Faculdade de Engenharia da Universidade do Porto

Licenciatura em Engenharia Informática e Computação

Improving the Indico Framework at the European Organization for Nuclear Research

Internship Report - LEIC 2006/2007

José Pedro Macedo Alves Ferreira

FEUP Supervisor: Jaime Villate

CERN Supervisor: Thomas Baron

August 2007

“All in all, I must say I enjoyed the visit to Sweden, in the end.

Instead of coming home immediately, I went to CERN, the European center for nuclear research in Switzerland, to give a talk. I appeared before my colleagues in the suit that I had worn to the King's Dinner -- I had never given a talk in a suit before -- and I began by saying, "Funny thing, you know; in Sweden we were sitting around, talking about whether there are any changes as a result of our having won the Nobel Prize, and as a matter of fact, I think I already see a change: I rather like this suit." Everybody says "Booooo!" and Weisskopf jumps up and tears off his coat and says, "We're not gonna wear suits at lectures!" I took my coat off, loosened my tie, and said, "By the time I had been through Sweden, I was beginning to like this stuff, but now that I'm back in the world, everything's all right again. Thanks for straightening me out!" They didn't want me to change. So it was very quick: at CERN they undid everything that they had done in Sweden.”

Richard Feynman

Abstract

This document describes the work developed by José Pedro Macedo Alves Ferreira, Informatics Engineering and Computing (LEIC) undergraduate student at the Engineering Faculty of the University of Porto (FEUP), in the context of the project “Improving the Indico Framework”. The project took place at the European Organization for Nuclear Research (CERN), in the framework of both the Technical Student Program of this organization, and the curricular internship of the aforementioned degree. The contents of this report refer to the internship period, the first half of the one-year Technical Student program.

The project aimed to introduce usability improvements into an already existing web application, the Indico platform, a integrated system for event scheduling and management, which was initially developed as a European project and continued by CERN, being currently used by several institutions worldwide. Indico presented some usability issues that for long had been noticed by the users and required correction, mainly problems in the navigation process (event browsing), and the lack of user-centered features (i.e. a “personal area”).

Initially, a thorough usability study was put in motion, including not only “checklist-based” analysis but “user-centered” approaches as well. A user survey and a laboratory test were conducted, and their results evaluated and taken into account in the definition of recommendations and requirements. The requirements whose priority was higher were selected, and functional prototypes were designed and implemented. This included an integrated search feature, a “favorite users” system, a personal area, and a contextual help system.

The result of the project was a set of prototypes, compliant with the previously defined recommendations, and ready to be included in the production version of Indico. The objectives were accomplished, and the project entered a new phase, in which the user interface will be totally refurbished, in order to comply with the established requirements, and the developed prototypes will be introduced as normal features, available to the regular user.

Acknowledgements

I’d like to thank everyone at CERN who helped me in some way with this project, especially my office mates David Bourillot, Piotr Włodarek and Tomáš Novotný, and everyone else in the AVC section, as well as the UDS group, through its leader, Tim Smith. They were always very supportive of my work, and welcomed me with arms wide open.

I’d like to mention also the contribution of Rosy Mondardini, from the IT Communications Team, who carefully planned and directed the “laboratory” usability tests, and designed the new interface for Indico. She brought “the light of usability” down to the dark dungeons where programmers selfishly code to their own image. Less metaphorically speaking, without her collaboration, this project would have been much less ambitious.

I should not forget the people at FEUP who helped me, especially my supervisor Prof. Jaime Villate, who was supportive from the beginning, when I was yet to know that had been selected at CERN.

And there would be no justice in these words, if I forgot to thank in particular my supervisor, project manager and section leader Thomas Baron, for his patience, open mind, and sympathy. I loved working in an environment where people discuss options, and even the opinion of the most inexperient technical student is taken into account and seriously considered.

Since good work requires an equally good mood, I think I should mention Pedro Pinto and Mário Pereira, for their friendship, patience and support during this “adventure”. These are the people who take the burden of living with me on their backs (and that includes waking up at night with the sound of my guitar).

I know it is *praxis*, but some people really mean it, and this is the case. A big thank you to my mother and my sister, for encouraging me into grabbing this oportunity, in spite of knowing that it wouldn’t definitely be easy for them.

Finally, I’d like to thank CERN’s Technical Student Program, for the financial support, which made possible the realization of this internship.

Contents

Figure Index

1 Introduction

This document reports the work developed in the context of the project “Improving the Indico Framework”, which took place at CERN, from March 2007 to August of the same year. The scope of the project centered itself in the question of usability and accessibility.

The issue of usability is, most of the time, seen as a ghost that haunts the software developers’ quiet lives, forcing them to spend extra effort in “non-technical” matters, such as design, ergonomics, comfort, and even some psychology. It is even stereotypical, this image of the computer scientist that would gladly communicate with all the external world by means of a single command line, and would be happy to develop all the user interfaces based on this principle. Thankfully, reality is not so extreme as it could. However, most applications (some of them even commercial and widely used) present several important usability issues, that trouble the users in their everyday usage of these systems. In the field of web applications the example of Hotmail[1] may be considered. In an study performed by nine independent organizations, which conducted nine different usability evaluations, three-hundred and ten different problems were reported. The authors are clear about the amount of usability issues present in a normal website:

“Assuming that Hotmail indeed represents the state-of-the-art within website usability, we can conclude that the number of usability problems in a typical website may be so large that one cannot hope to find more than a fraction of the problems in an ordinary usability test.” (Molich, et al., 2004)

The project hereby described has been designed with the intent of assessing the usability problems of the Indico platform, not only in the perspective of the developer, prone to adopt a global view of the system, centered on what he thinks to be the user experience, but also in the prism of the regular user, who has access solely to the application interface, and actually feels the problems of a deficient usability. Both views are important, since the user requirements would be useless if they weren’t feasible (and a developer is in a privileged position to evaluate this “feasibility”), and any “improvement” would be fallacious if not validated by real data, that only users can provide. Being so, communication channels had to be established between the developers and the end-users, and several evaluation processes were used in order to process this feedback.

1.1 Introducing CERN

In the aftermath of the Second World War, Europe found itself trapped between two rivaling powers, and weakened by a conflict which had taken her as the main theater. It was clear that no single European country was capable of developing, alone, a nuclear research program which would be competitive. Being so, the only solution was to create a joint effort between several European nations, so that they could all take part in the costs and benefits provided by this “laboratory”. Finally, in 1952, the CERN - *Conseil Européen pour la Recherche Nucléaire* (European Council for Nuclear Research) was created, and took the responsibility of building the new European laboratory. In 1954, the *Organisation Européenne pour la Recherche Nucléaire* (European Organization for Nuclear Research) was

[1] http://www.hotmail.com

established, and the construction works began, in the previously selected location, near Geneva, Switzerland. As to this day, the European Organization for Nuclear Research is still referred through the acronym “CERN”.

CERN has come to be the world's largest particle physics laboratory, employing 2600 full-time employees, and a temporary staff of around 6500 physicists and engineers, who take part in several research projects. The organization's current “flagship project” is the LHC (Large Hadron Collider), a giant particle accelerator, with a circumference of 27 Km, aimed at the detection of the “Holy Grail” of particle physics, the “Higgs boson”, a particle which was never detected, but is theoretically predicted by the current models. The LHC is being equipped with four main particle detectors: ATLAS, ALICE, CMS and LHCb, focused on different experiments.

The association of CERN with computer science and information technologies comes not only from the recent developments in the area of parallel computing (i.e., the LCG – LHC Computer Grid), but essentially from the legacy of Tim Berners-Lee and Robert Cailliau, who, in the late eighties and early nineties, developed the World Wide Web.

CERN is divided into seven different departments. Each one of these departments is constituted by “groups”, which, themselves, are small clusters of “sections”. A section is a group of people who work on a common subject. The work described by this document was developed at the AVC (Audio, Visual and Conferencing Services) section, from the UDS (User and Document Services) group of the IT (Information Technology) department.

1.2 The Indico Project

The Indico[2] (***In****tegrated* ***Di****gital* ***Co****nferencing*) Project was born as a European project[3], a joint initiative of CERN, SISSA[4], University of Udine, TNO[5], 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 (Baron, 2002):

Figure 1 - Logo from the Indico Project

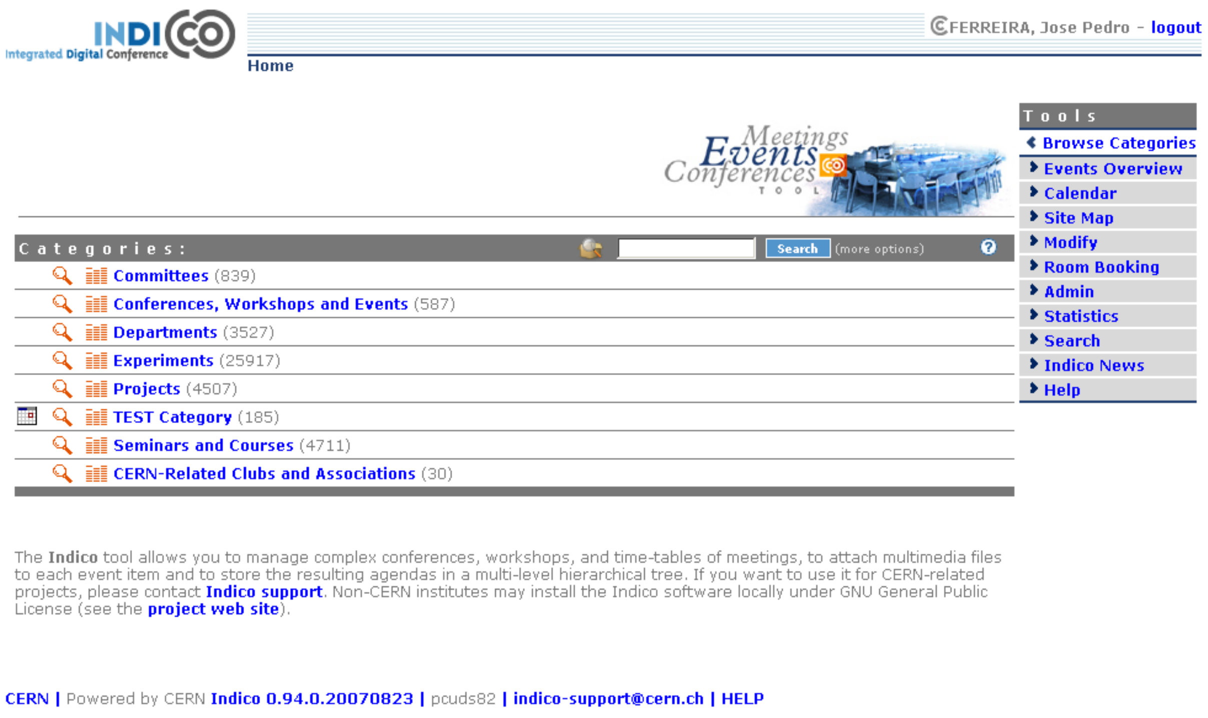
- Multimedia document storage system (iM2), developed by SISSA;
- Conference Metadata Database (iConference), also developed by SISSA;
- Conference Management Software (“Make a Conference” or simply *MaKaC*)
- Signal analysis and video/speech recognition modules, developed by the Univ. of Amsterdam and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

[2] http://www.cern.ch/indico

[3] IST-2001-34306

[4] International School for Advanced Studies, Trieste, Italy

[5] Netherlands Organization for Applied Scientific Research


FERREIRA, Jose Pedro - logout

Home

Tools
- Browse Categories
- Events Overview
- Calendar
- Site Map
- Modify
- Room Booking
- Admin
- Statistics
- Search
- Indico News
- Help

Categories: Search (more options)

- Committees (839)
- Conferences, Workshops and Events (587)
- Departments (3527)
- Experiments (25917)
- Projects (4507)
- TEST Category (185)
- Seminars and Courses (4711)
- CERN-Related Clubs and Associations (30)

The **Indico** tool allows you to manage complex conferences, workshops, and time-tables of meetings, to attach multimedia files to each event item and to store the resulting agendas in a multi-level hierarchical tree. If you want to use it for CERN-related projects, please contact **Indico support**. Non-CERN institutes may install the Indico software locally under GNU General Public License (see the **project web site**).

CERN | Powered by CERN Indico 0.94.0.20070823 | pcuds82 | indico-support@cern.ch | HELP

Figure 2 - The Indico Home Page, as from version 0.94

The project started in May 2002, and ended 2 years later. Apache[6]/mod_python[7] 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" (Baron, et al., 2004). The database chosen for *MaKaC* was ZODB[8], from the Zope[9] 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[10], although focusing on the *MaKaC* module only. Indico was the natural substitute for the aging *CERN 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 (see Figure 3). 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 currently hosted in CERN's Indico server.

---

[6] "The Apache HTTP Server Project" - http://httpd.apache.org

[7] "Mod_python – Apache/Python Integration" - http://www.modpython.org

[8] http://wiki.zope.org/ZODB

[9] http://www.zope.org/

[10] Indico at CERN – http://indico.cern.ch

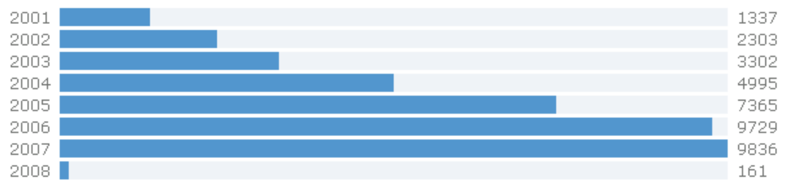


Figure 3 - Number of events scheduled in Indico, with starting date from 2001 to 2008

Features

Indico provides features for the management of the entire conference lifecycle, as well as for meetings and single lectures:

- Tree-like structure, organized into categories. Each category may either contain other categories, or contain simple events;
- Automatic web page creation for the events;
- Event evaluation surveys;
- Automatic notifications (i.e., automatically remembering all the participants in a meeting that it will take place today);
- For Conferences:
  - Registration form customization;
  - On-line payment support;
  - Abstract submission and reviewing;

Besides these basic features, Indico provides as well:

- An integrated room booking system, extensible, and currently in use at CERN (replacing the old CERN Room Booking System);
- Integrated support for videoconferencing software (i.e. VRVS[11]);
- Exportation of information in different formats: RSS feeds, iCal and MARCXML, for instance;
- Creation of DVDs with all the data from a Conference;

At the moment of the writing of this document, some other features were under development:

- Multilingual interface (internationalization);
- Support for different time zones;
- Accessible and usable interface – which is included in the scope of this document;

System Architecture

As Figure 4 shows, Indico has the typical three layer architecture[12]: presentation, business logic, and data access. It provides a variety of interfaces to the external world, being

[11] "VRVS – Virtual Rooms Videoconferencing System" – http://www.vrvs.org

[12] See (Sadoski, et al., 2000)

the web interface the most important. However, data may be exported as OAI-PMH, in order to be used by external applications. An example of such use is CDS Invenio, which is used to harvest and index Indico's contents, and provide the users with the possibility of searching through it.

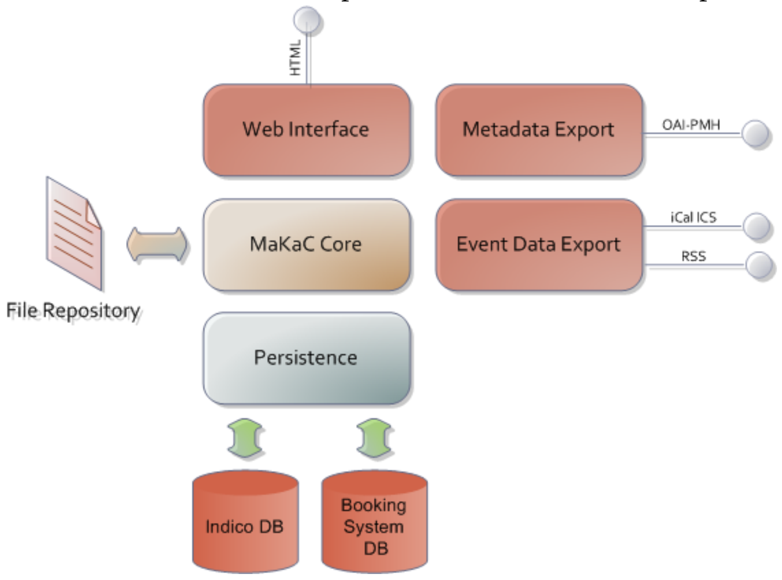


Figure 4 – A simplified view of Indico's architecture

Concerning the data access layer, Indico has a Persistence layer, composed of ZODB persistent classes, and a common interface for data access. This provides access to two databases: the "regular" Indico database, and the "Room Booking System" database, being that the latter is used for storage of room booking information, by Indico's Room Booking Module. Indico also stores files (presentations, papers, images, etc…) in an organized repository, located in the filesystem.

Most of the structures used by the application's business logic are contained in a set of "core classes", referred in the diagram as "MaKaC Core".

Infrastructure at CERN

Indico is heavily used at CERN, since it is the *de facto* tool for event scheduling and organization software, at the organization. Due to that, servers frequently suffer from a high load, at times when large conferences are taking place, sometimes more than one at the same time. In the past, this caused various downtime periods, which were harmful for the events that relied on the platform. Due to that, during the June-July 2007 period, a new server infrastructure was put in place, in order to create a faster, more reliable and fault-tolerant system.

As Figure 5 shows, the main Indico server is *sunuds95*, which is accessible to the internet in general. This server is a Sun Microsystems AMD Athlon 64 based server (dual processor), running the Debian GNU/Linux[13] operating system. The other two application servers (*pcuds80* and *pcuds82*) are two-processor Intel Core 2 Duo E6600 machines, running the SLC4[14] (CERN's official GNU/Linux distribution) operating system.

[13] http://www.debian.org

[14] Scientific Linux CERN 4 - http://linux.web.cern.ch/linux/scientific4/

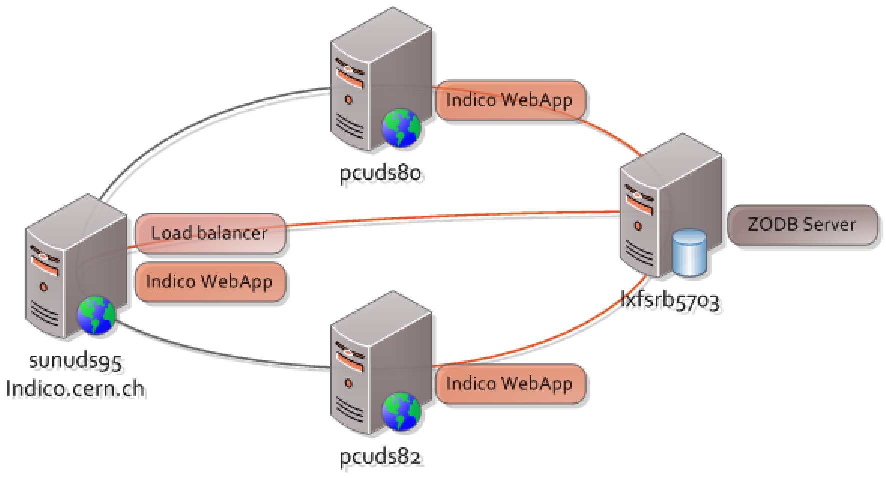


Figure 5 - Deployment of the Indico system, at CERN

Each one of the three servers runs an instance of Indico, on the Apache HTTPD Server. *sunuds95* runs an additional Apache HTTPD instance, which is the one associated with the *indico.cern.ch* sub-domain, that answers to client requests. The requests are then forwarded either to the other local HTTPD instance, or to one of the other two servers, based on a *round-robin* resource allocation strategy.

The *lxfsrb5703* server runs two instances of the ZODB Server, one for general Indico usage, and the other one for the Room Booking System.

Distribution

Indico is Free Software, released under the GNU General Public License[15]. This has made possible the adoption of the tool by several institutions around the world, and the contribution of code by third-party developers. There's an active user community, which almost every day provides new suggestions and bug reports. This contributes substantially to the degree of agility at which the Indico project currently works, providing immediate bug fixes, patches, and user support.

As of now, besides CERN, more than 40 known instances of Indico exist, in institutions like Fermilab (Chicago, USA), IN2P3 (France), EPFL (Lausanne, Switzerland) and the Chinese Academy of Science.

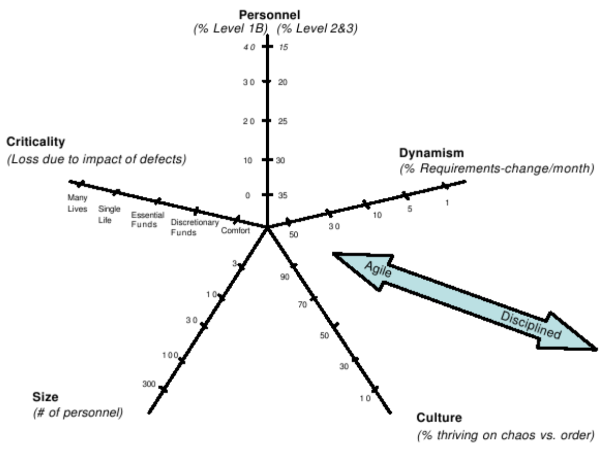


Figure 6 – Polar graph for agility vs. discipline.

(From **(Boehm & Turner, 2003)**)

Software Development Process

At the time of the writing of this document, the latest release of Indico was version 0.94, and 0.96 was under

[15] http://www.gnu.org/licenses/gpl.html

development. This versioning scheme gives a hint about the “unstable” nature of the current Indico implementation. Rather than “unstable” for a possible lack of reliability, Indico can be considered “unstable” in what concerns about some of its requirements, which rapidly and frequently change.

Using the approach that (Boehm, et al., 2003) introduced for classification of the degree of agility of a known project, based on a polar graph (Figure 9), the Indico Project would definitely locate itself very near to the center of the graph. This is due to the fact that we’re dealing with a very small (2/3 people), very experienced core team, under a large requirement change rate, and a relative low level of criticality. This team is definitely more “chaotic” than organized: information flows quickly between its members, there’s a “lightweight” sense of software development phases, and the approach is very prototype-based. Besides this, there is a high degree of user involvement: features are frequently directly proposed by the users themselves, or implemented “by request”. This brings a high level of dynamism to the software development process, and, as well, a degree of user-centeredness that a more “disciplined” approach would certainly not provide.

1.3 Improving the Indico framework

The main objectives of this internship, as described by the Indico project manager in the internship proposal, were:

- To improve the usability and ergonomics of the Indico tool;
- To better understand users’ needs in this particular area*;*
- To implement specific modules in the areas of personalization and administration;

These items were a direct response to the complaints of the users: they found the “classic” Indico interface not very intuitive, and very demanding in terms of time and effort.

The intended result was a set of fully-functional prototypes, ready to be transferred to the “production code branch”, which would implement the required features. Obviously, a deadline of six months was not enough to address every single problem, since the expected number of issues was not expected to be so small, and the project would require an analysis step, even before anything could be designed or implemented[16]. Being so, the most critical requirements were selected by the project manager, and marked for implementation, so that project would better fit in the context of the internship.

These results were reached, and the project eventually got to a phase at which it was possible to advance towards more ambitious objectives. A new layout for the Indico website (an important requirement, that had been postponed due to the amount of effort and time required to complete it) was prototyped and is currently at the middle of its implementation.

1.4 Document Contents and Organization

This document is organized in six different sections, which form a logical sequence, aiming to expose the different phases this project has gone through.

[16] See 2 – “The Problem”

First, an introduction (this same section) is provided, so that the reader familiarizes himself with CERN, the Indico Project, and the scope of this particular project. Then, a detailed description of the problems targeted by this project is provided (Section 2 – “The Problem”), including the studies that where conducted in order to classify and quantify the different concepts involved, as well as the proposed timeline for the project. After the explanation of the problem, a review of possible technologies which could be used for its solution is done, in order to make the reader comfortable with some of the concepts that are used in later sections.

Section 4 addresses this project as a solution for the problems mentioned in Section 2, explaining how each concrete case can be solved, and including real examples from the developed prototypes. The reader is then taken to the next section, “Prototype Implementation”, which addresses some of the technical details involved in the implementation of the solution, justifying, as well, some important decisions, taken during the process.

Finally, Section 6 – “Conclusions and Future Work” draws the conclusions derived from all the developed work, placing it in the technological context of nowadays, and evaluating its potential as an effective solution for the problems in question. The next steps in this project are specified, as well as some ideas for future improvements for Indico, and, specifically, for the survival of good usability practices during its development.

2 The Problem

It is normal, nowadays, to speak about “user friendliness” as a natural requirement for every piece of software that interfaces with at least one human being. This buzzword is not seldom 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.

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.

