HoliDes

Holistic Human Factors **Des**ign of Adaptive Cooperative Human-Machine Systems

HoliDes Platform Builder requirements

Project Number:	332933
Classification:	Confidential
Work Package(s):	WP1
Milestone:	MS5
Document Version:	V5.0
Issue Date:	September 2016
Document Timescale:	Project Start Date: October 1, 2013
Start of the Document:	Month 16
Final version due:	Month 21
Deliverable Overview:	**Main document**: D1.7–HF-RTP Version 2.0 inc. Methodology and Requirements Analysis update. <Confidential>
Compiled by:	Jordi Fonoll - ATOS
Authors:	Jordi Fonoll - ATOS
Reviewers:	Morten Larsen - AWI, Fabio Tango - CRF
Technical Approval:	Jens Gärtner, Airbus Group Innovations
Issue Authorisation:	Sebastian Feuerstack, OFF

DISTRIBUTION LIST		
Copy type[1]	Company and Location	Recipient
T	HoliDes Consortium	all HoliDes Partners

[1] Copy types: E=Email, C=Controlled copy (paper), D=electronic copy on Disk or other medium, T=Team site (AjaXplorer)

RECORD OF REVISION		
Date (DD.MM.JJ)	Status Description	Author
15.01.2015	First version	Nacho González - ATOS
22.01.2015	Completed first round of no functional requirements	Nacho González - ATOS
07.05.2015	Completed uses cases and functional requirements	Jordi Fonoll - ATOS
29.05.2015	Included new functionalities for the Platform Builder proposed in mini-workshop in Madrid	Jordi Fonoll - ATOS
10.06.2015	Version 1.0	Jordi Fonoll - ATOS
28.10.2015	New design based in CSS bootstrap	Jordi Fonoll - ATOS
04.11.2015	Version 1.1	Jordi Fonoll - ATOS
06.11.2015	Spelling corrections	Jordi Fonoll – ATOS Nacho Gonzalez - ATOS
09.11.2015	Minor corrections and types	Nacho González - ATOS
16.12.2015	Added new tool chain to my projects. PB app version 2.0	Jordi Fonoll - ATOS
04.04.2016	Added connectivity functionality. PB app version 3.0	Jordi Fonoll - ATOS
09.05.2016	Added regulations functionality. PB app version 4.0	Jordi Fonoll - ATOS
17.08.2016	Added adaptation framework in the functional requirements and modified GUI screens	Jordi Fonoll - ATOS
02.09.2016	External Review	Morten Larsen, AWI, Fabio Tango, CRF
09.09.2016	Final version	Sara Sillaurren, TEC

Table of Contents

List of Figures

1 Introduction

1.1 Purpose

The purpose of this document is to specify the software requirements and a logical model of the HoliDes Platform Builder system in a clear and consistent manner.

1.2 Scope

The scope of this document is the elicitation of the specific requirements that the Platform Builder development will accomplish.

1.3 Definitions, acronyms and abbreviations

Acronym	Meaning
HoliDes	Holistic Human Factors Design of Adaptive Cooperative Human-Machine Systems
Business Process	It is the system life cycle development and management process that represents a whole given process, covering development phase and product management phase. Life cycle management comprises tasks of product management process as: traceability, configuration and versioning, user and process management. In the other hand, life cycle development covers tasks of development process (as requirements elicitation and formalization, design, implementation, validation and verification).
RTP	Reference Technology Platform
PB	Platform Builder
MTTs	Methods, techniques and tools

2 General description

2.1 Platform Builder perspective

The system covers the proposal done in WP1 as a:

- Solution for improving the configuration and installation capabilities of the Holides RTP.
- Based on a Business Process Description.
- The result will be a RTP configuration for a specific domain development project.

The result can be one or more MTTs proposed by Platform Builder engine taking in account the filters added by user previously. Of course the MTTs have a connection between them; all this information has been stored previously in a database.

2.2 Platform Builder functions

The functionalities included in the Platform Builder app are:

- Platform Builder includes a search engine by parameters described in this document.
- Platform Builder provides a web interface with access only for the users by user/password, given previously by system administrator.
- Platform Builder proposes a list of MTTs as a final result.
- Users can choose specific MTT within the list of MTTs, this action is called *Instantiation*.
- Possibility to export data by MTTs selected.
- Tool chain build by phases in a HF-RTP instance.
- Possibility to generate statistics for the projects created by user.
- A graphical environment displaying the phases for each MTT.
- A tool to integrate the PB with AdCos, adaptation framework.

2.3 User characteristics

The Platform Builder app is prepared to cover different types of user rights.

The role *Administrator* has been created to manage all user rights and how applications can access the different users.

User Human Factor: can use the basic functions into the Platform Builder app. Filtering HF issues and receiving a list MTTs. Possibility to export information to external files.

User profile: all users has their own data updated in the HoliDes database and can modify their personal data by themselves.

2.4 General constraints

Some constraints should be considered:

- Platform Builder doesn't cover access for a "no HF users", only for authorization users.
- The parameters included in the selection screen should be introduced in the database before, Platform Builder doesn't accept parameters that are not introduced in the database.
- The list of MTTs proposed is not updatable, you can re-select the search again, but not modify the final result.

2.5 Assumptions and dependencies

Platform Builder assumes that all information received from all information received from HF experts and stakeholders is correct, for example the domains included in the input screen.
Platform Builder depends on the connection between MTTs and filters defined by users to propose the correct list of MTTs. Then this information should be stored in the database previously.

The tailoring between MTTS also should be proposed by partners with knowledge in HF issues.

3 User interfaces

3.1 User interfaces

In this section are included the user interfaces existing in the Platform Builder web application.
This information is detailed in the HoliDes_Platform_Builder_user_handbook_v1.1 document

The main screen has the user/password section to access to the Platform Builder app.

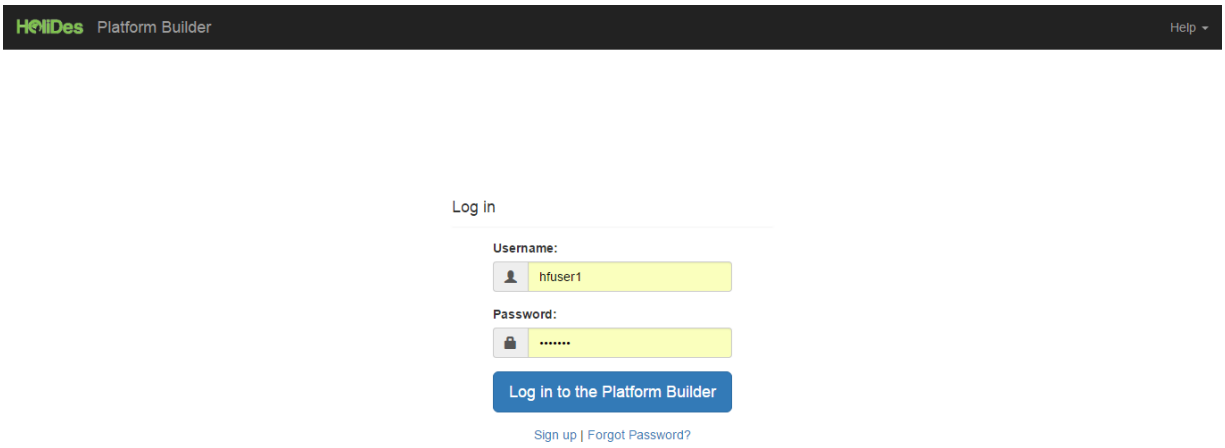


Figure 1 PB "Main" screen

Accessing to Platform Builder the input screen appears the different options to select, in order to retrieve the list of MTTs.

