Faculdade de Engenharia da Universidade do Porto

Sistema inteligente de gestão de serviços heterogéneos de videoconferência

InDiCo Integration and Services Management of Heterogeneous Videoconferencing Systems

Eng. João Fernandes

VERSÃO FINAL

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Telecomunicações

Orientador: Prof. Maria Teresa Andrade

Junho de 2008

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2007/2008

A Dissertação intitulada

"Sistema Inteligente de Gestão de Serviços Heterogéneos de Videoconferência"

foi aprovada em provas realizadas 18/Julho/2008

o júri

Presidente Professor Doutor Eurico Manuel Elias Morais Carrapatoso
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto

[illegible]

Professor Doutor José Manuel de Castro Torres
Professor Auxiliar da Faculdade de Ciência e Tecnologia da Universidade Fernando Pessoa

[illegible]

Professora Doutora Maria Teresa Magalhães da Silva Pinto de Andrade
Professora Auxiliar da Faculdade de Engenharia da Universidade do Porto

[illegible]

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor - João Carlos Correia Fernandes

[illegible]

Faculdade de Engenharia da Universidade do Porto

Resumo

Este projecto pretende especificar e desenvolver uma arquitecura inovadora para um serviço flexível que permita fazer a gestão de pedidos de clientes colaborativos heterogéneos (no sentido em que se podem utilizar diferentes sistemas de colaboração baseados em videoconferência, teleconferencia ou webconferencing) e a interligação com o sistema InDiCo. O sistema InDiCo consiste numa aplicação desenvolvida sob a licença GNU GPL (General Public License) que oferece funcionalidades de gestão de eventos tais como agenda ou repositório e arquivo de dados associados a conferências e videoconferências.

Pretende-se que o serviço a desenvolver possa ser acedido de uma forma universal, atendendo pedidos de utilizadores heterogéneos. O sistema deverá ocupar-se das ligações necessárias e de estabelecer a ponte com os eventos geridos através do sistema InDiCo.

Desta forma, o serviço a especificar e desenvolver pode ser visto como um módulo adicional ou plug-in do sistema InDiCo devendo permitir tornar transparente para o utilizador toda a complexidade de iniciar uma videoconferência, conferência telefónica, conferência mista de video sobre IP + telefone, etc. O serviço apresenta a ainda vantagem de se poder adicionar ou remover funcionalidade sem se alterar a sua arquitectura base.

Este serviço está a ser implementado adoptando uma abordagem de Serviços Web, que oferece a possibilidade de efectuar de um conjunto de operações e trocar mensagens usando APIs de sistema (por exemplo, baseadas no protocolo XML-RPC). No ambito do projecto, a prioridade foi dada ao design e especificação da arquitectura. Um prototipo foi implementado que reflecte um dos casos de uso. Tendo em conta que o design do módulo foi já formalizado, espera-se que os restantes caso de uso possam ser implementados rapidamente.

Numa fase posterior, pretendeu-se efectuar uma análise sumaria de possiveis funcionalidades 3G adicionais a implementar com o intuito de suportar a troca de mensagens transportando características e capacidades do terminal e mesmo preferências do utilizador (informação de contexto) para que o serviço gestor possa tomar decisões quanto à necessidade de adaptar o conteúdo a ser transferido (audio, vídeo e gráficos) nomeadamente para dispositivos 3G. Esta abordagem permitiria, por exemplo, que utilizadores usando diferentes dispositivos para além do tradicional sistema de videoconferência instalado numa sala ou o tradicional telefone (um dispositivo movél 3G com camara incorporada) possam também juntar-se à videoconferência, sendo-lhes enviada uma versão de mais baixa resolução vídeo. Foi efectuada uma pequena introdução a esta análise no ambito deste projecto.

Abstract

This project intends to specify a new architecture for a flexible service that allows the management of connections of heterogeneous collaborative services (traditional videoconferencing, phone conferencing or Web conferencing) as a module integrated in InDiCo system. InDiCo stands for INtegrated DIgital CoOnference and it was developed under the GNU GPL license. It basically consists in a conference management tool that allows scheduling events, from simple talks to complex conferences with many sessions and contributions.

This innovative module is aimed to be included in InDICo as a plug-in. It will allow to exponential reduce and hide the complexity for a common user to initiate a videoconference, phone conference or mixed conference (videoconferencing over IP + telephone), etc.

The service to be specified relies on a universal architecture intended to allow the management of connections of heterogeneous users and endpoints (traditional videoconferencing over IP endpoints, traditional telephone terminals, etc.), where adding or removing functionality will not change the module architecture. The implementation of the service is following a Web Services approach, allowing implementing a set of operations by exchanging messages using system APIs (for example, based on the XML-RPC protocol). Attention has been given to the design and specification of the module. A prototype with one of the use cases has been implemented. The following ones are expected to be done more rapidly as the design was already formalized.

In a later phase, there was also the intention to make an introductory analysis in order to add "intelligent" functionalities namely in what concerns 3G devices. These functionalities would allow not only terminal connection but also the transport of information related to the terminal capacities and user preferences. In this way, the module can take decisions about which kind of content should be negotiated and eventually sent (audio video and data). This would allow for example, a 3G mobile device to join the distributed meeting, receiving a video stream in agreement with the terminal capacities (lower video resolution). A short introductory approach is done in the scope of this project.

Agradecimentos

Quero agradecer ao meu Pai, à minha Mãe e ao meu Irmão Pedro pelo apoio e conselhos de toda a vida e pelas oportunidades e formação que me deram.

Gostaria tambem de agradecer à Sofia pelo apoio, conselhos dados e paciência.

Contents

Table of Figures

Abbreviations and Acronyms

3G	Designation referring to Third Generation Mobile Networks
API	Application Program Interface
ATLAS	A Toroidal LHC ApparatuS
CERN	European Centre for the Nuclear Research
CHEP	Computing High Energy Physics
ESnet	Energy Sciences Network
EVO	Enabling Virtual Organizations
EXE	Executable
FEUP	Faculdade de Engenharia da Universidade do Porto
GNU GPL	General Public License
H.221	An ITU standard for the framing structure of a videoconferencing transmission over a 64 to 1920 Kbits/sec channel
H.234m	A telecommunications standard for mobile networks
H.323	Umbrella Recommendation from the ITU Telecommunication Standardization Sector (ITU-T) that defines the protocols to provide audio-visual communication sessions on any packet network
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IETF	Internet Engineering Task Force
InDiCo	INtegrated DIgital COnference
IT-UDS-AVC	Information and Technology-User and Document Services-Audiovisual and Conferencing Services
JSON	JavaScript Object Notation
LHC	Large Hadrons Collider
MCU	Multipoint Control Unit
OCS	Office Communications Server
PABX	Private Automatic Branch Exchange

PSTN	Public Switching Telephone Network
SIP	Session Initiation Protocol
VC	Videoconferencing
XML-RPC	eXtended Markup Language
ZODB	Python object Persistence System
ZOPE	open source application server for building content management

Chapter 1

Introduction

This document reports the accomplished and undergoing work at European Organization for Nuclear Research (CERN[1]) in the IT-UDS-AVC[2] section (Audiovisual and Collaborative Services) in the scope of specifying and developing a ”Unified Communications module” for the InDiCo[3] Web application.

1.1 Goal

The availability and the quantity of remote collaboration systems to the users is, nowadays, quite large. Systems based over different networks (being the Internet or traditional phone lines) have a concrete usage, even if in many cases its operation is not as easy as one would wish. The future direction seems to be to turn them as easy to use as a TV set or a traditional telephone but, it was found that there is still a lot of work to do in this domain. Remote dispersed working groups use different collaborative systems for a wide variety of reasons: travel cost, user practices, network availability, physical location requirements, etc. Some of these issues, in some cases turn even more difficult the operation to set up a working session between people remotely dispersed.

The main objective of this project is to study approaches and provide a solution to the above mentioned problem, by unifying the remote communications operations under a single interface. To achieve this task, a novel modular architecture has been specified to be integrated in the CERN InDiCo Framework. This specification is being followed by an ongoing implementation. The system will provide in a first phase, the functionality to perform a universal management of connections of heterogeneous collaborative systems. The implementation of the complete functionality will allow to decrease the complexity of operations and to simplify end-user operations thus increasing the quality of the user experience.

[1] http://www.cern.ch
[2] Http://www.cern.ch/it-multimedia
[3] http://indico.cern.ch

1.2 CERN

The ongoing project is taking place at CERN. CERN, the European Organization for Nuclear Research, is one of the world's largest and most respected centres for scientific research. Its business is fundamental physics, finding out what the Universe is made of and how it works. At CERN, the world's largest and most complex scientific instruments are used to study the basic constituents of matter: the fundamental particles. By studying what happens when these particles collide, physicists learn about the laws of Nature. The instruments used at CERN are particle accelerators and detectors. Accelerators boost beams of particles to high energies before they are made to collide with each other or with stationary targets. Detectors observe and record the results of these collisions. Due to these challenges and others that have to be responded, CERN has also a very strong research activity in computing related areas such as the Grid Computing[4].

CERN was founded in 1954, and the Laboratory sits astride the Franco–Swiss border near Geneva. It was one of Europe's first joint ventures and now has 20 Member States. Portugal is a Member State since 1985. It involves more than 10000 scientists and engineers, where approximately 2600 employed directly by the Organization and roughly 8000 from 500 Universities from Europe and rest of the World. In Figure 1.2.1 one can see a map of the CERN complex.

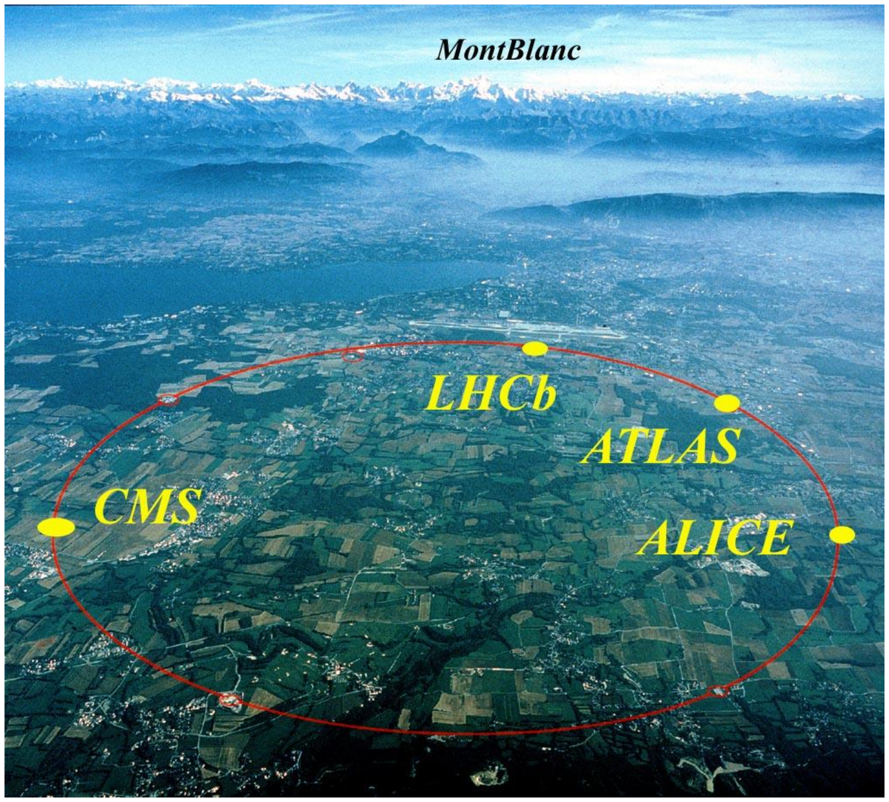


Figure 1.2.1 – CERN complex, in the Franco-Swiss border.

[4] Grid Computing consists in a set of services for computational and storage tasks and activities supported by a pool of distributed computing resources.

The current main focus of scientists and engineers at CERN is the LHC[5] experiment. The LHC is an accelerator which brings protons and ions into head-on collisions at higher energies than ever achieved before. This will allow scientists to penetrate still further into the structure of matter and recreate the conditions prevailing in the early universe, just after the "Big Bang". The LHC is built astride the Franco-Swiss border west of Geneva, at the foot of the Jura Mountains, in front of the Alps, installed in a tunnel 27 km in circumference, buried 100m underground.

LHC will produce head-on collisions between two beams of particles travelling through a vacuum comparable to outer space. Each beam will consist of almost 3000 bunches of 100 billion particles each. At full power, each beam will be about as energetic as a car moving at 1600 km/h. 3000 km of wires and fibres will carry information at the rate of 3200 terabytes per year, equivalent to around 3 billion books.

LHC has for Particle Detectors (Experiments), ATLAS[6], CMS[7] Alice[8] and LHCb[9]. In Figure 1.2.2 one can see the LHC tunnel with the four LCH Experiments.

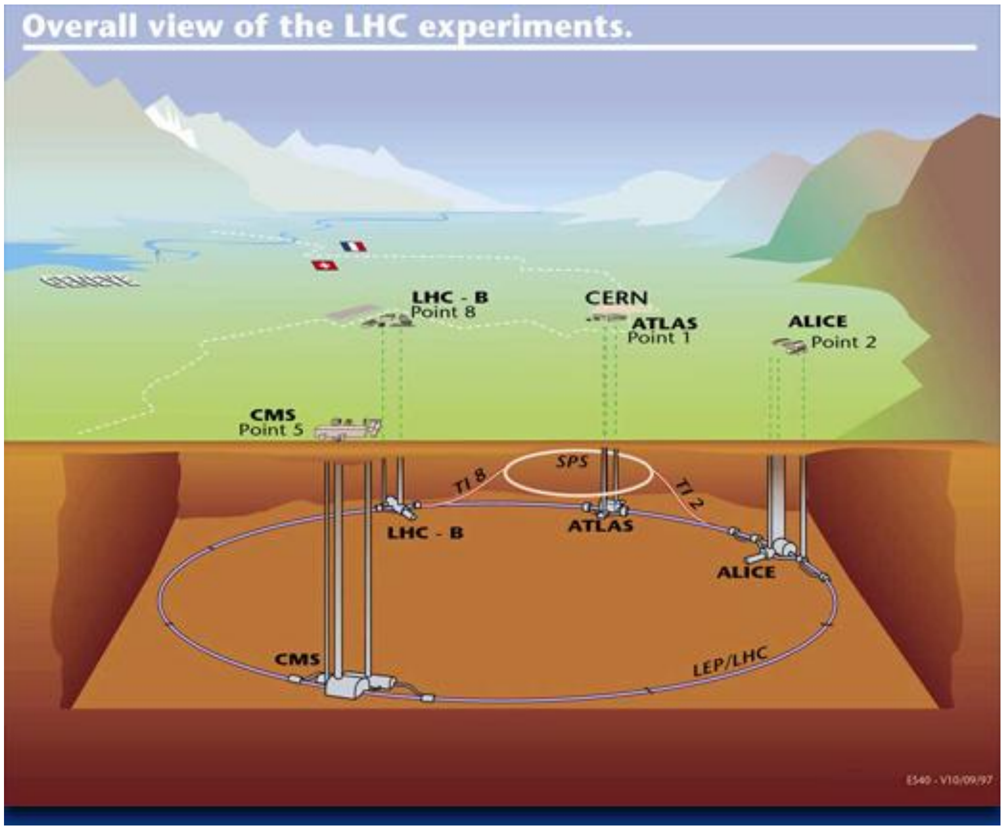


Figure 1.2.2 – LHC Experiments.

[5] The LHC (Large Hadrons Collider) is the particle accelerator at CERN in the process of being finished

[6] ATLAS Experiment (*A Toroidal LHC ApparatuS)* is one of the 4 particle detectors being finalized for the LHC.

[7]CMS stands for Compact Muon Solenoid. Like ATLAS, is one of the general purpose detectors for the LHC.

[8]ALICE Collaboration is building a dedicated heavy-ion detector to exploit the unique physics potential of nucleus-nucleus interactions at LHC energies.

[9]LHCb is an experiment set up to explore what happened after the Big Bang that allowed matter to survive and build the Universe we inhabit today.

In Figure 1.2.3 one can see the CMS Experiment.

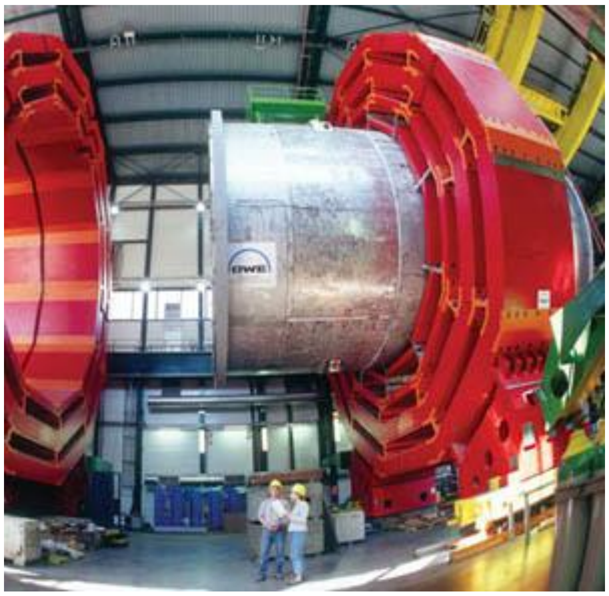


Figure 1.2.3 – CMS Detector.

Each of the LHC Experiments represents an independent research community, within the Particle Physics Community. These communities are highly mobile and dispersed around the world. For these reasons the usage of videoconference and collaborative tools in general is high among these communities in order to allow productive work. Taking this into account CERN management created a IT service dedicated to this activities in 2007.

Figure 1.2.4 – Tutorial given by the CERN VC service.

The main responsibilities of this service are the maintenance of the whole infrastructure and collaborative technologies used on site, the development and deployment of monitoring and automation tools for the service, the required user and operational support lines and, all the tasks for the research and development needed to improve and develop the operational activities.

Since the last two years the activities of the service are increasing, as the LHC reaches is building phase termination. There are more than 60 endpoint terminals installed at CERN, where around 50% are directly managed by the CERN VC Service. There are more than 6000 Desktops managed by CERN IT Department that can become potential videoconferencing/collaborative clients. The majorities of the services used support video and are based on the IP network. There is still a small fraction based on still on PSTN and an emerging usage of VoIP products (Skype and others).

There is an average of 40 remote distributed meetings per day, with more than 80 connections from Europe, Americas and Asia. These numbers are expected to reach exponentially as soon as the LHC is in production.

1.3 Current Operations

The current user operation for videoconferencing systems and collaborative tools in the scientific community is not as easy as one would wish. In some cases, it can become a quite difficult task, namely in the LHC community operation scenario. The LHC community is largely distributed around the Globe and highly mobile between the original local Research Institutes, Universities and CERN. This distribution forces most of the collaborative work to be done remotely, in disperse locations, through different converged networks managed by different entities, using different communication endpoints with different local setups, etc. In Figure 1.3.1 a typical CERN large meeting is shown.

