Technical University of Madrid

Computer Science Faculty

Development of a step based structure to create a Rapid Assessment for Markets application

Author:
Fernando Suárez

Supervisor:
Nelson Medinilla

Contents

Abstract

This project provides a Liferay portlet with the basic structure to create step based web applications. It will be the base for a Rapid Assessment for Markets web application which will help humanitarian workers to conduct market assessments in order to better assist disaster affected populations. The development has been carried out following modern software methodologies which provide a development and testing framework to ease further development and to ensure software quality objectives achievement.

1 Introduction

The humanitarian world is being increasingly specialized, and since the beginning of the century a significant number of tools have been being created in order to increase effectiveness, efficiency, impact and thereby quality of humanitarian assistance. Moreover, the empowerment of target populations has fortunately gained in importance, as well as the concepts of *dignity*, *sustainable livelihoods* or *population choices*. In this context, humanitarian assistance has increased its contextualization leaving the old-fashioned forms of charity and promoting recovery or development actions as far as possible guided by the target populations.

Thus, cash transfer interventions have acquired importance[1] inasmuch as it is a tool which really empowers target population since they are the beneficiaries who decide their priorities to recover from a disaster and, with a little of guidance if necessary, can take the rains of their future development. But a cash intervention needs, inter alia, functional markets in order to have the intended impact, so during last years some tools have been created in order to better analyze markets in crisis situations, as the best-known EMMA[2]. Rapid Assessment for Markets (RAM)[5] is, furthermore, the RC/RC Movement[3] tool to analyze markets in the emergency phase, and it is essentially derived from EMMA but aimed to carry out market assessments faster and reducing the resources costs (not only economic, but also regarding human and time). Despite that, difficulties have been found to spread RAM and make it available for a large number of volunteers and RC/RC members, so the Livelihoods Resources Centre (an IFRC centre whose mission is to increase awareness and use of effective livelihoods programming strategies within the RC/RC Movement) has considered necessary the creation of a web application to *guide* through RAM and make it *available* for potential practitioners.

This project is a *first phase* of that web application development and it has been conducted paying close attention to the development process and methodology in order to ensure both extensibility and requirements fulfillment.

[1] According to http://www.cash-atlas.org, since 2004 to 2014 cash projects beneficiaries increased in a 262% reaching 12.709.963 in the year 2014

[2] http://www.emma-toolkit.org/

[3] The Red Cross and Red Crescent Movement is comprised by the International Federation of Red Cross and Red Crescent (IFRC), the International Committee of Red Cross (ICRC) and the National RC Societies.

2 Objectives

The main objective of this project is to carry out the first phase of the development of a web tool to help practitioners to carry out Rapid Assessment for Markets. At the end of the project a working software should have been created with the basic structure to create on it a Rapid Assessment for Markets web tool which could be expanded with new features in the future. The real power of the web tool will lie on the possibility of following the RAM interactively and improve the web tool in the future to allow users input the data and analyze it within the tool.

As the web tool will be hosted by the Livelihoods Resources Centre[4], it should be embedded in its web site, which has been developed using Liferay Portal CE[5].

A secondary objective of the project is to create a working testing framework which will make easier in the future to guide the development and perform tests.

3 Development Methodology

The software development has intended to use the software engineering techniques and methodologies currently used in software business. As it will be shown, some of the techniques has been adopted during the course of the project in an effort to improve the software development methodology and hence software quality.

Agile Software Development has been practiced since it, as stated in the *Agile Manifesto*[4], come to value some items over other ones, which has been regarded a good practice for this project:

- **Individuals and interactions over processes and tools:** The fact of using methodologies oriented to individuals and interactions means that agile ones are less prescriptive than traditional methods, so it will not only facilitate the comprehension of the process by the non-technical stakeholders - in this case the customer -, but also will be more adaptative allowing to better fit the methodology to the project and team needs.
- **Working software over comprehensive documentation:** Inasmuch as the objective of the project is to do a web tool to guide through Rapid Assessment for Markets, the documentation is not a main objective and the efforts will be aimed mostly to make a working web tool. In despite of this and in order to make a properly documented Final Project, all the development phases and sprints will be well documented.
- **Customer collaboration over contract negotiation:** Since there is not a contractual relationship with the customer and the requirements are not quite clear, the involvement of the customer will be important during the project for the purpose of assuring the software accomplishes the goals.

[4] http://www.livelihoodscentre.org/
[5] https://web.liferay.com/community/liferay-projects/liferay-portal/overview

- **Responding to change over following a plan:** In the same vein, as the requirements and the software design were a little ambiguous on the customer side, changes on the fly might be done, so flexible methodologies were required.

3.1 Implementing agile methodologies

For the purpose of the project it has been founded more useful to combine techniques from agile (mainly Scrum[20, 8]) and Lean[17] methodologies, rather than following just one strictly. Therefore, the combination of several techniques allows to take the best of each one - as shown in [16] and [14] - and it also allows the development methodology to be adapted to the current project specifications and to evolve according to the project momentum.

To begin with, and in order to structure the development process, the development has been structured in *sprints* following the Scrum framework. In this way, the development is incremental and it allows to better being structured and documented. The process of each sprint is documented in section §6.

3.1.1 Kanban board with Taiga/Trello

In order to organize and visualize the different tasks and the work-flow a *Kanban board*[10] has been created. After some researching Taiga[6] platform was chosen since it allows to combine a Kanban board with backlog and time-line tools. Later, during the development it was migrated to Trello[7] platform, a capture is shown in figure 1, because of its integration with the time management technique *Pomodoro*[11], which was introduced lately to improve productivity.

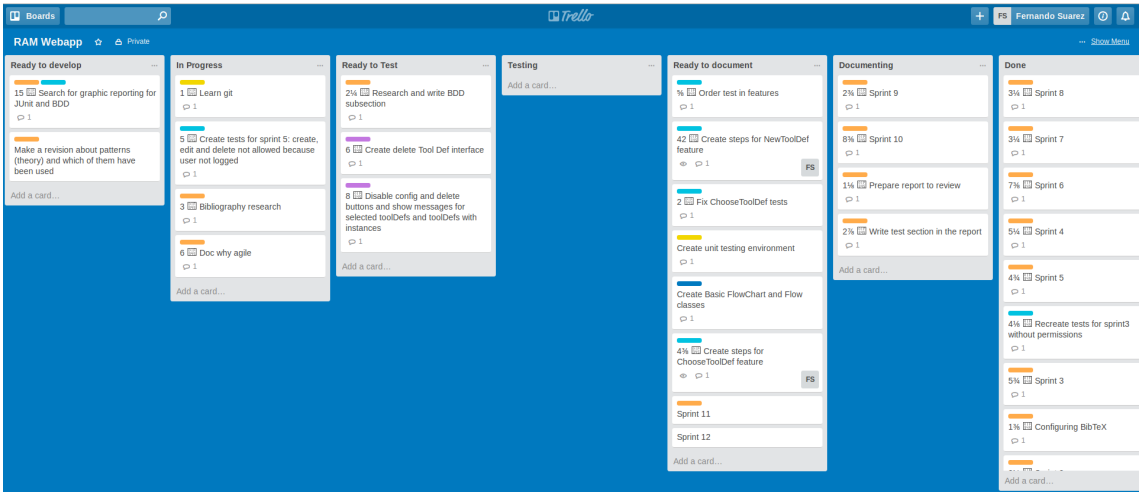


Figure 1: Kanban board powered by Trello

[6] http://www.taiga.io
[7] http://www.trello.com

3.2 Branching and version control

Branching is *"the duplication of an object under revision control (such as a source code file or a directory tree) so that modifications can happen in parallel along both branches"*[7] and is closely linked to version control as the branching strategy defines the versions structure. For the current project, the branching strategy has not been too complex since branches has only been created to add new features keeping the working code untouched. In fact, in the most of the cases a new branch has been created for each sprint and, at the end of the sprint has been merged to the master branch.

3.2.1 Git

To deal with branches and version control Git[8] has been chosen as the best option mainly because of its flexibility to local and remote branching and merging cheaply, in addition to the free and quality service provided by GitHub[9], which helps developers to have distributed git repositories easily.

The git repositories where the sources are located are:

- For the toolBuilder-portlet: https://github.com/fsuarezj/toolBuilder-portlet
- For the testing framework and tests: https://github.com/fsuarezj/toolBuilderTests

3.3 Behavior Driven Development

Behavior Driven Development[1] (BDD) is a development technique built upon Test Driven Development (TDD) and Acceptance Test Driven Development (ATDD) and first described by Dan North in 2006[2]. Is has been adopted for the current project after the third sprint. To better understand BDD it is necessary understanding what TDD and ATDD are:

- **TDD** is a software development process which helps creating well-written and working code. According to [15] *TDD cycle is a three-step process of writing a test, making it pass by writing just enough production code, and finally improving the design by disciplined refactorings*, so the three steps are testing, coding and refactoring. In figure 2 the cycle flowchart is shown.
- **ATDD** is a process based in TDD but including the concept of acceptance tests and highlighting the customer-tester-developer collaboration[18]. Acceptance tests are related to the specifications and check if they are met, so ATDD process is similar to TDD but working the acceptance tests which in fact means improving the collaboration among customer, tester and developer since customer must be involved in acceptance tests definition.

[8] https://git-scm.com/
[9] https://github.com/

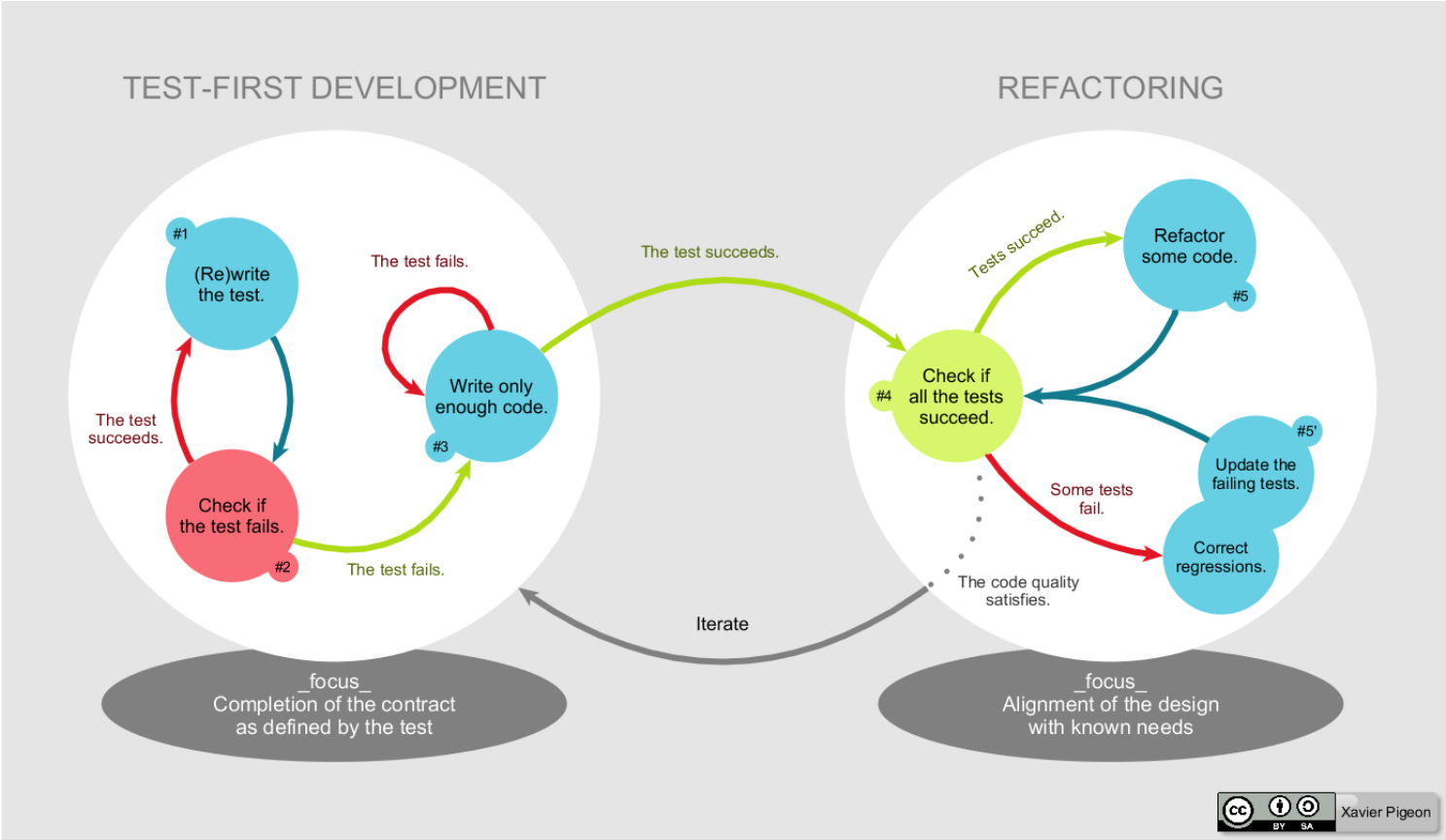


Figure 2: TDD cycle, from [9]

On the other hand, BDD introduces the concept of *behavior* which is a description of what the system *should do* using consistent vocabulary, so the traditional disconnect between business and technology disappears[1]. This increases the involvement of customer in the development, so for this project enhances the opportunity of involvement of future users, who could not have a technical background. Thus, talking about behaviors instead of tests and writing them in a natural language so anyone on the team can read it, improves the communication and the understanding among all the stakeholders.

3.4 Documentation

One of the advantages of BDD is that tests not only check the software, but also document it as they are descriptions of the software behavior. This is very useful because all the documentation related to the software behavior is already written.

Regardless of that, the code should be documented in order to make easier the involvement of new developers in the future, so Javadoc[6] has been used. Javadoc is a tool for generating code documentation by adding Java annotation to the code. The most important objects and methods in the code have been annotated so code will be more understandable.

4 Testing Framework

As it will be explained in section §6, the testing framework was not implemented in the first sprint. For the second sprint just unit tests were taken, while the rest of the tests were taken from third sprint on. In figure 3 the *Agile Testing Quadrants* from [12] are shown. These quadrants help to understand the different test phases and what has been the testing strategy during the project.

4.1 Quadrant 1 tests

Tests in Quadrant 1 are usually taken by the developers and are aimed to ensure the quality of small parts of the software as well as their interactions with other parts. Within the project unit tests has been automatized for some components as in section 6.2, but in many cases and due to the integration of the project in Liferay which will be discussed in section 5.1.1, unit tests were barely taken manually during development and the effort has been made in the automation of next quadrants tests, which allow to test the correct behavior of small parts in an easier way when they are integrated with the whole system.

Regarding the used technologies, JUnit4[10] has been the framework chosen for unit tests, as it has a good documentations and xUnit is one of the most popular testing

[10] http://junit.org/junit4/

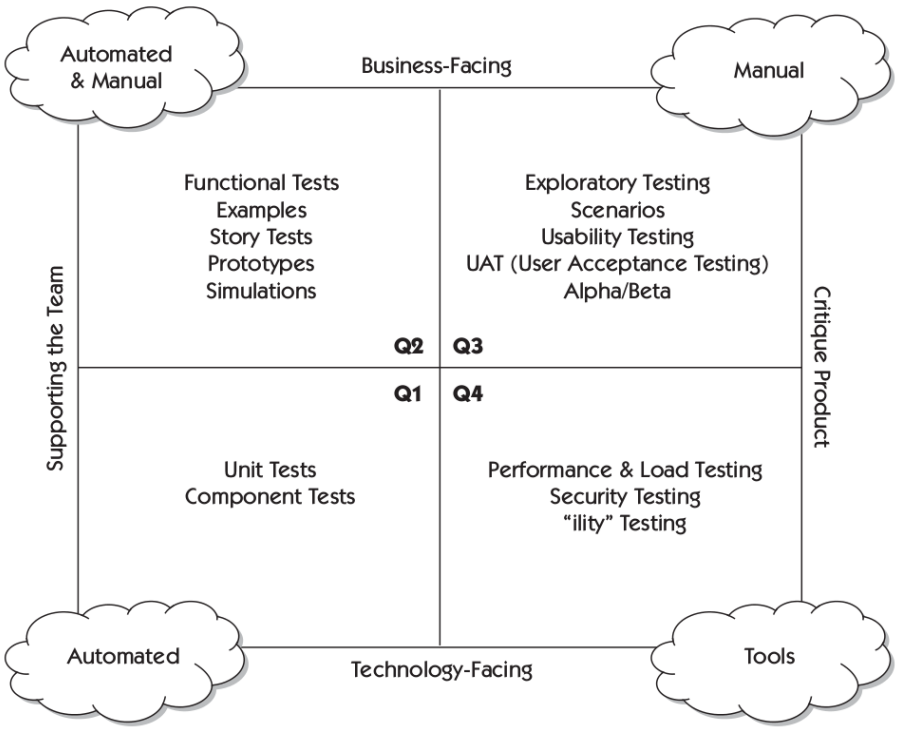


Figure 3: Agile Testing Quadrants

frameworks. In the reporting side, Allure framework[11] has been used because of the good integration with JUnit and the quality and customization of the reports.

4.2 Quadrant 2 tests

Quadrant 2 comprises the Business-Facing tests which address requirements of the software. It includes all the user stories and features desired and should also check for undesired behavior in specific cases. Thus, the BDD tests are included in this quadrant. It is important to understand that even if BDD are acceptance tests, they are different to User Acceptance Tests, which will be explained in the next quadrant.

For this project, the most of the testing efforts has been made in this quadrant as BDD has been the main testing methodology. The nature of the web app, which is embedded in Liferay and uses mature technologies such as Java Server Faces as explained below, implies that the best way to test the software is when it is already embedded and working since the surrounding technologies are not supposed to fail. In this way, the testing efforts are optimized and the quality of the software is ensured.

In order to implement BDD and carry out the tests, two main technologies have been employed in addition to the reporting *Allure* framework: *Selenium* and *Cucumber-JVM*.

[11] http://allure.qatools.ru/

4.2.1 Selenium

Selenium[12] is a framework which automates browsers. In the current project, the *Selenium WebDriver* has been used because it allows to create and automate tests programmatically through several languages, in this case the Selenium Java binding has been the chosen one.

Although many browsers are supported by *Selenium*, just Mozilla Firefox and Chrome have been used. In fact, although Firefox was used in several tests, Chrome has been the main testing browser since Firefox WebDriver had some issues depending on the Firefox version installed in the testing machine.

To improve tests development and scalability a little framework has been created to have separately the WebDriver and the pages handling, so two type of objects have been created:

`DriverFactory` This is a static object in charge of handling the WebDriver. This object creates and destroys the WebDriver and provides it when it is required.

`XyzPage` These objects which usually inherit from `AbstractPage` and where *Xyz* is the identifier of each page, provide the methods to access and handle the different page components.

4.2.2 Cucumber-JVM

Cucumber[13] is one of the most known and used tools to automate tests in Behavior Driven Development, and Cucumber-JVM is its implementation for Java Virtual Machine languages.

As shown in figure 4, Cucumber has different levels which should be defined in order to get the BDD framework working.

4.2.2.1 Business-Facing These levels are the core of the BDD methodology because it is where the software behaviors are defined. As stated in 3.3, behaviors are understood by customer, tester and developer so should be developed in close collaboration among all of them. Consequently, all these levels are written in a natural language so it can be understood by all the stakeholders, as explained below *Gherkin* language, the will be used to ensure that.

[12] http://www.seleniumhq.org/
[13] https://cucumber.io/

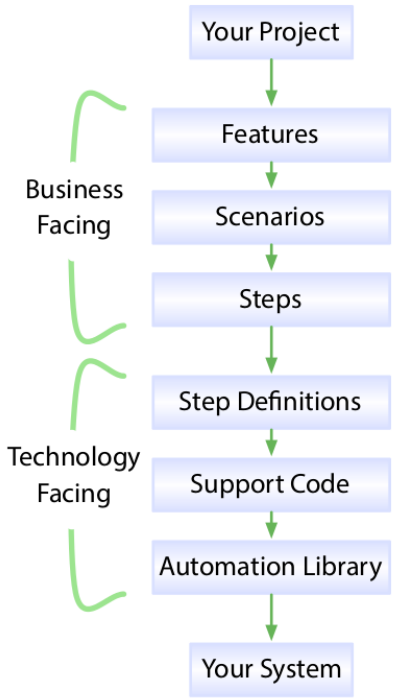


Figure 4: Cucumber testing stack, from [19]

- *Features:* Here is where features are defined. Many authors, as in [18], define features as high level requirements composed by user stories, which are smaller parts usually defined in the way

  `As a <role>, I want to <do something> so that <reason>.`

  But Cucumber does not has a *User Story* level, so it is a good practice to step down a level and define features as user stories naming firstly the feature and later using the `As a/I want to/so that` notation.

- *Scenarios:* Scenarios are the next level and they are akin to use cases. Thus, they are composed by steps and define the behavior of the system in a given use case. To concretely define the behavior, in Cucumber scenarios follow the structure `Given <condition>, when <event/action> then <result>` written in Gherkin language.

- *Steps:* The last level of the Business-Facing section are steps. Steps are single actions that are concatenated in Cucumber so they say the system *what* to do. Each step has its step definition in the Technology-Facing section, so the natural language is translated into a testable language.

As stated above, the main technology to power this business-facing section is the *Gherkin* language as it is the language used for writing Cucumber features, scenarios and steps. It is a natural language with a specific structure and a set of keywords to define features, scenarios and steps. In 1

Listing 1: Example of feature with one scenario written in *Gherkin*

```
Feature: Tool Instance Creation
   As a user
   I want to be able to create a new Tool Instance in the
      system
   So that I can work with that Tool Instance

   Scenario: A logged user creates a new Tool Instance
      Given the user is on the Tool Instances Navigation Page
      And she logs in the system as xx@xx.com with password
         xxx
      When she chooses to create a new tool instance
      And she provides the tool instance name as My New
         Instance
      And she creates the tool instance
      And she returns to the Tool Instances Navigation Page
      Then a tool instance called "My New Instance" exists
```

4.2.2.2 Technology-Facing These levels are related to the system to be tested. In this project, as Cucumber-JVM has been chosen, the most of the code is written in Java, but that choice depends on the kind of software to be tested and the language chosen.

- *Step Definitions:* As explained above these are the translations of the steps into testable procedures, so they say the system *how* to carry out the steps. In this case, as we are using Cucumber-JVM, steps definition are implemented using Java annotations and allows to include regular expressions in order to better define the steps. An example can be shown in the first line of 2. Steps definitions are defined in the `XyzStep` objects, so the classes diagram of the testing framework for the project is shown in figure 5.

- *Support Code:* As steps should be concrete and accurate, the support code should not be so large, just a few lines to link with the automation library, which will be the level which will really do the most of the automation work. In 2 just to lines are enough to call the automation library and tell it what to do.

  Listing 2: Example of step definition and support code

  ```
  @When("^(?:she|he|the user) logs in the system as ([^\"]*)
      with password ([^\"]*)$")
  public void logIn(String login, String passwd) {
     new ToolInstanceCommonPage(driver).clickLogIn();
     new ToolInstanceCommonPage(driver).fillLogin(login,
        passwd);
  }
  ```

- *Automation Library:* Here is where *Selenium* comes into play, hence as explained above is the chosen technology to automate the browsers.