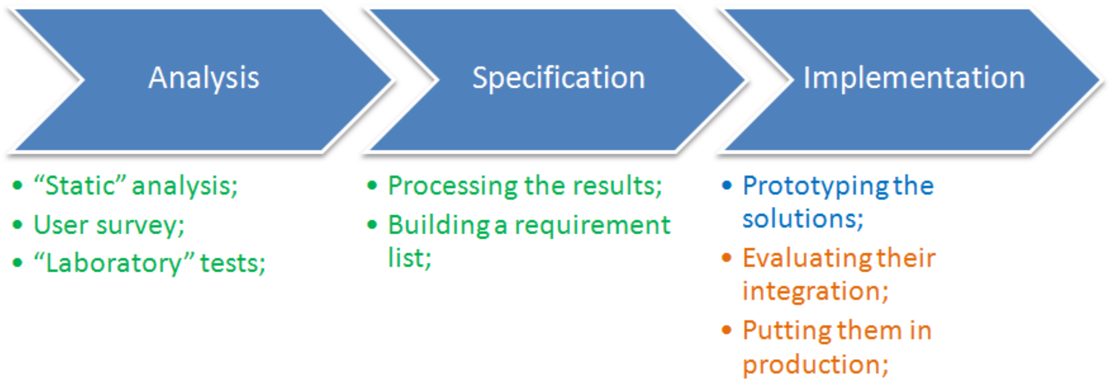

Analysis
Specification
Implementation
• “Static” analysis;
• User survey;
• “Laboratory” tests;
• Processing the results;
• Building a requirement list;
• Prototyping the solutions;
• Evaluating their integration;
• Putting them in production;

Figure 7 - The workflow for the "Usability Enhancement Project"

This “Usability Enhancement Project” for Indico was split into three different phases:

- The first one, “analysis”, which took the first month (March 2007), and consisted of an analysis of the problem, collection of user feedback on usability/ergonomics, as well as documentation in the aforementioned fields. The first “contacts” with Indico were established as well, and some tasks[17] were implemented, so that some knowledge on the application code and internals would be acquired;
- The second one, “specification”, which took the second month (April 2007), and consisted of an analysis of the gathered information, and the creation of a set of requirements for the accomplishment of the proposed objectives;
- The third one, “design/implementation”, which encompassed the rest of the internship time, and consisted in the design and implementation of the features

[17] The WYSIWYG poster editor for lectures, as well as the ability to create server-wide templates for badges/posters were implemented, and already included with the Indico 0.92 release.

with the highest priority (as classified by the project manager), using an agile approach, as the organizational culture suggested;

In order to better understand the nature of the problem, in the second phase, 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, after being approved by the IT-UDS group management, 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 (Couper, 2000), 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. These results were included in the accessibility/usability study which has resulted from the analysis (Ferreira, 2007b).

Also in May, Rosy Mondardini, from the *CERN IT Communication Team*, who has attended some training in the usability field (from Jakob Nielsen[18]), joined the usability improvement effort. Mrs. Mondardini proposed to produce a parallel analysis based on the Nielsen's (2001) "website deconstruction method", summarized in her report (Mondardini, 2007a). 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, in the presence of Mrs. Mondardini and two members of the Indico Development Team. The findings extracted from these tests are documented in (Mondardini, 2007b).

2.1 The Usability Problem

The ISO-9241-11 (1998) 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 (1993) proposes a set of concepts, represented in Figure 8, that can be related to the "phenomenon" of usability, and thus provide some more objective "metrics" for this concept. They are:

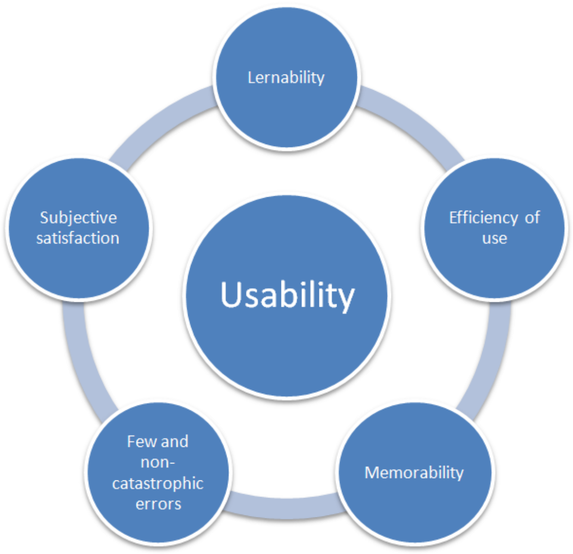


Figure 8 - The concept of Usability

- "Learnability", or "How easy is it to learn how to use the system?" – Intuitiveness plays an important role here;

[18] See http://www.useit.com/jakob;

- “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;

Much can be said about usability: it is a recurrent theme in a society where human-machine interfaces play an increasingly important role. However, this document will focus on the subject of web usability, and, in particular, on its application to the Indico Project.

Indico’s Usability

Usability rose as an issue through user feedback. It was obvious that Indico was hard to use. The report on Indico’s usability (Ferreira, 2007b) used the guidelines from the U.S. Dept. of Health and Human Services (2006) 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 one (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, (Mondardini, 2007a) analyzed Indico using the web site deconstruction technique, more focused on the web interface problems. This analysis 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). The report (Mondardini, 2007b) details the identified problems, most of them 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;

The document also provided some recommendations for the resolution of the mentioned issues:

- 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 other studies, using a relatively large sample of users.

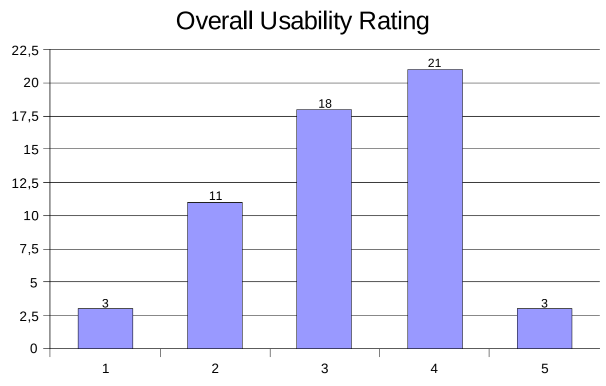


Figure 9 – The Overall Usability Rating attributed by users (1-5)

The results of this survey are documented by (Ferreira, 2007a). Figure 9 displays the histogram for the "Overall Usability Rating" attributed by Indico users to the application. It's clear that the mode is 4, although the average stays at 3.18 (median 3). It was a positive result, in spite of all the detected problems.

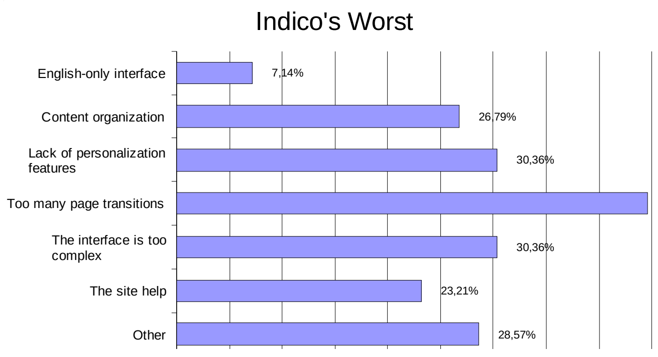


Figure 10 – What users dislike most

If we ask to Indico users to specify what they dislike most about Indico, they are very clear about their answer (Figure 10). Almost 45% of the users complain about the excess of page transitions, and about 30% of them complain about the interface complexity. A great amount of users (almost 30%) decided to leave their own observations. The latter may be found in the results document.

Pathological Cases

In order to base the conclusions drawn above on specific situations, two examples of “pathological cases” can be used. The first one is the most basic operation that an Indico user may perform: browsing for a desired event. The premises for this operation are already uncomfortable for the user: he has to know exactly where to find what he looks for. Since the Indico category tree has an unlimited branching factor, and there’s no other way to find an event (except using an existing bookmark or the external search tool), the user will certainly have problems if he/she doesn’t know where to look for.

The “browsing process” is described in Figure 11: the user has to iterate over the same “clicking process” several times, assuming that no mistakes will be committed.

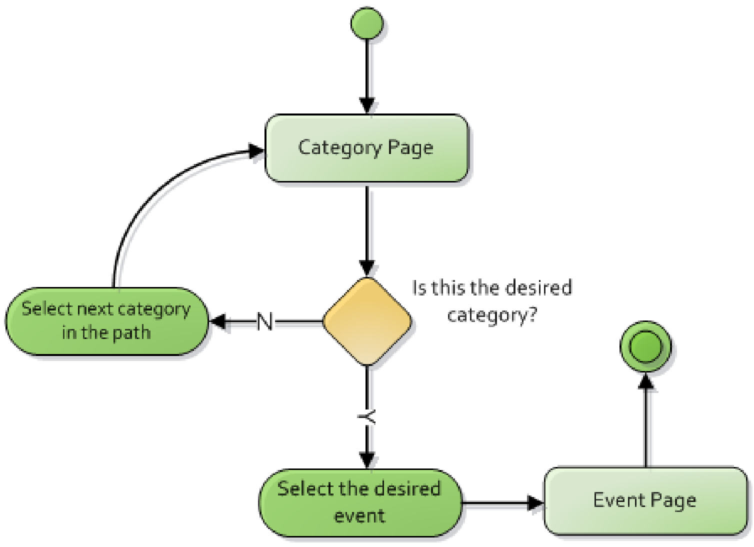

Category Page
Is this the desired category?
N
Select next category in the path
Y
Select the desired event
Event Page

Figure 11 - Browsing for an event page

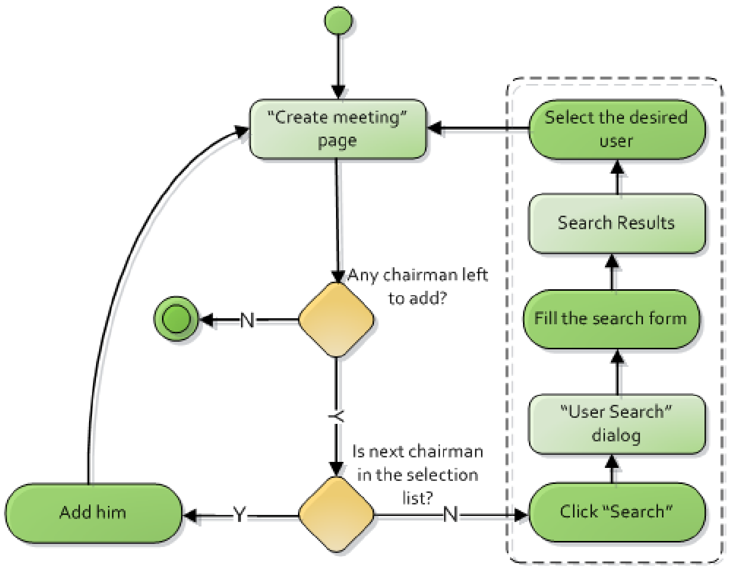


Figure 12 - Adding several chairmen to a meeting

Another troubling case was the addition of chairmen for a meeting. The process is described in Figure 12, and the highlighted area refers to the sequence that the user has to repeat each time he wants to add someone that is not in the selection box for chairman. This box will only contain the users who have been chairmen inside the category the meeting is being created in.

Each transition from a rounded box to a square one represents a page transition. For each added chairman, three page transitions are needed, and at least the same number of clicks as well. This was definitely one of the processes that required a refactoring effort, in order to make the life easier for the user.

2.2 The Accessibility Issue

According to (Wikipedia contributors, 2007a), “accessibility is a general term used to describe the degree to which a system is usable by as many people as possible.” In the case of Indico, the narrower field of “web accessibility” is the criterion. The W3C defines “web accessibility” as the degree at which “people with disabilities can perceive, understand, navigate, and interact with the Web, and that they can contribute to the Web” (W3C, 2005). The concept of “disability” encompasses different types of conditions which may affect some users:

- Visual disabilities (from blindness to color blindness);
- Hearing impairments;
- Physical disabilities;
- Speech disabilities;
- Cognitive/neurological disabilities;
- Aging-related conditions;
- Multiple disabilities (any combination of the aforementioned ones);

It is important to notice that “accessibility” is not only a matter of “obvious” impairments (deafness, blindness…), but also of some usually overlooked conditions, such as the so-called “learning disabilities” (i.e. Down Syndrome), or lighter impairments, such as dyslexia. This calls for changes not only in the way information in presented, but also in the semantic organization of web sites. Complex schemas and language should be avoided, and coherence between the different parts of the site should be enforced.

The main document providing guidelines for web accessibility is the W3C’s “WCAG - Web Content Accessibility Guidelines” (W3C, 1999b), which was used as a checklist for the report on Indico’s usability and accessibility (Ferreira, 2007b). The latter’s conclusions suggest that, at the time, the degree of accomplishment of the guidelines was not satisfactory, for a tool such as Indico, aimed at providing contents for a broad audience. The main issues were:

- Absence of semantic information – there was an excessive emphasis on the format, instead of the actual meaning of the information conveyed by the system. HTML provides tags and attributes which help defining mutually-associated concepts and additional information which can be used, for instance, by screen readers. Indico was not using them, most of the time.
- Deprecated HTML and insufficient conformance to the W3C standards – some obsolete HTML attributes were used, and CSS was not completely valid;

- Excessive layout complexity – tables were intensively used for layout purposes, information. Templates were "design-centric" instead of "information-centric". This means that the visual results would be acceptable, but information harvesting would be much more difficult.

It was clear that Indico failed in some of the fundamental premises for a satisfactory accessibility.

2.3 The Problem of User-centeredness

Good usability standards are only one of the premises for a good "user-centeredness". The latter is a concept that encompasses different disciplines, and may be seen through different prisms (Iivari, et al., 2006). One of them is that of "user-centeredness as personalization". In fact, personalization is a good solution for adapting "the system's content structure, presentation form and functionality to each user's characteristics, use behavior and usage environment" (Iivari, et al., 2006). The concept of "user profile" is not new, but, in the last years, the "Web 2.0" (O'Reilly, 2005) phenomenon has substantially raised its meaning: the user "profile" is not a set of simple configuration settings, anymore, but also a space where he/she can place information and share it with the rest of the community.

Indico possessed a very primitive user personalization feature set. Besides some basic user data (affiliation, email, password, address, etc…), the user was not able to customize the behavior of the system in any way. In addition to that, a deeper problem was verified: there was no place where the users could keep track of their own things: i.e. no way to know what conferences they were registered to, what meetings they would attend, or which events they had created themselves. This was a serious issue, since people would have to know by heart what events to search, and where to search them. The "Indico Usability Survey" provided some clues about this problem: roughly 30% of users rated "Lack of personalization features" as one of the worst defects of Indico, while "Content Organization" stood with approximately 27%.

3 Technological Review

This section has the purpose of providing an introduction to some of the technologies (all of them in some way related to the Web) that will arise in the next section, as part of the solution for the problems mentioned before. The approach will be sequential, starting with the “traditional” web technologies, and finishing with ideas which are as recent as from this project.

3.1 The “old” Web – REST Architecture

From the beginning, the World Wide Web has adopted a “Representational State Transfer” (REST) architectural style, although this term was only coined in 2000, by Roy T. Fielding (2000):

«The name "Representational State Transfer" is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through the application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use. » (Fielding, 2000)

The REST Architectural paradigm has the following characteristics:

- Stateless server/client core architecture;
- A small and well-defined set of methods for resource transfer (HTTP POST/GET and PUT/DELETE);
- Different media types (HTML, different types of images, plain text, etc…);
- Explicit invocation of remote actions;
- The static resources and application states are identified through unique URLs, which allow bookmarking and link-sharing;
- Hyperlinks are used as the base for application-state transitions;

This architecture, in spite of being consistent, scalable and fully tested during the last years, presents some limitations that in some ways make it difficult to implement new application models that have evolved recently:

- The Stateless nature of HTTP and web servers/clients – It is widely accepted that web applications require some kind of internal state, which is usually implemented at a higher level (through cookies or POST/GET fields);
- The purely explicit nature of the HTTP protocol forces synchronous communication: the server cannot asynchronously transfer information to the client;
- Most browsers don’t implement requests such as PUT, thus forcing the use of POST requests for the same end. This “corrupts” the implementation of the REST paradigm, in some way;

Given these limitations, new architectural patterns had to be found for the implementation of better, more complex applications.

3.2 Rich Internet Applications

The concept of "Rich Internet Application" was introduced in 2002, by the now defunct Macromedia Inc. (which is nowadays a part of Adobe[19]), as a description of a web application with the features and functionality of a normal desktop application (Allaire, 2002). Although the concept was, at the time, meant to separate itself from HTML/ECMAScript based applications, it is now commonly associated to DHTML as well.

There are several technologies capable of building RIAs:

- DHTML (HTML/ECMAScript) and AJAX – widely used, cross-platform, although somewhat limited. It has the advantage to keep (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 and not cross-platform;

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 ECMAScript. On the other hand, it allows one to build a system which is completely HTML-based (accessible for older browsers), providing, in parallel, some improvements for modern clients.

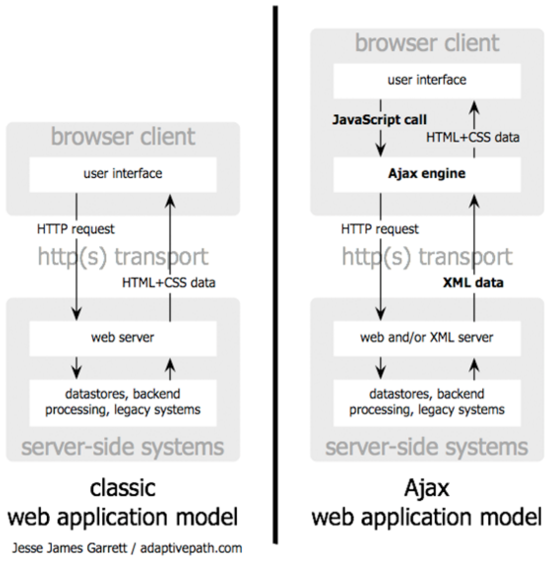


Figure 13 - Classic web application model vs. AJAX

(From (Garrett, 2005)*)*

3.3 Asynchronous JavaScript and XML

According to (Wikipedia contributors, 2007b), it was Microsoft (in 1996), who introduced the notion of asynchronous requests, through the *IFRAME* tag, back in Internet Explorer 3. But it was 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 (2005) that the term "AJAX – Asynchronous JavaScript and XML" was coined. The sudden "boom" of AJAX applications in the last two years may be explained through the following factors:

[19] http://www.adobe.com

- 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[20], of the *XMLHttpRequest* API, which opened the way for cross-browser utilization of this technology;
- The availability of better cross-browser *ECMAScript* standards support;
- The so-called Web 2.0 phenomenon, which boosted the demand for rich and intuitive web applications;

As we can see (Figure 13), an AJAX application includes an additional layer (usually programmed using ECMA/JavaScript), which is responsible for the asynchronous requests to the server, and the retrieval of XML (or any other format) data, which are then translated to actions over DOM components. In the process point of view, this translates into the evolution displayed in Figure 14.

As can be easily noticed, there's a "continuum" in user interaction. This way, the user is not interrupted during the execution of data operations. It is obvious that this evolution will have a huge impact on usability matters.

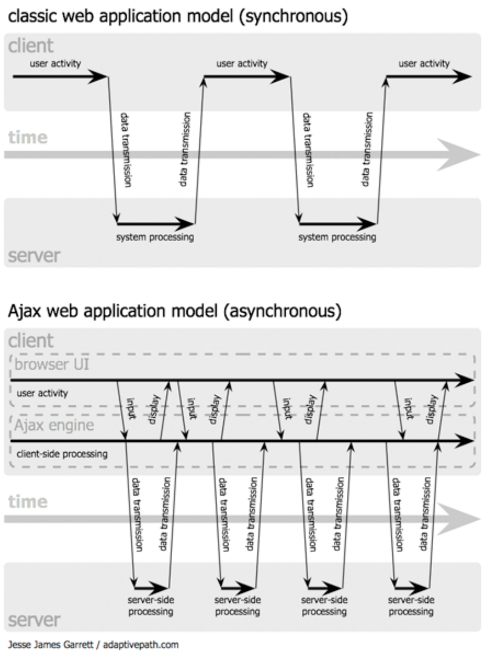


Figure 14 – AJAX in the process point of view

(From (Garrett, 2005)*)*

AJAX-based interfaces have several advantages when compared to traditional ones: they're completely asynchronous, which makes 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*[21] and *Docs & Spreadsheets*[22], 37signals' *Basecamp*[23] and Digg Inc.'s *digg*[24], 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 ECMAScript code is not read by web crawlers, as they usually do not execute scripts;

[20] "Mozilla.org" - http://www.mozilla.org

[21] http://www.gmail.com

[22] http://docs.google.com

[23] http://www.basecamphq.com

[24] http://www.digg.com

- 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;

3.4 Dynamic HTML

Dynamic HTML is powerful enough for the needs of Indico, but it is based on ECMAScript, a standard that is implemented in slightly different ways by each browser. For this reason, the use of “pure” ECMAScript for DHTML pages is very difficult and error-prone. Fortunately, one has not to reinvent the wheel, since there are base libraries which provide a homogeneous layer for interaction with the browser components. The most recognized of them is Prototype[25], which includes utility functions for class-based object-oriented programming (emulating it on the top of the prototype-oriented ECMAScript OO paradigm), and basic AJAX support. Prototype is used by frameworks such as the famous “Ruby on Rails”[26].

The advantage of using DHTML as the basis for a web application comes from the possibility to create improved widgets and visual effects that can provide a better user experience. There are many ECMAScript libraries that provide components for rich applications. The most famous ones are:

- The Dojo Toolkit[27] - A very complete toolkit, which includes several form-related widgets, normally found in desktop applications (date input through calendar, improved combo boxes, in-place edition, spinners, sliders, layout managers, tab and stack containers and pop-up menus), ideal for “pure” RIA development. It is however, somewhat heavy, due to the large amount of code involved;
- Scriptaculous[28] – A lightweight toolkit including basic widgets (in-place edition, drag and drop and auto-completion) and visual effects;
- MochiKit[29] - Another lightweight toolkit, including essentially utility functions (color manipulation, date and time, string formatting, iterators, logging…), drag and drop support and visual effects;
- Kabuki Ajax Toolkit[30] - Distributed as part of the Zimbra Collaboration Suite, but also as a standalone product, this toolkit is focused on AJAX, but includes some DHTML features, such as drag and drop support, basic shape manipulation, and widgets: calendars, color pickers, list views, menus, tooltips…

All these products are open-source solutions, involving no additional costs for the project. The adopted one was Scriptaculous, for its simplicity, easy integration, and lightness. Scriptaculous was also already used in a specific module of Indico (badge/poster editor), and

[25] http://www.prototypejs.org/

[26] http://www.rubyonrails.org/

[27] http://dojotoolkit.org/

[28] http://script.aculo.us/

[29] http://www.mochikit.com/

[30] http://www.zimbra.com/community/kabuki_ajax_toolkit_download.html

was being used in the development of the new CRBS. These facts helped the decision of keeping Scriptaculous as the base framework for the DHTML/AJAX improvements in Indico.

3.5 The WOHL Format

The WOHL[31] (***W****eb-****O****riented* ***H****elp Specification* ***L****anguage*) format (Ferreira, 2007c) 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 the page templates.

The need for WOHL, and its definition and implementation (in the context of Indico) are described in the following sections[32].

[31] Pronounced as in German “wohl” (“well”): IPA [voːl]

[32] See 4.3 - “Context Help” and 5.3 – “On the WOHL Interleaver”

4 Proposed Solution

During the analysis phase, a set of twenty five requirements for a better usability/accessibility was identified[33]. From these, four were non-functional:

- Intuitiveness – Easiness of use, simplicity and good organization of the interfaces;
- Accessibility – Make it easier for disabled people to use Indico, through the accomplishment of the WCAG, and 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 twenty-one functional requirements identified, the most important were selected by the project manager, being them:

- 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;
- Creation of a personal area for users – which would include a list with the events he/she has some kind of relation with, and a list/basket of "favorite" users, for further use;
- Single-click (in-place) edition for form fields;

After the completion of these prototypes, and using the information gathered from the "Laboratory Test", it was possible to advance towards the most ambitious of the requirements:

- 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;

This section explains, in a sequential way, the solutions reached for the implementation of these requirements.

4.1 A New Service-based Architecture

In order to improve the usability and overall user-friendliness of Indico, something that would go beyond the REST Architecture was needed. It is true that one can build successfully usable and accessible websites using the synchronous paradigm (the Google Search Engine[34], for instance, or the internationally awarded SIGARRA[35] Information System, from University of Porto). However, Indico is a special case, for its overloaded management interface (the complaints about the excess of options in a single page, from several users), and its intensive

[33] The complete list is available in (Ferreira, 2007b)

[34] http://www.google.com

[35] http://sigarra.up.pt/up/web_base.gera_pagina?P_pagina=2418

use of forms for creation, edition and management of events. SIGARRA does well without a rich interface, since it doesn't require, at least at the regular user level, systematic submission of information; and a search engine like Google has a reduced number of options, being that, most of the times, only the "keyword" field is used. Perhaps the mainstream web application whose user experience is most similar to Indico's is *Google Calendar*, which is far from being a normal REST application. It makes intensive use of DHTML/AJAX, and uses a service-based architecture for client-server communication.

The already mentioned AJAX paradigm, described in (Garrett, 2005), 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;

However, there was some hesitation regarding the introduction of this paradigm at this time, since the project manager and group leaders were aware of the problems that the mass adoption of ECMAScript has brought to a large segment of the web: cross-browser compatibility issues and a consequent turndown on accessibility. Doubts would, however, dissipate, after the production of the first prototypes.

After deciding that a DHTML/AJAX based approach would probably be the best way to follow, there was the problem of adapting Indico to this new idea, without having to reformulate the whole system. On the other hand, the new improvements should have a simple logic, and require a small amount of code, or it wouldn't be possible to migrate the system in a realistic amount of time.

The "traditional" Indico

In the "traditional Indico", a simple user request passes through different layers before it is actually answered by the application. In Figure 15, a form from page *P* is submitted, using a simple POST/GET request. A simple request handler, programmed using the *mod_python* framework, answers the request, and forwards it to a specific Request Handler (inside MaKaC/Indico) which is responsible for processing the parameters, and sending them to the Web Page Generation Layer. This layer contains Python classes which describe each webpage: which components they contain, what kind of relationship exists between them, and which functional group they belong to. This is gracefully done through the traditional mechanisms of class inheritance and composition, and will be explained in greater detail in the following section[36].

The Web Page classes delegate the actual HTML generation procedure to "Template Generators", classes which are "authorized" to produce HTML, and do so with the aid of "template files", HTML files which include some template generation data, in a particular syntax. After the HTML is produced, it is returned all the way back to the client, where it is displayed as page *Q*.