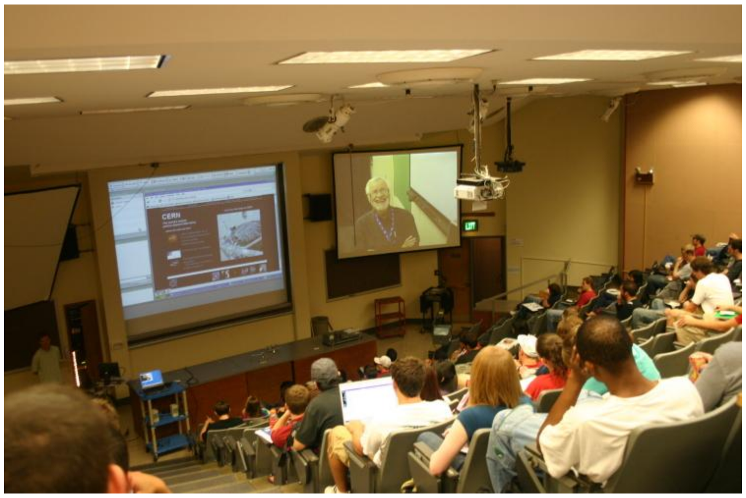


Figure 1.3.1 - Example of a remote talk at CERN (ATLAS Experiment).

Due to all this diversity, the number and the kind of operations needed to be performed is relatively high; Example: let's imagine a collaboration working group part of a CERN Experiment, for instance for ATLAS in the Muon Detector Software. This working group has 10 different universities in the collaboration where 50% of the people are based at CERN as Users of the Lab and the other 50% is based in their local institutes (in Europe, Americas and Asia). This working group needs to hold meetings twice per week. It uses InDiCo system to set the agenda, make a repository of documents, presentations, etc. Since it is not possible for all the members to be based or travelling to CERN for constant and long periods, a remote collaborative system has to be used. First approach: all members to use the same collaboration system over IP (IP to reduce costs); several problems arise: no expertise in their local institutes to provide support; bandwidth availability still differs largely from EU or US and the rest of the world, the quality of local equipments varies from very good to very poor, etc. second approach: 60% of people will be able to use the VC system over IP and the remaining 40% will have to use a normal telephone line. As the meetings to be hold will be always in a multipoint mode (more than two endpoints) with an average of 5, 10 people participating, a typical scenario will be that someone from the group, will have to book a physical room at CERN (where the core group is located), book a teleconference meeting, book a VC IP meeting, distribute the relevant data to connect to a given meeting among the group members and start every meeting at CERN or request assistance to the CERN VC service to start it... even if there is no technical problem a decent fraction of time has been lost only with the session setup.

A second example: at some point two completely different working groups have to collaborate together remotely. Let's imagine the same ATLAS Muon Detector Software group that now needs to work with ATLAS Inner Detector Software Group. Even if in the same Experiment (ATLAS), the universities and countries that participate in each working group are rather different. Being different and both disperse, these two groups use the same tool for setting the agenda of production meetings, make the repository of documents, etc. This tool is InDiCo. In what concerns the collaborative systems the two groups have completely different philosophies: the first one uses mainly phone as the locations of the Universities and Group members are mainly in Europe and the cost of calls is not so high; the second one uses VC over IP or VoIP since some of the locations are in Asia and South America and costs of phone calls are rather significant.

So in this case, to avoid changing the user normal procedures to collaborate, someone would have to book a physical room at CERN (where the core groups are located), book a teleconference meeting, book a VC IP meeting, distribute the relevant data to connect to a given meeting among the group members, ask CERN service to bridge the 2 systems, and start every meeting at CERN or request assistance to the CERN VC service to start it.

As one can easily see, in both examples there are several operations needed to be done that quickly can become critical points of failure and give origin to significant delays and relevant fractions of time lost that affect the core business of each working group (being pure physics research and not technology operational tasks or problems).

This overall scenario and challenging problematic described above in these examples describes exactly the problem this project aims to address and solve.

Chapter 2

State of Art

In this section, one intends to describe the current state of art not only in terms of systems and protocols but also in terms of current user and system operations. This state of art description aims to better indentify the problem to address and to justify the chosen approach (described later in the document).

2.1 InDiCo Framework

The InDiCo project was started in 2002 as a European collaboration between CERN, the University of Amsterdam (Netherlands), the International School for Advanced Studies of Trieste (Italy), the University of Udine (Italy) and the Netherlands Organization for Applied Scientific Research. Two years later, this project was stopped but CERN has decided to continue to develop the part related to conference management. InDiCo stands for INtegrated DIgital Conference (Fig. 2.1.1 depicts the InDiCo logo).

Figure 2.1.1- InDiCo Logo.

InDiCo has been used for the first time at CHEP'04[10] conference. It generally consists in a conference management tool that allows scheduling events (event oriented), from simple talks to complex conferences with several sessions and contributions. InDiCo main development platform is Python[11]. It runs on an Apache Web application server using the Python module (mod_python). It uses

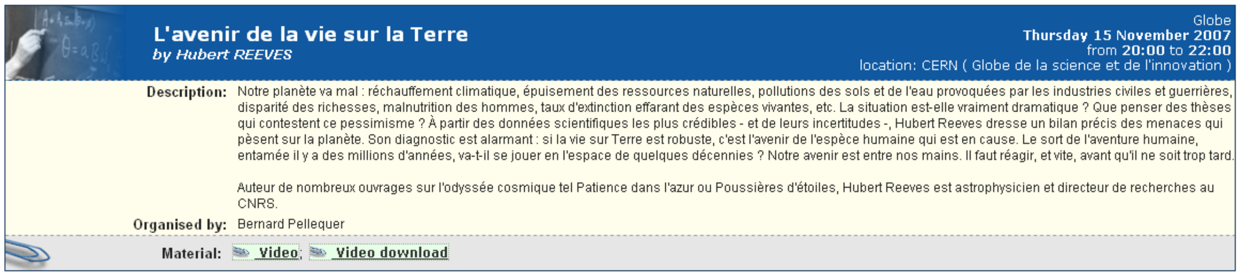
[10] CHEP stands for Computing in High Energy and Nuclear Physics. It's an International Conference
[11] http://www.python.org

an object oriented database implemented in Python, the Zope Object Database (ZODB) for storing conferences metadata. The submitted files and archives are directly stored on the server's file system. It uses XML and XSLT for timetable generation. The production version of InDiCo is currently running on a Linux platform, but it has been also tested in a Windows server. InDiCo is distributed under the GNU General Public License and is fully open source.

InDiCo is an event oriented system. There are 3 types of events in InDiCo. These events vary on their level of complexity. They are described as follows:

- Lectures:

The lecture (showed in Fig. 2.1.2) is supposed to be a simple event basically with one speaker and an audience. In this kind of event, the organizer is allowed to set who is the speaker and to create a list of attendees. The speakers can upload material for their lecture e. g. slides, other documents or the lecture video (if available).



L'avenir de la vie sur la Terre
by Hubert REEVES

Globe
Thursday 15 November 2007
from **20:00** to **22:00**
location: CERN (Globe de la science et de l'innovation)

Description: Notre planète va mal : réchauffement climatique, épuisement des ressources naturelles, pollutions des sols et de l'eau provoquées par les industries civiles et guerrières, disparité des richesses, malnutrition des hommes, taux d'extinction effarant des espèces vivantes, etc. La situation est-elle vraiment dramatique ? Que penser des thèses qui contestent ce pessimisme ? À partir des données scientifiques les plus crédibles - et de leurs incertitudes -, Hubert Reeves dresse un bilan précis des menaces qui pèsent sur la planète. Son diagnostic est alarmant : si la vie sur Terre est robuste, c'est l'avenir de l'espèce humaine qui est en cause. Le sort de l'aventure humaine, entamée il y a des millions d'années, va-t-il se jouer en l'espace de quelques décennies ? Notre avenir est entre nos mains. Il faut réagir, et vite, avant qu'il ne soit trop tard.

Auteur de nombreux ouvrages sur l'odyssée cosmique tel Patience dans l'azur ou Poussières d'étoiles, Hubert Reeves est astrophysicien et directeur de recherches au CNRS.

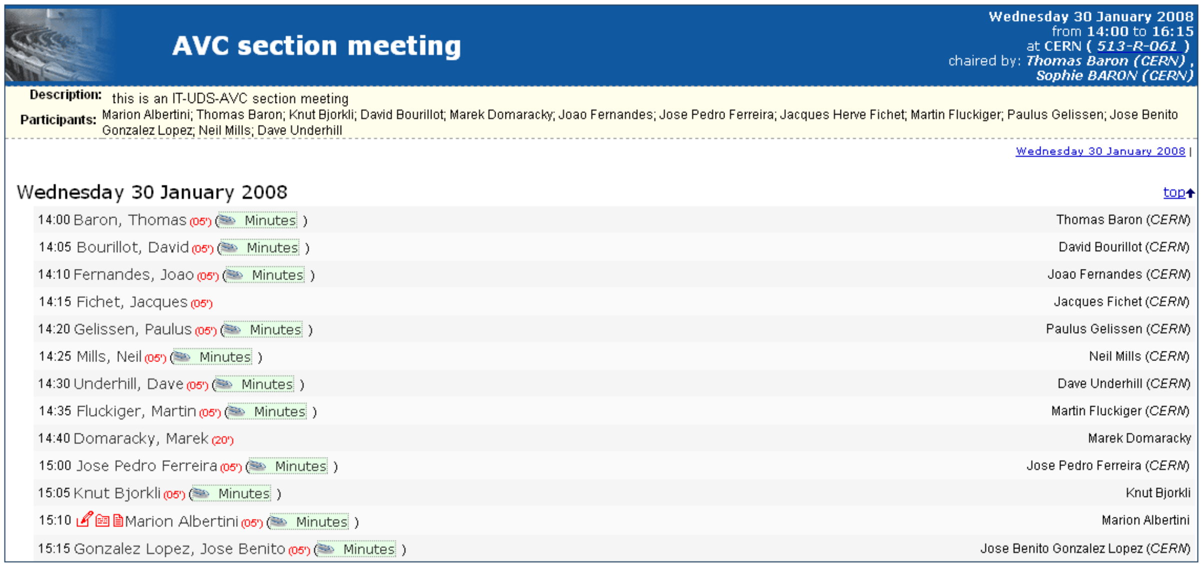
Organised by: Bernard Pellequer

Material: Video; Video download

Figure 2.1.2 - Example of a Lecture event managed by InDiCo.

- Meetings:

In Figure 2.1.3 one can see an example of a meeting. A meeting is an event with several participants that can be also speakers, like for example a working group round table meeting. In this event the organizer can create a timetable, which can include sessions. Participants can also be added to the event. It is also possible to book a physical room for the meeting. Participants of a meeting are authorized to upload some material (documents or minutes about the oral intervention).



AVC section meeting

Wednesday 30 January 2008
from 14:00 to 16:15
at CERN (513-R-061)
chaired by: *Thomas Baron (CERN)* , *Sophie BARON (CERN)*

Description: this is an IT-UDS-AVC section meeting
Participants: Marion Albertini; Thomas Baron; Knut Bjorkli; David Bourillot; Marek Domaracky; Joao Fernandes; Jose Pedro Ferreira; Jacques Herve Fichet; Martin Fluckiger; Paulus Gelissen; Jose Benito Gonzalez Lopez; Neil Mills; Dave Underhill

Wednesday 30 January 2008 |

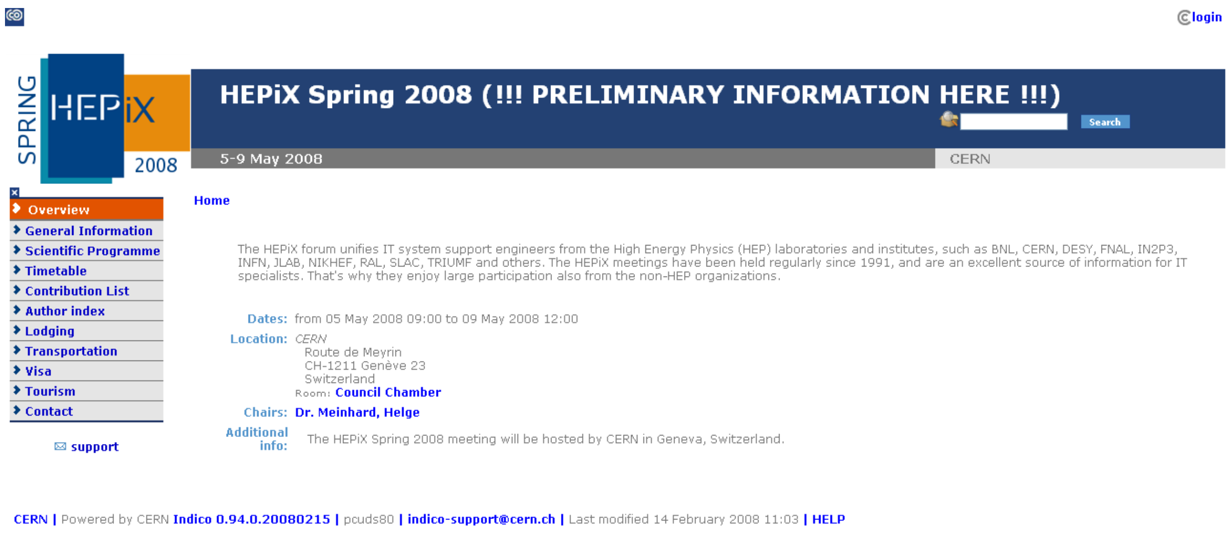
Wednesday 30 January 2008 top

14:00 Baron, Thomas (05') (Minutes)	Thomas Baron (*CERN*)
14:05 Bourillot, David (05') (Minutes)	David Bourillot (*CERN*)
14:10 Fernandes, Joao (05') (Minutes)	Joao Fernandes (*CERN*)
14:15 Fichet, Jacques (05')	Jacques Fichet (*CERN*)
14:20 Gelissen, Paulus (05') (Minutes)	Paulus Gelissen (*CERN*)
14:25 Mills, Neil (05') (Minutes)	Neil Mills (*CERN*)
14:30 Underhill, Dave (05') (Minutes)	Dave Underhill (*CERN*)
14:35 Fluckiger, Martin (05') (Minutes)	Martin Fluckiger (*CERN*)
14:40 Domaracky, Marek (20')	Marek Domaracky
15:00 Jose Pedro Ferreira (05') (Minutes)	Jose Pedro Ferreira (*CERN*)
15:05 Knut Bjorkli (05') (Minutes)	Knut Bjorkli
15:10 Marion Albertini (05') (Minutes)	Marion Albertini
15:15 Gonzalez Lopez, Jose Benito (05') (Minutes)	Jose Benito Gonzalez Lopez (*CERN*)

Figure 2.1.3 - Example of a Meeting Event managed by InDiCo.

- Conferences:

This is the most complex type of event. It is shown in Fig. 2.1.4. It is intended to be used when one has to (scientifically) organize a conference. This can include submission of contributions, session creation, complex timetables, webpage for the event and other related tasks. It also allows attendees to register and pay online. The scheduled speakers can submit their contributions and upload material about their talk.



HEPiX Spring 2008 (!!! PRELIMINARY INFORMATION HERE !!!)

5-9 May 2008 CERN

The HEPiX forum unifies IT system support engineers from the High Energy Physics (HEP) laboratories and institutes, such as BNL, CERN, DESY, FNAL, IN2P3, INFN, JLAB, NIKHEF, RAL, SLAC, TRIUMF and others. The HEPiX meetings have been held regularly since 1991, and are an excellent source of information for IT specialists. That's why they enjoy large participation also from the non-HEP organizations.

Dates: from 05 May 2008 09:00 to 09 May 2008 12:00
Location: *CERN*
Route de Meyrin
CH-1211 Genève 23
Switzerland
Room: **Council Chamber**
Chairs: **Dr. Meinhard, Helge**
Additional info: The HEPiX Spring 2008 meeting will be hosted by CERN in Geneva, Switzerland.

CERN | Powered by CERN Indico 0.94.0.20080215 | pcuds80 | indico-support@cern.ch | Last modified 14 February 2008 11:03 | HELP

Figure 2.1.4 - Example of a Conference Event managed by InDiCo.

In InDiCo, users are also divided in types, meaning there are different roles in which the users are categorized. These roles define the level of rights in a given event.

The following roles are defined:

- Chairman:

This is (as the name clearly indicates) the responsible for an event.

The Chairman is the event creator and decides which users can become event managers. The Chairman defines all the event settings (time, place, etc.) and can enable other options like call for abstracts or online payment.

- Manager:

The manager is allowed to modify the settings of the event. The number of rights can vary depending on what the chairman decides.

- Participant:

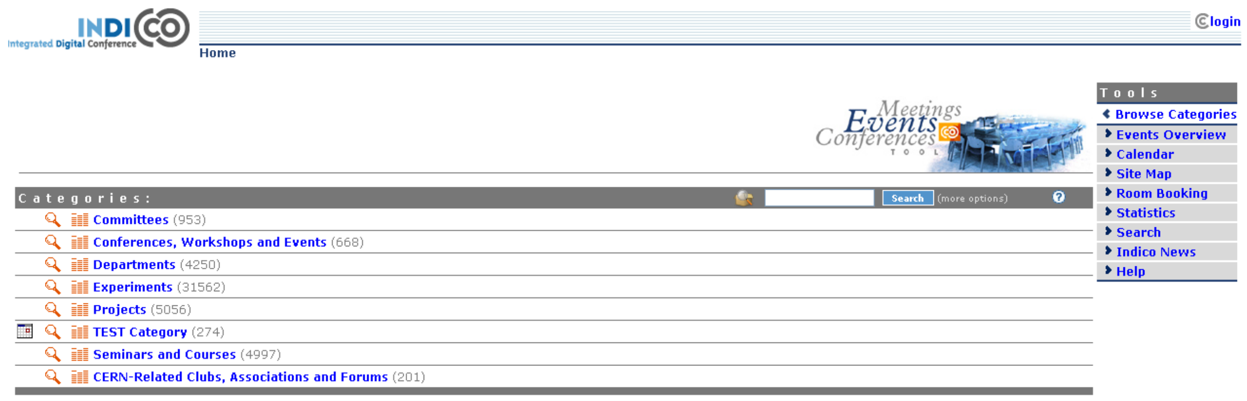
A Participant is a user that attends to an event.

2.2 InDiCo Features

Since 2002, InDiCo is a very dynamic project where new features are constantly being developed and added in production. This is the case for instance of the multilingual interface and time zone awareness functionalities that are being added into production during the time this document was written. Some current features are listed below as follows:

- Tree-like structure:

As shown in Fig. 2.2.1, InDiCo events are always organized in categories: as an example, in a case like CERN one has categories like ‘Experiments’, 'Projects', 'Departments’, etc. This classification is also useful to give a picture of the institution organization. Each category can then contain either sub-categories or events.



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

Categories: Search (more options)

Committees (953)
Conferences, Workshops and Events (668)
Departments (4250)
Experiments (31562)
Projects (5056)
TEST Category (274)
Seminars and Courses (4997)
CERN-Related Clubs, Associations and Forums (201)

Figure 2.2.1 – CERN InDiCo Homepage displaying different categories.