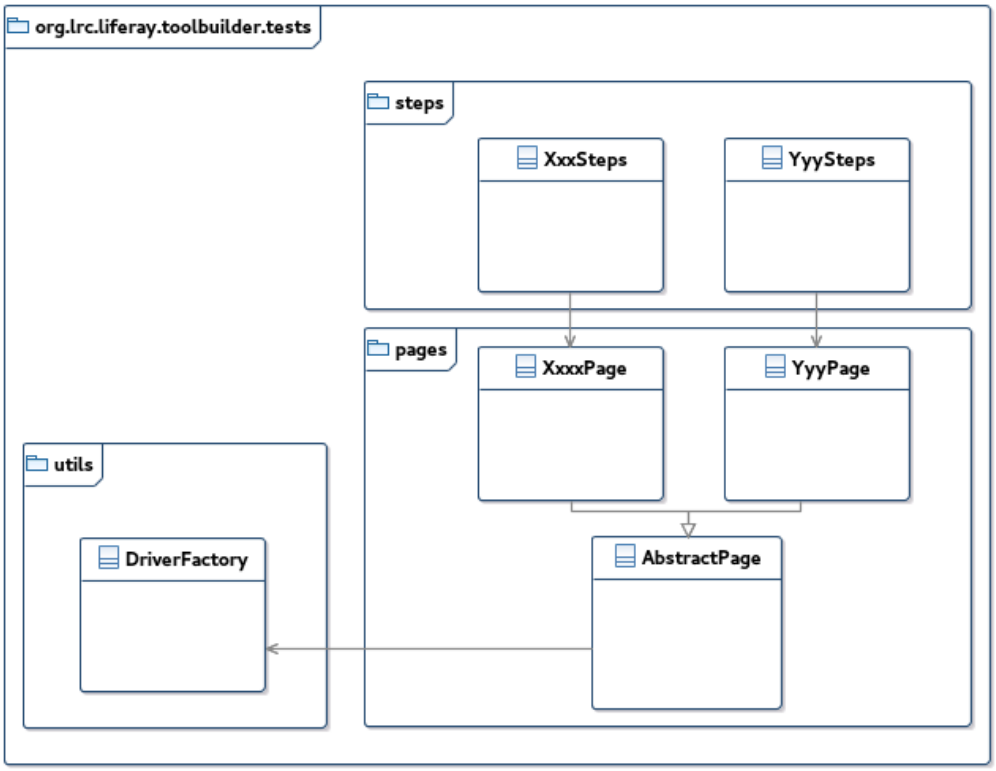


Figure 5: Testing framework design

4.3 Quadrant 3 tests

This tests are oriented to critique the product when it has been already developed and find any unwanted behaviors usually not stipulated in the requirements (if BDD has been well carried out). That is why this tests are not automatized and require the strong involvement of the customer, that is what is called User Acceptance Tests which should not be confused with the ATDD performed in Quadrant 2. Also here are included usability tests with different users and real-world cases.

In the project, some tests in this quadrant has been taken. But the functionalities are not quite numerous and, since it is a early development phase to create the structure, the usability is not a main objective. So the tests on this quadrant have been just a few manual tests with some members of the Livelihoods Resources Centre which have critique the web app and validate it.

4.4 Quadrant 4 tests

Finally, the Quadrant 4 comprises all those Technology-Faced tests aimed to critique the quality of the final product. This includes from security to performance, reliability or compatibility. Even if all those points have been considered when developing the software, there have not been specific tests as it is a huge domain of tests and is outside the scope of this project.

5 Analysis, technology requirements and design

At this point it is worth remembering a statement from [12]: *"The term test-driven development misleads practitioners who don't understand that it's more about design than testing. Code developed test-first is naturally designed for testability."* So it is important to move away from the idea that BDD does not entail design. In fact, as it will be shown in 6, almost all the sprints include a *Testing design* subsection and there are several refactoring sprints. The earliest ones regard the quote from [12] and lead the software development to get a product structured by features, the latter ones are sprints aimed to clean and restructure the code so it is more comprehensible, its complexity is reduced and therefore maintainability and extensibility are improved.

This section discusses about the design process and its final result, but before that it is important to know which and why have been the chosen technologies, since it has irretrievably affected and guided the design.

5.1 Technologies

As in previous sections the development methodology have been explained as well as the tools and technologies related to the methodology, this subsection only comprises those technologies and tools directly related to the development.

5.1.1 Liferay Portal

The choice of the web framework has been taken directly by the customer. As explained in 2, the final product should be a web application embedded in the Livelihoods Resource Centre web, so as that web has been developed with Liferay Portal Community Edition because of an organizational decision, there is not discussion about that: development in Liferay is a customer requirement.

Liferay Portal is an open-source enterprise portal distributed under the GNU LGPL[14]. As it is Java-based, and even it supports plug-in extends into multiple programming languages, the most natural, efficient and easy to install and maintain is a portlet in Java, so that is why the web app has been developed as a Liferay portlet.

Liferay Portal also provides some tools to facilitate the development. One of the most important ones is what is called *Service Builder*[15]. It is a code generation tool which provides an easy way to create object model generating a *service layer [...] that provides a clean separation between the object model and code for the underlying database.* Although it has some constraints as it will be shown in 6, it is an interesting tool to provide developers a way to create secure and uniform code for the model component and it is recommended by the Liferay team, so it has been used in the project.

[14] https://www.gnu.org/copyleft/lesser.html
[15] *https://www.gnu.org/copyleft/lesser.html*

5.1.2 JavaServer Faces (JSF) & Components Design

The two main ways to create a portlet for Liferay are: a Liferay MVC portlet (using the Liferay MVC framework) and a Liferay JSF portlet (based on the JavaServer Faces specification and implementation[16]). The decision of using JSF for this project has been mainly based in the fact that JSF is a standard also outside Liferay, it is more modern than a classic JSP solution and it is a web application oriented specification. Liferay supports JSF 2.2 specification but for Liferay Portal 6.2 (the version used in the LRC web) default JSF specification is JSF 2.1, so it has been the version used in the development to avoid compatibility problems.

JavaServer Faces is a framework to write MVC web application and its version 2.1 was defined in the Java Specification Request 314[17]. The main components of JSF, which are shown in figure 6, are:

- The ***Faces Servlet***, which controls the application flow and is configured through an *XML configuration file*. It plays the *Controller* role in the MVC architecture.
- The ***JSPs created with JSF UI***, which defines the application pages and uses *JSF UI* components as *tags*, *validators* and *events*. It is the *View* in the MVC architecture.
- The **Managed beans**, which stands for the Model in the MVC architecture, and connects with other business logic components and the persistence layer.

On the other hand, it is worth pointing out that JSF pages have a particular life cycle which should be considered when using this technology. As in a JSP page, when the client makes an HTTP request for the page, the server sends the response as the page in HTML. The figure 7, from [3], depicts the six phases of the JSF life cycle:

- **Restore view:** This is the first phase and is when the view of the page is built with all its tag components, event handlers and validators.
- **Apply request values:** During this phase each component built in last phase extracts its new value.
- **Process validations:** As its name suggests, this is the phase to process the validators.
- **Update model values:** After processing validators, the server-side model is updated, beans are in charge of that.
- **Invoke application:** At this point, any existing application level events such as submitting a form or linking to another page are handled.

[16] https://javaserverfaces.java.net/
[17] https://www.jcp.org/en/jsr/detail?id=314

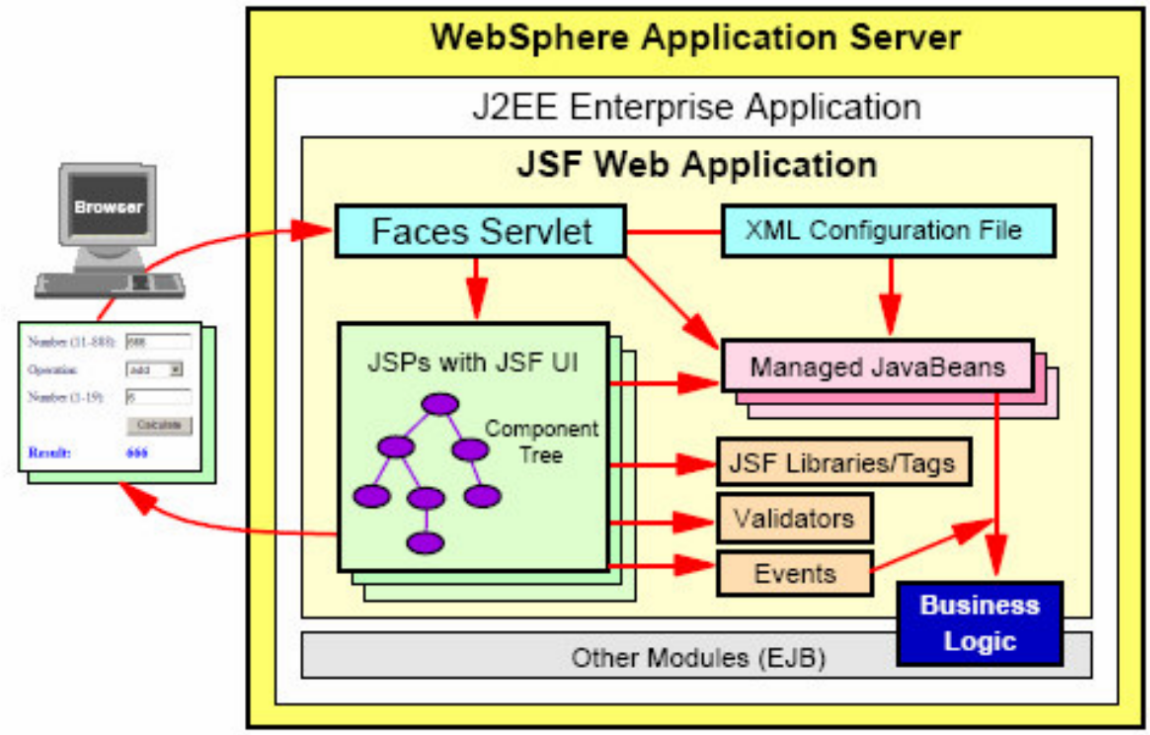


Figure 6: JSF components

- **Render response:** During this phase the page is rendered through the JSP container.

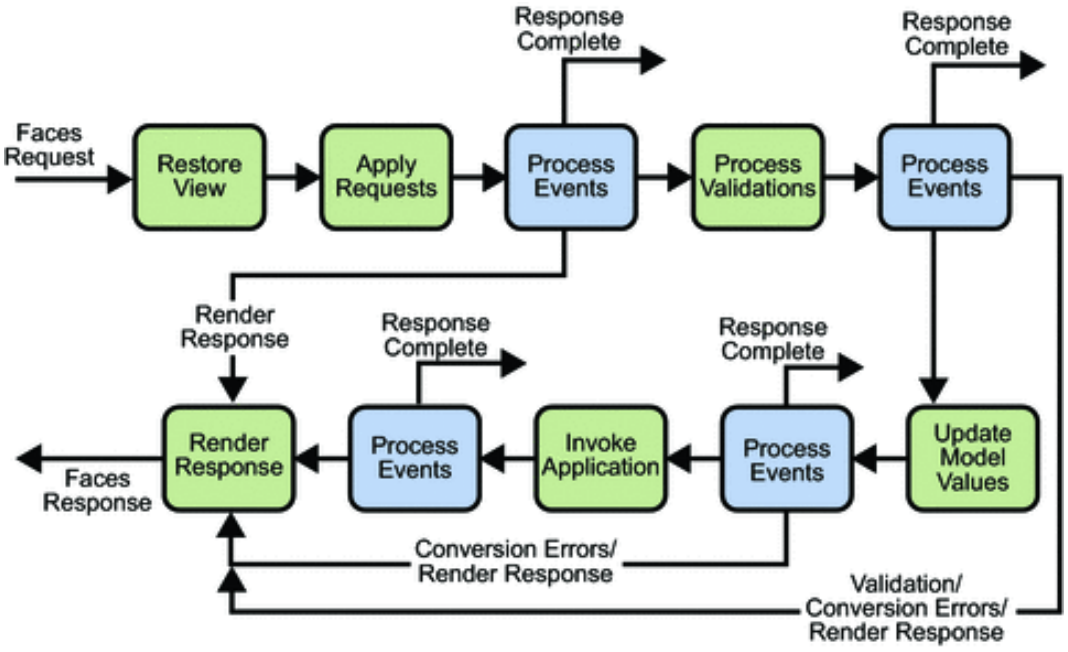


Figure 7: JSF life cycle

5.1.3 Primefaces

Primefaces[18] is an open source JSF component suite with built-in Ajax support and a rich set of components which make easier for developers to create interfaces. It has been the JSF library used in this project.

5.2 Object Model

Due to the development methodology, the object model has evolved during the project. Despite of the basic structure is given by the JSF framework as explained in last section, the beans and auxiliary objects which define the model have been changing and being adapted to fit the project evolution and requirements. In section §6 the different phases of the classes diagram will be shown.

5.3 Data Model

As the object model, data model has been developed through the development process refactoring it whenever it is necessary as it will be shown in next section. The final ER diagram for the database, which is indeed the data model, is shown in figure 8

6 Development

Hereafter and for the purpose of explaining the process followed in the development, an overview of the main issues related to each sprint has been written.

Preliminary work

Before beginning the development, a mockup design process was carried out with the customers in order to have a first approach of how the final web app should look. Even if the objective of this project is not to develop the final app but its first phase, this mockups have been kept in mind during the development process in order to ensure the final objective is achieved in the future. These mockups can be found in Appendix B on page 56 both in online and printed format. It is worth to notice that they have entailed a hard work of analysis of the Ram Assessment for Markets tool in order to transfer it to a web interface which make it more understable and available to non skilled users.

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

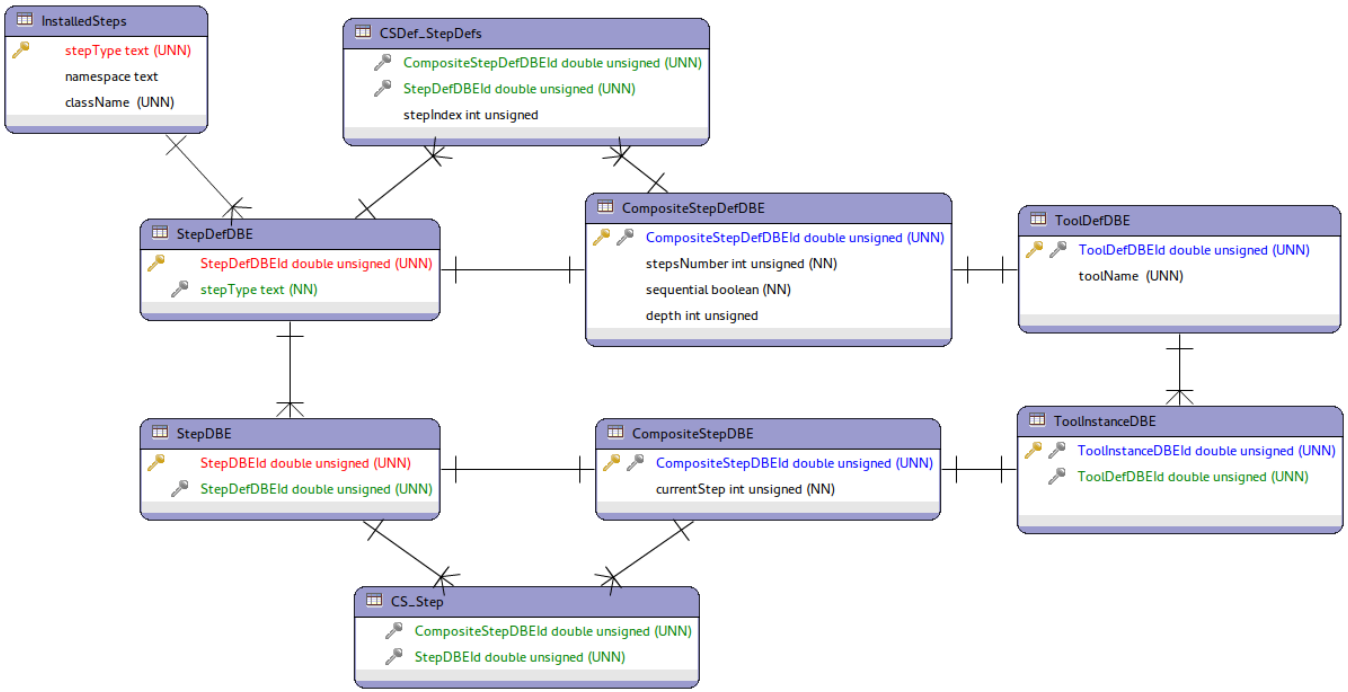


Figure 8: Final ER diagram

6.1 First Sprint

A basic web tool (portlet) to navigate through the RAM Steps and capable of save the current state of each assessment was created in this step following a classic development cycle (DESIGN -> DEVELOPMENT -> TESTING). The portlet was called `ram-webtool`.

6.1.1 Design

A first design was drafted to make implementation easier so the JSF architecture was kept in mind when coding. As discussed in section 5.1.2 we will focus in Model and View parts of a MVC architecture since the Controller is provided by the framework as the `Faces Servlet`.

The *Model* has two different parts due to the architecture provided by Liferay and JSF. On the one hand the model object provided by the *Liferay Service Builder (LSB)* which represents the persistence layer. As discussed in section 5.1.1: Liferay Portal, *Liferay* provides a service to create the persistence layer, so it was used to create the object `Assessment` which stores the current step of each created assessment, considering that this current steps does not depend on the session but is a value that must be saved in the system.

On the one hand, two beans were necessary in this sprint as part of the model:

- A *session scoped* bean to keep all the session information, which is needed to know where the user is, what is he doing, what are the next allowed actions, etc.

- A *request scoped* bean which will be responsible of bringing the needed information and which actually works as a bridge between the persistent object and the view.

Regarding to the view, namely the *JSF UI components* according to section 5.1.2, it consists of two main JSF pages with several components:

`mainView.xhtml` Which provides the interface to create new assessments and includes the `assessmentsList.xhtml` component, in charge of showing the existing assessments list.

`assessmentView.xhtml` Which depending on what the user is doing will show the `editAssessmentForm.xhtml` component, if user is editing the assessment, or a `ramStepX.xhtml` component, if user is consulting the assessment.

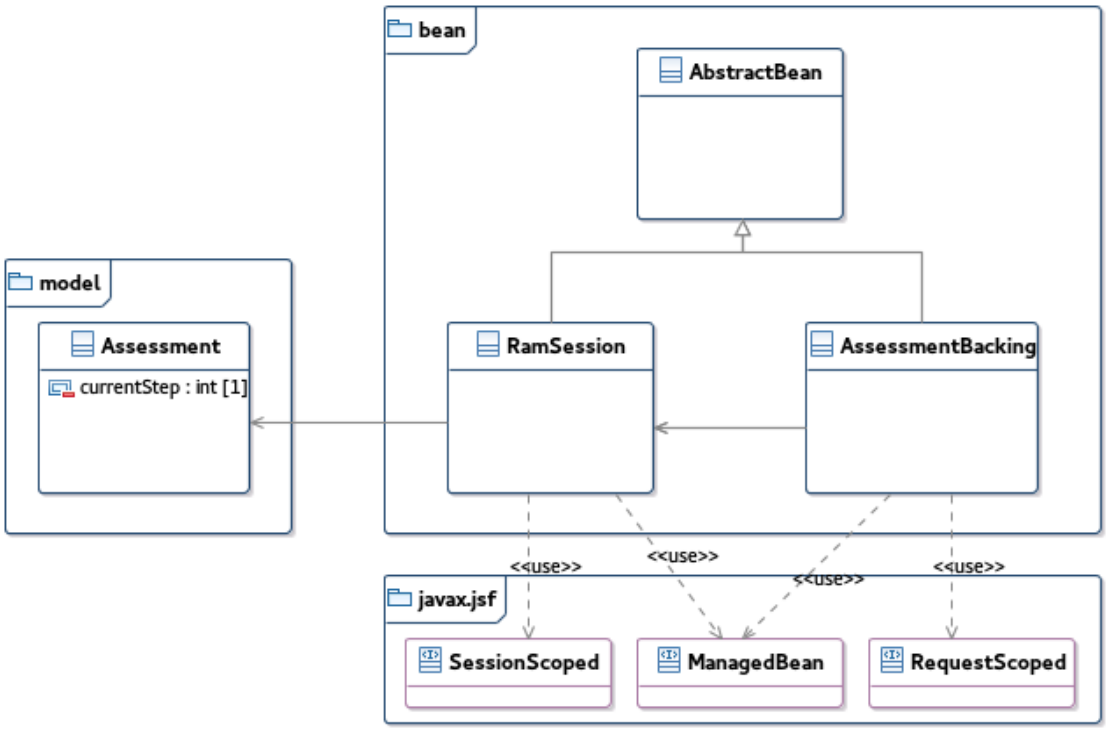


Figure 9: Design for the first sprint

6.1.2 Implementation

As explained above, for the persistence layer the Liferay Service Builder was used, which helps developers to code simple persistence objects which will work with the system relational database. After implementing the persistence layer, the `mainView.xhtml` was developed. This allowed the developer to draft the methods which would be needed from the beans. Lastly, the two beans were coded implementing the needed methods and inheriting their common characteristics from a common parent `AbstractBean`.

6.1.3 Testing

Testing in this sprint was merely human guided since the testing framework had not yet been developed.

6.2 Second Sprint

In order to implement in the future the flow chart tools within the RAM, a flow chart system should be developed. The flowchart library search didn't give results so a generic library `util` was developed.

6.2.1 Design

For the purpose explained above, four main classes were necessary, but in order to extend the library for future uses, a fifth class was created for non-strict flows:

`BasicNode<T,C>` This class is the core of the library and represents a node of the flowchart which is composed by a content of type `T` and an undefined number of edges of type `C`, which in turn inherits from `Comparable` in order to allow a flow choose among all the edges the correct one. This class also implements not only methods to create and destroy nodes, but also to find the next node in a flowchart for a given condition.

`BasicFlowChart<T,C>` Composed by BasicNodes implements the methods to create, edit and destroy a flowchart as well as the methods to navigate through the flowchart.

`Edge<T,C>` This is just a very simple auxiliary class only used to make easier the construction of `BasicFlowChart`.

`Flow<T,C>` This class represents a particular flow and the flow status, that is to say the current node of the flow.

`PermissiveFlow<T,C>` As explained before, this class has been created just to implement non-restrictive flows. In other words, flows allowing inputs not matching any of the conditions of the edges of the current node, in this case the flow will remain in the current node.

6.2.2 Implementation

The classes were implemented in ascending order of complexity, this means beginning with `BasicNode`, followed by `BasicFlowChart` and `Edge`, and finally `Flow` and `PermissiveFlow`.

6.2.3 Testing

Unit tests were developed using JUnit, as showed in figure 11, there are three suites of tests for this sprint:

BasicNodeTests Related to the `BasicNode<T,C>` class, it comprises 10 tests.

BasicFlowChartTests Related to the `BasicFlowChart<T,C>` class, which includes `Edge<T,C>`, it comprises 25 tests.

FlowTests Related to `Flow<T,C>` and `PermissiveFlow<T,C>` classes, it comprises 5 tests.