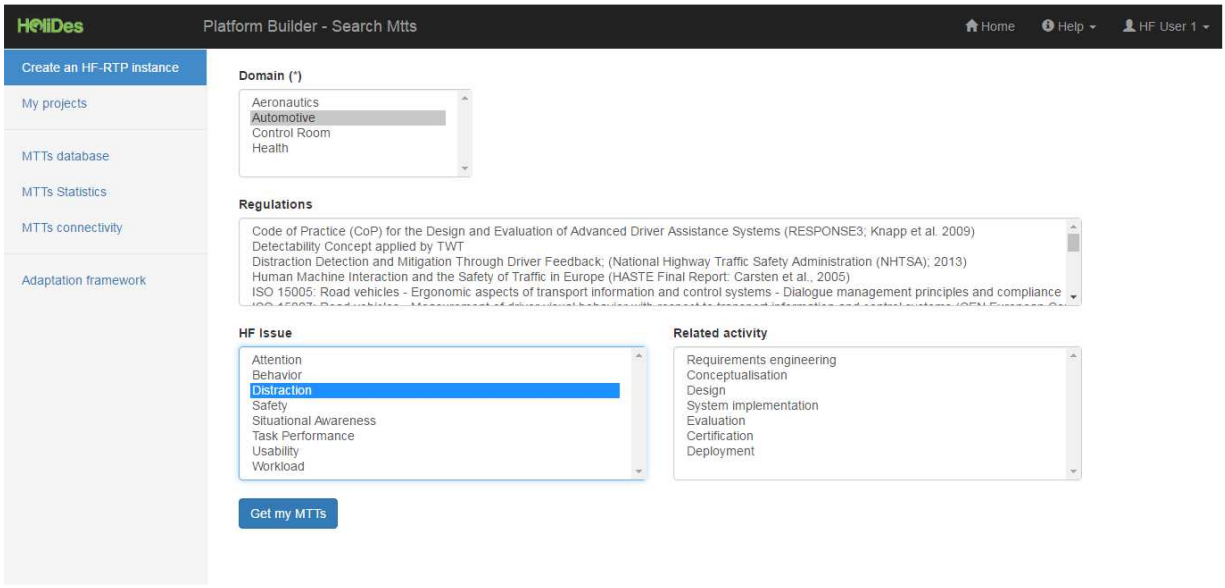


Figure 2 PB "Search MTTs" screen

Showing the final list of MTTs proposed by Platform Builder app:

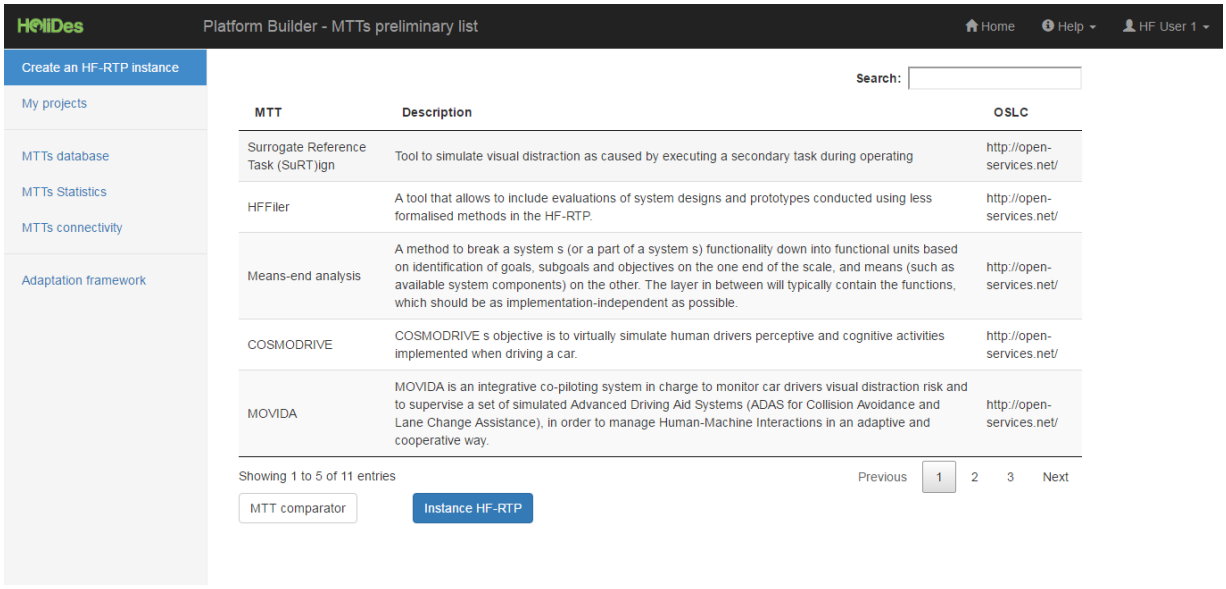


Figure 3 PB "Preliminary list" screen & tool comparator

Selecting one or more MTT/MTTs the user can create an instance of MTTs.

This screen will show the final result of the HF-RTP instantiation process. There are information related with data entered by the user, and the list of MTTs resulting:

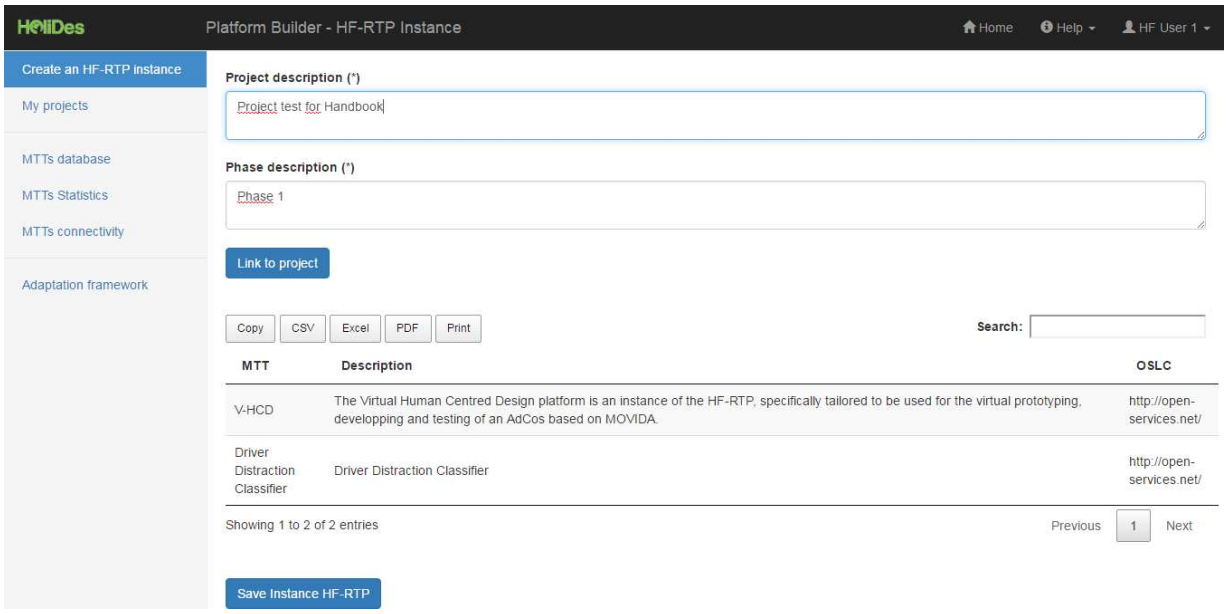


Figure 4 PB "HF-RTP instance" screen

Getting the user projects:

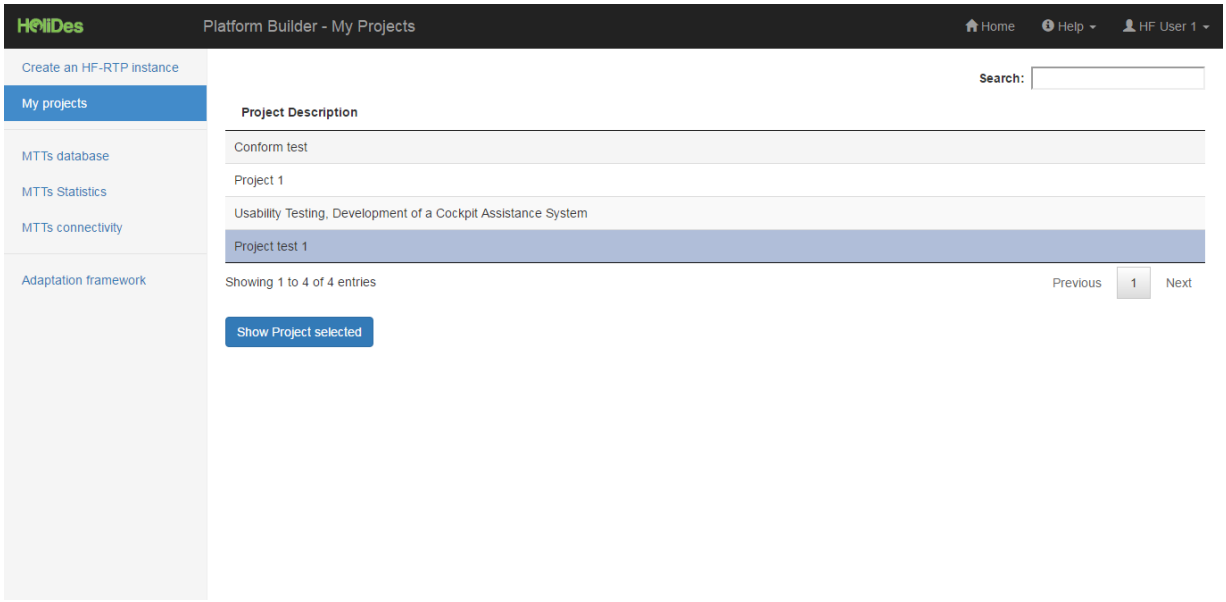


Figure 5 PB "My projects" screen

Showing the details by project:

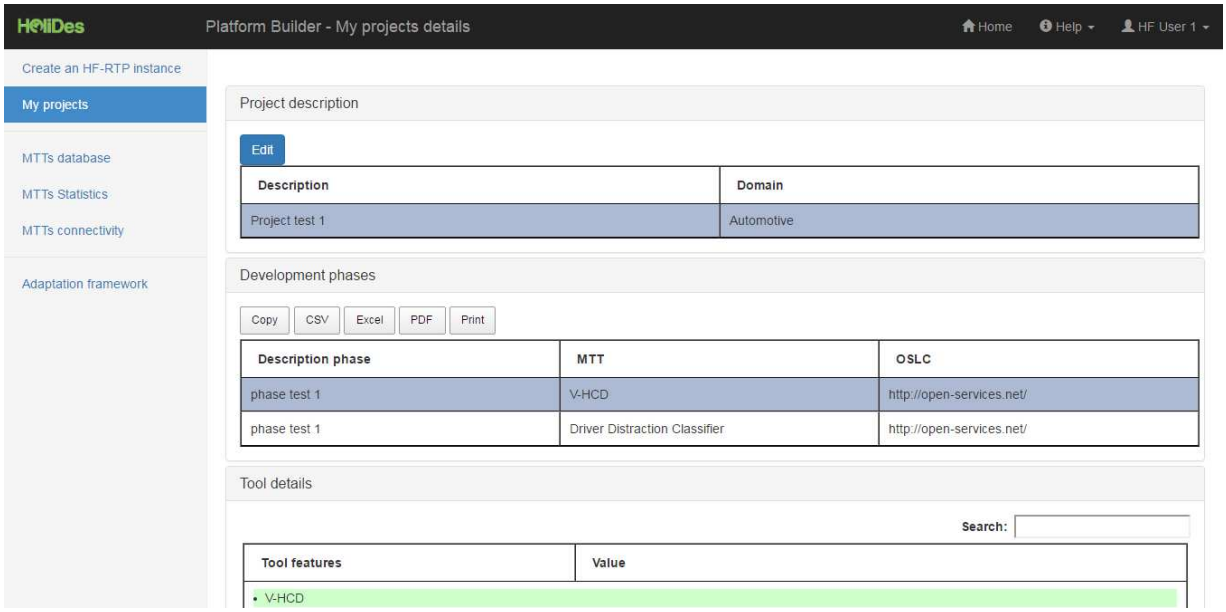


Figure 6 PB "My project details" screen

MTTs administration tool to insert new MTTs to the Platform Builder database:

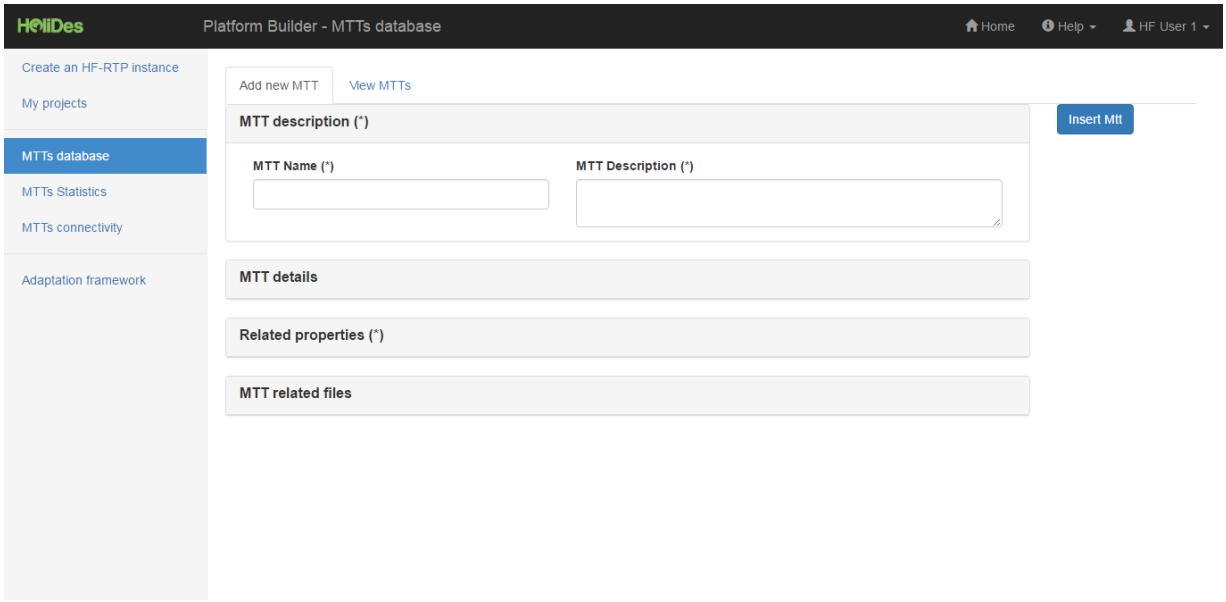


Figure 7 PB "MTT database" tool screen

Statistics of number of MTTs used by the user:

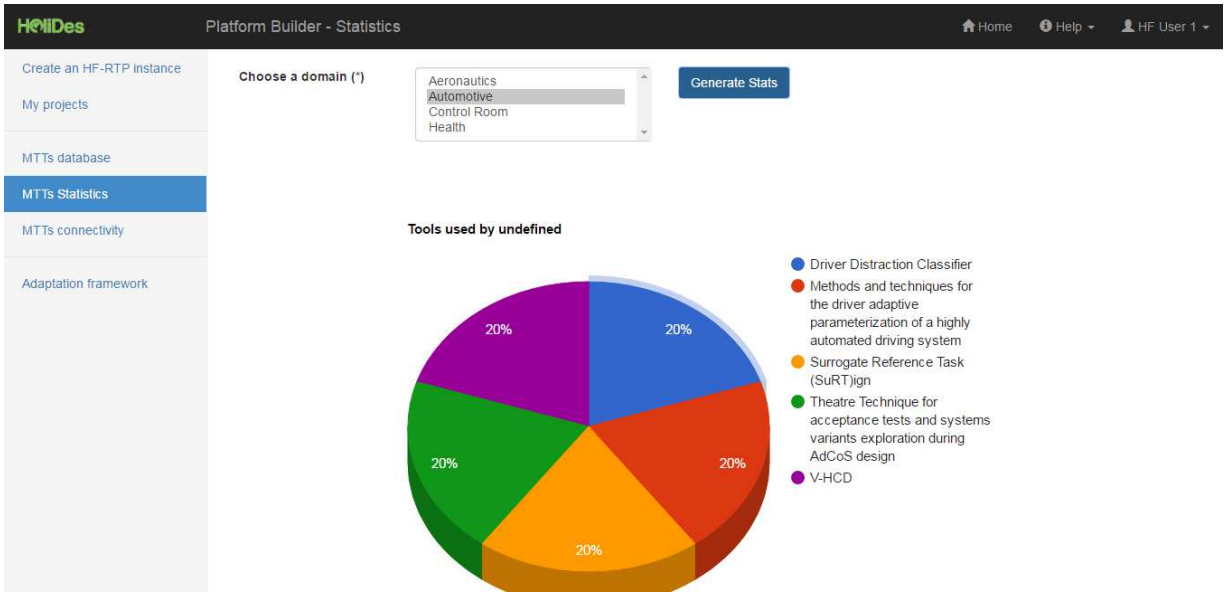


Figure 8 Platform Builder statistics screen

MTTs connectivity, shows the MTTs in a graphical environment(rectangular shapes) located in different related activities(anaysis, certification, design, conceptualization, evaluation, deployment and system implementation):