- Call for abstracts:

The creator of an event can activate the call for abstracts. Speakers can then submit the abstract of their talk for the conference.

- Material submission:

Users can upload files (being multimedia, presentations, etc.) in the events.

- Automatic web page creation for the events:

When an event is created in a webpage is also automatically created. Then, the creator of the event can customize it.

- Customizable event evaluation surveys:

The manager of an event can perform a survey to assess the satisfaction of participants.

- Automatic notifications:

Participants can for example, receive an automatic email notification by email before the event. The creator of a conference will receive a notification when an abstract has been submitted, etc.

- Registration management:

Registrations in an event can be done online (given that the creator of this event allows it).

- Online payment:

An online payment module is available for conference creators who want to allow the payment of fees online (participation to the event, to the social events associated, etc.).

- Room booking system:

InDiCo provides a module to manage room bookings, including time, conflicts solving, pre-bookings, etc.

- Basic integrated support for equipment in a given room:

When the creator or manager of an event books a videoconferencing room, there is the possibility to indicate an available videoconference system to use and also send a notification to ask for assistance in operating it.

- Exportation of information in several formats as PDF, XML, RSS feeds, iCal, etc.:

One has the possibility of exporting the timetable of a conference as an iCal. XML is used (among others) for the RSS feeds available in each category for awareness of the new events that appear in this category. It's also used to export information about various events (for example to display the list of events or its description on their own webpage).

- Search Tool (through CDS[12])

2.3 InDiCo Architecture and Structure

InDiCo is strongly based on the object-oriented paradigm. The main development platform is Python. Python is a dynamic object-oriented programming language, supporting multiple programming paradigms (functional programming, object oriented programming and imperative programming). It has an automatic garbage collector and a good exceptions management mechanism. It's very similar to Perl or Smalltalk languages. The Python syntax is simple and minimalist providing a large standard and extensive library. Python has been chosen for InDiCo for two main reasons: first, due to its generally good performances and second because new release versions of the language have a minimal impact on the project.

The InDiCo database, ZODB, is object-oriented which is fully developed in Python, what allows a very good integration with the development programming language. It also deals with persistent objects.

InDiCo is running on apache server platform using the mod python module (integrating Python into the HTTP server).

InDiCo has been designed following three-layer architecture, as depicted in Figure 2.3.1. The architecture allows further extension of the system or even the redesign of one of the layers without affecting the remaining ones.

[12] CERN Document Server (CDS, http://cdsweb.cern.ch/) is a repository that holds over 800,000 bibliographic records, including 360,000 full text documents for the High Energy Physics domain and related areas.

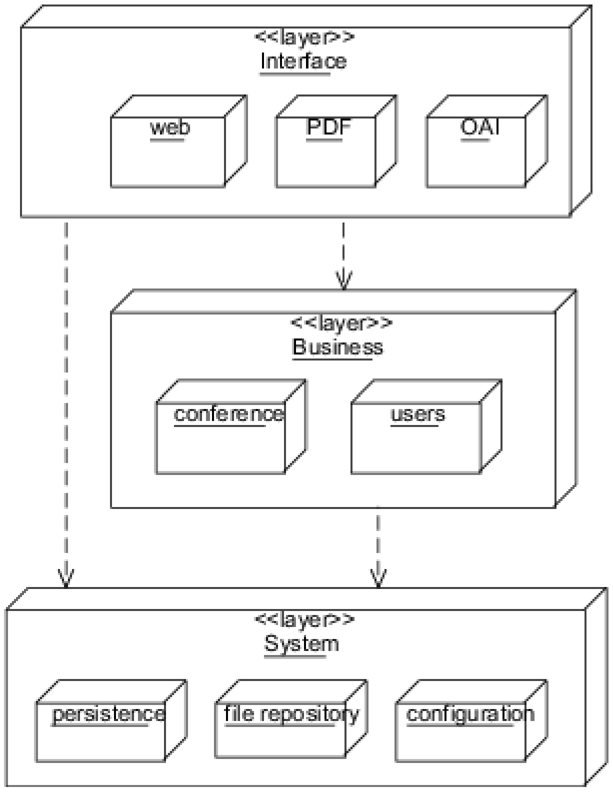


Figure 2.3.1 - InDiCo Three-layer Architecture.

The interface layer provides the communication with the outside world: the Web for the users; the OAI interfaces (Open Archive Initiative) to communicate with other applications, etc. The business layer contains the core classes, used by the application logic. The system layer is responsible for data storage. It contains the ZODB database and the file and archives repository.

2.4 Similar Frameworks

There are many event management systems available in the world, but not so many with such a rich set of features as InDiCo presents. However, one that can be pointed out is Eventseer[13]. It's an event oriented system, much like InDiCo, but far from being so powerful in terms of conference lifecycle integration (it seems to contain only descriptive data about the events) and number of features.

However, it has some concepts that InDiCo could adopt like namely a list of upcoming events or the capability of associating events with institutions.

These particular features can solve some current problems of data consistency in InDiCo: for instance, users have an "affiliation" field, but are not explicitly linked to institutions; events have a "location" field, but are not associated with a particular institution (or geographical place). This

[13] http://eventseer.net/

situation breaks the effort to index events and people by geographical location, and to generate statistics about it.

The InDiCo user interface complexity is another issue that needs attention. This is currently being addressed during the time this document has been written. The analysis of these aspects is not in the scope of this document.

One can also find videoconferencing network management systems, provided by the main videoconference vendors, but in these cases the concept is rather different. These frameworks, even if provide meeting scheduling functionalities and comprehensive management solutions for voice and video collaborative communications they are tailored for system administrators, as they provide tools to efficiently manage and monitor the video network infrastructure, to ensure efficient bandwidth utilization, device management and control, etc. These tools, for the obvious reasons described, don't cover the concept that this project is aimed to achieve since the base idea is user oriented in order to provide a user interface which decreases exponentially the number and kind of operational tasks needed.

2.5 Collaborative Protocols

2.5.1 H.323 protocol

H.323[14] is associated with complete hardware and software solutions providing high quality videoconferencing services at a corresponding price.

H.323 consists in a communication standard produced by the ITU, specifying a great deal of information about the properties and components that interact within the environment. It specifies the pieces that combine to provide a complete communication service:

- Terminal Endpoints: these are either PC software based or hardware stand alone devices.

They are the endpoints of the communication lines (ex. Polycom or Tandberg endpoints).

- Gatekeepers: these are the brains of the network. They provide services like addressing, identification, authorization, and bandwidth management.
- Gateways: these serve as translators when connecting to a dissimilar environment (such as a PSTN network, for example).
- MCU's (Multipoint Control Units): these allow multipoint conferencing, or communication between more than two parties at once (like a traditional conference call on a telephone).

[14] H.323 has been initiated in late 1996, and aimed at the emerging area of multimedia communication over LAN's (local area networks). It is an outgrowth of the traditional H.320 technology but optimized instead for the Internet.

H.323 also describes protocol standards, permissible audio and video codecs, RAS (registration, admission, and status), call and control signaling and specifies a mandatory level of compliance and support for the above specifications for all terminals on the network. Examples of H.323 based video conferencing service used within CERN LHC community are HERMES[15] and ECS (ESnet Collaborative Service) [16]. HERMES service, based in Europe, currently relies on a Codian 40-port MCU (Multipoint Control Unit), similar to the ESnet ECS 88 service (described ahead). It enables high-quality conferences that can be managed with great flexibility. ECS service, based in the US, relies on a Gatekeeper and 2 Codian MCUs, also providing a phone bridge. It currently offers two different quality level services:

- 85 service: Enhanced Video Quality, Full screen display, H.263, 30 fps, 384 kbps and above
- 88 service: Personal selection of display, highest quality, any bit rate, QuickTime and REAL stream viewing.

Modern H.323 devices, being endpoints, MCUs or Gateways, usually support an external API system based on XML or XML-RPC protocol (described later in this document), allowing a set of operations to be done by third-party systems. These APIs are very interesting especially for institutions that have already a founded IT infrastructure. They allow the addition of the functionality of the corresponding systems to the existing infrastructure decreasing the several implications that this could have for operations, user interfaces, etc.

2.5.2 SIP protocol

The Session Initiation Protocol (SIP) is a signaling protocol used for establishing sessions in an IP network. A session can be a simple point-to-point telephone call or it could be a collaborative multimedia conference session. The ability to establish these sessions means that a host of innovative services become possible, such as voice-enriched e-commerce, web page click-to-dial, Instant Messaging with buddy lists, and IP based services.

Over the last couple of years, the Voice over IP community has adopted SIP as its protocol of choice for signaling. SIP is an RFC standard (RFC 3261[17]) from the Internet Engineering Task Force[18] (IETF), the body responsible for administering and developing the mechanisms that comprise the Internet. SIP is still evolving and being extended as technology matures and SIP products are “socialized” in the marketplace.

The IETF's philosophy is one of simplicity: specify only what you need to specify. SIP is very much of this mould; having been developed purely as a mechanism to establish sessions, it does not know about

[15] HERMES is operated by CERN-IT-UDS together with IN2P3, CNRS and INSERM.

[16] Energy Sciences Network service (ECS ESnet) is based in Berkeley and funded by the United States Department of Energy Office of Science

[17] http://www.ietf.org/rfc/rfc3261.txt?number=3261

[18] http://www.ietf.org/

the details of a session, it just initiates, terminates and modifies sessions. This simplicity means that SIP scales, it is extensible, and it sits comfortably in different architectures and deployment scenarios.

SIP is a request-response protocol that closely resembles two other Internet protocols, HTTP and SMTP (the protocols that power the World Wide Web and email); consequently, SIP sits comfortably alongside Internet applications. Using SIP, telephony becomes another web application and integrates easily into other Internet services. SIP is a simple toolkit that service providers can use to build converged voice and multimedia services. Modern videoconferencing endpoint devices that rely on H.323 protocol are increasingly supporting also SIP. The major disadvantage of SIP is the fact of being such a wide standard, highly flexible. It still leads in some cases to interoperability problems between the different vendors' implementations. This protocol even if referred in this document is not being directly used in this project for the use cases described, since all the devices that support SIP support also H.323. The implementation using H.323 is thought to be more robust and doesn't have the disadvantages of SIP mentioned before.

2.5.3 XML- RPC protocol

XML-RPC protocol consists in a set of implementations that allow software running on disparate operating systems and in different environments to make procedure calls over the Internet. It's a remote procedure calling scheme using HTTP as transport and XML as the encoding. XML-RPC is designed to be as simple as possible (the spec is about two pieces of paper), while allowing complex data structures to be transmitted processed and returned.

Modern collaborative services (H.323/SIP based systems, PABXs, etc.) provide an external API that supports external requests normally based on XML-RPC, what turns easier any task of integration of these devices with third-party systems allowing for example, to get the system status, connect a client to a specific meeting, etc.

In particular for Python language (as the main InDICo development platform) the XML-RPC implementation is available in the module *xmlrpclib*, since Python version 2.2. This module provides a standard set of classes and methods that allow XML-RPC clients and servers to be written. For the clients, in order to invoke a remote XML-RPC object, one creates a proxy object which forwards requests to the server using XML-RPC. The proxy object looks and feels like a regular Python object, so requests are simple function calls. It is also possible to implement XML-RPC servers. The most straightforward way is to write an instance with methods that do what one needs, and then register this instance. An example of a XML-RPC client is shown below:

```
import xmlrpclib

#Port 80 is the default
server = xmlrpclib.ServerProxy("http://time.somewhere.com")

TimeObj = server.currentTime
```

```
currtime = TimeObj.getCurrentTime()

print currtime
print currtime.value
```

Python also comes with SimpleXMLRPCServer, a module for implementing XML-RPC servers. You can register functions or instances with an instance of the SimpleXMLRPCServer class in the module of the same name, in order to expose XML-RPC services.

2.6 Collaborative Systems

2.6.1 EVO system

EVO (Enabling Virtual Organizations) is a software based videoconferencing system developed by Caltech[19], which aims to provide to the High Energy and Nuclear Physics experiments a service that meets the requirements of usability, quality, scalability, reliability, and cost necessary for nationally and globally distributed research organizations such as the High Energy Physics community. The EVO system based on a new distributed architecture, leverages on the 10+ years of experience of developing and operating the distributed production of its predecessor VRVS (Virtual Rooms Videoconferencing System). EVO has been officially released during June 2007 and includes an integrated user interface, a rich feature set, H.323 protocol integration, support and adaptability to all operating systems and an improved overall level of operational efficiency and robustness.

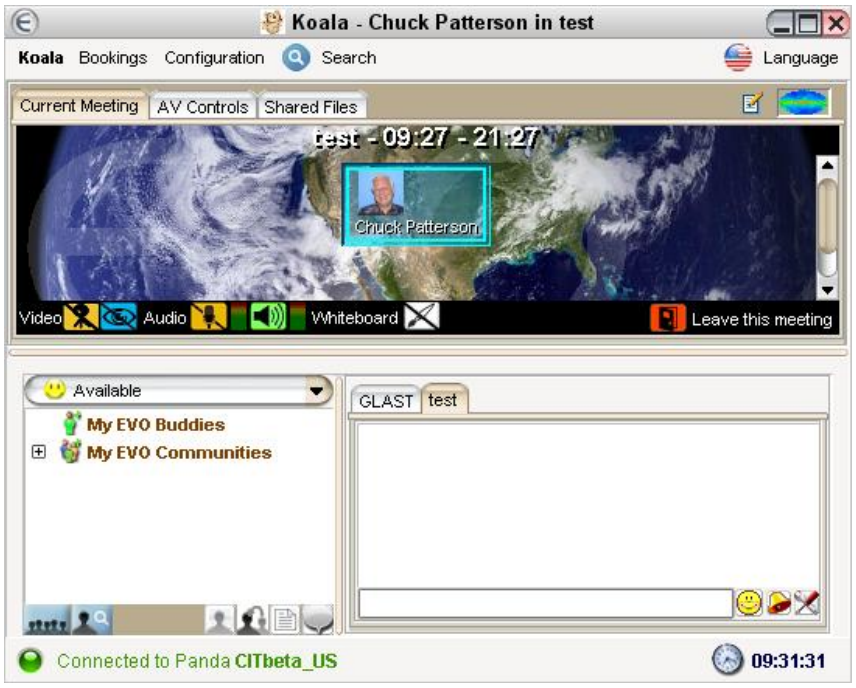


Figure 2.6.1 - EVO End User Client (Koala).

[19] http://www.caltech.edu

The Figure 2.6.1 shows the EVO JAVA client. The system is pure Java and it consists, among others, in two main components: a Java Web Start application client (that can be called remotely) supported in all the platforms and a backbone of Linux servers, responsible for audio, video and data intelligent streaming.

Its video application (Fig. 2.6.2) supports H.263 as video protocol and for audio it uses G.711 16 kHz. EVO provides a Java API for third-party system requests namely place and delete bookings, check the booking status or start meetings from an external application.

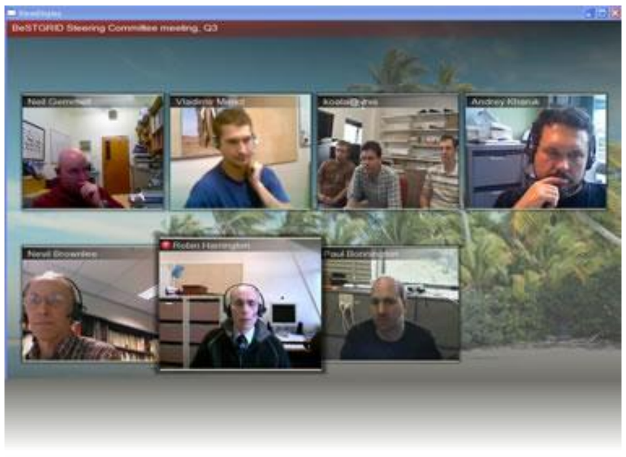
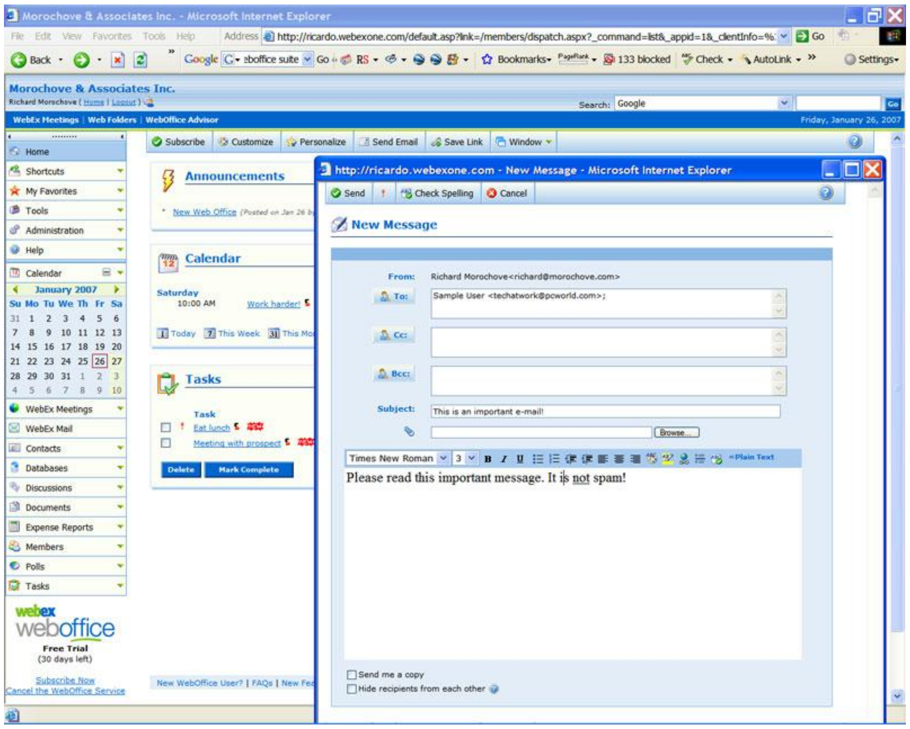


Figure 2.6.2 - EVO Video Application.

2.6.2 WebEx system

WebEx system is a commercial collaborative system (provided by Cisco), that provides on-demand collaboration, online meeting, web conferencing and video conferencing services. The current desktop client is shown in Figure 2.6.3. It’s a fully hosted application and generally consists in an ActiveX/Java-based web client supported in all the platforms connected to a set of meeting centers across the world (connection that is transparent to the user).

