ACE Deliverable A*1.3D1*

Report on VCE Architecture

Project Number: FP6-IST 508009
Project Title: Antenna Centre of Excellence
Document Type: Deliverable

Document Number: FP6-IST-508009-A1.3D1-2
Contractual date of delivery: 30 June 2004
Actual Date of Delivery: 18 July 2004
Workpackage: 1.3-1
Estimated Person Months: 3
Security (PP,PE,RE,CO): PU
Nature: Report
Version: 1.0
Total Number of Pages: 47
File name: ACE-A1.3D1-2.pdf
Editors: Fabio Giovacchini
Participants: IDS

Abstract

The purpose of this document is to provide the Architectural description of the Virtual Centre of Excellence of the ACE Network of Excellence.

The purpose of the VCE is to support the network Participants by "private Services" and, at the same time, to publish antenna research information to the scientific community.

In long term, this Virtual Centre aims to become the Internet hub for the antenna researchers and the related business.

In this marketing direction, it will be prepared to provide commercial service to generate income from industrial customers (by providing software, e-Learning, specific measurement information, etc.)

Keyword List

Antennas, Internet, Software, Virtual Centre of Excellence

Table of contents

1.Introduction

1.1. Purpose and Scope

The purpose of this document is to provide the Architectural description of the Virtual Centre of Excellence of the ACE Network of Excellence.

1.2. Reference

1.2.1. Referenced Documents

[RD1] ACE Contract

[RD2] ACE Contract Annex I – Technical Annex

1.2.2. Applicable documents

[AD1] IDS RQS/2004/001 ACE – Virtual Centre of Excellence Requirements

[AD2] IDS ADD/2004/020 ACE – Virtual Centre of Excellence Architecture

1.3. Acronyms and Definitions

1.3.1. Acronyms

CI Configuration Item
CSCI Computer Software Configuration Item

1.3.2. Definitions

Configuration Item or Computer Software Configuration Item:

Item designated for configuration management.

Configuration items can be of different complexity, size and type. The whole system is the top-level CI. The system composition in terms of Configuration Items shall be planned and agreed for each project.

General criteria for the selection of Configuration items are:

Reusability (units that can be used on other projects or in multiple roles on the project shall be designated as CI)

Developer (Items can be distinguished on the basis of the person/supplier/team who will develop the item (parallelisation of development activities)

A CI should be homogeneous in functionality, development language ...

Critical performance, cost, etc.

Maintenance and Testability

Susceptibility to change

2. SYSTEM OVERVIEW

The Antennas Virtual Centre of Excellence (Antennas VCE) is an Internet Application whose Graphical User Interface is rendered as a dedicated Portal for Antenna technology beginners and experts.

The context architecture of the VCE is reported in the following figure, where the basic blocks (modules) of the applications are reported; each block implements a service that is delivered to the user by a set of forms that are used to store and retrieve data from the underlying database.

Data exchange within the same module is implemented by argument passing, while interaction among different modules is implemented through communication with the database. File Sharing is not used. Files are only used by some modules to store information to be downloaded 'as-is' by remote users.

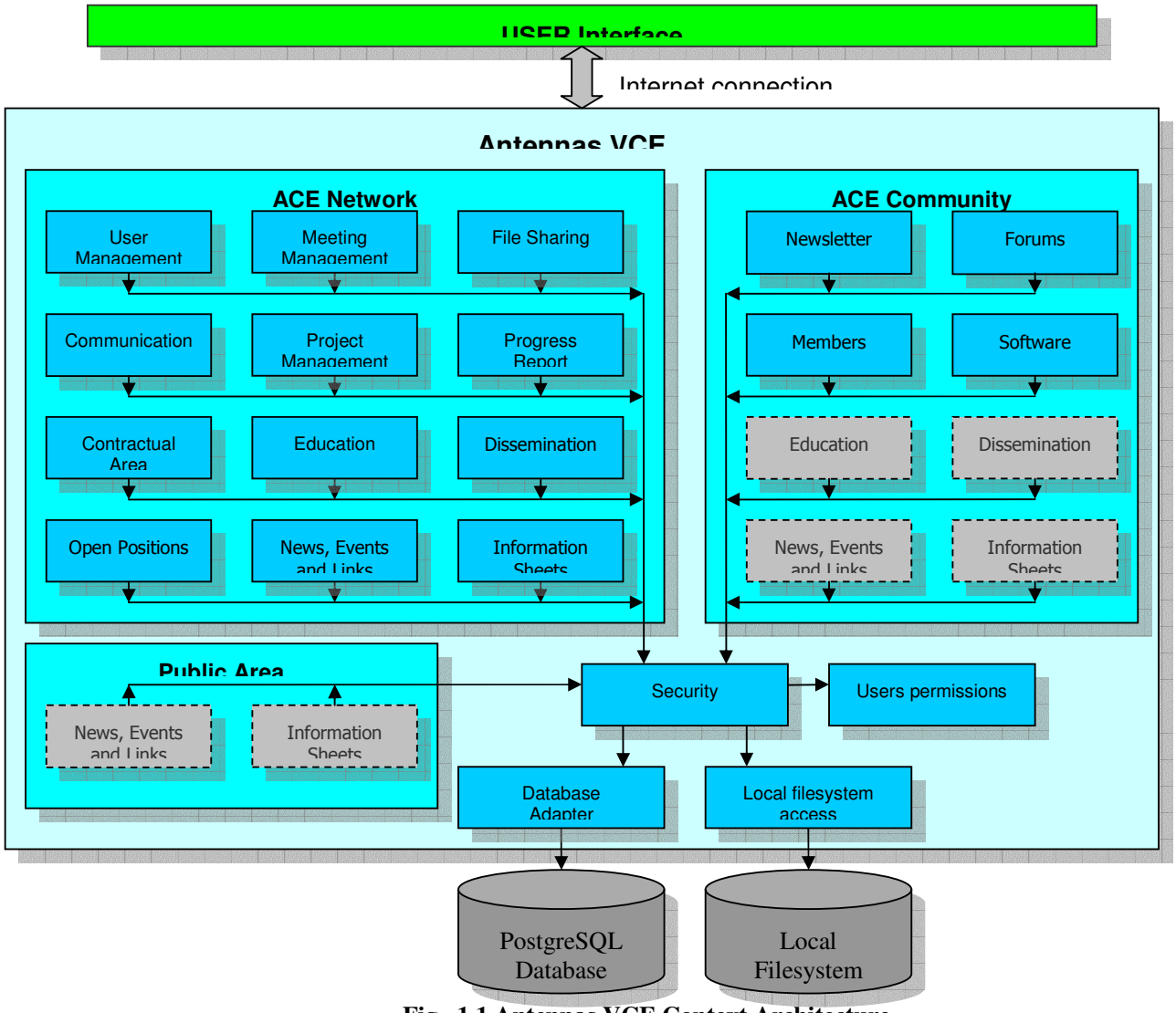

Fig. 1.1 Antennas VCE Context Architecture

At top level, the **Antennas VCE** is divided in three main areas:

- An **ACE Network** area, containing network restricted data, accessible only to ACE Network members using personal usernames and passwords

- An **ACE Community** area offering services to the Antenna Scientific Community, where organisations involved in Antenna Research activities can access by registering themselves with username and password for each person interested in participating the **ACE Community**.
- A **Public Area**, accessible without any restriction to anyone.

The **ACE Network** area aims at providing the necessary information technology infrastructure to support the needs of ACE Network members: communications facilities, meeting organisation, education and dissemination tools, etc.

The **Public Area** main goals are increasing the **Antennas VCE** visibility on the Internet and giving details about the benefits of joining **Antennas VCE**, as well as how to join.

2.1. ACE Network

The **ACE Network** is a top-level module in **Antennas VCE** system to support the activities of the ACE Network members.

For this purpose **ACE Network** provides the following services, implemented as CSCI:

- **User Management**
- **Security**
- **Meeting Management**
- **File Sharing**
- **Communication**
- **Project Management**
- **Progress Report**
- **Contractual Area**
- **Education**
- **Dissemination**
- **Open Positions**
- **News, Events and Links**
- **Information Sheets**

2.2. ACE Community

ACE Community is a top-level module in Antennas VCE system, designed to support the management of the Antenna Community.

ACE Community supplies a solution for the necessities of the administration of every function concerning researchers outside the ACE Network that would like to participate to an Antenna Scientific Community.

To support these requirements the module is composed by the following CSCI:

- **NewsLetter**
- **Forums**
- **Members**

- **Software**
- **Education**
- **Dissemination**
- **News, Events and Links**
- **Information Sheets**

2.3. Public Area

Public Area is a top-level module in Antennas VCE system, designed to enhance the visibility on the Internet of Antennas VCE system and to attract to ACE Community researchers and organisations involved in Antenna Research.