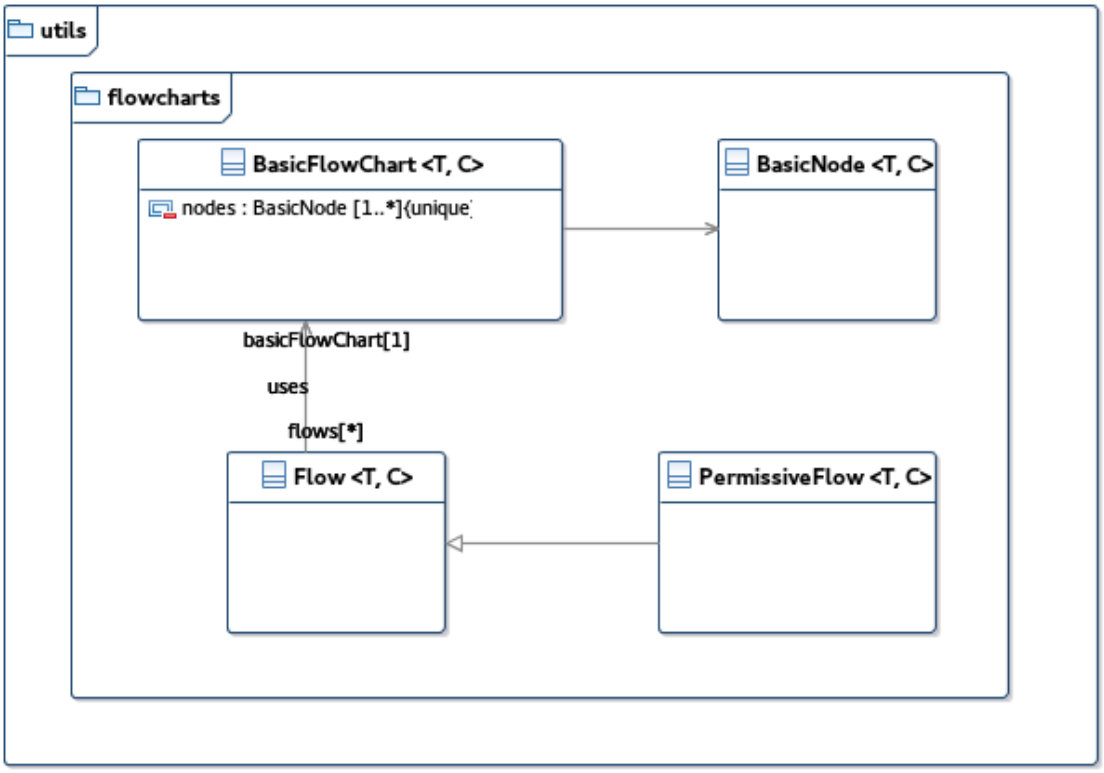


Figure 10: Design for the second sprint, the flowchart `util` library

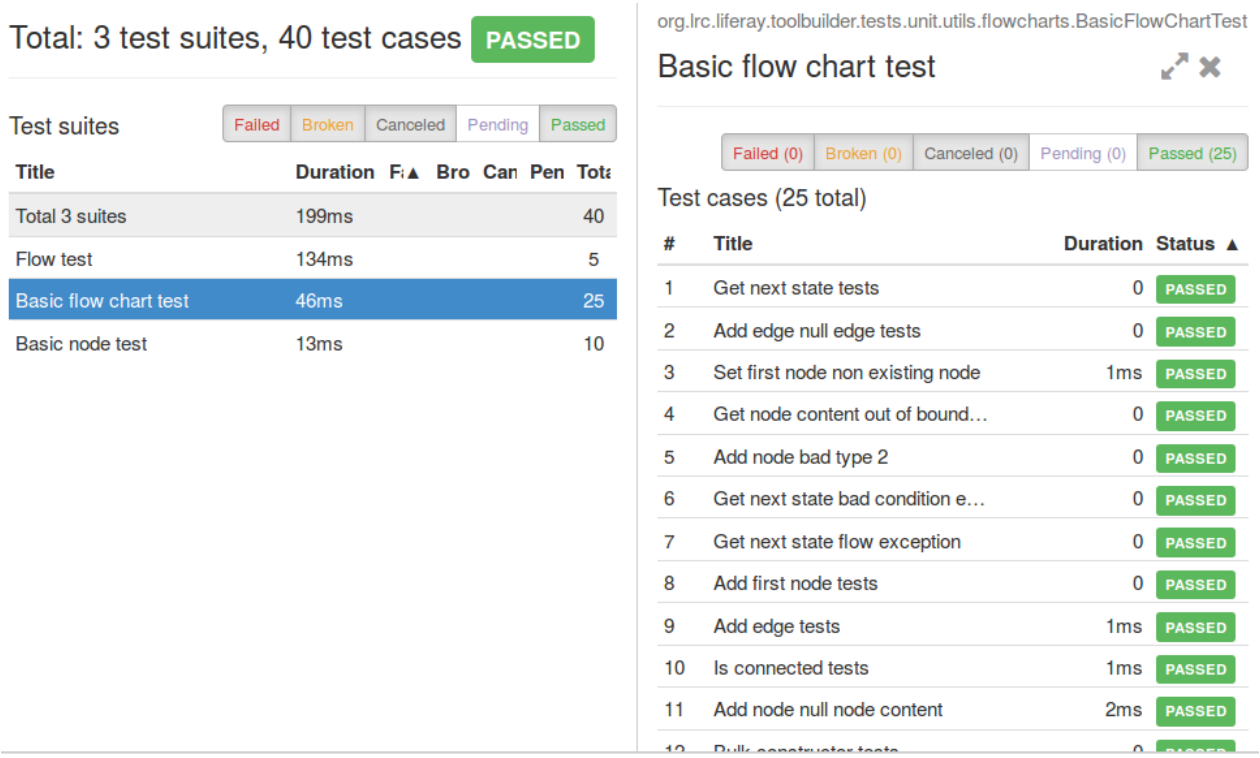


Figure 11: Allure screenshot with tests passed for sprint 2

Being unit tests, a little time is required to pass each test as it can be checked in figure 12.

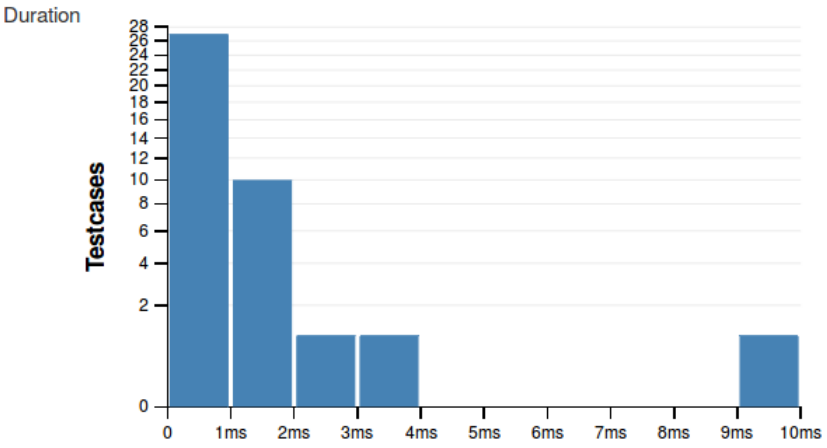


Figure 12: Time graphic for tests of sprint 2

6.3 Third Sprint

After some interviews with the client it was concluded that the development is quite limited and poorly flexible since it will just fit the current RAM tool requisites and it would be pretty complex to extend it. Finally it was decided to change the requisites and develop a web tool flexible enough to implement different kind of methodologies or tools as EMMA or HES[19]. That is how the project has become a tool builder, and so is called Toolbuilder, and the new portlet which implements the tools is called `tool-implementation`.

6.3.1 Design

Although the *beans* and the *JSF views* remained the same, some generalization were done in the model to conform to the new requirements. The former `Assessment` class is going to be replaced by a more complex structure:

- Firstly, as there must be several kind of steps depending on the specifications of the tool to be created, two type of steps (classes) were created in this sprint which inherit from the class `Step`:

  `VoidStep` Just a type of step for developing purposes without content and/or configuration.

  `CompositeStep` This is a step which contains other steps so an structure step/-substep can be implemented following a *Composite Pattern*[13]. As it should store information about the substeps it contains, it is persistent, therefore it was created with *Liferay Service Builder*.

[19]Households Economic Security

- Moreover, an object called `StepFactory` is in charge of the creation of the different steps following a *Factory Pattern*[13].

- Lastly, a new object called `ToolInstance` fundamentally performs the function of the former `Assessment` object storing all the information related to the each assessment, hereafter called *tool instance*.

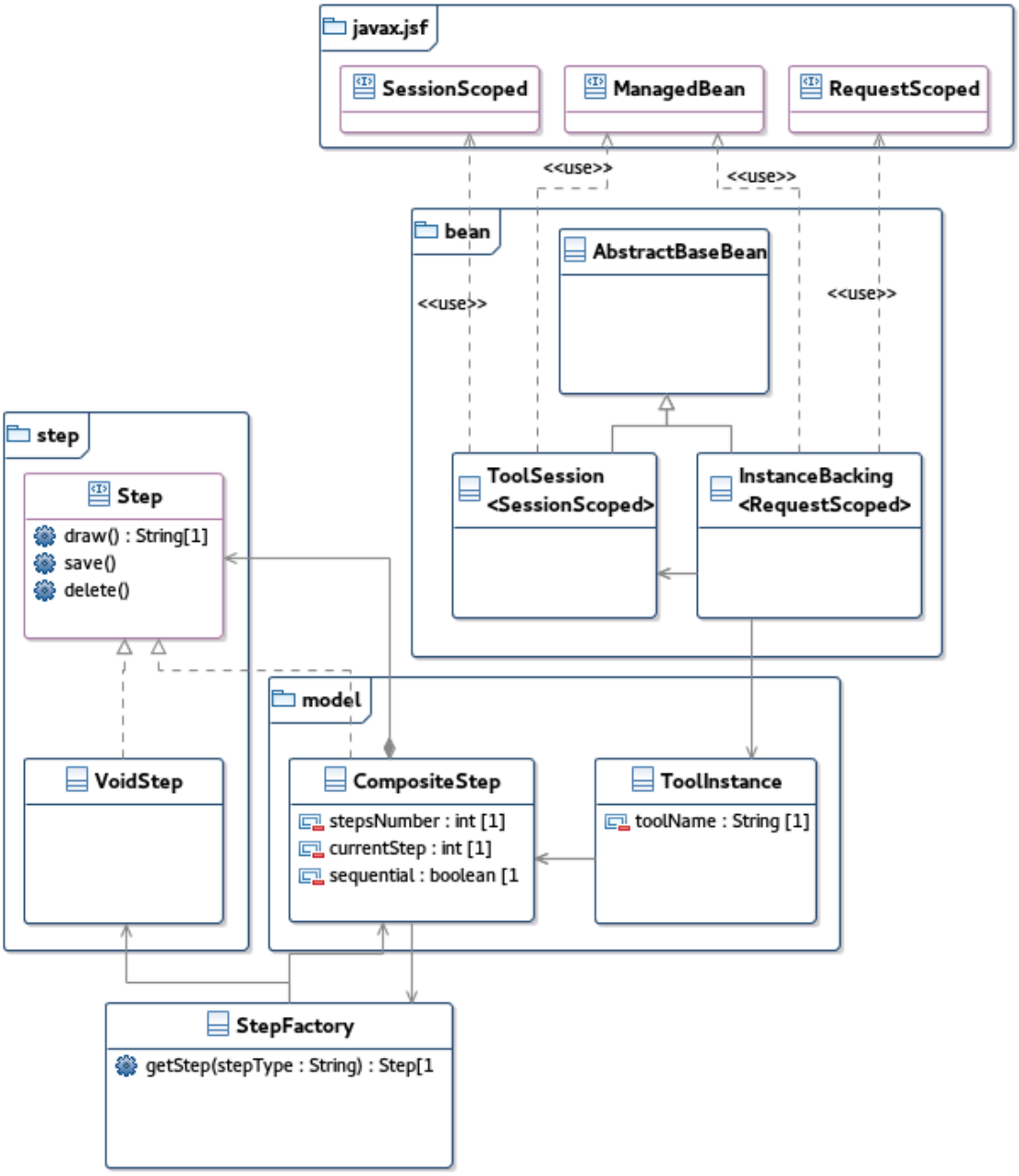


Figure 13: Design for the third sprint

6.3.2 Implementation

After replacing `Assessment` class by `ToolInstance`, the first step was to create the `Step` class, which would be the parent of all kind of steps. Later a new *Service Builder* entity was created for the new `CompositeStep` class, followed by the simple `VoidStep` class which mocks any kind of step. Finally, the `StepFactory` class was implemented to create steps.

6.3.3 Testing

In this sprint the testing framework was implemented and the Behavior Driven Development (BDD) was set up for next Sprints. Tests scenarios were developed for the three existing features:

1. Tool Instance Creation
2. Navigate through steps
3. Tool Instance Deletion

For each feature, the different scenarios or user stories (in compliance with the BDD methodology) were created. The development of the tests also required some debugging, as it is shown in figure 14 the first time tests were executed there were some defects. After debugging it was checked that defects were caused by bugs in the tests, particularly by the way *Selenium WebElements* were selected because the *XPath* did not match with the web elements outputted by the portlet.

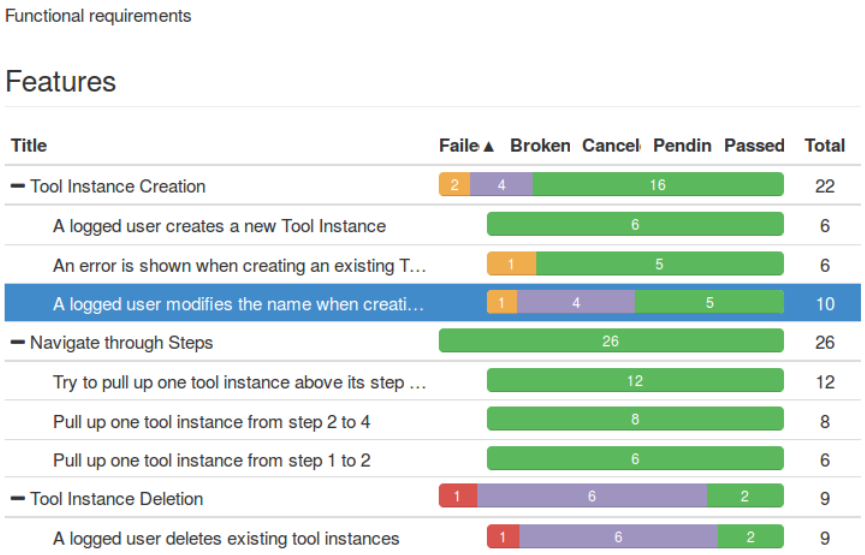


Figure 14: Sprint 3 tests failed at the beginning and required debugging

Finally in figure 15 is shown the tests result after debugging.

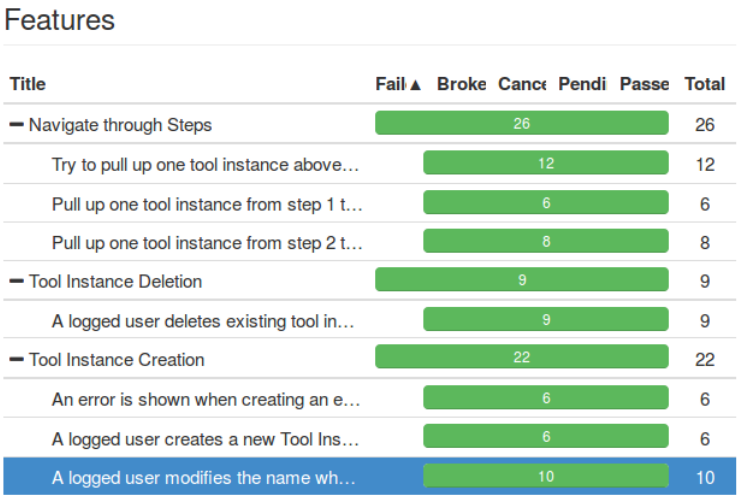


Figure 15: Sprint 3 tests passed successfully after debugging

6.4 Fourth Sprint

As tools should be able to be customized and many occurrences of each tool may be created, it becomes necessary to make code refactoring in order to divide the two parts: *Tool Definitions* and *Tool Instances*. So after refactoring we have separately the objects related to the tool structure - hereafter *Tool Definition* - and the objects related to the tool occurrences - hereafter *Tool Instances*.

6.4.1 Redesign / Refactoring

Tool definitions are separated from tool instances creating two different classes (`ToolDef` and `ToolInstance`), likewise steps definitions should be separated from step implementations, for that purpose abstract classes `StepDef` and `Step` were created to steps inheriting them, the first one representing the step definition and the second one representing the step within the *tool instance*, that is to say the step occurrence.

As shown in figure 16, a symmetric design has been implemented which will be replicated for each step:

- In blue the definition part, with `StepDef`, `CompositeStepDef` and `ToolDef` classes. All them related to the *tool definitions* which will define the structure and configuration of each tools and each step.
- In red the instances part, with `Step`, `CompositeStep` and `ToolInstance` classes. These classes represent to the status of each tool instance. Namely the current step for each tool instance, the status of each step and all the related information.

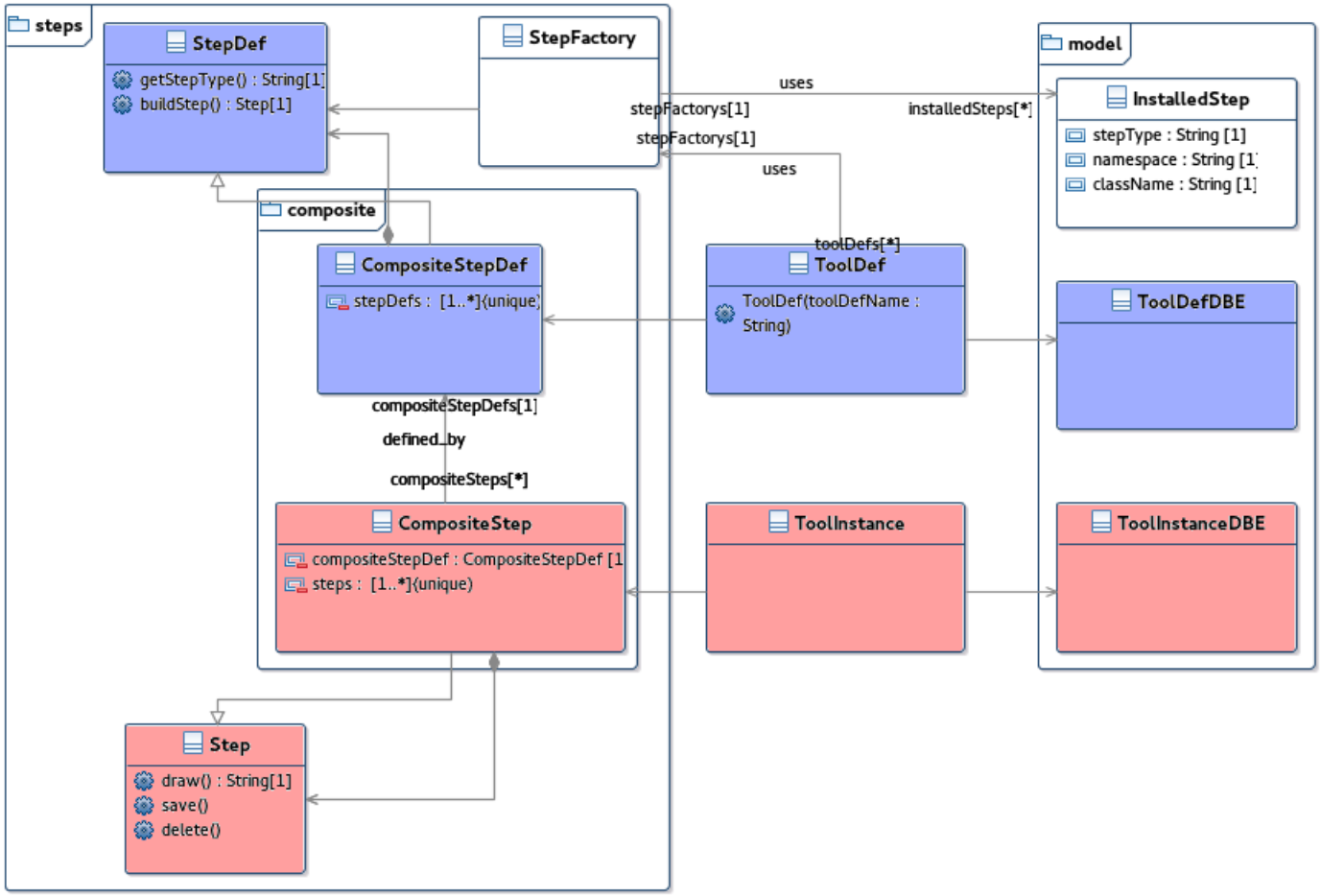


Figure 16: Design for the fourth sprint

As we can see in figure 16, *Factory* and *Composite Patterns* are still being used after refactoring.

In the same way, database needed to be redesigned in order to match with the new classes design. As shown in figure 17 there is also a symmetry between the definition part, on top, and the instances part. Each *Tool* (both definition and instance) contains a *Composite Step*, which as a relationship 1 to 1 with the *Step* table (as class `CompositeStep` inherits from `Step`) but also a relationship N to N because one *composite step* contains several *steps* and one *step* could be in several *composite step*s. Although there is data redundancy in the N to N relationships - which in fact can be simplified removing the relationship between tables *CompositeStepDBE* and *StepDBE* - at this sprint it was felt useful to make easier the development of the associated classes.

Finally, a new *managed bean application scoped* was created called `FactoryBean`. This bean manages and stores the different type of steps and existing tool definitions and it also stores the tool definition which is in use.

6.4.2 Test design

Tests remained the same since no new user features were implemented. Anyhow, the test were passed to guarantee refactoring has been successful and accomplishes all the required features.

6.4.3 Implementation

The only aspect to note during implementation was the difficulty of working with *Liferay Service Builder (LSB)* and its way of supporting N to N relationships. After exploring how to do it the persistent objects were created, the *LSB* allows to create N to N relationships but, as it would be realized later in section 6.10, it is not the best option due to the poor customization allowed.

6.5 Fifth Sprint

Due to the breakup between *tool definition* and *tool instance* it is needful to define permissions to access both areas. Thus, the definition should be used by tool administrators, while the instances should be available to logged users, but non logged users will be able to see tools instances not having permissions to use them. Therefore, the permission scheme should be defined to provide limited access based on user role following the scheme in 1. In addition to the `tool-implementation` portlet, who is in charge of the tool instances, a new portlet called `tool-definition` has been added to manage the tool definitions, it will be created in the next sprint.

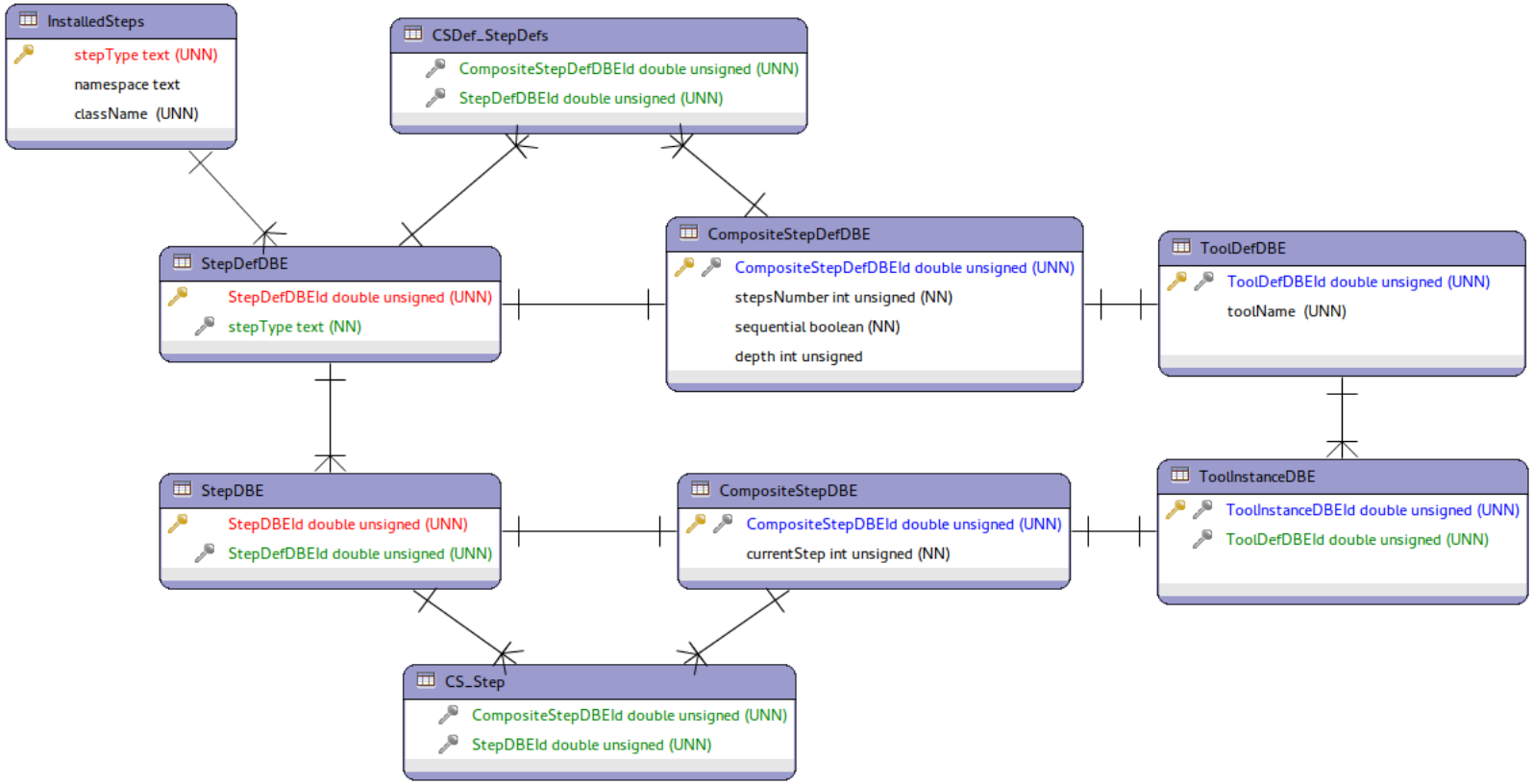


Figure 17: Database design for the fourth sprint

Portlet Name	Permission	Anonymous user	Logged user	Administrator
`tool-implementation`	Create Tool Instance		✔	✔
	Use/Edit Tool Instance		✔	✔
	Delete Tool Instance		✔	✔
	View Tool Instance	✔	✔	✔
`tool-definition`	Choose Tool Definition			✔
	Create Tool Definition			✔
	Delete Tool Definition			✔

Table 1: Permissions scheme for Sprint 5

6.5.1 Test design

The functional tests were modified to include permissions restrictions, so a step for log in the user was added to each existing scenario - as shown in figure 18 which in the step 2 logs the users. Also 3 more scenarios were added to test the restrictions for non logged users, one for each feature. No tests have been added for the `tool-definition` portlet as it will be implemented in the next step.