It started as a pure web conferencing application (desktop sharing only) but, as it became popular, other features have been added: in a first step pure telephony, then VoIP support and in the last years, MPEG-4 video. Ongoing developments for the integration of SIP protocol are underway. It provides a set of APIs for system integration, namely a URL-based API, a XML API and a TSP API. The URL-based API commands are powerful yet easy to implement. Depending on the level of requirements for integration, implementation can take place quickly by implementing these API commands. The URL-based API is appropriately named, as it is fundamentally based in API calls embedded in URLs or links and as such requires the use of a browser. These links can send (or "push") information to WebEx, e.g. creating a user, creating a meeting, updating a meeting, deleting a meeting, etc. These links cannot be used to retrieve information from WebEx like a list of users, or a list of meetings for a particular date. This type of "pulling" of information can be accomplished using the XML API. The XML API commands are also very powerful but more is required to implement them. The XML API, though it can also use the links method, is not limited to use with a browser. This API is fully portable to many different implementation types, including to an EXE (executable). The XML API links can not only push information to WebEx, as mentioned before, they can also pull information, e.g. generate a list of users in order to check before creating a new one, retrieving a list of meetings scheduled for a specific time frame so that perhaps the hosts may be notified with service affecting information.


Morochove & Associates Inc. - Microsoft Internet Explorer
http://ricardo.webexone.com/default.aspx?link=/members/dispatch.aspx?_command=list&_appid=1&_clientInfo=%
Morochove & Associates Inc.
WebEx Meetings | Web Folders | WebOffice Advisor
Friday, January 26, 2007
Home
Shortcuts
My Favorites
Tools
Administration
Help
Calendar
January 2007
WebEx Meetings
WebEx Mail
Contacts
Databases
Discussions
Documents
Expense Reports
Members
Polls
Tasks
webex weboffice
Free Trial (30 days left)
Subscribe Now
Cancel the WebOffice Service
Announcements
Calendar
Tasks
http://ricardo.webexone.com - New Message - Microsoft Internet Explorer
Send
Check Spelling
Cancel
New Message

From: Richard Morochove<richard@morochove.com>

To: Sample User <techatwork@pcworld.com>;

Cc:

Bcc:

Subject: This is an important e-mail!

Please read this important message. It is not spam!

Send me a copy

Hide recipients from each other

Figure 2.6.3 – WebEx Web Client.

The TSP API is the newest WebEx API offering and is specifically targeted at the Telco partners. With this new API, Telcos are able them to integrate their telephony with the WebEx meeting features allowing their customers to maintain their custom telephony PIDs, PINs, etc. or yet still access the WebEx service.

Chapter 3

Analysis

This section intends to give some details about the analysis that led to the approach to solve the problem. It is followed by the description of some of the use cases identified to be implemented. The main requirements defined for this module are also formalized and presented. Finally, it is shown a context of usage, from the user point of view, to better illustrate the overall functionality of the system.

3.1 Approach

In order to address the complexity of situations as the ones described in 2.6, there is the need of analyzing the current situation of the collaborative systems currently in the market, its tendencies and directions. Analyzing different literature provided by Wainhouse Research[20] and also current common operation experience, there is a strong belief currently in this domain that will be very difficult to converge to one single collaborative system worldwide both in Industry and Research, as future trends should be to have a multitude of solutions. This is due to a variety of reasons, such as: multiplicity of available systems, multiplicity of entities managing different converged networks, multiplicity of IT policies within the institutes and companies, historic reasons, etc.. These aspects lead to different setups and procedures. One also notes that to reduce the complexity of operation and system management, the provision of a customization and integration layer is needed, as depicted in Figure 3.1.1. One way of providing this layer is by in-house development.

[20] Wainhouse Research is a consulting firm that analyzes the market trends, technologies/products, vendors, applications, and related services in the real-time unified communications domain.

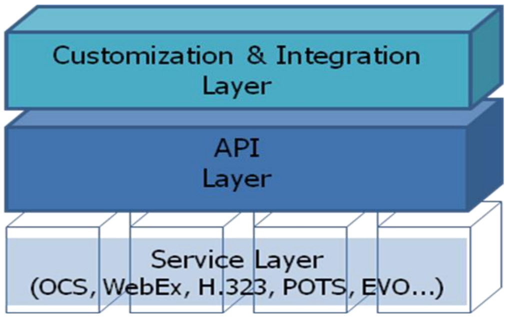

Customization & Integration Layer
API Layer
Service Layer (OCS, WebEx, H.323, POTS, EVO...)

Figure 3.1.1 - Customization and Integration Layer.

This layer should address different issues: to reduce the complexity and the number the user operations to start a multisystem collaboration session, to automate and gather all the relevant information, to create a level of customization where IT administrators can easily find information to manage or debug the collaborative systems, or technical problems in collaborative sessions, etc. However, at the same time, this layer needs to provide architecture as flexible as possible, since in a domain where technology is rapidly evolving and changing, the possibility of adding or removing modular functionalities and modify or add support of the underlying collaborative systems in an easy way, providing a high scalability, is, therefore, critical.

By analyzing the market, one can easily observe that the system vendors are aware that the third-party integration of their systems is important. The proof is that the great majority of systems provide Integration APIs[21] to allow the easy integration of the system in an existing IT infrastructure within a Organization. For all these reasons the approach chosen for this project relies on building a Layer on top of a system API Layer. This layer should be as much integrated with the current infrastructure as it can be, for obvious reasons, among them: gets easily architected (as there is up to some extent a knowledge reuse); the user interface can be more familiar to the users as there is an enforced standardization, the long-term maintenance of the system can be simplified as normally the infrastructure is already providing it, etc. If one analyses the current operations (see 1.3), one can easily conclude that in this case, InDiCo is a very strong and natural candidate to provide this layer, since it can be seen in some way an extension of the activities and event management functionalities that it already provides. Based on the essence of InDiCo, this construction will also allow a modular architecture, fulfilling a critical requirement, as one API implementation will not affect the functionality of others. In the mid-term requirements may have to change and the support of new or additional systems might be needed (e. g. Microsoft OCS[22]).

[21] An application programming interface (API) is a set of declarations of the functions (or procedures) that an operating system, library or service provides to support requests made by external software.

[22] Microsoft Office Communications Server (OCS), is an enterprise real-time communications server, providing the unified infrastructure to allow instant messaging, presence, audio and videoconferencing and web conferencing functionalities.

The vast majority of systems offer APIs based in XML, in particular using XML-RPC protocol (see 2.5.3).

3.2 Use Cases

In this section, some example use cases are presented which are intended to be supported by the module to be specified. One should note that with a scalable API approach many other use cases could have been presented. The ones present were chosen based on the current practical needs of priority and operational and user support.

As an initial approach, we will consider only the InDiCo meeting events, as these are the ones that need more often the use of collaborative tools and technologies. After a first implementation version of the module, possibly followed by a period of optimization, the same approach can be extended to lecture and conference events.

When defining use cases, one needs to define actors. As a matter of fact, the actor role is already defined: the InDiCo event Chairman. As mentioned before (see 2.1), he is the responsible for the event so the operation of the module will rely on him. Later on, one can also decide to define another actor that would be the event Manager. This would imply that the Chairman would give the right to operate the module (as this affects the settings of the event).

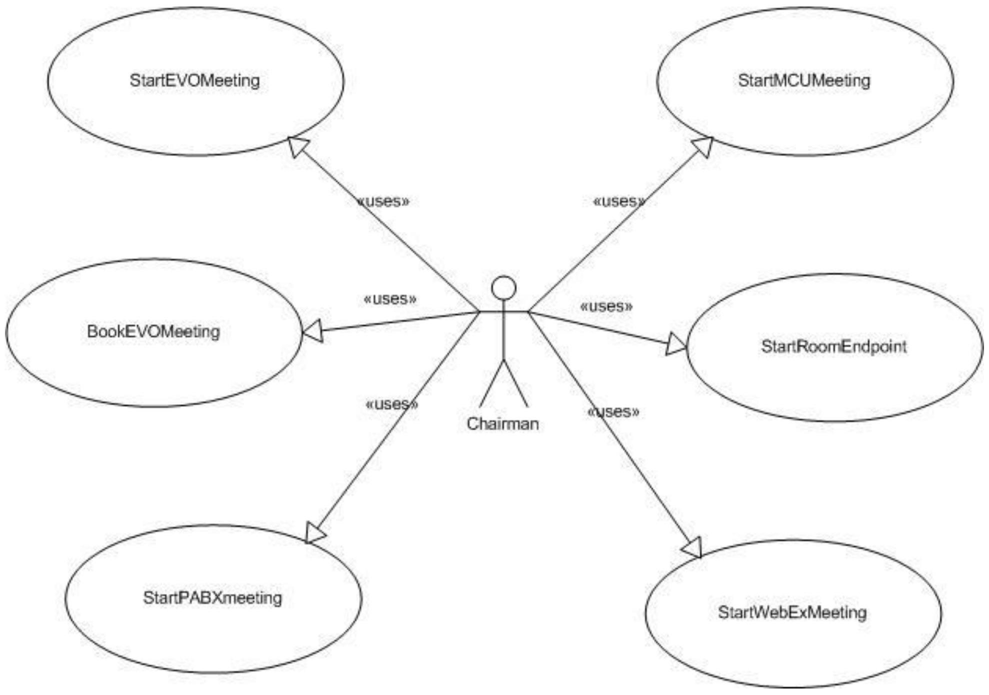


Figure 3.2.1 – Diagram with some of the use cases.

The Figure 3.2.1 depicts a general use case diagram with the interaction between use cases and actor. One should note that this is a sample set of use cases. Other could be added. In the following sections, some use cases are also analyzed. Each one of them describes the interactions between the actor and the system, giving some details by describing the sequence of actions. The sequence diagrams are a graphical description of the use cases, describing the actions in the order in which they will occur. The arrows represent the messages exchanged between the actors represented on top of the diagrams.

3.2.1 H.323/SIP Systems

The use cases described in this section are significantly important, because of the priority of implementation, as they cover the industry de facto standards for videoconferencing. One presents two examples to illustrate these use cases. First: a Chairman wants to start/join a collaborative meeting using a H.323 room endpoint. When it registers his meeting in InDiCo, he books also a videoconferencing room for the purpose. InDiCo will obtain the information of which terminal endpoint the user wants to connect at the time of the meeting. This information is provided by the user or it is already in the system. InDiCo will then present the user with a software connection button.

- Use case: Start Room Endpoint (Figure 3.2.2)
- Preconditions: The Chairman must be logged in the system. The Chairman already selected the meeting. The Chairman is already browsing the webpage of the Administrative Area of the meeting, Remote Collaboration tab, “VC Room Endpoint” selected. The Chairman already indicated the VC endpoint IP address (by having booked a VC room or indicating it in the IP address field)
- Normal Flow: 1. The Chairman confirms the action. 2. The system saves the choice and displays the current status. 3. Use case ends.
- Alternate Flow: If the Chairman doesn’t confirm his action use case ends.

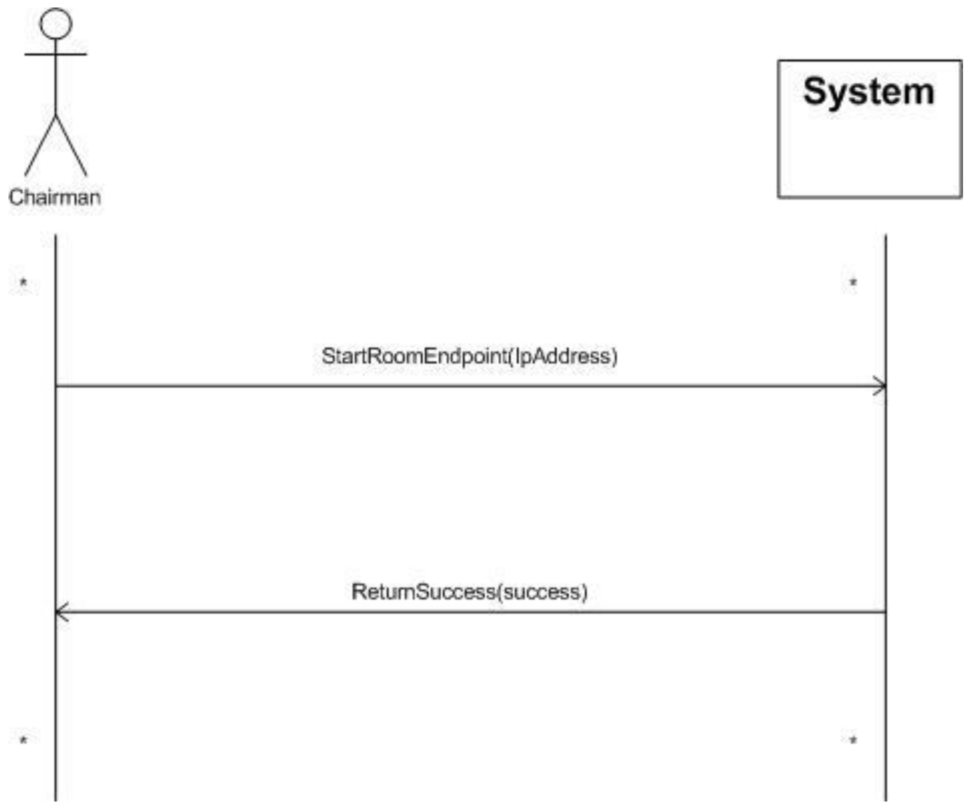


Figure 3.2.2 – Sequence diagram: Starting a Room Endpoint.

Another example: setting up a collaborative session in an ad-hoc MCU. In this case, the Chairman of the event will be able to specify an ad-hoc code for a MCU to be used. When the time of the meeting comes, the Chairman will be able to start manually the meeting by pushing one software button (later on, one can forecast that the session can also start automatically). If the chairman specified the IP addresses of the remaining remote participants, they will also be automatically connected to the meeting.

- Use case: Start MCU Meeting (Figure 3.2.3)
- Preconditions: The Chairman must be logged in. The Chairman already selected the meeting.

The Chairman is already browsing the webpage of the Administrative Area of the meeting, Remote Collaboration tab, “MCU System” selected. The Chairman already indicated the MCU IP address from a list (if not using the one provided by default), VC endpoint IP address (by having booked a VC room or indicating it in the IP address field) and alternatively, the remote participants IP addresses.

- Normal Flow: 1. The Chairman confirms the action. 2. The system saves the choice and displays the current status. 3. Use case ends.
- Alternate Flow: If the Chairman doesn’t confirm his action use case ends.

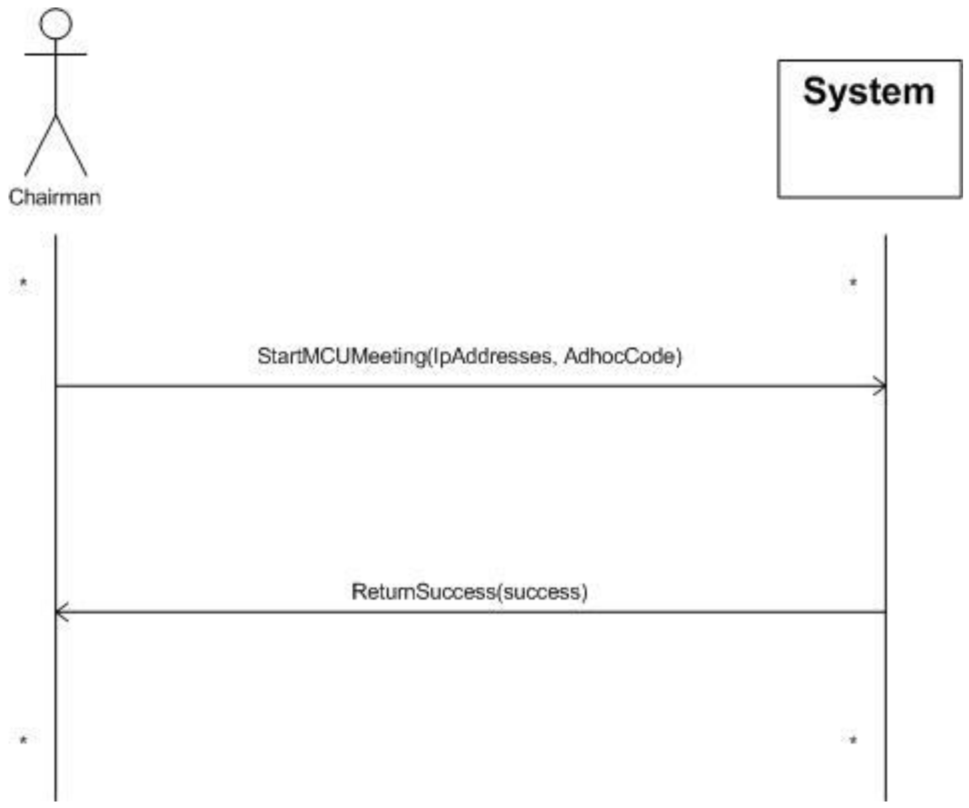


Figure 3.2.3 - Sequence diagram: Starting a MCU meeting.

3.2.2 EVO System

For this use case, the user will use EVO to hold his distributed remote meeting. In this case, the Chairman in order to start the meeting has to combine both uses cases of “Book EVO Meeting” and “Start EVO Meeting” (one can use only the second one, provided that the booking was already done). Example: After the meeting being created in InDiCo, the Chairman will book an EVO meeting and be able to start it through InDiCo.

- Use case: Start EVO Meeting (Figure 3.2.4)
- Preconditions: The Chairman must be logged in. The Chairman already selected the meeting.

The Chairman is already browsing the webpage of the Administrative Area of the meeting, Remote Collaboration tab, “EVO System” selected. The Chairman already indicated the VC endpoint IP address (by having booked a VC room or indicating it in the IP address field), type (PC or H.323 device) and his EVO user ID and Password.

- Normal Flow: 1. The Chairman confirms the action. 2. The system stores the choices and displays the current status. 3. Use case ends.
- Alternate Flow 1: If the Chairman doesn’t confirm his action of booking the meeting use case ends.

- Alternate Flow 2: if the Chairman doesn't' confirm the action to start the meeting, use case ends.

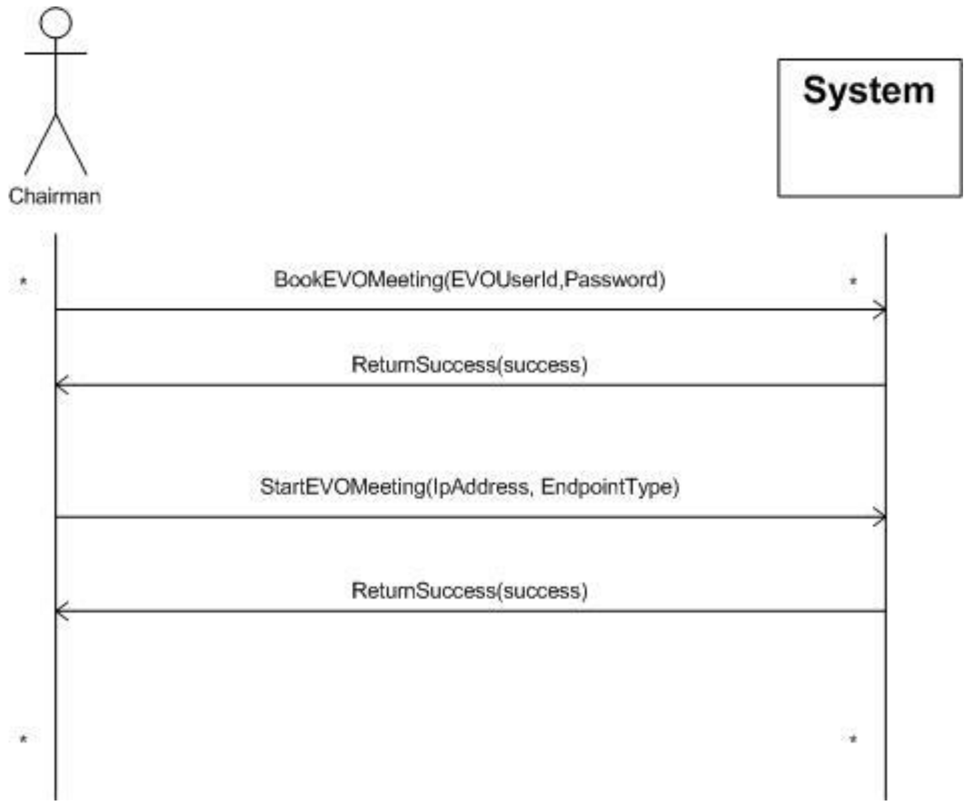

Chairman
System
BookEVOMeeting(EVOUserId,Password)
ReturnSuccess(success)
StartEVOMeeting(IpAddress, EndpointType)
ReturnSuccess(success)

Figure 3.2.4 – Sequence diagram: Booking and Starting a EVO meeting.

