Integra

A system for Project Management

Software Requirements Specification

Samuele Gantner

Departement of Computer Science
Swiss Federal Insitute of Technology
Zürich, Switzerland

Software Engineering for Outsourcing and Offshoring

Document Id	INRA-SRS
Version	0.1
Date	December 2006
File name	requirements.pdf

Contents

1 Introduction **9**
1.1 Purpose . . . 9
1.2 Scope . . . 9
1.3 Definitions, Acronyms and Abbreviations . . . 9
1.4 Overview . . . 11

2 Overall Description **12**
2.1 Product perspective . . . 12
2.1.1 User interfaces . . . 12
2.1.1.1 Standard Tasks view . . . 12
2.1.1.2 Staff overview . . . 12
2.1.1.3 Teams overview . . . 13
2.1.2 Software interfaces . . . 13
2.1.2.1 Origo interface . . . 13
2.1.2.2 External interface . . . 13
2.1.3 Memory constraints . . . 13
2.2 Product functions . . . 14
2.2.1 Accounts definition . . . 14
2.2.1.1 MaintainAccounts (Administrator) . . . 14
2.2.2 Project definition . . . 14
2.2.2.1 MaintainProjects (Project leader) . . . 14
2.2.2.2 MaintainTasks (Project leader) . . . 15
2.2.2.3 DefineSkills (Project leader) . . . 15
2.2.2.4 LinkTasks (Project leader) . . . 15
2.2.3 Staff management . . . 15
2.2.3.1 DefineTeams (Project leader / Project manager) . . . 15
2.2.3.2 MaintainStaff (Project manager) . . . 15
2.2.3.3 AssignStaff (Project manager) . . . 15
2.2.4 During the project . . . 15
2.2.4.1 MaintainSubtasks (Team leader) . . . 15
2.2.4.2 AssignTeamMembers (Team leader) . . . 15
2.2.4.3 ReportStatus (Team leader / Team member) . . . 15
2.2.5 Global functions (All Project members) . . . 15
2.2.5.1 ShowStatusOverview . . . 15
2.2.5.2 ManageSessionAndAccount . . . 16
2.3 User characteristics . . . 16
2.4 Assumptions and dependencies . . . 16

3 Specific Requirements **17**
3.1 External interface requirements . . . 17
3.2 Classes . . . 17
3.2.1 User and subclasses . . . 17
3.2.2 Absence . . . 18
3.2.3 Skill . . . 18
3.2.4 Team . . . 18
3.2.5 Task . . . 18
3.3 Functions . . . 18
3.3.1 Formats . . . 18
3.3.2 Common behaviors and standards . . . 19
3.3.2.1 A note about the format . . . 19
3.3.2.2 Errors . . . 19
3.3.3 MaintainAccounts (Administrator) . . . 19

3.3.3.1 CreateAccount . . . 20
3.3.3.2 SearchUser . . . 20
3.3.3.3 ModifyAccount . . . 20
3.3.3.4 RemoveAccount . . . 21
3.3.4 MaintainProjects (Project leader) . . . 21
3.3.5 MaintainTasks (Project leader) . . . 21
3.3.5.1 NavigateTasks . . . 21
3.3.5.2 SearchTask . . . 21
3.3.5.3 CreateTask . . . 22
3.3.5.4 ModifyTask . . . 23
3.3.5.5 RemoveTask . . . 23
3.3.6 DefineSkills . . . 23
3.3.6.1 CreateSkill . . . 23
3.3.6.2 SearchSkill . . . 23
3.3.6.3 ModifySkill . . . 23
3.3.6.4 RemoveSkill . . . 23
3.3.6.5 AddSkillToStaffMember . . . 23
3.3.6.6 RemoveSkillFromStaffMember . . . 23
3.3.7 LinkTasks . . . 23
3.3.7.1 AddPrerequisite . . . 23
3.3.7.2 RemovePrerequisite . . . 23
3.3.8 DefineTeams . . . 23
3.3.8.1 CreateTeam . . . 23
3.3.8.2 SearchTeam . . . 23
3.3.8.3 ModifyTeam . . . 23
3.3.8.4 RemoveTeam . . . 23
3.3.8.5 ProposeOptimalTeamForTask . . . 23
3.3.8.6 ViewTeamsOverview . . . 23
3.3.8.7 AssignTeamToTask . . . 23
3.3.8.8 RemoveTeamFromTask . . . 23
3.3.9 MaintainStaff . . . 23
3.3.9.1 AddStaffMember . . . 23
3.3.9.2 SearchStaffMember . . . 23
3.3.9.3 RemoveStaffMember . . . 23
3.3.9.4 ViewStaffOverview . . . 23
3.3.9.5 AddAbsence . . . 23
3.3.9.6 RemoveAbsence . . . 23
3.3.10 AssignStaff . . . 23
3.3.10.1 AddStaffMemberToTeam . . . 23
3.3.10.2 RemoveStaffMemberFromTeam . . . 23
3.3.10.3 DefineTeamLeader . . . 23
3.3.11 MaintainSubtasks . . . 23
3.3.12 AssignTeamMembers . . . 24
3.3.12.1 ProposeOptimalTeamMembersForSubtask . . . 24
3.3.12.2 AssignTeamMemberToSubtask . . . 24
3.3.12.3 RemoveTeamMemberFromSubtask . . . 24
3.3.13 ReportStatus . . . 24
3.3.13.1 ModifyTaskStatus . . . 24
3.3.13.2 ReportTaskCompleted . . . 24
3.3.13.3 LinkProducedDocuments . . . 24
3.3.13.4 UnlinkProducedDocuments . . . 24
3.3.14 ShowStatusOverview . . . 24
3.3.14.1 ShowCriticalPath . . . 24