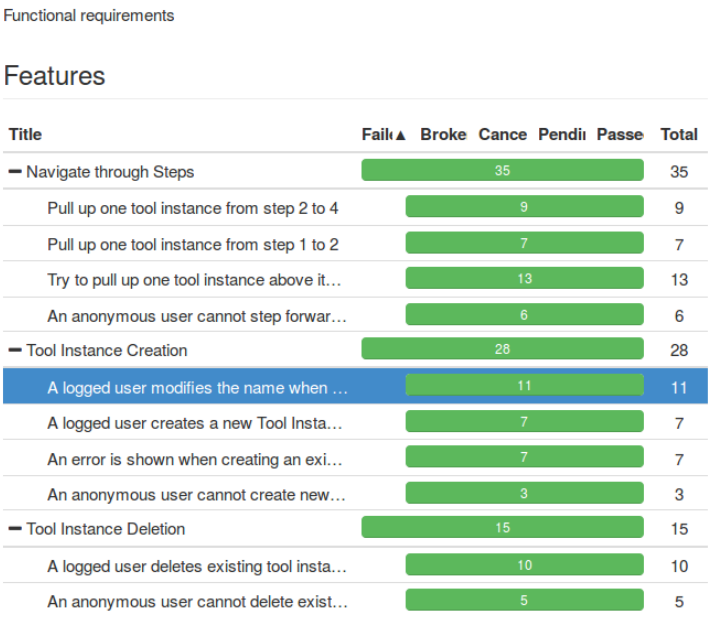


Figure 18: Sprint 5 tests passed successfully

6.5.2 Implementation

To code the permission restrictions, the Liferay permissions scheme has been used. It allows to define permissions for the portlets and the model based on the user's role through a xml file. As shown in 3, three permissions have been configured: add, delete and edit; those three permissions are allowed to any site member but restricted

Listing 3: Permissions configuration file

```
<resource-action-mapping>
   <model-resource>
      <model-name>org.lrc.liferay.toolbuilder.model</model-name>
      <portlet-ref>
         <portlet-name>tool-implementation</portlet-name>
      </portlet-ref>
      <permissions>
         <supports>
            <action-key>ADD_TOOL_INSTANCE</action-key>
            <action-key>DEL_TOOL_INSTANCE</action-key>
            <action-key>EDIT_TOOL_INSTANCE</action-key>
         </supports>
         <site-member-defaults>
            <action-key>ADD_TOOL_INSTANCE</action-key>
            <action-key>DEL_TOOL_INSTANCE</action-key>
            <action-key>EDIT_TOOL_INSTANCE</action-key>
         </site-member-defaults>
         <guest-defaults />
         <guest-unsupported>
            <action-key>ADD_TOOL_INSTANCE</action-key>
            <action-key>DEL_TOOL_INSTANCE</action-key>
            <action-key>EDIT_TOOL_INSTANCE</action-key>
         </guest-unsupported>
      </permissions>
   </model-resource>
</resource-action-mapping>
```

to guests, namely anonymous users. As can be noted only permissions for the `tool-implementation` portlet were defined in this sprint, since `tool-definition` portlet implementation will be addressed in the next sprint.

6.6 Sixth Sprint

Once the back-end for tool definition was developed it was necessary to create the interface so admin users can choose between the existing tool definitions, therefore in this sprint the portlet `tool-definition` was created.

6.6.1 Test design

Following BDD methodology, behavior of the system for this new feature was defined by developing some four scenarios. In figure 19 these tests, passed after the implementation, are shown.

- The admin user chooses an existing Tool Definition different to the chosen one.

- The admin user chooses another existing Tool Definition, which in fact means choosing back the last Tool Definition.
- Access to tool-definition portlet forbidden for a logged user who is not admin.
- Access to tool-definition portlet forbidden for an anonymous user.

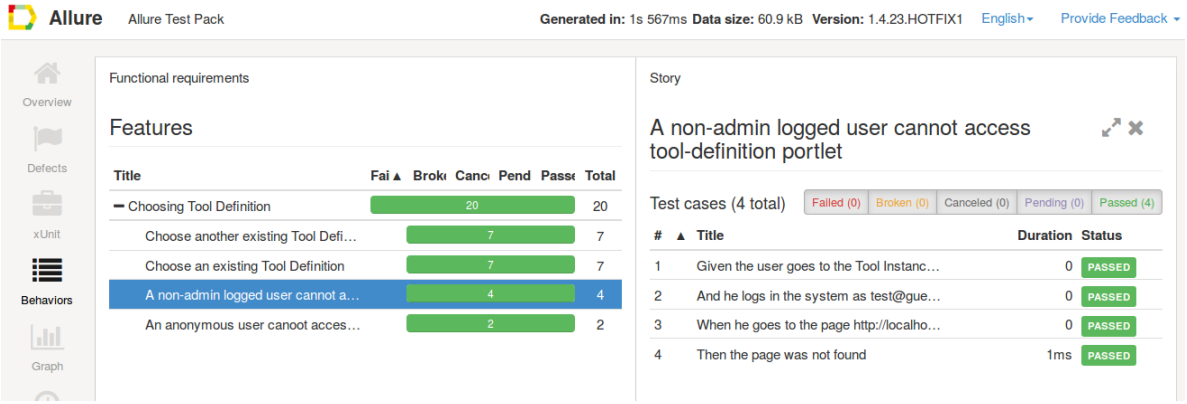


Figure 19: Sprint 6 tests passed successfully

6.6.2 Implementation

The new portlet called `tool-definition` was created only accessible for admin users and available through the *Site Administration -> Content* section, so the `control-panel-entry-category` parameter was defined as `site_administration.content`. Thus, the permission scheme from 3 was implemented because that section is only accessible for admin users.

As the current tool definition is stored in the *application scoped bean* `FactoryBean`, only the view `adminView.xhtml` needed to be developed in this sprint. The list of tool definitions was implemented through a `<p:datatable>` from *Primefaces*. Some troubles were also founded in order to refresh the page when a tool definition was chosen but, finally it was decided only to render again the form the button was embedded in, though an *Ajax* call.

6.7 Seventh Sprint

A use case to create a new tool definition is defined as the first step to configure that new tool definition which will be able to be selected in the `adminView.xhtml` and used by logged users.

6.7.1 Test Design

As shown in figure 20, the feature *Create Tool Definition* has been defined with three scenarios, two of them show an error when trying to create an existing tool definition.

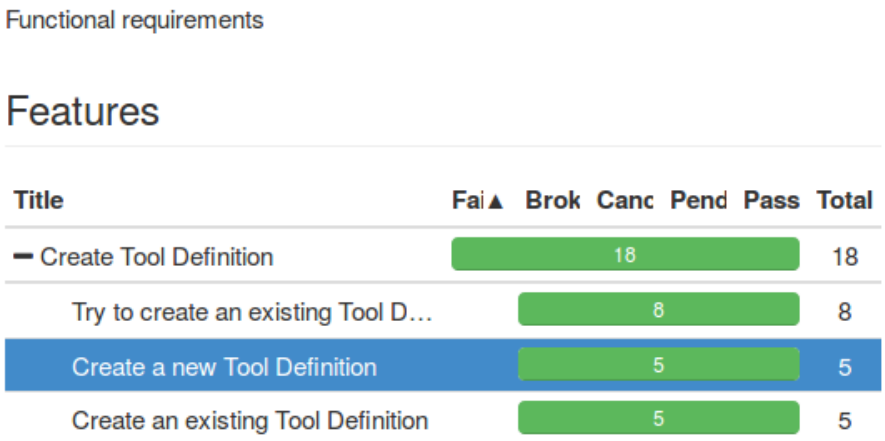


Figure 20: Test scenarios for sprint 7

6.7.2 Implementation

In order to have different views for tool definition choosing and tool definition edition, a new `toolDefConfig.xhtml` file was created which is called to configure the tool definitions. In this case, the name of the tool to work with is passed as part of the URL (using `GET` method instead of `POST`) so the page can be addressed straight away by any admin user foreseeing a future option of create a tool definition from other different page.

In addition, a way to show different step definitions should be defined, so a tabs view was used due to the familiarity of users with this kind of interface and the easiness of development provided by the Primefaces `tabView` resource.

Finally, it was necessary to develop two new managed beans, colored in blue in figure 21:

`ToolDefListBacking` It is a *request scoped* bean and it is called when creating a new tool instance in order to verify the name - check if it is not void or if it already exists, so it prevents database exceptions - and to create the URL to create the new tool definition.

`EditToolDefBacking` It is a *view scoped* bean and it is related to the view `toolDefConfig.xhtml`, so it stores the necessary data and provides methods to save it.

6.8 Eighth Sprint

An interface to add steps to the tool definition should be created so after creating a new tool definition, admin users were able to add different steps and give content to the tool.

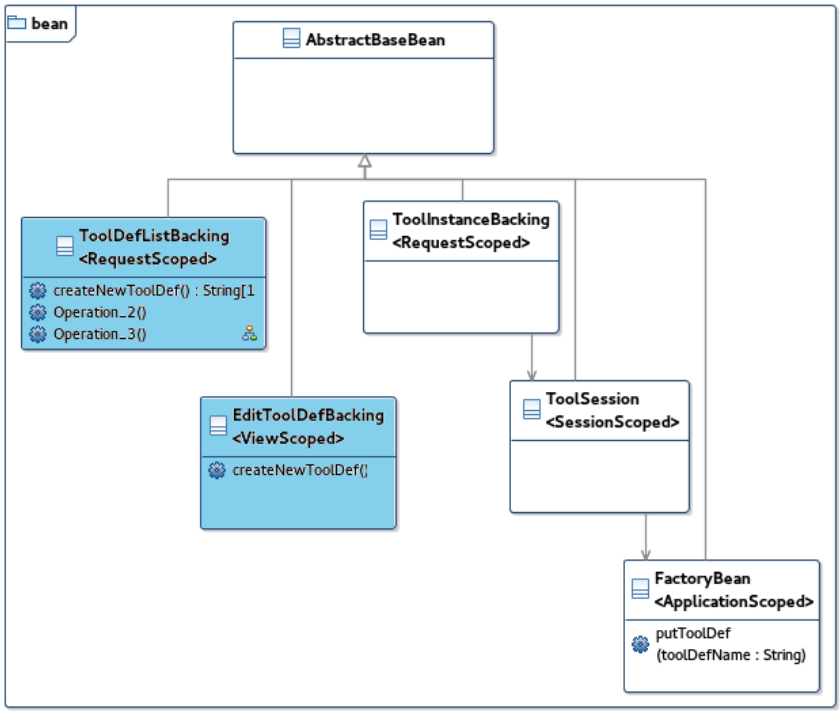


Figure 21: Existing managed beans in sprint 7

6.8.1 Test Design

The fact of adding steps to a new tool definition was considered as part of the behavior *Create Tool Definition*, so some new scenarios were appended to the existing feature. There were some issues when coding the steps since, as it is explained in the next implementation section, the used framework creates a new `<iframe>` for a new dialog; this implied that the elements were not founded by the *Selenium WebDriver*. Finally it was fixed when finding out that *Selenium WebDriver* defines the `iframe` which is working in, so if it changes it should be switched - calling the method `driver.switchTo.frame()` - in order to allow the driver to find those elements.

6.8.2 Implementation

The new view should be embedded in the `toolDefConfig.xhtml` view, but it is worth noting that it should be independent since it was going to be called not only to add steps to the main tool definition, but also to add steps to any of the nested composite steps.

In order to have an independent view which could be embedded, the *Primefaces Dialog Framework* was used so a new view called `selectStepDefDialog.xhtml` was created and it is shown as a dialog (figure 23) when the user is going to create a new step definition. Thus, the choice of the new step type is made by the user in this new dialog, which returns it (the step type) to `toolDefConfig.xhtml` via `POST` method. The *Primefaces Dialog Framework* creates a new `<iframe>` and embeds

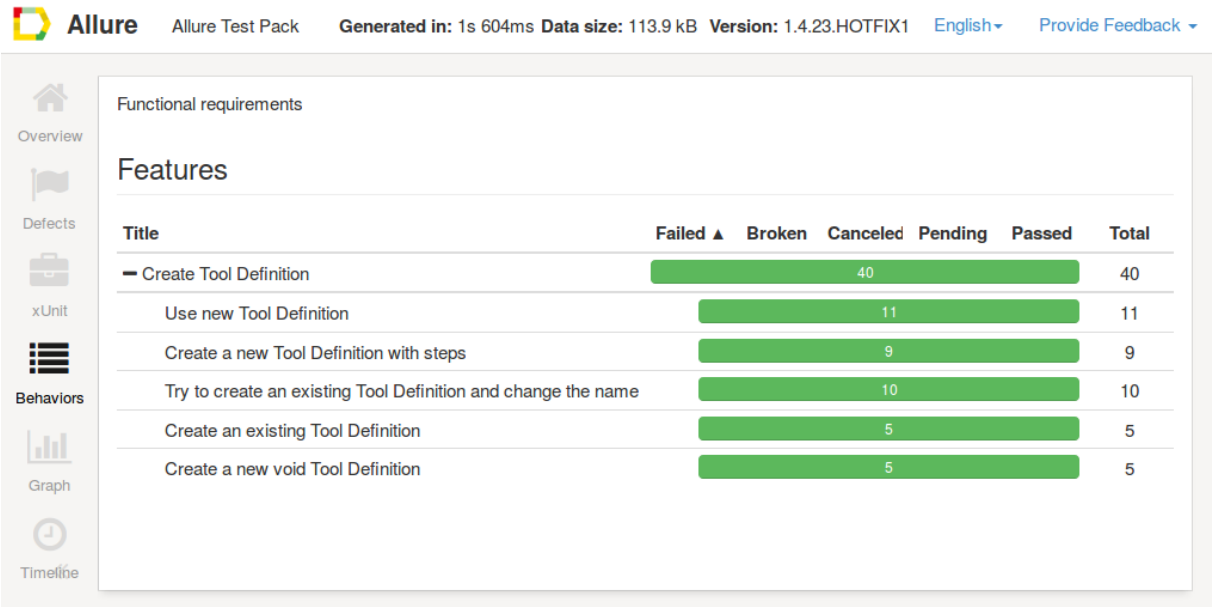


Figure 22: Test scenarios for sprint 8 belong to the same feature than sprint 7

the new view, so it is completely independent and it allows to call the view from other different pages.

On the other side, the view `toolDefConfig.xhtml` was modified to show the created steps, the chosen method to do that was using the *Primefaces* `<p:tabview>` tag as the tab interface uses to be familiar to users.

6.9 Ninth Sprint

After creating tool steps, next needed feature was to delete tool step definitions, which is addressed in this sprint.

6.9.1 Test Design

The step definition deletion operation is made from the `toolDefConfig.xhtml` and should consider some different cases: from one to several steps deletion and from an already existing or a new tool definition. Therefore, four cases were designed which, as shown in 24, were successfully passed. There were some code refining when testing, since the processing of deletion should depend on the load onto the database of newly created step definitions.

6.9.2 Implementation

The implementation of this sprint has been done through the attribute `closable` of the tab elements which executes and *Ajax* call when the `tabClose` event triggers, so the *Ajax* call was defined to delete the register of the persistence layer. The only

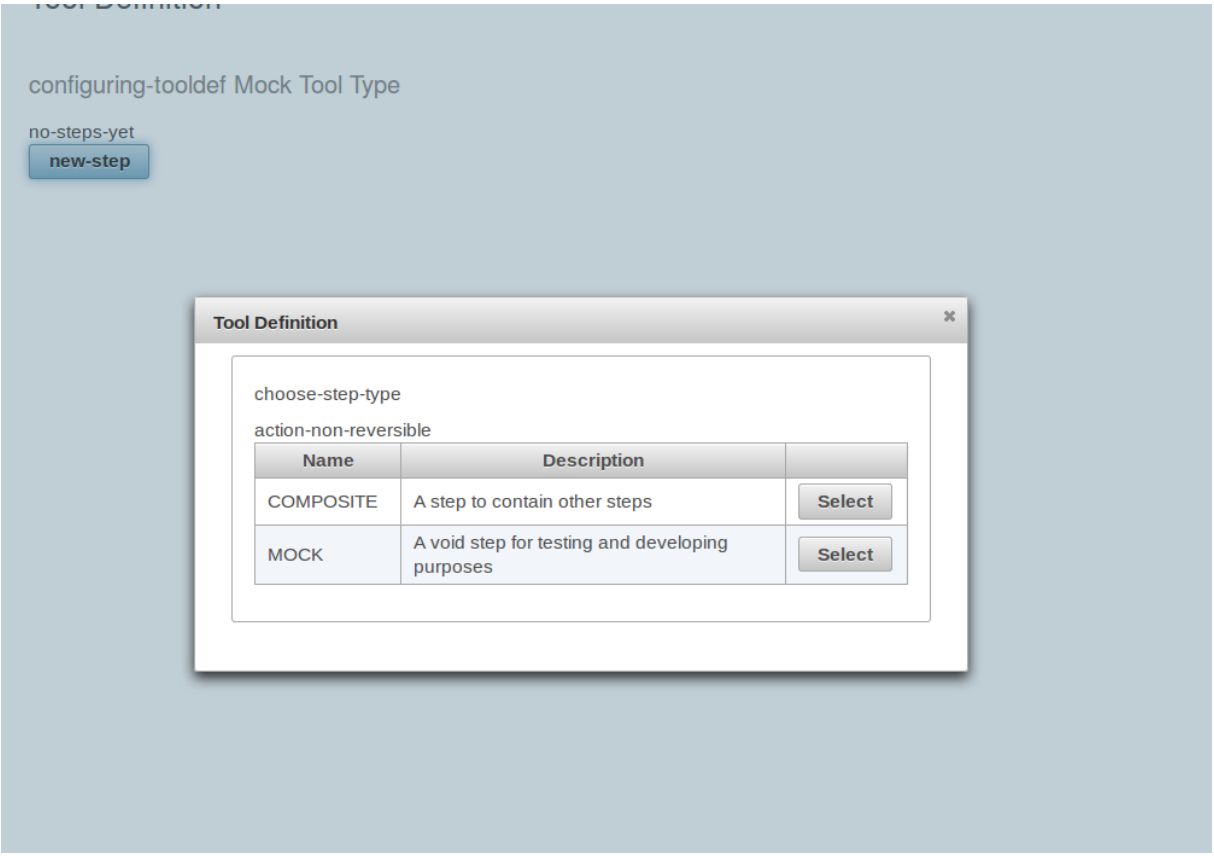


Figure 23: Dialog to choose the new step type

difficulties founded, as explained in the test design, were regarding the different cases related to the database load, the system should ensure the closed tab had been created in the database before proceeding to delete it from the database, so it should be considered if a tab was just created before deleting it. Finally, after the testing process all issues were solved.

6.10 Tenth Sprint

At this point, and due to an interview with the client, it was realized that steps order should be able to be modified within the tool definition configuration. Up to this moment, steps were added in consecutive order, from first to last, so if users wanted to modify the order the only way was deleting existing steps and creating them again. To develop this new feature it was necessary to modify the back-end since the model did not support steps order, so a refactoring was addressed in this sprint.

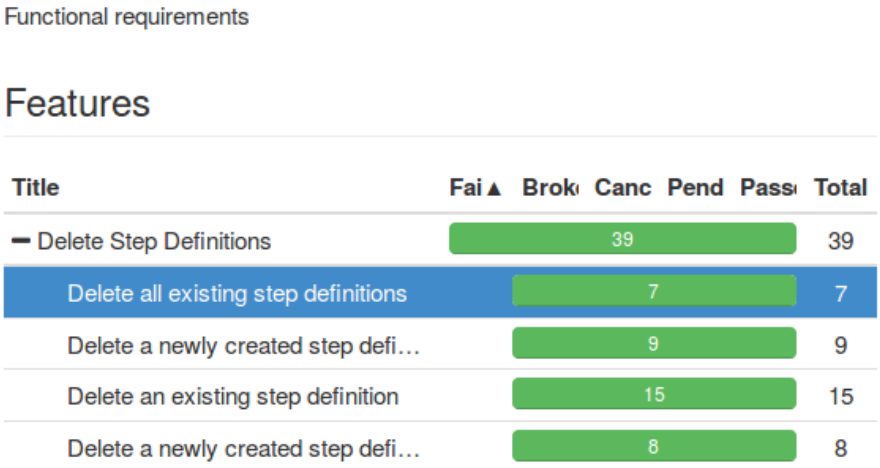


Figure 24: Test scenarios for sprint 9

6.10.1 Test Design

Tests for this sprint are the same tests that had already been passed but checking if the database was also storing and deleting the new added step order. To do that, the `java.sql` library was used and, instead of creating new integration tests which would entail a lot of work to deal with mocks, an `assert` check was included in the step definition in the already existing and passed tests of the eighth sprints. All the database communication was managed by the new class `DBChecker`, so the testing framework was updated as in figure 25. Therefore, it was checked if the new database registers regarding the step order were created.

6.10.2 Implementation

The database update implied, as showed in figure 26, merely adding a step order field in the N to N relationships between `CSDef_StepDef` and `CS_Step` tables. The problem arose when trying to do it through the *Liferay Service Builder (LSB)*, after some researching and attempts it was clear that it was not supported. So finally the *LSB* feature to create N to N relationships was not used, and the relationships tables were created just any other table with *LSB* and adding the new `stepIndex` field.

As the database was modified, the related methods for adding and removing were also modified to include the new field. Also two new methods to update the relationship called `reorderStepDef` and `saveNewStepsOrder` were created to implement the new feature in the back-end.

