Indico Central - Events Organisation, Ergonomics and Collaboration Tools Integration

José Benito González López¹, José Pedro Ferreira, Thomas Baron
CERN IT-UDS-AVC, 1211 Geneve 23, Switzerland

E-mails: {jose.benito.gonzalez, jose.pedro.ferreira, thomas.baron}@cern.ch,

Abstract. While the remote collaboration services at CERN slowly aggregate around the Indico event management software, its new version which is the result of a careful maturation process includes improvements which will set a new reference in its domain. The presentation will focus on the description of the new features of the tool, the user feedback process which resulted in a new record of usability. We will also describe the interactions with the worldwide community of users and server administrators and the impact this has had on our development process, as well as the tools set in place to streamline the work between the different collaborating sites. A last part will be dedicated to the use of Indico as a central hub for operating other local services around the event organisation (registration epayment, audiovisual recording, webcast, room booking, and videoconference support)

1. The Indico Software

The Indico (Integrated Digital Conferencing) Software was born as a European project, a joint initiative of CERN, SISSA, University of Udine, TNO, and Univ. of Amsterdam. The main objective was to create a web-based, multi-platform conference storage and management system. This software would allow the storage of documents and metadata related to real events.

Four different modules were originally identified [1]:

- Multimedia document storage system (iM2), developed by SISSA;
- Conference Metadata Database (iConference), also developed by SISSA;
- Conference Management Software (“Make a Conference” or simply MaKaC) and
- Signal analysis and video/speech recognition modules, developed by the Univ. of Amsterdam and TNO;

The project started in May 2002, and ended 2 years later. Apache/mod_python was chosen as the platform for the implementation, since Python provides a “high-level, dynamic data typing and a reduced learning curve that fit very well into the project requirements and development process” [2]. The database chosen for MaKaC was ZODB, from the Zope Project, an object-oriented DB, which fully integrates with the Python language, and transparently handles object persistence.

After the end of the European project, CERN decided to keep the development of Indico, and put it in production, focusing on the MaKaC module. Indico was the natural substitute for the aging CERN

¹ To whom any correspondence should be addressed.
Agenda, the event management software in use at CERN, until May 2007. From that time onwards, Indico fully replaced the Agenda.

Indico is currently intensively used at CERN (Figure 1), hosting more than 70,000 events and 320,000 contributions. Most of the events that take place in the organization are scheduled through Indico, so that the whole community can consult them and collaborate. Things such as section/group meetings are easily manageable, allowing the participants to submit materials and share them with others. Many other events (mainly conferences) which happen to take place outside CERN are also hosted in CERN's Indico server.

Figure 1. Number of events scheduled in Indico, from 2004 to April 2009

Indico is Free Software, released under the GNU General Public License. This has made possible the adoption of the tool by more than 40 institutions world-wide, and the contribution of code by third-party developers. There’s an active user community, which almost every day provides new suggestions of improvements. This contributes substantially to the degree of agility at which the Indico project currently works, providing immediate bug fixes, patches, and user support.

2. The Problem

It is normal, nowadays, to speak of “user friendliness” as a natural requirement for every piece of software that interfaces with at least one human being. This buzzword is frequently brought up by software developers and vendors as a major argument in the promotion of their products. In the field of Software Engineering, it is common to associate this concept with “usability” and “accessibility”, two different but closely connected dimensions in which we can measure the quality of an interface.

The ISO-9241-11 [3] standard defines “usability” as being the “extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use”. It is not hard to notice the subjective nature of such a concept, from this definition. However, Nielsen [4] proposes a set of concepts that can be related to the “phenomenon” of usability, and thus provide some more objective “metrics” for this concept. They are:

- “Learnability”, or “How easy is it to learn how to use the system?” – Intuitiveness plays an important role here;
- “Efficiency of Use” – The effect of the system on the productivity of its users;
- “Memorability” – The degree to which a user won’t forget how to use the system, or at least the basic features;
- “Few and non-catastrophic errors” – Errors from the part of the user and the system should be avoided;
- “Satisfaction” – The measure of how pleasant it is to use a system;

Indico was built by computer scientists, with the scientific community in mind as the main user group, and suffers from the same issues that most “feature-centered” applications do. Although the main concern was to provide a large set of features, the impact of these features in the usability/accessibility perspective was sometimes neglected. In order to fight this effect, the enhancement of Indico’s usability and ergonomics was set as a priority by the project management.
In order to better understand the nature of the problem, a detailed study on the accessibility/usability problems of Indico was conducted. This study, of “static nature”, consisted on the verification of Indico against third-party checklists, so that the most evident problems could be identified. However, it was clear that some direct feedback from the users was required: initially, the support mailing list archives were used as a base, but the need for something more up-to-date and filtered, a direct connection with the user community, was needed. In May 2007, a user survey was put in place, and linked from Indico’s main page. It is known that the statistical accuracy of voluntary sample-based surveys is worthy of contestation [5], but there were no other feasible options at the time. On the other hand, if it is possible to somehow estimate the bias that affects the data, this information may become useful.