Alternatively, if the type of device is not specified at the time of the booking the system will try to use the local PC terminal settings (where the meeting is effectively started), if they exist.

3.2.3 PABXs and WebEx Systems

This section describes two additional use cases for later implementation. The ones immediately foreseen are the common institutional PABXs systems and the WebEx platform. PABX systems are telephone exchange systems that serve a business or institution. Current trends are of having PABXs supporting VoIP systems and providing web Interfaces and System APIs to access their main functions.

- **PABXs Use Case**

InDiCo will have a similar behavior as for use cases described earlier; Through InDiCo the Chairman will be able to get a reservation code in the PABX system being then able to start the meeting.

- Use case: Start PABX Meeting (Figure 3.2.5)
- Preconditions: The Chairman must be logged in. The Chairman already selected the meeting.

The Chairman is already browsing the webpage of the Administrative Area of the meeting, Remote Collaboration tab, “PABX System” selected. The Chairman already indicated the endpoint address (by having booked a VC room or indicating it in the IP address field), type (H.323 device with phone capability or soft phone or normal telephone terminal) and his PABX user ID and Password.

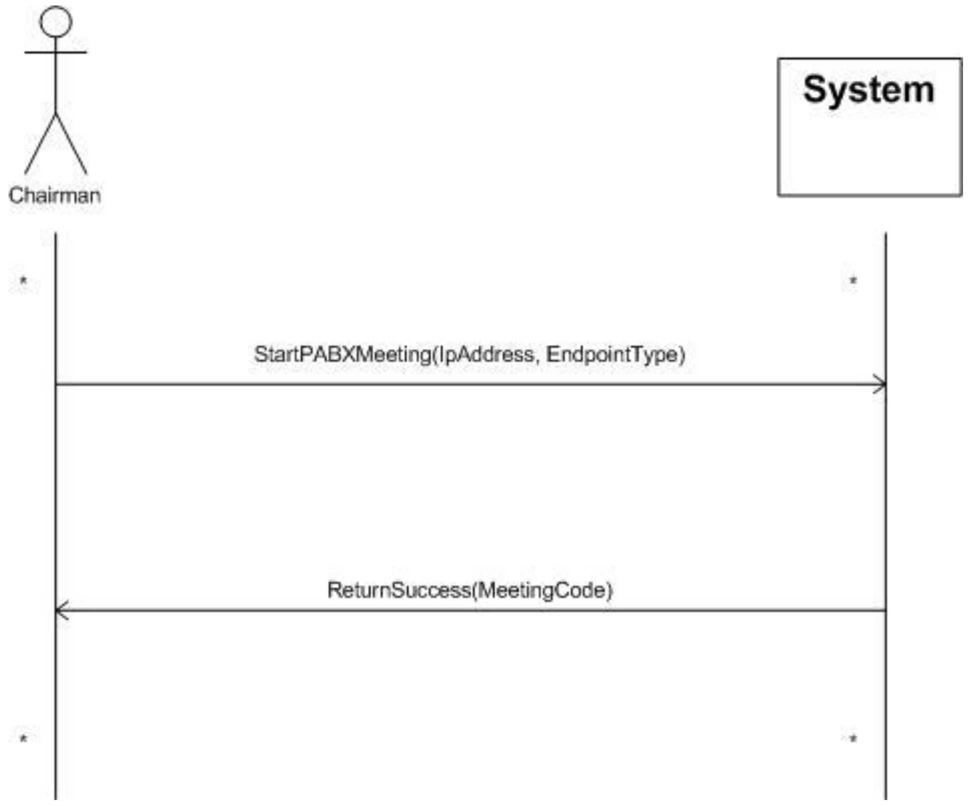

Chairman
System
StartPABXMeeting(IpAddress, EndpointType)
ReturnSuccess(MeetingCode)

Figure 3.2.5 – Sequence diagram: Starting a PABX meeting.

- Normal Flow: 1. The Chairman confirms the action. 2. The system stores the choices and displays the current status. 3. Use case ends.
- Alternate Flow 1: If the Chairman doesn’t confirm his action use case ends.

- **WebEx Use Case**

In this case, the Chairman will use InDiCo to specify that his meeting is using WebEx. Similarly as in the cases before, InDiCo will be provided with the endpoint type and location and connect it to the WebEx platform.

- Use case: Start WebEx Meeting (Figure 3.2.6)
- Preconditions: The Chairman must be logged in. The Chairman already selected the meeting.

The Chairman is already browsing the webpage of the Administrative Area of the meeting, Remote Collaboration tab, “WebEx System” selected. The Chairman already indicated the endpoint IP address (by having booked a VC room or indicating it in the IP address field) and type (H.323 device with phone capability, or normal telephone endpoint) and his EVO user ID and Password. Alternatively, more additional phone numbers of remote participants can also be provided.

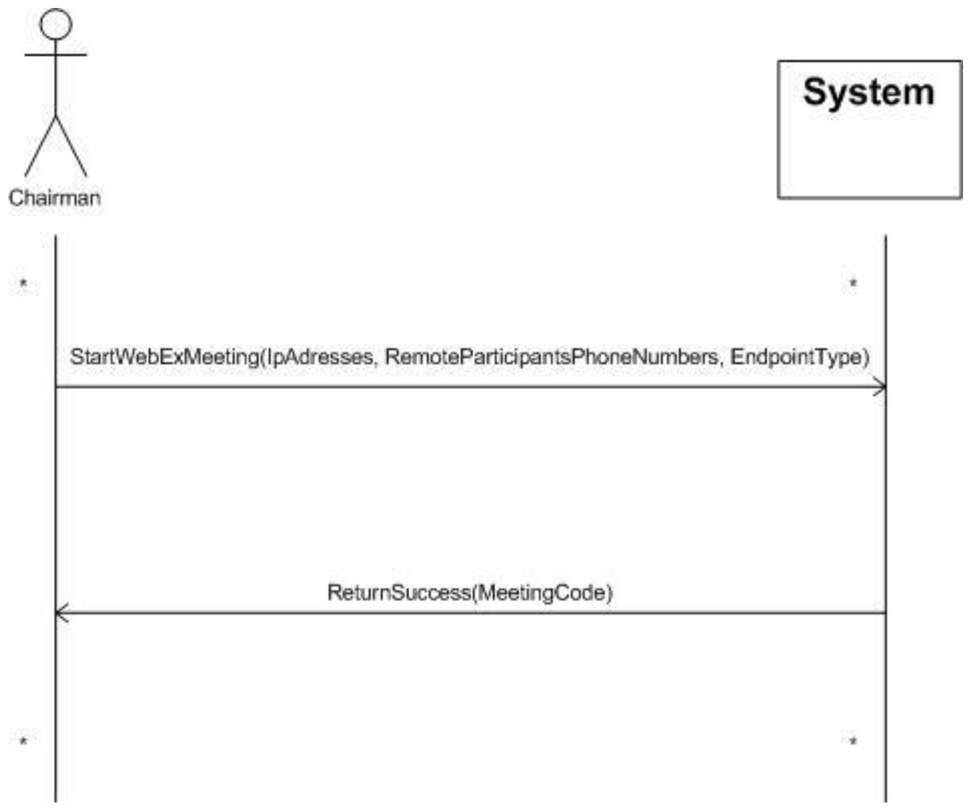

Chairman
System
*
*
StartWebExMeeting(IpAdresses, RemoteParticipantsPhoneNumbers, EndpointType)
ReturnSuccess(MeetingCode)
*
*

Figure 3.2.6 - Sequence diagram: Starting a WebEx meeting.

- Normal Flow: 1. The Chairman confirms the action. 2. The system stores the choices and displays the current status. 3. Use case ends.
- Alternate Flow 1: If the Chairman doesn’t confirm his action use case ends.

3.2.4 Mixed Distributed Meetings

In the event that there is a remote meeting with a mixed set of collaborative systems, this use case is a combination of the use cases specified before. One common case would be for instance, the combination of an H.323 MCU meeting with an EVO meeting (due to bandwidth restrictions, for example EVO Desktop clients have advantages in connecting in low bandwidth locations what is not the case for H.323 devices).

- Use Case: Combination of Start EVO Meeting with Start MCU Meeting (Figure 3.2.7)
- Preconditions: The Chairman must be logged in. The Chairman already selected the meeting. The Chairman is already browsing the webpage of the Administrative Area of the meeting, Remote Collaboration tab, “MCU System” selected. The Chairman already indicated the MCU IP address from a list (if not the using the one by default), VC endpoint IP address (by having booked a VC room or indicating it in the IP address field) and alternatively, the remote participants IP addresses. The Chairman also indicated that he is bridging the meeting with another system (EVO in this case).

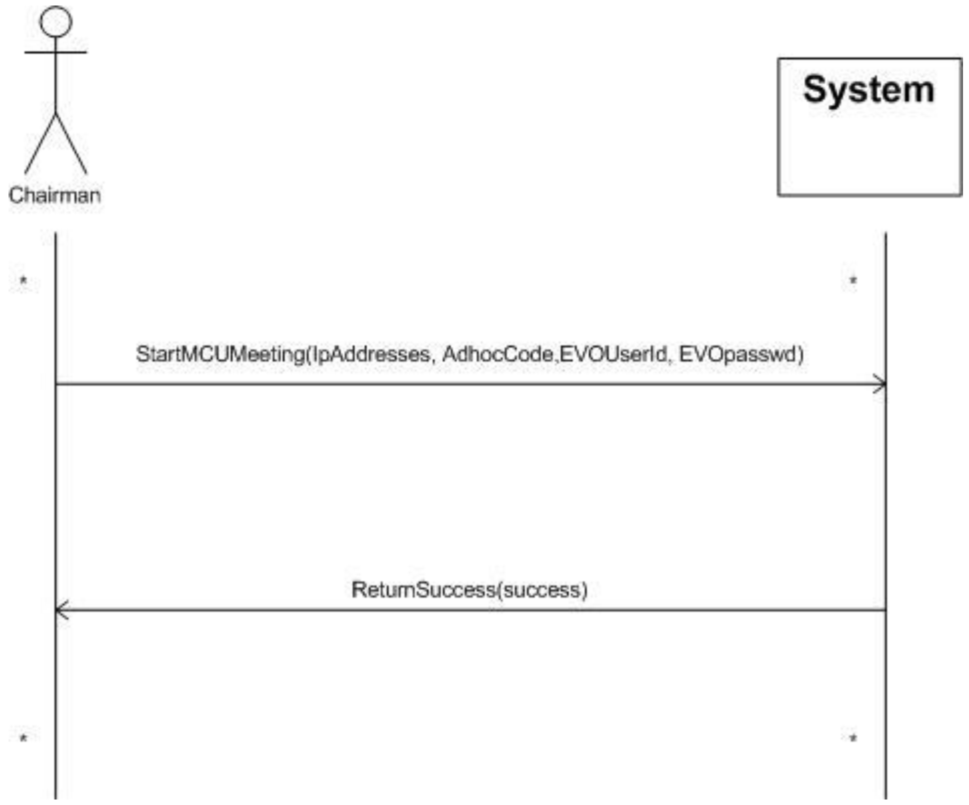

Chairman
System
*
*
StartMCUMeeting(IpAddresses, AdhocCode,EVOUserId, EVOpasswd)
ReturnSuccess(success)
*
*

Figure 3.2.7 - Sequence diagram: Example of a mixed meeting.

- Normal Flow: 1. The Chairman confirms the action. 2. The system stores the choices and displays the current status. 3. Use case ends.
- Alternate Flow 1: If the Chairman doesn’t confirm his action use case ends.

3.3 Requirements Definition

The requirements for this project were identified. The main one is obviously to unify the most current cases of remote collaboration; some of them have been formalized in section 3.2.

Other requirements are identified as follows:

- Module to be fully integrated in the InDiCo framework

- To rely in a modular architecture where APIs for a given system can be added or removed without affecting the functionality of each other.
- To support the connectivity of H.323 devices directly and via an MCU
- To support EVO bookings and EVO connections.
- To be cross platform (requirement fulfilled by the integration on the InDiCo web framework)

Due to both technology availability and time constraints, the following requirements are also defined, but the implementation is to be done at a later stage:

- To support the connection of traditional large organization PABX systems
- To support of WebEx connections

In a third phase, one also identifies some additional requirements:

- To support the connectivity of 3G devices
- Add a degree of intelligence to the system, in order to be able to take decisions in whom to connect and in which conditions.

3.4 Context of usage

The schema in Figure 3.4.1 depicts the general user context of the module and the relations between components and scenarios, as the final goal of the project. The cases gathered are only a sample of cases. These are the ones that the system is intended to fulfill in the mid-term. As the technology and user requirements rapidly evolve one should note that the presented cases can be modified or even removed, as new ones can become important to implement (for instance the addition of the support of Microsoft OCS). OCS is particularly interesting as it presents already a “unifying” in itself architecture being able to integrate several collaborative technologies from instant messaging to phone.

All these aspects emphasize the importance of having a modular architecture for this InDiCo subsystem, where functionality can be added or removed without affecting the entire structure.

As of functionality, using the provided interface, the module is supposed to contact any of the systems depicted (being an H.323 terminal, EVO client, etc.) through its respective API in order to connect to it get the relevant information (meetings, users, system status, etc.). This information is then used to connect users to the respective InDiCo managed event.

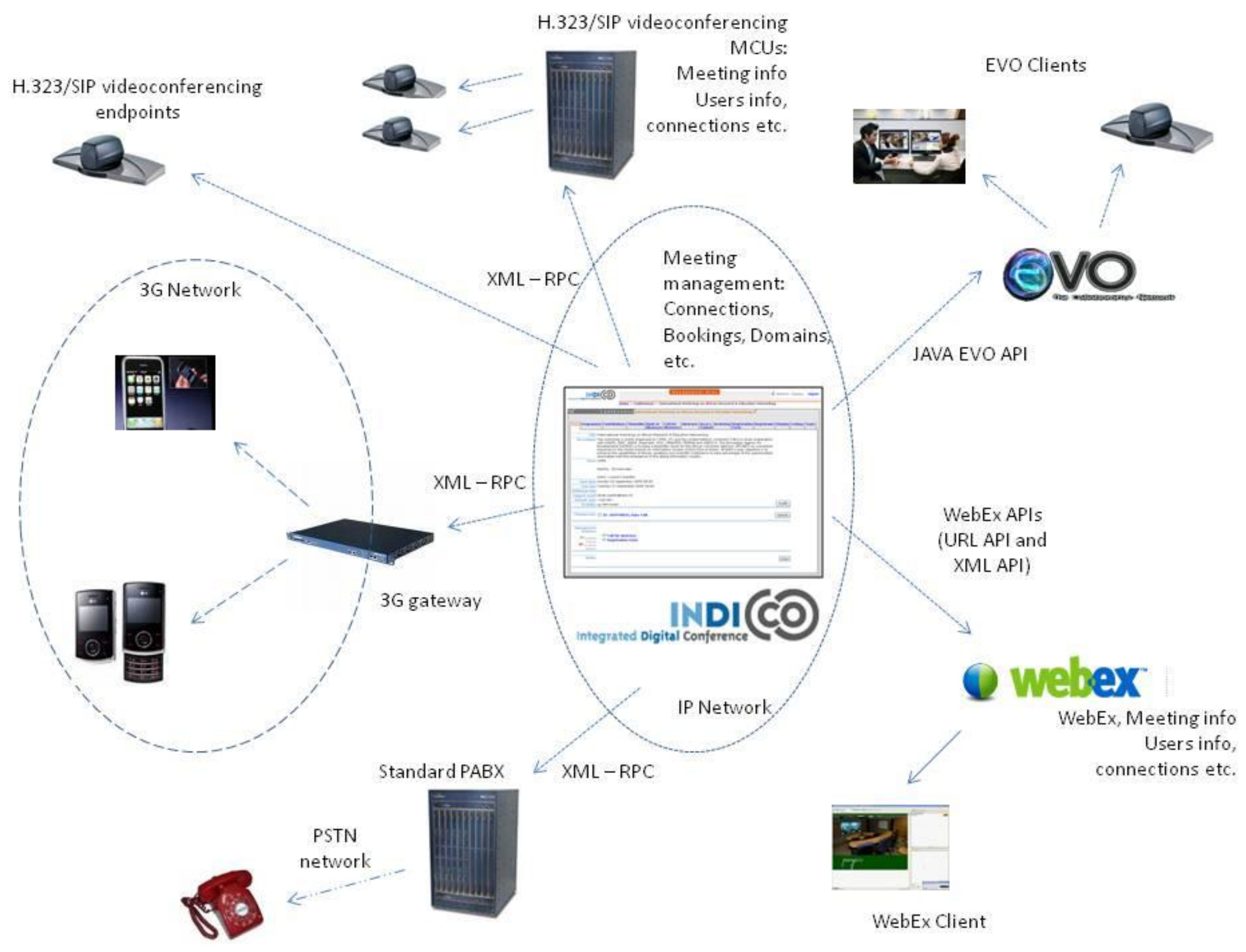

H.323/SIP videoconferencing MCUs: Meeting info Users info, connections etc.
H.323/SIP videoconferencing endpoints
EVO Clients
3G Network
XML – RPC
Meeting management: Connections, Bookings, Domains, etc.
JAVA EVO API
XML – RPC
3G gateway
WebEx APIs (URL API and XML API)
INDICO Integrated Digital Conference
IP Network
webex
WebEx, Meeting info Users info, connections etc.
Standard PABX
XML – RPC
PSTN network
WebEx Client

Figure 3.4.1 - General Context of Usage.

