SOCIETIES
FP7-ICT
Contract no.: 257493

SOCIETIES

Deliverable D7.4

Initial Integration and Test Plan for Scenario Specific Services

Editor:	PTIN
Deliverable nature:	R=Report
Dissemination level: (Confidentiality)	PU
Contractual delivery date:	M19 (Nov 2011)
Actual delivery date:	M19 (Apr 2012)
Suggested readers:	WPL, Developers, Integrators of the Societies Platform
Version:	1.0
Total number of pages:	18
Keywords:	Integration, Testing, Development, Versioning, Services

Abstract

This document gives an initial guideline of the integration and test process for scenario-specific, third-party services developed for the Societies Platform. These services leverage on the Societies architecture and design, making use of its capabilities. They possess different components, which run on different nodes of the Societies platform. Therefore the integration and test process shall take into account at least two dimensions: the rich-client part of the service and the mobile-client part of the service. The document describes the specific considerations taken for the integration of third-party services, proposing an integration plan.

[End of abstract]

Disclaimer

Impressum

[Full project title] Self Orchestrating Community Ambient Intelligence Spaces

[Short project title] SOCIETIES

[Number and title of work-package] : WP7: Integration and testing (Internal Validation)

[Document title] D7.4 Initial Integration and Test Plan for Scenario Specific Services

[Editors: Name, company] Sancho Rêgo (PTIN)

[Work-package leader: Name, company] Bruno Jean-Bart (TRIALOG)

[Estimation of PM spent on the Deliverable] 2

Executive Summary

This document presents the initial guideline for the integration and testing strategy for the scenario specific services that will be developed in order to demonstrate the capabilities of the Societies Platform.

We start by establishing the context of WP7 and its connections to other workpackages, then presenting the specific role of T7.2 within WP7. The purpose of this deliverable is explained, presenting the overall objective of the task and its purpose: to prepare and validate the development done in WP6 into consolidated scenarios that can be delivered to WP8, which will then use them in the user-trials.

As the specific purpose of T7.2 is to manage the integration process for third-party services, these are then considered. Their purpose is presented and their development and deployment is touched upon, as considerations are drawn on the implications they have on the integration and testing plan. A list of services is also presented, divided by the scenario to which they refer to.

The following chapter defines the overall integration process; it will overall follow the guidelines set forth for development in SOCIETIES by WP7. The primary strategy is presented and the tools for testing, versioning and source-code management are referred.

The plan for integration is then presented, basing itself on monthly releases and goals.

The conclusion of the document provides an overview of the next steps of the integration and test process.

List of authors

Company	Author
PTIN	Sancho César Rego
HWU	Sarah Gallacher
Soluta.Net	Guido Spadotto
INTEL	David McKitterick
TRIALOG	Bruno Jean-Bart

Table of Contents

1 Introduction

Workpackage 7 focuses on the integration and testing of the development work done in WP4, 5 and 6. This way, the work done can be validated internally, by the development team, before being presented to the end-users during the trials prepared by WP8. While WP4 and WP5 deal with the SOCIETIES platform itself, and are thus under the umbrella of Task 7.1, WP6 deals with the third-party, scenario-specific services that will be used to demonstrate the capabilities of SOCIETIES. It is the responsibility of Task 7.2 to coordinate the integration and testing of these services. Thus, its objectives are:

- To provide an integration plan for the third-party services developed in WP6.
- To manage the integration and testing process for each individual service developed.
- To manage the integration and testing process of the various services with the actual scenarios.

Task 7.2 will follow the lead and guidelines elaborated by Task 7.1 for development and integration in SOCIETIES and apply them to the specific necessities of the third-party services. The same methods and tools will be used, as applicable, in order to provide a coherent, consistent integration process.

The current document describes the initial integration plan for the third-party services. An analysis of the status and specific development structure of third-party services is necessary, taking cues from T3.2 and from T6.1, to define the integration process to be used. This will cover the objectives of the task, from testing each individual service and its integration with the SOCIETIES platform to the integration of the various scenario-specific services into a complete scenario.

Collaborating with the various development tasks of WP6, an integration plan is then presented for the development of the specific services for the various scenarios.