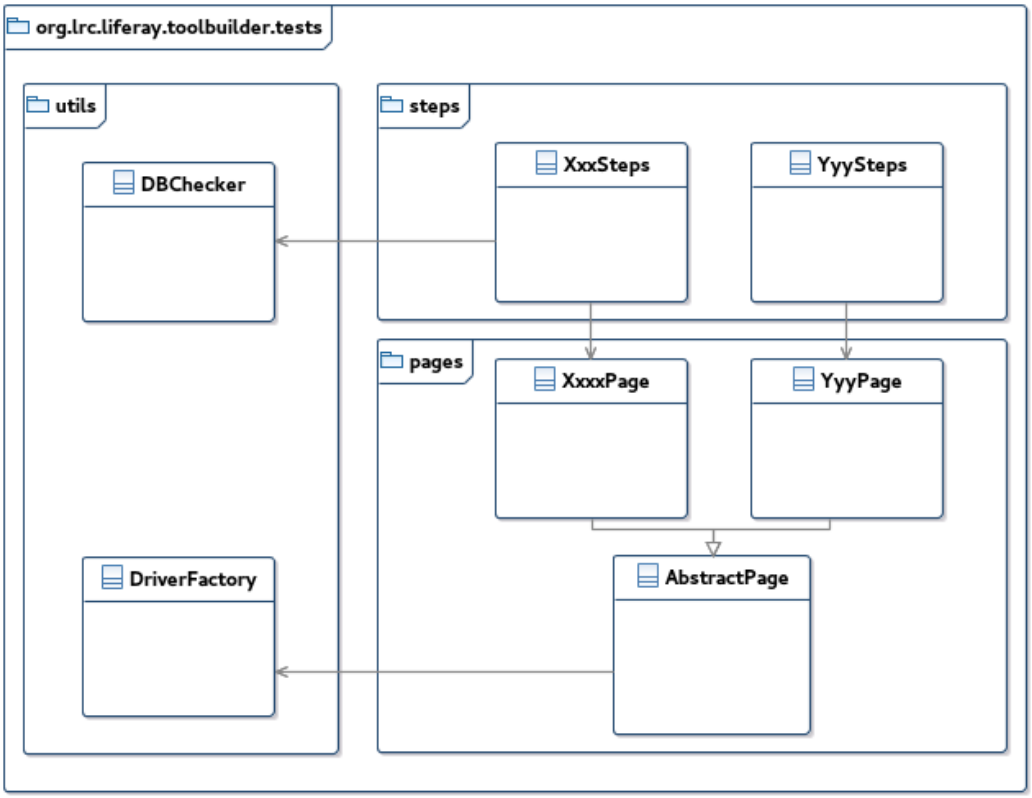


Figure 25: Testing framework with `DBChecker`

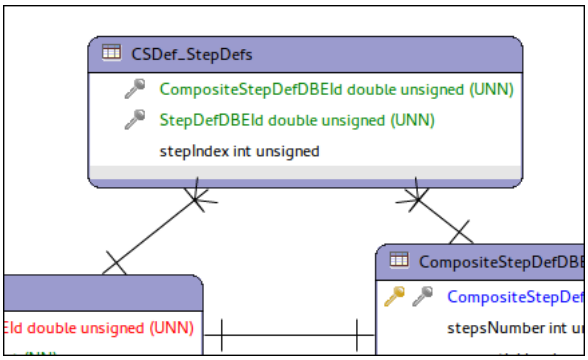


Figure 26: Database modification of relationship table

6.11 Eleventh Sprint

Once the back-end supports steps order it was time to modify the interface so admin users can change the order of the steps in a tool definition. It involved looking the possibilities of doing it in a user-friendly drag-and-drop way with the current tabs view. Given the impossibility of doing that with the existing Primefaces `<p:tab />` tag, the whole steps interface was redesigned.

6.11.1 Test Design

Tests for this sprint are those to move the steps but with regression tests from eighth sprint in order to assure the proper functioning of the redesigned interface. Some tests were developed for this feature, from simple steps sorting one step to more complex sort actions involving several steps. There were found difficulties to drag and drop thorough the Selenium API as it does not do it in a graphic way, so it did not trigger the events properly. Then a different solution was adopted with the Java Robot object (`java.awt.Robot`), which worked perfectly to drag and drop the objects simulating the user actions. As shown in 27, eight different cases were tested.

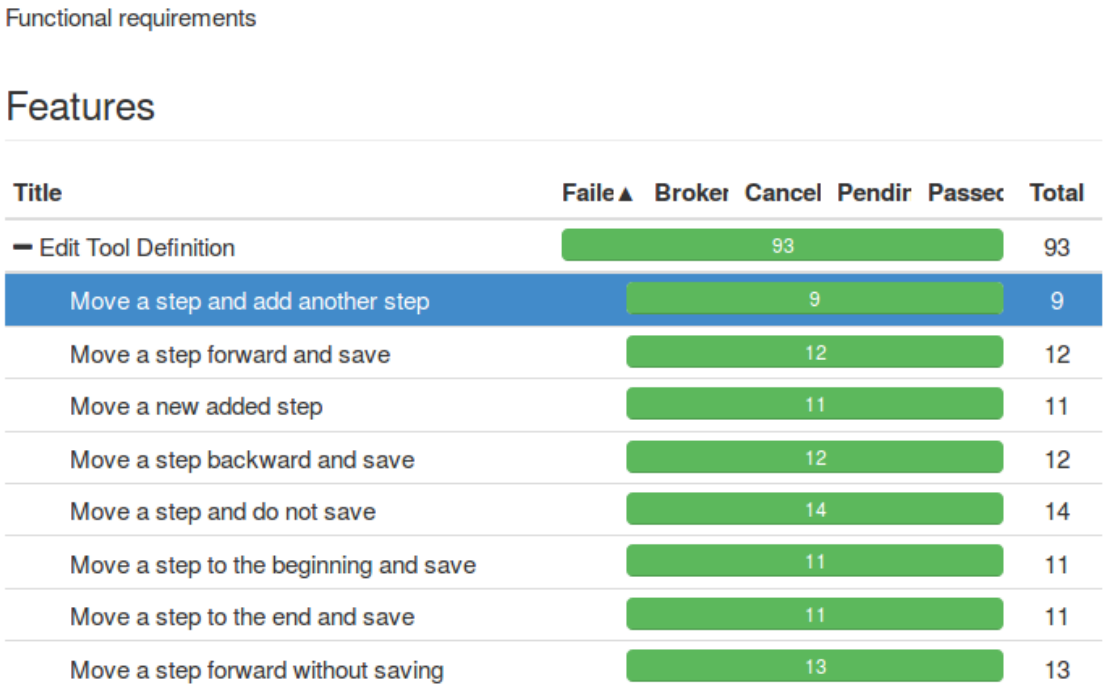


Figure 27: Test scenarios for sprint 11

6.11.2 Implementation

As explained, the implementation of this feature involved changing the step definitions interface. From a tabs system it was changed to a panels system where panels (which are indeed steps containers) could be moved so their order was changed. Creating and deleting steps features remained the same since the Primefaces panels have also a `closable` property. To save the new order a save-order-steps button was added, so if not pressed the new order is not stored in the database. In the new interface is shown.

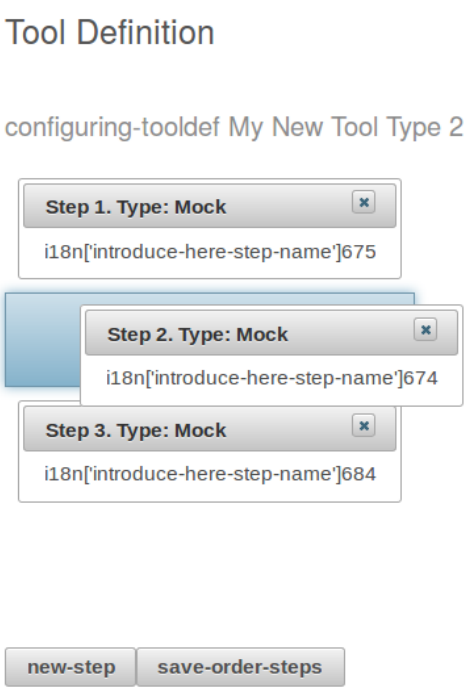


Figure 28: Panels interface to reorder step definitions

6.12 Twelfth Sprint

At this moment, client and developer realized that it was not possible to delete Tool Definitions and it could be needed, so this feature should be added. Also it was necessary to avoid the user to delete or edit tool definitions which are selected or which have already created tool instances so there were some restrictions which should be added.

6.12.1 Test Design

Following the BDD methodology, the behaviors were designed for both features in order to create their corresponding tests:

Deleting A delete button exists for each tool definition so when the user clicks on it a dialog to confirm will appear, if it is confirmed the tool definition will disappear from the view. To check if it was been also deleted in the database the previously created `DBChecker` object was used.

Restricting When a tool definition is selected or it has created tool instance it will be impossible to delete or edit it. So buttons should be disabled and a tool-tip will showed when moving the mouse over them. Four scenarios were created for this feature: three for the cases of not allowing edit or deletion and one where the system must allows it.

As shown in 29, at this moment regression tests were passed to ensure the compatibility of these new features with the rest of the system.

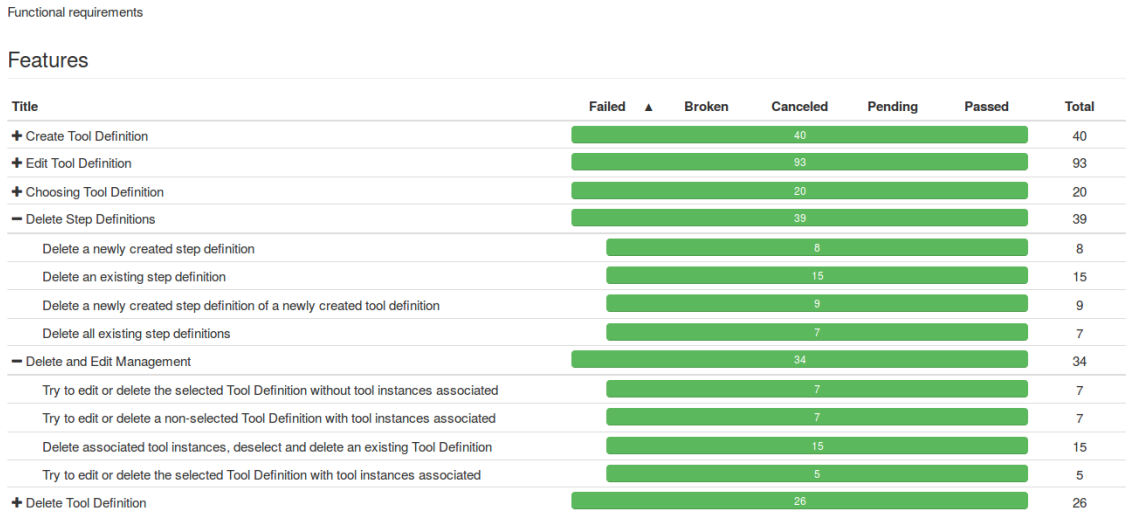


Figure 29: Regression tests passed in sprint 12

6.12.2 Implementation

Coding the delete button only required adding a new button to each row of the `<p:dataTable>` inside `adminView.xhtml`. That button calls the `removeToolDef(String toolDefName)` function from the `FactoryBean` object, which is responsible of calling the persistent object and all the steps to ensure their deletion.

Regarding the restrictions, they were implemented disabling the corresponding button whether the tool definition is selected or it has existing tool instances, it done through the Primefaces `disabled` property. Also a tool-tip was added when the mouse is over a disabled button to show why it has been disabled. One found issue was that the Primefaces tool-tip implementation did not work properly if was linked to `commandButton`, so finally it was attached to a new div which contains all the buttons so the result is the same.

7 Results and Discussion

At the end of this project, a working and tested first phase of a Liferay portlet to guide through the RAM has been developed. This portlet provides a recurrent structure which will allow to develop the necessary steps and substeps to create not only a web application to guide through RAM, but also through any other tool structured in - sequential or not - steps and substeps. Thus, the scalability of the software has been increased following the customer's modifications during the project.

As it can be noted in the development process (section §6), a large proportion of the developing efforts have been aimed to create an administration portlet to

configure the web tool. It will be this portlet, when customized steps are developed, which allows users to create their own web applications to guide through different processes or tools. This has become particularly relevant during the project because the application has been redesigned to be useful not only for a RAM web tool.

Initially for testing purposes, but also to make easier the development of the steps, a mock step has been included which can be used as a template for new steps development. This mock step has been well documented and it has the required structure for the new steps to be included in the system along with the different methods needed. Also a guide for developers has been included in Appendix A: Guide to develop steps. In this way, it is intended to facilitate the extension of the web tool through the creation of new step types.

Concerning the methodology, adopting new techniques has required an extra effort and time in setting-up frameworks and procedures. Undoubtedly it has been worth in order to improve the development methodology and hence the quality, scalability and future development opportunities. Nonetheless, it has also implied changes and adaptations during the development which led to a learning process and an effort to change procedures. With a view to future projects, it would be desirable to establish methodology and processes at the beginning of the project in order to well set-up the development framework and avoid on the fly changes which contribute to slow down the development.

On the other hand, the biggest issue in the development process has been the estimation of the time needed to develop each of the sprints, that has usually been too optimistic. In addition to the code refining in order to solve bugs found in the testing process, which finally took longer than expected, it has also been necessary in many cases to improve the knowledge about the technology, either Primefaces, JSF or even Liferay itself. Moreover the tests development, particularly the Selenium support code and the web elements management, demanded also more time than expected. To sum up, roughly the 50% of the time has been used in development the web application and the other half of the time has been invested in setting up the testing framework and testing itself. It is difficult to estimate the time spent on the project as it has been discontinued and shared with other works, so another lesson learned is the usefulness of maintaining a time log which could be easily implemented if using time management techniques such as *Pomodoro*[11]. Since it has been introduced lately in the project, the estimation of the time spent has been done using those late sprints, which could have been about 40 hours each sprint.

Regarding to the quality assurance, the testing framework which has been created and the fact of using a test-first methodology were aimed to offer a significant degree of quality, ensuring the customer requirements are accomplished without errors. This test-first methodology (particularly BDD as explained in section 3.3) has entailed a mind change in the way the development process is conceived. Since the test are developed before coding, the accomplishment of functional requirements is the core, and code is developed without such a deep previous thinking as it is done using cascade classic methodologies.

Initially, that way of developing can cast doubts if we are used to classical approaches, but when you begin coding you realize that is a much more effective

method. Regarding the doubts cast about how good designed can be a software developed in this way I have realize two main things: the first is that indeed design is thought before coding even if it is minimal, but you think at the design also when you create tests, so it is not just coding in an unstructured way. The second one is that, when some code is not as readable or structured as necessary, a refactoring process can be done in order to improve it and reduce complexity, so code remains structured and intelligible. Particularly in this project, creating symmetric structures when refactoring has simplified the design and made it more comprehensible, as shown in section 6.4 and section A.2.

As stated during previously in this document, the final product of this project is a structure to develop a Ram Assessment for Markets application, that is why the screenshots showed in Appendix C do not look as the mockups of Appendix B. Either way, the correspondence between both designs can be guessed in figure 33, which slightly matches with Home mockup at page 62.

It is also worth pointing out that tests related to Quadrant 4 (4.4) have not been carried out. But it does not mean that characteristics associated to Quadrant 4 have not been considered. In fact compatibility have been always kept in mind in the choice of the technologies from the general aspects (creating a Liferay portlet) to the more specific ones (creating interfaces with Primefaces). Also reliability has been always considered by improving tests, both automated and human guided ones. And of course scalability which, as it has been explained , has been one of the main goals of the project. Lastly and for the future, performance tests should been carried out in order to find improvements, but it has not been an objective in this first phase.

8 Conclusion

At the end of the project it is possible to state that it has provided a functional quality software which fits the requirements of the customer. The created software will allow to develop any kind of steps to create web applications to guide through any tool or process structured in steps and substeps. So it can not only be RAM, but also many others humanitarian methodologies as EMMA or HES.

On the other hand, the objectives of the project have been achieved as a result of using adequate developing technologies which involve the customer and ensure their requirements are kept as the core of the project, in addition to implement quality assurance; particularly important as this point is the choice of adequate technologies which can make the software usable by the customer.

So finally, a learning process has been carried out during the project which have resulted in a useful software and a working framework to scale it in the future and to carry on with the development.

9 Bibliography

References

[1] Behavior driven development wiki. http://behaviourdriven.org/.

[2] Introducing bdd. https://dannorth.net/introducing-bdd/.

[3] *The Java EE 5 Tutorial*. http://java.sun.com/javaee/5/docs/tutorial/doc/.

[4] Manifesto for agile software development, 2001. http://agilemanifesto.org/.

[5] *Rapid Assessment for Markets*, 2014. https://www.icrc.org/eng/assets/files/publications/icrc-002-4199.pdf.

[6] Javadoc, 2015. http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html.

[7] Branching (version control) - wikipedia, 2016. https://en.wikipedia.org/wiki/Branching_ (version_control).

[8] The scrum guide, 2016. http://www.scrumguides.org/.

[9] Test driven development - wikipedia, 2016. https://en.wikipedia.org/wiki/Test-driven_development.

[10] David J. Anderson and Andy Carmichae. *Essential Kanban condensed*. LeanKanban University Press, 2016.

[11] Francesco Cirillo. *The Pomodoro Technique*. 2009. http://www.pomodorotechnique.com/.

[12] Lisa Crispin and Janet Gregory. *Agile Testing: A Practical Guide for Testers and Agile Teams*. Addison-Wesley Professional, 1 edition, 2009.

[13] Erich Gamma, Richard Helm, and R. J. J. V. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[14] Henrik Kniberg and Mattias Skarin. *Kanban and Scrum - Making the Most of Both*. InfoQ, 2010.

[15] Lasse Koskela. *Test Driven: Tdd and Acceptance Tdd for Java Developers*. Manning Publications, 2007.

[16] Andrew Thu Pham and David Khoi Pham. *Business-Driven IT-Wide Agile (scrum) and Kanban (Lean) Implementation: An Action Guide for Business and IT Leaders*. Productivity Press, 2012.

[17] Mary Poppendieck and Tom Poppendieck. *Lean Software Development: An Agile Toolkit*. Addison-Wesley Professional, 2003.

[18] Ken Pugh. *Lean-Agile Acceptance Test-Driven Development: Better Software Through Collaboration*. Net Objectives Lean-Agile Series. Addison-Wesley Professional, 2011.

[19] Seb Rose, Matt Wynne, and Aslak Hellesoy. *The Cucumber for Java Book: Behaviour-Driven Development for Testers and Developers*. Pragmatic Bookshelf, 2015.

[20] Ken Schwaber. Scrum development process. In *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA*, pages 117–134, 1995.

APPENDICES

A Guide to develop steps

One of the main features of the portlet is to allow developers to create new steps in order to extend features and create new tools, this guide shows how new steps should be created.

A.1 Technologies

As the portlet has been developed using JSF 2.1, the standard JSF version for Liferay 6.2, the steps should also be developed with JSF 2.1. It is recommended the previous reading of the Liferay tutorial "Writing a JSF application using Liferay Faces"[20] in order to understand how JSF portlets are developed in Liferay.

A.2 Step structure

The `MOCK` step can be used as a template to create new steps since it is well documented and commented to make easier the development of new steps. It can be downloaded from the github repository.

The core of new step structure must follow the general structure designed for all the portlet, that is to say a symmetric structure comprised of the step definition part and the step instance part:

- The **step definition** regards all those options and configurations defined when creating the tool and including the step, namely in the `tool-definition` portlet created in the Sixth Sprint. For example, if we create a generic step to attach files the step definition may allow to define a text to show in the step like "Add files for..." and the max number of files which can be added in the step. It should be a class inheriting from `StepDef` class.

- The **step instance** regards all those options or actions that can be taken when the user access the step in the `tool-implementation` portlet. For example in the previous case of a step to attach files, the step instance will manage the specific files added for a step in a particular tool instance. It should be a class inheriting from `Step` class.

These two classes should be packaged as `org.lrc.liferay.toolbuilder.steps.XXX`, where XXX is the name of the step, and should connect with the persistent layer, to

[20] https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/writing-a-jsf-application-using-liferay-faces

be consistent with the rest of the portlet it is recommended to build the persistent layer with *Liferay Service Builder*[21].

Besides the two core classes, some backing beans may be needed for the views. It is recommended to use *View Scoped* or *Request Scoped* beans and if some variable is needed to persist during the session save it in the `ToolSession` bean, which is the *Session Scoped* bean and can be accessed from any other bean as a JSF `ManagedProperty`. It is important to note that all beans inherit from `AbstractBaseBean`. To maintain consistency with the rest of the portlet these beans should be packaged as org.lrc.liferay.toolbuilder.bean.steps.XXX, where XXX is the name of the step.

On the other hand, and related to the views, at least two views should be created: one regarding the step definition and other one regarding step instance. By convention they should be located in the directory `WEB-INF/views/steps/XXX/` under the docroot, where XXX is - again - the name of the step. In 30 is shown a diagram of the classes and views which may be developed, in blue those classes related to the step definition and in red those ones related to the step instance.

A.3 Installing new step type

To install a new step type it is necessary to call it from the `StepFactory` class static function `installStep`:

```
public static void installStep(String stepType, String stepDescription,
        String namespace, String className)
```

`stepType` The name of the step type, by convention in capital letters.

`stepDescription` A description of the step type.

`namespace` The package where the core classes are located.

`className` The name of the step instance class, to get the name of the step definition class it will be added "Def" at the end.

For the example of a step to add files the call could be like this:

```
StepFactory installStep("FILES_ATTACH", "A step to attach files",
        "org.lrc.liferay.toolbuilder.steps.filesAttach",
        "FilesAttachStep");
```

[21] https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/generating-a-persistence-framework-using-service-builder

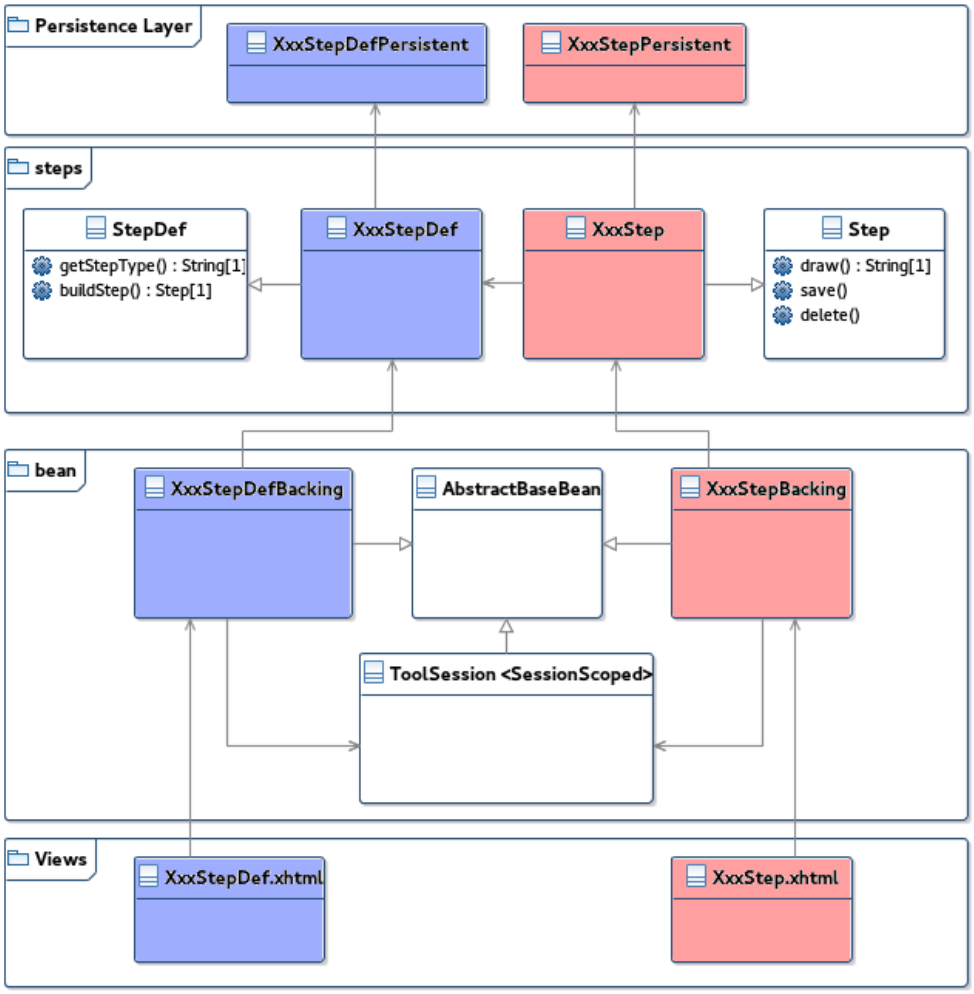


Figure 30: Diagram for new Steps, in blue classes for the step definition and in red classes for step instance

B Mockups of the RAM web app

As explained in the subsection Preliminary work, mockups for the final web application were designed with the customer before beginning the development process. Although an interactive version of those mockups can be found on the following URL:

`http://www.livelihoodstoolbox.org/RAM_design/`

In the next pages are shown the main mockups which were designed as before beginning the application development. Those mockups are preceded by the application navigation map which was designed in order to well understand the structure.