3.3.14.2 ShowStaffOverview . . . 24
3.3.14.3 BeginSimulation . . . 24
3.3.14.4 EndSimulation . . . 24
3.3.15 ManageSessionAndAccount . . . 24
3.3.15.1 Login . . . 24
3.3.15.2 Logout . . . 24
3.3.15.3 ChangePassword . . . 24
3.4 Usability . . . 24
3.4.1 Administrator . . . 24
3.4.2 Project leader . . . 24
3.4.3 Project manager . . . 24
3.4.4 Team leader . . . 24
3.4.5 Team member . . . 25
3.5 Reliability . . . 25
3.5.1 Data protection . . . 25
3.5.2 Concurrent access . . . 25
3.5.3 Simulation mode . . . 25
3.5.4 Connections . . . 25
3.6 Security . . . 25
3.6.1 Sessions . . . 25
3.6.2 External interface . . . 25
3.7 Performance . . . 25
3.7.1 Execution time . . . 25
3.8 Maintainability . . . 26
3.8.1 Change of developers . . . 26
3.8.2 Minor changes . . . 26
3.9 Design Constraints . . . 26
3.10 On-line User Documentation and Help System Requirements . . . 26
3.11 Licensing Requirements . . . 26

4 Supporting information **27**

List of Figures

1 Project, tasks, teams and people 10
2 Integration with Origo 12
3 Functions overview 14
4 Class diagram 17

List of Tables

1 Input formats 18

Creation

Project Role	Author	Date	Signature
Project Manager	S. Gantner	December 2006	

Revision History

Author	Version / Date	Description	Signature
S.Gantner	0.1 / December 2006	First Version	

Review and Approval

Project Role	Author	Version / Date	Signature

1 Introduction

1.1 Purpose

The purpose of this document is to delineate a detailed software requirement specification for a system for software project management. It is addressed to any member of a software producer company or individual, and to any potential software products customer.

1.2 Scope

The product described in this document is a system for software project management. From now on we will use the codename *Integra* to describe it. We will focus our attention on the following aspects of project management:

- Project schedule
- Task management
- Staff management

Aspect related to more technical details, such as integration with configuration management tools (like CVS or SourceSafe), bug tracking and communication between the people involved in the project will not be part of this specification. However the system will be designed in such a way that future implementation or interfacing shall be possible.

Due to the elevated number of proposed functions, some of the specifications of Section 3 are missing. We consider that for the purpose of this (academic) document it is more important to present a general view of a complex system that may in the future be actually developed, that to restring the SRS to a smaller number of simpler functions. After the analyze of this document the reader should understand the concept behind the proposed software. Any missing part can be filled in the future.

The most important application of this software is the task and people management of software project of any size. Even if the main target consists of projects with more than twenty people involved, smaller projects can take benefit from the tools provided in *Integra* , at least partially. The most important benefits of this application are the following:

- Complete specification of the project schedule, which helps to define costs and duration.
- Clear overview of the project completion status, which allows to identify where the teams must focus.
- Easy and effective assignment of human resources, an essential and often difficult task for large projects.
- Improved flexibility toward changes in schedule, assignment and tasks.
- Safe storage of data.

1.3 Definitions, Acronyms and Abbreviations

User A user is any person interacting with the application. Users can be categorized as follows:

- Administrator: the system administrator.
- Project leader: the leader of a project.
- Project manager: the manager of a project.
- Team leader: the leader of a development team.
- Team member: a member of the development team.

When it is not important to distinguish between team leader and team member, the general word *staff* (or staff member) will be used. A project member is a Project leader, Project manager or staff member of a certain project.

The difference between Project leader and Project manager resides in the more technical role of the first. In practice concerning *Integra* the project leader defines the tasks, while the role of the project manager is more business and people oriented, he should be more concerned about the respect of the schedule and the costs and the management of the staff.

API One of the secondary purposes of an API is to describe how computer applications and software developers may access a set of (usually third party) functions (for example, within a library) without requiring access to the source code of the functions or library, or requiring a detailed understanding of the functions' internal workings[1].

Task In project management a task is an activity that needs to be accomplished within a defined period of time. Tasks can be linked together to create dependencies. [1]

To this definition we add the fact that tasks can be nested, which means that one or more tasks can be part of a larger task. The depth of a task is defined as the nested depth. Basic tasks have depth 1.