A glossary of the main terms used in the project is available at http://www.ict-societies.eu/glossary/

2 Third Party Services

This section aims at presenting the basic characteristics of third-party services, as understood by SOCIETIES, and their development process. This will permit us to determine the necessary phases to undertake in the integration and testing process and start defining the integration plan.

2.1 What is a third-party service?

D3.1[1][11] provides a definition on third-party services, as understood by SOCIETIES.

"*In SOCIETIES by third party services we mean here an extension mechanism that can be used in conjunction with SOCIETIES software or hardware but that is not associated with either the manufacturer or the user. By manufacturer we mean the team that develops and maintains the SOCIETIES platform.*"

Understandably, the scenario-specific third-party services developed within the scope of WP6 technically do not fulfil this definition, as they are developed by the partners that developed the core SOCIETIES platform. This is a minor point, however, as their development will follow the same guidelines established for external developers. As stated, the objective of these services is to demonstrate the potential of SOCIETIES by leveraging the capabilities of the platform, as developed in WP4 and WP5, and extend them in order to offer new, interesting functionality that is of value to the end-user.

They will present a Graphical User Interface (GUI) to the user, which can be seen in either a rich-client node or a light-client node, and through which the capabilities of the services, and by extension the SOCIETIES platform, can be invoked.

Third-party services may also interact with services external to the SOCIETIES platform, through means independent of the SOCIETIES platform; for example, a calendar service could make use of an external calendar provider like Google Calendar. These interactions are considered out of scope of the integration process defined in this document; more accurately, the integration of a third-party service with an external service is considered part of the internal development of the service and part of a service's "unit tests".

2.2 Development of Third-Party Services

WP6 will develop various services, which can be divided into three main categories according to the type of scenario for which they are intended: enterprise, student and disaster. These correspond to tasks T6.2 through T6.4; from an integration point of view, each task will be treated equally and follow the same integration process. The importance of the categories lies in their correspondence to a specific scenario and user-trial; this will play a part in the final stage of the integration process. As stated in the previous section, one of the objectives of T7.2 is to ensure that the individual services are integrated with each other in order to form a coherent, consolidated scenario. A full list of the planned services for the first-trials can be found in D8.2, but is reproduced here, along with the information relevant to T7.2

Enterprise Services:

Service name	Devices	Deployment Details
Conference Registration	• Rich or Light Client	• Backend implementation on Virgo server in cloud node. • Web-App front-end on user-device
Collaboration Tools	• Android device	• Backend implementation running on the Virgo cloud node. • Front-end proxy & GUI running on light-client
Networking Zone	• Rich or Light Client	• Backend implementation running on Virgo server • Front end proxy & GUI running on consumer client devices
Shared calendar/Personalised	• Rich or Light Client	• Back-end implementation on • Interaction with external calendar provider.

Agenda		• Front-end proxy & GUI on user-device
Context-Aware Wall	• Rich or Light Client	• Backend implementation running on server • Front end proxy & GUI running on consumer client devices

Student Services:

Service name	Devices	Deployment Details
Master Service	• Pressure mats • Plasma screens • Xbox Kinect • Smart phone	• Backend implementation running on server • Front end proxy & GUI running on consumer client devices
Collaborative Quiz	• Pressure mats • Plasma screens • Directional Speakers • Xbox Kinect	• Backend implementation running on server • Front end proxy & GUI running on consumer client devices
Personalised Display	• Plasma screens • Xbox Kinect • Directional speakers	• Backend implementation running on server • Front end proxy & GUI running on consumer client devices
Task Posting Service	• Plasma screens • Smart phones (light client)	• Backend implementation running on server • Front end proxy & GUI running on consumer client devices.
Co-browsing Service	• Plasma screen (with an intelligent device, such as a computer associated) • Tablet	• Backend on server hosted by NEC • Front end & GUI as web pages suited to mobile clients
NearMe	• Android Smart Phones • Cloud Nodes • Web Clients	• Backend implementation running at the server side • Front end proxy & GUI running on Smart Phone with Android

Disaster Management Services:

Service name	Devices	Deployment Details
IWantToHelp	• Rich or Light client	• Webservice running in the cloud, proxy in the CSS
YouRNotAlone	• Rich or Light client	• Webservice running in the cloud, proxy in the CSS
iDisaster	• Light-Client (Android)	• Android App • Back-end component in virgo container. • Plug-in proxies to external webservices.