Chapter 4

Implementation

This section, describes the approach adoptedfor developing the proposed system, the work conducted and the (preliminary) results obtained. In a first step, the architecture specified for the module is presented, taking into account the set of requirements defined earlier. This is followed by some details regarding the operations of and the first prototype implementation available. The problem that one has to address has been clearly identified: to reduce complexity of users when setting up remote distributed sessions by unifying the user operations under a single interface. The interface chosen is embedded in InDiCo, as InDiCo already provides many of the functionalities regarding the management of events. The first prototype developed covers the H.323 use case (see 3.2.1). Before directly implementing this use case, there was the need of developing a common interface embedded in the InDiCo management area. This interface is presented in InDiCo as a new tab labelled “Remote Collaboration”. This sub interface was developed in a way that can gather all the API implementations, ongoing APIs and future ones that might turn relevant. This allows gathering all the common functionality in a modular approach not only in a functional perspective but also in a user perspective.

The implementation of the H.323 use case was done using the XML-RPC protocol, namely, its Python implementation (*xmlrpclib*).

The full implementation of this use case will allow the user to quickly setup a meeting with an H.323 device in a H.323 MCU, without the need of distributing IP addresses, access codes and other relevant connectivity information among the participants of the remote distributed session.

The second use case defined (see 3.2.2) is currently being developed using the JAVA EVO API provided by the EVO system. This API will allow the possibility of adding the basic functionality of booking and starting an EVO meeting from the InDiCo “Remote Collaboration” interface.

4.1 Proposed Module Architecture

The module architecture specified follows and basically fulfills two of the main requirements defined: to be fully integrated in InDiCo structure and to be the most flexible and scalable possible in terms of modular functionality. The Figure 4.4.1 depicts the InDiCo core structure.

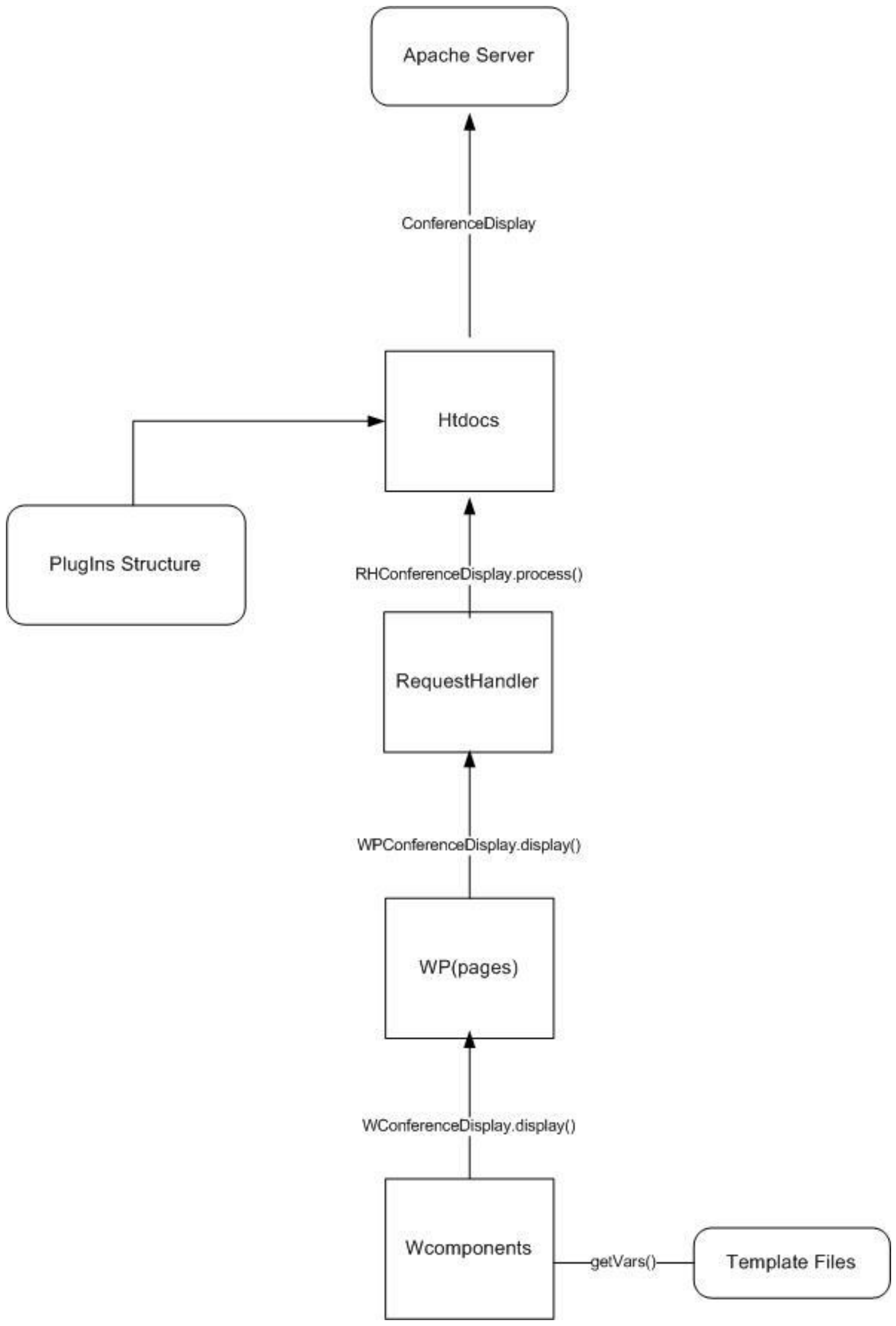
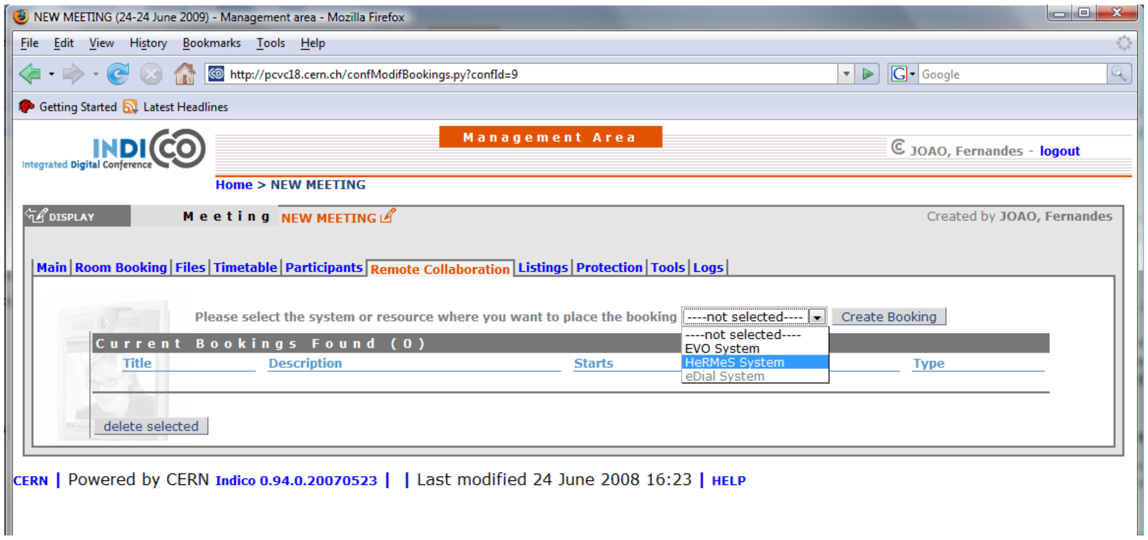


Figure 4.1.1 - InDiCo Core Structure.

The several components of this architecture are explained later in this document (see 4.2). In order to satisfy the first requirement just mentioned, the same flux of information approach of InDicCo was followed and used as base for the plug-in module (described in the next section). This was done in order to facilitate and to maximize the integration of the module in the InDiCo structure.

In order to address the second one, the module has been defined has a plug-in. To define the module as a plug-in has some direct advantages. It allows activating on administrator request the whole module in an InDiCo instance. It also offers a deeper modular granularity, as it additionally gives the possibility of customizing the activation of the underlying supported actions without affecting any functionality of the core system. For instance, within an Organization, one may want to activate only the ability of booking and connecting to EVO sessions but it is not interested in being able to start generic MCU sessions. As an example, the Figure 4.4.1.2 depicts the prototype interface embedded in InDiCo management area, showing two out of three of these possibilities active.


File Edit View History Bookmarks Tools Help
Getting Started
Latest Headlines
Google
INDICO
Integrated Digital Conference
Management Area
logout
DISPLAY
Main | Room Booking | Files | Timetable | Participants | Remote Collaboration | Listings | Protection | Tools | Logs
Create Booking
Title
Description
Starts
Type
delete selected

NEW MEETING (24-24 June 2009) - Management area - Mozilla Firefox

http://pcvc18.cern.ch/confModifBookings.py?confId=9

JOAO, Fernandes

Home > NEW MEETING

Meeting NEW MEETING

Created by JOAO, Fernandes

Please select the system or resource where you want to place the booking ----not selected----

----not selected----
EVO System
HeRMeS System
eDial System

Current Bookings Found (0)

CERN | Powered by CERN Indico 0.94.0.20070523 | | Last modified 24 June 2008 16:23 | HELP

Figure 4.1.2 - Prototype showing different systems, to book or use.

It is also foreseen that one will have the option of installing this plug-in, at InDiCo installation time (the advantage of a plug-in is exactly to be installed when needed). This will be possible by adding information about existence of this module to the XML configuration file used for InDiCo installation (*config.xml*). This XML configuration file gathers all the relevant information regarding the installation of InDiCo. For this particular case, it will keep relevant data about module (options, flags to enable the module in different type of events, system IP addresses and DNS, system passwords, etc). All these parameters will then be defined and configured at installation time. This brings concrete advantages otherwise as the underlying systems can very easily change for instance a server IP address or DNS name, one would be force to make this modifications directly in the source code.

4.2 Module Operations

The module has been defined as a plug-in. As mentioned before, the module has the same request procedures of the InDiCo architecture for the reasons already explained (see 4.1). In the diagram in Figure 4.2.1, the operations based in the data flow of the module can support are depicted in the figure below (Figure 4.2.1).

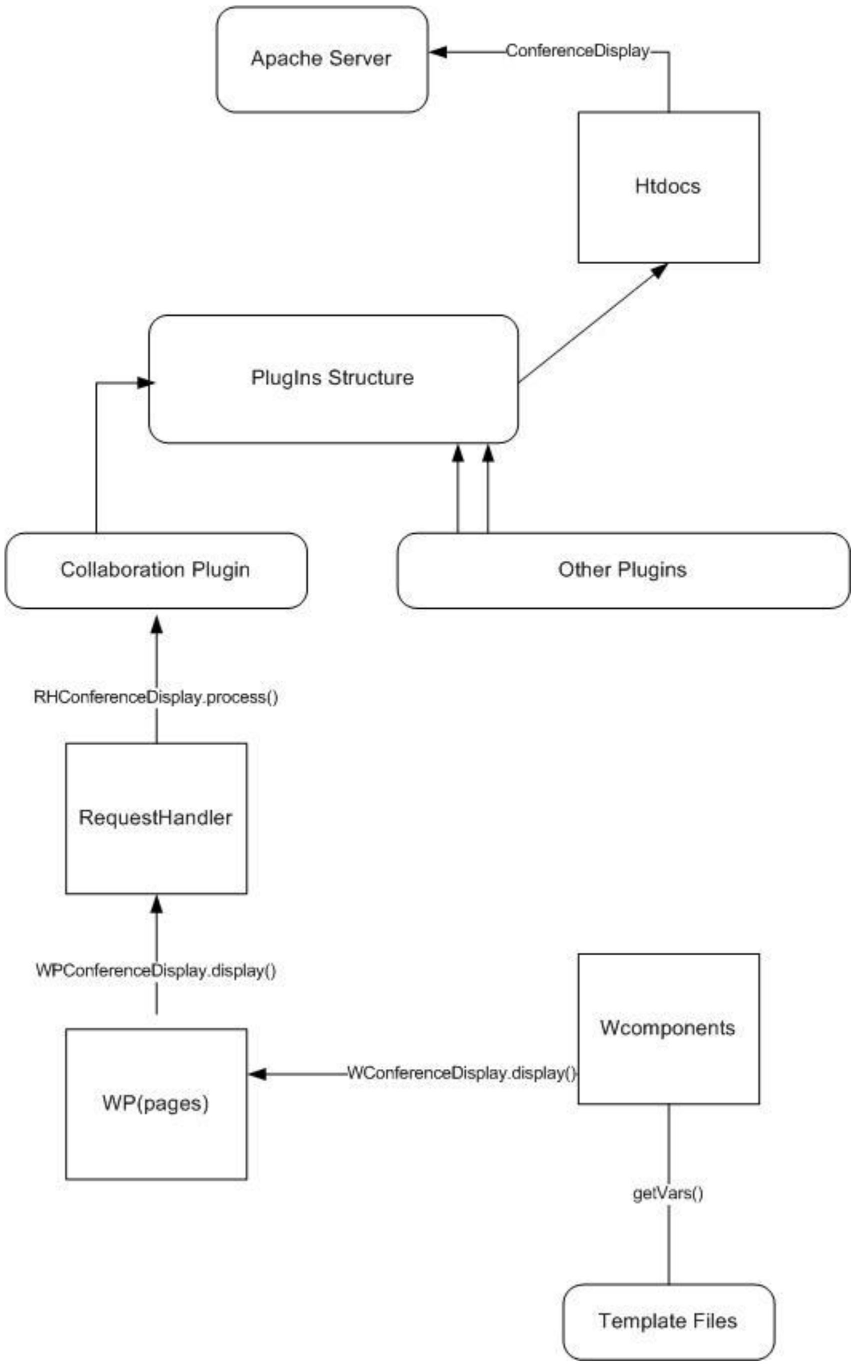
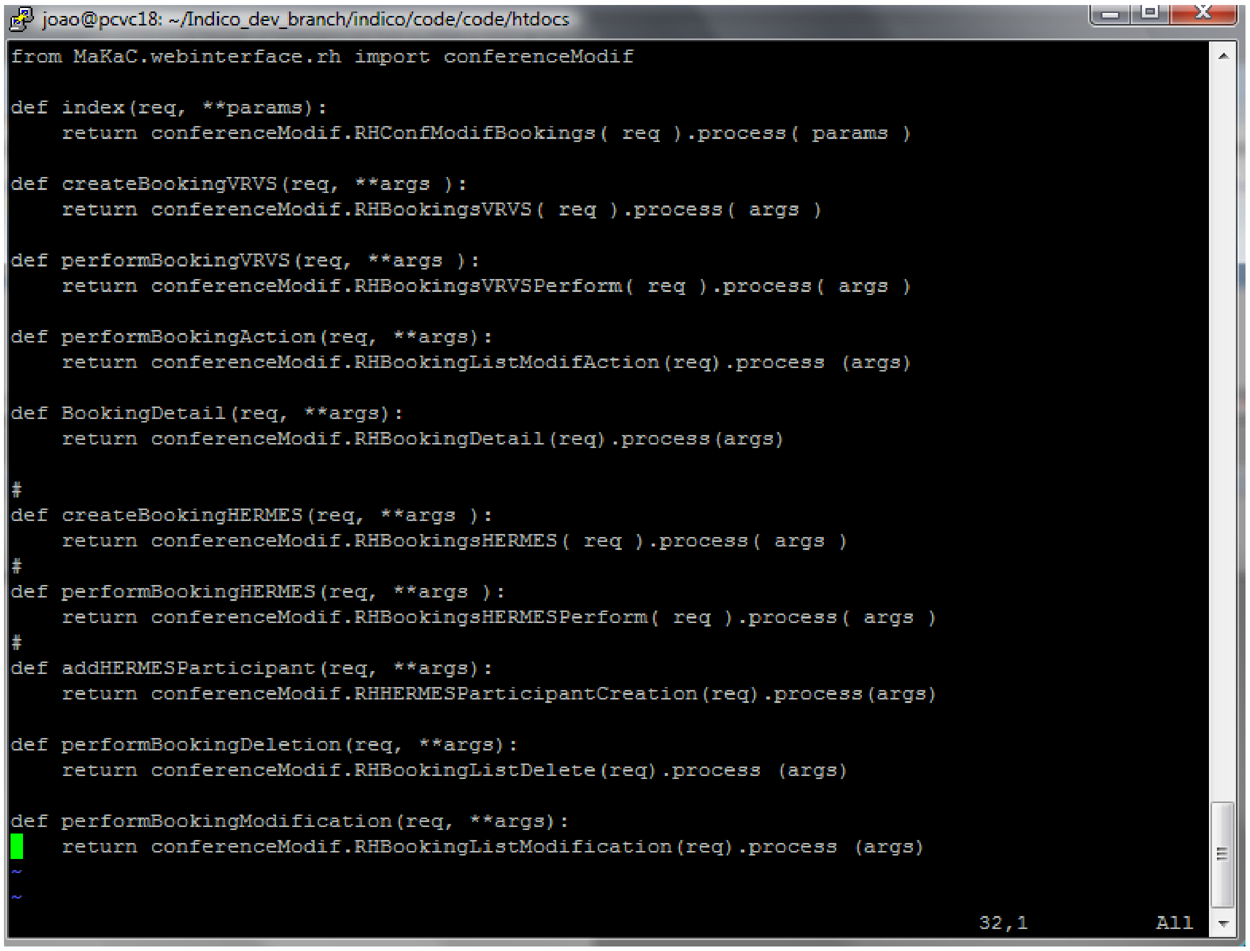


Figure 4.2.1 - Data Flow of the Module.