Project, tasks, teams, and people -

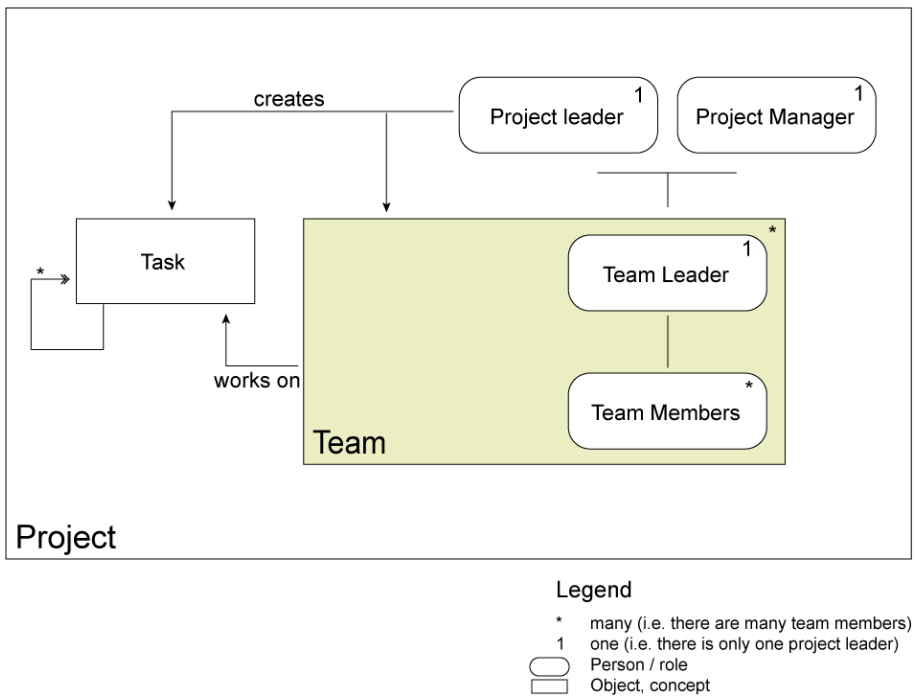

Figure 1: Project, tasks, teams and people

As informally shown in figure 1, a project is defined by the Project leader and the Project manager. The Project manager defines the basic tasks of the project, the project manager elects the team leaders and assigns the team members. Every task is then assigned to a team. The team leader can define subtasks for his given tasks, and assign these subtasks to the team members.

Skill We define as skill any general ability related to a software project. A skill can be for example *GUI designer* or *C++ experience*. Every skill as a ranking scale. Related to the previous example *C++ experience* could take a value between *0* and *4*.

SRS Software Requirements Speficication, see [2]

Subtask We will call subtask any task which is part of a larger task, i.e. task B can be a subtask of task A, and task C can be a subtask of task B. In this case task A is called the parent of task B, and task B is called a child of task A.

1.4 Overview

The rest of this SRS analyzes the detailed requirements of *Integra* . The document is organized as follows:

Section 1 provides an overview of the software described in this document. The reading of this section is recommended since it help to understand the basics and the purpose of this document and of *Integra* . Some important definition are also given.

Section 2 digs further in the product specification, delineating the perspective of this product, the functions, and other general information.

Section 3 is a technical description of the functionality of this software. It is used to define without ambiguity the exact behavior of the desired function. It should be used as the information and specification base for the implementation.

Section 4 contains a short glossary (for the definition that are not covered in Section 1) the bibliography and the index.

This SRS follows the guidelines of IEEE 830-1998 [2].

2 Overall Description

2.1 Product perspective

This product can be viewed as a standalone application, in fact the basic functionalities are independent from any other product. However it becomes a complete suite if integrated with other products, so that all the functionalities already cited in the introduction (*section 1.2 on page 9*) can be provided. We will hence present *Integra* in the perspective of an integration with Origo [3].

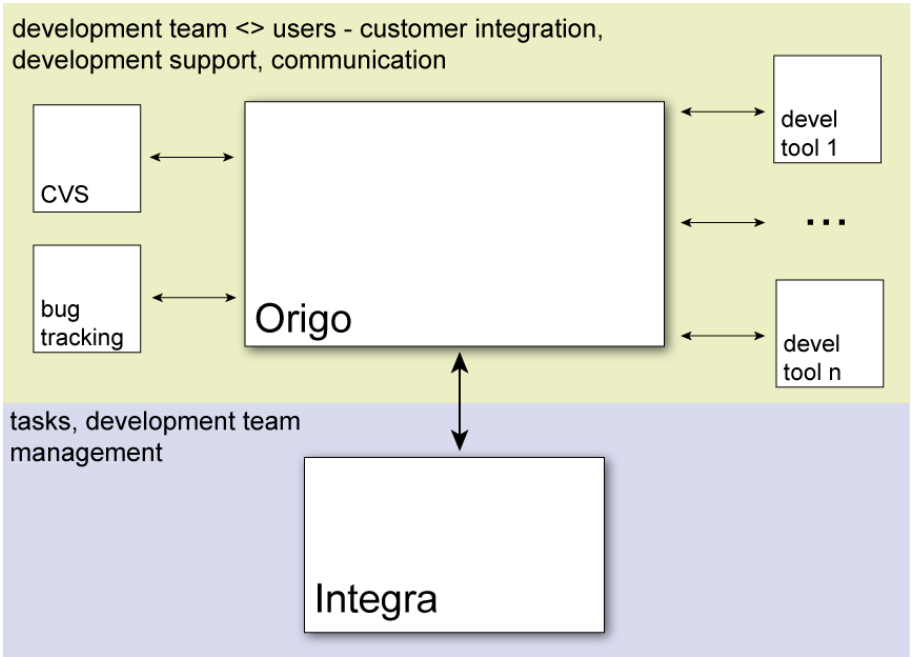

Figure 2: Integration with Origo

In this optic the product shall not only communicate with Origo, but also share certain information. More details about this will be presented in the rest of the document, however it is important to notice that login information and user data will be shared.

Figure 2 shows how the functionalities provided by *Integra* perfectly integrate in the Origo environment, since there are no overlapping parts and the global product in a complete tool.

2.1.1 User interfaces

The main user interface shall be a web browser, where all the functionalities of the product are available. Additional interfaces include emails, optional interfaces can include SMS and WAP (but are not discussed in this document). The last three interfaces shall not provide full features, they will only be used (due to their limitations) for reports and reminders.

2.1.1.1 Standard Tasks view

The Standard Tasks View allows the user to see a graphical representation of the Tasks. It shall be presented in the form of PERT chart. If a particular Task contains Subtasks, those shall be visible as well.

2.1.1.2 Staff overview

The Staff overview allows to see a graphical representation of all the staff of the project. The staff can be grouped by Skills, Team, Occupation or assigned Task.

2.1.1.3 Teams overview

The Teams overview allows to see a graphical representation of the teams of the projects. Two modes are possible:

- Teams Skills: displays all the available information about the team and a chart of the skills. On the x-axis the project skills, on the y-axis the sum of the skill proficiency of each team member divided by the number of team members. In more formal terms the function used to obtain the chart is defined as follows. Let $S = \{s_1, ..., s_n\}$ be the set of defined skills, t_i a specific team and $M = \{m_1, ..., m_k\}$ the set of team members of t_i. We define $pro(s, m)$ the level of proficiency of team member m in skill s and $mem(t) \subseteq M$ the set of team members of team t. Then

$$f_{skills}(t_i, s_j) = \frac{1}{k} \sum_{m \in M} pro(s_j, m)$$

  In the chart $x \in S$ and $y = f_{skills}(t_i, x)$.