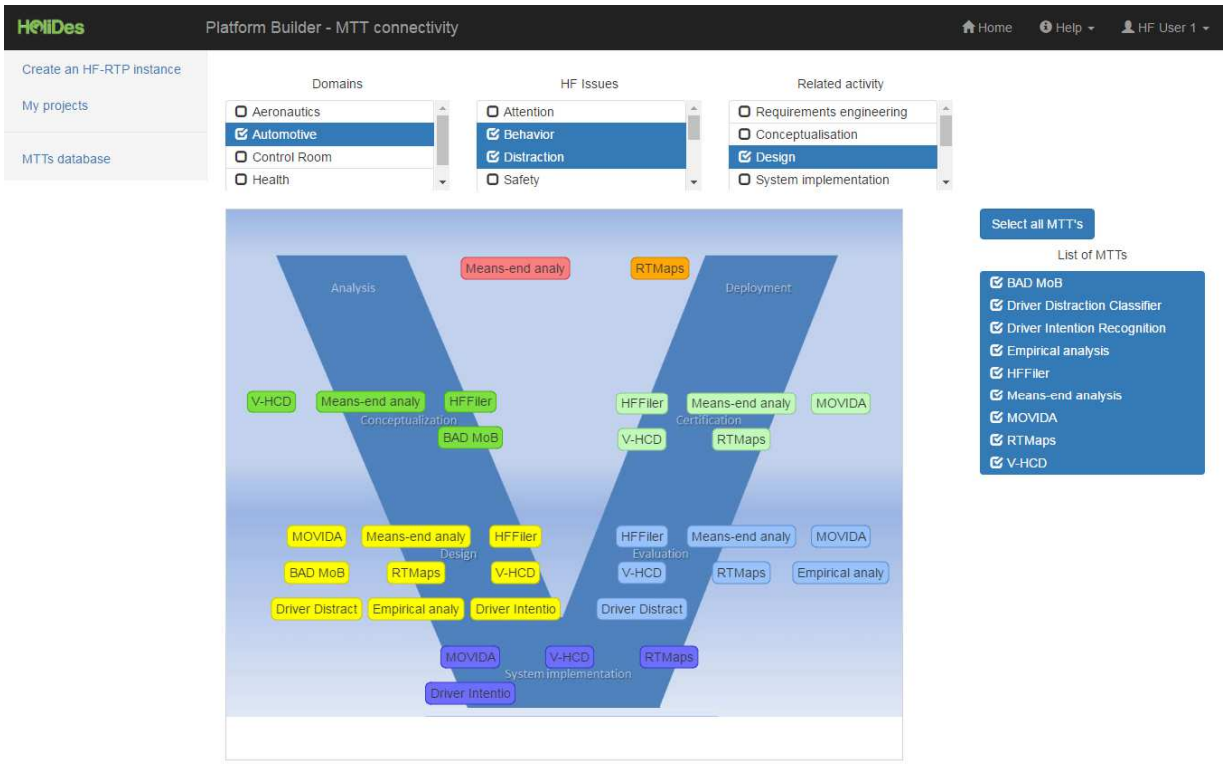


Figure 9 MTTs connectivity screen

3.1.1 Communication interfaces

The application follows the HTTP communication to show the interfaces in a web format. Using Tomcat Server as a web server, uploading the web in www.holides.eu/PlatformBuilder URL and communicating with database by MySql server port.

The picture below is a graphic example of this kind of communication interface:

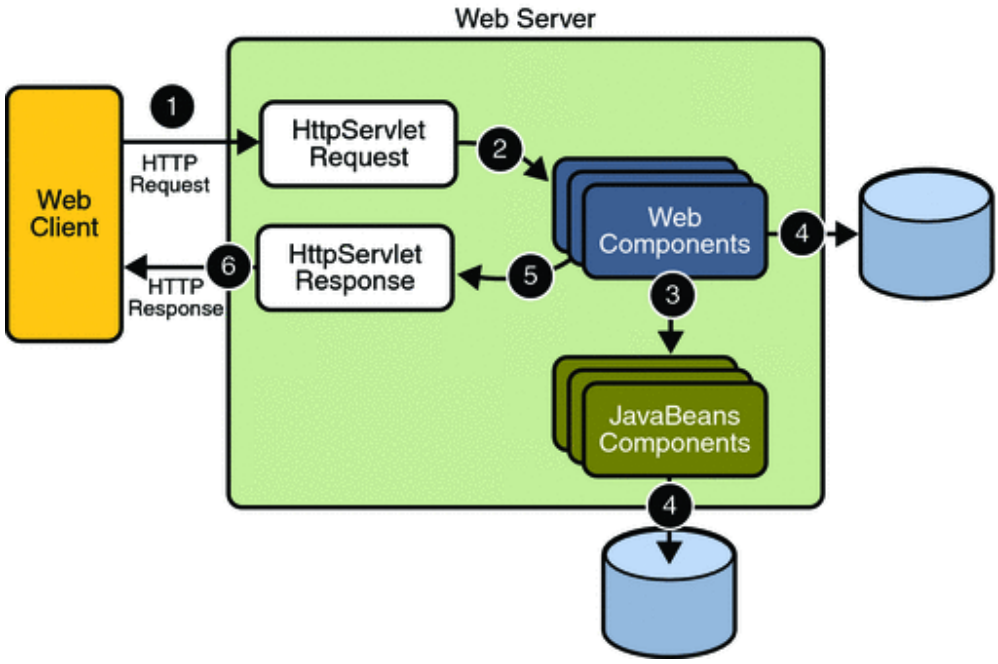


Figure 10 Web Server Architecture

3.2 Functional requirements

This sub section contains the functional requirements. These kind of requirements are specific for Platform Builder app and should accomplish to make the correct performance.

Requirement ID	REQ-F1	**Type**	functional
Name	Filtering		
Description	Show preconditions for use of MTTs. Using: filters proposed by HF experts. "HF Issues", "Domains", "Regulations" and "Related Activities".		

Requirement ID	REQ-F2	**Type**	functional
Name	Rights		
Description	Users can access to the Platform Builder app including their user/password. Different user rights are provided, one for user.		

Requirement ID	REQ-F3	**Type**	functional
Name	Result		
Description	List of MTTs is proposed by the Platform Builder app, searching by filters selected.		

Requirement ID	REQ-F4	**Type**	functional
Name	Compare		
Description	Tool to compare the different features for each MTT. Helping the users to choose a MTT.		

Requirement ID	REQ-F5	**Type**	functional
Name	Select		
Description	Select one or more MTTs proposed by Platform Builder to make your instantiation.		

Requirement ID	REQ-F6	**Type**	functional
Name	Export		
Description	Possibility to export all the results in some format files in order to export all the information to other platforms.		

Requirement ID	REQ-F7	**Type**	functional
Name	Storing		
Description	Possibility to store your projects in the database and get these when you want to.		

Requirement ID	REQ-F8	**Type**	functional
Name	Search Engine		
Description	Help to users to find MTTs in the list of categories proposed.		

Requirement ID	REQ-F9	**Type**	functional
Name	Statistics		
Description	Statistics for Platform Builder: how many MTTs are chosen by users, domains, and other entities in the instantiation RTP		

Requirement ID	REQ-F10	**Type**	functional
Name	Connectivity		
Description	Provide a graphical environment to see the connection between MTTs.		

Requirement ID	REQ-F11	Type	functional
Name	AdCos Integration		
Description	Possibility to connect the Platform Builder with some AdCos.		

3.3 No functional requirements

No functional requirements place constraints on how the Platform Builder will accomplish the functional requirements.

Validation ID	REQ -NF01	Type	Non functional
Name	Accesibility		
Description	The Platform Builder can be used by people with disabilities. More specifically, people with disabilities can perceive, understand, navigate and interact with the Platform Builder.		

Requirement ID	REQ-NF02	Type	Non functional
Name	Capacity, current and forecast		
Description	The capacity of the solution should be planned for an unexpected increase of resources needed.		