"DDC" (Disaster Data Collector)	• GPS • Arduino-Devices • Chest-Belt • Android Phones • Compass • Cameras	• Webservice running in the cloud, proxy in the CSS. • Android App that may be triggered from iDisaster
AnalyzeThis	• Rich or Light client • Camera	• Webservice running in the cloud, proxy in the CSS. • Android App that may be triggered from iDisaster

It should be noted, however, that it is possible for services developed for one scenario to be later included in another scenario, should it be decided by WP8 that it would add value to the scenario. Furthermore, there might be functionalities pertaining to the services in one scenario that can be reused in another scenario. From a T7.2 perspective, this will not affect the development and testing of individual services; however, for each service added to a scenario, the full integration testing must be repeated to ensure the scenario is validated and tested.

2.3 Deployment of Third-Party Services

The services developed within the scope of WP6 have multiple deployment configurations, as may be seen from the previous section. A common configuration is to have core functionality in a back-end server component and then a front-end component that deals with the user interaction.

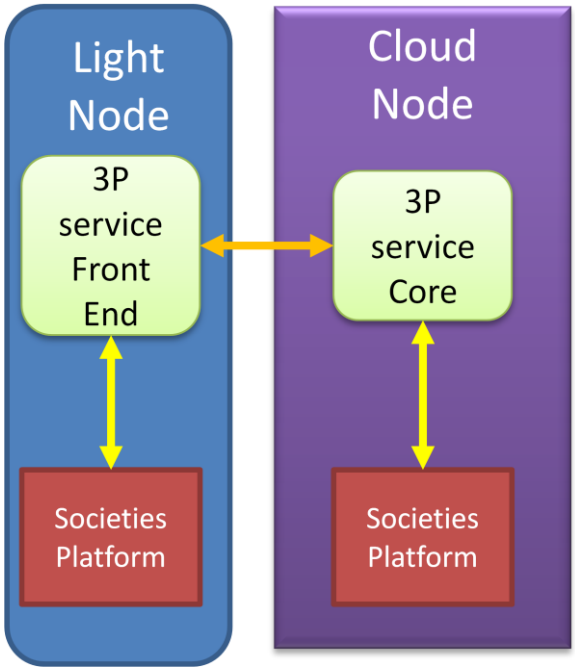

Figure 1 *– The back-end server on cloud node / front-end on user device deployment*

The back-end server may be an external service housed outside of SOCIETIES that is interacted with as a web-servicer, or it may be an actual OSGI component that is deployed on a Virgo container in a cloud-node. The front-end might be a light-client application on Android, or even a web-app offered in the Virgo container of the rich-client.

Furthermore, some services might also interact with external components or services for added functionality. It is important to note that not all services will have these components, as each individual service will have its own requirements and characteristics. An example component diagram is presented:

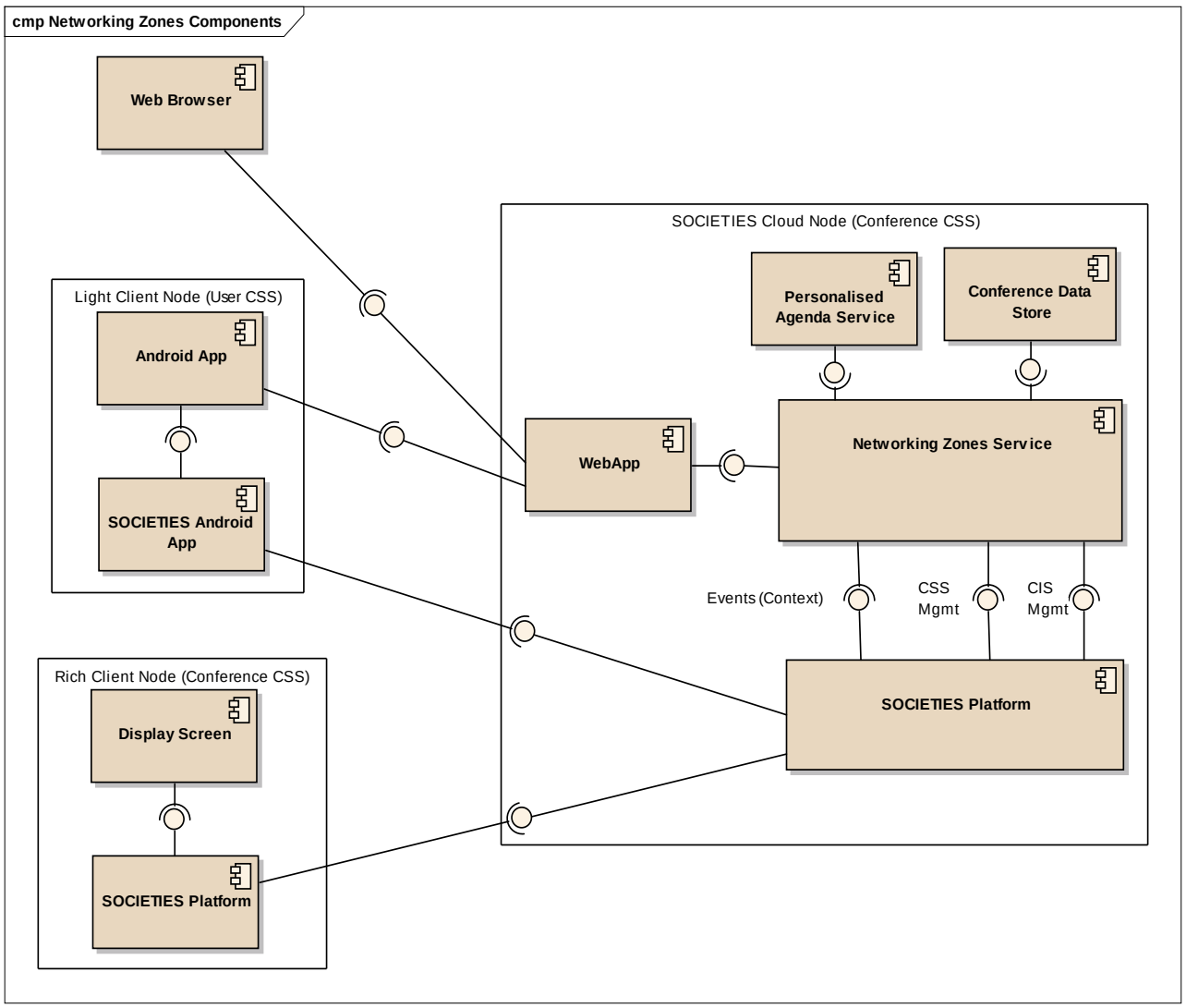

Figure 2 *– Example service (Networking Zone)*

Following the example from T7.1, and also in accordance to the development plans of WP6, these various parts that compose a service (back-end server, rich-client, android client...) will be treated as individual components that require integration and testing. Each component of a service will need to be tested individually and then integrated with the core SOCIETIES components that it interacts with; finally, the various components will need to be integrated with each other in order for a given service to be considered tested and integrated.

3 Integration Process

This chapter presents the core integration and testing process for the scenario-specific third-party services developed in WP6. The general procedure will be presented, ranging from tools to be used, different phases of testing, roles and responsibilities and the structure for development. Much like T7.1 sets the overall guidelines for development in SOCIETIES, focusing on the core platform, it is intended that T7.2 do the same for the specific case of third-party service development.

Integration in SOCIETIES is understood as a continuous, iterative process that helps drive and guide development. T7.1 defines the integration and testing strategy for the SOCIETIES project; T7.2 will not deviate from the established tools and procedures, adapting only as necessary.

3.1 Integration Levels

In D7.1[2], a classification of the software modules to be tested was presented, as related to the core Societies platform. WP4 and WP5 modules would be considered "components", and be fully tested internally before being delivered to WP7. WP7 would then perform the integration of these components, testing their interaction together at the platform level. Test-Cases would be defined for this purpose, evaluating the needed functionality of each component in order to fulfil the requirements of the platform.