- Teams Power: this view behaves like Teams Skills, the only difference is that the sum of the skill proficiency is not divided by the number of team members. Formally

$$f_{power}(t_i, s_j) = \sum_{m \in M} pro(s_j, m)$$

  In the chart $x \in S$ and $y = f_{power}(t_i, x)$.

This two views allow the Project manager to understand the skills of the teams. The first one can be viewed as the quality of team. The second one as the "brute force" power of the team.

2.1.2 Software interfaces

Two different software interfaces shall be implemented; the first one for Origo, the second one for future, and at the moment unknown, possible expansions. This second interface will from now on be called *External interface*.

2.1.2.1 Origo interface

The purpose of this interface is to share information related to login information and user data. Specifically the login for *Integra* shall be integrated with the Origo's, and the data concerning registered users shall be provided by Origo. The message content and format is the one already provided by Origo, in the general form of Eiffel method calls.

2.1.2.2 External interface

All the functions available for the users shall be available to external programs too, as happens with Origo, in the form of an open API. The message content and format is the one already provided by Origo, in the general form of Eiffel method calls.

2.1.3 Memory constraints

The are no specified limit on secondary memory, however the application shall run on any reasonable LAMP platform. We will assign a limit of 100 MB of storage data for a project consisting of 100 tasks and 50 people.

Concerning primary memory, for the same reasons expressed above, we will assign a limit of an average of 4 MB for every open session (logged user), and a peak limit of 64 MB for every operation.

2.2 Product functions

We will discuss in this section the functions of *Integra* . Note that for the purpose of this introduction only the general aspect will be considered. For a more more details about each function please see *section 3 on page 17*.

In order to better understand the intended flow of the system, the function will be discussed in the order of a general workflow. Please note however that the use of the system is not restricted to this particular order.

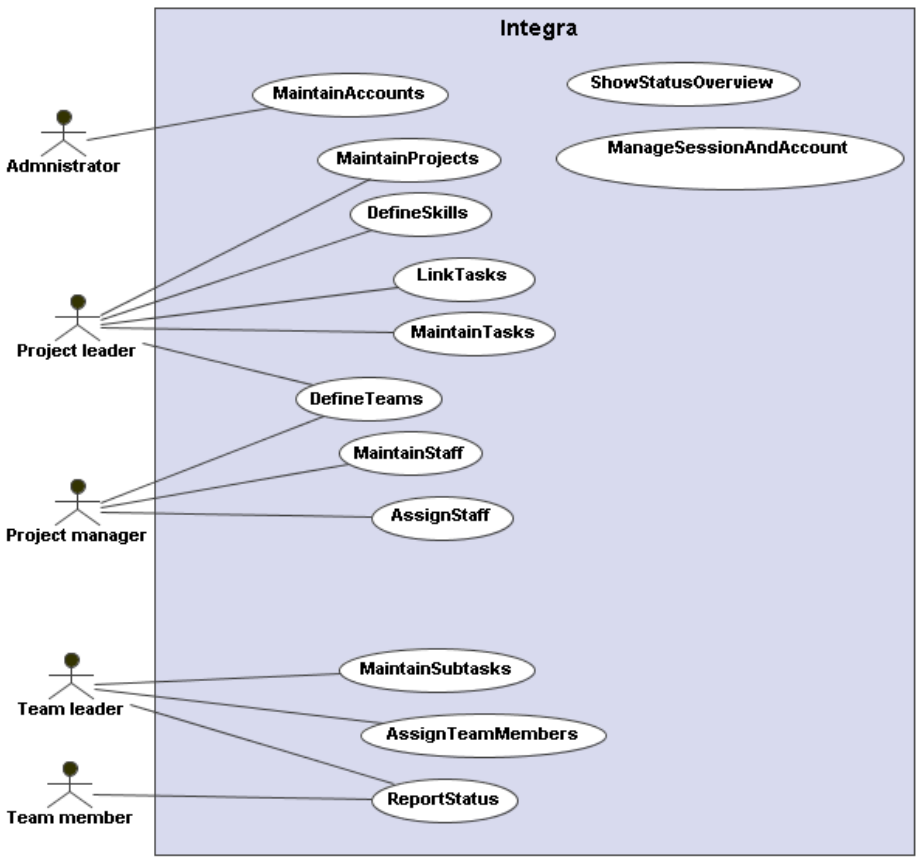

Figure 3: Functions overview. Use cases not linked with any role are available to all the project members.

2.2.1 Accounts definition

2.2.1.1 MaintainAccounts (Administrator)

As first step of the use of the system, the Administrator defines the accounts and the basic roles for all the future users. The roles are Project leader and user.

2.2.2 Project definition

2.2.2.1 MaintainProjects (Project leader)

The Project leader creates, modifies or deletes projects. During the creation of a project he defines all the attributes of that project and he chooses a Project manager between the registered users.

2.2.2.2 MaintainTasks (Project leader)

The Project leader creates, modifies or deletes the basic tasks of the project, beginning from tasks of depth 1, and if necessary adding subtasks. All the operation are performed directly on the Standard Task view.

2.2.2.3 DefineSkills (Project leader)

The project leader defines the list of the personal skills relevant to the project and assigns a scale to every skill. In a second phase he assign skills to the staff.

2.2.2.4 LinkTasks (Project leader)

The Project leader links the tasks of the projects. Linking a task means defining which task shall be completed before being able to start a certain other task.