[36] See 5.1 – "Implementing the Service-based architecture"

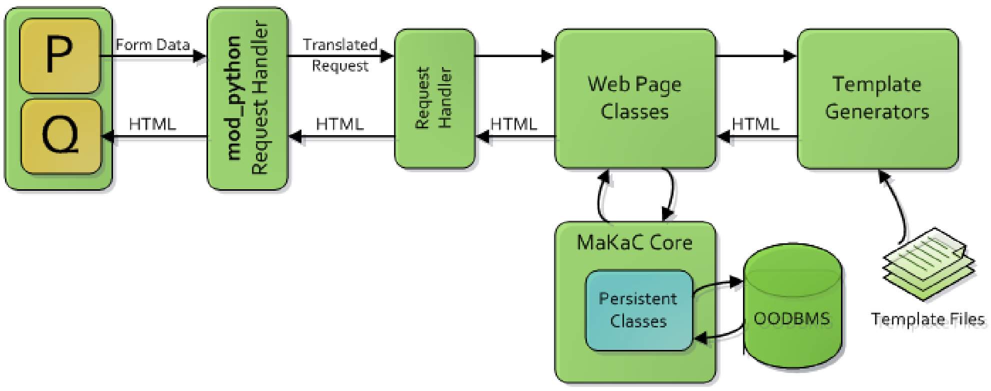


Figure 15 - The Data Flow for a request

This architecture provides a good degree of isolation between the different layers, and effectively avoids the most common problems of layer interleaving. However, it deserves some criticism:

- It takes too much work in order to create a simple page: one needs to create four different entities, so that all the four layers are connected;
- In the beginning, there was no way to create complex behavior in the templates (meaning that thing such as an "if" or a "for" were not feasible), only string injection was possible. Being so, HTML generation from complex structures had to be hardcoded inside the Template Generators. Recently, this has been changed, with the introduction of a PHP-like syntax for inserting Python code in the templates. However, most of the template code still uses the old model, and a great amount of HTML code is still contained inside Python classes. This is not advisable, and should be solved by a gradual migration to a new template syntax.
- By migrating all the templates from the old model to a new syntax, the Template Generators will lose their importance. Their only task will be to call a template file. Being so, this layer will be a candidate for suppression.
- A great amount of code work is required in order to pass a parameter from the request handler to the end of the chain. The variable must be passed from layer to layer, and this has to be done manually. Recently, a "hack" which allows access to the Request Handler from the template code was introduced, but it is not expected from the architecture, being as well a violation of the "layer isolation" principles.
- The request handlers directly deliver HTML pages. All the application is HTML-centric, and there's no easy way to translate data acquired from request handlers to other formats, without changing each one of them;

The case of SPIAR and the hybrid model

In (Mesbah, et al., 2007), the authors introduce the notion of SPIAR (*Single Page Internet Application Architecture*). This is only the formalization of a model which was introduced by some of the Web 2.0 applications mentioned earlier in this document[37].

[37] *Gmail* and *Google Docs and Spreadsheets*, for instance

However, other applications, like 37signals' "Basecamp" have adopted a sort of hybrid model, which unifies the benefits of both single-page and multi-page approaches. There are no known reasons to believe that Indico would benefit from the adoption of "pure" SPIAR, and a few not to do so:

- Indico provides highly organized and hierarchized information (similar to a tree), which is obviously best consulted through a multi-page interface;
- It is difficult to link to something inside a SPIAR. Since Indico needs to provide URLs for specific events, this would introduce an unnecessary degree of complexity;
- Indico is multi-page. It would be hard to make it single-page, and there seems to be no reason to do so;

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 16 introduces an "hybrid" architectural model for Indico. The two 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;
- The deletion of the "Template Generators" layer, for the reasons mentioned above;

Unfortunately, a long time would be required for the "Template Generators" to disappear, and this layer will be, at least for now, kept as it is.

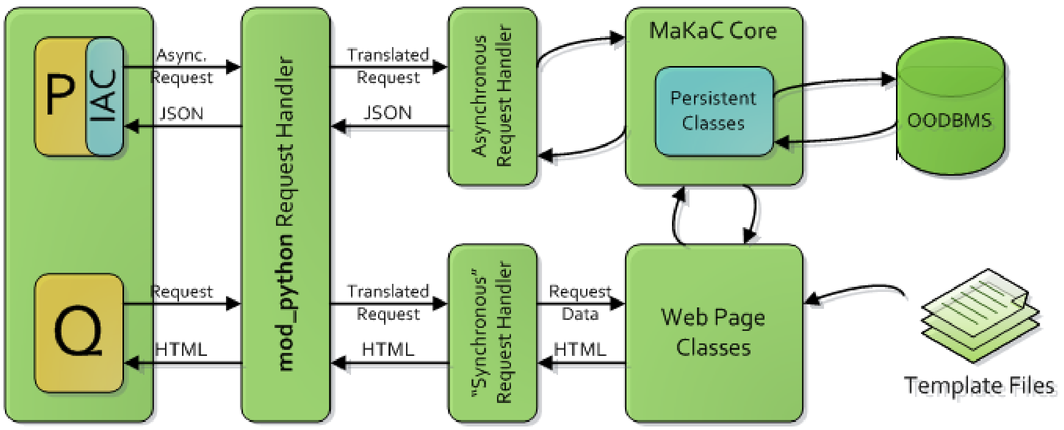


Figure 16 - Hybrid architecture for Indico (*Asyndico*)

At the client side, an ECMAScript 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[38] format. JSON was selected over XML, because it is lightweight, and libraries such as Prototype include support for serialization/unserialization of JavaScript objects as JSON. The asynchronous request

[38] JSON – JavaScript Object Notation – http://www.json.org

handlers would simply execute the desired operations, without the need for "middlemen", directly using the functionalities provided by the *MaKaC* core. Since the concept of "web page as a result of a request" disappears from an AJAX-based application, the web page-generating classes would gradually collapse, being replaced by simpler asynchronous request handlers, and some client-side DOM manipulation (provided by the IAC). On the other hand, a "basic" set of pages has to be generated (for the "regular pages", as explained above).

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 (Figure 17);
- "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;

However, these features would all be useless, without a set of widgets capable of making them presentable to the user.

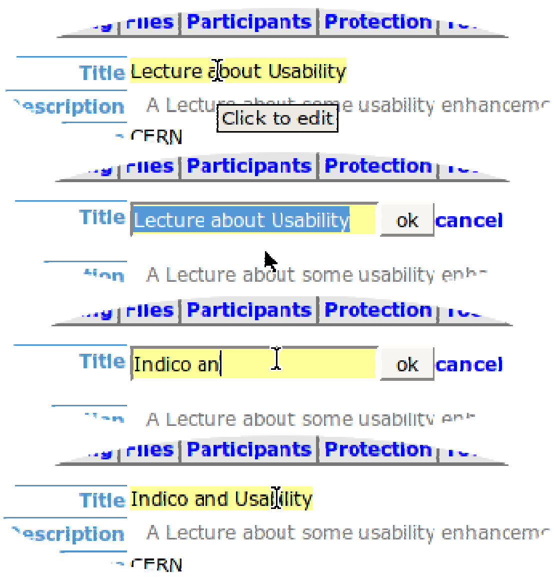
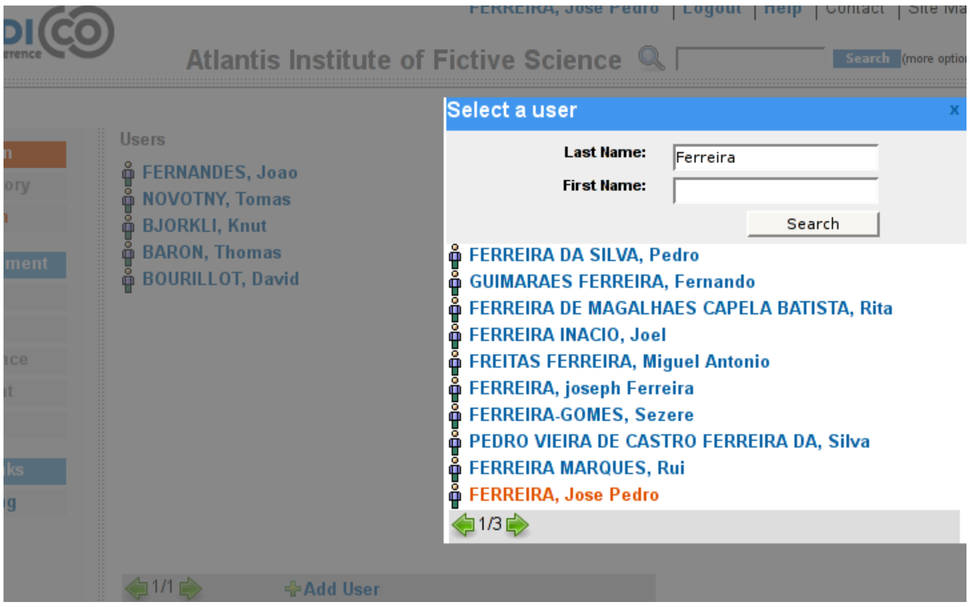

Files | Participants | Protection
Title Lecture about Usability
Description A Lecture about some usability enhancement
Click to edit
CERN
Title Lecture about Usability ok cancel
Title Indico an ok cancel
Title Indico and Usability

Figure 17 - In-place editing, as implemented in the *Asyndico* prototype (Firefox 2.0.0.7)

4.2 *IndicoUI* ECMAScript Library

In the previous subsection, the idea of *IAC – Indico Ajax Components* was introduced. This client-side "module" would not only take care of the interaction between the client and the Asynchronous Request Handlers (as referred earlier), but also provide a set of DHTML/AJAX based widgets, which would be easily embeddable in every Indico page, and easy to extend, so that new features could be adapted to this paradigm with minimal code replication and minimal effort. Based on this definition, IAC was named *IndicoUI*, the "*Indico User Interface ECMAScript Library*".

Besides the obvious requirement of in-place edition, there should be a widget whose existence would be a proof-of-concept for *Asyndico*. Based on one of the pathological cases[39] of usability, a simple user search widget was defined as a target. The widget should be a list view, capable of being populated with users. These users would be selected from a dialog which would pop up each time the user would click "add". The dialog would allow users to search using the name of the desired person.


FERREIRA, Jose Pedro | Logout | Help | Contact | Site Ma

Atlantis Institute of Fictive Science Search (more optio

Users
- FERNANDES, Joao
- NOVOTNY, Tomas
- BJORKLI, Knut
- BARON, Thomas
- BOURILLOT, David

1/1 Add User

Select a user

Last Name:	Ferreira
First Name:	

Search

- FERREIRA DA SILVA, Pedro
- GUIMARAES FERREIRA, Fernando
- FERREIRA DE MAGALHAES CAPELA BATISTA, Rita
- FERREIRA INACIO, Joel
- FREITAS FERREIRA, Miguel Antonio
- FERREIRA, joseph Ferreira
- FERREIRA-GOMES, Sezere
- PEDRO VIEIRA DE CASTRO FERREIRA DA, Silva
- FERREIRA MARQUES, Rui
- FERREIRA, Jose Pedro

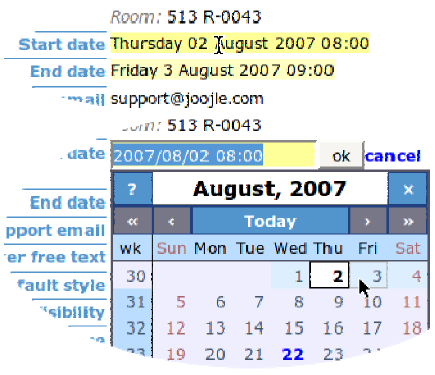
1/3

Figure 18 - Functional prototype of the user search (rendered by Firefox 2.0.0.4)

Figure 18 shows the final result of the user search widget: the list seen in background contains the selected users, and the search dialog, in the foreground, has a 3-page list of people with a common surname. This widget is a *UserSelection* widget (see Figure 20), which contains a search form and a *UserPickListPanel*, a user list component which allows users to be picked from it. The widget in the background is a *UserBasketListPanel*, easily noticed by the presence of the "add" button.

Some additional widgets include an in-place date editor and combo box.

The inheritance graph for the most important widgets can be seen in Figure 20.


Room: 513 R-0043

Start date Thursday 02 August 2007 08:00

End date Friday 3 August 2007 09:00

support@joojle.com

513 R-0043

date 2007/08/02 08:00 ok cancel

End date

pport email

er free text

fault style

sibility

```
?          August, 2007          ×
«    ‹         Today         ›    »
wk  Sun Mon Tue Wed Thu Fri Sat
30                   1   2   3   4
31   5   6   7   8   9  10  11
32  12  13  14  15  16  17  18
    19  20  21  22  23
```

Figure 19 - The in-place date editor, from the *Asyndico* prototype (Firefox 2.0.0.7)

[39] See 2.1 - "Pathological Cases"

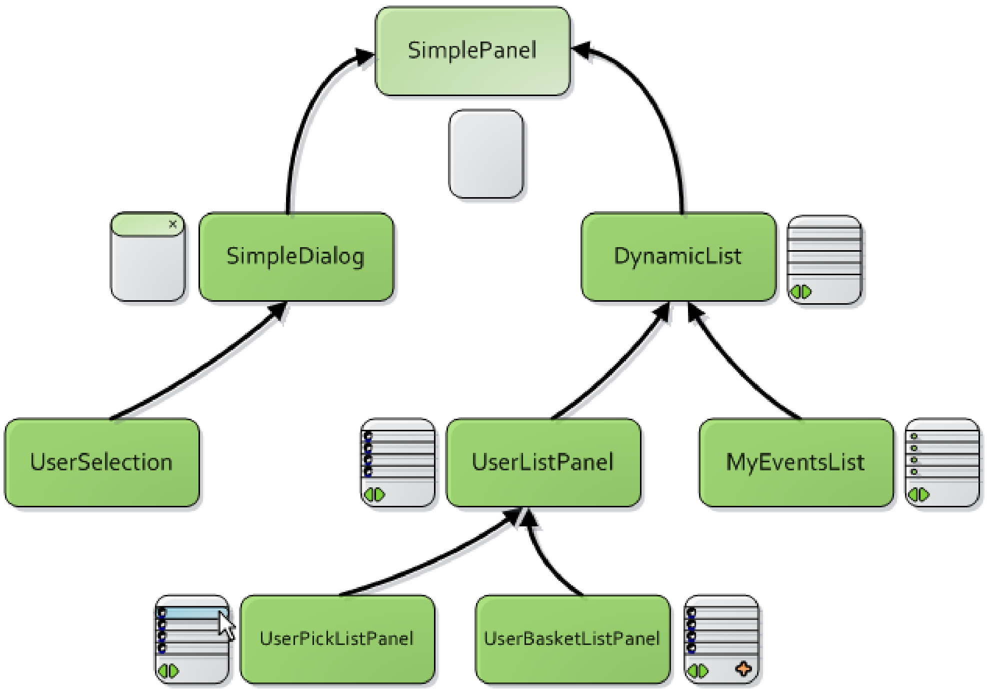


Figure 20 - Main IndicoUI Widgets

IndicoUI is based on *Prototype* and *Scriptaculous*. Together with the *Asyndico* architecture, it provides the foundations for an AJAX-powered Indico.

4.3 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 an 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 (going at the time of the writing of this document) of the Indico interface will 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 a 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.

Towards a Semantic Web

The Web is slowly transporting computational effort towards the client, and the Web 2.0 is the latest step in this direction, according to the W3C (2007), not only as a way of reducing application latency in RIAs, but also in an attempt to evolve towards a semantic, information-centric and rendering-independent repository of information (the so-called

"Semantic Web" or "Web 3.0" that Berners-Lee et al. (2006) talk about). The inclusion of XSLT processors in the major web browsers had already provided a tool which would help separating content from presentation, thus enabling a good degree of abstraction of information. In spite of its Turing-completeness (Kepser, 2004), XSLT is not the easiest language to deal with for most of the purposes. Direct (X)HTML generation, using page templates is still the technology that empowers the vast majority of web applications, and Indico is no exception. This is naturally hazardous in a perspective of availability of information for non-human processing, but solvable by alternative means (for instance, Indico makes all events available through RSS and iCal feeds, as well as MARCXML).

Modern browsers include no built-in capabilities for web page enrichment with Contextual Help. One of the reasons for this might be that there are no defined standards for this kind of information (either from the W3C or any other entity). One could argue that there's no need for it, since the contextual help mechanism is usually built in. In fact, contextual help may be considered a cross-cutting concern, a concept that traverses all the application, with no apparent isolation from the rest of the features. However, the limited scope of the context help structures needed by Indico makes it possible to establish a set of constructs that can be isolated from the normal, informative content, and specified in a separate repository. Current browser technology won't solve the problem of presenting this information: it will have to be "interleaved" with the regular content. However, some advantages arise from this.

- Maintaining the help content in the server side will be easier, and code scattering will be smaller than with hardcoded contextual help;
- In the future, if browsers ever provide the possibility of rendering this information without a previous "interleaving" process, the separation is already done, and it will be easy to translate the contents to whatever information exists at the time;
- This information can be specified in a technology-independent way, so that it can be used not only with (X)HTML, but other possible presentation formats;

The WOHL Format

It was said earlier that there is no current standard able to answer to the requirements of Indico in the area of contextual help. This area seems to be somewhat overlooked by technology providers. In this context, the necessity to create a custom-tailored solution for Indico quickly emerged.

As stated earlier[40], WOHL is an acronym for *Web-Oriented Help-specification Language*, and aims to be an XML idiom for context help specification, regardless of technologies or document presentation details. The only requirement is that the target document to be "enriched" with the context help structures be in an XML-based (or at least HTML-based) format. This is due to the fact that WHOL uses XPath for defining the points of the document where the contextual help structures will be "injected". Figure 21 shows the inputs and output of an HTML/WOHL interleaving engine: content and presentation information is stored in the HTML file, and context help structures, in a WOHL (XML) file. The interleaving mechanism injects the context help information in certain points of the

[40] See 3.5 - " The WOHL Format"

HTML document, and an “enriched” HTML file is produced. The way in which the new information is presented is not determined by the WOHL file, but by the WOHL/HTML interleaving engine itself.

WOHL includes (at the moment of the writing of this document) two different contextual help structures:

- Tooltips – Informative boxes that appear when the mouse pointer is placed over a particular area of the page. They can contain both textual content, and references (*linkref* tag) to static documentation. Tooltips can be either:
  - Explicit – An indication of the presence of the tooltip will appear, next to the target area, and the text will only be shown when the mouse pointer is placed over it; or
  - Implicit – There’s no indication shown: the text will appear when the mouse pointer is placed on the target area;
- Informative areas – Simple blocks of text, properly signaled as “help” content, that provide extra information to the user;

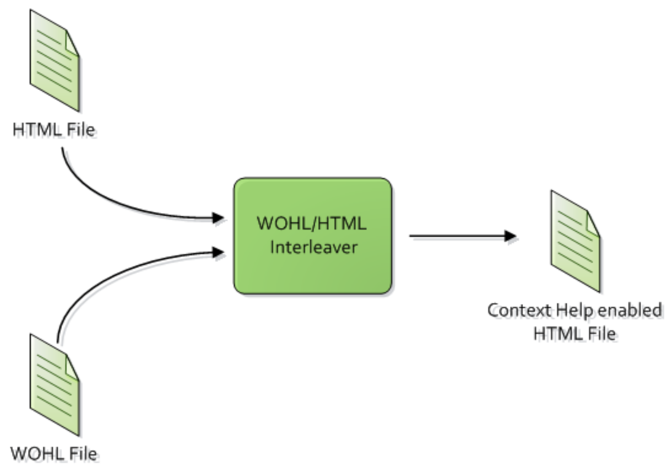
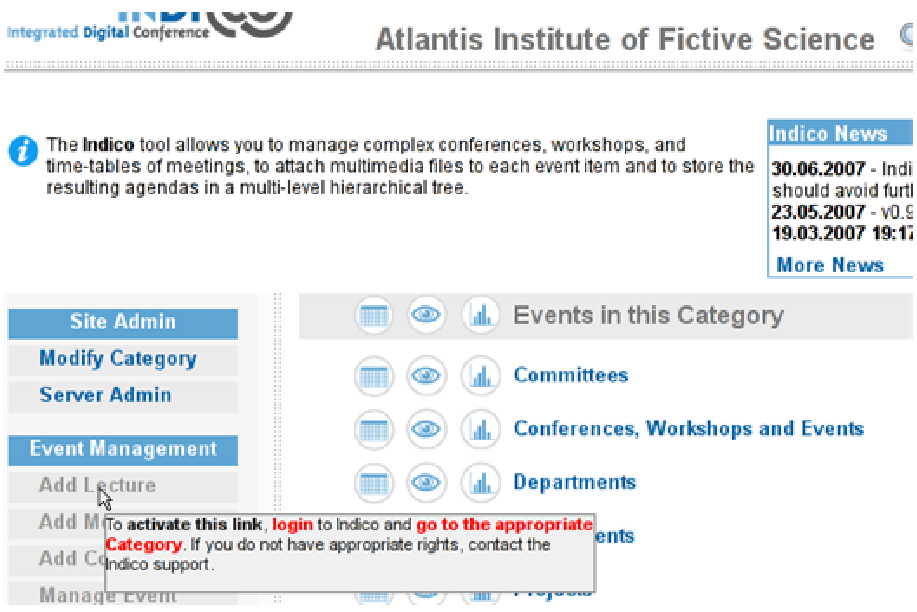


Figure 21 - The production of a Context Help enabled HTML page

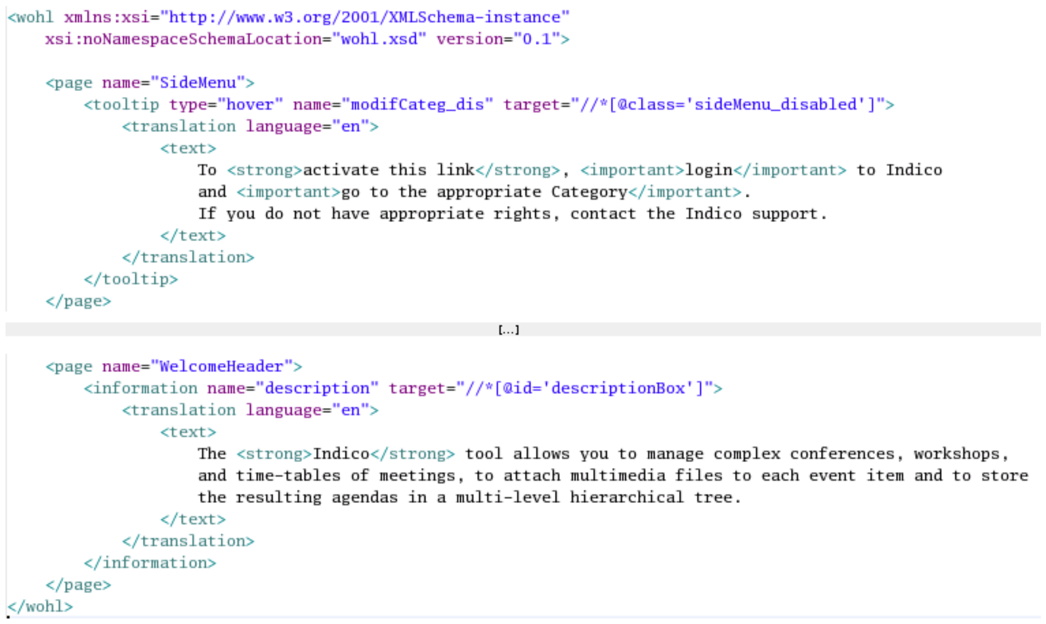
Figure 22 shows a screenshot of a WOHL-enabled template (as rendered by Indico’s WOHL engine), where an implicit tooltip is clearly visible, and an informative area (marked by the circled “i” icon) is displayed as well. The fragments of WOHL code that generate this output can be seen in Figure 23.



The **Indico** tool allows you to manage complex conferences, workshops, and time-tables of meetings, to attach multimedia files to each event item and to store the resulting agendas in a multi-level hierarchical tree.

To activate this link, **login** to Indico and **go to the appropriate Category**. If you do not have appropriate rights, contact the Indico support.

Figure 22 - Screenshot of a Context Help enabled page (rendered by Firefox 2.0.0.4)



```
<wohl xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="wohl.xsd" version="0.1">

    <page name="SideMenu">
        <tooltip type="hover" name="modifCateg_dis" target="//*[@class='sideMenu_disabled']">
            <translation language="en">
                <text>
                    To <strong>activate this link</strong>, <important>login</important> to Indico
                    and <important>go to the appropriate Category</important>.
                    If you do not have appropriate rights, contact the Indico support.
                </text>
            </translation>
        </tooltip>
    </page>
                                          [...]
    <page name="WelcomeHeader">
        <information name="description" target="//*[@id='descriptionBox']">
            <translation language="en">
                <text>
                    The <strong>Indico</strong> tool allows you to manage complex conferences, workshops,
                    and time-tables of meetings, to attach multimedia files to each event item and to store
                    the resulting agendas in a multi-level hierarchical tree.
                </text>
            </translation>
        </information>
    </page>
</wohl>
```

Figure 23 - Some WOHL fragments

WOHL also provides an XML-based syntax for rich textual content, once again based on semantics instead of the final look:

- **<strong>**…**</strong>** – for expressions which require some kind of highlighting, for their meaning (i.e. keywords);
- **<em>**…**</em>** - for expressions which require emphasis, or some simple differentiation from the rest of the text (i.e. citations);
- **<important>**…**</important>** - for expressions which require a high degree of attention from the user, for their high importance;

More information about the WOHL format (as well as it XSD specification) may be obtained from (Ferreira, 2007c).