To support these requirements the module is composed by the following CSCI:

- **News, Events and Links**
- **Information Sheets**

3. SYSTEM BREAKDOWN

3.1. User Management

3.1.1. Type

This CSCI is a basic module in Antennas VCE system to manage the flow of input/output data from and to the final user.

3.1.2. Functions

The **User Management** module implements the user management functionalities, that is all operations that involve users of the Antennas VCE system or his role in the virtual community is performed through the User Management module:

- adding and deleting users;
- adding and deleting roles to the user;
- assign/change the organisation in which the user works;
- view the user information (identity, roles, organisations);
- show users list and user properties list using various filtering methods.

This is a basic module of the system because every other module relies on information created/updated in the database by the User Interface module.

3.1.3. Language

The **User Management** CSCI is implemented using **Python** and the **Zope DTML** languages.

3.1.4. Interfaces

The User Interface module relies on data provided by the Security module (except for his own data) as all others modules in the VCE system. Data provided by this module is available to all others modules.

There is no particular interface with other modules of the system; the only information exchange between **User Management** module and other modules is based on data contained in the **DBMS** that all modules share.

So to describe the interfaces to the other modules is necessary to describe the tables that are managed by the **User Management** module:

users table, used for store personal information about a VCE system's user:

- id_user, an integer that is used as primary key of table;
- name, a string that contains the first name of user;
- surname, a string that contains the last name of user;
- username, a string that contains the username used in the login procedure;
- password, a string that contains the password used in the login procedure;
- id_organisation, an integer that is used as a pointer to a row in the organisations table. It is used to know for what organisation the user works;
- departement, a string that contains, if it exists, the name of the organisation's department in which user works;

- address, a string that contains the user's work address, if it is distinct from that of the organisation;
- tel, a string that contains the user's work telephone number, if it is distinct from that of the organisation;
- fax, a string that contains the user's work fax number, if it is distinct from that of the organisation;
- mobile, a string that contains the user's work mobile telephone number;
- hide_mob, a boolean value that contains a TRUE value if the user's has made the choice to hide his mobile telephone number. In that case the user's mobile number will not be shown in his public information and will be used only by the system's SMS web facility;
- description, a string that can contains a description about the user;
- notes, a string that can contains some about the user;
- email, a string that contains the user's work email address;
- url, a string that contains the user's work homepage url;
- status, a string that contains the user's current status with regards of the VCE system. Actually the only status defined are “active” and “deleted”;

roles table, used for store the roles associated to an user:

- id_roles, an integer that is used as primary key of roles table;
- id_user, an integer that is used as a pointer to a row in the users table. It is used to know for what user this role apply;
- id_act_wp, an integer that is used as a pointer to a row in the act_wp table. It is used to know for what activity or work package this role apply;
- role, a string that contains the role that an user play on an activity/work package. The roles actually defined are “Participant User”, “Network Coordinator”, “Technical Coordinator”, “Executive User”, “Activity Leader”, “Workpackage Leader”, “Contact User”, “Contractual User” and “Researcher”;
- priority, an integer that actually is uses only to sort the user's roles in the user detail view. The association between role and priority is specified by the following table;

Role	*Priority*
Participant User	1
Network Coordinator	2
Technical Coordinator	3
Executive User	4
Activity Leader	5
Workpackage Leader	6
Contact User	7
Contractual User	8
Researcher	9

- status, a string that contains the role's current status with regards of the VCE system. Actually the only status defined are “active” and “deleted”;
- begin, a date-time value stating the creation time of the role;
- end, a date-time value stating the “deletion” time of the role. It is set only when the status field from “active” become “deleted”;

organizations table, used for store information about the organisations involved in the ACE consortium:

- id_organisation, an integer that is used as primary key of organisations table;
- name, a string that contains the full name of the organisation;
- short_name, a string that contains the "short" name of the organisation;
- legal_headquarte, a string that contains the address of the organisation's "legal headquarter";
- operative_headquarte, a string that contains the address of the organisation's "operative headquarter";
- id_logo, a string that contains the filename of the organisation's logo.

act_wp table, used for store information about the activities/work packages utilized by the VCE system:

- id_act_wp, an integer that is used as primary key of act_wp table;
- id_parent, an integer that is used as a pointer to a row in the act_wp table. If it has a zero value, this row represents an activity, otherwise this row is a work package and the id_parent value is used for knowing to which activity this work package belongs;
- acronym, a string that contains the activity/work package acronym;
- title, a string that contains the activity/work package full title;

3.1.4.1. Configuration Files

Not applicable.

3.1.4.2. Input from windows

Not applicable.

3.1.4.3. Input from file

Not applicable.

3.1.4.4. I/O from database

The **User Management** module, as previously stated, operates mainly using I/O on database. Every subroutine (see sec 3,1,6) of this module executes some operation on the DBMS; some modules make only a reading operation, others make only a writing operation while others make read/write operation.

The detail of every operation performed by the subroutine is contained in the following table:

Subroutine	Users table	Roles Table	Org. Table	Act_wp Table	SQL Query Used
add_act_leader			R	R	sql_select_organisations sql_select_activities
add_act_leader_role		R/W			sql_check_if_act_leader_from_uid sql_insert_act_leader_role
add_contact_wpkg		R/W			sql_check_if_contact_by_uid_awpid sql_insert_contact_wpkg
add_contractual_user_role		R/W			sql_check_if_contractual_user_from_uid sql_insert_contractual_user_role
add_executive_user_role		R/W			sql_check_if_executive_user_from_uid sql_insert_executive_user_role
add_participant_user			R		sql_select_organisations
add_participant_user_role		R/W			sql_check_if_participant_user_from_uid

Subroutine	Users table	Roles Table	Org. Table	Act_wp Table	SQL Query Used
					sql_insert_participant_user_role
add_researcher					
add_researcher_wpkg		R/W			sql_check_if_reserchер_by_uid_awpid sql_insert_researcher_wpkg
add_tech_coordinator_role		R/W			sql_check_if_tech_coordinator_from_uid sql_insert_tech_coordinator_role
add_user					
add_wpkg_leader_role		R/W			sql_check_if_wpkg_leader_from_uid sql_insert_wpkg_leader_role
del_role		W			sql_delete_role
del_user	W	W			sql_delete_user sql_delete_user_roles
index_html	R	R	R	R	sql_select_wpkgs sql_select_organisations sql_select_roles_and_org_by_uid_distinct sql_select_users_and_organisations sql_select_distinct_roles sql_select_roles_by_uid_distinct
insert_act_leader	W	W			sql_insert_user sql_insert_act_leader
insert_participant_user	W	W			sql_insert_user sql_insert_participant_user
insert_researcher	W	W			sql_insert_user sql_insert_researcher_last
insert_user	W				sql_insert_user
mod_user	R		R		sql_select_user_from_uid sql_select_organisations
mod_user_roles		R	R	R	sql_select_roles_from_uid sql_select_act_wp_from_id sql_select_roles_and_org_by_uid_distinct sql_select_wpkgs_of_act_by_awpid sql_select_wpkgs sql_select_activities
update_user	W				sql_update_user
users_list_html	R	R	R	R	sql_select_wpkgs sql_select_organisations sql_select_users_and_organisations sql_select_distinct_roles
view_org_detail			R		sql_select_org_detail_from_oid
view_user_detail	R				sql_select_user_detail_from_uid

3.1.4.5. Output on file

Not applicable.

3.1.4.6. Calls

The **User Management** module, as all others modules in the VCE system, calls the following routines from the Security module:

- check_auth();

- get_user();
- do_login();
- do_logout().