Also in May, Rosy Mondardini, from the CERN IT Communication Team, joined the usability improvement effort. Rosy proposed to produce a parallel analysis based on Jakob Nielsen’s “website deconstruction method” [6]. She projected as well a “Laboratory Test” for Indico, where six users from different backgrounds (“hallway test”) were asked to perform a set of tasks (previously defined with the help of the development team) using Indico, and to express aloud their lines of thought. The sessions were recorded (audio and video) for further analysis.

2.1. Findings

The first report on Indico’s usability used the guidelines from the U.S. Dept. of Health and Human Services [7] as a checklist for measuring usability in an “objective” way. It was the first study on Indico’s usability, and the first to identify specific usability flaws:

- The inability to select input standards for date/time, which requires from the user an additional processing step;
- The lack of context help, and the insufficient help documentation;
- The reduced visibility of some icons/options, such as “login”;
- The absence of the “news” section in the main page;
- The navigation panel was in the right hand side, instead of being in the left (recommended);
- Some links/options had confusing names;
- Indico required too many clicks and page transitions;
- Indico was not HTML 4.01 Transitional compliant, as it should;

Subsequently, the analysis of Indico using the web site deconstruction technique provided some additional information:

- Navigation elements were not clearly differentiated – navigation links and tools were mixed;
- Menus were not consistently placed on the pages – some were put on the left hand side, some on the right side (not recommended), and others at the top. There was also no common template for the menus: they all looked different;
• Icons were inconsistently used throughout the site: some were used as links, others as tools, and others for decoration purposes. Besides that, they didn’t have a common style;

The “Laboratory test” confirmed most of the identified issues, and provided yet another additional set of results, these ones more “specific” (since it was then possible to analyze complete task execution sequences). Most of the identified problems were related to:

• Terminology issues – users easily confused some concepts, since their names are not the most adequate ones: “timetable” vs. “programme”, “slot” and “track”, and other examples of closely related concepts that may generate confusion;

• Ambiguities – the result of some of the decisions taken by the users was not clear – for instance, a presentation has an “author” and a “presenter”, and some users would choose to fill only the “author” field. However, this “author” field would not appear in the generated timetable, but the “presenter” would.

• Excessive number of options – users got overwhelmed by the enormous amount of links, options and tabs in the management pages;

• Placement of important information – for instance, pages that occupy more than a screen make some important features only accessible through scrolling;

Some recommendations for the resolution of the mentioned issues came up from this last study:

• Reworking the terminology used in the interface, in order to harmonize it with the jargon used in major conferences;

• Adding contextual help to the interface;

• Hiding the more advanced options from regular users;

• Providing a simple explanation of the main concepts in the homepage;

• Avoid unnecessary steps in the execution of some of the tasks (i.e. activating “call for abstracts”);

• Reworking the FAQ, and focusing most of the “static” help documentation on it;

Finally, the “Indico Usability Survey” provided a means of quantifying the degree of satisfaction of the users, in order to identify critical areas for usability improvement, and confirm the results of the previous studies, using a relatively large sample of users.

3. Solution

During the analysis phase, a set of requirements for a better usability/accessibility was identified. From these, four were non-functional:

• Intuitiveness – Easiness of use, simplicity and good organization of the interfaces;

• Accessibility – Improve cross-browser compatibility;

• Aesthetics – A better look and feel, a cleaner interface, where users feel more comfortable;

• HTML standard compliance – Indico should pass on the validation tests for the HTML standard, as well as with the other formats (CSS, RSS, iCal, etc…);

From the functional requirements identified, the most important were selected:

• Improvement of the “Search” feature – integrating it with the site, and making it available throughout the application;

• Implementation of a Contextual Help System – making it possible to add tooltips and other auxiliary means of information, in order to provide some “hints” to the user;

• Single-click (in-place) edition for form fields;

• Refactoring of Indico’s layout – Eliminating some bad practices, like using HTML tables for layout purposes, and refactoring the whole interface, according to usability and accessibility guidelines, and accomplishing the non-functional requirements;

The aforementioned objectives implied deep technological changes in Indico.
3.1. Technological review

Indico was a traditional web application that should move towards the concept of “Rich Internet Application”. This concept was introduced in 2002, by Macromedia Inc. (which is nowadays a part of Adobe), as a description of a web application with the features and functionality of a normal desktop application. Although the concept was, at the time, meant to separate itself from HTML/JavaScript based applications, it is now commonly associated to DHTML as well.