This approach must be adapted to T7.2, given the particulars of third-party services. By their intent and nature, they are more separate and independent entities unto themselves, rather than parts of a larger whole, such as a platform. This reduces the level of integration needed between services, though it should be kept in mind that some services might need to work together in the proper context.. A clear example would be various enterprise-related services making use of the Personalised Agenda, or the sharing of data between services, making use of the same CIS Furthermore, the services related to each category must be consolidated into a scenario to be delivered to WP8.

Another point to note is that services might have multiple components that interact; for example, a back-end server that provides the main functionality and then an android component for the light client. This must be taken into account in the integration efforts, as these components will often be deployed in different nodes and make use of the underlying platform to communicate.

Keeping these factors in mind, the following approach will be taken in T7.2 as regards various integration and testing levels:

Component – this level corresponds to a specific part of a developed service; for example, the light-client Android app that presents the front-end to the final user. In essence, it corresponds to a software unit that might be deployed in a given node. A service might have one or more components.

It is expected that testing at this level is performed by the WP6 developer, which will supply to WP7 a functional software unit; needed Societies platform features should be mocked up with the appropriate testing framework. Connections between a component and an external provider, for example a map service, however, are considered internal to the component and outside the scope of WP7. These should be fully tested by the developer.

This level of testing does not require a fully-functional underlying platform, as mock-ups will be used to simulate the needed functionalities.

Service – this level corresponds to a complete, individual service, encompassing any components that it might have and their interaction. An example is the communication between the light-client component of a service and the back-end that runs on the cloud-node. The expected output is a functional service whose various components correctly interact and communicate.

It is expected that different components make use of the underlying communications framework supplied by the core platform; should this be the case, then for this integration testing to occur the core functionality of the Societies platform (as supplied by WP4) must be present. However, any functionalities of the Societies platform not pertaining to the direct communication between different services (such as WP5 user-experience features) may be mocked up.

Platform– this level corresponds to the integration between a third-party service and the underlying features of the Societies platform. The output is a complete, functional service that may be deployed to any instance of the Societies platform.

A service is considered fully tested at a platform level if every component can be deployed to the appropriate node and all of its interactions with the WP4 and WP5 components are completely tested.

Testing at this level will follow the principles of platform testing defined in D7.1[2]. Test suites will be defined for the various services in order to validate the interactions between the service and the WP4 and WP5 platform functionalities. Given the presence of GUIs, there will also be script-based testing based on manual user-interaction tests.

Scenario – this level corresponds to the consolidation of a given set of services into a complete scenario for delivery to WP8. The services that compose a given scenario must all be deployed and not interfere or cause conflicts with each other. Furthermore, it must be possible to run the entire scenario, as intended for the user-trials, from start to finish.

Interaction between two or more services, for example if a third-party service makes use of the data of another third-party service, is also included in these tests. This may occur either by direct interaction, for example using API calls, or indirect interaction (shared data models / database).

Testing at this level will be primarily based on manual user-interaction, following scripts that represent the actions a user might take.

3.2 Testing Methodology

T7.2 will adopt the strategy of creating a series of test suites for each service to be tested. These test-suites will cover the various facets of integration needed for the services, from the communication between different components of a given service, interaction between services and WP4/WP5 platform functionalities and also the interaction between different services to form a consolidated scenario.

It is desirable that these test-suites be automated as much as possible; for cases where this is applicable, WP7 will implement the necessary code and execute the necessary tests. However, it is expected that many of the necessary tests will not be easily automated, in particular when dealing with scenario-level tests. The reason for this is that the objects under test are third-party services oriented for the end-user; as such, they may possess GUI interactions or require a set of complex behaviours that must be input by human-interaction.

For test-suites where automation is not applicable, then a script will be made that a tester must follow to validate the desired behaviour.

Regardless of whether the tests are automated or manual, they will follow the same structure and be defined in a similar manner. The concept of functional chains, used in T7.1 for the gradual and progressive integration of architectural components, fully applies to third-party services and may be applied to all levels of integration.