3.1.5. Routines

The **User Management** module creates and uses the following routines:

- add_act_leader, shows the form needed to add an user with the role of "Activity Leader" in the VCE system;
- add_act_leader_role, adds the "Activity Leader" role to a user already present in the VCE system;
- add_contact_wpkg, , adds the "Contact User" role to a user already present in the VCE system;
- add_contractual_user_role, adds the "Contractual User" role to a user already present in the VCE system;
- add_executive_user_role, adds the "Executive User" role to a user already present in the VCE system;
- add_participant_user, shows the form needed to add an user with the role of " Participant User " in the VCE system;
- add_participant_user_role, adds the "Participant User" role to a user already present in the VCE system;
- add_researcher, shows the form needed to add an user with the role of "Researcher" in the VCE system;
- add_researcher_wpkg, adds the "Researcher" role in the specified work package to a user already present in the VCE system;
- add_tech_coordinator_role, adds the "Technical Coordinator" role to a user already present in the VCE system;
- add_user, shows the form needed to add a generic user, i.e. an user without any predetermined role, in the VCE system;
- add_wpkg_leader_role, adds the "Workpackage Leader" role in the specified work package to a user already present in the VCE system;
- del_role, deletes a role (i.e. sets the role's "status" field to "deleted") from the roles table of the DBMS.
- del_user, first deletes a user (i.e. sets the user's "status" field to "deleted") from the users table, then deletes all roles (i.e. sets the role's "status" field to "deleted") of the specified user from the roles table.
- index_html, shows a web page containing a list of the users of the VCE system. This page also contains a form that allow various filtering methods on the effective users present in the list;
- insert_act_leader, first adds an user in the VCE system, then adds the "Activity Leader" role to the user just created;
- insert_participant_user, first adds an user in the VCE system, then adds the "Participant User" role to the user just created;
- insert_researcher, first adds an user in the VCE system, then adds the "Researcher" role to the user just created;
- insert_user, adds a generic user , i.e. an user without any predetermined role, in the VCE system;

- mod_user, shows the form needed to modify the user data;
- mod_user_roles, shows the form needed to modify the user's role in the VCE system;
- update_user, updates the user data in the users table of the DBMS;
- users_list_html, shows a web page containing a list of the users of the VCE system. This page also contains a form that allows various filtering methods on the effective users present in the list. Similar to "index_html" but with less options;
- view_org_detail, visualizes a web page that contains one seen detailed of the specific organisation;
- view_user_detail, visualizes a web page that contains one seen detailed of the specific user;

3.2. Security

3.2.1. Type

This CSCI is a basic module in VCE system that implements the access check and verify the user permission during the navigation.

3.2.2. Functions

The **Security** module implements the mapping user -> capabilities, i.e. It is used to determine if an operation performed by an user of the VCE system is permitted on to the base of his role in the virtual community.

It perform this functionality using data created e/o updated by the user interface module and stored in the DBMS system.

3.2.3. Language

The **Security** CSCI is implemented using **Python** and the **Zope DTML** languages.

3.2.4. Interfaces

The **Security** module, it puts to disposition of the other modules the following routines:

- check_auth();
- get_user();
- do_login();
- do_logout().

All others modules in the VCE system call, every time some of his subroutine is executed, at least the check_auth() routine from the Security module; the others routines of the **Security** module may be also called if needed.

3.2.4.1. Configuration Files

Not applicable.

3.2.4.2. Input from Windows

Not applicable.

3.2.4.3. Input from file

Not applicable.

3.2.4.4. I/O from database

The **Security** module, as previously stated, operates using I/O on database. The check_auth() and the get_user() subroutines of this module executes read only queries on the DBMS.

3.2.4.5. Output on file

Not applicable.

3.2.4.6. Calls

The **Security** module does not call subroutines from other modules; instead it accesses directly, using SQL queries, the data stored on DBMS by the user interface module.

3.2.5. Routines

The **Security** module defines the following routines:

- check_auth(), return a boolean value that tell if the user that is try to call a subroutine of some other module has the permission to perform the operation;
- get_user(), returns all information related to the user's roles within the VCE system. This routine was created to be used mainly by the Security module itself, but can be used by other modules too;
- do_login(), is used to set in the session the information about an user that has just logged in the VCE System;
- do_logout(), is used to remove from the session the information about the user's login in the VCE System.

3.3. Meeting Management

3.3.1. Type

This CSCI is a basic module in Antennas VCE system that support the Meeting announcement, registration, meeting-related files up/download (agenda, meeting minutes, etc.) and provide the meeting historic information.

3.3.2. Functions

The **Meeting Management** module supplies a tool for the needs of the Antennas VCE users to view and organise the information associated to the meetings of ACE Network of Excellence members.

To support these requirements the module allows the following operations:

- Workpackage Leaders, Activity Leaders, Technical Coordinator and Network Coordinator can create meetings;
- the owner (creator) of a meeting can modify it and can send an e-mail to all meeting participants;

- all users can view the meetings list and the detailed information of each meeting, including the meeting participants and the documents and meeting minutes related to a meeting;
- all users can register to / unregister from a meeting;

3.3.3. Language

The **Meeting Management** CSCI is implemented using **Python** and the **Zope DTML** languages.

3.3.4. Interfaces

There is no particular interface with other modules of the system; the only information exchange between **Meeting Management** module and other modules is based on data contained in the **DBMS** that all modules share.

So to describe the interfaces to the other modules is necessary to describe the tables that are managed by the **Meeting Management** module:

meeting table, used for store information about meetings of the VCE system:

- id_meeting, the table primary key;
- meeting_title, a string containing the meeting titel;
- date_start, a DATE type containing the meeting first day;
- time_start, a TIME type containing the meeting start hour and minutes;
- date_end, a DATE type containing the meeting last day;
- time_end, a TIME type containing the meeting end hour and minutes;
- meeting_desc, a string containing the meeting description;
- address, a string containing the address of the meeting location;
- city, a string containing the town name where the meeting will happen;
- country, a string containing the country where the meeting will happen;
- id_user, an integer referring to the user that created the meeting. This is also the meeting owner;
- url, a string that contains the meeting's homepage web page, if any;

meeting_partic table, used to store users' registration to meetings:

- id_meeting_partic, the table primary key;
- id_meeting, an integer referring to the **meeting** table;
- id_user, an integer referring to the **users** table;

meeting_files table, used to store files related to meetings:

- id_file, the table primary key;
- meeting, an integer referring to the **meeting** table;
- filename, a string containing the file name;
- title, a string containing the file title;

3.3.4.1. Configuration Files

Not applicable.

3.3.4.2. Input from windows

Not applicable.

3.3.4.3. Input from file

Not applicable.

3.3.4.4. I/O from database

The **Meeting Management** module, as previously stated, operates mainly using I/O on database. Every subroutine of this module executes some operation on the DBMS; some modules only perform read operations, others only perform write operations while others perform both read and write operations.

3.3.4.5. Output on file

Not applicable.

3.3.4.6. Calls

The **Meeting Management** module, as all others modules in the VCE system, calls the following routine from the Security module:

- AUTHENTICATED_USER.getRoles()

3.3.5. Routines

As it happens in the entire Antennas VCE portal, in the **Meeting Management** module there is a one-to-one mapping between Zope folders where routines are stored, and web pages.

The **Meeting Management** module defines and uses the routines in the following folders:

- /ACE/Meeting/view: DTML methods to show the list of meetings and to access all other module functionalities (insert, modify, register, unregister, upload file, download file);
- /ACE/Meeting/view/detail: DTML methods to view the details about a meeting, to insert view and download files related to a meeting;
- /ACE/Meeting/view/participants: DTML methods to view the participants to a meeting;
- /ACE/Meeting/insert: DTML methods to create a new meeting;
- /ACE/Meeting/modify: DTML methos to modify an existing meeting;
- /ACE/Meeting/sendmail: DTML methods to send an e-mail to all meeting participants;

3.4. File Sharing

3.4.1. Type

This CSCI is a module in VCE system that is used to exchange files among the Network researchers.

3.4.2. Functions

The **File Sharing** module supplies one solution for the needs of storage of documents and any other file utilized by VCE's users, and for the exchange of such file between VCE's users and/or external users.

To support these requirements the module allows the following operations:

- adding file and folder;

- deleting and renaming files and folders;
- cut, copy and paste files and folders;
- browse the folders contents;
- download the file.

3.4.3. Language

The **File Sharing** CSCI is implemented using **Python** and the **Zope DTML** languages.

3.4.4. Interfaces

There is no particular interface with other modules of the system; the only information exchange between **File Sharing** module and other modules is based on data contained in the **DBMS** that all modules share.

So to describe the interfaces to the other modules is necessary to describe the tables that are managed by the **File Sharing** module:

files table, used for store information about files of the VCE system:

- id_file, an integer that is used as primary key of table;
- parent, an integer that is used as a pointer to a row in the **folders** table. It is used to know which folder current file belongs to;
- name, a string that contains the name of the file;

folders table, used for store information about folders of the VCE system:

- id_folder, an integer that is used as primary key of table;
- parent, an integer that is used as a pointer to a row in the **folder** table. It is used to know which folder current folder belongs to;
- name, a string that contains the name of the folder;

3.4.4.1. Configuration Files

Not applicable.

3.4.4.2. Input from windows

Not applicable.

3.4.4.3. Input from file

Not applicable.

3.4.4.4. I/O from database

The **File Sharing** module, as previously stated, operates mainly using I/O on database. Every subroutine of this module executes some operation on the DBMS; some modules make only a reading operation, others make only a writing operation while others make read/write operation.

3.4.4.5. Output on file

Not applicable.