There are several technologies capable of building RIAs:

- DHTML (HTML/JavaScript) and AJAX – widely used, cross-platform, although somewhat limited. It has the advantage of keeping (X)HTML as the underlying format, allowing information to be harvested and indexed, in spite of the interface improvements. It is, in addition, supported by all the modern browsers, without the need to install any kind of extension;
- Adobe Flash – a widely adopted technology, cross-platform, powerful, but not very content-oriented;
- Java Applets – this technology had a boom in the early stages of the Web, for its cross-platform nature, but is slowly falling into disuse, due to its bad performance, big file sizes, and lack of web-oriented features;
- Microsoft Silverlight – a recent technology, powerful, but still young;

These are only a few examples. For obvious reasons, DHTML/AJAX is the natural solution for Indico: it is not only content-oriented (which is important for an application like Indico), but also allows the improvements to be done gradually, since small changes may be introduced as JavaScript.

3.2. Asynchronous JavaScript and XML

In 1996, Microsoft introduced the notion of asynchronous requests, through the IFRAME tag, back in Internet Explorer 3. But it was only in 1999, with the introduction of the XMLHttpRequest API (Internet Explorer 5), that what we know today as “AJAX” became possible. In spite of the early introduction of this technology, it was not before Garrett [8] that the term “AJAX – Asynchronous JavaScript and XML” was coined. The sudden “boom” of AJAX applications in the last four years may be explained through the following factors:

- An increase in the connection speed, and available bandwidth (broadband is today available to most internet users);
- The implementation, in 2002, by the Mozilla project, of the XMLHttpRequest API, which opened the way for cross-browser utilization of this technology;
- The availability of better cross-browser JavaScript standards support;
- The so-called “Web 2.0 phenomenon”, which boosted the demand for rich and intuitive web applications;

AJAX-based interfaces have several advantages when compared to traditional ones: they’re completely asynchronous, making them suitable for multiple parallel requests, and they don’t require the browser to reload the page all over again, which spares time and bandwidth; the feedback is immediate, and field validation can be done in real-time.

Successful web applications, such as Google’s Gmail, Docs & Spreadsheets and Facebook, make intensive use of this technology. AJAX has become almost a cliché in the Web 2.0 movement, being used over and over again for web application implementation and refactoring.

However, AJAX has its own disadvantages:

- DOM Content generated directly by the JavaScript code is not read by web crawlers, as they usually do not execute scripts;
- The “traditional” web paradigm is distorted: for instance, a “pure” AJAX-based RIA would now allow bookmarks, since it would not include many page transitions;
3.3. A New Service-based Architecture

The already mentioned AJAX paradigm looked as a perfect solution for the introduction improvements in Indico’s interface, provided that:

- Old and new interfaces could coexist, since the technological base would remain the same, and only new “asynchronous” features would be added;
- The possibility to add in-place editing and DHTML-based widgets would be very helpful reducing the number of screen transitions and click required for the execution of certain tasks;

Being so, a hybrid model would be the ideal choice, since “regular” pages need to be generated, and asynchronous services for AJAX must be provided as well. Figure 3 introduces a “hybrid” architectural model for Indico. The essential modifications would be the addition of a layer of “Asynchronous” request handlers – the old “synchronous” ones cannot be used, because they’re bound to HTML generation methods, and would not be able to export XML or JSON [9].

![Figure 3: Hybrid architecture for Indico (Asyndico)](image)

At the client side, a JavaScript module (in Figure 16, IAC – Indico Ajax Components), which would be common to all the Indico pages, and provide them with basic widgets and methods for asynchronous operations, would invoke the “asynchronous” request handlers, and communicate with them using the JSON format. JSON was selected over XML, because it is lightweight, and most JavaScript libraries include support for serialization/unserialization of JavaScript objects as JSON. The asynchronous request handlers would simply execute the desired operations, without the need for “middlemen”, directly using the functionalities provided by the MaKaC core.

This architecture, codenamed Asyndico (for “Asynchronous Indico”) would, at the same time, provide a good chance for the user experience enhancement that Indico seeks, and take advantage of the already existing structure, while preserving most of the code base. It is only the first step in the development of a usable interface for Indico.