For each service a set of behaviours will be described, ranging from the initial conditions, the actions to be taken and the desired effect. These will then be decomposed to identify the involved functional chains and interactions between software components. In turn, this will allow WP7 to more concretely define the necessary tests to form the desired test-suites. The process is iterative and progressive, ranging from more

simple functional chains to more complex ones as more features and interactions between service and platform are tested.

The same process applies directly to scenario-level testing and the functional chain approach is still viable; in this case, the interactions and behaviours to test are between services, not services and specific core platform components. Indeed, the complete, consolidated scenario could be considered the most complex functional chain and thus may be translated to the final test for a given scenario.

The following table, taken directly from D7.1, presents the basic structure of a generic test as will be used to define the test-suites in T7.2, It can be used for all levels of integration, and both for automated and manual tests. The System Under Test (SUT) may be a single service or a collection of services.

Test ID	Test Name	Test Purpose
The test ID shall provide a way to identify where the test case is placed in the test suite structure. e.g. Func_Famility_Number	A generic expression to identify the function to be tested. e.g. Create a new User.	This field shall provide a summary of the objectives of the test cases.
Preamble	This part is used to set the SUT into the mandatory state before run the test body. E.g. Populate the SUT with the right data.	
Test Body	This part contains the script of the test itself. The sequence of actions to be performed to the SUT, the response expected from the SUT, the comparison test between the expected responses and the actual responses providing the verdict: PASS, FAIL, INCONCLUSIVE.	
Postamble	This part of the test case is necessary to set the SUT into an idle state so that the next test case can run.	

3.3 Integration Tools & Management

T7.2 will not deviate from T7.1 in terms of the overall testing tools, as defined in D7.1[2]. A brief list of the most important tools is presented:

3.3.1 Code testing

The tool of choice adopted in SOCIETIES for creating test suites, both for component testing but also for automating platform level testing, is JUnit[4]. D7.1, chapter 3, delves into this topic at length. It is expected that individual service developers fully test their services internally making use of JUnit, if applicable. Furthermore, testing suites based on JUnit will be developed by WP7 for all applicable services (i.e. those based on Java development, OSGI services for deployment in a Virgo environment).

In addition to JUnit, mocking frameworks like Mockito[5] are of particular importance for service testing, as they will allow developers to simulate the behaviour of the SOCIETIES platform; this will permit them both development and unit testing without requiring the full functional core platform. Again we refer to D7.1 for a more extensive explanation and usage guide.

For testing Android[9] components of services, D7.1 contains an adequate explanation of the procedure in section 3.1.2; this focus not only the testing of actual Android code, using JUnit and Eclipse[10], but also the HTML5 and PhoneGap testing procedures.

3.3.2 Bug-Reporting and Management

Redmine[6] will be used for both bug-reporting and overall management of the integration process. Usage of this tool was presented in D7.2[3] by Task 7.3.

Appropriate tickets will be issued not when a bug is found, but also to coordinate the integration plan for each individual service, according to each release.

3.3.3 Repository and Source-Code Management

Much like the case of the core SOCIETIES platform, T7.2 will use a git[7] for version control and source code management. Third-party service development will have its own repository in github[8], with its own folder structure.

https://github.com/societies/SOCIETIES-SCE-Services

The folder structure will be kept simple. There will be three main folders, corresponding to the scenario categories: Enterprise, Student and Disaster. Inside each of these folders, each third-party service will have its own folder.