4.4 The Search Option

From the identification of the event search/browsing process as one of the “pathological cases” of Indico’s usability[41], it became obvious that a better search mechanism was required. Besides the usability problems mentioned previously, there were a few issues that the search feature suffered from:

- There was no integration at all with Indico: the search option was just a link for a search page, defined in the application configuration options. At CERN, the CDS Invenio server *indicosearch.cern.ch* was used, with a custom look and feel, so that it looked as if it were inside Indico. Invenio would get the data from Indico by harvesting it using OAI-PMH;
- The previous point made it impossible to place search boxes embedded in the web interface, as there was no common interface between Indico and the search engine;

This “loose coupling” between Indico and the search engine was meant to be kept, since Indico had no manpower, nor time to produce its own search engine, and Invenio was a mature in-house solution, precisely focused on the task of harvesting and indexing scientific content, and thus a natural candidate for doing this complex work for Indico. On the other hand, Indico could not advance onto a deeper degree of integration with Invenio, since other institutions working with Indico should be able to use their own search engines as an alternative to the former.

The best solution to keep this “loose coupling” and improving the degree of integration with the search engines at the same time came through the concept of “Adapter software design pattern”[42]. The situation was clearly calling for it: there was the need to provide a translation between two different interfaces: a standard one, provided by Indico, and a specific one, different for each search engine. These “interfaces” were called “Search Engine Adapters” (SEA).

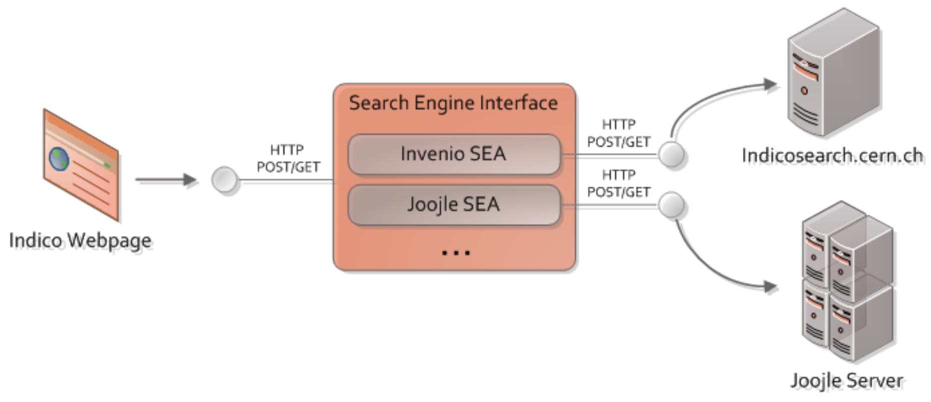


Figure 24 - How the Search Engine Adapters integrate with the puzzle of Indico

Figure

[41] See 2.1 - “The Usability Problem”

[42] GoF 139 - (Gamma, et al., 1995 p. 139)

24 illustrates, in a simple way, how the concept of SEA integrates with Indico and the *Asyndico* paradigm in particular. The new Search Engine Interface receives information about the query to be done on the search engine. Then, it uses the correct SEA to translate this request's parameters to those of the target search engine. The SEA which will be used is defined in the Indico server configuration file. The default SEA implementation is tailored for Invenio, but the architecture of the Search Engine Interface was made in such a way that it is very easy to create an adapter for virtually whatever search engine. Figure 24 presents a second SEA, the so-called "Joojle SEA", imagined for a phony search engine.

The other side of the problem was deciding what to do with the obtained search results. Ideally, Indico would gather the data in a machine-readable format, and present it through its own interface. However, several reasons required a compromise on this point:

- Invenio has a good web interface, content-oriented, and capable of aggregating resources according to the events they belong to;
- The presentation of the information inside Indico was not considered a priority by the project management, and would require a considerable additional effort for its implementation;

Being so, it was decided that, for the short term, the search results would be display by the search engine itself, without any kind of processing done by Indico.

The new Search Engine Interface was integrated in the web interface, as from the 0.94 version[43]. It allows users to perform a global search, or to seek contents inside a specific category or event.

Figure 25 shows the "search box" that was introduced. Using DHTML, a panel with additional search options would open.

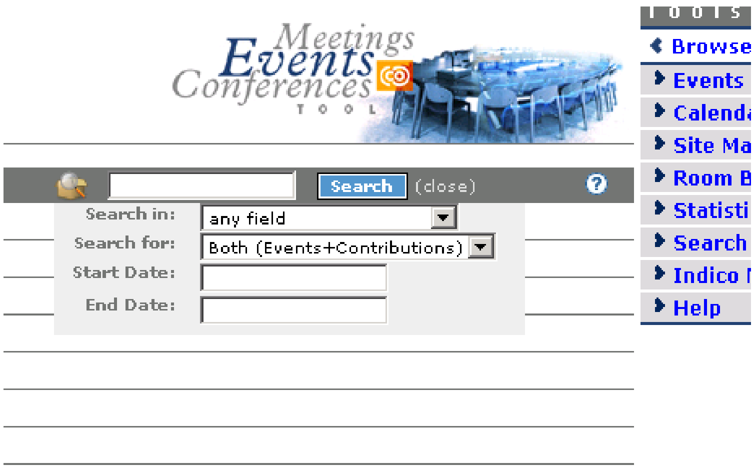


Figure 25 - The search box, as in Indico 0.94, with the "more options" dialog open (rendered by Internet Explorer 7.0)

[43] put in production in May 2007 and released in August 2007

4.5 Personal Features

The already stated requirements for a better usability clearly mentioned a "personal area" where users could keep track of the events they were involved in, and manage their "favorite users. This feature was prototyped in two separate parts:

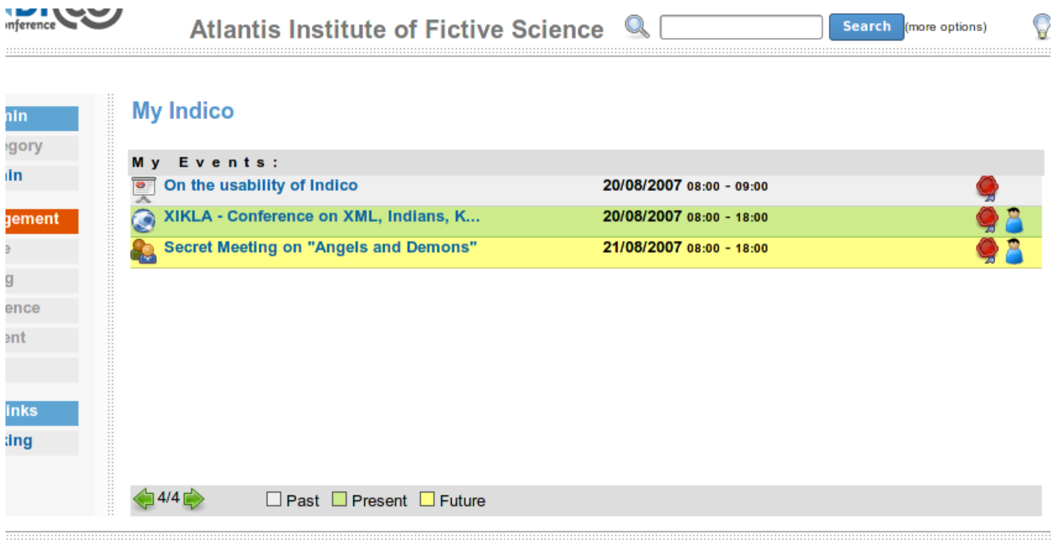
- My Indico – The list of events the user is related to in some way;
- User Baskets/Favorites – The list of favorite users, and the mechanisms used to manage it;

These modules have been completely based on the *IndicoUI* library, and the *Asyndico* architecture.

My Indico

The "My Indico" area is supposed to be the "starting point" for the less experienced Indico user. It should as well provide a quick "shortcut" for even those who know perfectly how to find what they want. The idea is very simple: providing a summary of the events the user is associated with, so that he can quickly check what will happen next, and immediately access the event pages.

The core of this personal feature is the "My Events" widget, from the *IndicoUI* toolkit[44]. It is implemented by the *MyEventsList* class, which provides an user-friendly list of events, with past, present and future event colored in different ways, and icons describing the kind of event, and the "degree of participation" of the user.



My Indico

My Events:

On the usability of Indico	20/08/2007 08:00 - 09:00
XIKLA - Conference on XML, Indians, K...	20/08/2007 08:00 - 18:00
Secret Meeting on "Angels and Demons"	21/08/2007 08:00 - 18:00

4/4 ☐ Past 🟩 Present 🟨 Future

Figure 26 - The "My Events" widget, as rendered by Konqueror 3.5.7

Figure 26 shows an example of this widget. The user is the creator of the three events, and is registered as participant in the last two ones. The icons are not final versions, since they will be integrated in the current theme, as soon as this set of new prototypes are adopted as new features. Through this dialog, users can easily verify:

[44] See 4.2 - "*IndicoUI* ECMAScript Library"

- The title, date and hour of the event, as well as its position in time, relative to the present moment (past, present, future);
- The type of event:
  - Conference;
  - Meeting;
  - Lecture;
- Their degree of participation in the event:
  - Creator of the event – has created the event;
  - Participant – is registered as participant in the event;
  - Manager – has the right to modify the event;

The introduction of this feature required some modifications in the way Indico deals with users and events. These will be explained below[45].

User Baskets (Favorites)

The "User Baskets" feature was introduced as a way for a user to store his "favorite other users", or the ones he uses most frequently. This is particularly interesting for people doing administrative work for departments and experiments at CERN, since they are usually required to specify users from a small set of names (usually belonging to the division they work at), for several consecutive times, so that they can create group meetings, or lecture cycles. The new "favorites" feature substantially reduces their workload, since they can select a small set of names that will be instantly available when they want to create a meeting. The term "basket" was borrowed from CDS Invenio, since this system already has an implementation of baskets for document storage. However, the recent developments on the usability studies have suggested that the term "baskets" should not be used in the final version of the interface, since it is semantically connected to "shopping baskets" or "shopping carts", and may cause confusion to the user. For this reason, the term "favorites" will be employed in the final user interface.

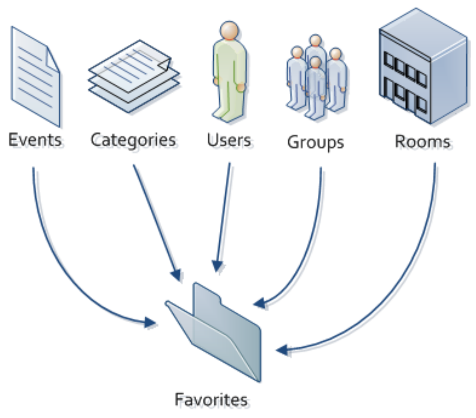
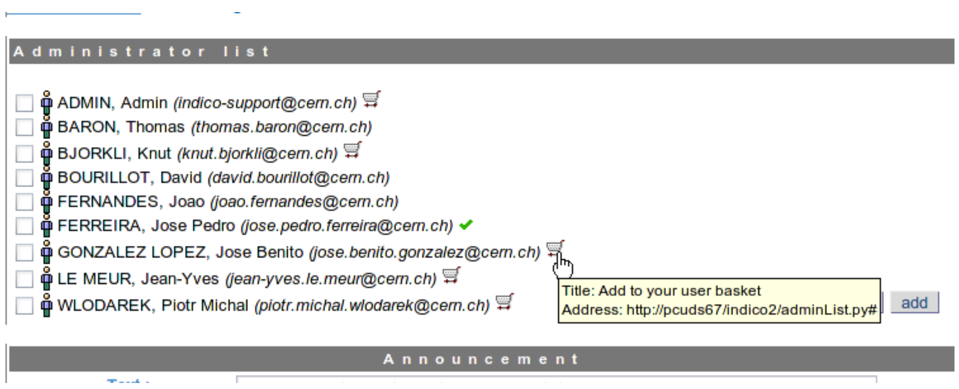


Figure 27 - In the future, the user will be able to store different entities in his favorites. For now, it is possible to do it with other users only.

This idea may, as well, be applied to other contents: events, categories, user groups and rooms, for instance. These would, as well, be very useful for different classes of user. For the prototype of the "favorites" feature, only baskets of users were considered at the interface level (since the introduction of these features requires several changes at the interface level, and the "users" feature had the highest priority).

[45] See "Personal Features" (pg. 30)


Administrator list

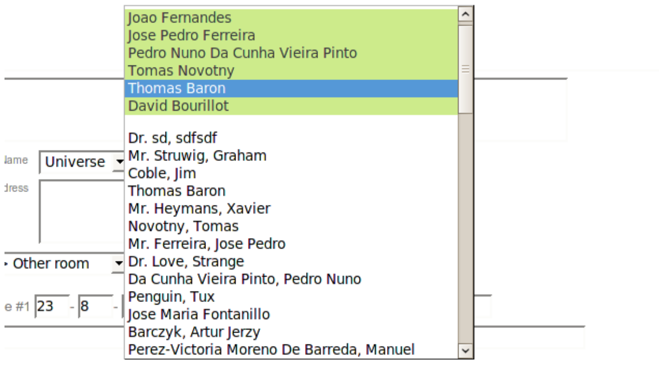
ADMIN, Admin *(indico-support@cern.ch)*
BARON, Thomas *(thomas.baron@cern.ch)*
BJORKLI, Knut *(knut.bjorkli@cern.ch)*
BOURILLOT, David *(david.bourillot@cern.ch)*
FERNANDES, Joao *(joao.fernandes@cern.ch)*
FERREIRA, Jose Pedro *(jose.pedro.ferreira@cern.ch)* ✓
GONZALEZ LOPEZ, Jose Benito *(jose.benito.gonzalez@cern.ch)*
LE MEUR, Jean-Yves *(jean-yves.le.meur@cern.ch)*
WLODAREK, Piotr Michal *(piotr.michal.wlodarek@cern.ch)*

Title: Add to your user basket
Address: http://pcuds67/indico2/adminList.py#

Figure 28 - The "Add to your user basket" icons (seen through Opera 9.23)

One of the problems the "favorites" feature brought with it was the best technique to add new users in a simple and fast way. The "traditional" method would be at the "basket management page", through a "search" dialog (already seen in Figure 18). However, users should as well be able to add a desired favorite user, at the moment they saw it. Memorizing or writing down a name, going through the interface, performing the search, and filtering the results would be a process that could keep some users from using the feature. Being so, the "Add to you user basket" icons were created (Figure 28). These are small icons (a shopping cart, in the prototype interface), which, when clicked, will add the desired user to the favorites list. After the element is added, they turn into a green checkmark, providing a visual feedback.

This feature would be of no use if no-one could use the stored users for something. In this case, the favorites are added to every user selection box in the interface, and placed at the top of the list, with a green background (Figure 29).


Joao Fernandes
Jose Pedro Ferreira
Pedro Nuno Da Cunha Vieira Pinto
Tomas Novotny
Thomas Baron
David Bourillot

Dr. sd, sdfsdf
Mr. Struwig, Graham
Coble, Jim
Thomas Baron
Mr. Heymans, Xavier
Novotny, Tomas
Mr. Ferreira, Jose Pedro
Dr. Love, Strange
Da Cunha Vieira Pinto, Pedro Nuno
Penguin, Tux
Jose Maria Fontanillo
Barczyk, Artur Jerzy
Perez-Victoria Moreno De Barreda, Manuel

Figure 29 - The favorite users, in green, at the top of a user selection box (Firefox 2.0.0.7)

4.6 Standards Compliance

(InDiCo Project, 2002) has made it clear that standard formats should be used whenever possible, explicitly indicating HTML and XML as examples. However, during the last years, the quality of HTML code has gradually degraded, eventually putting Indico out of the strict group of HTML standard-compliant web applications. A careful validation of the HTML syntax before each release (an easy procedure, since several automatic validators are available) could have avoided this, but the absence of this practice, accumulated through different revisions of the web interface done by different developers, has made Indico non HTML 4.01 Transitional compliant. It should be remembered that HTML 4.01 Transitional is not the most "orthodox" implementation of the standard, and supports various obsolete tags. In spite of that, Indico is not able to fulfill this essential requirement for a good usability level and browser-compliance.

Following this line of thought, it was decided that each improvement introduced by this usability enhancement project, and thus described in this document, would accomplish in the best way possible all the existing standards and recommendations, from either the W3C or other sources. In addition, all the improvements done to the date of writing of this document have been done according to the HTML 4.01 Strict standard, in an effort to bring Indico closer to the modern XHTML standard.

Since the usability enhancement project includes a refactoring of the general layout of the application, the most serious issues are expected to be corrected as a consequence of it. It has been proposed, as well, to the development team, that every future improvement of the Indico interface be checked by an HTML validator. The same applies for auxiliary formats, such as CSS.

4.7 New Layout

The current web interface that Indico provides to its users suffers from some usability/accessibility problems already described above[46]. After the realization of the "laboratory" usability tests, and the subsequent production of the findings document (Mondardini, 2007b), Mrs. Mondardini has offered to design an improved layout for Indico, according to the usability and accessibility guidelines, and having in mind the conclusions of all the studies realized to the date. The project manager and the development team promptly welcomed, once again, Mrs. Mondardini's collaboration, and the opportunity to benefit from the help of someone whose expertise is the field of design and usability. It would be, without any doubt, a great development for Indico's user-centeredness, and a breath of fresh air into the somewhat developer-oriented interface design process.

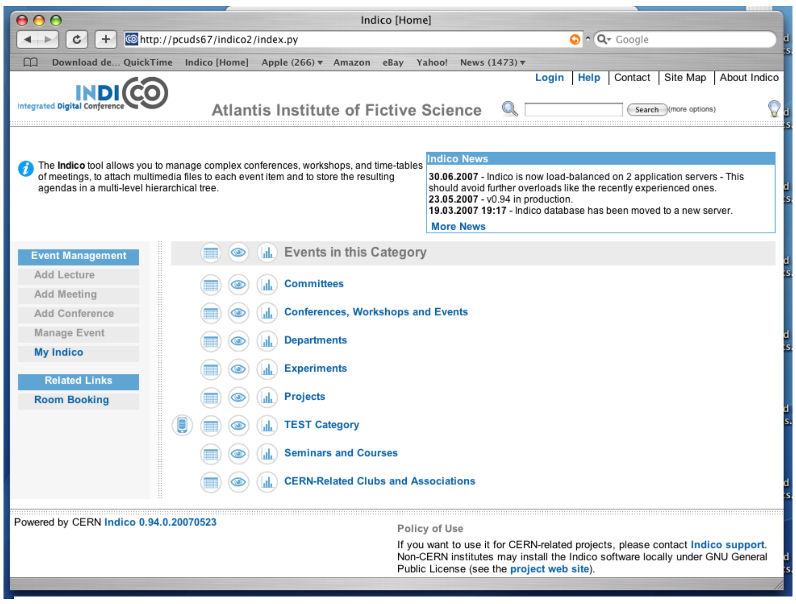
Mrs. Mondardini quickly prototyped the new interface, and presented it to the development team, which did not hesitate in adopting it. The main changes were:

- 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 implementation of the new interface has immediately started, and, at the date of writing of this report, the main interface pages have been migrated to the new layout.

Besides being HTML 4.01 Strict compliant, the new interface layout follows the guidelines defined by (W3C, 1999b), especially in what concerns to the presentation of information to screen readers. 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.

[46] See 2.1 – "The Usability Problem" and 2.2 – "The Accessibility Issue"


Indico [Home]
http://pcuds67/indico2/index.py
Google
Download de... QuickTime Indico [Home] Apple (266) Amazon eBay Yahoo! News (1473)
Login | Help | Contact | Site Map | About Indico
INDICO Integrated Digital Conference
Atlantis Institute of Fictive Science
Search (more options)
Indico News
More News
Event Management
Add Lecture
Add Meeting
Add Conference
Manage Event
My Indico
Related Links
Room Booking
Events in this Category
Committees
Conferences, Workshops and Events
Departments
Experiments
Projects
TEST Category
Seminars and Courses
CERN-Related Clubs and Associations
Powered by CERN Indico 0.94.0.20070523
Policy of Use

The **Indico** tool allows you to manage complex conferences, workshops, and time-tables of meetings, to attach multimedia files to each event item and to store the resulting agendas in a multi-level hierarchical tree.

30.06.2007 - Indico is now load-balanced on 2 application servers - This should avoid further overloads like the recently experienced ones.
23.05.2007 - v0.94 in production.
19.03.2007 19:17 - Indico database has been moved to a new server.

If you want to use it for CERN-related projects, please contact Indico support. Non-CERN institutes may install the Indico software locally under GNU General Public License (see the project web site).

Figure 30 - The main page for Indico's new layout, as rendered by Safari 2.0.4

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;
- Opera 9 (Presto layout engine);

The layout is also easily displayed by text-mode browsers, since markup elements such as headers and lists are being heavily used, so that most pages become functional even with CSS turned off.

4.8 Other Features

Acronyms and Abbreviations

From its birth, Indico was mostly used in the context of High-Energy Physics. This brings an additional degree of complexity for the user: we can say that one doesn't need to be a "rocket scientist" to be able to navigate through most Indico instances, but a reasonable knowledge on particle and accelerator physics is required. Most users won't realize that the "PAF/POFPA WG" category means "**P**roton **A**ccelerators for the **F**uture/**P**hysics **O**pportunities for **F**uture **P**roton **A**ccelerators **W**orking **G**roup". This may sound as a useless concern for most physicists or applied engineers that work at CERN, since they all know what these acronyms mean. However, it must not be forgotten that Indico is a web application, which provides public access to most of its contents, and not only a simple schedule-like software for white-bearded scientists to store their contents. Since there is no association between acronyms and their actual meaning, a user interested in the subject of “Proton Accelerators” will possibly not succeed in finding the content he/she looks for.

Nowadays' browsers already provide some mechanisms, specified in the (X)HTML standards, for displaying some auxiliary semantic content. This kind of information is as well useful for non-human agents, such as screen readers and crawlers. Some examples of this concept are:

- Acronyms and Abbreviations ((X)HTML's *<acronym />* and *<abbr />*[47]) ;
- Titles for images and lists[48];
- Association between labels and fields, in forms[49];

These mechanisms are also recommended by the W3C for accessibility enhancement (W3C, 1999b).

4.9 Conclusion

The solutions introduced above were considered the most adequate for the problems that Indico posed. Most of them derive from the emerging paradigm of the “dynamic web” and employ the technologies commonly associated with it.

After finding this set of solutions for the implementation of the requisites which had been stated previously, it was expected that working prototypes could be rapidly produced, in order to prove the feasibility and reliability of the aforementioned ideas.

[47] See (W3C, 1999a, 9.2.1)

[48] See (W3C, 1999a, 13.2)

[49] See (W3C, 1999a, 17.9.1)

5 Prototype Implementation

The implementation of the enhancements described in the previous section was realized in the form of autonomous prototypes. This section will describe some aspects on the implementation of the following ones:

- *Asyndico* service-based architecture – How the new asynchronous services implement several mechanisms, such as access control and communication between server and client;
- *IndicoUI* ECMAScript Library – The need for a templating mechanism;
- Context Help:
    - The WOHL/HTML Interleaving mechanism – How HTML injection is done;
- Search Engine Interface – Implementation of the Search Engine Adapters, and the InvenioSEA in particular;
- Personal Features – Modifications in the Indico data structure:
    - “My Indico”;
    - “User baskets”;
- New site layout – Things implemented so far, and implementation concerns;

At the end, there will be an evaluation of the implementation status for the various features.

5.1 Implementing the Service-based architecture

As it was explained in 4.1, the first layer in Indico, from the point of view of an incoming request, is composed by the so-called “Request Handlers” (gently denominated “RH-classes” by the developers), which take care of processing the parameters and calling the web page generation classes, the next step in the request handling process.

Since the implementation of a service-based asynchronous architecture requires the request handler to process the request (probably encoded in JSON, although encapsulated inside an HTTP request, as it is the case with Indico) and return the result in a machine-inteligible format (JSON, once again) as well, it became evident that the “plain old request handlers” would not be suitable for the desired effect, as they are done in such a way that the response is directly sent as HTML. This situation implied one of two solutions:

- “Porting” the required Request Handlers to the new architecture, and creating some kind of “adapter” layer which would provide the “old” interface to the “old” pages.
- Creating new, asynchronous request handlers which would exist in parallel with the “old” ones, and slowly “migrate” the “old” pages to new ones.

The second option seemed the best one, since there would be no creation of “disposable code” (the “adapters” that sooner or later would be removed for the sake of consistency), and a pacific coexistence of the two models, without interferences. This has motivated the creation of the *MaKaC.webinterface.async_rh* module, which would contain the first asynchronous Request Handlers, in the same way that *MaKaC.webinterface.rh* contained the “traditional” ones.

The Problem of Security

One of the problems in the implementation of a service-based architecture is guaranteeing that no-one will be able to access the provided services, in a non-expected manner (out of the control of the client interface), in such a way that the security of the system (authentication, privilege granting, or even data) might be compromised. Indico has been audited, in the past, against security vulnerabilities, by third-parties, and has been found to be completely secure in what concerns to possible data leaking, code injection, or even user identity theft. It would be harmful for the system if the architectural modifications were to introduce any kind of vulnerability which could be exploited by malicious users.

Being so, the Asynchronous Request Handlers, as it happens with the “synchronous” Request Handlers, were built with security concerns in mind. The model implemented by Indico uses class inheritance to create different levels of access on which the request handlers can be built. These “access levels” were modified for the context of asynchronous handlers, and the new “services” created on the top of them.