3.4.4.6. Calls

The **File Sharing** module, as all others modules in the VCE system, calls the following routines from the Security module:

- check_auth();
- get_user();

3.4.5. Routines

The **File Sharing** module defines and uses the following routines:

- index_html, shows a web page containing a list of the files contained in the file-system repository. This page also contains a toolbar with actions icon;
- createFolderForm, shows the form needed to add a new folder;
- uploadFileForm, shows the form needed to add a new file;
- renameForm, show the form needed to rename selected items;
- createFolder, insert a new row in to folders table;
- upload, insert a new row in to files table and save file contents in to file-system repository;
- doCmd, copy, cut, paste, delete or rename folders and files updating DBMS and file-system repository;

3.5. Communication

3.5.1. Type

This CSCI is a basic module in VCE system that allow an efficient communication in the ACE Network by providing e-mailing lists, SMS bridges, instant messaging, etc.

3.5.2. Functions

The **Communication** module supplies one solution for the needs of communication between the VCE's users using various media, such as forum, mailing list and SMS facility.

To support these requirements the module allows the following operations:

- adding and deleting forum on a per activity/work package basis;
- adding and deleting user's messages to a forum;
- assign/change the information related to the forum (description, deadline, advancement, contributors, etc..);
- view the forums list and the detailed information of each forum;
- create a mailing list of VCE's users on a per activity/work package/role basis;
- create a list of VCE's users using a forum using various information filtering methods;
- send email and/or SMS to a list of VCE's users on a per activity/work package/role basis;

3.5.3. Language

The **Communication** CSCI is implemented using **Python** and the **Zope DTML** languages.

3.5.4. Interfaces

There is no particular interface with other modules of the system; the only information exchange between **Communication** module and other modules is based on data contained in the **DBMS** that all modules share.

So to describe the interfaces to the other modules is necessary to describe the tables that are managed by the **Communication** module:

forums table, used for store information about forums of the VCE system:

- id_forum, an integer that is used as primary key of table;
- id_act_wp, an integer that is used as a pointer to a row in the act_wp table. It is used for knowing to which activity/work package the forum belongs;
- name, a string that contains the name of the forum;
- description, a string that can contains a description about the forum;

messages table, used for store information about user's message to a forum:

- id_message, an integer that is used as primary key of table;
- id_forum, an integer that is used as a pointer to a row in the forums table;
- id_reference, an integer that is used to select a message in this table for which this message is a reply;
- id_user, an integer that is used as a pointer to a row in the users table;
- message, a string that is used to store the user's message to the forum;
- title, a string that is used to store the title of the message;

3.5.4.1. Configuration Files

Not applicable.

3.5.4.2. Input from windows

Not applicable.

3.5.4.3. Input from file

Not applicable.

3.5.4.4. I/O from database

The **Communication** module, as previously stated, operates mainly using I/O on database. Every subroutine of this module executes some operation on the DBMS; some modules make only a reading operation, others make only a writing operation while others make read/write operation.

3.5.4.5. Output on file

Not applicable.

3.5.4.6. Calls

The **Communication** module, as all others modules in the VCE system, calls the following routines from the Security module:

- check_auth();

get_user().

3.5.5. Routines

The **Communication** defines and uses the following routines:

- add_forum, shows the form needed to add a new forum;
- add_message, shows the form needed to add an user's message to a specified forum;
- compose_mail, shows the form needed to compose an email to be sent to a specified user/users/work package/activity;
- send_mail, send an email to a specified user/users/work package/activity;
- compose_sms, shows the form needed to compose an SMS to be sent to a specified user/users/work package/activity;
- send_sms, send an SMS to a specified user/users/work package/activity;
- mailinglist, shows a web page containing a list of email address of ACE's users. This page also contains a form that allow various filtering methods on the effective forums present in the list;
- del_forum, deletes a forum (i.e. sets the forum's "status" field to "deleted") from the forums table of the DBMS.
- del_message, deletes a user's message from the messages table of the DBMS.
- index_html, shows a web page containing a list of the forums previewed to the inside of the ACE. This page also contains a form that allow various filtering methods on the effective forums present in the list;
- insert_forum, adds a forum into the forums table of the DBMS;
- insert_message, adds an user's message to a specified forum in messages table of the DBMS;
- mod_forum, shows the form needed to modify the forum's data;
- update_forum, updates the forum data in the forums table of the DBMS;
- messages_html, shows a web page containing a list of the forums in which the user is actually working;
- view_forum_detail, visualizes a web page that contains one seen detailed of the specific forum;

3.6. Project Management

3.6.1. Type

This CSCI is a module in Antennas VCE system used to declare the efforts provided in the research activities and the foreseen work till the end of the project.

3.6.2. Functions

The **Project Management** module supplies a tool for the needs of the VCE users to control the information associated to the advancement of the ACE's projects.

To support these requirements the module allows the following operations:

- manage hours "worked so far" and "estimated to the end" per work package;
- view reports on project's time table, divided by participant or user;

3.6.3. Language

The **Project Management** CSCI is implemented using **Python** and the **Zope DTML** languages.

3.6.4. Interfaces

There is no particular interface with other modules of the system; the only information exchange between **Project Management** module and other modules is based on data contained in the **DBMS** that all modules share.

So to describe the interfaces to the other modules is necessary to describe the tables that are managed by the **Project Management** module:

act_wp table, used for store information about wosk package/activity gerarchy:

- id_act_wp, unique identifier
- id_parent, parent work package or activity
- acronym

roles table, used for store information about role of user in a work package/activity

- id_role, unique identifier
- id_user, user identifier
- id_act_wp, activity/work package identifier
- role, role of user (id_user) in activity/worck package (id_act_wp)
- priority, used for revision of roles
- status, status of the role

project_management table, used for store information about hours "worked so far" and "remaining to work" for each researcher and work package.

- id_user, identifier of user related to table **users**
- id_act_wp, identifier of activity/worck package
- year
- quarter
- estimated, estimated worked hours up to the end of the project for the user (id_user) in the work package/activity (id_act_wp)
- worked, worked hours so far of the project for the user (id_user) in the work package/activity (id_act_wp)

project_proposal_time table, used for store information about initial estimation for each organisation

- id_organization, organization identifier
- id_act_wp, activity/worck package identifier
- month, initial estimation in month

project_ac_time table, as project_management table, but used only for additional cost model (AC).

3.6.4.1. Configuration Files

Not applicable.

3.6.4.2. Input from windows

Not applicable.

3.6.4.3. Input from file

Not applicable.

3.6.4.4. I/O from database

The **Project Management** module, as previously stated, operates mainly using I/O on database. Every subroutine (see sec. 3.3.6) of this module executes some operation on the DBMS; some modules make only a reading operation, others make only a writing operation while others make read/write operation.

3.6.4.5. Output on file

Not applicable.

3.6.4.6. Calls

The **Project Management** module, as all others modules in the VCE system, calls the following routines from the Security module:

- AUTHENTICATED_USER.getRoles ().

3.6.5. Routines

The **Project Management** defines and uses the following routines:

- CommissionSummary, show summary for commission;
- Edit, partecipant user input of hours for him user;
- PartecipantSummary, show summary for commission.
- View, show summary for work package/activity leader.
- WPSummary. Show report for work package/activity leader.
- box_right, define html interfaces.

3.7. Progress Report

3.7.1. Type

This CSCI is a basic module in VCE system that is used to assemble the Periodic Reporting to the Commission.

3.7.2. Functions

The **Progress Report** module supplies one solution for the cooperation in the management of report documents produced from the VCE's users.

To support these requirements the module allows the following operations:

- adding, editing and deleting user's contributions to a report;
- uploading documents to append at the end of the report.
- checking the document status by the HTML preview of the document or by the status list any single contribution.

- making a printable PDF document.

3.7.3. Language

The **Progress Report** CSCI is implemented using **Python** and the **Zope DTML** languages.

3.7.4. Interfaces

There is no particular interface with other modules of the system; the only information exchange between **Progress Report** module and other modules is based on data contained in the **DBMS** that all modules share.

So to describe the interfaces to the other modules is necessary to describe the tables that are managed by the **Progress Report** module:

progress_reports table, used for store information about user's contribution to a report:

- id_role, an integer that is used as a pointer to a row in the roles table.
- year, an integer that is used to specify the year of the document.
- (only one document in year)
- section, a string that is used to store the identifier of the paragraph that this contribution represents;
- status, a string that contains the report's current status. Actually the only status defined are “todo”, “working” and “terminated”;
- data, a string that is used to store the user's contribution to the report;

3.7.4.1. Configuration Files

Not applicable.

3.7.4.2. Input from windows

Not applicable.

3.7.4.3. Input from file

Not applicable.

3.7.4.4. I/O from database

The **Progress Report** module, as previously stated, operates mainly using I/O on database. Every subroutine (see sec. 3.3.6) of this module executes some operation on the DBMS; some modules make only a reading operation, others make only a writing operation while others make read/write operation.