Requirement ID	REQ-NF03	Type	Non functional
Name	Compliance		
Description	The Platform Builder may be developed in compliance with specifications created by reference industry bodies, such as the IETF.		

Requirement ID	REQ-NF04	Type	Non functional
Name	Documentation		
Description	All the software engineering process of the Platform Builder should be documented.		

Requirement ID	REQ-NF05	Type	Non functional
Name	Disaster recovery		
Description	Security measures aimed to prevent the lack of access due to different kinds of disaster or unexpected contingencies.		

Requirement ID	REQ-NF06	**Type**	Non functional
Name	Efficiency		
Description	The Platform Builder should accomplish is mission in an efficient and optimal way.		

Requirement ID	REQ-NF07	**Type**	Non functional
Name	Effectiveness		
Description	The Platform Builder should be able to achieve its objectives and the extent to which targeted problems are solved.		

Requirement ID	REQ-NF08	**Type**	Non functional
Name	Extensibility		
Description	If is needed, the application developed can be extended and more features added in the future.		

Requirement ID	REQ-NF09	**Type**	Non functional
Name	Fault tolerance		
Description	Errors should be considered and managed in an optimal way.		

Requirement ID	REQ-NF10	**Type**	Non functional
Name	Interoperability		
Description	Components developed in different technologies should be able to communicate their selves.		

Requirement ID	REQ-NF11	**Type**	Non functional
Name	Maintainability		
Description	The maintenance of the Platform Builder should be a continuous process and easy to achieve.		

Requirement ID	REQ-NF12	**Type**	Non functional
Name	Privacy		
Description	The personal data of the users should be protected with privacy mechanisms.		

Requirement ID	REQ-NF13	**Type**	Non functional
Name	Portability		
Description	The functionalities of the Platform Builder should work in the		

	same way for different computing platforms.

Requirement ID	REQ-NF14	**Type**	Non functional
Name	Quality		
Description	The Platform Builder app should comply to a given design based in the functional requirements described in this document.		
Requirement ID	REQ-NF15	**Type**	Non functional
Name	Reliability		
Description	The Platform Builder app should be able to be tested as failure-free for a specified period of time in a specified environment.		

Requirement ID	REQ-NF16	**Type**	Non functional
Name	Resilience		
Description	The Platform Builder should be a resilient system. This means that it can be hot in a critical component and recover and come back for more in a known, bounded, and generally acceptable period of time.		

Requirement ID	REQ-NF17	**Type**	Non functional
Name	Response time		
Description	The Platform Builder should use an acceptable time for responding the request of a final user.		

Requirement ID	REQ-NF18	**Type**	Non functional
Name	Robustness		
Description	The Platform Builder should be able to cope with errors during execution time.		

Requirement ID	REQ-NF19	**Type**	Non functional
Name	Scalability		
Description	The Platform Builder should be able to handle a growing amount of work in a capable manner or to be enlarged to accommodate that growth.		

Requirement ID	REQ-NF20	**Type**	Non functional
Name	Security		

Description	Security constraints should be covered in all the software engineering process of the Platform Builder.

Requirement ID	REQ-NF21	**Type**	Non functional
Name	Testability		
Description	The developed Platform Builder should be able to be tested in a set of different environments and technical constraints.		

3.4 Use cases

The use cases defines interaction between a role (actor, in our case the Platform Builder user) and the system (Platform Builder app).

3.4.1 HF-RTP Input Use Case

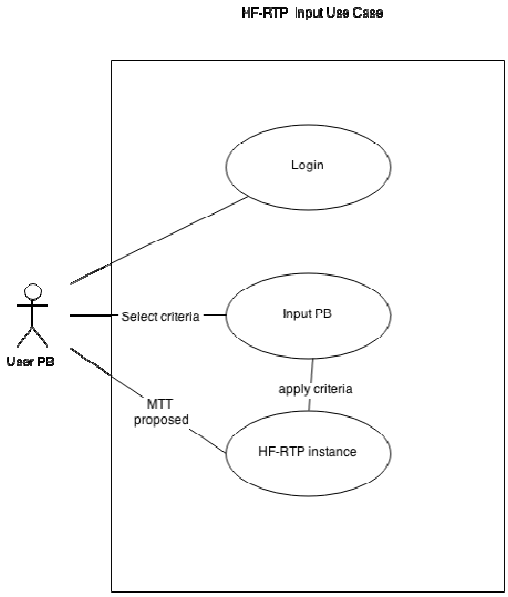


Figure 11 Use Case Input HF-RTP

1. The use case starts when "Platform Builder User" is logged to the with his/her user/password identification.
2. The system checks the data in the database and shows next screen if the credentials are correct.
3. The system displays the data domain from database.
4. The system filters each regulation, HF Issue and related activity depending on the domain selected
5. The user "Platform Builder user" click on the button "Get my HF-RTP" to get the a list of MTTs.
6. The system filters for criteria selected by "Platform Builder User" and obtains the data from database.
7. The system show the screen "Preliminary List" with MTTs proposed based on the filters selected.

3.4.2 MTTs Comparator Use Case

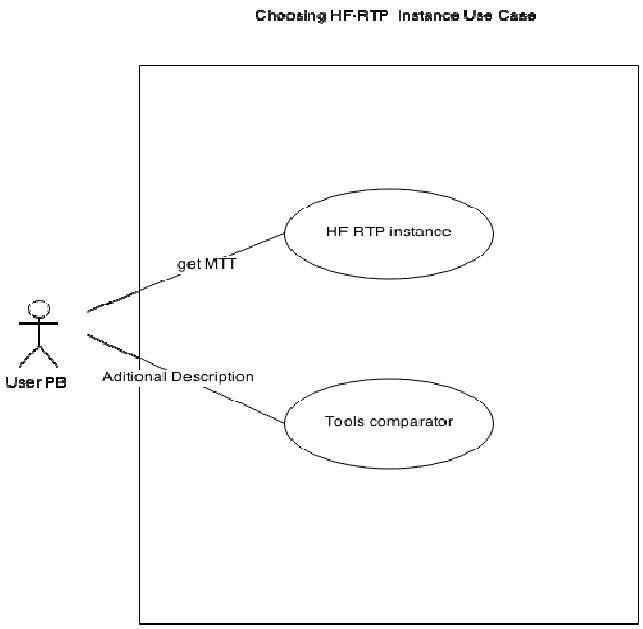


Figure 12 Platform Builder RTP Instantiation Use Case

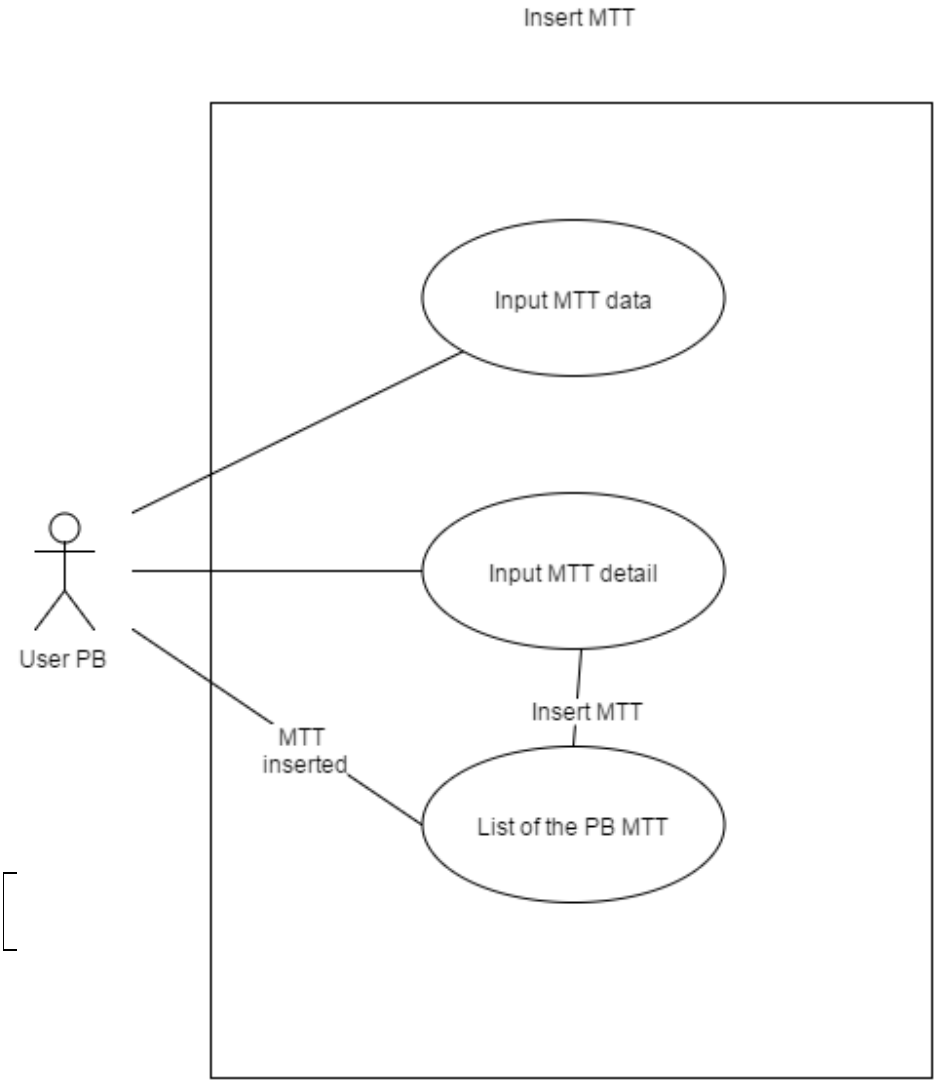
1. The system shows the screen "Preliminary MTTs List" with MTTs proposed based on the filters selected in the "Input HF" Screen.