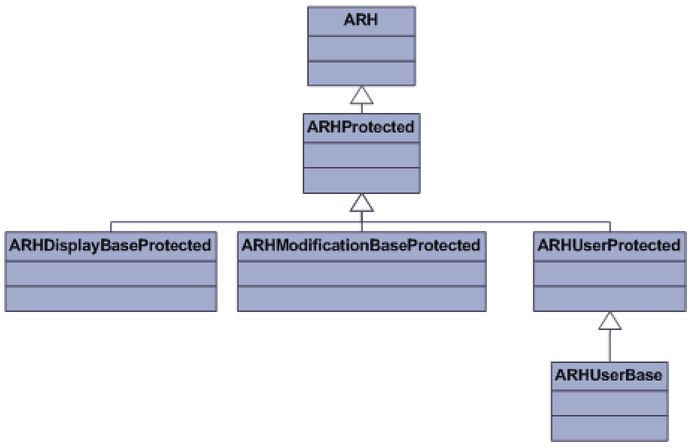


Figure 31 - Class Diagram for the base classes of the Asynchronous Request Handlers

Figure 31 shows the implemented class hierarchy:

- *ARH* is the base for every Asynchronous Request Handler, being responsible for calling the parameter processing functions, and initiating the response cycle;
- *ARHProtected* implements the most basic protection scheme on the top of ARH: authentication. If the user is not authenticated, an *ARHProtected*-derived class will throw an exception before executing anything;
- *ARHDisplayBaseProtected* checks the “target” of the handler (an event, another user, a category, a contribution, etc…) and verifies that the current user has permission to see it. If he/she does not, it will throw an exception;
- *ARHModificationBaseProtected* behaves like *ARHDisplayBaseProtected*, but verifies permissions for content modification, instead of display;
- *ARHUserProtected* is the base for user-centric services (the target is the user himself). It only checks if the user is authenticated, since all the information returned by the classes which derive from it should only concern the user himself.

- *ARHUserBase* derives from *ARHUserProtected*, and implements, as well, translation of parameters from JSON to Python, and vice-versa.

After laying the foundation for asynchronous request handling, the same work had to be done in the client side, through the use of ECMAScript.

5.2 The ECMAScript *IndicoUI* Library

The ECMAScript *IndicoUI* Library, already described in 4.2, provides asynchronous connection to the Indico server, using the *XmlHttpRequest* API method. Instead of implementing natively all the procedures needed for this communication, *IndicoUI* relies on the *Prototype* ECMAScript library for handling it. Besides this library, Indico uses, as well:

- Scriptaculous (as stated multiple times through this document);
- DHTML Calendar[50] – A cross-browser DHTML Calendar ECMAScript widget, distributed under GNU LGPL;

Prototype is used as the base for the entire library, since it provides several utility methods, not only for AJAX, but for ECMAScript enhancement as well (some new "standard library" functions).

5.3 On the WOHL Interleaver

The main technological challenge brought by the WOHL interleaving system was the problem of "injecting" the tooltips next to the desired fields. There could be layout problems with the final result (since the behavior of injected components will depend on the behavior of the ones that surround them), which would be easily solved through the utilization of a "placeholder" for the tooltip/information (i.e. creating a well-positioned div, where the tooltip will be injected). However, a technological problem remained: Indico pages are HTML, not XHTML. Or rather, Indico pages are not XML. Since they're not XML, most DOM parsers won't read them. Fortunately, the *libxml2*[51] library, already used in Indico for XSLT style sheet application, includes a parser capable of storing an HTML document as a node tree, and manipulating this tree through addition or deletion of HTML elements. Being so, the WOHL/HTML interleaving engine uses the regular XML DOM parser from the Python Library for WOHL (which is valid XML), and *libxml2* for the HTML content. The need for a tree-like parser is due to the necessity to "inject" HTML inside an element specified through XPath.

The ECMAScript-based tooltips used for rendering the WOHL specification are currently introduced using the domTT library[52]. However, this library presented some problems regarding the appearance/disappearance of the tooltips according to certain mouse events, thus requiring a future adaption of the library, adoption of a different one, or even production of "in-house" code for the same effect.

[50] http://www.dynarch.com/projects/calendar/

[51] http://xmlsoft.org/

[52] http://www.mojavelinux.com/projects/domtooltip/

5.4 Search Engine Interface

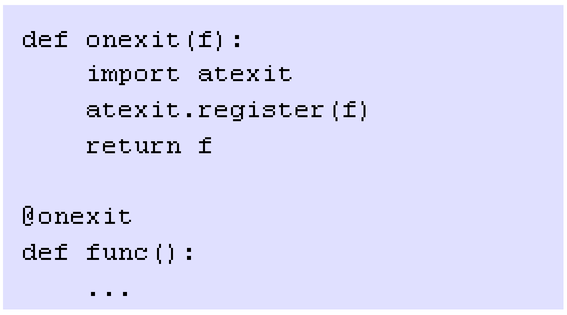
In 4.4, it was said that Search Engine Adapters were built with extensibility in mind: it should be easy to create a new one, for a different search engine, using simple mechanisms. The "adapter" nature of the SEAs made inevitable the analogy of the AC plug adapters (something that foreign people who work in Switzerland are used to): a SEA is like an AC adapter, which interfaces an european plug (Indico) with a swiss socket (Invenio, for instance). If we move to the US (adopting our fictitious search engine, "Joojle"), we'll need a different SEA to provide the interface. Whenever we move to a strange land, where sockets have more "exotic" shapes, we'll need to quickly produce a new adapter, and easily switch from the old adapter to the new one. The "real world" problem comes in two different flavors:

- How to enable a server administrator to easily create a SEA to interface Indico with the "local" search engine?
- How to provide a way for him to select his new invention as the one that should be used?

The first question would be addressed through the use of a "declarative" language, and the second one, through a mechanism of class selection (which would necessarily imply reflection).

Using Python as a declarative language

Since version 2.4, the Python programming language provides support for "Decorators" [1] (Smith, et al., 2007). These syntactic structures provide a way of applying the "Decorator Software Design Pattern"[53] to simple functions or methods, wrapping them and allowing manipulation of their input/output. A decorator is specified using the "@" symbol, and precedes the declaration of the function/method it applies to (Figure 32).



```
def onexit(f):
    import atexit
    atexit.register(f)
    return f

@onexit
def func():
    ...
```

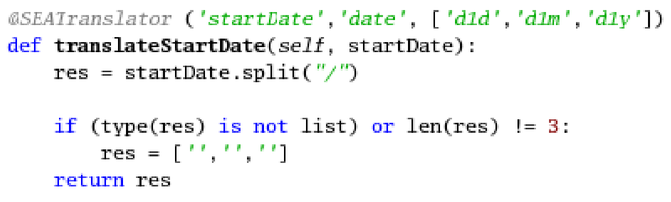
Figure 32 - An example of decorator usage, taken from (Smith, et al., 2007)

Decorators appeared as a way to include declarative information about a function or method, and this was just what was needed to define the so-called "translation functions".

In order to translate the parameters provided by the Indico search interface to the ones accepted by the search engine, a mapping between both had to be established, and specified along with the SEA. One of the arguments of this mapping would be the "translation function", which would transform the values, before passing them to the search engine. Let us take the case of Invenio: the Indico search interface provides a date in the form of a string

[53] GoF 175 - (Gamma, et al., 1995 p. 175)

parameter, called “startDate”, where the fields are separated by slashes (i.e. “27/11/1984”), but Invenio requires three parameters called “d1d”, “d1m” and “d1y”, for the day, month and year, respectively.



```
@SEATranslator ('startDate','date', ['d1d','d1m','d1y'])
def translateStartDate(self, startDate):
    res = startDate.split("/")

    if (type(res) is not list) or len(res) != 3:
        res = ['','','']
    return res
```

Figure 33 - Excerpt from the InvenioSEA definition

Figure 33 shows the actual code for the “Start Date” translation function: the content is just a simple strings manipulation, but the actual “assignment” of this method to this “1 to 3” association from “startDate” to the parameters required by Invenio is specified by the “@SEATranslator” decorator: the original parameter, the field type[54] and the final parameters are passed as strings (or list of strings, for multiple output parameters).

This structure provides an easy way to specify an interface without the need for too much code, and makes the specification of a mapping relation between the two points almost purely declarative (except for the translation function). When the search module is loaded, all the mappings are stored in a “translation table”, and the translation functions are called whenever needed. The whole process of translation is managed by the Search Engine Interface, using an “inversion of control” logic: the “process” method from the “SearchEngineAdapter” class takes care of the translation, and calls the translation methods from the derived class (Figure 34).

Class Selection through Reflection

Another objective of the Search Engine Interface architecture was to make the SEAs completely pluggable without any modification of code inside Indico. This implied that the desired SEA were loaded, trough a non-intrusive selection mechanism. Being so, the SEA name would be specified in Indico’s configuration file, and the SEA would be loaded through reflection mechanisms. This proved to be easy, since Python includes reflection mechanisms built-in since version 1.5 (van Rossum, 1997).

[54] The “field type” is provided for future enhancement of this feature.

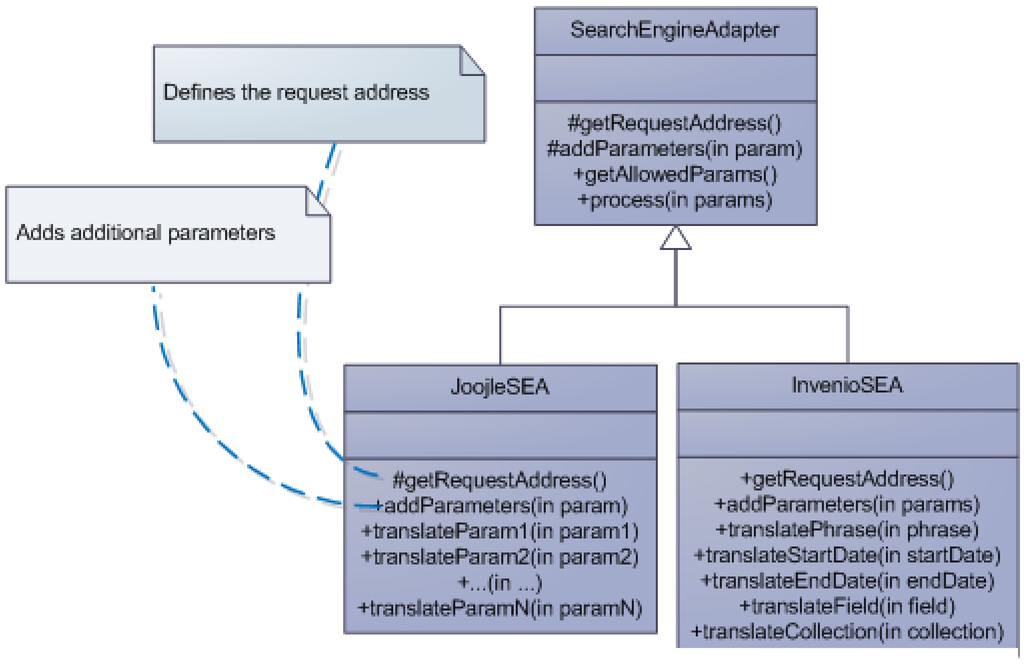

SearchEngineAdapter
#getRequestAddress()
#addParameters(in param)
+getAllowedParams()
+process(in params)
Defines the request address
Adds additional parameters
JoojleSEA
#getRequestAddress()
+addParameters(in param)
+translateParam1(in param1)
+translateParam2(in param2)
+...(in ...)
+translateParamN(in paramN)
InvenioSEA
+getRequestAddress()
+addParameters(in params)
+translatePhrase(in phrase)
+translateStartDate(in startDate)
+translateEndDate(in endDate)
+translateField(in field)
+translateCollection(in collection)

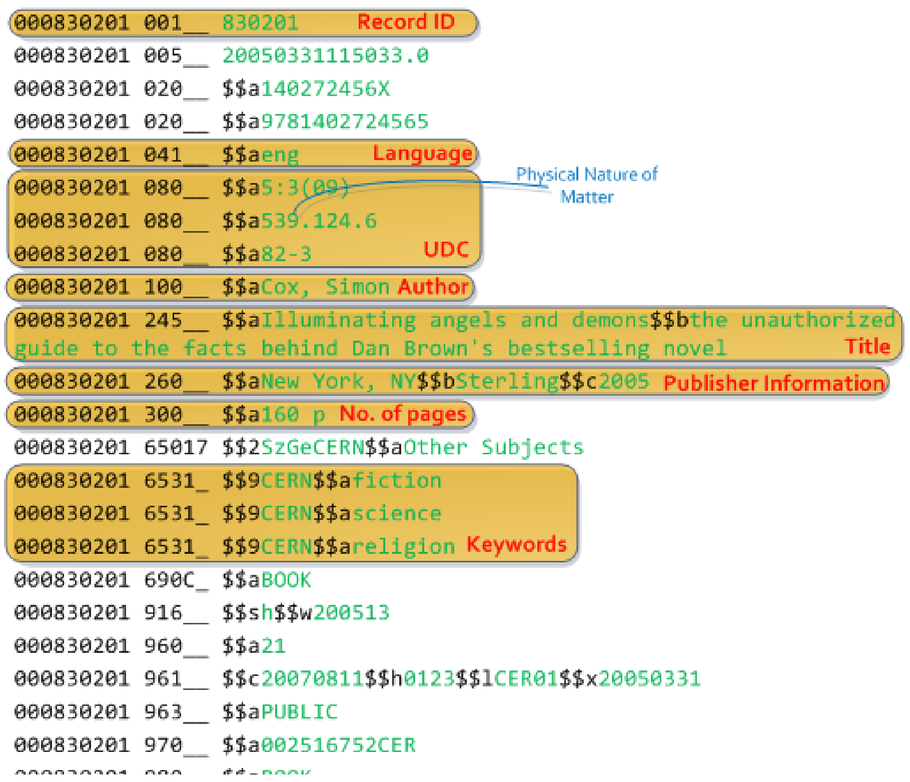
Figure 34 - Implementation of extensibility for the SEAs

The InvenioSEA

CDS Invenio was built using MARC 21 (NDMSO, 2006) as the "underlying bibliographical standard" (CERN Document Server, 2005). It is through this format (although encapsulated in XML – MARCXML) that it communicates with Indico, harvesting the latter through OAI-PMH. MARC provides support for an enormous amount of bibliographical information, and allows several types of classification (i.e. UDC) to be included (see Figure 35):

- Control numbers for libraries, copyright numbers, ISBN, publisher code, language codes, geographic area codes and fingerprinting information;
- Title, abbreviated title, title translation and former title;
- Edition information: place, editor, year of edition;
- Data about physical description (if it's a "physical" book), physical medium and price;
- Different types of notes: summary, target audience, reproduction, original version, etc…;
- Authorship data;
- Associated URLs, for electronic sources;

Invenio's search engine can as well, be queried using MARC, so that records with a specific author or date can be filtered.



```
000830201 001__ 830201
000830201 005__ 20050331115033.0
000830201 020__ $$a140272456X
000830201 020__ $$a9781402724565
000830201 041__ $$aeng
000830201 080__ $$a5:3(09)
000830201 080__ $$a539.124.6
000830201 080__ $$a82-3
000830201 100__ $$aCox, Simon
000830201 245__ $$aIlluminating angels and demons$$bthe unauthorized
guide to the facts behind Dan Brown's bestselling novel
000830201 260__ $$aNew York, NY$$bSterling$$c2005
000830201 300__ $$a160 p
000830201 65017 $$2SzGeCERN$$aOther Subjects
000830201 6531_ $$9CERN$$afiction
000830201 6531_ $$9CERN$$ascience
000830201 6531_ $$9CERN$$areligion
000830201 690C_ $$aBOOK
000830201 916__ $$sh$$w200513
000830201 960__ $$a21
000830201 961__ $$c20070811$$h0123$$lCER01$$x20050331
000830201 963__ $$aPUBLIC
000830201 970__ $$a002516752CER
```

Figure 35 - An example of a MARC record from the CERN library, with some of the relevant fields highlighted

Obviously, Invenio only uses a subset of MARC in order to index Indico's contents, most importantly the following fields:

- 111 – "Main Entry – Meeting Name" – Which stores the event name and location;
- 245 – "Title Statement" – Stores the title of the entry;
- 518 – "Date/Time and Place of an Event Note" – Stores the date of the event;
- 520 – "Summary, Etc." – Stores the summary of the event;
- 650 – "Subject Added Entry – Topical Term" – used for storage of the category of the event (either if it is a meeting/lecture/conference, and the Indico category it belongs to)
- 856 – "Electronic Location and Access" – used for storage of URLs related to the event;
- 970 – The fields starting with 9 are reserved for free use by the institutions. 970 is used by Indico to store the code that identifies an event or contribution;

One of the problems with the loose connection between Indico and Invenio was that Invenio had no notion of "event X that belongs to category Y", or "contribution W which belongs to event Z". Invenio was totally library-oriented, and Indico was seen as yet another repository of information to explore. Being so, Indico category data was, from the beginning, stored using MARC field 650, as a simple colon-separated list of category codes ("0:123:456", for instance), which describes the path from the root of the "Indico category tree" to the "leaf" where the event is contained. The connections between events and the contributions inside them were handled by MARC field 970.

By default, Invenio performs every search on the whole contents of Indico, since it is unable to recognize the different categories. However, we can overload this behavior, by providing a MARC query together with the search keywords. This MARC query would filter out all the results that wouldn't comply with the specified category or event/contribution code. Being so, the following MARC queries are used:

- 650_7:"*<categoryPath>**" where <*categoryPath*> is the full path to the category;
    - i.e. to find all the events inside the "AVC Section Meetings" category:
        - 650_7:"0:1l27:2l83:232:233:508*"
- 970__:"INDICO.*<eventCode>**" where <*eventCode*> is the code of the event;
    - i.e. to find all the contributions belonging to the "CHEP07" conference:
        - 970__:"INDICO.3580*"

Adding these queries to the search keyword, it is possible to tell Invenio where to look for the results. This way, search inside events and categories was successfully implemented for CDS Invenio.

5.5 Personal Features

The implementation of the Personal features ("My Indico" and "User Baskets"), as stated earlier, relies on the already described *IndicoUI* library, and the *Asyndico* ("Asynchronous Indico") architecture, both developed in the context of this usability enhancement project.

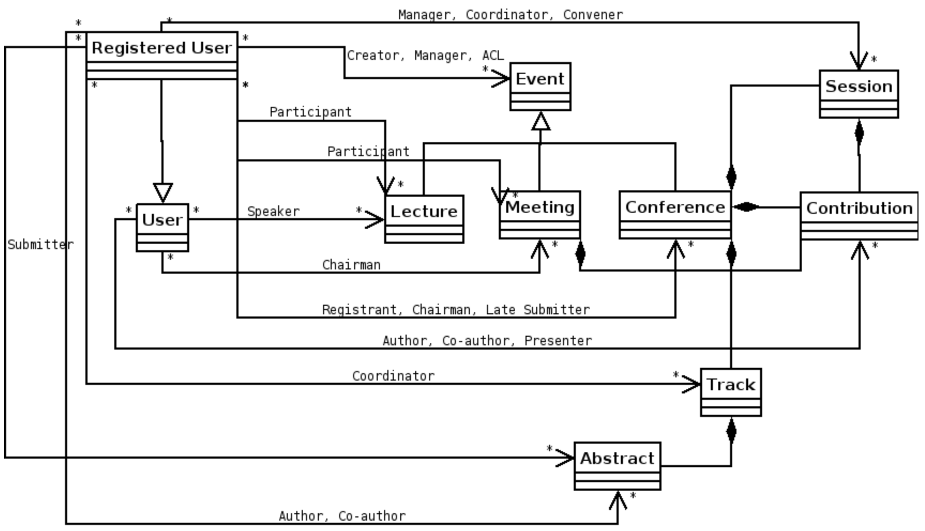


Figure 36 - Associations between users and events

Connecting Users to Events

The requirement for an area which provided the events that a user was in some way connected to, demanded for a structure which could establish a "labeled" association from the latter to the formers. The conceptual model of these relations (Figure 36) involves not only events themselves, but also some structures which compose them, such as contributions, tracks and sessions (for which one can assign users with different roles). Luckily, Indico already had a built in structure (a Python "dictionary", or "map"), which described the different role that the user had in each event he was connected to. However, this structure was not being used at its full potential, since the users had no way to know its contents.

In spite of its low complexity, the model from Figure 36 may be confusing for the average user. For instance, the difference between “manager” and “coordinator” of a session might not be clear for the average Indico user: manager has full access rights to the session, and coordinator has access only to certain rights that were assigned to him; but, in a user point of view, they look very similar. The case is worse for “registrants” and “participants”, which are the same thing, but with different names (and implementations), as “registrants” relate to conferences, and “participants” to lectures and meetings. Other concepts like “author” and “co-author” may be simply merged, for the sake of simplicity.

```
foreach elem in pastList:
        if elem.endDate > now:
                moveToPresent(elem)
foreach elem in futureList:
        if elem.startDate < now:
                moveToPresent(elem)
foreach elem in presentList:
        if elem.endDate < now:
                moveToPast(elem)
        else if elem.startDate > now:
                moveToFuture(elem)
```

Figure 37 - Algorithm for list resynchronization

Being so, the prototype of the “My Events” feature was built with three main user roles in mind (creator, manager, and registrant/participant), which are the most important ones, and the simplest to implement.

The “My Events” dialog aimed to provide a time-ordered list of events, which would have to be generated each time the user accessed the page. In order to avoid a client-side sorting step, and so that the server would not be overloaded with this work, a strategy of storing the information using three different lists (for the past, present and future) was chosen. Elements would be interchanged between them only when they became inconsistent (i.e. an element whose “end date” was yesterday is still in the “Present” list). These lists would exist in parallel with the linking table. The algorithm for “list resynchronization” is described in Figure 37. Initially, sorted lists were used, since these would substantially reduce the execution time of the algorithm (since there would be no need to verify the “past” and “future” lists in their whole length), however, this option was abandoned, since this would require the lists to be reordered, each time that a rescheduling of an event occurred. This would require some changes in some of Indico’s base classes, which were not compatible with the “prototypal” nature of the “My Events” module. Being so, sorted lists will be reconsidered only after the prototype is put in production.

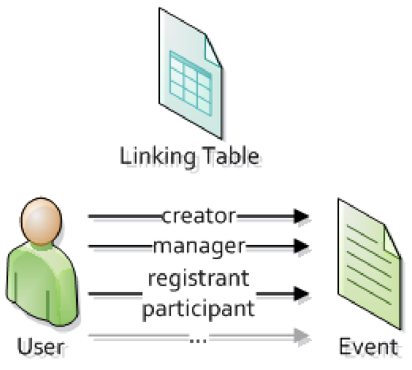


Figure 38 - The "linking table" provides the connection between users and events

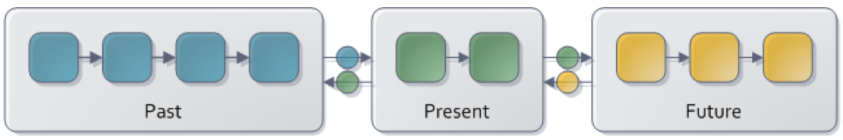


Figure 39 - The Time-ordered event list, which is, in fact, made of three separate lists

The time-ordered list is stored persistently by the ZODB database, as an attribute of the “Avatar” objects. “Avatars” represent users of the application, and the information associated with them.

Adding Support for User Baskets

The implementation of the user baskets, in the server side, was done with the idea of the “multi-type” basket in mind. This means that the class which represents the “user baskets” (or rather a “unified user basket”) is able to conveniently store users, groups of users, events, categories of events and rooms. This is done in a simple and intuitive way: there’s one single method for addition of content to the basket, and five “internal containers” (Python dictionaries) which store, each one, a different type of element. Taking advantage of the characteristics of the Python programming language, the type of content to be added is verified in run-time, so that it can be stored inside the right container. The existence of five containers makes it simple and faster to retrieve the elements of a specific nature, in the future. The single “add” method introduces a degree of transparency and flexibility, making it possible to add objects to the basket, without being worried about their type, and the way they should be handled.

The “Personal Basket” has been added as a persistent class, instantiated by “Avatar” objects.

5.6 New Layout

In order to make the new web page layout compliant with the requirements, a certain amount of CSS work was required. The problem resided in avoiding a table-based layout. Instead of tables, HTML DIVs (W3C, 1999a) were used in order to position the elements on the required places. However, as it is known for most people who ever tried to work out some relatively complex layout using CSS, each browser implements the standard in a different way, and different results are frequently obtained using the same code. The process required some study about the different rendering engines used by browsers, but finally a satisfactory solution was reached, fully compliant with all the tested browsers.

5.7 Evaluation

The implementation of the aforementioned prototypes was definitely successful, if the degree of accomplishment of the functional requirements is taken as the criterion:

- The bases for the Asynchronous service-based architecture has been established, and has passed successfully the tests it has been subject to;
- The *IndicoUI* client-side library has proven to be a flexible tool for creation of different types of tooltips;
- “*Personal Features*” have been successfully built on the top of the *IndicoUI* library and the *Asyndico* architecture. They still require a certain degree of “validation” from the users before conclusions are drawn;
- “Search feature” was the first (and only, at this time) to hit the production version and a public release, and has not experimented any difficulties so far, aside from

some missing results[55], that are thought to result from a problem in the transmission of data from Indico to Invenio, during the OAI-PMH harvesting process. This issue is not related with the Search Engine Interface, and is currently queued for investigation;

- The new layout is a work in progress. It was not in the initial list of requirements, and the fact this project was able to invest in such an ambitious effort is, for itself, prove of its success.

The accomplishment, in the long term, of the non-functional requirements, can only be precisely measured through the user feedback that the project will receive after the features are put in production.

[55] See users' complaints, in (Ferreira, 2007a)

6 Conclusions and Future Work

Some days before the writing of this report, Robert Cailliau, the recently retired engineer from CERN who invented, together with Tim Berners-Lee, the World Wide Web, has said in an interview about the creation of the Web, that “freely contributing to web content was one of the original goals: the web should be as easy to write as it is to read. Another function we lost due to too fast development.” (Wikinews, 2007). In fact, prior to the Web 2.0 revolution, it was very difficult for the user to create content and place it on the Web. This wasn’t only a matter of good will from the hosting providers and IT companies. It was also a question of technological maturity: it would be insane, ten years ago, to base a web application in technologies such as JavaScript – there was no mature standard and not enough browser compliance. Nevertheless, nowadays, the “web giants” are firmly betting on this field, and the truth is that users like it. The “social phenomenon” that the web started to experiment can be certainly related to the technological development that accompanied it (the statistical data about the proliferation of internet access through the world should not be forgotten, though).