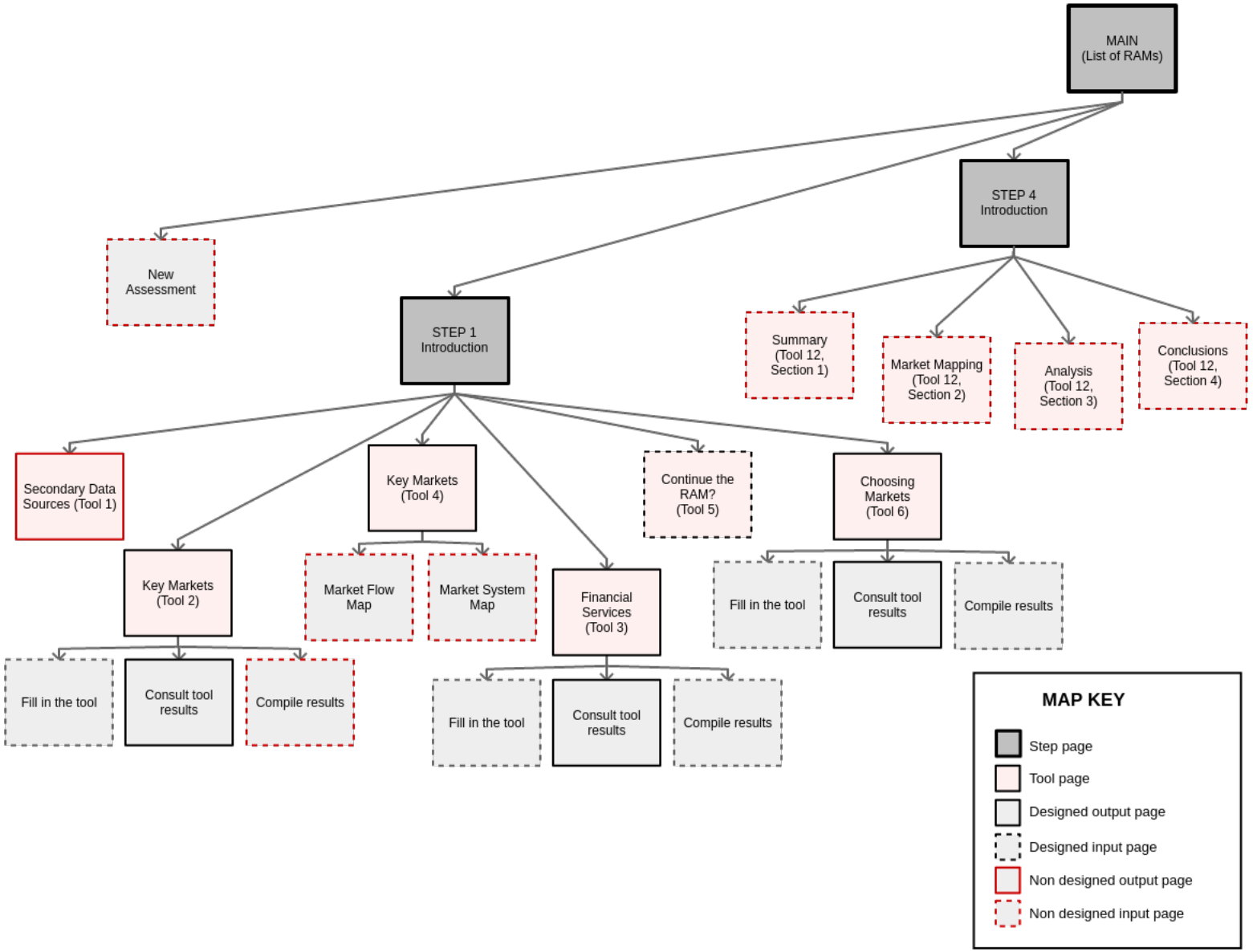


Figure 31: Navigation map for the RAM web app (part 1)

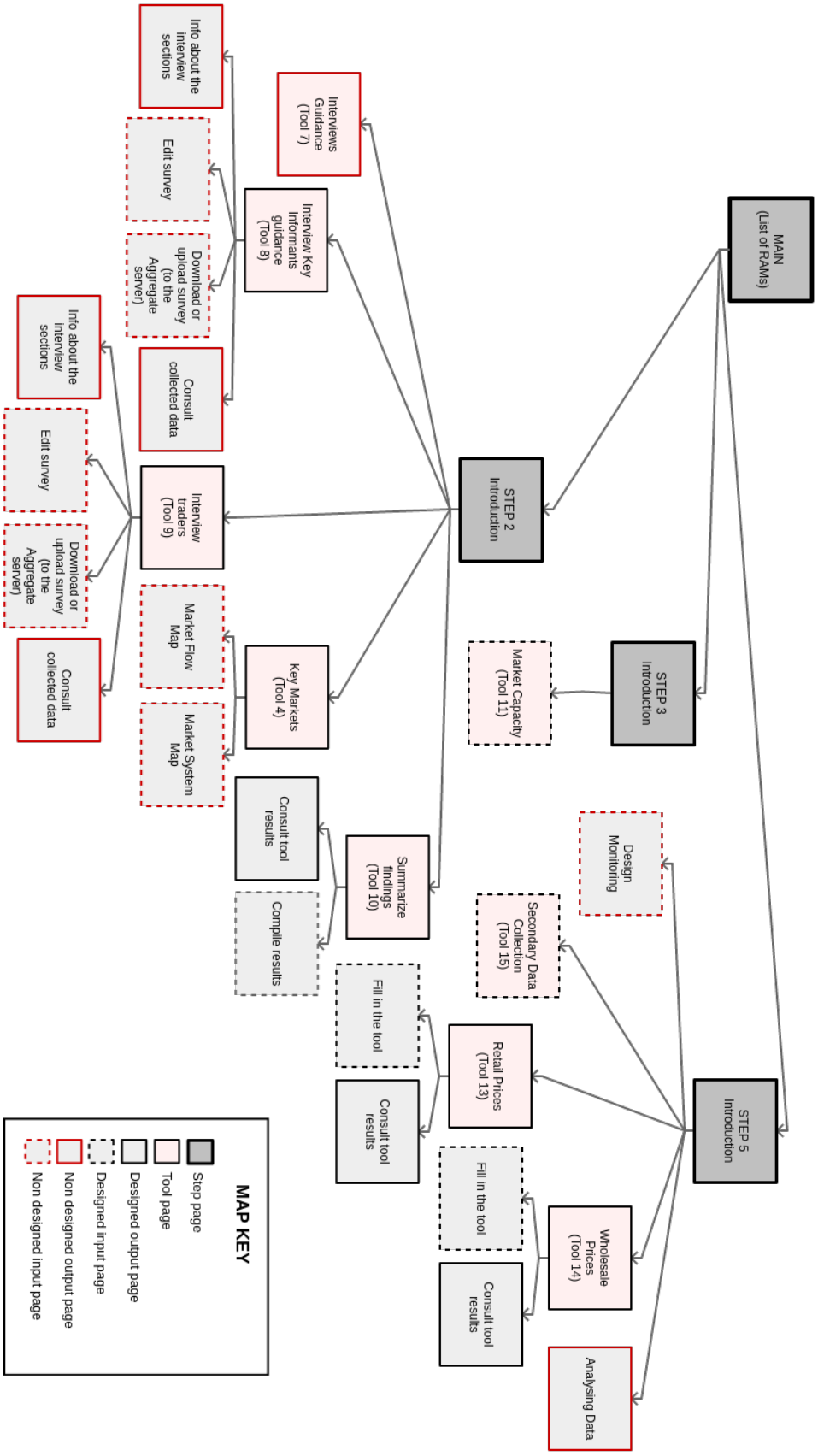


Figure 32: Navigation map for the RAM web app (part 2)

Inicio

Livelihoods Centre
knowledge creation | knowledge sharing | knowledge networking

LANGUAGE | CONTACT |

WHAT ARE LR? ABOUT RESOURCES TRAINING NETWORKING ACCESS

Resources > RAM Webtool

Rapid Assessment for Markets

Down below you'll find the available RAM's so you can consult them or participate (if not finished).

RAM Name	Year	Location	State
Rapid Assessment for millet market	*2016*	*Maradi, NIger*	*Step 1*
Rapid Assessment for corn seeds market	*2016*	*Comayagua, Honduras*	*Step 4*
Rapid Assessment for labour market	*2015*	*Kathmandu, Nepal*	*Finished*
Assessment for wood market	2014	Yurimaguas, Peru	*Finished*

Create a new Assessment

Contact | Legal warning

follow us on:

creating knowledge | **sharing** knowledge | **exchanging** knowledge

Federación Internacional de Sociedades de la Cruz Roja y de la Media Luna Roja

En colaboración con: Cruz Roja Española

Financed by ECHO

Humanitarian Aid and Civil Protection

Step 1

Rapid Assessment for millet markets in Maradi, Niger, 2015

Step 1: Defining the scope and content of the assessment

Step 1 of the RAM is dedicated to defining the scope and focus of the assessment and gathering initial information on the availability and accessibility of financial services. It provides RAM users with guidance and tools (Tools 1 – 6) that lead the user to answer a series of questions that include:

- What key commodities are needed by the shock-affected population?
- (i.e. What market systems should the RAM focus on?)
- What financial services are available to the shock-affected population?
- Does it make sense to continue the RAM?
- If so, what marketplaces should the RAM focus on?

This is an important part of the RAM as it enables an under standing of whether or not the traders in/near the affected area can provide the required commodities for the population that use/can use that market.

Complete Step guide explaining the sections

Step 1.2

Livelihoods Centre
knowledge creation | knowledge sharing | knowledge networking

LANGUAGE | CONTACT |

WHAT ARE LR? ABOUT RESOURCES TRAINING NETWORKING ACCESS

Resources > RAM Webtool

Rapid Assessment for millet markets in Maradi, Niger, 2015

CLOSE RAM
PROPERTIES

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5

Step 1: Defining the scope and content of the assessment

- INTRODUCTION
- SEC. DATA SOURCES
- KEY MARKETS
- MARKET MAPS
- FINANCIAL SERVICES
- CONTINUE THE RAM?
- CHOOSING MARKETS

Under construction
Secondary data sources list

Contact | Legal warning

follow us on:

creating knowledge | **sharing** knowledge | **exchanging** knowledge

Federación Internacional de Sociedades de la Cruz Roja y de la Media Luna Roja

En colaboración con: Cruz Roja Española

Financed by ECHO

Humanitarian Aid and Civil Protection

Step 1.3

Livelihoods Centre
knowledge creation | knowledge sharing | knowledge networking

LANGUAGE | CONTACT |

WHAT ARE LR? ABOUT RESOURCES TRAINING NETWORKING ACCESS

Resources > RAM Webtool

Rapid Assessment for millet markets in Maradi, Niger, 2015

CLOSE RAM
PROPERTIES

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5

Step 1: Defining the scope and content of the assessment

- INTRODUCTION
- SEC. DATA SOURCES
- **KEY MARKETS**
- MARKET MAPS
- FINANCIAL SERVICES
- CONTINUE THE RAM?
- CHOOSING MARKETS

This tool requires the involvement of all RAM team members and key informants as it is important that everyone has a common understanding of the situation, the questions that need answering and decisions made.

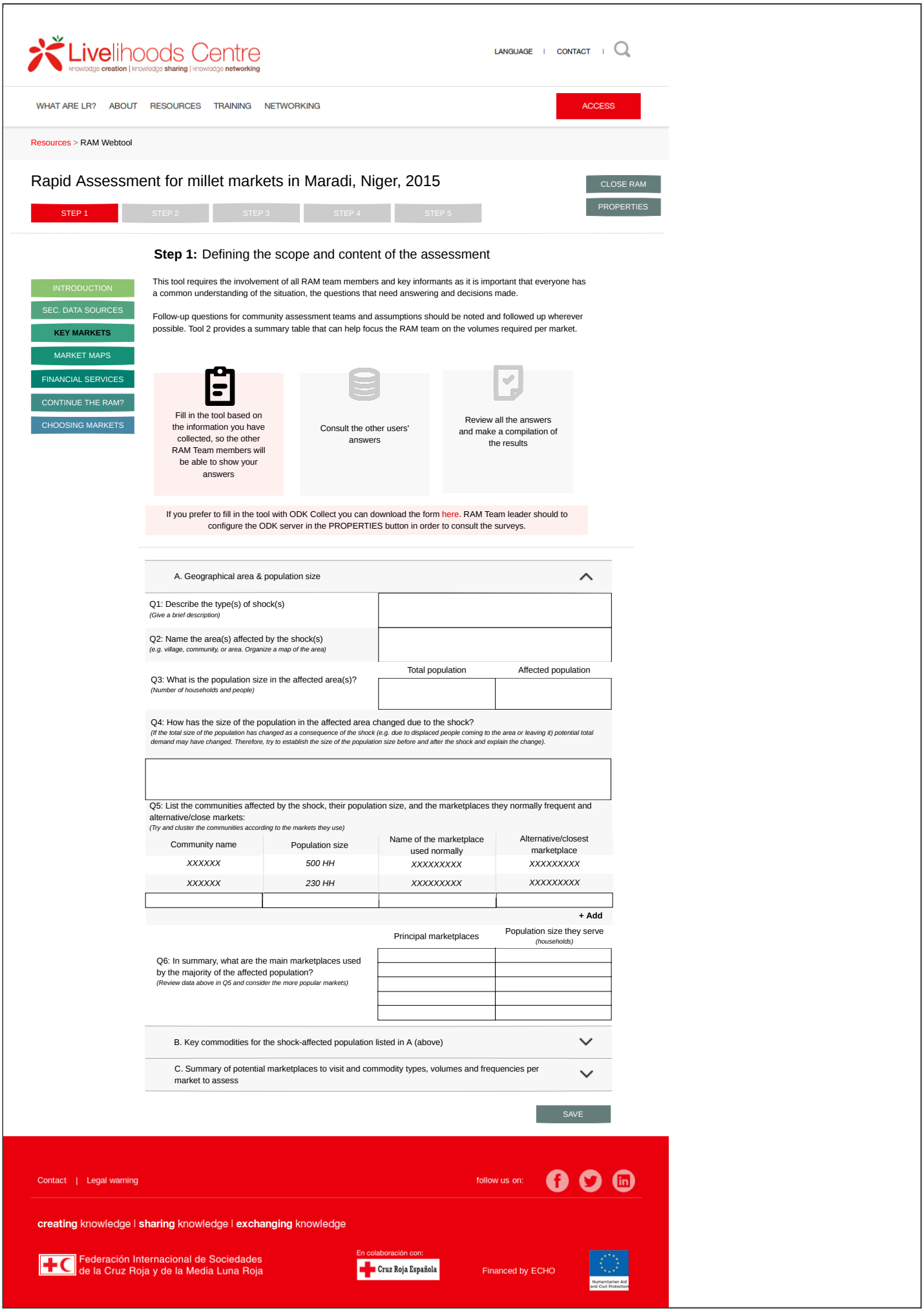
Follow-up questions for community assessment teams and assumptions should be noted and followed up wherever possible. Tool 2 provides a summary table that can help focus the RAM team on the volumes required per market.

Fill in the tool based on the information you have collected, so the other RAM Team members will be able to show your answers

Consult the other users' answers

Review all the answers and make a compilation of the results

Step 1.3.1


Livelihoods Centre
knowledge creation | knowledge sharing | knowledge networking
LANGUAGE | CONTACT |
WHAT ARE LR? ABOUT RESOURCES TRAINING NETWORKING
ACCESS
Resources > RAM Webtool
Rapid Assessment for millet markets in Maradi, Niger, 2015
CLOSE RAM
PROPERTIES
STEP 1 STEP 2 STEP 3 STEP 4 STEP 5
Step 1: Defining the scope and content of the assessment
INTRODUCTION
SEC. DATA SOURCES
KEY MARKETS
MARKET MAPS
FINANCIAL SERVICES
CONTINUE THE RAM?
CHOOSING MARKETS
This tool requires the involvement of all RAM team members and key informants as it is important that everyone has a common understanding of the situation, the questions that need answering and decisions made.
Follow-up questions for community assessment teams and assumptions should be noted and followed up wherever possible. Tool 2 provides a summary table that can help focus the RAM team on the volumes required per market.
Fill in the tool based on the information you have collected, so the other RAM Team members will be able to show your answers
Consult the other users' answers
Review all the answers and make a compilation of the results
If you prefer to fill in the tool with ODK Collect you can download the form here. RAM Team leader should to configure the ODK server in the PROPERTIES button in order to consult the surveys.
A. Geographical area & population size
Q1: Describe the type(s) of shock(s)
(Give a brief description)
Q2: Name the area(s) affected by the shock(s)
(e.g. village, community, or area. Organize a map of the area)
Total population
Affected population
Q3: What is the population size in the affected area(s)?
(Number of households and people)
Q4: How has the size of the population in the affected area changed due to the shock?
(If the total size of the population has changed as a consequence of the shock (e.g. due to displaced people coming to the area or leaving it) potential total demand may have changed. Therefore, try to establish the size of the population size before and after the shock and explain the change).
Q5: List the communities affected by the shock, their population size, and the marketplaces they normally frequent and alternative/close markets:
(Try and cluster the communities according to the markets they use)
Community name
Population size
Name of the marketplace used normally
Alternative/closest marketplace
XXXXXX
500 HH
XXXXXXXXX
XXXXXXXXX
XXXXXX
230 HH
XXXXXXXXX
XXXXXXXXX
+ Add
Principal marketplaces
Population size they serve
(households)
Q6: In summary, what are the main marketplaces used by the majority of the affected population?
(Review data above in Q5 and consider the more popular markets)
B. Key commodities for the shock-affected population listed in A (above)
C. Summary of potential marketplaces to visit and commodity types, volumes and frequencies per market to assess
SAVE
Contact | Legal warning
follow us on:
creating knowledge | sharing knowledge | exchanging knowledge
Federación Internacional de Sociedades de la Cruz Roja y de la Media Luna Roja
En colaboración con:
Cruz Roja Española
Financed by ECHO
Humanitarian Aid and Civil Protection

Step 1.3.2

Rapid Assessment for millet markets in Maradi, Niger, 2015

Step 1: Defining the scope and content of the assessment

INTRODUCTION
SEC. DATA SOURCES
KEY MARKETS
MARKET MAPS
FINANCIAL SERVICES
CONTINUE THE RAM?
CHOOSING MARKETS

This tool requires the involvement of all RAM team members and key informants as it is important that everyone has a common understanding of the situation, the questions that need answering and decisions made.

Follow-up questions for community assessment teams and assumptions should be noted and followed up wherever possible. Tool 2 provides a summary table that can help focus the RAM team on the volumes required per market.

Fill in the tool based on the information you have collected, so the other RAM Team members will be able to show your answers

Consult the other users' answers

Review all the answers and make a compilation of the results

Author	Date
Elvira Rornen	*16/10/2015*
Kevin Worgman	*17/10/2015*
Ricardo Bustillos	*17/10/2015*
Ana Peláez	*18/10/2015*

Step 1.3.3_idea

Rapid Assessment for millet markets in Maradi, Niger, 2015

CLOSE RAM
PROPERTIES

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5

Step 1: Defining the scope and content of the assessment

This tool requires the involvement of all RAM team members and key informants as it is important that everyone has a common understanding of the situation, the questions that need answering and decisions made.

Follow-up questions for community assessment teams and assumptions should be noted and followed up wherever possible. Tool 2 provides a summary table that can help focus the RAM team on the volumes required per market.

Fill in the tool based on the information you have collected, so the other RAM Team members will be able to show your answers

Consult the other users' answers

Review all the answers and make a compilation of the results

A. Geographical area & population size

Q1: Describe the type(s) of shock(s) *(Give a brief description)*	
Q2: Name the area(s) affected by the shock(s) *(e.g. village, community, or area. Organize a map of the area)*	

Q3: What is the population size in the affected area(s)? *(Number of households and people)*	Total population	Affected population

Q4: How has the size of the population in the affected area changed due to the shock?
(If the total size of the population has changed as a consequence of the shock (e.g. due to displaced people coming to the area or leaving it) potential total demand may have changed. Therefore, try to establish the size of the population size before and after the shock and explain the change).

Q5: List the communities affected by the shock, their population size, and the marketplaces they normally frequent and alternative/close markets:
(Tiy and cluster the communities according to the markets they use)

Community name	Population size	Name of the marketplace used normally	Alternative/closest marketplace
XXXXXX	*500 HH*	*XXXXXXXXX*	*XXXXXXXXX*
XXXXXX	*230 HH*	*XXXXXXXXX*	*XXXXXXXXX*

+ Add

Q6: In summary, what are the main marketplaces used by the majority of the affected population? *(Review data above in Q5 and consider the more popular markets)*	Principal marketplaces	Population size they serve *(households)*

B. Key commodities for the shock-affected population listed in A (above)

C. Summary of potential marketplaces to visit and commodity types, volumes and frequencies per market to assess

SAVE

Step 1.4

Livelihoods Centre
knowledge creation | knowledge sharing | knowledge networking

LANGUAGE | CONTACT |

WHAT ARE LR? ABOUT RESOURCES TRAINING NETWORKING ACCESS

Resources > RAM Webtool

Rapid Assessment for millet markets in Maradi, Niger, 2015

CLOSE RAM
PROPERTIES

STEP 1 STEP 2 STEP 3 STEP 4 STEP 5

Step 1: Defining the scope and content of the assessment

INTRODUCTION
SEC. DATA SOURCES
KEY MARKETS
MARKET MAPS
FINANCIAL SERVICES
CONTINUE THE RAM?
CHOOSING MARKETS

Drafting market maps in this step enables team discussion by providing an initial visualization of the preliminary findings. Markets can be mapped in different ways. Common to all market maps is that they need to be simple and easy to interpret. Thus, RAM users should focus on aspects that are important for the market system and play a role with respect to the shock and a potential relief intervention.

Two types of market maps are introduced here:

A **Production and Market Flow Map** is a useful tool to represent commodity flows.It describes the geographic flows and points of exchange (marketplaces) for a commodity from the region in which it is produced to the region it is consumed – i.e. the target region.

Market System Maps represent markets graphically by three linear components: the market chain; the supporting infrastructure and services; and the external environment. In a second step it also accounts for the effects of the shock on the market system.

Step 1.5

Rapid Assessment for millet markets in Maradi, Niger, 2015

CLOSE RAM
PROPERTIES

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5

Step 1: Defining the scope and content of the assessment

Understanding financial services is an important part of identifying appropriate transfer mechanisms (i.e. in-kind or cash-based), if assistance is to be provided. The aim is to develop an understanding of:

- The financial institutions that are present in the area of interest.
- The extent to which the target population uses these financial institutions.
- Potential factors that can limit the target population's access to these financial institutions.

The questions should be discussed by RAM team members and representatives from the finance department. When and where it is feasible, key informants may be invited to join the discussion. Tool 3 can be used to guide the discussion and note the respective conclusions.

Fill in the tool based on the information you have collected, so the other RAM Team members will be able to know your answers

Consult the other users' answers

Review all the answers and make a compilation of the results

Implements RAM's Tool 3 in a similar way to "KEY MARKETS" section

Step 1.6

Livelihoods Centre
knowledge creation | knowledge sharing | knowledge networking

LANGUAGE | CONTACT |

WHAT ARE LR? ABOUT RESOURCES TRAINING NETWORKING ACCESS

Resources > RAM Webtool

Rapid Assessment for millet markets in Maradi, Niger, 2015

CLOSE RAM
PROPERTIES

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5

Step 1: Defining the scope and content of the assessment

- INTRODUCTION
- SEC. DATA SOURCES
- KEY MARKETS
- MARKET MAPS
- FINANCIAL SERVICES
- **CONTINUE THE RAM?**
- CHOOSING MARKETS

RAM team needs to decide whether it is necessary and possible to continue the assessment. This decision is based on understanding the needs of the shock-affected people in an area and the accessibility of markets used to cover the needs.

Answer following questions based on last substeps results. It is recommended that the representatives of management and the logistics and finance departments answer the questions, too.

Do shock-affected households use local markets to buy or sell commodities?

(●) Yes ( ) No

Consult related data

Are shock-affected households physically able to access local markets?