2.2.3 Staff management

2.2.3.1 DefineTeams (Project leader / Project manager)

The Project leader or the Project manager creates, modifies or deletes the teams. He also assigns team to tasks. The system is able to propose the optimal team for a given task.

2.2.3.2 MaintainStaff (Project manager)

The project manager add or removes staff members from the list of registered users. He defines the details about the staff members. He also assigns absences (due to holidays, sickness or any other reason).

2.2.3.3 AssignStaff (Project manager)

The project manager defines which staff members will be team leaders, and assigns them a team. He assigns also the staff members to a team (as team members).

2.2.4 During the project

2.2.4.1 MaintainSubtasks (Team leader)

The Team leader maintain the subtasks of the tasks his team is assigned to. This set of functions behaves in the same manner as for MaintainTasks (Project leader). The only difference is that in this case the Team leader can only manage tasks of greater depth than the tasks he is assigned to.

2.2.4.2 AssignTeamMembers (Team leader)

The Team leader assigns the members of his teams to the subtasks he defines in *MaintainSubtasks*. The system is able to propose the optimal team members for a given task.

2.2.4.3 ReportStatus (Team leader / Team member)

The team leader or the team members reports the completion status of a task they are assigned to. He also links the requested documents with the produces documents.

2.2.5 Global functions (All Project members)

2.2.5.1 ShowStatusOverview

A Project member check the global status of the project, using various view method. It is here possible to enter in Simulation mode, all the operations executed in that mode are not applied to the project. The simulation mode allows to see the consequences of possible changes.

2.2.5.2 ManageSessionAndAccount

A Project member performs login and logout operations or changes his password.

2.3 User characteristics

The intended users of the system are active members of software projects. Members of this category have by definition a certain level of technical expertise and a well defined educational background. The system is not designed for people with no experience in the general use of a computer.

2.4 Assumptions and dependencies

The only assumption of this SRS is related to the development of Origo. We suppose here that the integration of CVS and bug-tracking in Origo will be completed before the the beginning of the development of *Integra* . In case this assumption should not be realized, the implementation of all the functionalities related to CVS and bug-tracking will be postponed.

3 Specific Requirements

In this section the specific requirements of *Integra* will be discussed. In order to provide a clear and simple explanation an object oriented approach will be taken (see [2] 5.3.7.3). Please note that the classes provided here will not necessarily correspond to the classes of later part of the project. In fact their purpose is only to specify in a more formal approach the precise elements of the software and not to analyze the implementation details. For this reason the presented classes will only have attributes.

3.1 External interface requirements

See *section 2.1.1 on page 12* and *section 2.1.2 on page 13*.

3.2 Classes

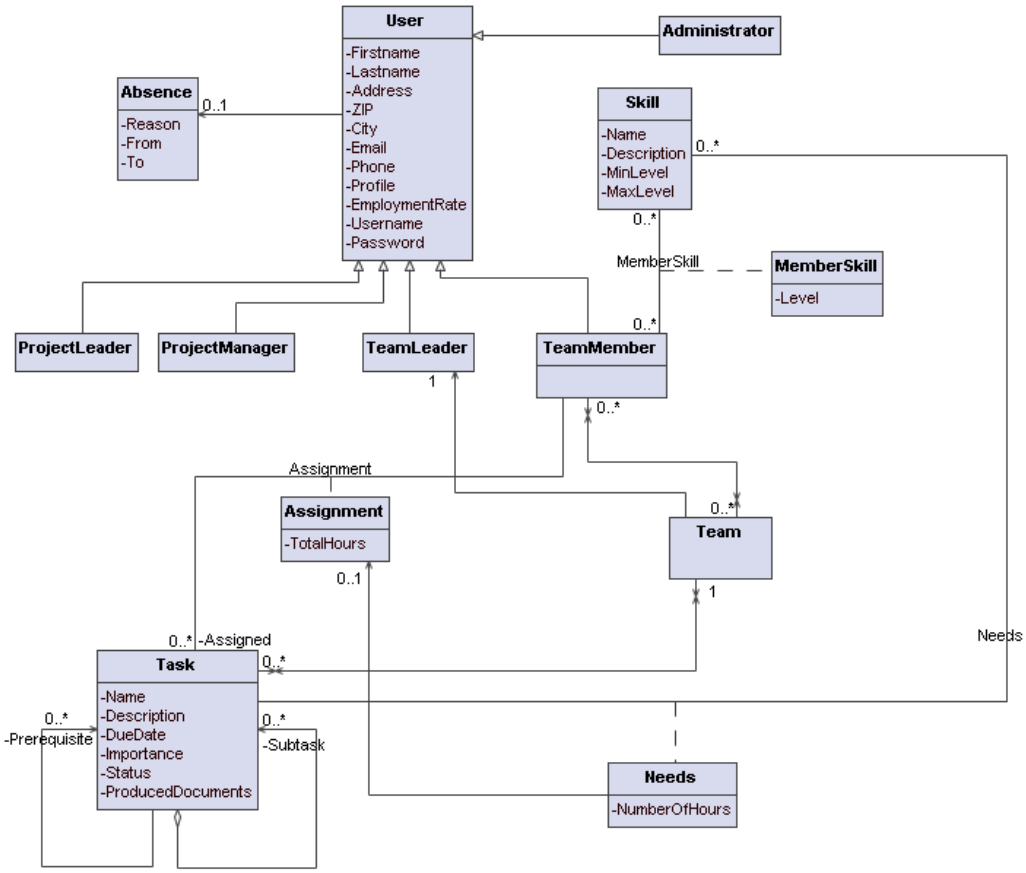

Figure 4: Class diagram (standard UML2.0 notation)

We will describe the most important attributes of each class before continuing with the functional requirements.