3.7.4.5. Output on file

Not applicable.

3.7.4.6. Calls

The **Progress Report** module, as all others modules in the VCE system, calls the following routines from the Security module:

- getWpkgDetail();
- getUserId().
- getOrgId().

- getOrgShortName().
- getRole().

3.7.5. Routines

The **Progress Report** defines and uses the following routines:

- add_contribution, shows the form needed to add an user's contribution to a specified report;
- mod_contribution, shows the form needed to update an user's contribution;
- index_html, shows a web page containing a list of the contributions in which the user is actually working;
- insert_contribution, adds an user's contribution to a specified report in progress_report table of the DBMS;
- update_contribution, update an user's contribution in progress_report table of the DBMS;
- upload_annex, upload a pdf document to append at the end of a specified report;
- preview, show an HTML preview of a specified report;
- build_pdf, build a PDF document of a specified report;
- check_status, show a webpage containing the list of "incomplete" user's contribution to a specified report;

3.8. Contractual Area

3.8.1. Type

This CSCI is a basic module in Antennas VCE system supporting the contractual and administrative needs of ACE Network.

3.8.2. Functions

The **Contractual Area** module provides the necessary infrastructure to view and store up-to-date contractual communications, contractual documents and amendments, provide cost declaration, keep trat of contractual statement shipment status and receive payments notifications. Only Participant Users and Contractual Users can access the **Contractual Area** module.

The **Contractual Area** module is divided in several sections:

- **Communications**, where communications relevant to Contractual Users can be viewed. Contractual Users belonging to IDS can also add, edit and delete communications.
- **Documents**, where contractual documents and amendments can be downloaded. Contractual Users belonging to IDS can also add, and delete contractual amendments.
- **Cost Declaration**, where each organisation can enter its yearly cost summary. Contractual Users belonging to IDS can also view cost summaries entered by all other organisations.
- **Statement Shipment**, where each organisation can notify the shipment to IDS of the cost statement and can monitor its progress as IDS receives and reviews it.
- **Payments**, where each organisation can view the payment notifications entered by IDS.

3.8.3. Language

The **Contractual Area** module CSCI is implemented using **Python** and the **Zope DTML** languages.

3.8.4. Interfaces

There is no particular interface with other modules of the system; the only information exchange between **Contractual Area** module and other modules is based on data contained in the **DBMS** that all modules share.

So to describe the interfaces to the other modules is necessary to describe the tables that are managed by the **Education** module:

financial_news, used to store contractual communications:

- id_news, the table primary key;
- date, a DATE type with the communication date;
- title, a string containing the communication tittle;
- text, a string containing the communication detail;
- permission, an integer specifying the organisation that can view this communication (0 means all organisations);

financial_costs, used to store yearly cost declarations:

- id_cost, the table primary key;
- organization, an integer referring to the organisation that declared this yearly cost;
- year, the year this cost declaration refers to;
- pmonth_budget, pmonth_period1, pmonth_period2, the initial person-month budget and the person-month efforrts in 2004 and 2005;
- costp_budget, costp_period1, costp_period2, the initial personnel budget and the personnel costs in 2004 and 2005;
- costx_budget, costx_period1, costx_period2, the initial travel budget and the travel costs in 2004 and 2005;
- costy_budget, costy_period1, costy_period2, the initial equipment budget and the equiment expenditures in 2004 and 2005;
- costo_budget, costo_period1, costo_period2, the initial budget for other costs and the other costs in 2004 and 2005;

financial_shipments, used to store cost statement shipments:

- id_shipment, the table primary key;
- organization, an integer referring to the organisation that sent the cost statement;
- year, the financial year the shipment refers to. It is also a foreign key referring to the **financial_shipments_years** table (see below);
- date, a DATE type containing the shipment date;
- state, a 1-character string containing the shipment status: ‘S’ sent, ‘R’ received, ‘V’ valid, ‘N’ not valid;
- courier, a string containing the name of the courier used to sent the shipment;

- track_no, a string containing the tracking number used by the courier to uniquely identify the shipment;

financial_shipments_years, used to store the list of financial shipments years:

- year, the financial shipment year and also the table primary key;

financial_payments, used to store payments from IDS to ACE Network organisations:

- id_payment, the table primary key;
- organization, an integer referring to the organisation that received this payment;
- date, a DATE type containing the payment day, month and year.
- bank_coord, a string containing the exact banking coordinates the payment was sent to;
- reason, a string containing the exact reason for payment used in the bank transaction;
- amount, a float containing the amount payed, in euro;

3.8.4.1. Configuration Files

Not applicable.

3.8.4.2. Input from windows

Not applicable.

3.8.4.3. Input from file

Not applicable.

3.8.4.4. I/O from database

The **Contractual Area** module, as previously stated, operates mainly using I/O on database. Every routine of this module executes some operation on the DBMS; most routines only perform read operations, the routines only accessible to IDS also perform read/write operations.

3.8.4.5. Output on file

Not applicable.

3.8.4.6. Calls

The **Contractual Area** module, as all others modules in the VCE system, calls the following routines from the Security module:

- AUTHENTICATED_USER.getRoles();

3.8.5. Routines

As it happens in the entire Antennas VCE portal, in the **Contractual Area** module there is a one-to-one mapping between Zope folders where routines are stored, and web pages.

The **Contractual Area** module defines and uses the routines in the following folders:

1. General

- /ACE/Contractual: DTML methods to show and navigate in the list of sections in the Contractual module (**Communications, Documents, Cost Declaration, Statement Shipment**, **Payments**);

2. Documents
   - /ACE/Contractual/Documents: DTML methods to list and download contractual documents and amendments. Also allows IDS users to upload new amendments;
3. Communications
   - /ACE/Contractual/Communications: DTML methods for the **Communications** section, showing the list of visible communications. DTML methods to delete a communication (accessible to IDS only);
   - /ACE/Contractual/Communications/detail: DTML methods to show the detailed contents of a visible communication;
   - /ACE/Contractual/Communications/insert: DTML methods to create a new communication (accessible to IDS only);
   - /ACE/Contractual/Communications/modify: DTML methods to modify an existing communication (accessible to IDS only);
4. Cost Declaration
   - /ACE/Contractual/Cost_Declaration: DTML methods to show and edit the cost declaration of the user's organization;
   - /ACE/Contractual/Cost_Declaration/view: DTML methods to show the list of all organisations' cost declarations (accessible to IDS only):
5. Statement Shipment
   - /ACE/Contractual/Shipments: DTML methods to show the list shipments performed by the user's organisation. For IDS, shows the last shipment performed by each organisation and allows to create new shipments years;
   - /ACE/Contractual/Shipments/insert: DTML methods to insert a new shipment for the user's organisation. Only accessible if there is no previous shipment or last shipment was determined by IDS not to be valid.
   - /ACE/Contractual/Shipments/detail: DTML methods to show the shipments performed by an organisation and update their status from the following list: Sent, Received, Valid, Not Valid;
6. Payments
   - /ACE/Contractual/Payments: DTML methods to show the payments from IDS to the user's organisation. For IDS, shows the last payment to each organisation;
   - /ACE/Contractual/Payments/detail: DTML methods to show the payments to an organisation (accessible to IDS only);
   - /ACE/Contractual/Payments/insert: DTML methods to insert a new payment to an organisation (accessible to IDS only);

3.9. Education

3.9.1. Type

This CSCI is a basic module in Antennas VCE system supporting the education activities in progress in the ACE Network.

3.9.2. Functions

The **Education** module provides the necessary infrastructure to allow publishing on the Internet structured and categorised information (courses, lectures, etc.) for electronic learning purposes in the electromagnetism and antennas technologies area. The permission to view and download each material can be independently granted or denied to the following categories of users:

- ACE Community members and guests;
- ACE Network members.

The **Education** module also provides access to the **European Shool of Antennas** and **Virtual Laboratory (VALab)** external services.

The **E-Learning** section stores files on the local file-system. Files are read and written on the local filesystem on 'as-is'basis and their content is not relevant for the **Education** module.

The **E-Learning** section of this module is divided in two areas:

- a **view** area, where all users can navigate through the information visible to them. Information is presented as a tree of folders and files. The **view** area is accessible from both the ACE Community and the ACE Network sections of Antennas VCE portal.
- an **admin** area, where certain ACE Network members (all ACE Executive Board members plus designated users belonging to Virtual Centre Of Excellence Activity) can add, delete and update the available information and its visibility.

The **admin** area offers the following functionalities:

- available information is presented as a tree of folders and files, allowing easy navigation. All other functionalities can be accessed by clicking on icons, buttons and folder or file names;
- folders can be added, edited and deleted. Their visibility can be viewed and changed. Each folder can contain a description;
- files can be added and deleted. Files inherit visibility settings from their parent folder.