Many people have the opinion that DHTML/AJAX-enabled pages, in most of the cases, damage the usability of a website (Nielsen, 2005). Some others consider that a correct use of the technologies and standards ensures as well a good degree of accessibility (Gibson, 2006). The W3C, itself, while still recommending all the pages to be accessible with ECMAScript turned off[56], is working on the subject of “Dynamic Accessible Web Content” (W3C, 2006), in order to empower the inevitable “dynamic revolution” with tools for accessibility. There’s still a great debate about the equilibrium that the technologies which nowadays support the web can provide to the usability/accessibility binomial: does “dynamic” mean “less accessible”? For now, maybe that’s true for the most severe cases of inaccessibility: there are simply no tools for establishing a good degree of communication between a screen reader and a DHTML page. Would it be possible for a blind person to use *Google Docs and Spreadsheets*[57]? Not for now, since we’re speaking of a rich application that uses the current technology at its full extent, a house built with matches, or rather a desktop-like application made with content-oriented technology (HTML) and some additional scripting mechanisms (ECMAScript). Some may argue that these latter mechanisms should not even be used, but that would imply the return to the time when the web was static at the client side, and usability was limited by the “synchronous” nature of the technology. A purely REST[58] Web would be very different. One cannot say that things would be better or worse, since we never had a “Web” in the sense that Berners-Lee and Cailliau wanted it – Cailliau, once again through his interview, regrets this: “we advanced far too fast with far too many developers who ran away in far too many different directions without much thinking. I don't want to be Cartesian here, but it would have been better if we had more time to build on our ideas before letting the beast loose”. The emergent technologies which powered the Web 2.0 may thus be seen as an attempt to “fix” the void left by an unfinished implementation of somewhat that was meant to be a collaborative platform, and not a one-sided “broadcasting station”.

[56] See the WCAG (W3C, 1999b)

[57] http://docs.google.com

[58] See 3.1 – “The “old” Web – REST Architecture”

The “unorthodox” approach that this project followed, adopting the latest technologies and usability enhancement techniques as the base for the improvements[59], instead of limiting itself to a simple “reorganization” of features, or creation of new ones using the “traditional” paradigm, might look risky and problematic. However, as the conclusions from section 5 report, there were no problems with the implementation of the new features, either of technological or structural nature. It is obvious that in order to get a real idea about the degree of usability of these improvements, the development team will have to conduct new tests after the integration of these features, but the feedback obtained, so far, from several “hallway samples”, has been completely positive. On the other hand, the theoretical guidelines that rule this field have been followed, and thus one may expect nothing but an increase in the dimension of usability.

The most important fact, among technologies, metrics and concepts, is that all the work performed in terms of system redesign and implementation was supported by multiple studies[60], some of them done by people who were not directly involved with Indico and its development[61]. These studies had both a “static” and “dynamic” nature, in the sense that the latter ones addressed the user directly, and the former had a more “forensic” nature, looking for answers through the knowledge that had been accumulated during decades of experimentation.

This “Usability Enhancement Project” has been, no doubt, a valuable contribution for Indico, and included the first complete usability study ever performed at CERN. Its results cannot be measured in a time span of six months, but will certainly be acknowledged by the users of future versions of Indico.

6.1 New Perspectives

The future of Indico is planned but not precisely defined. The already mentioned software development process[62] is based on flexibility and a close relationship with the user. Indico v1.0 is expected to be released somewhere in 2008, after the code becomes stable enough, and a set of "immutable" features is defined. This "Usability Enhancement" Project is supposed to prepare Indico for deployment in a XXI century Web, where users seek easiness of use, intuitiveness, and understandable content. Along with internationalization, this effort will take Indico to the level of most open source and commercial web-based CMS.

At the end of the layout refactoring process, there will be still much work to do, but not specifically in the area of usability. The main issues are expected to be solved with this last, time-consuming step. Obviously, some plumbing will be needed, and the user will play an important role, helping on smoothing some corners.

After stepping into the paradigm of the "dynamic web", it is uncertain what will be Indico's next move. Since v1.0 is intended to be stable and solid, no new significant features are expected to be introduced in the near term. However, this fact doesn't prohibit the formulation of some ideas:

[59] See 3 – “Technological Review”

[60] See 2 – “The Problem”

[61] See 2.1 – “The Usability Problem”

[62] See 1.2 - “Software Development Process”

- Stepping in the direction of the “semantic-powered Web”, through the use of Microformats - some specific formats, like *hCalendar*[63] and *hCard*[64] are easy to integrate, and fit very well inside the scope of the application - this will enable users to quickly store information, and transfer it to other programs and systems. In spite of the immaturity of these technologies, nothing inhibits Indico from experimenting with these new approaches to content organization, since both Firefox 3 and Internet Explorer 8 are expected to include native support for Microformats;
- Evolving towards the possibility of creating networks of Indico instances, with common subjects (i.e. High Energy Physics), capable of sharing information between them, not only in what relates to events, but also regarding user authentication mechanisms and content.

The "Usability Enhancement Project" should not stop right here, since the rate of mutation of the project requirements, in spite of a probable decrease, is likely to remain high, in a project of this nature. This project should, in this context, provide continuing aid to the development process:

- Assuring that the usability standards remain satisfactory;
- Providing an even better connection between the development team and the users (through the realization of tests at the time of the introduction of new features, for instance);
- Working for the accomplishment of all the standards involved in the development of Indico: HTML, CSS, iCal, RSS, etc...;
- Keeping alive a culture of agility and direct contact with the users (inside and outside CERN), as well as providing a critical approach to questions connected with its scope, through the involvement of "independent" third parties in the process;

This could be applied, as well, to the other web applications that exist at CERN, and play an important role in the daily life of its employees: CDS Invenio, EDH and EDMS for instance. The possibility of establishing a “task force” for usability at the CERN IT Department, in order to provide assistance to the different projects, should be considered, as CERN has an example role to play as the place where the web was born, and, consequently, a worldwide repository of web content, permanently available through Indico, CDS Invenio, and many other systems, open to the world.

6.2 Final Remarks

I wouldn’t finish this report without writing some words about this internship, and the way it happened.

I definitely do not regret the moment I had the idea of applying for a Technical Student position at CERN. It could have been a bad move, since I was uncertain about the kind of work I would perform. I tried to know the best way possible what I would be doing, but the truth is that we never know how the environment will be (inside and outside CERN), how it will be constantly communicating in a foreign language at work, what will be required from us, and, finally, if the work we’ll be performing will provide us pleasure or infinite hours of boredom.

[63] http://microformats.org/wiki/hcalendar

[64] http://microformats.org/wiki/hcard

Working at CERN is a unique experience, independently of what one is doing: being able to work in one of the greatest scientific institutions in the world, where the most ambitious projects in science are being developed is, for itself, already great. But CERN goes beyond that, providing a multicultural environment, where different people from different origins work (most of them temporarily) towards the answer to the fundamental questions about the universe. I have to thank this institution for the training that I could benefit from, not only through my work, but as well through the lectures and other activities that it promotes.

I was lucky, as well, with the project I was assigned to. Being able to work on open source software in "full-time" is an opportunity that not everyone gets. Being able to do it in an open-minded, dynamic, and agile environment is even better. Indico is currently very useful to CERN and other institutions connected to High Energy Physics, and I believe that it will soon grow outside the borders of HEP, and enjoy the notoriety of a Moodle[65], or a Plone[66]. I guess the secret is to keep the connection with the users, and encourage them to contribute, in the way they can (from feature suggestion to "hands-on" development).

Fortunately, my internship project may be considered a success in every way: the integration was quick and smooth, the project was stimulating and multifaceted, and the objectives were accomplished. In addition to that, I had the opportunity to enrich my professional and personal experience in a way that I would previously think as impossible for someone who's now starting his professional career.

[65] http://www.moodle.org

[66] http://www.plone.org

Glossary

Apache HTTPD – Famous web server software, developed by the Apache Foundation;

AJAX – Asynchronous JavaScript and XML;

CDS Invenio – Invenio is a web application, maintained by the CDS (CERN Document Server), which is an integrated web library system, for the management of large size document repositories. It is used at CERN, and distributed under GPL;

CERN – European Organization for Nuclear Research (fr. *Organisation européenne pour la recherche nucléaire*);

CRBS – CERN Room Booking System – The room booking system at use at CERN, either the old version (now discontinued), or the new Indico::CRBS, the implementation as a module inside Indico;

CSS – Cascading Style Sheets – A language used to specify the presentation of a markup-based document (most commonly (X)HTML);

DOM – Document Object Model – A W3C standard that defines a platform and language independent object model for the representation of XML and HTML documents. It is used by the ECMAScript engines embedded inside web browsers, for document access and manipulation; A **DOM Parser** is a parser that loads an XML file to a tree-like structure, making it accessible through the DOM API;

ECMAScript – Scripting language, ECMA-262 standard, usually referred to as JavaScript, in the context of web browsers. It is a prototype-based language;

EDH – Electronic Document Handling – The web application that gives support to the CERN business processes, through an electronic workflow management system;

EDMS – Engineering Data Management System – A DMS used at CERN for engineering documentation and technical data storage;

GPL – GNU Public License – Free Software License, maintained by the Free Software Foundation;

Hallway Test – Usability test technique, in which a random sample of usually five people (“dragged from the hallway”), with no previous training as “testers”, is used to test the software;

HTTP – Hypertext Transfer Protocol – TCP/IP-based protocol, the base technology for the Web. Allows retrieval of page content and provides form submission mechanisms (POST/GET);

(X)HTML – (eXtensible) Hypertext Markup Language – A markup language for the creation of web pages. XHTML is the newer, XML based, evolution of HTML. HTML’s latest version is 4.01. There are three variants of HTML 4.01 and XHTML: “Strict”, which is the rigorous implementation of the standard, “Transitional” and “Frameset”. The latter ones are provided for legacy compatibility, keeping some characteristics from previous versions;

iCal – iCalendar (not to be confused with the software with the same name) – A format for calendar data exchange;

In-place Editing – A concept, brought from desktop applications to the web, which consists in providing data input whenever output is present. The input is usually toggled by clicking on the desired field;

IPA – International Phonetic Alphabet;

ISBN – International Standard Book Number – A 13-digit long code that uniquely identifies each and every commercially-available book;

ISO – International Organization for Standardization;

Layout Engine – A software engine that renders web content. It's normally used by web browsers;

JSON – JavaScript Object Notation – A format, which is a subset of the ECMAScript language, for serialization of ECMAScript object and data structures;

OAI-PMH – Open Archives Initiative's Protocol for Metadata harvesting - An XML-based protocol, developed by the Open Archives Initiative, which enables transmission of metadata (about records) from a repository to a harvester;

MARC – Machine-Readable Cataloging – A set of standards, developed by the Library of Congress (USA), for representation and communication of bibliographical information; **MARCXML** is and XML dialect for representation of MARC21 information;

Microformats – Formats that reuse (X)HTML structures as semantic containers. The technique consists in adding semantic meaning to existing data, through special tag attributes. Some examples of Microformats are *hCalendar*, for event data, and *hCard*, for personal contact information;

Mod_python – An Apache HTTP Server module which enables the use of Python as a web programming language;

PHP – PHP: Hypertext Processing – A scripting language for web application programming;

Python – A multi-paradigm programming language, created by Guido van Rossum in 1991. It is dynamically typed, and includes automatic garbage collection. It is used for various purposes: as a desktop/web application creation language and "shell"/embedded scripting language;

POST / GET – POST and GET are two different methods for HTTP request. While, in the first one, information is carried in the request body, the second carries it encoded in the request URL;

RIA – Rich Internet Application – A web application with features and capabilities similar to those of a desktop application;

RSS - Really Simple Syndication – A family of XML-based formats for representation of frequently-updated content (blog posts, podcasts, events, etc…);

SAX - Simple API for XML – A parser API for XML, based on a serial parsing strategy. It is used as an alternative to DOM parsers;

Screen reader – Software for screen interpretation, normally used for text-to-speech or Braille output;

SOAP – A standard protocol for exchange of XML messages over the web. SOAP is widely used by Service-Oriented Applications.

SPIAR – Single-page Interface Architecture – An architecture for AJAX-based applications;

UCD – User-Centered Design;

VRVS - Virtual Room Videoconferencing System – A system for videoconferencing, in use at CERN;

Web 2.0 – Concept defined by (O'Reilly, 2005), of a “new web” where user experience is enhanced through collaboration. The flux of information is many-to-many instead of the traditional one-to-many;

Web crawler – A program which searches the web in a systematic way, usually with the objective of gathering and indexing information;

Widget – An interface component the user interacts with;

W3C – World Wide Web Consortium;

WCAG – Web Content Accessibility Guidelines – A set of Accessibility Guidelines defined by the W3C;

WYSIWYG – “What You See Is What You Get” – the paradigm that states that the appearance of a document during the editing process is the same as that of the final product;

XML – eXtensible Markup Language – a general-purpose markup language, highly extensible, and recommended by the W3C;

XPath – XML Path Language – A language for addressing parts of an XML document;

XSD – XML Schema – An XML-based language for definition of the contents of a specific XML Schema;

XSLT - Extensible Stylesheet Language Transformations – An XML dialect for specification of transformations between XML and other formats;

XUL – XML User Interface Language – An XML dialect for user interface specification;

References and Bibliography

Allaire, J. (2002). *Macromedia Flash MX—A next-generation rich client.* Macromedia Inc.

Baron, T. (2002). The InDiCo Project - A Tool for Managing Conferences through the Web. *Presentation at the JACoW Workshop (2002).*

Baron, T., Gonzalez, J., Le Meur, J.-Y., Sanchez, H., & Turney, V. (2004). INDICO - the software behind CHEP 2004. *Proceedings of CHEP'04.* Geneva: CERN.

Berners-Lee, T., Shadbolt, N., & Hall, W. (2006). The Semantic Web Revisited. *IEEE Intelligent Systems* , 91-101.

Boehm, B., & Turner, R. (2003). Observations on Balancing Discipline and Agility. *Proceedings of the Agile Development Conference (ADC'03).* IEEE Computer Society.

CERN Document Server. (2005, August 08). *CDS Invenio Overview*. Retrieved August 14, 2007, from CDS Software Consortium: http://cdsware.cern.ch/invenio/

Couper, M. P. (2000). Web surveys: A review of issues and approaches. *Public Opinion Quarterly* , pp. 464-494.

Ferreira, P. (2007a). *Indico Usability Survey - Results and remarks.* CERN IT-UDS-AVC.

Ferreira, P. (2007b). *The Usability of Indico - An evaluation of usability, accessibility and user-centeredness.* CERN IT-UDS-AVC.

Ferreira, P. (2007c). *WOHL Specification.* CERN IT-UDS-AVC.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures (PhD Thesis).* Irvine: Univesity Of California.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns - Elements of Reusable Object-Oriented Software.* Addison-Wesley.

Garrett, J. J. (2005). *Ajax: A New Approach to Web Applications.* Retrieved 6 1, 2007, from adaptive path: http://www.adaptivepath.com/publications/essays/archives/000385.php

Gibson, B. (2006). JavaScript & Ajax Accessibility. *WWW2006.*

HHS. (2006). *Research-Based Web Design and Usability Guidelines.* U.S. Department of Health & Human Services.

Iivari, J., & Iivari, N. (2006). Varieties of User-Centeredness. *Proceedings of the 36th Hawaii International Conference on System Sciences.* IEEE Computer Society.

InDiCo Project. (2002). IST-2001-34306 - Contract for Shared-Cost RTD.

ISO. (1998). Part 11 : Guidance on usability. In ISO, *Ergonomic requirements for office work with visual display terminals (VDTs) (ISO-9241).* ISO.

Kepser, S. (2004). A Simple Proof for the Turing-Completeness of XSLT and XQuery. *Extreme Markup Languages 2004: Proceedings.*

Library of Congress - Network Development and MARC Standards Office (NDMSO). (2006). MARC 21 Concise Format for Bibliographical Data (2006 Concise Edition).

Mesbah, A., & van Deursen, A. (2007). An Architectural Style for Ajax. *Working IEEE/IFIP Conference on Software Architecture (WICSA'07).* IEEE Computer Society.

Molich, R., Ede, M. R., Kaasgaard, K., & Karyukin, B. (2004). Comparative usability evaluation. *Behaviour & Information Technology* , Jan-Feb , pp. 65-74.

Mondardini, R. (2007a). *Indico Web Interface - Usability Analysis.* CERN IT-COMM-TEAM:

http://it-comm-team.web.cern.ch/it-comm-team/repository/indico/IndicoWebUsability.pdf

Mondardini, R. (2007b). *Usability Report Quick Findings - Indico.* CERN IT-COMM-TEAM:

http://it-comm-team.web.cern.ch/it-comm-team/repository/indico/UsabilityReport-QuickFindings.pdf

Nielsen, J. (1993). *Usability Engineering.* Morgan Kaufmann.

Nielsen, J. (2005). *Why Ajax Sucks (Most of the Time).* Retrieved August 20, 2007, from Jakob Nielsen's Alertbox: http://www.usabilityviews.com/ajaxsucks.html

Nielsen, J., & Tahir, M. (2001). *Homepage Usability: 50 Websites Deconstructed.* New Riders Press.

O'Reilly, T. (2005). *What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software.* Retrieved August 15, 2007, from http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

Sadoski, D., & Comella-Dorda, S. (2000). *Technology Descriptions - Three Tier Software Architectures.* Retrieved August 14, 2007, from Software Engineering Institute - Carnegie Mellon: http://www.sei.cmu.edu/str/descriptions/threetier.html

Smith, K. D., Jewett, J. J., Montanaro, S., & Baxter, M. (2007). *PEP 318 - Decorators for Functions and Methods.* Retrieved August 14, 2007, from Python Enhancement Proposals: http://www.python.org/dev/peps/pep-0318/

van Rossum, G. (1997). *Metaclasses in Python 1.5.* Corporation for National Research Initiatives.

W3C. (2006). *Dynamic Accessible Web Content Roadmap.* Retrieved from W3C: http://www.w3.org/WAI/PF/roadmap/DHTMLRoadmap040506.html#N1008C

W3C. (1999). *HTML 4.01 Specification.* Retrieved from W3C.

W3C. (2005). *Introduction to Web Accessibility*. Retrieved August 14, 2007, from Web Accessibility Initiative (WAI): http://www.w3.org/WAI/intro/accessibility.php

W3C. (2007). *W3C Semantic Web FAQ.* Retrieved August 17, 2007, from W3C: http://www.w3.org/2001/sw/SW-FAQ

W3C. (1999). *Web Content Accessibility Guidelines 1.0.* Retrieved from W3C.

Wikinews. (2007). Wikinews interviews World Wide Web co-inventor Robert Cailliau.

Wikipedia contributors. (2007a). *"Accessibility"*. Retrieved August 14, 2007, from Wikipedia, The Free Encyclopedia: http://en.wikipedia.org/w/index.php?title=Accessibility&oldid=150664466

Wikipedia contributors. (2007b). *"Ajax (programming)"*. Retrieved August 15, 2007, from Wikipedia, The Free Encyclopedia: http://en.wikipedia.org/wiki/Ajax_(programming)

APPENDIX A: *"The Usability of Indico - An evaluation of its usability, accessibility and user-centeredness"*

The Usability of Indico

An evaluation of its usability, accessibility and user-centeredness

Pedro Ferreira, CERN IT-UDS-AVC, April-August 2007

Introduction

Purpose

The purpose of this document is to provide an overview about Indico's usability and ergonomics. This includes questions like navigation, user-friendliness, personalization, and everything which relates to the assertion of a good relationship between user interfaces and the users themselves. This is not intended to be a "psychological" human-machine interaction report, or an in-depth analysis of the interface and features of Indico. The main objective is to identify the main fragilities which Indico suffers from, in the areas above mentioned, and to provide a set of "requirements" in the form of "user stories", which, once implemented, will improve significantly Indico's user experience. I've gathered information from several sources: user feedback provided by mail through the last years, requirements specified at Indico's Savannah, and a user survey conducted at CERN's Indico website. I've also made an analysis of the compliance of Indico against some standards and regulations, namely the *W3C Web Content Accessibility Guidelines* (W3C:1999), which provide some important hints on content organization and accessibility.

Context

The Indico Project is a web application which was raised at *CERN* as a substitute for the old *CDS* Agenda. It is a *CMS* aimed at event management and consulting. Indico's objective is to provide the users with a usable, flexible and fully-featured system, which allows them to organize their lectures, meetings and conferences, in the role of either convener or participant. This implies the inevitable existence of long forms, intensive submission of data, and the need to present the contents in the better way possible. Being so, usability is one of the essential requirements in an application of this kind. From the exploration of the correspondence of the Indico Support Team which I've done, I'd say that the transition from *CDS* Agenda to Indico wasn't totally "peaceful". Many users considered that Indico lacked the "agility" of the old Agenda, in favor of a new interface, with new features, but consequently more complex. Through the years, the Indico Support Team has worked intensively in order to fix the most important usability issues. However, usability is only one of the many things that need time spent on, and equilibrium had, eventually, to be reached. Being so, the main forces in favor of Indico's usability improvement have always been the criteria of the developers, and the user feedback, provided through support requests and mailing lists.

Sources of Information

This document was based on three repositories of information:

- The Indico mailing lists
    - The *indico-support* mailing list archives;
    - A compilation of mails, kindly provided by Thomas Baron, containing user feedback received over the years;
- The *Savannah tasks* - A set of tasks extracted from the Indico Savannah page (a Savane[67] *CMS*), which propose some improvements related to the fields of study of this document;
- The *Indico Usability Survey*

But the main source of information was Indico itself. As a software product, Indico is testable, and, through working with it, we can extract something which resembles the experience of a regular user in the system.

Analysis

Accessibility

It is an obvious requirement of a tool like Indico that it may be used by virtually every user inside and outside *CERN*. It is also obvious that some features are not possible to be used by every kind of browser: for instance, the badge editor would never work on a text-based browser, or even a browser without JavaScript support. However, it is important that other features such as event addition, modification and, most importantly, consulting, work correctly. Here, we find ourselves in the borderline which separates usability from accessibility. In the recent years, the advent of new client-side web technologies, such as client-side scripting and dynamic web pages, have boosted the development of the so-called "Web 2.0″, and a whole new generation of more dynamic and desktop-like applications.

These new applications tend, however, to sacrifice accessibility in the name of user-friendliness and usability. New guidelines for "Accessible Rich Internet Applications" (W3C:2006) are being developed right now by the *W3C*, but these will require support from technologies and formats which are not yet either implemented by the mainstream browsers, or solidly established. Being so, most *RIA*s rely on duplication of content as a way of ensuring accessibility. That's a price that some "dot com" companies may be willing to pay, in order to boost usability to its maximum without excluding any user. However, smaller projects like Indico are not capable of carrying the burden of having to maintain two versions of the same web application. Thus, equilibrium between accessibility and usability must be achieved.

Client Applications

From the Indico web server statistics, we may extract the following:

on March 2007

Browser	Percentage
Mozilla Firefox	**40.5%**
Mozilla Firefox 2	18.6%

[67] https://gna.org/projects/savane

Mozilla Firefox 1.5	17.5%
Mozilla Firefox 1	2.7%
Mozilla Firefox (Other Versions)	1.7%
Internet Explorer	**33.5%**
Internet Explorer 7	20.4%
Internet Explorer 6	12.7%
Internet Explorer (Other Versions)	0.4%
Mozilla	**11.2%**
Safari	**8.1%**
Other	**6.7%**

It is easy to notice that the most Indico users use modern web browsers to access Indico. If we add up the Firefox 1.5 & 2, and the *IE* 6 & 7 users, we get almost 93% of Indico's user base. This means that more than 90% of Indico users use browsers which support JavaScript, *DHTML* and *XmlHttpRequest()*. It is not possible to know if the rest will have any problem with dynamic pages, but since they represent approximately 7% of the user base, it would not be wise to consider them "residual". However, on the other hand, ignoring that 93% of the users are able to use Rich Internet Applications would consist in a deadweight loss.

The current status of Indico (April 2007)

The *W3C* Web Content Accessibility Guidelines (W3C:1999) provide a set of rules which intend to guide webmasters through the process of construction of pages which are accessible to everyone, regardless of his/her disabilities, hardware or software limitations. It is the most widely accepted standard in the area of accessibility.

I've evaluated the status of Indico in terms of the *WCAG*, and obtained the following results:

1. Providing equivalent alternatives to auditory and visual content - **Fails** (0/1)
   - There are at least some images without *alt* attribute. This should be corrected as soon as possible;
2. Not relying on color alone - **Passes** (2/2)
   - Indico is perfectly usable by colorblind people, and uses text styles (not only color) to differentiate things;
3. Using markup and style sheets and doing so properly - **Fails** (2/6)
   - Some deprecated *HTML* properties are being used, such as *width*. Absolute units are used all the time, even when it isn't needed. There lacks a structural organization: for instance, the menu in the main page could be presented as a list, instead of a table. It would be much easier for disabled people to understand them.
4. Clarifying Natural Language usage - **Fails** (0/3)
   - There is no reference to the language of the document, in the whole code;

- Acronyms and abbreviations are never used. They are useful, and useful for screen readers;

5. Creating tables that transform gracefully - **Fails**
    - Tables are extensively used for layout purposes;
    - There are no table summaries;
    - There's no separation between logical levels inside a table;
    - This confuses a lot the screen readers, and makes the document much more complex (lots of *td* and *tr*). Layout should be handled by *CSS*.
6. Ensuring that pages featuring new technologies transform gracefully - **Passes** (4/5)
    - Indico is usable without *CSS*;
    - However <*noscript*> tags lack;
    - A user should know when he's not able to use the feature because of technological restrictions;
7. Ensuring user control of time-sensitive content changes - **N/A**
    - Indico does not contain blinking text, moving, scrolling, or other kind of self-moving material;
8. Ensuring direct accessibility of embedded user interfaces - **Fails** (0/1)
    - There areas in the site (badge editor, for instance), where things can't be done without a mouse;
    - However, since nowadays nothing can be done without a mouse, I'd set the priority of this correction as low;
9. Designing for device-independence - **Passes** (2/4)
    - Links and form elements can be iterated using *tab*;
    - There doesn't seem to be any device-dependent code in Indico's client scripts;
    - There are no keyboard shortcuts for actions. However, that doesn't seem to be prioritary;
10. Using interim solutions - **Passes** (2/2)
    - Indico doesn't spawn new windows, and does not redirect the user anywhere else;
    - Labels are correctly positioned, near the form controls;
11. Using W3C technologies and guidelines - **Passes** (2/3)
    - Indico uses W3C-defined formats, presenting some issues, though;
    - As was said before, deprecated attributes are being used;
    - There's a degree of content negotiation, but it could be improved to the language level, for instance;
12. Providing context and orientation information - **Fails** (0/2)
    - There are input fields with no labels associated to them;
    - Elements are not logically grouped;
13. Providing clear navigation mechanisms - **Passes** (7/9)
    - Link targets are clear;
    - There's a site map, the layout is intuitive, and there are navigation bars;
    - However, once again the organization of the documents is not the best;
    - No metadata is available for the pages (RDF, for instance);
    - Text sections are not clearly identified;
14. Ensuring that documents are clear and simple - **Passes** (3/3)
    - The site is consistent, and the language is clear;

In 14 guidelines, Indico passes 7 and fails 6, being that guideline 7 does not apply to this case. From the 7 "passed" guidelines, only 3 are fully accomplished.