2. The system shows a comparative between the different MTTs.

3. The system retrieves from database all additional data from each MTT.

4. The system put additional data in a comparative frame, showing what are the advantages and disadvantages for each MTT.

5. The "Platform Builder User" can choose the best MTT/MTTs for their issue, taking into account the data shown in the comparative details frame.



3.4.3 Adding new MTTs

Figure 13 Insert MTT

1- The system shows the MTT administration tool.

2- The Platform Builder User fills the fields MTT name and MTT description.

3- The Platform Builder User adds MTT details.
4- The Platform Builder User clicks Insert MTT.
5- The system checks the information and if it's "OK" include the MTT in the database and shows the list of MTTs with the new MTT included.

3.4.4 Managing projects

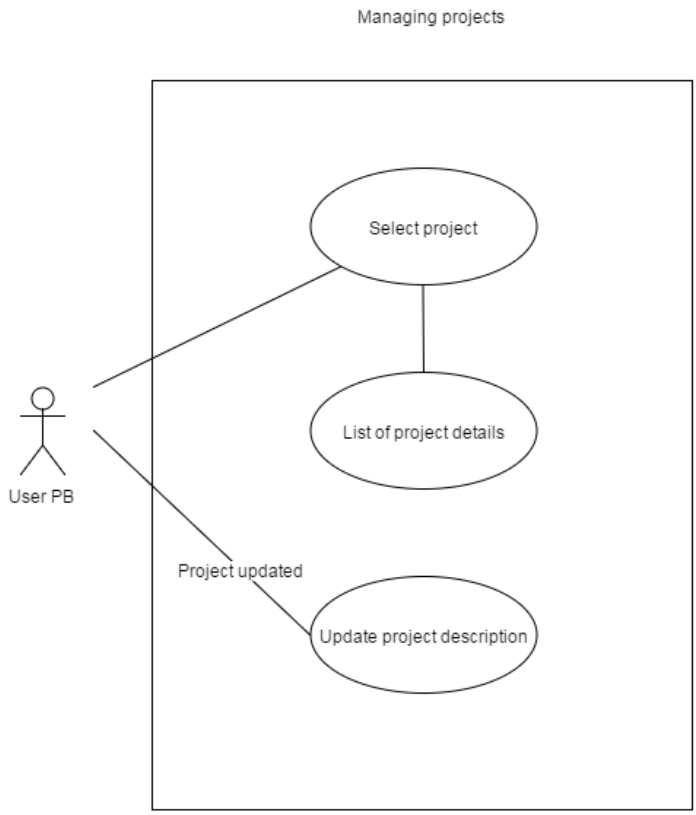


Figure 14 Managing projects

1- The system shows a list of projects saved by the Platform Builder User.
2- The Platform Builder User selects a project and click on show project selected.
3- The system shows the list of project details.
4- The Platform Builder User can update the project description.
5- The system shows the project details with the changes done by Platform Builder user.

3.4.5 Getting statistics

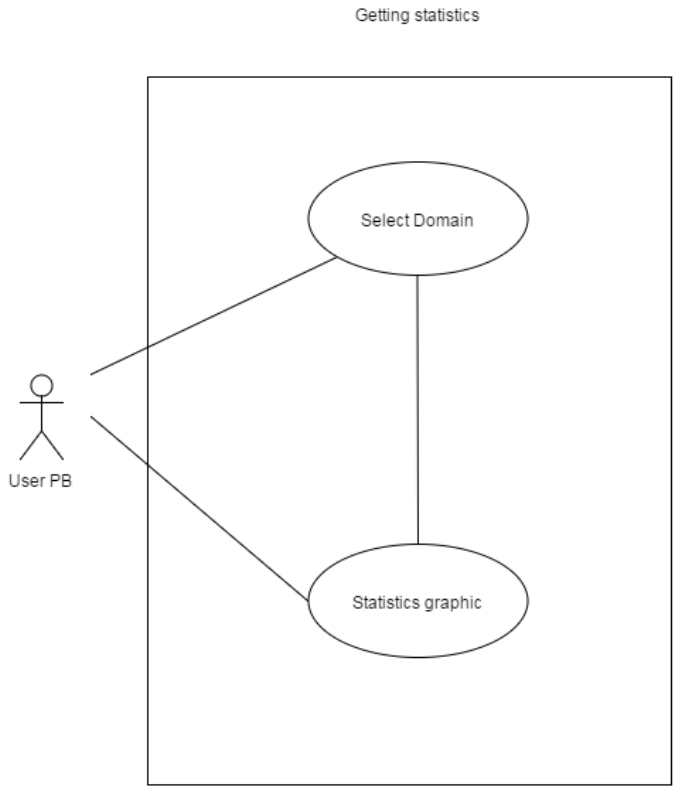


Figure 15 Getting statistics

1- The system shows a list of domains.
2- The Platform Builder User select one domain of the list proposed by the system
3- The system shows a graphic with the number of MTTs used by the user

3.5 Classes/Objects

This is the part of software related with the business layer detailed in the document "Holides Platform Builder functional analysis", it's specific for Platform Builder app containing the relations between classes.

In the picture below is showing the data model, containing the classes used in the software design.

UML Diagram:

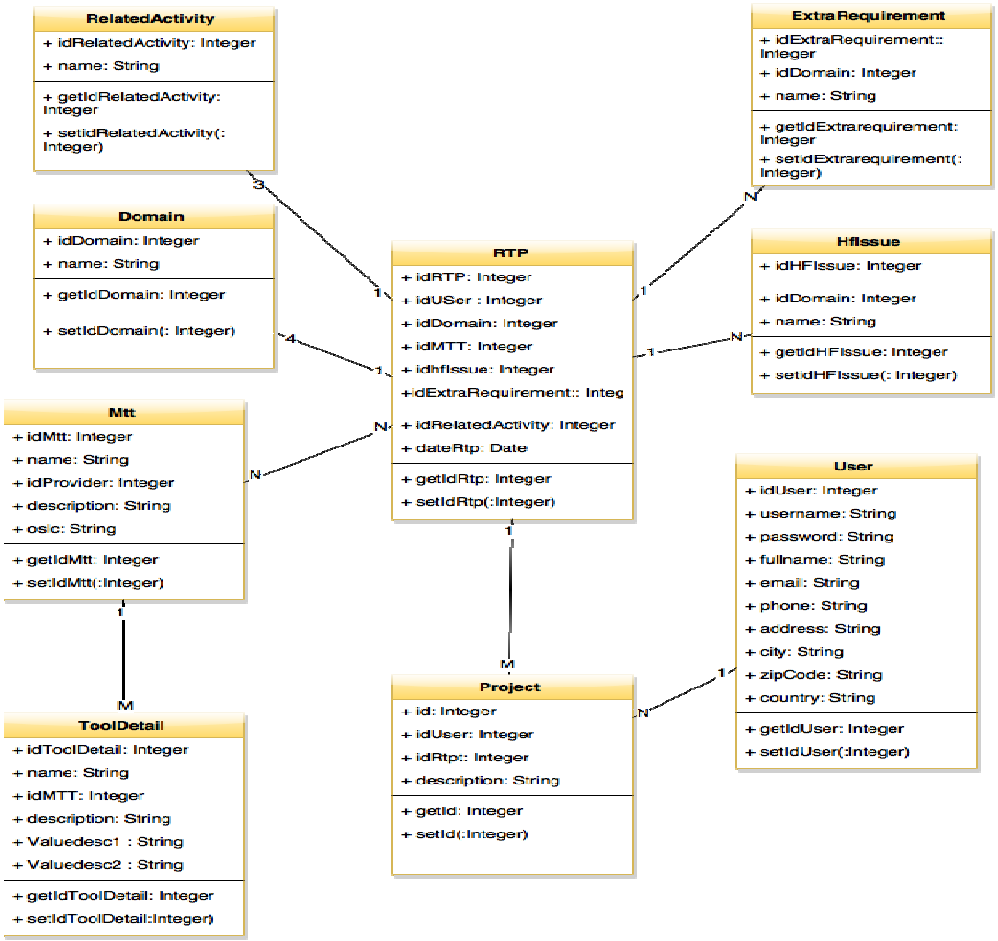


Figure 16 UML Diagram

Below is the detail of the properties and methods by class:

3.5.1 Class Domain

```
public class Domain {

    private Integer id;
    private String name = new String();
    public Domain() {

    }
    public Domain(Integer id, String name) {
           super();
           this.id = id;
           this.name = name;
    }
    public Integer getId() {
           return id;
    }
    public void setId(Integer id) {
           this.id = id;
    }
    public String getName() {
           return name;
    }
    public void setName(String name) {
           this.name = name;
    }
}
```

3.5.2 Class HFIssues

```
public class HFIssue {
       private Integer id;
       private String name = new String();
       private Domain domain;
       public Integer getId() {
              return id;
       }
       public void setId(Integer id) {
              this.id = id;
       }
       public String getName() {
              return name;
```

```
	}
	public void setName(String name) {
		this.name = name;
	}
	public Domain getDomain() {
		return domain;
	}
	public void setDomain(Domain domain) {
		this.domain = domain;
	}
}
```

3.5.3 Class RelatedActivity

```
public class RelatedActivity {
	private Integer id;
	private String name = new String();
	public Integer getId() {
		return id;
	}
	public void setId(Integer id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
}
```

3.5.4 Class MTT

```
public class Mtt {
	private Integer id;
	private String name = new String();
	private String description = new String();
	public Integer getId() {
		return id;
	}
	public void setId(Integer id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
```

```
		this.name = name;
	}
	public String getDescription() {
		return description;
	}
	public void setDescription(String description) {
		this.description = description;
	}
}
```

3.5.5 Class RTP

```
public class Rtp {

	private Integer idRtp;
	private Integer idUser;
	private Integer idDomain;
	private Integer idMtt;
	private Integer idHfIssue;
	private Integer idRelatedActivity;
	public Integer getIdRtp() {
		return idRtp;
	}
	public void setIdRtp(Integer idRtp) {
		this.idRtp = idRtp;
	}
	public Integer getIdUser() {
		return idUser;
	}
	public void setIdUser(Integer idUser) {
		this.idUser = idUser;
	}
	public Integer getIdDomain() {
		return idDomain;
	}
	public void setIdDomain(Integer idDomain) {
		this.idDomain = idDomain;
	}
	public Integer getIdMtt() {
		return idMtt;
	}
	public void setIdMtt(Integer idMtt) {
		this.idMtt = idMtt;
	}
	public Integer getIdHfIssue() {
		return idHfIssue;
	}
```

```
	public void setIdHfIssue(Integer idHfIssue) {
		this.idHfIssue = idHfIssue;
	}
	public Integer getIdRelatedActivity() {
		return idRelatedActivity;
	}
	public void setIdRelatedActivity(Integer idRelatedActivity) {
		this.idRelatedActivity = idRelatedActivity;
	}
}
```

3.5.6 Class ToolDetail

```
public class ToolDetail {

	private Integer id;
	private String name = new String();
	private Integer idMtt;
	private String description = new String();
	private String value = new String();
	public Integer getId() {
		return id;
	}
	public void setId(Integer id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public Integer getIdMtt() {
		return idMtt;
	}
	public void setIdMtt(Integer idMtt) {
		this.idMtt = idMtt;
	}
	public String getDescription() {
		return description;
	}
	public void setDescription(String description) {
		this.description = description;
	}
	public String getValue() {
		return value;
	}
	public void setValue(String value) {
```

```
                this.value = value;
        }
}
```

3.5.7 Class User

```
public class User {
        private Integer id;
        private String username = new String();
        private String password = new String();
        private String fullname = new String();
        private String email = new String();
        private String phone = new String();
        private String address = new String();
        private String city = new String();
        private String zipcode = new String();
        private String country = new String();
        public Integer getId() {
                return id;
        }
        public void setId(Integer id) {
                this.id = id;
        }
        public String getUsername() {
                return username;
        }
        public void setUsername(String username) {
                this.username = username;
        }
        public String getPassword() {
                return password;
        }
        public void setPassword(String password) {
                this.password = password;
        }
        public String getFullname() {
                return fullname;
        }
        public void setFullname(String fullname) {
                this.fullname = fullname;
        }
        public String getEmail() {
                return email;
        }
        public void setEmail(String email) {
                this.email = email;
        }
```

```
	public String getPhone() {
		return phone;
	}
	public void setPhone(String phone) {
		this.phone = phone;
	}
	public String getAddress() {
		return address;
	}
	public void setAddress(String address) {
		this.address = address;
	}
	public String getCity() {
		return city;
	}
	public void setCity(String city) {
		this.city = city;
	}
	public String getZipcode() {
		return zipcode;
	}
	public void setZipcode(String zipcode) {
		this.zipcode = zipcode;
	}
	public String getCountry() {
		return country;
	}
	public void setCountry(String country) {
		this.country = country;
	}
}
```

Analysis models

3.6 Sequence diagrams

This diagram explains the steps to follow by the user to get the MTTs instantiation by the system.
The user gets the entities proposed in the user interface and selects its contents to do a specific search.

