D5.1 – Integration Framework Specification

Deliverable ID	**D5.1**
Deliverable Title	**Integration Framework Specification**
Work Package	**WP5**
Dissemination Level	**PUBLIC**
Version	**1.0**
Date	**2018-07-27**
Status	**Final**
Lead Editor	**Sisay Chala (FRAUNHOFER)**
Main Contributors	**BQ, CNET, FRAUNHOFER, GAPES, ISMB**

Published by the GOEASY Consortium

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776261.

Document Version History

Version	Date	Author(s)	Description
0.0	2018-06-5	Chala (FIT)	First Draft with TOC
0.1	2018-06-22	Chala (FIT)	Updated Introduction, Scope, Terms and Definitions and Related Documents
0.2	2018-06-28	Chala (FIT)	added Section 2 “Continuous Integration and Testing”, Updated Section 3 “Components specification and integration testing”, Added Section 5 “Integration Plan"
0.3	11.07.2018	Chala (FIT), ISMB	Incorporated contributions of ISMB in Sections 3.1 and 3.2
0.4	12.07.2018	Chala (FIT), CNET	Incorporated contributions of CNET in Sections 3.1 and 3.2
0.5	13.07.2018	Chala (FIT), GAPES	Incorporated contributions of GAPES in Sections 3.1 and 3.2
0.6	17.07.2018	Chala (FIT), BQ	Incorporated contributions of BQ in Sections 3.1 and 3.2
0.7	17.07.2018	Chala (FIT)	Complete version ready for internal review
0.8	23.07.2018	Chala (FIT), BQ	Applied review from BQ
0.9	25.07.2018	Chala (FIT), GAPES	Applied review from GAPES
1.0	27.07.2018	Chala (FIT)	Submitted version

Table of Contents

1 Introduction

This work package, i.e., WP5, is responsible for developing the two mass-market applications used to validate the GOEASY concept, namely ApesMobility and AsthmaWatch, as well as providing integration and testing for the whole project. Within WP5, this task, i.e., T5.1, devises a comprehensive integration plan of how different components, software modules and subsystems will be integrated in agile fashion into a single platform, ready to be deployed into pilots using the two aforementioned applications. In addition to integration framework specification, the plan also includes automated testing aspects, so to ensure automatic verification of interoperability among major components prior to deploying new releases of components in the platform.

This deliverable specifies definition of the integration framework and associated tools as well as components developed during the course of this project and their integration issues. Moreover, this document supports GOEASY in defining, documenting, and implementing a software testing methodology and test management governance framework that can be adopted in order to achieve improvement in overall software quality. More precisely this document provides the initial integration specification and associated tools set up under which GOEASY can operate to deliver testing services to the various system components in order to ensure that all the requirements specifications stipulated in D2.1 are realized.

The overall software testing goal is to produce high-quality systems, using a set of managed and controlled processes, which meet the requirements and expectations.

1.1 Scope

This document elaborates the requirements described in Vision Scenarios and Use Case Definition (D2.1) and the design presented in the Initial GOEASY Platform and Pilots reference Architecture (D2.2) document. It then illustrates the required components and their interactions as well as test cases for the integration and system tests. The deliverable in this document will serves as the basis for D5.3, D5.5 and D5.7, among others as shown in Figure 1.1.