3.9.3. Language

The **Education** module CSCI is implemented using **Python** and the **Zope DTML** languages.

3.9.4. Interfaces

There is no particular interface with other modules of the system; the only information exchange between **Education** module and other modules is based on data contained in the **DBMS** that all modules share.

So to describe the interfaces to the other modules is necessary to describe the tables that are managed by the **Education** module:

elearning_files table, used for store information about files:

- id_file, the table primary key;
- parent, an integer that identifies the folder where this file resides;
- filename, a string that can contains the name of the file

elearning_items table, used for store information about folders:

- id_item, the table primary key;
- visibility, an integer containing a bitmask of the visibility flags (2=ACE Community, 4=ACE Network);
- deleted, a 1-character string that can contains '1' if the folder was deleted, '0' otherwise;

- parent, an integer that identifies the parent of this folder;
- title, a string that contains the folder name;
- date, a DATE type that represents a relevant date for the folder (typically the creation date);
- text, a string that contains the folder description.

3.9.4.1. Configuration Files

Not applicable.

3.9.4.2. Input from windows

Not applicable.

3.9.4.3. Input from file

Not applicable.

3.9.4.4. I/O from database

The **Education** module, as previously stated, operates mainly using I/O on database. Every routine of this module executes some operation on the DBMS; the **view** section of the modules only performs read operations, the **admin** section performs read/write operations.

3.9.4.5. Output on file

Not applicable.

3.9.4.6. Calls

The **Education** module, as all others modules in the VCE system, calls the following routines from the Security module:

- AUTHENTICATED_USER.getRoles();

3.9.5. Routines

As it happens in the entire Antennas VCE portal, in the **Education** module there is a one-to-one mapping between Zope folders where routines are stored, and web pages.

The **Education** module defines and uses the routines in the following folders:

- /ACE/Education: DTML methods for the Education module top-level template page Redirects to /ACE/Education/E_Learning/view.
- /ACE/Education/E_Learning: redirects to /ACE/Education/E_Learning/view.
- /ACE/Education/E_Learning/view: DTML methods to navigate through the visible files and folders. Allows downloading visible files.
- /ACE/Education/E_Learning/admin: DTML methods to navigate through all e-learning files and folders. Gives access to the ‘insert folder’ and ‘modify folder’ pages, allows uploading files and deleting files and folders.
- /ACE/Education/E_Learning/admin/insert: DTML methods to create a new folder.
- /ACE/Education/E_Learning/admin/modify: DTML methods to edit an existing folder.

3.10. Dissemination

3.10.1. Type

This CSCI is a basic module in Antennas VCE system supporting the dissemination activities in progress in the ACE Network.

3.10.2. Functions

The **Dissemination** module provides the necessary infrastructure to allow publishing on the Internet structured and categorised information (conferences, workshops, books and journals, liaisons, etc.) for dissemination purposes in the electromagnetism and antennas technologies area. The permission to view and download each material can be independently granted or denied to the following categories of users:

- ACE Community members and guests;
- ACE Network members.

The **Dissemination** module stores files on the local file-system. Files are read and written on the local filesystem on 'as-is'basis and their content is not relevant for the **Dissemination** module.

The **Dissemination** module is divided in two areas:

- a **view** area, where all users can navigate through the information visible to them. Information is presented as a tree of folders and files. The **view** area is accessible from both the ACE Community and the ACE Network sections of Antennas VCE portal.
- an **admin** area, where certain ACE Network members (all ACE Executive Board members plus designated users belonging to Virtual Centre Of Excellence Activity) can add, delete and update the available information and its visibility.

The **admin** area offers the following functionalities:

- available information is presented as a tree of folders and files, allowing easy navigation. All other functionalities can be accessed by clicking on icons, buttons and folder or file names;
- folders can be added, edited and deleted. Their visibility can be viewed and changed. Each folder can contain a description;
- files can be added and deleted. Files inherit visibility settings from their parent folder.

The folders managed by the **Dissemination** module can be of one out of three types:

1. level-1 folders: 'categories'. They are the top level folders.
2. level-2 folders: 'topics'. They are contained in 'categories'.
3. level-3 folders: 'items'. They are contained in 'topics'. Only level-3 folders can contain files.

3.10.3. Language

The **Dissemination** module CSCI is implemented using **Python** and the **Zope DTML** languages.

3.10.4. Interfaces

There is no particular interface with other modules of the system; the only information exchange between **Dissemination** module and other modules is based on data contained in the **DBMS** that all modules share.

So to describe the interfaces to the other modules is necessary to describe the tables that are managed by the **Dissemination** module:

dissemin_files table, used for store information about files:

- id_file, the table primary key;
- item, an integer that identifies the folder (item actually) where this file resides;
- deleted, a 1-character string that can contains '1' if the file was deleted, '0' otherwise;
- filename, a string that can contains the name of the file

dissemin_items table, used for store information about level-3 folders:

- id_item, the table primary key;
- visibility, an integer containing a bitmask of the visibility flags (2=ACE Community, 4=ACE Network);
- deleted, a 1-character string that can contains '1' if the folder was deleted, '0' otherwise;
- date, a DATE type that represents a relevant date for the folder (typically the creation date);
- title, a string that contains the folder name;
- topic, an integer that identifies the parent (a level-2 folder) of this folder;
- text, a string that contains the folder description.

dissemin_topics table, used for store information about level-2 folders:

- id_topic, the table primary key;
- visibility, an integer containing a bitmask of the visibility flags (2=ACE Community, 4=ACE Network);
- deleted, a 1-character string that can contains '1' if the folder was deleted, '0' otherwise;
- date, a DATE type that represents a relevant date for the folder (typically the creation date);
- title, a string that contains the folder name;
- category, an integer that identifies the parent (a level-1 folder) of this folder;
- text, a string that contains the folder description.

dissemin_categ table, used for store information about level-1 folders:

- id_category, the table primary key;
- visibility, an integer containing a bitmask of the visibility flags (2=ACE Community, 4=ACE Network);
- deleted, a 1-character string that can contains '1' if the folder was deleted, '0' otherwise;
- title, a string that contains the folder name;
- text, a string that contains the folder description.

3.10.4.1. Configuration Files

Not applicable.

3.10.4.2. Input from windows

Not applicable.

3.10.4.3. Input from file

Not applicable.

3.10.4.4. I/O from database

The **Dissemination** module, as previously stated, operates mainly using I/O on database. Every routine of this module executes some operation on the DBMS; the **view** section of the modules only performs read operations, the **admin** section performs read/write operations.

3.10.4.5. Output on file

Not applicable.

3.10.4.6. Calls

The **Dissemination** module, as all others modules in the VCE system, calls the following routines from the **Security** module:

- AUTHENTICATED_USER.getRoles();

3.10.5. Routines

As it happens in the entire Antennas VCE portal, in the **Dissemination** module there is a one-to-one mapping between Zope folders where routines are stored, and web pages.
The **Dissemination** module defines and uses the routines in the following folders:

- /ACE/Dissemination: DTML methods with the Dissemination modules top-level templates. Redirects to /ACE/Dissemination/view.
- /ACE/Dissemination/view: DTML methods showing the contents of the current Dissemination folder.
- /ACE/Dissemination/admin: DTML methods containing the templates for the Dissemination **admin** section.
- /ACE/Dissemination/admin/Attachments: DTML methods showing the files contained in the current level-3 folder.
- /ACE/Dissemination/admin/Items: DTML showing the level-3 folders (items) contained in the current level-2 folder.
- /ACE/Dissemination/admin/Items/insert: DTML methods to create a new level-3 folder (item).
- /ACE/Dissemination/admin/Items/modify: DTML methods to edit an existing level-3 folder (item).
- /ACE/Dissemination/admin/Topics: DTML showing the level-2 folders (topics) contained in the current level-1 folder.
- /ACE/Dissemination/admin/Topics/insert: DTML methods to create a new level-2 folder (topic).
- /ACE/Dissemination/admin/Topics/modify: DTML methods to edit an existing level-2 folder (topic).
- /ACE/Dissemination/admin/Categories: DTML showing the level-1 folders (categories).
- /ACE/Dissemination/admin/Categories/insert: DTML methods to create a new level-1 folder (category).
- /ACE/Dissemination/admin/Categories/modify: DTML methods to edit an existing level-1 folder (category).

3.11. Open Positions

3.11.1. Type

This CSCI is a basic module in Antennas VCE system supporting the publishing of open positions relevant to ACE Network members.

3.11.2. Functions

The **Open Positions** module provides the necessary infrastructure to expose to ACE Network members information about the available job and research opportunities that may be relevant to ACE Network members.