( ) Yes ( ) No

Consult related data

Step 1.7

Livelihoods Centre
knowledge creation | knowledge sharing | knowledge networking

LANGUAGE | CONTACT |

WHAT ARE LR? ABOUT RESOURCES TRAINING NETWORKING ACCESS

Resources > RAM Webtool

Rapid Assessment for millet markets in Maradi, Niger, 2015

CLOSE RAM
PROPERTIES

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5

Step 1: Defining the scope and content of the assessment

- INTRODUCTION
- SEC. DATA SOURCES
- KEY MARKETS
- MARKET MAPS
- FINANCIAL SERVICES
- CONTINUE THE RAM?
- **CHOOSING MARKETS**

This tool can be used to assist in the selection of the marketplaces to be visited and in summarizing the commodities of interest (commodity type and quantity). It should be used to guide a discussion among RAM team members and, when and where possible, key informants with knowledge about the marketplaces that shock-affected households use.

The number of marketplaces to be visited is dependent on the size of the team, geographical and logistical constraints, the size/importance of the markets, the number of interviews to be conducted, and the time available. The team should be realistic when estimating the number of marketplaces it can assess and the number of interviews per marketplace.

Fill in the tool based on the information you have collected, so the other RAM Team members will be able to show your answers

Consult the other users' answers

Review all the answers and make a compilation of the results

Implements RAM's Tool 6 in a similar way to "KEY MARKETS" section

STEP FORWARD

Contact | Legal warning

follow us on:

creating knowledge | **sharing** knowledge | **exchanging** knowledge

Federación Internacional de Sociedades de la Cruz Roja y de la Media Luna Roja

En colaboración con: Cruz Roja Española

Financed by ECHO

Step 2

Rapid Assessment for millet markets in Maradi, Niger, 2015

Step 2: Collecting market information

INTRODUCTION
INTERVIEWS GUIDANCE
INTERVIEW KEY INFOR.
INTERVIEW TRADERS
MARKET MAPS
SUMMARIZE FINDINGS

In Step 2 of the RAM, the RAM team will collect information necessary to obtain a quick and basic understanding of the market situation, with a focus on the key commodities. It provides the team with guidance and tools to answer the following questions:

- What was the physical damage to the selected marketplaces?
- What were the consequences of the physical damage for the selected marketplaces?
- What is the traders' capacity to supply key commodities since the shock?
- How has the people's demand for the selected key commodities changed since the shock?
- How have the prices for the selected key commodities changed since the shock?

Complete Step guide explaining the sections

Step 2.3

Livelihoods Centre
knowledge creation | knowledge sharing | knowledge networking

LANGUAGE | CONTACT |

WHAT ARE LR? ABOUT RESOURCES TRAINING NETWORKING ACCESS

Resources > RAM Webtool

Rapid Assessment for millet markets in Maradi, Niger, 2015

CLOSE RAM
PROPERTIES

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5

Step 2: Collecting market information

INTRODUCTION
INTERVIEWS GUIDANCE
INTERVIEW KEY INFOR.
INTERVIEW TRADERS
MARKET MAPS
SUMMARIZE FINDINGS

The objective of this initial interview is to get an overview of the marketplace. Ideally, market representatives would be interviewed, but in their absence, officials or representatives with a broad overview of the local marketplace can be involved, for example:

- Local government representatives (e.g. Ministry of Finance);
- Representatives of the chamber of commerce, trader associations, etc.;
- Community representatives (e.g. traditional authorities, elders).

Read some few aspects about the sections to better understand it in order to interview the key informants.

Edit survey to fit your necessities

Download the ODK survey or upload it to your server

Consult collected data

Contact | Legal warning

follow us on:

Step 2.4

Livelihoods Centre
knowledge creation | knowledge sharing | knowledge networking

LANGUAGE | CONTACT |

WHAT ARE LR? ABOUT RESOURCES TRAINING NETWORKING ACCESS

Resources > RAM Webtool

Rapid Assessment for millet markets in Maradi, Niger, 2015

CLOSE RAM
PROPERTIES

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5

Step 2: Collecting market information

INTRODUCTION
INTERVIEWS GUIDANCE
INTERVIEW KEY INFOR.
INTERVIEW TRADERS
MARKET MAPS
SUMMARIZE FINDINGS

The objective of the trader interview is to get detailed information on the selected key commodity markets, the local marketplace, and the traders present. Whenever possible, RAM users should triangulate data they obtain from the traders with the information they received during other trader and market representative interviews. The RAM users should interview both:

- Wholesalers (if possible large ones and small ones), and;
- Retailers (if possible large ones and small ones).

Read some few aspects about the sections to better understand it in order to interview the key informants

Edit survey to fit your necessities

Download the ODK survey or upload it to your server

Consult collected data

Implements RAM's Tool 9 in a similar way to "INTERVIEW KEY INFORMANTS" section

Contact | Legal warning follow us on:

Step 2.5

Livelihoods Centre
knowledge creation | knowledge sharing | knowledge networking

LANGUAGE | CONTACT |

WHAT ARE LR? ABOUT RESOURCES TRAINING NETWORKING ACCESS

Resources > RAM Webtool

Rapid Assessment for millet markets in Maradi, Niger, 2015

CLOSE RAM
PROPERTIES

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5

Step 2: Collecting market information

- INTRODUCTION
- INTERVIEWS GUIDANCE
- INTERVIEW KEY INFOR.
- INTERVIEW TRADERS
- **MARKET MAPS**
- SUMMARIZE FINDINGS

In this step it could be necessary reviewing the market maps. Remember, RAM users should focus on aspects that are important for the market system and play a role with respect to the shock and a potential relief intervention.

The two types of market maps drafted in Step 1 are linked here:

A **Production and Market Flow Map** is a useful tool to represent commodity flows.It describes the geographic flows and points of exchange (marketplaces) for a commodity from the region in which it is produced to the region it is consumed – i.e. the target region.

Market System Maps represent markets graphically by three linear components: the market chain; the supporting infrastructure and services; and the external environment. In a second step it also accounts for the effects of the shock on the market system.

Step 2.6

Livelihoods Centre
knowledge creation | knowledge sharing | knowledge networking

LANGUAGE | CONTACT |

WHAT ARE LR? ABOUT RESOURCES TRAINING NETWORKING ACCESS

Resources > RAM Webtool

Rapid Assessment for millet markets in Maradi, Niger, 2015

CLOSE RAM
PROPERTIES

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5

Step 2: Collecting market information

- INTRODUCTION
- INTERVIEWS GUIDANCE
- INTERVIEW KEY INFOR.
- INTERVIEW TRADERS
- MARKET MAPS
- **SUMMARIZE FINDINGS**

At the end of the data-collection process in each marketplace, the RAM team members should gather and discuss the general findings for that marketplace before leaving it, just in case there are any significant issues that require immediate clarification, and to ensure a common conclusion before presenting the finding to the wider team/team leader.

Consult team members answers

Review all the answers and make a compilation of the results for each marketplace

Implements RAM's Tool 10 in a similar way to "Step 1/KEY MARKETS"

STEP FORWARD

Contact | Legal warning

follow us on:

creating knowledge | **sharing** knowledge | **exchanging** knowledge

Federación Internacional de Sociedades de la Cruz Roja y de la Media Luna Roja

En colaboración con: Cruz Roja Española

Financed by ECHO

Humanitarian Aid and Civil Protection

Step 3

Rapid Assessment for millet markets in Maradi, Niger, 2015

Step 3: Analysing the market information

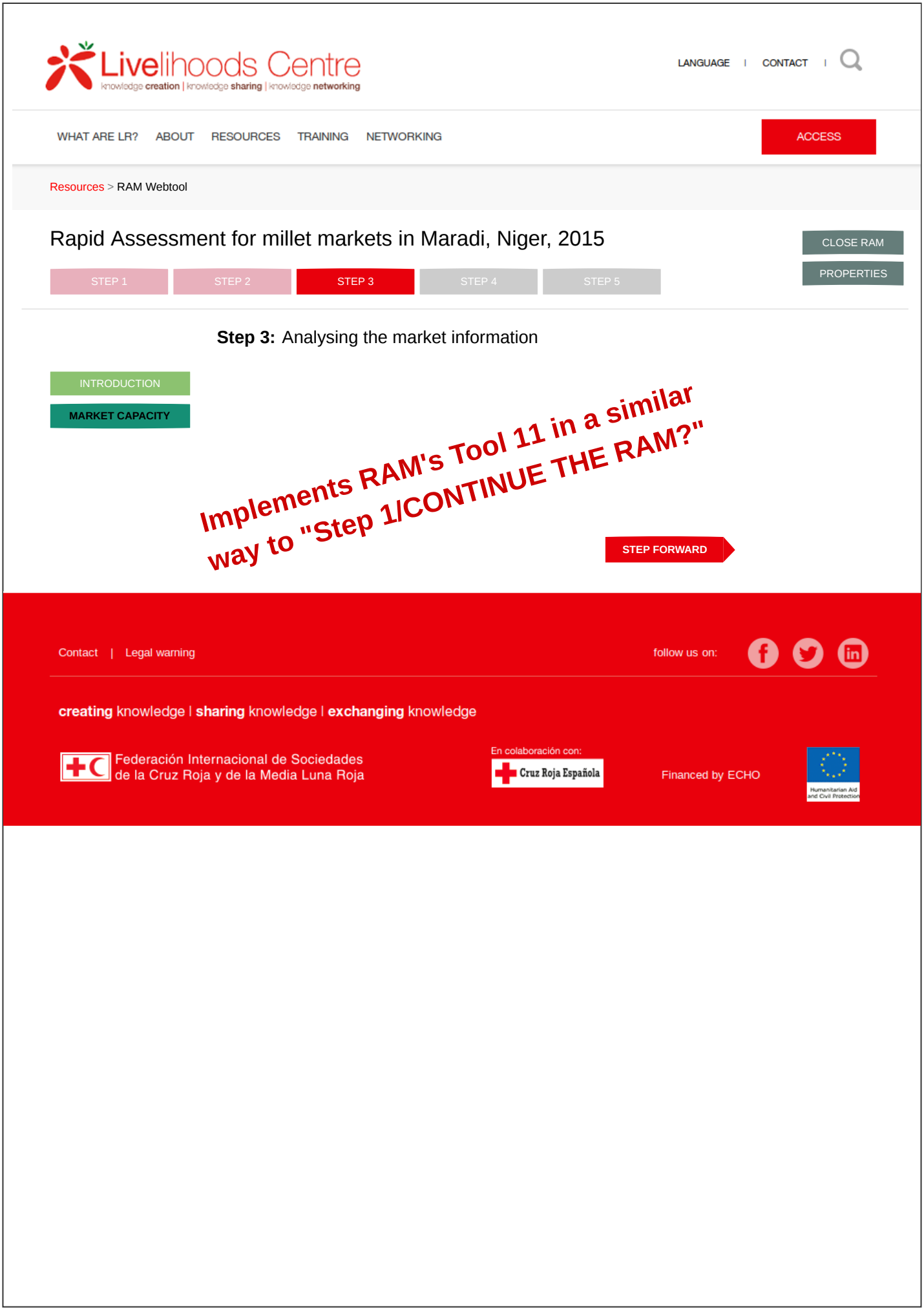
The aim of Step 3 is to determine whether each marketplace has the capacity to supply sufficient quantities of the key commodities for the population using it, and to identify response option recommendations (cash / in-kind / market-based interventions) for later consideration during the response analysis stage. Step 3 provides RAM userswith a tool (Tool 11), guidance, and a conclusion tree to answer the following questions:

- Are traders in the market operating?
- Are the key commodities available in the marketplace?
- Are the traders able to supply sufficient quantities of the key commodities?
- How are the prices of the key commodities expected to change?

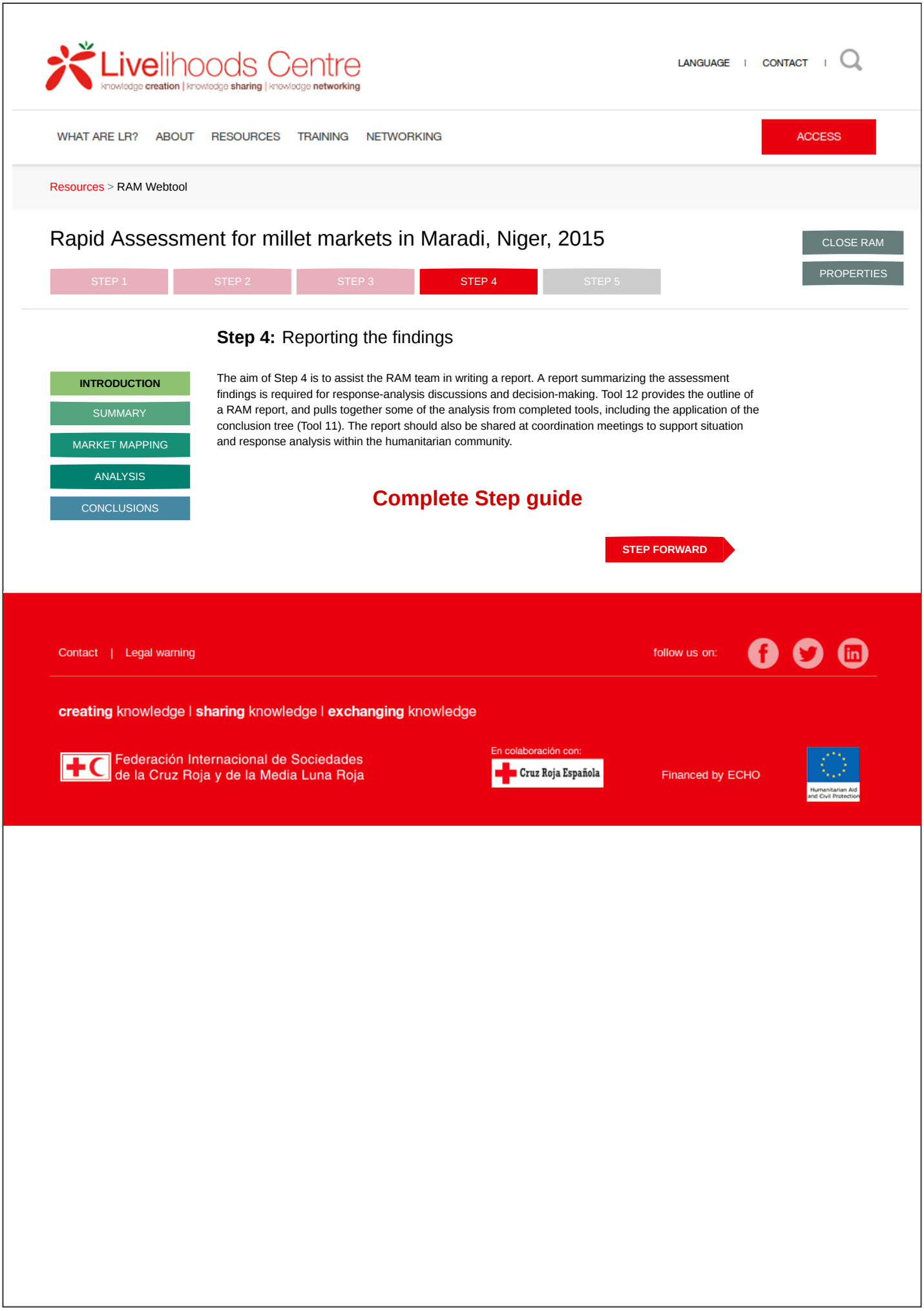
This is an important pre-requisite to Step 4, 'Reporting the findings'

Complete Step guide

Step 3.2


Livelihoods Centre
knowledge creation | knowledge sharing | knowledge networking
LANGUAGE | CONTACT |
WHAT ARE LR? ABOUT RESOURCES TRAINING NETWORKING
ACCESS
Resources > RAM Webtool
Rapid Assessment for millet markets in Maradi, Niger, 2015
CLOSE RAM
PROPERTIES
STEP 1 STEP 2 STEP 3 STEP 4 STEP 5
Step 3: Analysing the market information
INTRODUCTION
MARKET CAPACITY
Implements RAM's Tool 11 in a similar way to "Step 1/CONTINUE THE RAM?"
STEP FORWARD
Contact | Legal warning
follow us on:
creating knowledge | sharing knowledge | exchanging knowledge
Federación Internacional de Sociedades de la Cruz Roja y de la Media Luna Roja
En colaboración con:
Cruz Roja Española
Financed by ECHO
Humanitarian Aid and Civil Protection

Step 4


Livelihoods Centre
knowledge creation | knowledge sharing | knowledge networking
LANGUAGE | CONTACT |
WHAT ARE LR? ABOUT RESOURCES TRAINING NETWORKING
ACCESS
Resources > RAM Webtool
Rapid Assessment for millet markets in Maradi, Niger, 2015
CLOSE RAM
PROPERTIES
STEP 1 STEP 2 STEP 3 STEP 4 STEP 5
Step 4: Reporting the findings
INTRODUCTION
SUMMARY
MARKET MAPPING
ANALYSIS
CONCLUSIONS
The aim of Step 4 is to assist the RAM team in writing a report. A report summarizing the assessment findings is required for response-analysis discussions and decision-making. Tool 12 provides the outline of a RAM report, and pulls together some of the analysis from completed tools, including the application of the conclusion tree (Tool 11). The report should also be shared at coordination meetings to support situation and response analysis within the humanitarian community.
Complete Step guide
STEP FORWARD
Contact | Legal warning
follow us on:
creating knowledge | sharing knowledge | exchanging knowledge
Federación Internacional de Sociedades de la Cruz Roja y de la Media Luna Roja
En colaboración con:
Cruz Roja Española
Financed by ECHO
Humanitarian Aid and Civil Protection

Step 5

Rapid Assessment for millet markets in Maradi, Niger, 2015

Step 5: Monitoring the evolution of the markets

Step 5 of the RAM is dedicated to the monitoring of key markets. Markets are dynamic in nature and their evolution is difficult to predict, particularly following a disaster, as both traders and customers adapt to the altered situation. To maintain an up-to-date understanding of the market context it is necessary to monitor the key commodities and market places over time.
There are a lot of market-related aspects that can be monitored. The most common ones are prices, commodity availability, and quality. Specific aspects to monitor will depend on the context and the relief intervention implemented. This step provides RAM users with guidance and tools (Tool 13, 14 and 15) to answer the following questions:

- How do prices of the key commodities develop?
- Are people able to buy the key commodities they need in the marketplaces?

Complete Step guide

Step 5.3

Rapid Assessment for millet markets in Maradi, Niger, 2015

Step 5: Monitoring the evolution of the markets

INTRODUCTION
DESIGN MONITORING
SEC. DATA COLLECTION
RETAIL PRICES
WHOLESALE PRICES
ANALYSING DATA

Prices can be monitored using secondary or primary data. Using secondary data helps to save time and resources. Secondary data should be used whenever feasible.

Chambre Regionale d'Agriculture

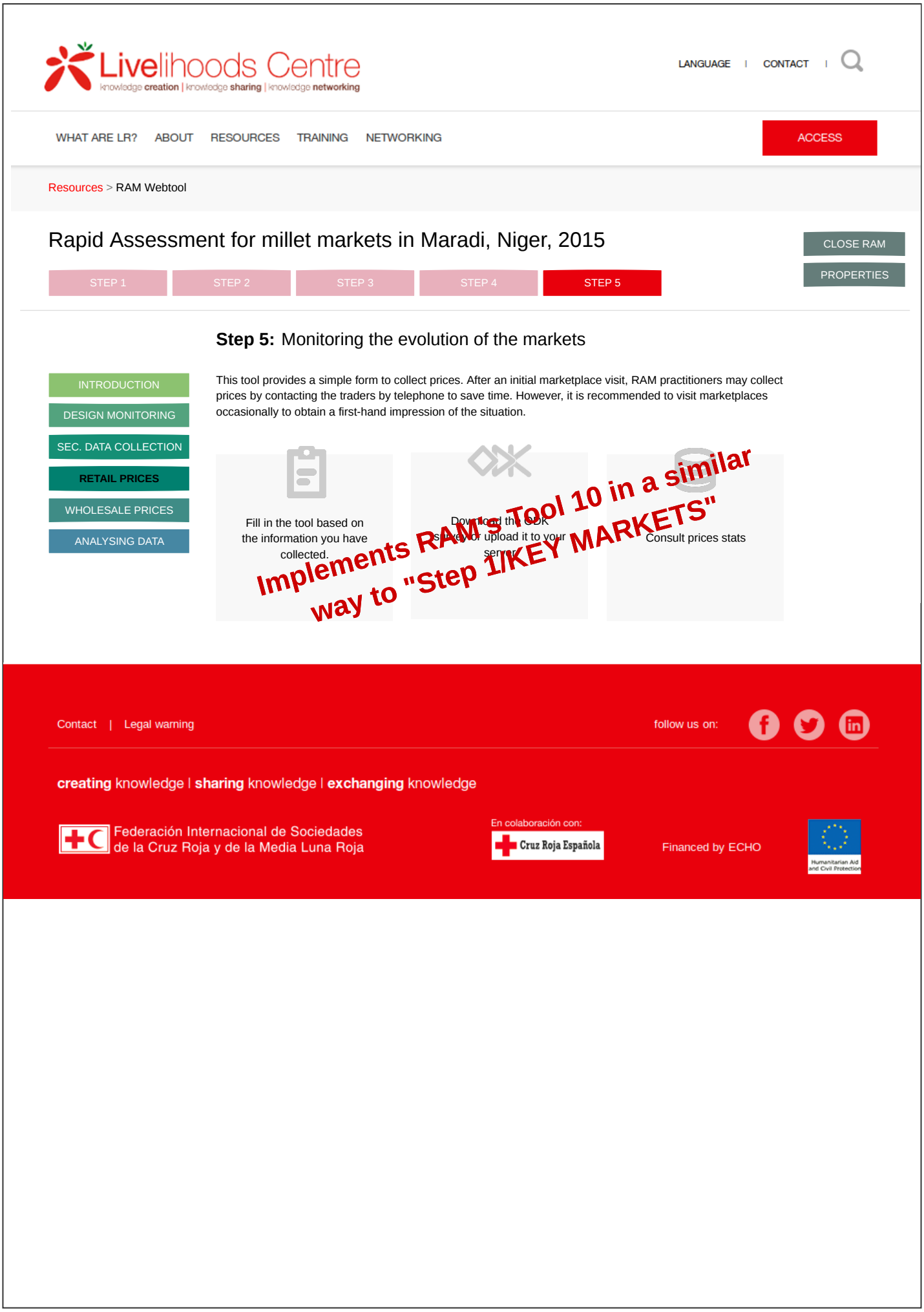
Marketplace	Commodity	Frequency	Type
Guidan Roumdji, Maradi Region	*Millet*	*Monthly*	*Average*
Guidan Roumdji, Maradi Region	*Sorgho*	*Monthly*	*Average*
Madarounfa, Maradi Region	*Millet*	*Weekly*	*One-time*
Madarounfa, Maradi Region	*Sorgho*	*Weekly*	*One-time*

FEWS.NET

FAO

Add new Secondary Source

Step 5.4


Livelihoods Centre
knowledge creation | knowledge sharing | knowledge networking
LANGUAGE | CONTACT |
WHAT ARE LR? ABOUT RESOURCES TRAINING NETWORKING
ACCESS
Resources > RAM Webtool
Rapid Assessment for millet markets in Maradi, Niger, 2015
CLOSE RAM
PROPERTIES
STEP 1 STEP 2 STEP 3 STEP 4 STEP 5
Step 5: Monitoring the evolution of the markets
INTRODUCTION
DESIGN MONITORING
SEC. DATA COLLECTION
RETAIL PRICES
WHOLESALE PRICES
ANALYSING DATA
This tool provides a simple form to collect prices. After an initial marketplace visit, RAM practitioners may collect prices by contacting the traders by telephone to save time. However, it is recommended to visit marketplaces occasionally to obtain a first-hand impression of the situation.
Fill in the tool based on the information you have collected.
Download the ODK software or upload it to your server
Consult prices stats
Implements RAM's Tool 10 in a similar way to "Step 1/KEY MARKETS"
Contact | Legal warning
follow us on:
creating knowledge | sharing knowledge | exchanging knowledge
Federación Internacional de Sociedades de la Cruz Roja y de la Media Luna Roja
En colaboración con:
Cruz Roja Española
Financed by ECHO
Humanitarian Aid and Civil Protection

Step 5.5

Livelihoods Centre
knowledge creation | knowledge sharing | knowledge networking

LANGUAGE | CONTACT |

WHAT ARE LR? ABOUT RESOURCES TRAINING NETWORKING ACCESS

Resources > RAM Webtool

Rapid Assessment for millet markets in Maradi, Niger, 2015

CLOSE RAM
PROPERTIES

STEP 1 STEP 2 STEP 3 STEP 4 STEP 5

Step 5: Monitoring the evolution of the markets

INTRODUCTION
DESIGN MONITORING
SEC. DATA COLLECTION
RETAIL PRICES
WHOLESALE PRICES
ANALYSING DATA

This tool provides a simple form to collect prices. After an initial marketplace visit, RAM practitioners may collect prices by contacting the traders by telephone to save time. However, it is recommended to visit marketplaces occasionally to obtain a first-hand impression of the situation.