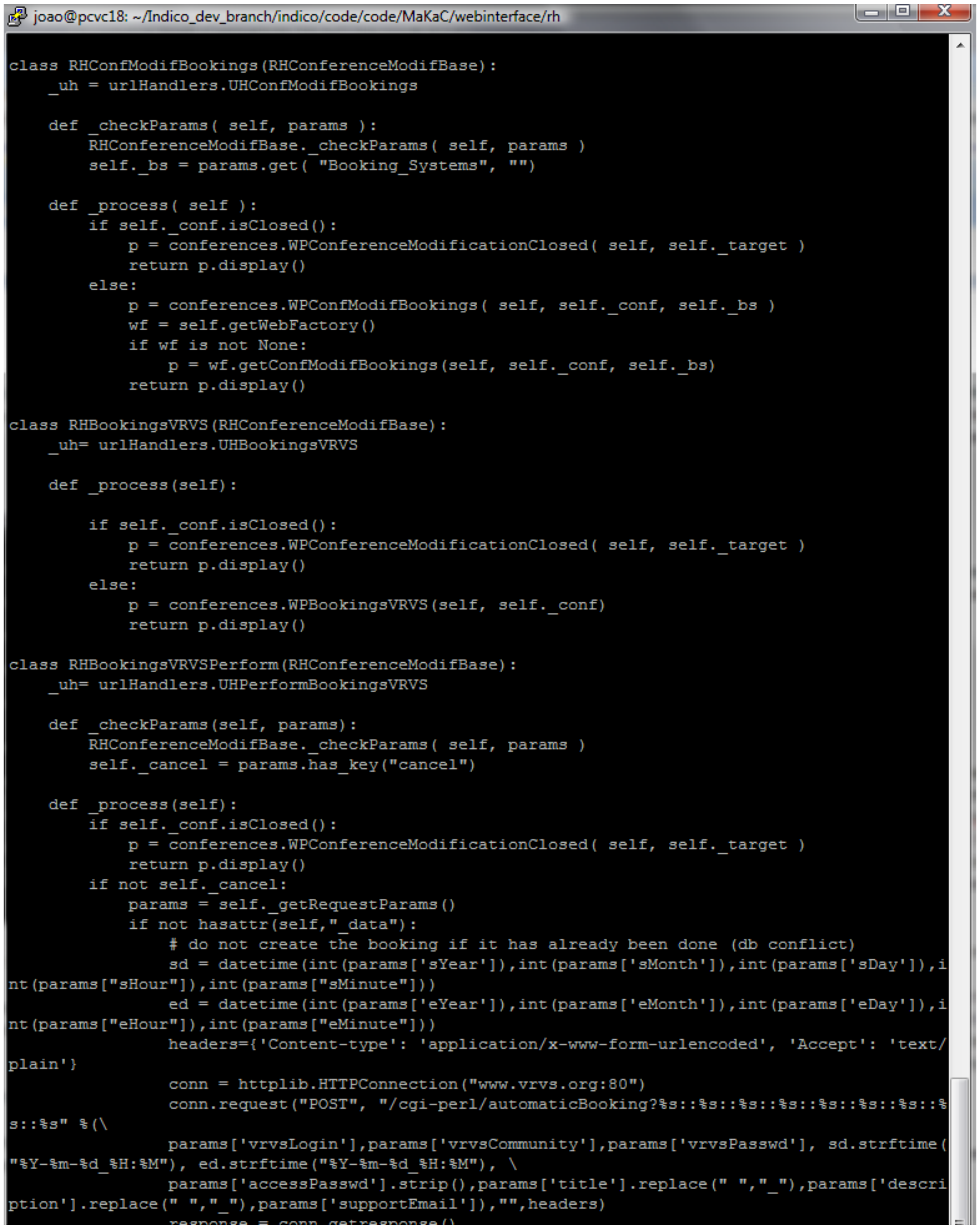
The files in htdocs in InDiCo core structure, contain functions called by the browser when a user makes a request (webpage or performs an action). These functions make a redirection to the functions contained in the request handler classes. An example of an htdocs file is depicted in Figure 4.2.2.



```
from MaKaC.webinterface.rh import conferenceModif

def index(req, **params):
    return conferenceModif.RHConfModifBookings( req ).process( params )

def createBookingVRVS(req, **args ):
    return conferenceModif.RHBookingsVRVS( req ).process( args )

def performBookingVRVS(req, **args ):
    return conferenceModif.RHBookingsVRVSPerform( req ).process( args )

def performBookingAction(req, **args):
    return conferenceModif.RHBookingListModifAction(req).process (args)

def BookingDetail(req, **args):
    return conferenceModif.RHBookingDetail(req).process(args)

#
def createBookingHERMES(req, **args ):
    return conferenceModif.RHBookingsHERMES( req ).process( args )
#
def performBookingHERMES(req, **args ):
    return conferenceModif.RHBookingsHERMESPerform( req ).process( args )
#
def addHERMESParticipant(req, **args):
    return conferenceModif.RHHERMESParticipantCreation(req).process(args)

def performBookingDeletion(req, **args):
    return conferenceModif.RHBookingListDelete(req).process (args)

def performBookingModification(req, **args):
    return conferenceModif.RHBookingListModification(req).process (args)
~
~
                                                          32,1          All
```

Figure 4.2.2 - Example of an htdocs file.

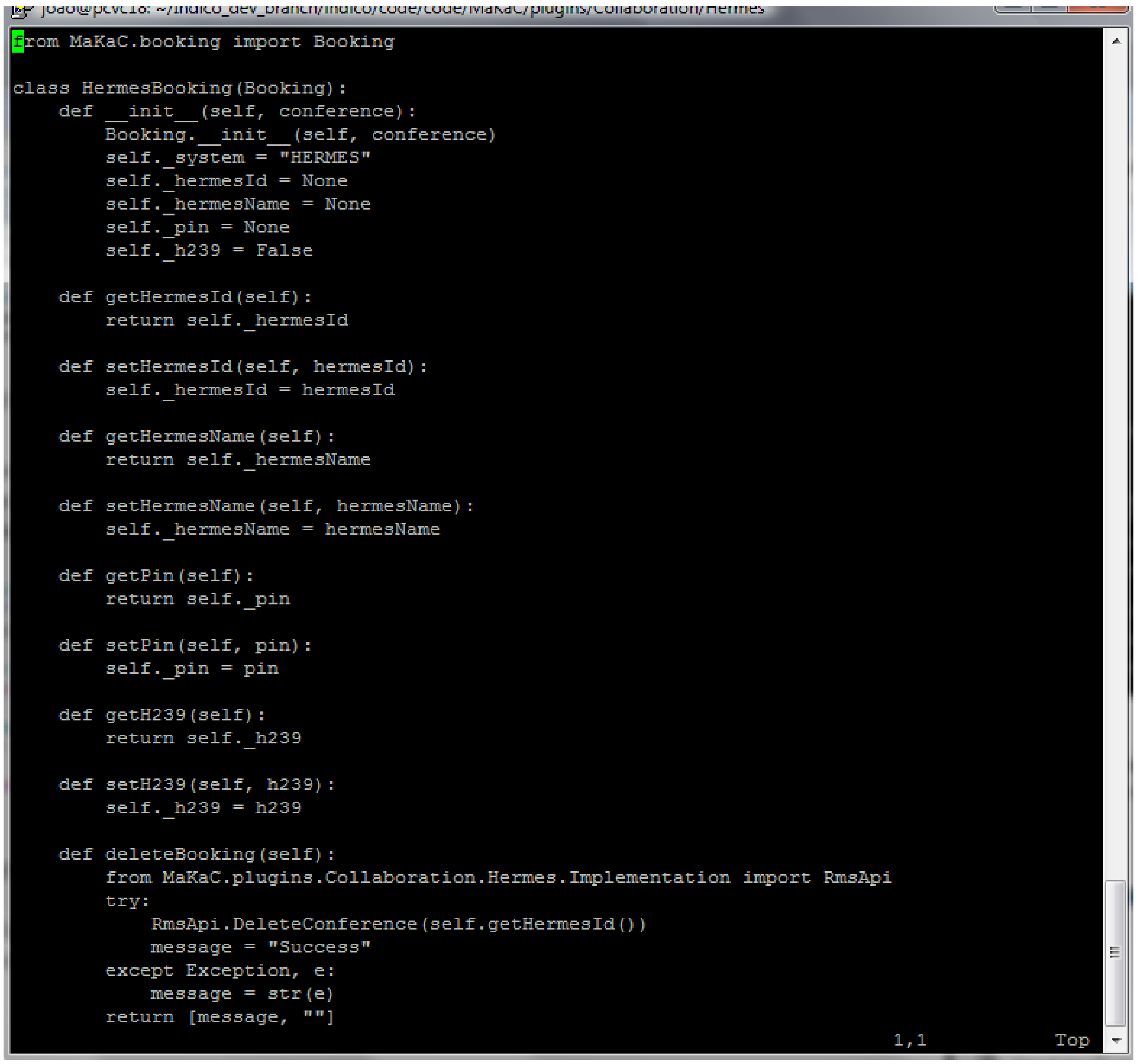
The files in the request handler (RH) are used when the user performs an action, in order to handle the request and call the functions of the core classes needed to execute the action. Figure 4.2.3 depicts an example of a request handler file.


joao@pcvc18: ~/Indico_dev_branch/indico/code/code/MaKaC/webinterface/rh

```
class RHConfModifBookings(RHConferenceModifBase):
    _uh = urlHandlers.UHConfModifBookings

    def _checkParams( self, params ):
        RHConferenceModifBase._checkParams( self, params )
        self._bs = params.get( "Booking_Systems", "")

    def _process( self ):
        if self._conf.isClosed():
            p = conferences.WPConferenceModificationClosed( self, self._target )
            return p.display()
        else:
            p = conferences.WPConfModifBookings( self, self._conf, self._bs )
            wf = self.getWebFactory()
            if wf is not None:
                p = wf.getConfModifBookings(self, self._conf, self._bs)
            return p.display()

class RHBookingsVRVS(RHConferenceModifBase):
    _uh= urlHandlers.UHBookingsVRVS

    def _process(self):

        if self._conf.isClosed():
            p = conferences.WPConferenceModificationClosed( self, self._target )
            return p.display()
        else:
            p = conferences.WPBookingsVRVS(self, self._conf)
            return p.display()

class RHBookingsVRVSPerform(RHConferenceModifBase):
    _uh= urlHandlers.UHPerformBookingsVRVS

    def _checkParams(self, params):
        RHConferenceModifBase._checkParams( self, params )
        self._cancel = params.has_key("cancel")

    def _process(self):
        if self._conf.isClosed():
            p = conferences.WPConferenceModificationClosed( self, self._target )
            return p.display()
        if not self._cancel:
            params = self._getRequestParams()
            if not hasattr(self,"_data"):
                # do not create the booking if it has already been done (db conflict)
                sd = datetime(int(params['sYear']),int(params['sMonth']),int(params['sDay']),i
nt(params["sHour"]),int(params["sMinute"]))
                ed = datetime(int(params['eYear']),int(params['eMonth']),int(params['eDay']),i
nt(params["eHour"]),int(params["eMinute"]))
                headers={'Content-type': 'application/x-www-form-urlencoded', 'Accept': 'text/
plain'}
                conn = httplib.HTTPConnection("www.vrvs.org:80")
                conn.request("POST", "/cgi-perl/automaticBooking?%s::%s::%s::%s::%s::%s::%s::%
s::%s" %(\
                params['vrvsLogin'],params['vrvsCommunity'],params['vrvsPasswd'], sd.strftime(
"%Y-%m-%d_%H:%M"), ed.strftime("%Y-%m-%d_%H:%M"), \
                params['accessPasswd'].strip(),params['title'].replace(" ","_"),params['descri
ption'].replace(" ","_"),params['supportEmail']),"",headers)
```

Figure 4.2.3 - Example of an RH file.

The core classes are the persistent part of the application. They contain the functions that let the program run, being its kernel as they contain the persistent objects and functions. An example of a core class can be seen in Figure 4.2.4.

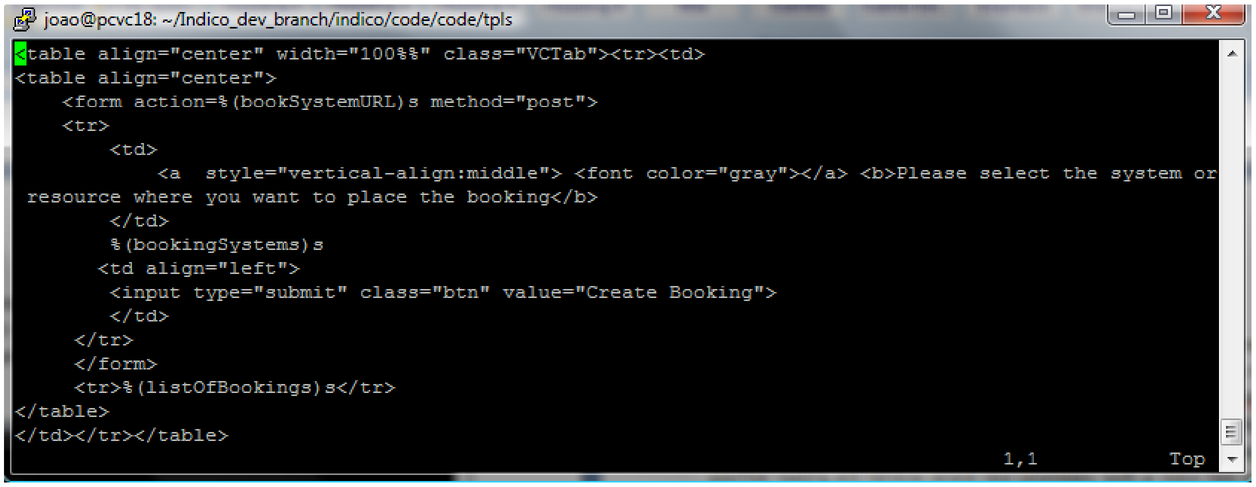


```
from MaKaC.booking import Booking

class HermesBooking(Booking):
    def __init__(self, conference):
        Booking.__init__(self, conference)
        self._system = "HERMES"
        self._hermesId = None
        self._hermesName = None
        self._pin = None
        self._h239 = False

    def getHermesId(self):
        return self._hermesId

    def setHermesId(self, hermesId):
        self._hermesId = hermesId

    def getHermesName(self):
        return self._hermesName

    def setHermesName(self, hermesName):
        self._hermesName = hermesName

    def getPin(self):
        return self._pin

    def setPin(self, pin):
        self._pin = pin

    def getH239(self):
        return self._h239

    def setH239(self, h239):
        self._h239 = h239

    def deleteBooking(self):
        from MaKaC.plugins.Collaboration.Hermes.Implementation import RmsApi
        try:
            RmsApi.DeleteConference(self.getHermesId())
            message = "Success"
        except Exception, e:
            message = str(e)
        return [message, ""]
                                                                              1,1           Top
```

Figure 4.2.4 - Example of a Core Class.

We then have the interface classes. The interface classes are responsible for building the HTML answer to the HTTP request. To build this answer, they can either contain directly HTML code or communicate with the HTML templates by providing variables to a template or receiving data when a user performs an action.

In practice, these classes build the front-end of the application. The template files are HTML that creates the structure of the pages. They receive and provide data (via the use of variables) from/to the interface classes. An example of a template file is shown in Figure 4.2.5.



joao@pcvc18: ~/Indico_dev_branch/indico/code/code/tpls

```
<table align="center" width="100%%" class="VCTab"><tr><td>
<table align="center">
    <form action=%(bookSystemURL)s method="post">
    <tr>
        <td>
            <a  style="vertical-align:middle"> <font color="gray"></a> <b>Please select the system or
 resource where you want to place the booking</b>
        </td>
        %(bookingSystems)s
       <td align="left">
        <input type="submit" class="btn" value="Create Booking">
        </td>
     </tr>
     </form>
     <tr>%(listOfBookings)s</tr>
</table>
</td></tr></table>
                                                                    1,1           Top
```

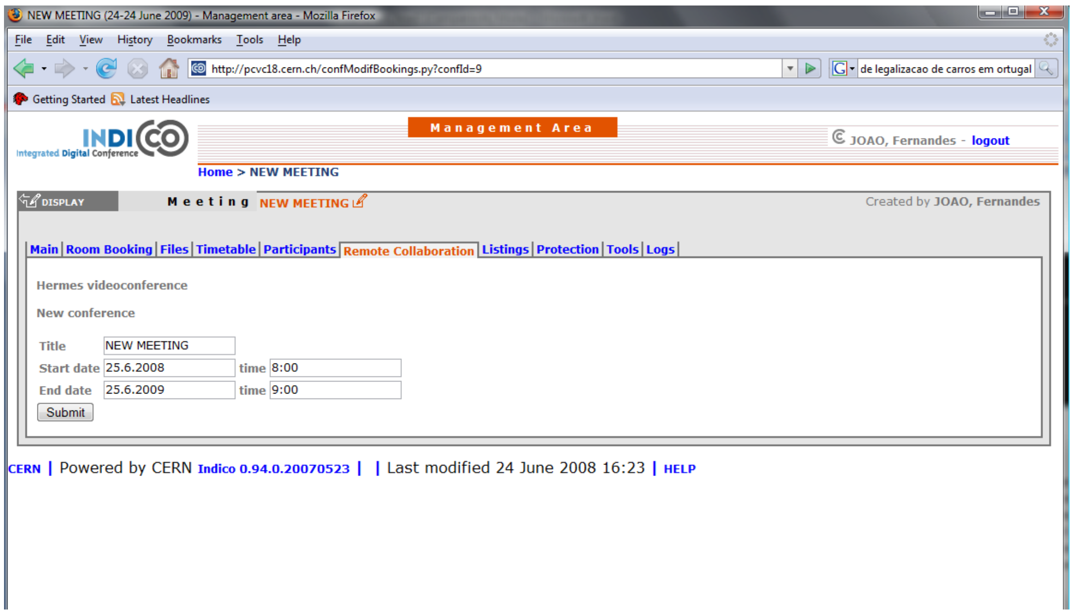
Figure 4.2.5 - Example of a Template file.

4.3 Prototype

The implemented prototype is basically gathering three different parts of development: the user interface embedded in the InDiCo Management Area (the “Remote Collaborations” tab), the allocation interface for the use case implemented (currently the H.323 use case only) that actually creates the booking, the interface that gathers the external system information and gives the user the possibility to start a meeting and the underlying functionality of contacting the XML-RPC API available in an H.323 generic MCU, in this case, used by the these last two aforementioned interfaces.

The interface embedded in InDiCo Management Area (Figure 4.4.1.2) follows the structure of InDiCo core code. The InDiCo code is strongly based in the inheritance paradigm present in Object-Oriented languages, as Python. Using this paradigm, the classes that build this sub interface (see interface classes in 4.2) inherit the common state and behavior of InDiCo core classes. These classes both define web pages that can define the whole interface or Web components that define only parts of it. This allows reconstructing the user interface flexibly focusing only on the parts needed to be changed. This interface will be the main one for this module. It has two main roles: giving the user the ability of choosing the remote collaborative system to be used (from a pop-up list of supported or/and activated systems) and, at the same time, displaying the status of bookings or reservations already done for the current event. This is achieved by displaying the main parameters such as time of the booking, name and type of system used. This information status provides also a link to the respective interface to connect to the meeting.

The interface to create the booking is displayed once the user clicks on “Create Booking” just after choosing the system to use. Currently only the H.323 use case interface is available (Figure 4.3.1). This interface allows the user to choose the time and date for the meeting and title for the booking (by default the same as the one of the InDiCo event in question).


File Edit View History Bookmarks Tools Help
Getting Started
Latest Headlines
INDICO Integrated Digital Conference
Management Area
logout
DISPLAY
Main | Room Booking | Files | Timetable | Participants | Remote Collaboration | Listings | Protection | Tools | Logs
Submit
CERN | HELP

NEW MEETING (24-24 June 2009) - Management area - Mozilla Firefox

http://pcvcl8.cern.ch/confModifBookings.py?confId=9

de legalizacao de carros em ortugal

JOAO, Fernandes

Home > NEW MEETING

Meeting NEW MEETING

Created by JOAO, Fernandes

Hermes videoconference

New conference

Title	NEW MEETING		
Start date	25.6.2008	time	8:00
End date	25.6.2009	time	9:00

Powered by CERN Indico 0.94.0.20070523 | | Last modified 24 June 2008 16:23

Figure 4.3.1 - Prototype user interface for the H323 use case to enter the data to connect.