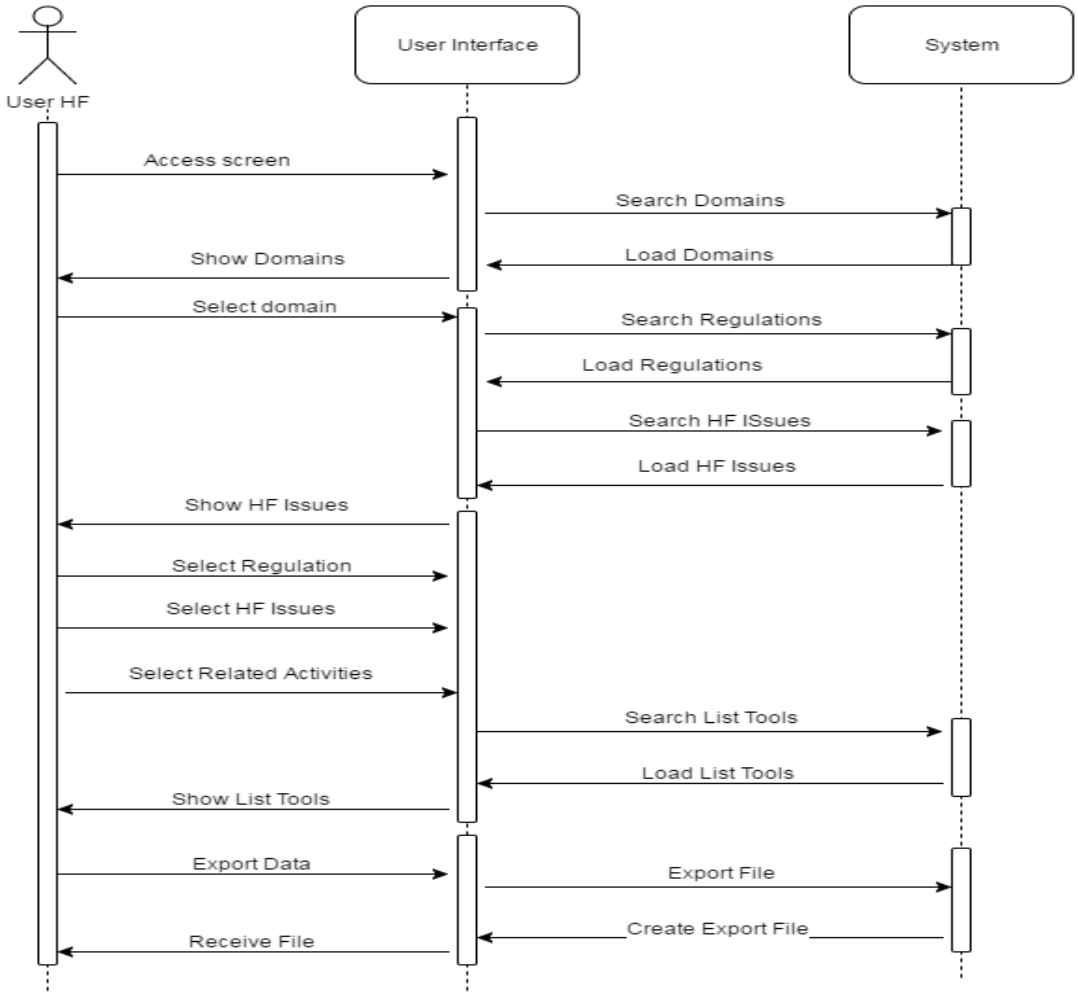


Figure 17 Sequence Diagram

3.7 Data flow diagrams (DTD)

It's a graphical representation of the data flow and it's possible to visualize the data processing in a diagram point of view.

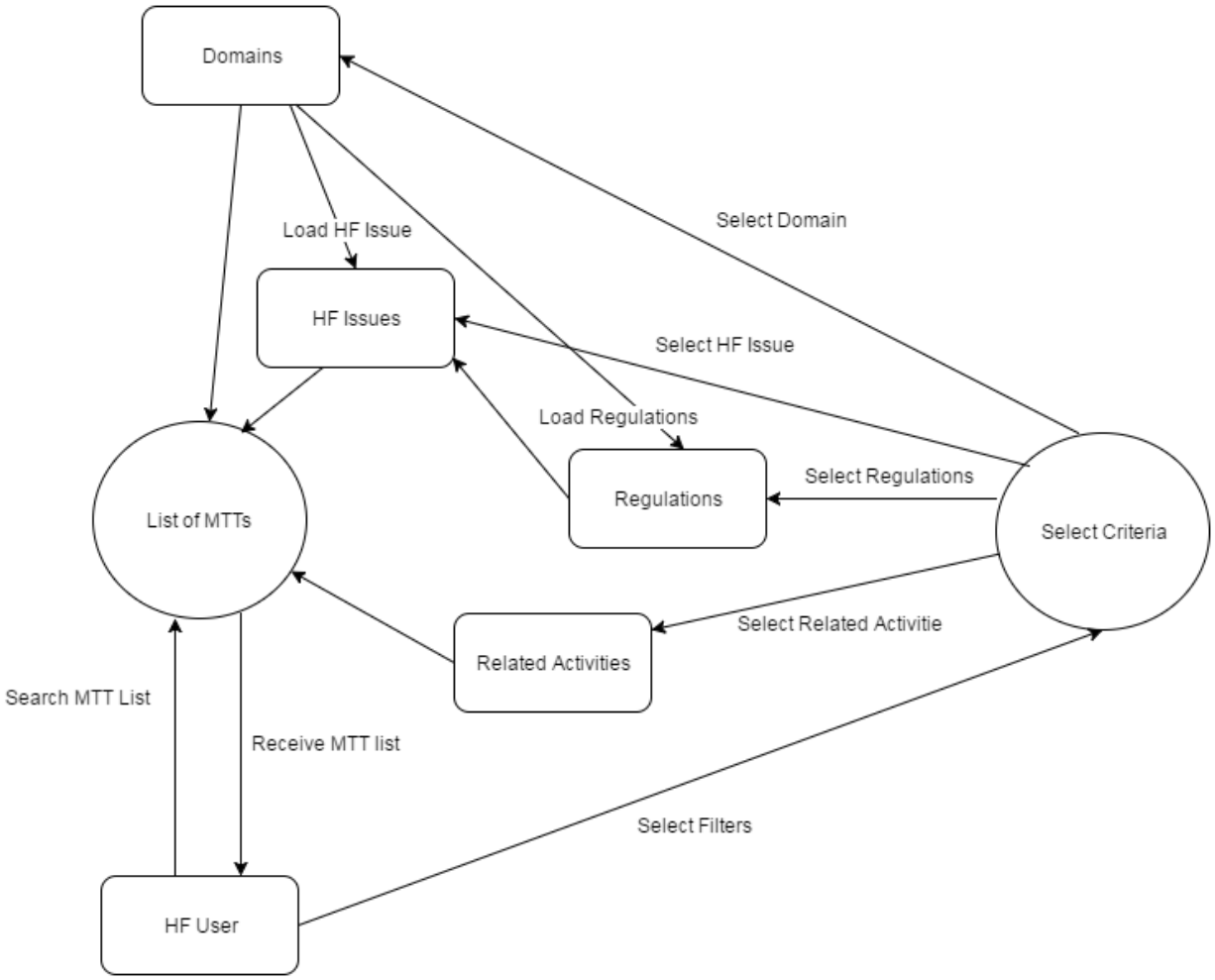


Figure 18 Data Flow Diagram

3.8 State-transition diagrams (STD)

The state-transition diagram describes the behaviour of the system.

Starts when the user creates the search and if it's "OK" the system proposes a MTTs list, if not, the system advises user with a message containing the problem existing.

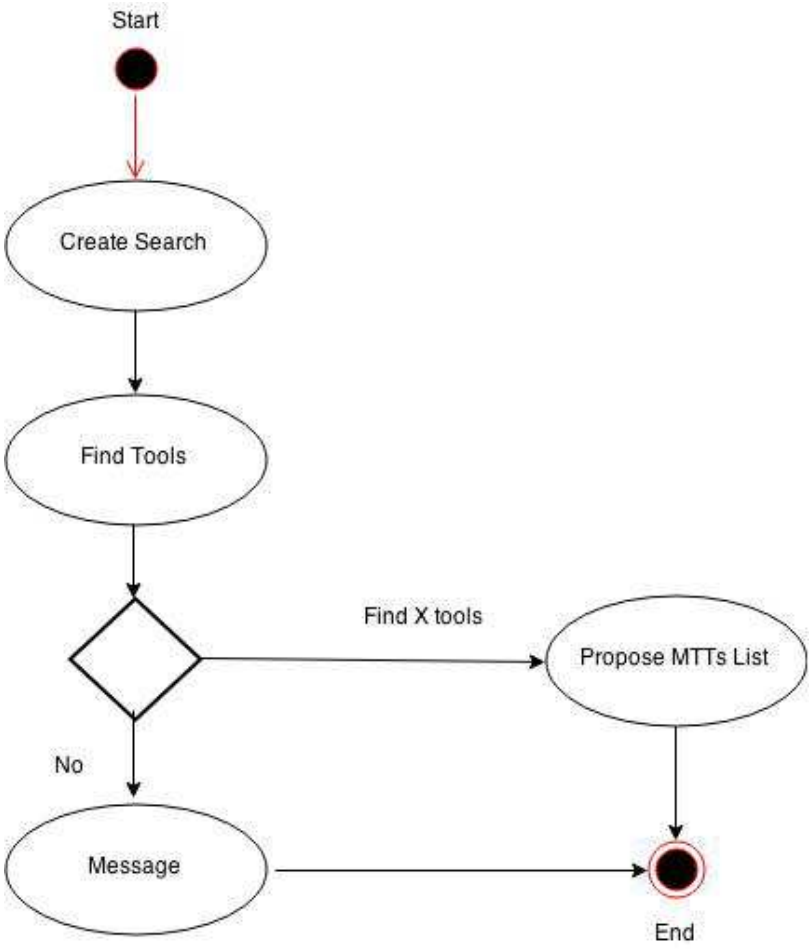


Figure 19 State-Transition diagram