We can extract the most important issues with Indico's accessibility from here:

- Guideline 1 - *WCAG* assigns priority 1 to its checkpoints;
- Guideline 4 - *WCAG* assigns priority 1 to the missing checkpoint;
- Guideline 5 - *WCAG* assigns priority 1 to one of the missing checkpoints;

Other guidelines are less prioritary:

- Guideline 3 - *WCAG* Priority 2;
- Guideline 12 - *WCAG* Priority 2;
- Guideline 8 - *WCAG* Priority 2 - since the features in question are not very important;

Usability

(HHS:2006) provides a large set of guidelines for user-centered and usable web applications. Each guideline is rated in terms of "Relative Importance" and "Research Support", being this last one the "amount" of evidence, provided by research, that supports the guideline. In spite of providing an exhaustive list of research-based and common sense directives, the authors insist that:

"One of the basic principles of user-centered design is the early and continual focus on users. For this reason, user involvement has become a widely accepted principle in the development of usable systems. Involving users has the most value when trying to improve the completeness and accuracy of user requirements. It is also useful in helping to avoid unused or little-used system features." (Guideline 1:4)

This guideline is given the maximum importance, in spite of the hesitations of some researchers to agree with it. The tests conducted by Baroudi, Olson and Ives (Baroudi:1986) permit *"the tentative conclusion that user involvement in system development leads to increased user information satisfaction and increased system usage."* The authors admit, however, the frailty of their own conclusions, since the study was targeted at a specific type of user. Besides that, they strongly advice the involvement of users in system design activities. This leads us to the need of extracting user experience information, and user feedback, which will be addressed later.

Optimizing the User Experience

- 2:3 - Standardize Task Sequences
    - *Allow users to perform tasks in the same sequence and manner across similar conditions.*
    - This guideline is directly related to intuitiveness, and we may say that Indico implements it in an elegant way. The presence of a sidebar for easy navigation, the division of event configuration in different tabs, a set of icons which represent certain tasks through the whole system… all these elements help to create a "standard" of widgets and symbols, which is consistent through the entire application.
- 2:4 - Reduce the User's workload
    - *Allocate functions to take advantage of the inherent respective strengths of computers and users.*
    - Indico partially accomplishes this guideline. For instance, the addition of a lecture provides some "Speakers" preloaded in the "Speaker" combo-box. This makes it easier for the user to pick a frequently used speaker.

    - However, Indico could improve a lot in this point. For instance, there are small details like: when we change the number of “Dates” of a lecture, from 1 to 2, the exact date and time of the first are replicated in the second date; if the day were increased, it would be less probable that the user had to manually change it, as lectures tend to be extended in consecutive days, at the same hour.
- 2:5 - Design for Working Memory Limitations
    - *Do not require users to remember information from place to place on a Web site.*
    - If it is true that Indico does not require users to remember long strings and numbers, codes, etc… on the other hand, it requires from the user a considerable mental effort, in order to remember all the conferences he/she is involved in and where they’re located in the “Indico tree”. The e-mail alerts are not enough in this case. The user should be presented with a list of the conferences he is subscribed to, or manages. It would spare him/her a lot of mental work.
- 2:8 - Display Information in a direcly usable format
    - *Display data and information in a format that does not require conversion by the user.*
    - Indico suffers from this problem, once again, due to the lack of user-centered features. It is impossible to choose the time and date formats which the user is used to: 25/12/2007 or 12/25/2007, 17:00 or 5:00pm, 5:00 CET or 4:00 GMT. This is a strong issue, given that Indico is used all over the world.
- 2:16 - Provide Assistance to Users
    - *Provide assistance for users who need additional help with the Web site.*
    - Indico looks the same for the experienced and the new user. It was designed as an efficient and simple system, but new users will normally have problems understanding some of the fields in a form, or the consequences of a certain action.
    - Some contextual help and slight interface improvements would be enough to make it much easier for the users.

Hardware and Software Issues

- 4:1 - Design for Common Browsers
    - *Design, develop and test for the most common browsers.*
    - Indico is, for now, usable in the most common browsers: Internet Explorer, Firefox, Mozilla, Safari… but there are no extensive tests to browser compliance, and, occasionally, some issues arise.
    - The use of automatic web interface tests could be a way of ensuring that, at least in functional terms, a testable subset of the features would work in different browsers. The design differences between rendering engines would still have to require human evaluation.

The Homepage

- 5:1 - Enable Access to the Homepage
    - *Enable users to access the homepage from any other page on the Web site.*
    - Indico provides a link back to the homepage. The “Home” link is explicit, and at the root of the breadcrumb.
- 5:2 - Show All Major Options on the Homepage
    - *Present all major options on the homepage.*

    - There seems to be no problem regarding the implementation of this guideline, except the “login”. It looks very discrete, and it’s one of the fundamental options for the system.
- 5:3 - Create a Positive First Impression of Your Site
    - *Treat your homepage as the key to conveying the quality of your site.*
    - In spite of being a non-commercial project, Indico could have a more appealing homepage, not only regarding the layout and presentation, but also the options offered to the user.
- 5:8 - Announce Changes to a Web Site
    - *Announce major changes to a Web site on the homepage—do not surprise users.*
    - This guideline calls for the inclusion of news in the main page. Warnings about system maintenance, possible problems, etc… could be presented there.

Page Layout

- 6.2 - Place Important Items Consistently
    - *Guideline: Put important, clickable items in the same locations, and closer to the top of the page, where their location can be better estimated.*
    - Indico seems to follow this guideline.
- 6:7 - Align Items on a Page
    - *Visually align page elements, either vertically or horizontally.*
    - Indico has some problems with form layout. The forms looks unaligned, and that confuses the user . Some buttons are misplaced, too.

Navigation

- 7:2 - Differentiate and Group Navigation Elements
    - *Clearly differentiate navigation elements from one another, but group and place them in a consistent and easy to find place on each page.*
    - Indico complies with this guideline, in most cases. However, some details could be improved: the “Add event” menu, for instance, is very close to the “Tools” menu (general navigation options), and it could be placed at the top of the page, just below the category title.
- 7:4 - Provide Feedback on Users’ Location
    - *Provide feedback to let users know where they are in the Web site.*
    - This is perfectly accomplished by Indico, through the use of breadcrumb trails and nested windows.
    - The nested windows may look confusing sometimes. Maybe some interface improvements would solve this.
- 7:5 - Place Primary Navigation Menus in the Left Panel
    - *Place the primary navigation menus in the left panel, and the secondary and tertiary menus together.*
    - This is an interesting question, which is worth discussing, since Indico does exactly the opposite (main menu on the right), and (at least to my eyes), it doesn’t look bad at all.
- 7:6 - Use descriptive tab labels
    - *Ensure that tab labels are clearly descriptive of their function or destination.*
    - This could be improved. I believe some tabs (for instance, “display” and “protection”) could have a better name.
- 7:7 - Present Tabs Effectively

    - *Ensure that navigation tabs are located at the top of the page, and look like clickable versions of real-world tabs.*
    - This is the kind of thing that only requires aesthetic adjustments (prettier tabs, more “real-world”).

Headings, Titles and Labels

- 9:7 - Use Headings in the Appropriate *HTML* Order
    - *Use headings in the appropriate HTML order.*
    - That’s an important issue, right now. *HTML* headers are not being used as they should, being replaced by bold text or table headers.

Text Appearance

- 11:4 - Ensure Visual Consistency
    - *Ensure that the format of common items is consistent from one page to another.*
    - Indico keeps this kind of consistency.
- 11:9 - Color-Coding and Instructions
    - *When using color-coding on your Web site, be sure that the coding scheme can be quickly and easily understood.*
    - Indico provides color-coding in timetables, and information on the meaning of the colors;

Lists

- 12:1 Order Elements to Maximize User Performance
    - *Arrange lists and tasks in an order that best facilitates efficient and successful user performance.*
    - This point needs to be explored, specially in what concerns about menus and tabs.

Widgets

- 13:1 - Distinguish Required and Optional Data Entry Fields
    - *Distinguish clearly and consistently between required and optional data entry fields.*
    - This should be taken in account, as it is not being followed;
- 13:2 - Label Pushbuttons Clearly
    - *Ensure that a pushbutton’s label clearly indicates its action*
    - Some of the buttons have vague designations, they should be more explicit.
- 13:5 - Label Data Entry Fields Clearly
    - *Display an associated label for each data entry field to help users understand what entries are desired.*
    - This aspect seems well implemented in Indico.
- 13:6 - Minimize User Data Entry
    - *Do not require users to enter the same information more than once.*
    - Perhaps we could extract more information from NICE accounts, or store more information about the user in the database. However, the auto-completion is working pretty well.
- 13:11 - Anticipate Typical User Errors

  - *Use the computer to detect errors made by users.*
  - Indico checks dates and e-mails, but there should be a first layer of client-side verification, so that the used would be spared from a page transition (and the server from a page request).
- 13:13 - Use a Single Data Entry Method
  - *Design data entry transactions so that users can stay with one entry method as long as possible.*
  - Indico forms look consistent with this policy, with some exceptions. The input method for dates should be reconsidered, too.
- 13:18 - Display Default Values
  - *Display default values whenever a likely default choice can be defined.*
  - Indico forms already provide default values;
- 13:19 - Place Cursor in First Data Entry Field
  - *Place (automatically) a blinking cursor at the beginning of the first data entry field when a data entry form is displayed on a page.*
  - This issue needs to be fixed, as it improves a lot the agility on the user side;

Writing Web Content

- 15:1 - Make Action Sequences Clear
  - *When describing an action or task that has a natural order or sequence (assembly instructions, troubleshooting, etc.), structure the content so that the sequence is obvious and consistent.*
  - Maybe this guideline could be adopted for user input in forms. It would make it much more intuitive.
- 15:4 - Define Acronyms and Abbreviations
  - *Do not use unfamiliar or undefined acronyms or abbreviations on Web sites.*
  - Indico is very likely to display acronyms (we're talking about scientific institutions). There could be a mechanism of acronym definition.

Content Organization

- 16:5 - Minimize the Number of Clicks or Pages
  - *Guideline: To allow users to efficiently find what they want, design so that the most common tasks can be successfully completed in the fewest number of clicks.*
  - Indico is definitely needing a lot of work in this area. Users complain a lot about having to click and to change of page too many times.
- 16:9 - Use Color for Grouping
  - *Use color to help users understand what does and does not go together.*
  - Colors could be used more extensively, for instance, in order to distinguish the different types of events. However, information should never rely only on color alone (see *WCAG*).

Standards Compliance

REPENT!
THE ROAD TO HELL IS PAVED WITH SPACER GIFS AND NESTED TABLES!

(from "Why tables for layout is stupid"[68])

HTML

It is known that the project started in 2001, when the HTML 4.01 specification was still very recent, and a year had passed since its definition as an ISO standard1. The W3C chose to discontinue HTML, in favor of its successor, XHTML, an XML (in opposition to SGML) conformant language. Nowadays (as of 2007), HTML is slowly getting into disuse, while the second version of XHTML is still being drafted. Indico would be easily portable to newer formats (XHTML 1.1), or future versions, if it fully accomplished the HTML 4.01 Strict specification. The problem is that Indico is currently built using HTML 4.01 Transitional , and, even considering this, it cannot fully comply with the W3C specifications.

The main reason has to do with the use of some obsolete attributes, and the placement of forms within tables and nesting of tables. The main problem with HTML 4.01 Transitional is, in contrary to what most people would think, browser support. For instance, Internet Explorer only enforces some "CSS standard behaviors" (like correct support for the "fixed" alignment) if working under strict mode. Being so, Indico probably faces more cross-browser incompatibilities for using a transitional specification, than it would do if it were based on a more recent standard. Being so, I consider the adoption of HTML 4.01 Strict a priority for the modernization of Indico (given that support for XHTML from the part of Internet Explorer is very poor), and the improvement of its accessibility and usability.

ECMAScript

The poor browser support provided for ECMAScript (or its ancestors JavaScript and JScript) was one of the problems that kept, for years, many web developers from implementing dynamic client-side content. The problem had mainly to do with the differences existing between two rival implementations (Netscape's JavaScript, and Microsoft's JScript), and to the incomplete implementation that many browsers provided. Since 1996, following the creation of the language, an harmonization effort has been carried on by the ECMAScript Committee, in order to produce standards that could at the same time satisfy the demand of web applications, and remain portable between browsers and platforms. Even of 2007, there are still differences between the two major implementations. However, these are easy to overcome, and, if we take into account the existence of many frameworks which provide a compatibility layer, (for instance, Prototype), they are barely noticeable. Being so, it is not a surprise that so many web applications now provide rich and dynamic web interfaces, some of them even presenting the whole site in a single, dynamic page. As from 1998, the *W3C* provides a standard for the ECMAScript DOM bindings which browsers should implement.

[68] http://www.hotdesign.com/seybold/

Indico seen by its users

The results from the User Survey are described by the document "Indico Usability Survey - Results and Remarks"[69].

Conclusion

This report provides a an overview of the main usability and accessibility and usability problems found within Indico, using both a "cold, scientific" perspective and a user-centered manner. Although it cannot be claimed, in any way, that most usability problems were found (since no analysis is so broad and exhaustive), it is certain that at least the most significant problems (those which trouble users the most) were identified.

The proposed solutions should be implemented, down to the second level of priority, as soon as possible. Those from the third level should be considered in a long-term plan, but definitely implemented.

References

1. **W3C** (1999) *Web Content Accessibility Guidelines 1.0.* (http://www.w3.org/TR/WAI-WEBCONTENT/)
2. **W3C** (2006) *Roadmap for Accessible Rich Internet Applications (WAI-ARIA Roadmap).* (http://www.w3.org/TR/aria-roadmap/)
3. **U.S. Dept. of Health and Human Services** (2006) *Research-Based Web Design and Usability Guidelines.* U.S. Dept. of Health and Human Services.
4. **Jack J. Baroudi , Margrethe H. Olson and Blake Ives** (1986) *An empirical study of the impact of user involvement on system usage and information satisfaction.* Commun. ACM, 29(3):232–238.
5. **Jesse James Garrett** (2005) *Ajax: A New Approach to Web Applications.* (http://www.adaptivepath.com/publications/essays/archives/000385.php)
6. **Roy T. Fielding and Richard N. Taylor** (2002) *Principled Design of the Modern Web Architecture".* (http://citeseer.ist.psu.edu/fielding02principled.html)
7. **Ali Mesbah and Arie van Deursen** (2007) *An Architectural Style for Ajax.* (http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0608111)
8. **Jakob Nielsen** (2000) *Why You Only Need to Test With 5 Users.* (http://www.useit.com/alertbox/20000319.html)
9. **Pedro Ferreira** (2007) *Indico Usability Survey – Results and Remarks*

[69] (Ferreira:2007)

Appendix: Detailed accessibility analysis

Guidelines

- Provide equivalent alternatives to auditory and visual content. 0/1
  - Checkpoint 1.1 - **Fails**
  - Checkpoint 1.2 - **N/A**
  - Checkpoint 1.3 - **N/A**
  - Checkpoint 1.4 - **N/A**
  - Checkpoint 1.5 - **N/A**

Note
1.1 - there are images with no *alt* attribute

- Don't rely on color alone. 2/2
  - Checkpoint 2.1 - **OK**
  - Checkpoint 2.2 - **OK**

Note
2.2 - is Indico visible to colorblind people? Check here[70].

- Use markup and style sheets and do so properly. 2/6
  - Checkpoint 3.1 - **OK**
  - Checkpoint 3.2 - **OK**
  - Checkpoint 3.3 - **Fails**
  - Checkpoint 3.4 - **Fails**
  - Checkpoint 3.5 - **Fails**
  - Checkpoint 3.6 - **Fails**
  - Checkpoint 3.7 - **N/A**

Note
3.3 *<b></b>* is used through the templates. *<strong></strong>* should be used instead.

3.4 Absolute units are used

3.5 Headings are not used

3.6 The menu, for instance, could be marked up as a list

- Clarify natural language usage 0/3
  - Checkpoint 4.1 - **Fails**
  - Checkpoint 4.2 - **Fails**
  - Checkpoint 4.3 - **Fails**

Note
4.2 Acronyms are never used… should they be used?

- Create tables that transform gracefully 2/5
  - Checkpoint 5.1 - **OK**
  - Checkpoint 5.2 - **Fails**

Note
5.2 Logical levels are not separated, inside tables

5.3 Tables are extensively used for layout

5.5 No table summaries

[70] http://colorfilter.wickline.org/

- Checkpoint 5.3 - **Fails**

- Checkpoint 5.4 - **OK**
- Checkpoint 5.5 - **Fails**
- Checkpoint 5.6 - **N/A**

Note
11.2 things like *<img … border="1" />* are deprecated

- Ensure that pages featuring new technologies transform gracefully 4/5

  - Checkpoint 6.1 - **OK**
  - Checkpoint 6.2 - **Fails**
  - Checkpoint 6.3 - **OK**
  - Checkpoint 6.4 - **OK**
  - Checkpoint 6.5 - **OK**

Note
6.2 There are no *<noscript>* tags.

6.3 Of course, the badge and poster editors are exceptions

- Ensure user control of time-sensitive content changes. 0/0
  - Checkpoint 7.1 - **N/A**
  - Checkpoint 7.2 - **N/A**
  - Checkpoint 7.3 - **N/A**
  - Checkpoint 7.4 - **N/A**
  - Checkpoint 7.5 - **N/A**

- Ensure direct accessibility of embedded user interfaces 0/1

  - Checkpoint 8.1 - **Fails**

Note
8.1 There are no accessibility concerns about the scripted parts. A mouse is definitely needed for badge editing.

- Design for device-independence 2/4

  - Checkpoint 9.1 - **N/A**
  - Checkpoint 9.2 - **Fails**
  - Checkpoint 9.3 - **OK**
  - Checkpoint 9.4 - **OK**
  - Checkpoint 9.5 - **Fails**

Note
9.1 See 8.1

9.5 There seem not to exist any keyboard shortcuts

- Use interim solutions 2/2
  - Checkpoint 10.1 - **OK**
  - Checkpoint 10.2 - **OK**
  - Checkpoint 10.3 - **N/A**
  - Checkpoint 10.4 - **N/A**
  - Checkpoint 10.5 - **N/A**
- Use W3C technologies and guidelines 2/3

  - Checkpoint 11.1 - **OK**
  - Checkpoint 11.2 - **Fails**

- Checkpoint 11.3 - **OK**
- Checkpoint 11.4 - **N/A**

- Provide context and orientation information 0/2

  - Checkpoint 12.1 - **N/A**
  - Checkpoint 12.2 - **N/A**
  - Checkpoint 12.3 - **Fails**
  - Checkpoint 12.4 - **Fails**

Note
12.3 Elements are not logically grouped

12.4 There are input fields with no labels associated

- Provide clear navigation mechanisms 7/9

  - Checkpoint 13.1 - **OK**
  - Checkpoint 13.2 - **Fails**
  - Checkpoint 13.3 - **OK**
  - Checkpoint 13.4 - **OK**
  - Checkpoint 13.5 - **OK**
  - Checkpoint 13.6 - **OK**
  - Checkpoint 13.7 - **OK**
  - Checkpoint 13.8 - **Fails**
  - Checkpoint 13.9 - **OK**
  - Checkpoint 13.10 - **N/A**

Note
13.2 Metadata is missing

13.8 Indico’s main page is an example of that

- Ensure that documents are clear and simple 3/3
  - Checkpoint 14.1 - **OK**
  - Checkpoint 14.2 - **OK**
  - Checkpoint 14.3 - **OK**

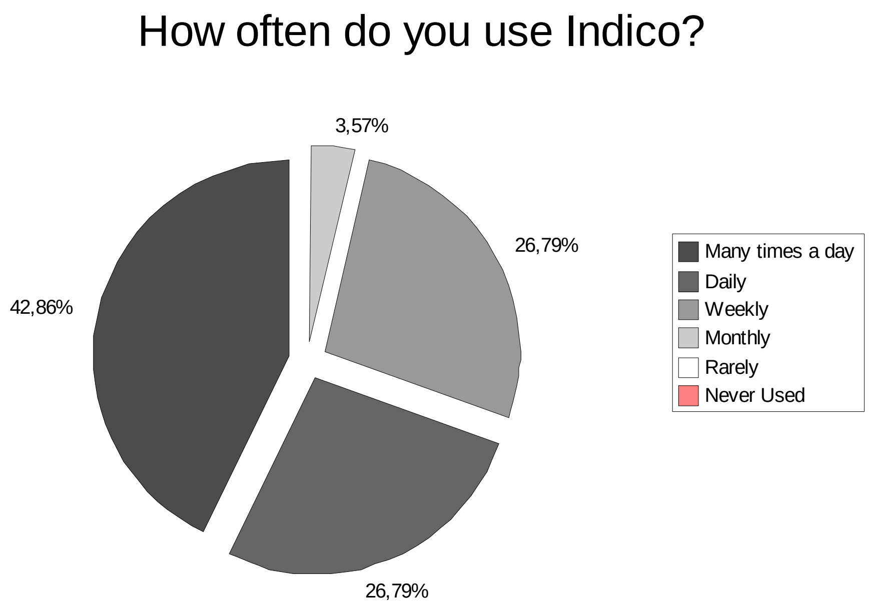
APPENDIX B: ***"Indico Usability Survey – Results and Remarks"***

Indico Usability Survey – Results and Remarks

José Pedro Ferreira, August 2007 – CERN IT-UDS-AVC

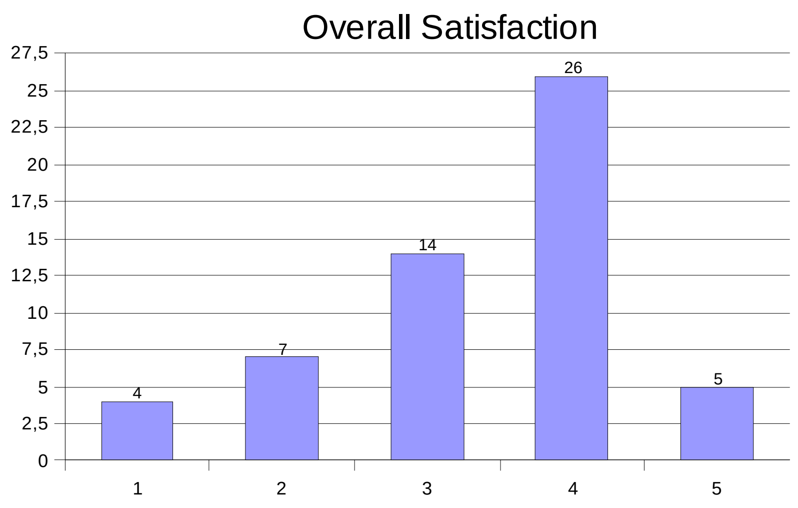
Data

- **56 voluntary samples, gathered on-line;**
- **From June to mid-July**



- Maybe there's some bias, introduced by the voluntary nature of the survey. It is probable that most of the users who answered the survey are interested in doing so, because they need Indico for they work, thus they are probably frequent users of Indico.

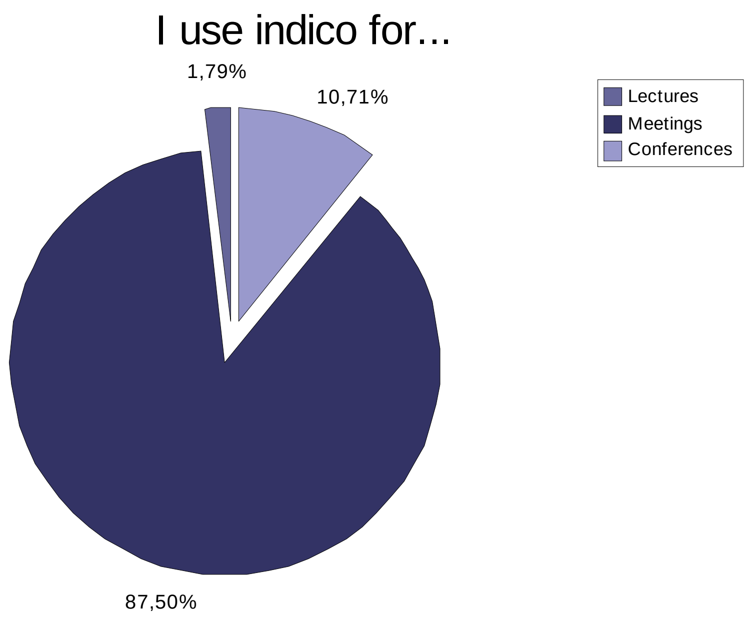
Are you satisfied with Indico?