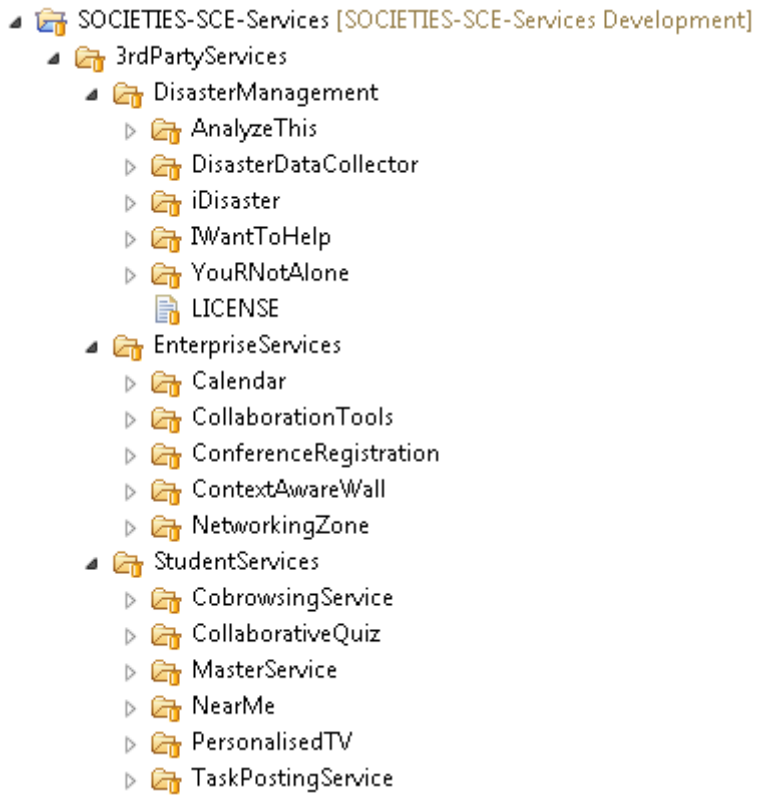

Figure 3 *– Folder structure in Github*

The file and project structure inside each folder corresponding to a given service will not be rigidly defined; rather, given the independent nature of development of each third-party service, developers will be mostly free to organise their individual projects. However, a few guidelines are expected to be followed:

- Individual components of a given service (for example, the back-end server and android client) are considered separate projects, and as such should be developed as such.
- For both projects meant for the rich-client and for Android projects, the overall development recommendations established for SOCIETIES are expected to be maintained. By this we mean the guidelines regarding the naming of classes, packages and interfaces.
- If the service is meant to be used and integrated with another folder, it should contain an api project.
- Service developers should modify only contents of their own service folders.

This covers the folder structure for the development of the services. Test-Case Suites will also be implemented and added to this github repository. The folder structure will mirror that of the "3rdPartyServices" folder.

4 Initial Integration Plan

This chapter presents the initial development and integration plan for the third-party, scenario specific services developed in WP6. It intends to support and guide the WP6 development and ensure that WP7 delivers integrated and complete scenarios for WP8 to use in the user-trials.

4.1 Planning considerations

Given the final objective of providing integrated scenarios to WP8, we must necessarily take into account two important factors into the planning. The first relates to the different trials and their dates, as this essentially defines the deadlines for the integration and testing tasks; the second relates to the integration of the SOCIETIES platform itself, as the services may not be fully tested and integrated with its features until these are ready.

4.1.1 User-Trials

The services to be integrated can be classified into to three categories which correspond to the enterprise, student and disaster management scenario trials. As the trials do not occur simultaneously, the integration and development plan should reflect this by giving adequate priority to the integration and testing of the services depending on their trial. Naturally, the earlier the trial, the higher priority testing and validating the scenario becomes and more resources will be dedicated to it.

Nevertheless, all scenarios and corresponding services will be tested and integrated independently and in parallel. This mirrors the development process of the services in WP6, in which they are divided by category into a given task. As tasks T6.2 to T6.4 run in parallel, and the integration process is continuous, by extension the testing and integration of each scenario must be parallel as well.

The first trial is intended to be the student trial, which occurs at the beginning of the school year in September. As such, this marks the point when the student scenario and its related services must be fully integrated, validated and tested for delivery to WP8. September 2012 will, therefore, be considered as the target goal for the first iteration of T7.2.

4.1.2 Core platform integration

Another aspect to take into consideration is that the services are meant to demonstrate the features of the SOCIETIES platform, and as such integrate with it. WP4 supplies the core CSS Platform functionality, on top of which services are deployed and discovered, device abstraction is done and CSS and CIS are created and maintained; third-party services need this backbone to exist and be delivered to the users. WP5 supplies the user-centric features such as learning, personalisation and context; third-party services need this to provide their interesting, innovative features to the users.