With the introduction of Asyndico, new things would be possible, such as:

- In-place editing – just click the text, change it, and it is saved on the server side;
- “Silent” submission of information – submission of form data with no page transitions, and presentation of the results in the same page;
- Real-time validation of fields – instead of having to submit a form, the interface would automatically point possible errors previously only detectable after the request;
- Asynchronous download of information – new widgets could be created, that obtained their contents not from the data which the server sends as response to the initial request, but to subsequently downloaded information, through asynchronous requests;
3.4. Context Help

After its identification through the usability studies, the “Context Help” feature has been introduced as a fundamental requirement for the project. Its user-oriented nature should substantially shift Indico’s “gravitational centre” towards the user side, in an attempt to reach equilibrium of forces.

However, the introduction of a feature of this nature at a global application level is something that requires an enormous effort, and a great amount of time. In addition, the multilingual translation effort of the Indico interface would multiply this effort by N languages. Solving this question through “hardcoding” techniques would eventually become impossible.

Being so, an easy way to add contextual help to the Indico pages had to be found: something that would not require much work on the templates, and repetition of code; something oriented towards content, capable of delegating the responsibility of actually rendering the interface part to automated means.

Hence, the WOHL (Web-Oriented Help Specification Language) format [10] was created especially for Indico, as an XML dialect for context-sensitive help specification. The inexistence of any kind of language for such a use motivated the creation of this format, which is aimed to be simple and totally platform-independent. A WOHL definition works like a “decorator”, which enriches the target document with tooltips and help boxes. The target document may be any HTML or XML-based document (i.e. XHTML or XUL). WOHL doesn’t define any implementation-specific details, or even how the elements will be decorated: it instead defines a common vocabulary for context-sensitive help specification. It is a clean way of separating normal information from context help (auxiliary content), and avoiding the “hard coding” of tooltips and other text directly in page templates.

3.5. New Layout

The current web interface that Indico provides to its users suffers from some usability/accessibility problems already described above. After the realization of the “laboratory” usability tests and the subsequent production of the findings document, an improved layout for Indico was designed, according to the usability and accessibility guidelines, and having in mind the conclusions of all the studies realized to the date.

The prototype of the new interface presented important changes:

- A new, uniform style for icons;
- Reorganization of the menus: placement of the main menu on the left side and clustering of options according to functional categories;
- Inclusion of contextual help, as a means of providing the user with hints about the required actions (for instance, a tooltip on grayed-out items, telling the users to login, as in Figure 22);
- Inclusion of a “news box” in the main page;
- New settings for font sizes and colors;

The layout is being built in a completely table-less way, which means that no HTML tables are being used for any purposes which not of showing strictly tabular data. All the positioning and formatting of HTML elements is done through CSS, reducing the amount of presentation information in the HTML to the possible minimum.

The new page layout is compliant with the following browsers and layout engines:

- Gecko-based browsers:
  - Mozilla Firefox up to version 3.0;
  - Mozilla Seamonkey;
  - Epiphany;
- Trident-based browsers:
  - Internet Explorer ver. 6 and 7;
- KHTML/WebCore-based browsers:
  - Apple Safari;
- Konqueror;
- Google Chrome
- Opera 9 (Presto layout engine);

Figure 4. New home page

4. Future

At CERN, there is an enormous usage of “collaborative tools” in order to allow scientists and engineers to collaborate and work together, even if they are not physically in the same places or simply to spread the scientific knowledge world-wide. Those tools include videoconference systems, such as EVO [11] and H.323 through an MCU (Multipoint Control Unit) [12], as well as recording systems and webcasting.

Many organizers and participants of Indico events also need to use collaborative tools, thus creating the need for integrating them within the web application. Indico users would be able to create an event and almost automatically to book a videoconference virtual room, start recordings/webcasting and so forth.

This integration is being accomplished in a way such that other institutions can easily extend and manage any of the plug-ins controlling collaborative tools. A brand new plug-in system will allow users to interact with these tools.

Some other improvements are planned for Indico in the medium term: paper reviewing module, highly interactive drag-and-drop timetable, several tools for personalization and performance optimization, amongst others.
5. Conclusions

These are times of rapid change for the Indico software. Work is being carried on in different fronts: front-end, collaboration mechanisms and back-end. Indico’s front-end will be totally changed to a more advanced web application, coherent with the current state of art. This is the first step towards a “social” Indico in which scientists can better collaborate. Obviously, the integration with collaborative tools plays an important role in the matter. This first evolution of Indico tries to get rid of all the usability problems that were detected through the various studies that have been done. This is not just a mutation of Indico’s front-end but also an improvement that will notably ameliorate the user experience.

As well, it is worth mentioning that all the technological improvements which have been adopted will allow proceeding with a second phase plan: improving Indico’s performance. This and the former will imply a quality leap for the application.

6. References

[1] Baron T 2002 The InDiCo Project - A Tool for Managing Conferences through the Web (Presentation at the JACoW Workshop)
[9] JSON (JavaScript Object Notation) http://www.json.org/