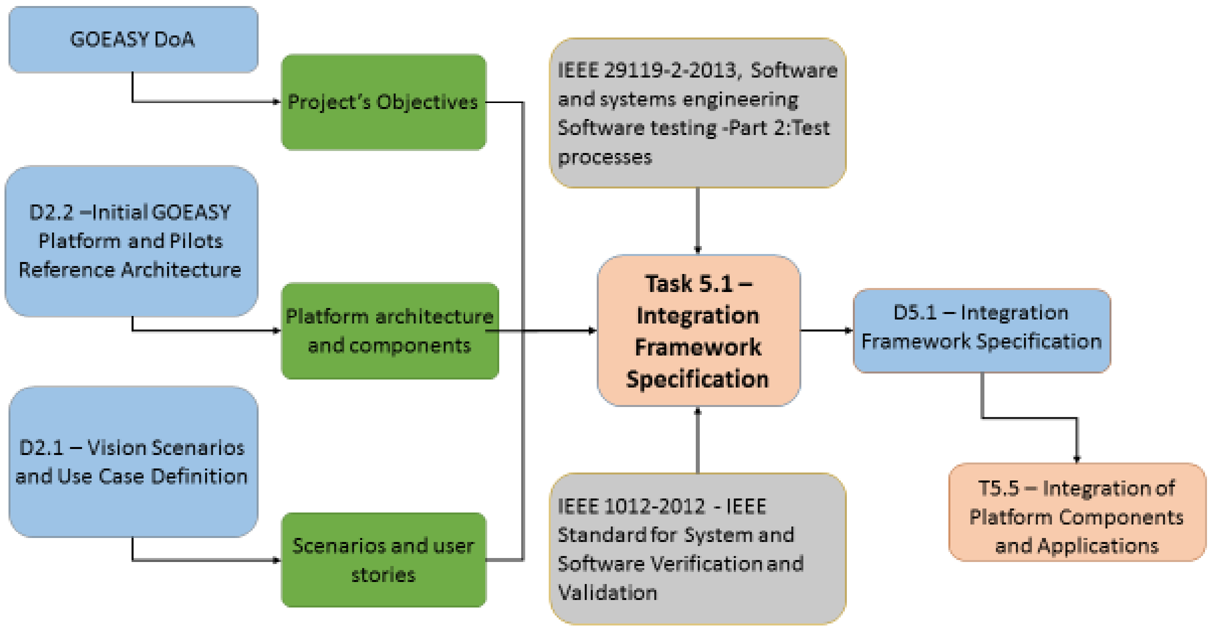
GOEASY DoA
Project's Objectives
IEEE 29119-2-2013, Software and systems engineering Software testing -Part 2:Test processes
D2.2 –Initial GOEASY Platform and Pilots Reference Architecture
Platform architecture and components
Task 5.1 – Integration Framework Specification
D5.1 – Integration Framework Specification
D2.1 – Vision Scenarios and Use Case Definition
Scenarios and user stories
IEEE 1012-2012 - IEEE Standard for System and Software Verification and Validation
T5.5 – Integration of Platform Components and Applications

Figure 1.1 Dependency Relationship of this Document

1.2 Terms and Definitions

Terms	Definitions
Unit Test	Also known as component, module or program test. The tasks in unit test is to search for defects in, and verify function of, software module programs, objects, classes, etc., that are separately testable.
Integration Test	Tests interfaces between components, interactions with different parts of a system, and interfaces between systems.
System Tests	Tests the behaviour of a whole system/product. In system testing, the test environment should correspond to the final target or production environment as much as possible.
Specification based tests	Are used to derive the test cases from the system or software requirements specifications in order to validate if the requirements are met.

1.3 Related documents

ID	Title	Reference	Version	Date
[RD.1]	D2.1 - Vision Scenarios and Use Case Definition	D2.1	1.0	2018-02-27
[RD.2]	D2.2 - Initial GOEASY Platform and Pilots reference Architecture	D2.2	2.0	2018-06-21

2 Continuous Integration and Testing

System integration and testing is a type of software testing that makes sure that tests such as the components and their integration are done before releasing the system. Software testing follows strict sets of rules and guidelines to make sure each that individual component of the software is thoroughly checked, before it is put into use. The procedure is meant to minimize errors as much as possible and ensure that the software runs to serve the intended purpose.