The **Open Positions** module is divided in two areas:

- a **view** area, where all users can navigate through the available information.
- an **admin** area, where certain ACE Network members (all ACE Executive Board members plus designated users belonging to Virtual Centre Of Excellence Activity) can add, delete and update the available open positions.

3.11.3. Language

The **Open Positions** module CSCI is implemented using **Python** and the **Zope DTML** languages.

3.11.4. Interfaces

There is no particular interface with other modules of the system; the only information exchange between **Open Positions** module and other modules is based on data contained in the **DBMS** that all modules share.

So to describe the interfaces to the other modules is necessary to describe the tables that are managed by the **Open Positions** module:

open_positions table, used for store information about open positions:

- id_position, the table primary key;
- deleted, a 1-character string: ‘0’ if open position is visible, ‘1’ if deleted;
- permission, an integer specifying the visibility of this open position (currently unused);
- title, a string containing the open position’s title;
- id_organization, an integer referring to the organisation offering the position, or 0 if the organisation is outside ACE Network;
- name_organization, (for organisations not in ACE Network) a string containing the organisation name;
- start_date, a DATE type with the open position beginning date;
- requirements, a string describing required titles and skills for the open positions;
- position_role, a string containing the role that will be assumed by the person accepting the open position;
- contact_name, a string with the contact person’s first name;
- contact_surname, a string with the contact person’s surname;
- contact_email, a string with the contact person’s e-mail address;
- text, a string containing the open position’s description.

3.11.4.1. Configuration Files

Not applicable.

3.11.4.2. Input from windows

Not applicable.

3.11.4.3. Input from file

Not applicable.

3.11.4.4. I/O from database

The **Open Positions** module, as previously stated, operates mainly using I/O on database. Every routine of this module executes some operation on the DBMS; the **view** section of the modules only performs read operations, the **admin** section performs read/write operations.

3.11.4.5. Output on file

Not applicable.

3.11.4.6. Calls

The **Open Positions** module, as all others modules in the VCE system, calls the following routines from the Security module:

- AUTHENTICATED_USER.getRoles();

3.11.5. Routines

As it happens in the entire Antennas VCE portal, in the **Open Positions** module there is a one-to-one mapping between Zope folders where routines are stored, and web pages.

The **Open Positions** module defines and uses the routines in the following folders:

- /ACE/OpenPosition: DTML methods to show the list of available open positions.
- /ACE/OpenPositions/detail: DTML methods to show the details about a given open position.
- /ACE/OpenPositions/admin: DTML methods to list available open positions. Gives access to the 'insert' and 'modify' pages and allows deleting open positions.
- /ACE/OpenPositions/admin/insert: DTML methods to create a new open position.
- /ACE/OpenPositions/admin/modify: DTML methods to edit an existing open position.

3.12. News, Events and Links

3.12.1. Type

This CSCI is a basic module in Antennas VCE system supporting the publishing of news, events and links relevant to ACE Network and ACE Community.

3.12.2. Functions

The **News, Events and Links** module provides the necessary infrastructure to publish news, events and links into the three different parts of the Antennas VCE web site: ACE Network, ACE Community and Public.

The **News, Events and Links** module is divided in two areas:

- a **view** area, where all users can view the list of news, events or links visible to them;
- an **admin** area, where certain ACE Network members (all ACE Executive Board members plus designated users belonging to Virtual Centre Of Excellence Activity) can add, delete and update the available news, events or links.

3.12.3. Language

The **News, Events and Links** module CSCI is implemented using **Python** and the **Zope DTML** languages.

3.12.4. Interfaces

There is no particular interface with other modules of the system; the only information exchange between **News, Events and Links** module and other modules is based on data contained in the **DBMS** that all modules share.

So to describe the interfaces to the other modules is necessary to describe the tables that are managed by the **News, Events and Links** module:

news table, used for store information about news:

- id_news, the table primary key;
- date, a DATE type with the news date;
- title, a string containing the news title;
- permission, an integer specifying the visibility bitmask of this news (1 = public, 2 = ACE Community, 4 = ACE Network);
- text, a string containing the news description.

events table, used for store information about events:

- id_event, the table primary key;
- visibility, an integer specifying the visibility bitmask of this event (1 = public, 2 = ACE Community, 4 = ACE Network);
- date, a DATE type with the event date;
- title, a string containing the event title;
- text, a string containing the event description.

links table, used for store information about links:

- id_link, the table primary key;
- visibility, an integer specifying the visibility bitmask of this link (1 = public, 2 = ACE Community, 4 = ACE Network);
- date, a DATE type with the link date;
- title, a string containing the link title;
- text, a string containing the link description.

3.12.4.1. Configuration Files

Not applicable.

3.12.4.2. Input from windows

Not applicable.

3.12.4.3. Input from file

Not applicable.

3.12.4.4. I/O from database

The **News, Events and Links** module, as previously stated, operates mainly using I/O on database. Every routine of this module executes some operation on the DBMS; the **view** section of the modules only performs read operations, the **admin** section performs read/write operations.

3.12.4.5. Output on file

Not applicable.

3.12.4.6. Calls

The **News, Evetns and Links** module, as all others modules in the VCE system, calls the following routines from the Security module:

- AUTHENTICATED_USER.getRoles();

3.12.5. Routines

As it happens in the entire Antennas VCE portal, in the **News, Events and Links** module there is a one-to-one mapping between Zope folders where routines are stored, and web pages.

The **News, Events and Links** module defines and uses the routines in the following folders:

News:

- /News: DTML methods to show the list of Public news.
- /Community/News: DTML methods to show the list of ACE Community news.
- /ACE/News: DTML methods to show the list of ACE Network news.
- /ACE/News/admin: DTML methods to manage news. Gives access to the 'insert' and 'modify' pages and allows deleting news.
- /ACE/News/admin/insert: DTML methods to create news.
- /ACE/News/admin/modify: DTML methods to edit existing news.

Events:

- /Events: DTML methods to show the list of Public events.
- /Community/Events: DTML methods to show the list of ACE Community events.
- /ACE/Events: DTML methods to show the list of ACE Network events.
- /ACE/Events/admin: DTML methods to manage events. Gives access to the 'insert' and 'modify' pages and allows deleting events.
- /ACE/Events/admin/insert: DTML methods to create an event.
- /ACE/Events/admin/modify: DTML methods to edit an existing event.

Links:

- /Links: DTML methods to show the list of Public links.
- /Community/Links: DTML methods to show the list of ACE Community links.
- /ACE/Links: DTML methods to show the list of ACE Network links.
- /ACE/Links/admin: DTML methods to manage links. Gives access to the 'insert' and 'modify' pages and allows deleting links.
- /ACE/Links/admin/insert: DTML methods to create a link.
- /ACE/Links/admin/modify: DTML methods to edit an existing link.

3.13. Information Sheets

3.13.1. Type

This CSCI is a basic module in Antennas VCE system supporting the publishing of public information sheets describing Antennas VCE activity and membership benefits.

3.13.2. Functions

The **Information Sheets** module provides the necessary infrastructure to expose to the public information about the Antennas VCE activity, the membership benefits and how to join.

The **Information Sheets** module is divided in two areas:

- a **view** area, where all users can navigate through the available information.
- an **admin** area, where certain ACE Network members (all ACE Executive Board members plus designated users belonging to Virtual Centre Of Excellence Activity) can add, delete and update the available open positions.

3.13.3. Language

The **Information Sheets** module CSCI is implemented using **Python** and the **Zope DTML** languages.

3.13.4. Interfaces

There is no particular interface with other modules of the system; the only information exchange between **Information Sheets** module and other modules is based on data contained in the **DBMS** that all modules share.

So to describe the interfaces to the other modules is necessary to describe the tables that are managed by the **Information Sheets** module:

information_areas table, used for store information about groups of information sheets:

- id_area, the table primary key;
- deleted, a 1-character string: '0' if open position is visible, '1' if deleted;
- title, a string containing the information area's title;
- text, a string containing the information area's description.

information_files table, used for store information about information sheets documents:

- id_file, the table primary key;

- id_area, an integer referring to the information area this document belongs to;
- visibility, an integer specifying the visibility of this document (0 = public, 1 = on request, 2 = restricted to ACE Community);
- filesize, an integer containing the document's length
- title, a string containing the document's descriptive title;
- filename, a string containing the document's file name.

3.13.4.1. Configuration Files

Not applicable.

3.13.4.2. Input from windows

Not applicable.

3.13.4.3. Input from file

Not applicable.

3.13.4.4. I/O from database

The **Information Sheets** module, as previously stated, operates mainly using I/O on database. Every routine of this module executes some operation on the DBMS; the **view** section of the modules only performs read operations, the **admin** section performs read/write operations.