Average	3,38
Std. Dev	0,87
Median	4
Moda	4

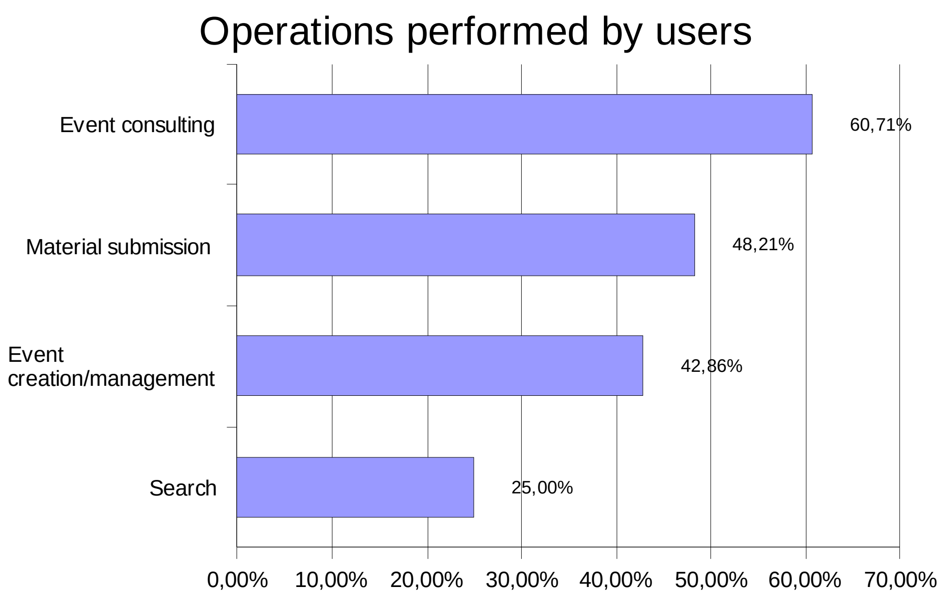
- From *1 - "No, definitely, not"* to *5 - "Definitely, yes"*;
- There should be a bias, in the negative direction. People will probably tend to answer the survey if they're unhappy with Indico. However, it's only a supposition, with no statistical value;

You use Indico mostly for...?

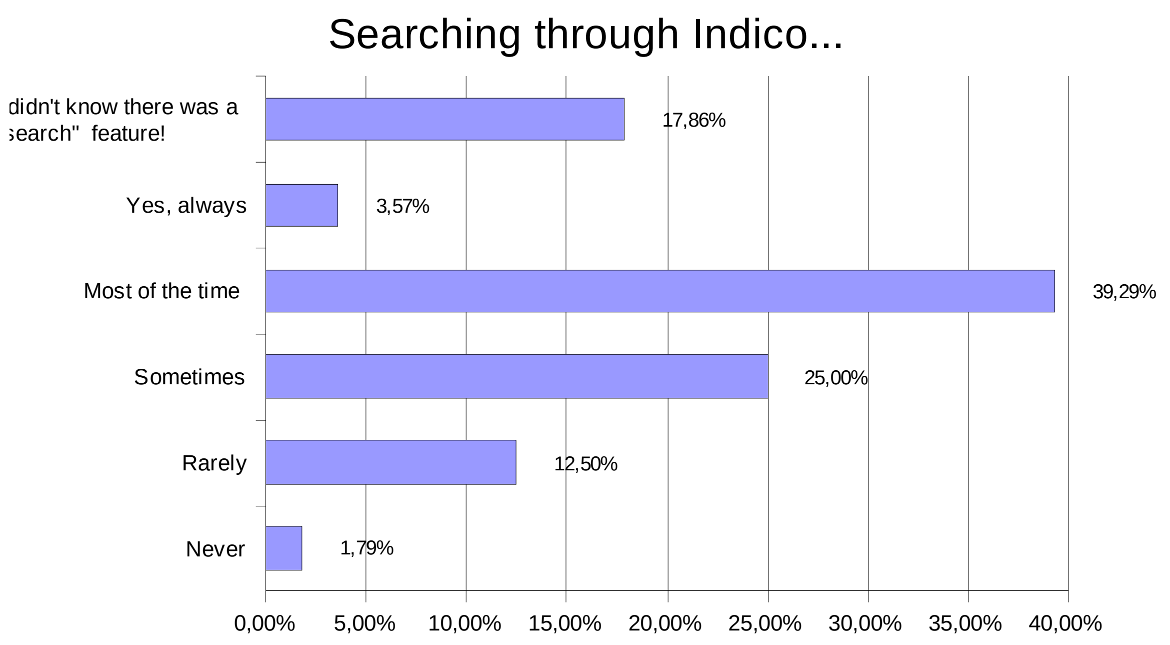


- This quite "overwhelming" value for the "meetings" category is no surprise. People who use Indico regularly probably use it for meeting appointment, since meetings are more frequent than lectures or conferences;

What kind of operations do you perform most?



When searching through Indico, do you usually find out what you want?



- The “I didn't know there was a search feature” group is significantly large, compared to the other ones. It is natural, since the feature was only recently introduced;
- The balance is negative: “Didn't know” + “Sometimes” + “Rarely” + “Never” > 50%

How would you rate the usability of the following features (1-5 or N/A)?

	N/A	Resp. Rate	Average	Median	Moda	Std.Dev
Event creation dialogs	16	0,71	3,18	3	3	0,88
Material management	9	0,84	3,62	4	4	0,87
Abstract management (conferences)	32	0,43	3,46	4	4	1
Contribution management (conferences)	29	0,48	3,41	4	4	1,03
Registration/Payment configuration (conferences)	43	0,23	3,23	3	3	0,78
Event page configuration (conferences)	34	0,39	3,41	3	3	0,94
Timetable management (conferences/meetings)	12	0,79	3,25	3	2	1,02
Listings (conferences/meetings)	19	0,66	3,35	3	3	0,94
Participant management (lectures/meetings)	20	0,64	3,03	3	3	0,98
Acess control management	15	0,73	2,9	3	3	0,96
Extra tools (badge editor, offline website…)	39	0,3	3,12	3	3	0,87
Logging management	29	0,48	3,19	3	3	0,99

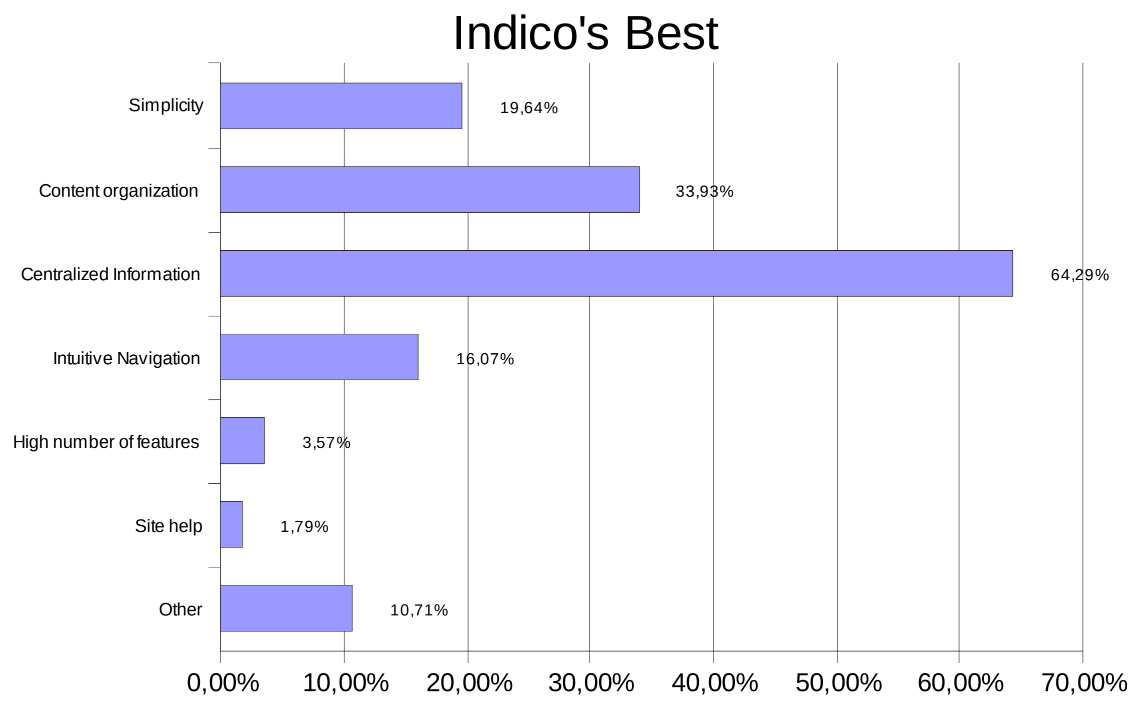
- There's nothing we can tell that is too good or too bad;

Please rate Indico in terms of...

	N/A	Resp. Rate	Average	Median	Moda	Std.Dev
Design	2	0,96	3,41	3	3	0,9
Menus	3	0,95	3,13	3	4	0,91
Context Help	12	0,79	2,77	3	3	0,82
Accessibility	7	0,88	3,39	3	3	0,86
Organization	4	0,93	3,06	3	3	0,92
Performance	2	0,96	3,17	3	4	1,07
Reliability	3	0,95	3,19	3	4	1,07
Search Options	20	0,64	2,61	3	2	0,91

- Once again, poor knowledge about search options;

What do you like best in Indico?

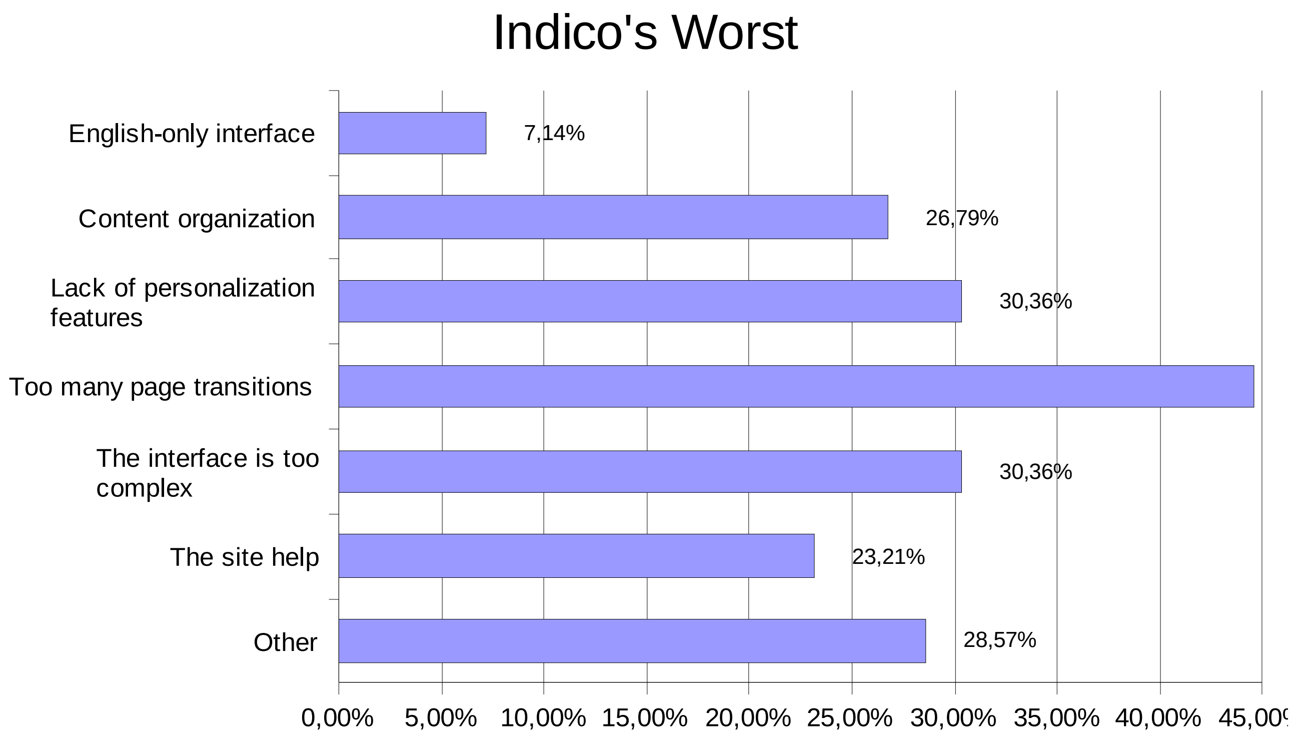


- “Site help” is no surprise;
- “Centralized Information” is a clear winner;
- “Intuititive navigation” is probably the loser;

Other field:

- o “CDS agenda server is much easier to use”;
- ✓ “Database - it keeps talks transparencies forever”;
- o “To be honest, compared to cds, very little”;
- ✓ “Content repository”;
- o “Nothing”;
- ✓ “Ability to reschedule an item in the timetable and have all following items adjusted accordingly”;

What do you like least in Indico?



- People are more likely to click checkboxes in the "least" question, than in the "best";
- "Too many page transitions" is an expected winner;
- "English only interface" a not very surprising loser;
- Almost 30% of the sample has left its custom suggestion;

Other field:

- "Some stuff just doesn't work (paypal..), convener don't see what other conveners do (abstract management - abstract ins multiple tracks), control management: convener need an easy to use interface-diff. between convener, access management is a real nightmare";
- "Content inflexibility - how about templating?";
- "Search feature is crappy";
- "Unfixed bugs";
- "Search";
- "Not at all intuitive";
- "Very verbose, non-intuitive, too many non-useful functions";
- "Many features and tools are hard to find and not customizable";
- "Search always gets it wrong";
- "Slowness";
- "Content organization - Category-based structure; (hard to guess where things are,

why not use tags?)";

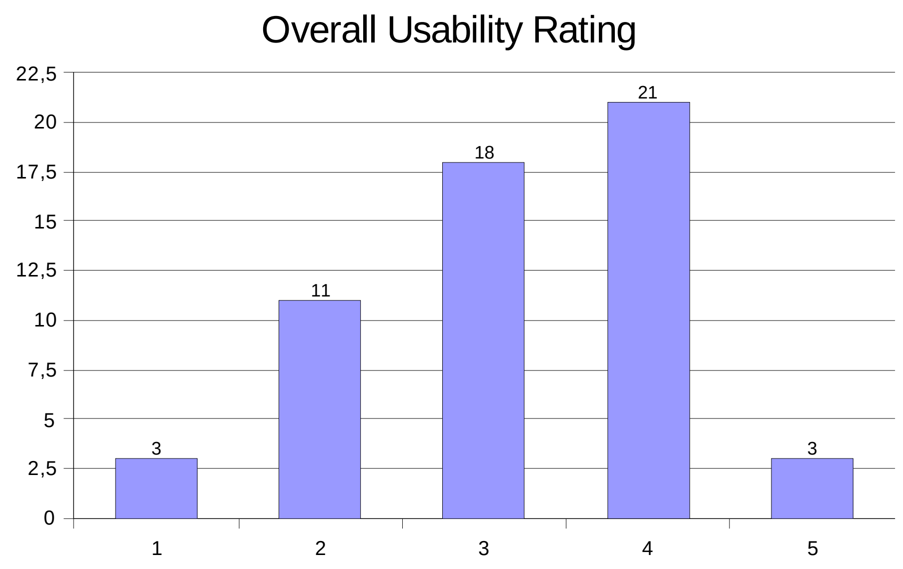
- "Poor reliability";
- "UI is not intuitive. Cursor doesn't default to most obvious field in each form";
- "Outdated html code";
- "Many crashes";

How would you rate Indico's usability according to each of the following features?

	N/A	Resp. Rate	Average	Median	Moda	Std.Dev
Category browsing	10	0,82	3,39	4	4	0,9
Login/Logout pages	8	0,86	3,63	4	4	0,79
Event pages	8	0,86	3,67	4	4	0,75
Event management pages	15	0,73	3,2	3	4	1,04
Calendar	15	0,73	3,39	4	4	0,87
Search	19	0,66	2,78	3	2	0,99
Help	20	0,64	2,78	3	3	0,78

- Once again, the "Search" is rated low;

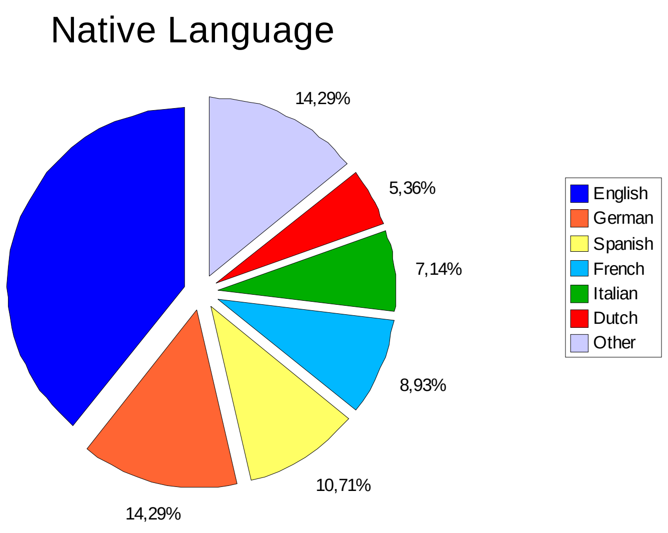
How would you rate Indico's overall usability?



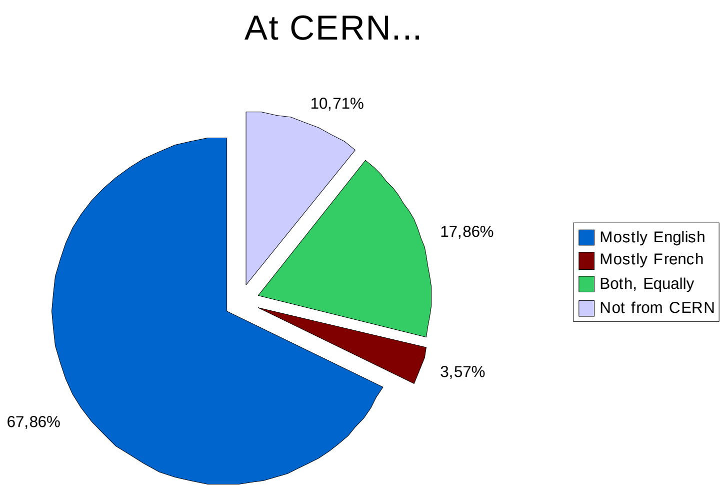
Average	3,18
Std. Dev	0,81
Median	3
Moda	4

- Positive overall usability feedback;

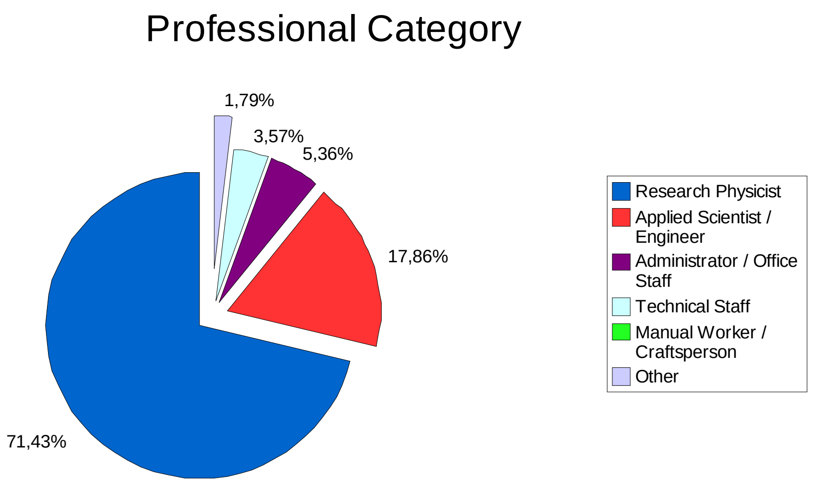
“Demographics”



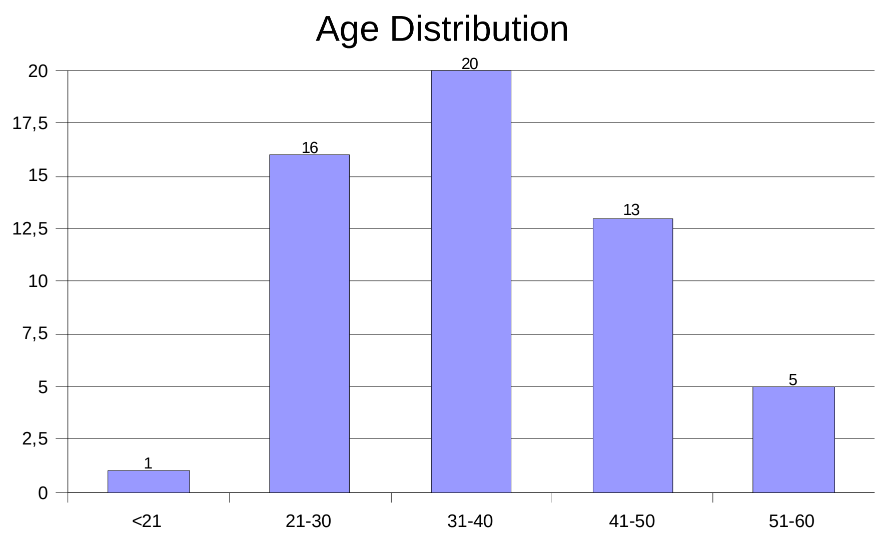
Language spoken at CERN



Professional Category



Age Groups


Age Distribution
20
17,5
15
12,5
10
7,5
5
2,5
0
1
16
20
13
5
<21
21-30
31-40
41-50
51-60

APPENDIX C: "WOHL Specification"

WOHL

Web-oriented Help-specification Language - Specification - v0.2

Pedro Ferreira, CERN IT-UDS-AVC, April-August 2007

Introduction

WOHL is a context help specification language, developed for Indico, so that regular content and auxiliary information would not end mixed up, and no hardcoding of tooltips would exist. Its aim is to provide an abstract syntax for definition of constructs such as tooltips and informative hints. The format allows, as well, the inclusion of references to external documentation.

WOHL was built to enrich (X)HTML documents, and is mainly focused on this technology as its "host". However, virtually any XML-based format (or HTML-based, under the circumstance that DOM parsing is possible) can be addressed as the target document format. The process of "enriching" is done through injection of the "decorating elements" in the code, through the specification of the target areas, through XPath.

"WOHL" should be pronounced approximately as "*vaul*" (IPA [voːl]), the German word for "well".

Requirements:

- Adapt to Indico's needs, without losses in extensibility and without making it too *indico-specific*;
- Provide a high-level and extensible language for context help specification;
- Total absence of formatting specifications (color, font, etc…) - only content and purpose;
- Separate regular content from auxiliary content;
- Different types of help mechanisms:
    - Tooltips - First one to emerge
        - Icon-based (default) - A tooltip icon is added to the target object in the document;
        - Object-based - Triggered by some event related to an existing object;
    - Informative areas - static blocks of information, associated to some particular container;

Example

```
<?xml version="1.0"?>
<wohl xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="wohl.xsd" version="0.1">
	<page name="SearchBox">
		<tooltip name="searchBar" target="//*[@id='searchBarTip']">
```

```
            <translation language="en">
                <text>
                    Indico\'s <strong>search</strong> feature only searches for <strong>publicly available
content</strong>. <strong>No data from protected events will be displayed.</strong>
                </text>
            </translation>
        </tooltip>
    </page>
</wohl>
```

Specification

EBNF-like

(Very simplified - no attributes)

- wohl ::= (page)*
- page ::= (tooltip|area)*
- area ::= (translation)*
- tooltip ::= (translation)*
- translation ::= text (helpref)*

XSD

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <xsd:element name="wohl" type="WOHL"/>

   <xsd:complexType name="WOHL">
      <xsd:all>
         <xsd:element name="page" type="Page" minOccurs="1"/>
      </xsd:all>
      <xsd:attribute name="version" type="xsd:string" use="required" />
   </xsd:complexType>

   <xsd:complexType name="Page">
      <xsd:sequence>
         <xsd:element name="tooltip" type="Tooltip" minOccurs="0" maxOccurs="unbounded" />
         <xsd:element name="information" type="Information" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string" use="required" />
   </xsd:complexType>

   <xsd:complexType name="Tooltip">
      <xsd:sequence>
         <xsd:element name="translation" type="Translation" minOccurs="1" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="target" type="xsd:string" use="required" />
      <xsd:attribute name="type" type="ttType" default="explicit" />
   </xsd:complexType>

   <xsd:complexType name="Information">
      <xsd:sequence>
         <xsd:element name="translation" type="Translation" minOccurs="1" maxOccurs="unbounded" />
```

```
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string" use="required" />
        <xsd:attribute name="target" type="xsd:string" use="required" />
    </xsd:complexType>

    <xsd:complexType name="Translation">
        <xsd:sequence>
            <xsd:element name="text" type="Text" minOccurs="1" maxOccurs="1" />
            <xsd:element name="helpref" type="HelpRef" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="language" type="xsd:string" use="required" />
    </xsd:complexType>

    <xsd:complexType name="HelpRef">
        <xsd:attribute name="url" type="xsd:string" use="required" />
        <xsd:attribute name="text" type="xsd:string"/>
    </xsd:complexType>

    <xsd:complexType name="Text" mixed="true">
        <xsd:choice minOccurs="0"  maxOccurs="unbounded">
            <xsd:element name="strong" type="Text"/>
            <xsd:element name="em" type="Text" />
            <xsd:element name="important" type="Text" />
        </xsd:choice>
    </xsd:complexType>

    <xsd:simpleType name="ttType">
        <xsd:restriction base="xsd:string">
        <xsd:pattern value="explicit|hover"/>
      </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```