With this in mind, for the third-party services developed by WP6 to be fully integrated and tested in order to form a consolidated scenario, the underlying SOCIETIES platform and features must be present and available. Both WP4 and WP5 have their own testing and integration plan, which is developed by T7.1. T7.2 will work closely with T7.1 in this regard, setting its own integration goals to match the available features as defined and tested by T7.1.

4.2 Plan Structure

The integration plan will be similar to that of the core platform, as defined in T7.1. There will be monthly releases, tracked in Redmine. Each release will have its set of associated functionality for each developed service, and target integration goals, as well as a collection of test suites that must be passed.

The specific goals for each release will vary from service to service, but the natural pattern is that each service must be first tested internally, focusing on the features that do not depend on the SOCIETIES platform. Mock-up frameworks substitute the platform functionality in testing.

Once a platform feature is tested and validated by T7.1, the services that make use of it will be subjected to the relevant test suites. This will constitute the Platform-level testing mentioned in the integration process chapter of this document.

While each service will have its own integration plan and test suites, not all services will be in the same integration level concurrently. Therefore, the final stage of integration must necessarily be the Scenario integration level. This type of testing, which validates the interaction between different services, may begin only once the related services have terminated their Platform-level testing. As each service passes its platform-level tests, it may be added to the Scenario-level testing; this process will continue up to the delivery date of the full, consolidated scenario.

4.3 Overall Goals

The initial schedule for the testing and integration planning is be monthly, as referenced in the previous section, with the following major goals:

May 2012

- Integration plan defined for all services.
- Definition of test-suites for service-level and platform-level integration.

June 2012

- Service development and component-level testing completed.
- Implementation of test-suites and scripts for platform-level integration for all services.
- Service-level integration and Platform-level integration initiated.

July 2012

- Service-level and Platform-level Integration completed
- Test-Cases for Scenario-level Integration prepared.

August 2012

- Scenario-level Integration
- Delivery of Student Scenario to WP8
- Delivery of Enterprise Scenario to WP8
- Delivery of Disaster Management Scenario to WP8

As stated previously, the first trial to occur is the Student one and as such the corresponding scenario-integration will have absolute priority. With this in mind, should the plan not be fulfilled and unforeseen delays occur, then the final delivery of the enterprise and disaster management scenario will be postponed, as their trial-dates are later. This will be done so that focus can be put on the first user-trial, for the students.

5 Conclusions

D7.4 provides a first set of initial guidelines for integration and test of the scenario-specific third-party services developed in SOCIETIES. The model established by T7.1 will be followed in general, with the necessary adaptations for third-party service development.

The process divides the integration into various levels, as services are first integrated internally between their different components, integrated with the platform core components and then up to integration with each other to form a complete scenario. The overall plan will have monthly releases and be coordinated with the SOCIETIES core platform integration plan.

The next steps of task T7.2 are:

- To work with WP6 developers and define the concrete integration plan for each individual service.
- To establish overall scenario-wide integration goals for each of the three target scenarios.
- To define and implement test cases and scripts for both specific services and the overall scenario.

The above tasks shall be ready by June 2012, after which testing may begin in full.

The first integration and test phase will start in June 2012 and will run until September 2012, the date of the first student-trial. The result will be a fully integrated student-service scenario which WP8 can utilize.

Furthermore, the integration and testing work will continue beyond that, to ensure the delivery of the enterprise and disaster management scenarios to their respective trials.

6 References

[1] Societies D3.1: Service Model Architecture, Societies Consortium. March 2012.

[2] Societies D7.1 v2: Initial Integration and Test Plan of the SOCIETIES System, Societies Consortium. February 2012 .

[3] Societies D7.2: Integration Testbed Specification, Societies Consortium. October 2011.

[4] JUnit : http://www.junit.org/

[5] Mockito: http://code.google.com/p/mockito

[6] Redmine : https://redmine.ict-societies.eu/

[7] Git: http://git-scm.com/

[8] Github: https://github.com/

[9] Android: http://www.android.com/

[10] Eclipse: http://www.eclipse.org/

[11] Societies Glossary: http://www.ict-societies.eu/glossary/