Fill in the tool based on the information you have collected.

Download the ODK software or upload it to your server.

Consult prices stats

Implements RAM's Tool 10 in a similar way to "Step 1/KEY MARKETS"

Contact | Legal warning

follow us on:

creating knowledge | **sharing** knowledge | **exchanging** knowledge

Federación Internacional de Sociedades de la Cruz Roja y de la Media Luna Roja

En colaboración con: Cruz Roja Española

Financed by ECHO

C Screenshots of the first phase of the RAM web app

C.1 Tool Implementation Portlet

Figure 33: Tool Instance selection

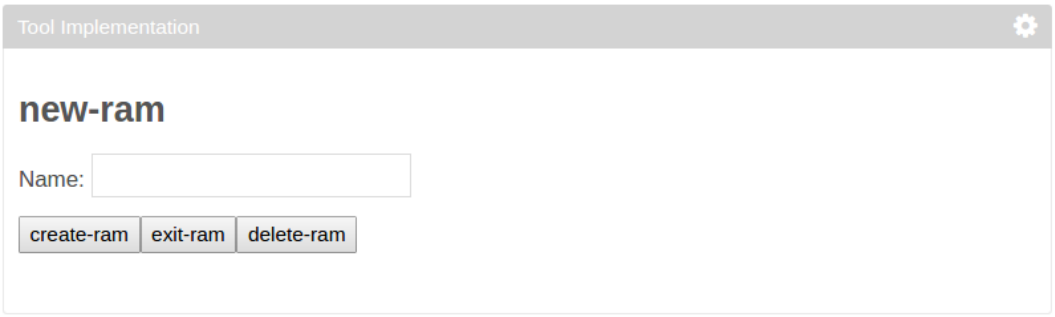


Figure 34: Tool Instance creation

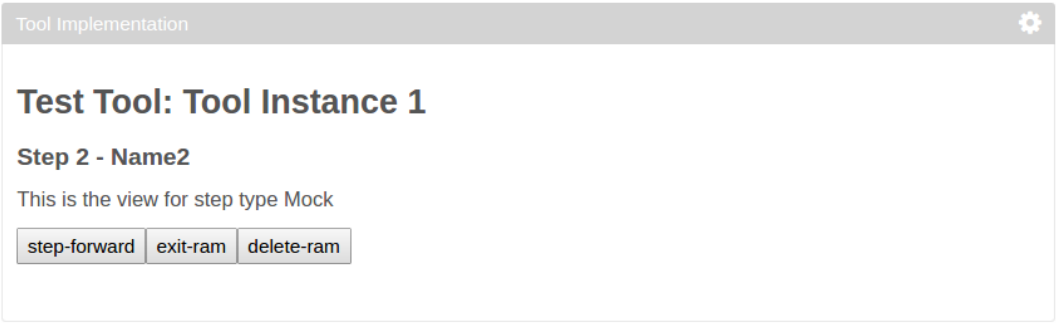


Figure 35: Tool Instance navigation

C.2 Tool Definition Portlet

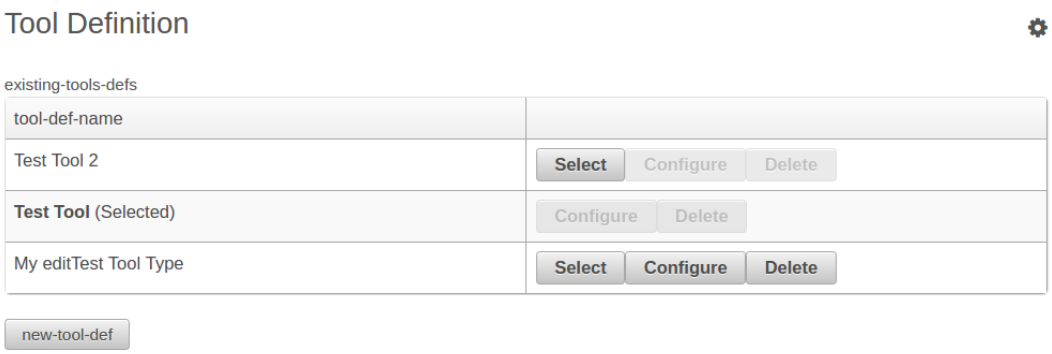


Figure 36: Tool Definition selection

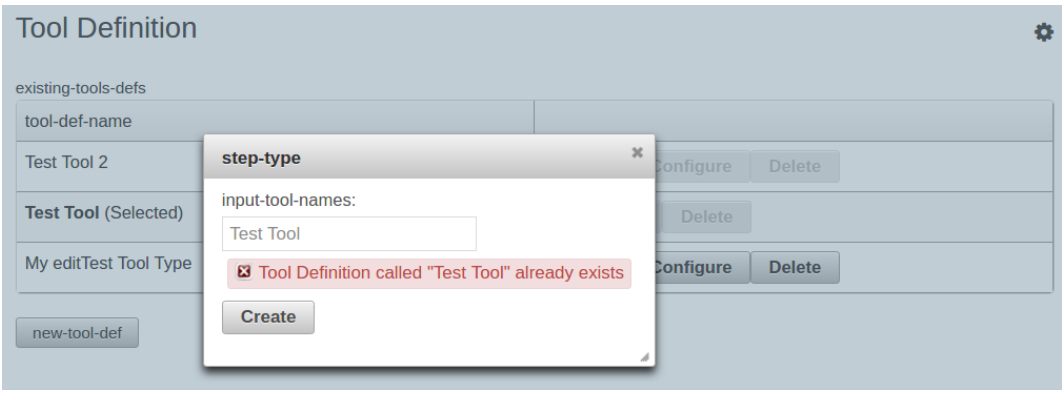


Figure 37: Tool Definition creation with name validation

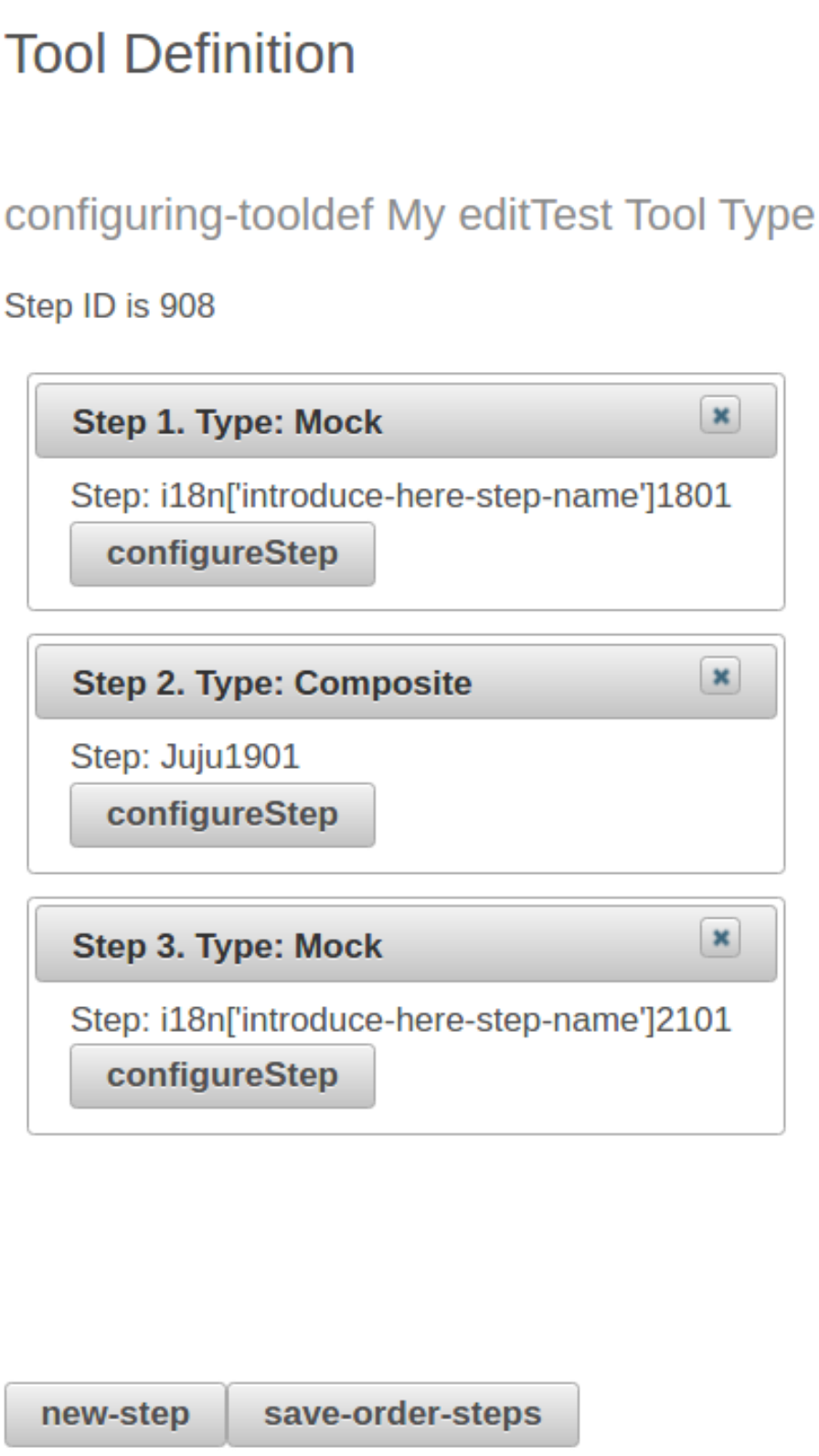


Figure 38: Tool definition edition

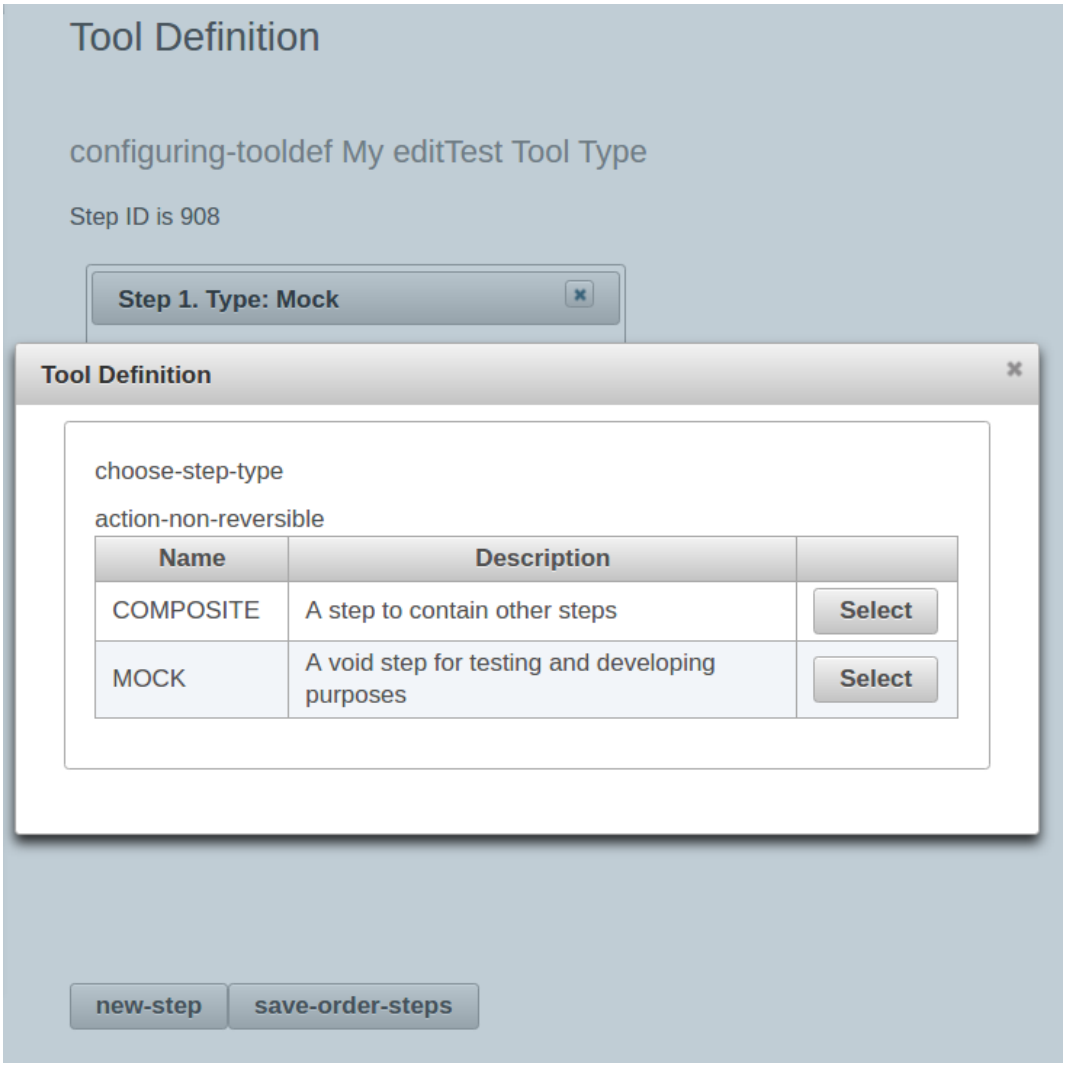


Figure 39: Step creation dialog

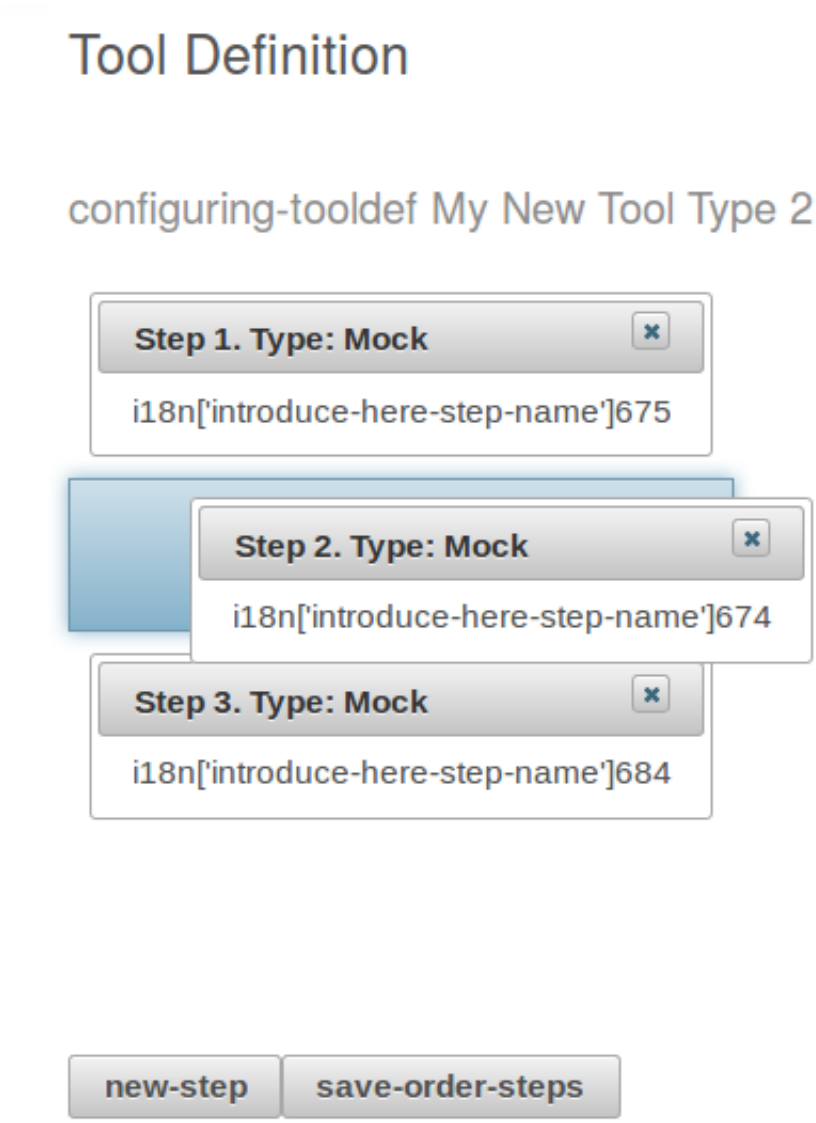


Figure 40: Reordering steps

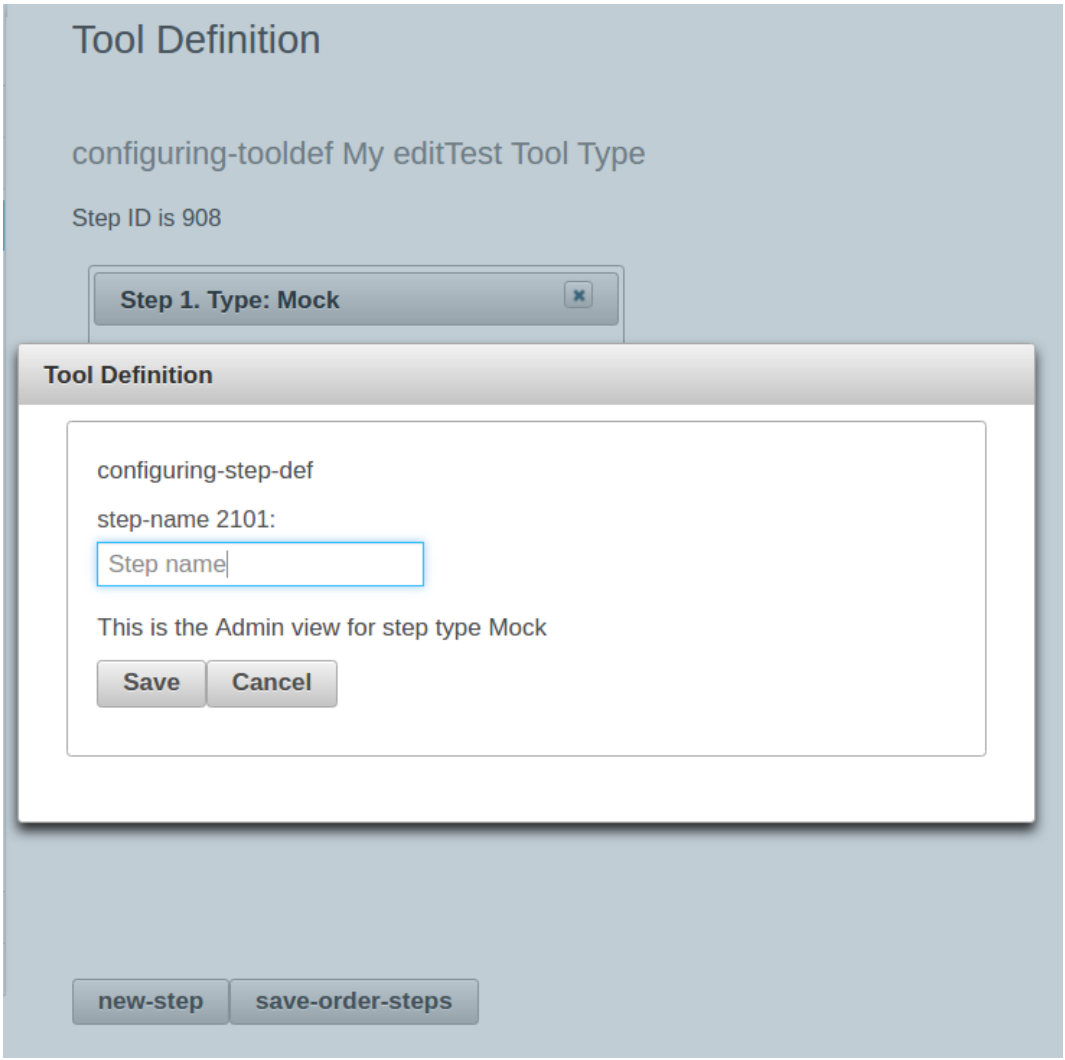


Figure 41: Mock step configuration

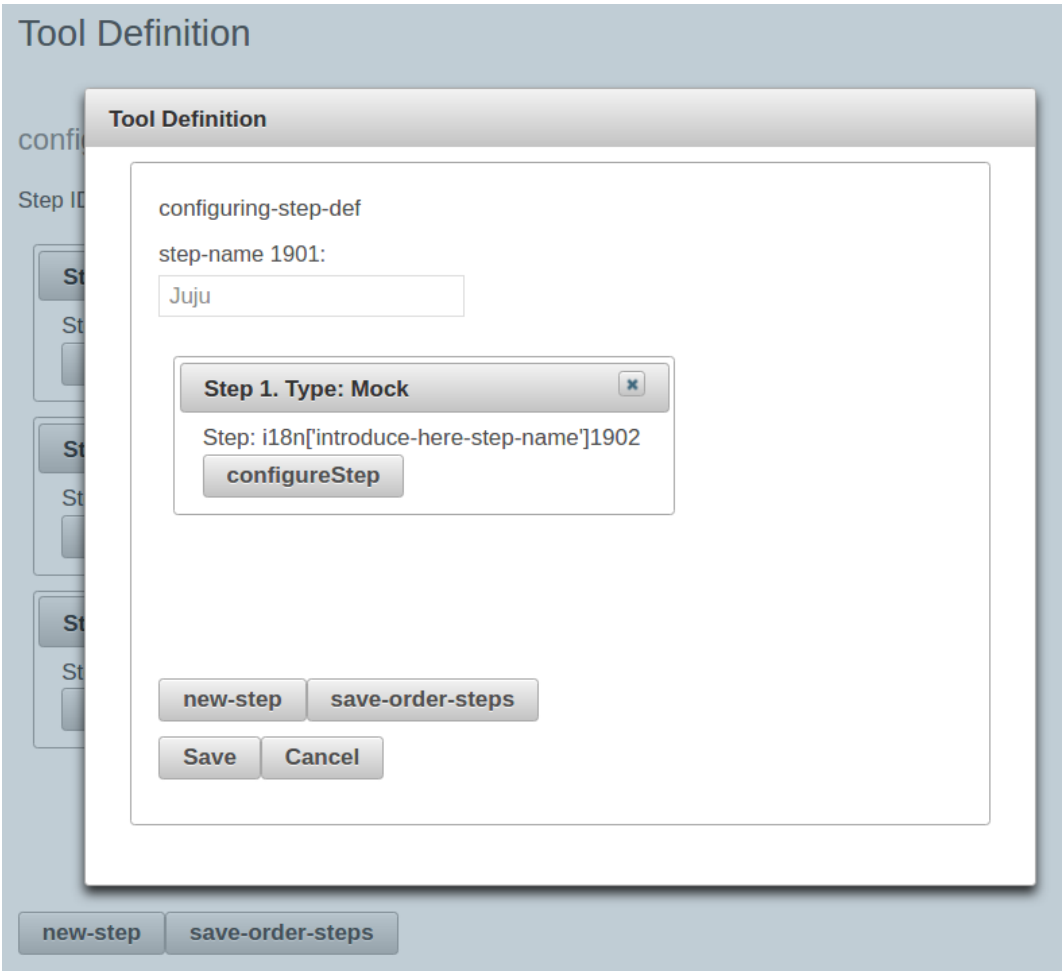


Figure 42: Nested composite step configuration