whole set-up into a new building when the department was getting refurbished. From the editorial side, colleagues Wolfgang Lorenz and Gerda Hartl helped tremendously in layouting the proceedings. Speaking of layout, we also want to thank Henri Achten for showing us how to do this *with style*, based on his team's great work with the eCAADe 2012 proceedings. Our main volume of thanks goes to all authors, who did most of the work: It is on their shoulders that we all stand. Last but not least, here is to you Martin Winchester, thank you for the OpenConf support which you do every year, and for the chance to pull an analysis of the abstract submission.

REFERENCES

Fredriksson, EH (eds) 2001, *A Century of Science Publishing: A Collection of Essays*, IOS Press

King, P (eds) 2004, *Digital Documents: Systems and Principles: 8th International Conference on Digital Documents and Electronic Publishing*, Springer, Berlin/Heidelberg

Martens, B and Turk, Z 1990, 'The Creation of a Cumulative Index on CAD: "CUMINCAD"', *ACADIA Quarterly*, 19(3), p. 18–19

Russey, WE, Bliefert, C and Villain, C (eds) 1995, *Text and graphics in the electronic age : desktop publishing for scientists*, Wiley-VCH

[1] http://lynx.isc.org/