3.2.1 User and subclasses

A general user is characterized by some personal details (like Firstname, Lastname, etc). The subclasses correspond to the specification of the type of users already provided. The EmploymentRate attribute represent the percentage of the time the user is employed.

3.2.2 Absence

Class absence is used to represent the periods in time of absence of the user from work. The attributes are self explanatory.

3.2.3 Skill

This class represent the skills related to the project context. Skills can be associated with TeamMembers by providing a level of proficiency.

3.2.4 Team

A team represents the association of a Team leader and of 0 or more Team members. A team can be assigned to 0 or more tasks.

3.2.5 Task

Tasks are the key concept of *Integra* . A Task has a Name and a Description. The DueDate represent the deadline for the completion of the task. The Importance is a flag given to determinate the relative importance. The Status represents the completion level of the task. The ProducedDocuments attribute is a list of the documents that should be produced during that task. A task has a list of prerequisite tasks (tasks that need to be completed before it can begin), and a list of subtasks. TeamMembers are assigned to a specific task. Finally a task is associated to a list of skills. This association is used to specify the typology of the desired Team members that should be assigned to that task. The association also provides the possibility to define an estimate of the time needed for each potential Team member to complete the task.

3.3 Functions

In this subsection we specify all the functions of *Integra* . The specification tries to be as accurate and complete as possible. Any eventual missing part shall be interpreted as normally happens in general software.

3.3.1 Formats

We define the following input formats

Name	Description
Literal l	$l \in \{a, b, ..., z, A, B, ..., Z\}$
String $S = s_1 s_2 ... s_n$	s_i is a Literal. If not stated otherwise the minimum length of a String for the input validation is 3 characters ($i = 2$).
Number $N = n_1 n_2 ... n_k$	$n_i \in \{0, 1, ..., 9\}$
Text $T = t_1 t_1 ... t_n$	t_i is a String, a Number, a Special Character or a Blank
Special Character Sp	$Sp \in \{+, *, \&, ?, ., :, (,), ;, -, /, \$, \}$
Blank b	b is a blank character (like tab, newline, space, ...)
	Remark: if not stated otherwise the maximum length for the input formats is of 255 characters.

Table 1: Input formats

3.3.2 Common behaviors and standards

We present here a list of common behaviors of the functions presented in the rest of the document. These are to be considered general rules to be applied constantly.

3.3.2.1 A note about the format

The function presented in this section are in the following format:

FunctionName

Description	A short description.
Type	The type of the function. *User function* means that the function will be executed by a user.
Precondition	Eventual precondition.
Input	The list of input arguments; [InputName] means that the argument is optional.
Input validation	The type of validation that shall be executed on the inputs.
Operations	One of the most important parts of the specification, here is described what the function shall do.
Output	The output of the function.
Errors	What happens in case of error.

3.3.2.2 Errors

Overflow In case of overflow of the user's input an error message shall be displayed and the operation shall be aborted.

Wrong input In case of wrong user's input (not conform with the input validation) an error message shall be displayed and the operation shall be aborted.

Internal error In case of internal error an error message shall be displayed and the operation shall be aborted. If not stated otherwise no special recovery is necessary.

Failed precondition In case the precondition should evaluate to false, an error message shall be displayed and the operation shall be aborted.

3.3.3 MaintainAccounts (Administrator)

See *section 2.2.1.1 on page 14*.

3.3.3.1 CreateAccount

Description	The administrator creates a new User.
Type	User function.
Precondition	The User is an Administrator.
Input	Firstname, Lastname, Address, ZIP, City, Email, Phone, Profile, EmplymentRate.
Input validation	Firstname, Lastname, Address, City, Profile are Strings; ZIP is a Number of at least 4 characters; Email is a valid email address; Phone is in the form (+X Y) where X is a Number of length 2 and Y is a Number of length ≥ 3; EmplymentRate is a number $\in [0, 100]$.
Operations	The system shall create a new User with the given input. The Username is obtained by taking the first character of the Firstname, and appending the Lastname; The Password is random generated, it is a Text of 8 characters without Blanks.
Output	A confirm message; an email message sent to the new User specifying Username and Password.
Errors	See *section 3.3.2.2 on page 19*.

3.3.3.2 SearchUser

Description	The administrator search for an existing User.
Type	User function.
Precondition	The User is an Administrator.
Input	[Firstname], [Lastname], [Address], [Username].
Input validation	If present Firstname, Lastname, Address, Username are Strings.
Operations	The system shall search for a User conform to the input. If no input is given the system shall return all the Users.
Output	The list of Users meeting the input.
Errors	See *section 3.3.2.2 on page 19*; if no User is found a message shall be displayed;

3.3.3.3 ModifyAccount

Description	The administrator modifies an existing User.
Type	User function.
Precondition	The User is an Administrator.
Input	TargetUser (result of a SearchUser operation).
Input validation	TargetUser exists.
Operations	The system shall request the list of parameters to be modified (except Username and Password); the User provides those parameters; the system shall validate the parameters according to the same rules of CreateUser; the system shall update the TargetUser.
Output	A confirm message.
Errors	See *section 3.3.2.2 on page 19*; if the TargetUser cannot be found an error message shall be displayed and the operation shall be aborted.

3.3.3.4 RemoveAccount

Description	The administrator removes an existing User.
Type	User function.
Precondition	The User is an Administrator.
Input	TargetUser (result of a SearchUser operation).
Input validation	TargetUser exists.
Operations	The system shall request the list of parameters to be modified; the User provides those parameters; the system shall validate the parameters according to the same rules of CreateUser; the system shall update the TargetUser.
Output	A confirm message.
Errors	See *section 3.3.2.2 on page 19*; if the TargetUser cannot be found an error message shall be displayed and the operation shall be aborted.