3.13.4.5. Output on file

Not applicable.

3.13.4.6. Calls

The **Information Sheets** module, as all others modules in the VCE system, calls the following routines from the Security module:

- AUTHENTICATED_USER.getRoles();

3.13.5. Routines

As it happens in the entire Antennas VCE portal, in the **Information Sheets** module there is a one-to-one mapping between Zope folders where routines are stored, and web pages.

The **Information Sheets** module defines and uses the routines in the following folders:

- /ACE/Information/admin: DTML methods to list available open positions. Gives access to the 'insert' and 'modify' pages and allows deleting open positions.
- /ACE/Information/admin/insert_area: DTML methods to create a new group of information sheets.
- /ACE/Information/admin/modify_area: DTML methods to edit an existing group of information sheets.
- /ACE/Information/admin/insert_file: DTML methods to create a new document.
- /ACE/Information/admin/modify_file: DTML methods to edit an existing document.
- /Information and /ACE/Information: DTML methods to show the list of visible information sheets groups and documents.

- /Community/Information/requestfile/sendmail, /Community/Information/requestfile/confirm and /Community/Information/requestfile/download: DTML methods to ask for, grant permission and download on-request documents.

3.14. NewsLetter

3.14.1. Type

This CSCI is a basic module in ACE Community system to manage newsletter to all members of community.

3.14.2. Functions

The Newsletter module implements a broadcast of e-mail written by the Community Administrator to all users in ACE Community and also keeps a copy of them. All stored copies are readable all ACE Community Members.

3.14.3. Language

The **NewsLetter** CSCI is implemented using **Python** and the **Zope DTML** languages.

3.14.4. Interfaces

There is no particular interface with other modules of the system; the only information exchange between **NewsLetter** module and other modules is based on data contained in the **DBMS** that all modules share.

3.14.4.1. Configuration Files

Not applicable.

3.14.4.2. Input from windows

The **NewsLetter** module have only one document for all ACE Community members. The document is customized upon roles of each member. Only the administrator can see a form to send a new message. The only one thing that have to do is fill subject field, message field and click on send button.

3.14.4.3. Input from file

Not applicable.

3.14.4.4. I/O from database

Not applicable.

3.14.4.5. Output on file

Not applicable.

3.14.4.6. Calls

The **NewsLetter** module does not call subroutines from other modules; instead it accesses directly the data stored in Zope built-in database.

3.14.5. Routines

The **NewsLetter** module defines the following routines:

- send, to create and send an e-mail;
- memorize, to store a copy of e-mail in zodb;

3.15. Forum

3.15.1. Type

This CSCI is a basic module in ACE Community system that implements a simple forum.

3.15.2. Functions

The **Forum** module implements a moderated forum, where every message needs forum administrator permission to be published. Only ACE Community members can post a new topic. ACE Community visitors can only read and reply an existing topic. All topics are stored in the DBMS.

The **Forum** has two forms: one to create a new topic and one to reply to an existing topic. The first form is rechable from **Forum** main page, using "Add new Topic" button. The user has to fill only two fields of a new topic: title and message. All other fields are added automatically.

The second form allows users only to write a reply message: the topic title is inherited.

3.15.3. Language

The Forum is implemented using **Python,** the **Zope DTML** languages and PostgreSQL.

3.15.4. Interfaces

There is no particular interface with other modules of the system; the only information exchange between **NewsLetter** module and other modules is based on data contained in the **DBMS** that all modules share.

So to describe the interfaces to the other modules is necessary to describe the tables that are managed by the **Forum** module:

topics table, used for store information about forums topics:

- id_topic, the table primary key;
- data, a time stamp;
- id_user, the user who created this topic;
- title the topic title;
- msg, the topic contents;
- ack, a boolean flag: false means forum administrator did not review this topic yet (default value), true means forum administrator granted permission to publish this topic.
- reffer, the parent of this topic in this table, i.e. this topic is a reply of 'reffer' topic.

3.15.4.1. Configuration Files

Not applicable.

3.15.4.2. Input from Windows

Not applicable.

3.15.4.3. Input from file

Not applicable.

3.15.4.4. I/O from database

The **Forum** module storse and reads all topics from DBMS.
Main methods are:

- add_topic(data, id_user, title, msg), insert a new topic;
- delete_topic(id_topic), delete an existing ;
- reply_topic(reffer, data, id_user, title, msg, role), insert a reply to an existing topic;
- full_topic(id_topic), show all messages of a thread;
- list_topic(role), show all topics.

Administrator methods are:

- publish_topic(id_topic), enable all members to read a message;
- reject_topic(id_topic), deny all members to read a message.
- login_2_id(username), finds a id_user from the user's login name.

3.15.4.5. Output on file

Not applicable.

3.15.4.6. Calls

The **Forum** module does not call subroutines from other modules; instead it accesses directly the data stored on DBMS by using SQL queries.

3.15.5. Routines

The **Forum** module contains two routines that also correspond to two web pages:

- add, to insert a new topic;
- view, to reply, publish, reject and delete topics.

3.16. Members

3.16.1. Type

This CSCI is a basic module in ACE Community system that implements a simple list of all ACE Community members and their organisations.

3.16.2. Functions

The **Members** module display a list of all organisations registered to ACE Community, divided in ACE Organisations and Community Organizations. For each organisation, the list of members, a logo and all contact information are displayed.

3.16.3. Language

The Members module is implemented using **Python** and the **Zope DTML** languages.

3.16.4. Interfaces

There is no particular interface with other modules of the system; the only information exchange between **Members** module and other modules is based on data contained in the **DBMS** that all modules share.

3.16.4.1. Configuration Files

Not applicable.

3.16.4.2. Input from Windows

Not applicable.

3.16.4.3. Input from file

Not applicable.

3.16.4.4. I/O from database

The **Members** module uses two SQL methods to get all organisations and all users from a single organisation:

- list_members(), shows all organisations;
- members_from(id_organization), shows all members from an organisation.

3.16.4.5. Output on file

Not applicable.

3.16.4.6. Calls

The **Members** module does not call subroutines from other modules; instead it accesses directly the data stored on DBMS by using SQL queries.

3.16.5. Routines

The **Members** module contains a single routine that also corresponds to a single web page:

- index_html, shows all ACE Community members and their organisations.

3.17. Software

3.17.1. Type

This CSCI is a basic module in ACE Community system that implements a card repository. Every card is a description of an existing software.

3.17.2. Functions

The **Software** module allows searching, inserting and managing software cards uploaded by ACE Community members. A member can view all cards but can only modify his own cards.

To search a software card, a keyword or software name initial character is needed.

To upload a new software card, a multi-step procedure is needed. The procedure starts from the main Software module page, clicking "Upload your software description". Next step is to fill all required fields in the software description form. Only price and file are optional. Finally, click on button "upload" to store the software card. Once the card is stored, a second form with optional fields appears.

The owner of a software card can modify it from the view page of the card.

3.17.3. Language

The **Software** module is implemented using **Python** and **Zope DTML** languages.

3.17.4. Interfaces

There is no particular interface with other modules of the system; the only information exchange between **Software** module and other modules is based on data contained in the **DBMS** that all modules share.

3.17.4.1. Configuration Files

Not applicable.

3.17.4.2. Input from Windows

Not applicable.

3.17.4.3. Input from file

Not applicable.

3.17.4.4. I/O from database

The **Software** module defines and uses these SQL methods:

- add_sw(), upload a new card;
- add_sw2(), modify a card;
- info_sw(), get a software card;
- update_sw(), store changes on a card;
- enable_sw(), enable a card owner to publish his software cards;
- list_sw_sub(), show all cards not published;
- list_sw(), show all cards published;
- list_sw_filtered(), show a list of cards filtered bya keywords;
- list_sw_named(), show a list of cards where names start with a determinated char;

3.17.4.5. Output on file

Not applicable.

3.17.4.6. Calls

The **Software** module does not call subroutines from other modules; instead it accesses directly the data stored on DBMS by using SQL queries.

3.17.5. Routines

- list_software, the search form
- list_software_full, shows the complete list all software cards
- detail, shows details about a single software card
- add_software, form to insert required information for a new software card
- add2_software, form to insert optional information for a new software card

3.18. Consortium and Researchers

3.18.1. Type

This CSCI is a basic module in VCE system that provide the information related to the Institutions and the researchers involved in the ACE Network.

3.18.2. Functions

The module of **Consortium and Researchers** supplies one solution for the necessities of the presentation of every user and organisation involved in the ACE consortium.

These requirements are simply a subset of the functionalities provided by the User Interface module (in particular the subset composed by the “read-only” functionalities); see sec. 3.1 for the implementation notes.