The H.323 use case prototype is able to start and display the status of the connection to a generic MCU used at CERN (based at a collaborator institute, IN2P3 in France). To gather the information from the MCU, the module uses the *xmlrpclib* module provided by Python to connect as a client to the XML-RPC server accessible via the XML-RPC API of the MCU hardware unit.

XML-RPC it is a simple specification for remote procedure calls (RPC) that uses HTTP as the transport protocol and an XML vocabulary as the message payload. Using XML-RPC clients in Python can be very straightforward, since the Python module *xmlrpclib* has all the needed machinery. A typical Python function for the creation of a meeting in an MCU using Python *xmlrpclib* module would be:

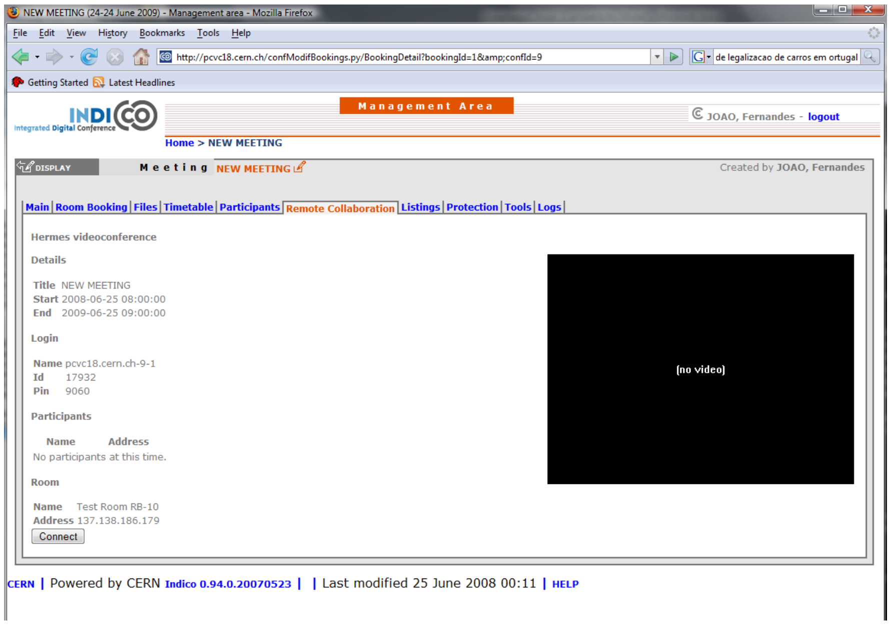
```
import xmlrpclib
def CreateConference(name, start, end, pin):
    serverProxy = xmlrpclib.ServerProxy("http://serverhostname/")
    serverUser = "user"
    serverPasswd =  “passwd”
    startDateTime
    durationSeconds
    if (pin == None): pin = ""
    else: pin = str(pin)
    return serverProxy.conference.create(startDateTime, durationSeconds, name, pin, serverUser,
    serverPassword)
```

As one can see, the server is proxied and set up by initializing an instance of the *ServerProxy* class. One has to pass in the full URL of the remote server (including the URL scheme "http://"). Port 80 is the default.

In this particular case, this approach to create or destroy a conference was not exactly followed although the functionality remains the same. The practical reason for this is the fact that the generic MCU in question is also accessed by other third-party applications. For this reason, the module instead of contacting directly the MCU XML-RPC server contacts a custom Web Service that manages all the calls to the XML-RPC server. In this way, one avoids that the data about creation or destruction of a conference in the MCU is disrupted or becomes unreliable, since the custom Web Service is aware of all the actions of conference creation or conference destruction. For simple system query or conference status queries, the direct XML-RPC server approach can be used.

This interface needs still several improvements and refinements. A main feature missing is the ability to specify a different user endpoint that the one set by default (the one present in the physical location of the meeting). Another feature missing is the possibility to allow the specification of external participant's endpoints. This would allow at the following stage, to connect not only the user but also other participants. These refinements are seen as crucial to increase the usefulness of the interface when in production.

Once all the relevant data is submitted (IP address, meeting PIN, date, time, length, etc.), InDiCo stores this information in its Object Oriented database. When the user connects the meeting, different information is retrieved like several data parameters, status data and a video stream from the meeting (if the streaming is enabled for the meeting). Figure 4.3.2 depicts the prototype interface to connect to the meeting.


NEW MEETING (24-24 June 2009) - Management area - Mozilla Firefox

http://pcvc18.cern.ch/confModifBookings.py/BookingDetail?bookingId=1&amp;confId=9

de legalizacao de carros em ortugal

Management Area

JOAO, Fernandes - logout

Home > NEW MEETING

Meeting NEW MEETING

Created by JOAO, Fernandes

Main | Room Booking | Files | Timetable | Participants | Remote Collaboration | Listings | Protection | Tools | Logs

Hermes videoconference

Details

Title NEW MEETING
Start 2008-06-25 08:00:00
End 2009-06-25 09:00:00

Login

Name pcvc18.cern.ch-9-1
Id 17932
Pin 9060

Participants

Name Address
No participants at this time.

Room

Name Test Room RB-10
Address 137.138.186.179

(no video)

CERN | Powered by CERN Indico 0.94.0.20070523 | | Last modified 25 June 2008 00:11 | HELP

Figure 4.3.2 - Prototype interface to connect to the meeting.

Similarly, as in the case described before, this interface also needs refining. Apart from minor rearrangements of the display status, it is probably convenient to remove the video streaming content from the embedded Web page placing it in a separate window being a Real Player window (the option of starting the streaming would always be present whenever the interface is displayed). This allows the general InDiCo interface to be freed up for other tasks related to the management of the conference. One can also think, at a later stage, to implement the API commands to manipulate the video layout transmitted to the participants. In a MCU system, it is possible to manipulate the video layout having several possibilities: voice switched allows the video layout to change in function of the participant that speaks in a given moment (giving the floor to this current speaker), time switched that specifies a timer to switch the layout between the videos of the participants (by giving the floor to one participant at the time), all videos shown in small widgets, four participant vides plus speaker video, five participant videos plus speaker video, etc. The implementation of these commands will then allow the Chairman to modify and adapt the video layout that is transmitted to the participants through InDiCo. Another improvement foreseen is the ability of enumerating participants in real-time, meaning that the status of anyone that joins or leaves the meeting can be displayed. Since the addition of these functionalities would emphasize the importance of the data interchange, one possibility can be as well having this interface using the Python JSON[23] libraries, which implementation could be ideal for this kind of dynamic data displaying, avoid page reloading.

These are only some examples of improvements or modifications currently ongoing. The potential of an MCU API [7] is high and allows the provision of several features. There are certainly other minor features or modifications that will arise once the module is tested in production and the user feedback is received.

4.4 Ongoing and Future Work

There are several tasks that will follow on this first prototype. One of the main ones would be to reach a final state of the use case described before, not only by making the several interface optimizations already mentioned but also by implementing the possibility of connection with simple H.323 endpoints (in addition to the H.323 MCUs). This would be achieved by using the same approach, developing another interface accessible by the pop-up list in the “Remote Collaboration” tab and by providing a similar connectivity interface and underlying connectivity approach based in XML calls.

A second step would be to make the same kind of implementation for the remaining use cases, based on its priority (this priority is defined among others, in function of the user needs).

From these uses cases, the EVO system use case is already in ongoing development as it assumes a relatively high priority and importance, mainly for the LHC user community. This use case presents

[23] http://www.json.org

essentially the same philosophy of the implementation presented in 4.3, especially in what concerns the meeting creation interface design and development. This interface will only differ in the type of some of the parameters to be submitted or displayed. However, the underlying API functionality is rather different as it is based in a JAVA API instead of XML.

This API allows placing bookings in EVO system and retrieving the relevant information concerning the booking submitted. It also provides the functionality of creating and generating a Java Web Start (JWS) link that allows direct access to the relevant meeting. This direct link will allow the EVO client to be automatically started and the meeting to be directly accessed without the need of authentication. The relevant information needed to connect is passed as parameters of this JWS link.

As a requirement this API will need the Java Virtual Machine (as all Java software) installed in the server where the InDiCo instance is running, in order to work properly.

After having a pool of use cases implemented, one could then pass to the next step that would be the optimization of the module. Part of this optimization for example is certainly adding an option in which the user could setup InDiCo to start the meeting automatically at a given time. This is seen as a starting point to provide the system with a certain degree of intelligence.

One expects at a stage where this two use cases aforementioned are fully implemented, that the user will already feel the comfort and the utility of having InDiCo providing inside its event management mechanism, a unifying interface both to book and start its remote distributed sessions related to the managed event. The implementation of additional use cases as the WebEx system, traditional institutional PABXs, and specially the ability of mixing systems in the same distributed session (see 3.2) will certainly reinforce even more the utility of the module and turn its usage even more solid.

At a later phase, after some period of maturity of the implementations described, one can forecast also the integration of the 3G platform.

As a first approach, this task would be implemented by being able to use and contact a 3G gateway. The Figure 4.4.1 shows a recent model of a 3G Gateway provided by Tandberg[24].

Figure 4.4.1 - Example of a 3G gateway from Tandberg.

A 3G gateway is a network element that connects 3G and IP networks, allowing calls between 3G phones and IP based videoconferencing equipments. It supports industry standard protocols including

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

ISUP[25] or PRI[26] interfaces and interoperates with all 3G handsets and terminals. There are some models already in the market provided mainly by Tandberg and DyLogic and more certainly are to come. The approach chosen would include the same strategy as in other use cases: be able to contact the a 3G gateway API and exchange the information needed to connect to the conference, get system status, negotiate the content sent to 3G endpoints, etc. By making a summary analysis to the market in this domain, it seems there were still no standard APIs (XML or more generally Web Services based) for these devices. Even if it is probably a question of time until they appear as these devices increase in popularity, some gateways provide already support for a set of commands based on Telnet, for example. These commands are generally divided into different groups: System Configuration Commands, General 3G Gateway Commands, System Status, Debug and Special Commands.

The System Configuration commands allow manipulating the configuration of the gateway system. For example, configure direct or via H.323 Gatekeeper calling and set the Gatekeeper parameters.

The General 3G gateway commands allow to manipulate system wide options like define an external management tool for the gateway or set an NTP server for example. The system status commands allow getting information about the current status, like Dial Plan in current use, hard disk state, Ethernet interfaces, etc.

The debug commands enable real time logs for several protocols, namely H323, SIP, H.221, H.234m, RTSP, etc. activities. The Special commands allow a more operational activity like disconnecting calls based on session or terminals identification.

One immediate approach could be about constructing a simple custom Web Services API that could parse and combine these commands in order to give the possibility to contact the gateway by an external system (InDiCo in this case) and get several types of information (e .g type of devices connected, their capabilities, etc.). This custom API would give the flexibility of customizing the functionalities in function of the defined needs, for example setting as a requirement the ability to negotiate the content to transmit to an endpoint based on the analysis of the acquired information of an endpoint terminal.

[25] ISDN User Part or ISUP is part of the Signaling System #7 which is used to set up telephone calls in Public Switched Telephone Networks

[26] The primary rate interface (PRI) is a telecommunications standard for carrying multiple DS0 voice and data transmissions between two physical locations.

Chapter 5

Conclusions

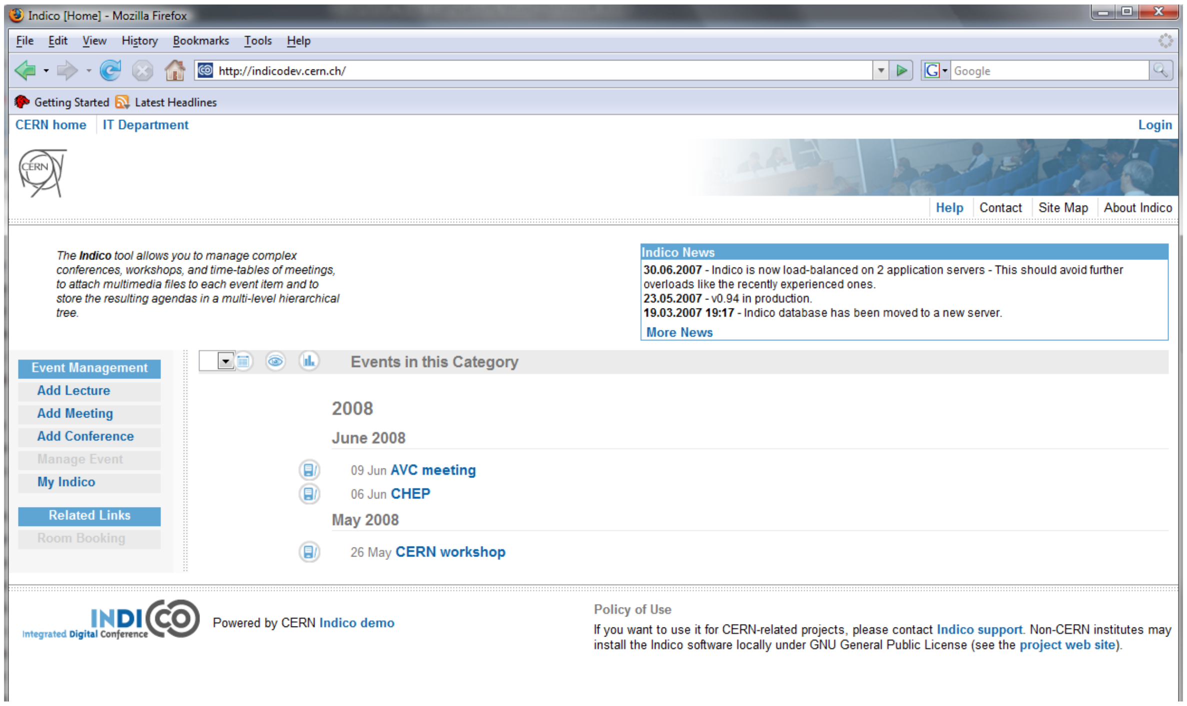
The goal of the project was to specify a “Unified Communications” module for InDiCo. The project started by performing a formal analysis of the user needs, with the identification of specific use cases and correspondent user requirements. Having thus obtained a clear characterisation of the problem, it was then possible to obtain a functional specification of the system and to select the appropriate technical approach for its implementation. In terms of implementation, a part of the module has been implemented that reflects one of the use cases described. The general outcome of this project is a tool that hopefully will make life of meeting organizers easier by gathering and automating the operations when setting up collaborative systems for remotely dispersed meetings. Due to its innovative approach for the architecture, it can be added with several functionalities very easily and rapidly without affecting the present ones. The module is well integrated in the existing InDiCo Web application.

Most of the time in this project has been dedicated to the analysis and architecture formalization work. This part it is seen as the most important of such a project, as others can always continue the implementation, provided that the design and documentation are clear and well done.

In a personal point of view, this project brings me a lot of benefits. At CERN, one is currently in a phase of creating a Collaborative Service that had already took several steps like refurbishing and standardizing the current CERN videoconferencing infrastructure or setting up a set of tools for monitoring and automating the most of systems within the service. These tasks are being followed by the current scope of this project. Due my current position, it is also very useful to get an in-depth knowledge about all the main collaborative technologies available in the market, the future trends and directions. It is very important as well to get a better and better knowledge of InDiCo framework (even not being the direct responsible), in order to optimize the synergies of integrating videoconferencing (more generally collaborative tools) and InDiCo. Another very positive aspect is the fact that since the working group where I’m on is purely operations (with a background research work to improve the support of the service operations), this project made me also have a more detailed view and understanding of how these operations can be improved or optimized. It gave me also the opportunity to rediscover Python as a very interesting language with a very different philosophy from the most common languages like C++ or Java. Working in a multinational and multilingual environment is also something that brings me a lot of benefits for all the years I have been at CERN being my understanding that it will continue to do it.

As always in the case of complex projects, this project is very far from being closed, in fact, it is just in the first steps. The major next step is to implement the next use cases defined in this document in order to get a broader spectrum of the performance of the application. Since the type of tasks of this project are not purely software but strongly rely also in expensive hardware like MCUs, Endpoints Terminals, PABXs, etc. one found some difficulties in having the collaborative systems relevant for this application available for testing and development within the defined timing. This can explain some of delays in the implementation phase.

Another challenge is InDiCo core development. The system has already lots of features and a quite complex architecture of files and classes. Some major changes are currently taking place, such as complete review of the user interface and internationalisation of the tool in several languages. Figure 5.1 shows the InDiCo front page with the new user interface (only available on InDiCo development version).



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.

Indico News

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.

2008

June 2008

09 Jun AVC meeting
06 Jun CHEP

May 2008

26 May CERN workshop

Powered by CERN Indico demo

Policy of Use

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 5.1 - New InDiCo interface, currently in development.

Even if it’s not an easy task to understand how InDiCo works in a first approach, is very important to keep the know-how up-to-date in order to leverage this acquired knowledge and be able to adapt the module according with the announced major changes. In what concerns the development technologies, one can always think that other platforms than Python could be more attractive for such architecture. In this project, this was not an issue due to the fact that InDiCO is fully implemented in Python and the main requirement is to integrate this module in the existing InDiCo architecture. So even if the overall architecture can be complicated to understand for future developers, it pays off as the overall system becomes a more efficient as an integrated solution, leveraging for instance the need of a middleware layer.

Another task that one can also easily foresee is a close monitoring of the module activity once in production and analyse how it can be optimized based on this feedback information in addition to the user feedback.

A second major and challenging step will be the addition of intelligence to the system in order for example to have the option of starting a given meeting automatically without user intervention or to actually allow the service to make decisions especially in what concerns the possible support of the 3G platform. This last point will imply a follow up study and clear understanding of how to specify the support of 3G Gateways in the module inside the existing architecture.

Finally and resuming all the work done up to now, it is expected that even with all the time and systems availability constraints, a pre-production version with the major use cases defined can be ready to be tested in the next following weeks/months. It’s also expected that this report can contribute as future reference documentation for the work still to be accomplished.

References

[1] InDiCo, available at http://indico.cern.ch

[2] EVO, available at http://evo.caltech.edu

[3] XML-RPC protocol, available at http://www.xmlrpc.com/

[4] ViDe, videoconferencing cookbook available at, http://www.vide.net/cookbook/cookbook.en/

[5] WebEx APIs Documentation Manuals, 2008

[6] Tandberg 3G Gateway Application Programmer Interfaces, available at http://www.tandberg.com/collateral/documentation/Application_Programmer_Interfaces/TANDBERG%203G%20Gateway%20API.pdf

[7] Tandberg/Codian 4200 series Management API, available at http://www.ivci.com/pdf/videoconferencing-codian-mcu-4200-management-api.pdf

[8] Tandberg MXP series APIs, available at http://www.tandberg.com/collateral/documentation/Application_Programmer_Interfaces/TANDBERG_MXP_API.pdf

[9] "Indico: the software behind CHEP 2004", J.Y. Le Meur, H. Sanchez, T. Baron, J. Gonzalez-Lopez, V. Turney, 2005

[10] Wainhouse Research, available at http://www.wainhouse.com

[11] “Programming Web Services with XML-RPC”, Simon St. Laurent, Joe Johnston, Edd Dumbill. O’Reilly, June 2001