3.3.4 MaintainProjects (Project leader)

See *section 2.2.2.1 on page 14*.

For the purposes of this document and considering that it is not an industry SRS this part will not be discussed. We will suppose that one installation of *Integra* is necessary for every project.

3.3.5 MaintainTasks (Project leader)

See *section 2.2.2.2 on page 15*

3.3.5.1 NavigateTasks

Description	The Project leader navigate the Tasks.
Type	User function.
Precondition	The User is a Project leader.
Input	none.
Input validation	not applicable.
Operations	The system shall present the Standard Tasks view; The User chooses to enter or exit a particular Task (if applicable)[1]; The system updates the view and continues with the previous part until the User aborts.
Output	none.
Errors	See *section 3.3.2.2 on page 19*.

3.3.5.2 SearchTask

Description	The Project leader search for an existing Task.
Type	User function.
Precondition	The User is a Project leader.
Input	none.
Input validation	not applicable.
Operations	The system shall present the standard Tasks view; The User NavigateTasks until selecting one or aborting.
Output	The selected Task or nothing if abort.
Errors	See *section 3.3.2.2 on page 19*.

[1] Enter/Exit a Task: it is possible to Enter a task if it has at least one subtask. The result of the operation is a new Standard Task View based on the content of the parent task. With the same concept it is possible to Exit a task that has a parent Task, the result is a new Standard Task View based on the content of the parent.

3.3.5.3 CreateTask

Description	The Project leader creates a new Task.
Type	User function.
Precondition	The User is a ProjectLeader.
Input	Name, Description, DueDate, Importance, ProducedDocuments, [ParentTask] (result of a SearchTask operation), [OptimalSkills] (result of a SearchSkill operation).
Input validation	Name, Description are Strings; Importance is a Number $\in [0, 10]$; ProducedDocuments is a list of Strings; OptimalSkills is a list of Skills.
Operations	The system shall create a new Task with the given input, and link it with ParentTask if necessary. For each of the OptimalSkill the system shall ask the required level; the User specifies the level according to the possible values of that Skill.
Output	A confirm message.
Errors	See *section 3.3.2.2 on page 19*; if the ParentTask cannot be found an error message shall be displayed and the operation shall be aborted.

3.3.5.4 ModifyTask

3.3.5.5 RemoveTask

3.3.6 DefineSkills

3.3.6.1 CreateSkill

3.3.6.2 SearchSkill

3.3.6.3 ModifySkill

3.3.6.4 RemoveSkill

3.3.6.5 AddSkillToStaffMember

3.3.6.6 RemoveSkillFromStaffMember

3.3.7 LinkTasks

3.3.7.1 AddPrerequisite

3.3.7.2 RemovePrerequisite

3.3.8 DefineTeams

3.3.8.1 CreateTeam

3.3.8.2 SearchTeam

3.3.8.3 ModifyTeam

3.3.8.4 RemoveTeam

3.3.8.5 ProposeOptimalTeamForTask

3.3.8.6 ViewTeamsOverview

3.3.8.7 AssignTeamToTask

3.3.8.8 RemoveTeamFromTask

3.3.9 MaintainStaff

3.3.9.1 AddStaffMember

3.3.9.2 SearchStaffMember

3.3.9.3 RemoveStaffMember

3.3.9.4 ViewStaffOverview

3.3.9.5 AddAbsence

3.3.9.6 RemoveAbsence

3.3.10 AssignStaff

3.3.10.1 AddStaffMemberToTeam

3.3.10.2 RemoveStaffMemberFromTeam

3.3.10.3 DefineTeamLeader

3.3.11 MaintainSubtasks

The functions available here are the same as in MaintainTasks. The only difference is that the user is a TeamLeader and AddTask, RemoveTask and ModifyTask operations are only available for Subtasks of the Tasks the TeamLeader is assigned to.

3.3.12 AssignTeamMembers

3.3.12.1 ProposeOptimalTeamMembersForSubtask

3.3.12.2 AssignTeamMemberToSubtask

3.3.12.3 RemoveTeamMemberFromSubtask

3.3.13 ReportStatus

3.3.13.1 ModifyTaskStatus

3.3.13.2 ReportTaskCompleted

3.3.13.3 LinkProducedDocuments

3.3.13.4 UnlinkProducedDocuments

3.3.14 ShowStatusOverview

3.3.14.1 ShowCriticalPath

3.3.14.2 ShowStaffOverview

3.3.14.3 BeginSimulation

3.3.14.4 EndSimulation

3.3.15 ManageSessionAndAccount

3.3.15.1 Login

3.3.15.2 Logout

3.3.15.3 ChangePassword

3.4 Usability

In this subsection we will discuss the usability requirement of *Integra* . The requirements are organized by user type.

3.4.1 Administrator

An experienced System Administrator (we intend here a System Administrator proficient with Unix, Apache, MySQL, Perl with at least 2 years experience) shall be able to install the system and configure the users for the first time in 1 day with the help of the User's manual.

3.4.2 Project leader

An experienced Project leader (with a degree in computer science and prior exposure to project management tools) shall be able to use all the functions provided to him in 3 days with the help of the User's manual.

3.4.3 Project manager

An experienced Project manager (no particular degree needed, exposure to team management tools, basic usage knowledge of Windows, OSX, or Linux and web) shall be able to use all the functions provided to him in 4 days with the help of the User's manual, or in 1 days following a training session with a qualified *Integra* trainer.

3.4.4 Team leader

An experienced Team leader (same requirements as Project leader) shall be able to use all the functions provided to him in 2 days with the help of the User's manual.

3.4.5 Team member

A Team member with programming experience and no exposure to management tools shall be able to use all the functions provided to him in 2 days with the help of the User's manual.

3.5 Reliability

In this subsection we will discuss all the reliability needs of *Integra* . Since the term reliability is not always clear, we define it as *the probability of a failure-free system* considering failure in a broad perspective (unexpected behavior).

3.5.1 Data protection

It shall be possible to configure the system in such a way that all the stored data is automatically saved in more that one single server.

3.5.2 Concurrent access

The system does not need to consider concurrent access to protected resources. In case of concurrent access the last change shall be accepted.

3.5.3 Simulation mode

All the changes executed in simulation mode shall not be maintained after leaving the simulation mode unless explicitly requested by the user. While one user is in simulation mode, the other users shall not be affected.

3.5.4 Connections

All the connections from outside the DMZ shall be protected and encrypted over SSL.

3.6 Security

In this subsection we address the two main concerns about security.

3.6.1 Sessions

An open session shall expire after 60 minutes of inactivity the connection was opened from within the DMZ. In case of external connection the expiration time fixed after 10 minutes of inactivity.

3.6.2 External interface

The open API of the External interface (see *section 2.1.2.2 on page 13*) shall only allow for the operations available for the users. In specific terms all the rules regarding login, access and preconditions also applies for this interface.

3.7 Performance

We discuss in this session the minimal performance requirements of the system.

3.7.1 Execution time

The time limit for the 80% of executions of any function on a server with at least the following properties:

- CPU: P4 3Ghz;
- RAM: 2 GB DDR;

- HDs: 2 HD SATA I raid 1;
- LAN: 2 Mb/s internet access.

and on a client with at least the following properties:

- CPU: P4 2 Ghz;
- RAM: 512 MB DDR;
- Internet: xDSL/Cable 1000kb/s (down), 50kb/s (up).

shall be less that 1 seconds (only considering the server operations) and 3 seconds (considering server operations and data transfer over internet secured connection).

The time limit for the 95% of executions with the previous configuration shall be less than 1.5 seconds (respectively 5 seconds).

There are nor requirement for the 100% of executions.

3.8 Maintainability

3.8.1 Change of developers

In case the developer team or company should change after the first release of the system, the new developers[2] shall be able to begin working on the system in less than 5 days, if instructed by the old team.

3.8.2 Minor changes

It shall be possible to execute all minor changes[3] (including the update of all the documents) in less than 16 man-hours work[4].

3.9 Design Constraints

No particular design constraint is imposed, except for the target platform of the system. The target platform shall be a standard LAMP (PHP 5 or later, MySQL 4 or later, Perl 5 or later, Apache 2.1 or later) environment, the supported operating systems shall be Windows XP (SP 2 or later), Windows Vista, Linux (kernel 2.2.26 or later), Mac OS X (10.4.1 or later)

3.10 On-line User Documentation and Help System Requirements

The User's manual shall meet the requirements expressed in the Usability subsection (see *section 3.4 on page 24*). It shall be divided in 5 parts, one for every user type (Administrator, Project leader, Project manager, Team leader, Team member); every part shall be independent form the others.

Moreover the application shall include an online version of the User's manual. This version shall allow to search for keywords and be indexed. It shall be available at every moment during the use of the system.

The User's manual shall be written in English. However it shall be possible to translate it in other languages by editing plain text documents.

3.11 Licensing Requirements

The system shall be developed under The GNU General Public License (GPL)[4]. All the produced documents shall follow the same licence.

[2] The general ability of the new developers should match in this case that of the old developers.

[3] As minor changes we intend changes that do not modify the structure of the project, i.e. no changes that influence more than 1 function at the time. An example of minor change is updating the GUI by changing the position of a button, of adding a special treatment for a particular input.

[4] We define man-hours work the sum of the hours spent by every member during the realization of a task.

4 Supporting information

Glossary

DMZ In computer security terminology, a DMZ is a network area that sits between an organization's internal network and an external network, usually the Internet[1].

TSL/SSL Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols which provide secure communications on the Internet for such things as web browsing, e-mail, Internet faxing, and other data transfers. There are slight differences between SSL 3.0 and TLS 1.0, but the protocol remains substantially the same[1].

References

[1] Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/wiki/.

[2] IEEE Std 830-1998, Recommended Practice for Software Requirements Speficications, revision of IEEE Std 830-1993.

[3] Origo, software development, management and distribution platform, http://origo.ethz.ch/index.php/Origo.

[4] The GNU General Public License, http://www.gnu.org/copyleft/gpl.html

Index

Integra , 9

Abbreviations, 9
Absence, 18
Accounts, 14
Acronyms, 9
Administrator, 9, 24
API, 10
Assumptions, 16

Changes, 26
Classes, 17
Concurrency, 25
Constraints, 26

Data, 25
Definitions, 9
Dependencies, 16
Developers, 26
Document overview, 11
Document Purpose, 9
Document Scope, 9
Documentation, 26

Enter, 21
Errors, 19
Execution time, 25
Exit, 21
External interface, 13, 17

Format, 19
Funcitons, 18
Functions, 14

Global functions, 15

Input, 18, 19
integration, 12
Internal error, 19

License, 26

Maintainability, 26
Memory, 13

Origo, 12
Origo interface, 13
Overflow, 19

Performance, 25
Preconditions, 19
Product perspective, 12
Project, 14
Project leader, 9, 24
Project manager, 9, 24
ProjectLeader, 17
ProjectManager, 17

Reliability, 25

Simulation mode, 25
Skills, 10, 18
SMS, 12
Software interfaces, 13
SRS, 11
Staff, 15
Staff overview, 12
Standard Task view, 12
Standards, 19
Subtasks, 11

Task, 10, 18
Team, 18
Team leader, 9, 24
Team Member, 17
Team member, 9, 25
TeamLeader, 17
Teams overview, 13

Usability, 24
Use, 15
User, 9, 17
User interfaces, 12
User's manual, 26
Users, 16

WAP, 12