2.1 Software Testing Types

Software testing involves different levels, namely: unit test, integration test, system test and specification based test (also known as user acceptance test) as described below:

- Unit Test – is also known as component, module or program test. The tasks in unit test mainly focus on searching for defects in, and verifying function of, software module programs, objects, classes, etc., that are separately testable.
- Integration Test – tests interfaces between components, interactions with different parts of a system, and interfaces between systems.
- System Test – is used to test the behaviour of a whole system. In system testing, the test environment should correspond to the final target or production environment as much as possible.
- Specification based tests – are also known as user acceptance tests and are used to derive the test cases from the system or software requirements specifications in order to validate if the requirements are met.

In software testing each of these testing levels builds on the previous level, i.e., integration test builds on unit test, system test builds on integration test. Therefore, it is important that the testing is done in the correct order with the lower level tests being completed before moving on to the higher level tests. The way that integration testing works is by getting the individual modules, which have already been through the unit testing phase, and integrating each module into a group. The integration testing phase checks, when the modules are being integrated, for any problems such as errors or bugs caused due to the integration of the modules, and makes sure that they are eliminated. Integration testing does not deal with the integration of the whole system (as in system test) but deals with the integration of components and their interfaces in the system.

In the integration testing stage there are three things that are created, to ensure that the integration of the modules is successful: integration test objective and plan, test cases with test data, and acceptance conditions.

i) Integration Test Objective and Plan

An integration test plan is a collection of integration tests focusing on functionality. The production of a test plan must include the following information:

- A test objective and strategy to be used when testing the integrated modules and specifying how the tests will be conducted.
- What will be tested for example software components and/or features.
- Assignee or responsible personnel.
- Testing pass and fail conditions.
- Risk involved

The items listed above are among the most important elements that shall be included in a test plan, but as the testing progresses, more item-specific components can arise and, eventually, be included in the upcoming iterations of this document. Test plans are approved and revised (when needed) with the target user, so that they may have to include more information during the pilot test.

ii) Integration Test Cases and Data

Test cases consist of potential entries of processes and data that simulate the system running. Test cases are prepared to contain testing input, test process and expected output. They are created to make sure that the output of the integrated modules are producing the expected output and are working exactly how they are supposed to work. This is simply a way to spot and fix any errors or bugs that might have been made in the integration phase. The tester will then work through the program and document all the data using the test case that were created, according to the test plan.

There may be various test cases that might need to be created to test separate sections of the program. These test cases are gathered together and referred to as test suites, which is a set of test cases. Test data would normally be used in a test case as this would be used to check inputs and expected outputs.

iii) Integration Test Acceptance Conditions

Acceptance conditions specify criteria to judge the behaviour of the program as evaluated by the test cases. While test data is used in order to test the execution of the program or the integrated modules, acceptance conditions define conditions that indicate if the program execution and/or the results thereof are acceptable or not.

Tables Table 3.3, Table 3.2and Table 3.3contain user stories used in the GOEASY project, i.e., GOEASY Platform stories, ApesMobility-specific stories and AsthmaWatch-specific stories, along with respective test cases and acceptance conditions.

2.2 Integration Testing Approaches

There are different approaches to perform integration testing, namely, bottom-up, top-down, big bang or a combination of one or more of them. Bottom-up, top-down and big bang integration testing are discussed as follows:

i) Bottom-up testing

Bottom-up testing tests the lower level components followed by the higher level components, i.e., individual sub-components are tested before their combined component. The components will be separated into the level of granularity and the least important modules will be worked on first, then slowly you work your way up, by integrating components at each level before moving upwards. Bottom up integration testing proceeds as follows: first the lowest level modules are tested individually; lowest modules are then combined to form subsystems; the formed subsystems are tested, and so on.

The advantage of bottom-up testing is that with this method you are able to maintain code more easily and there is a clearer structure on how to proceed. Whereas the main disadvantage is that when releasing a prototype you cannot see a properly working prototype until nearly all the program has been completed: this might take a long time before happening. There may be a lot of errors related to the GUI and programming in the later stages of development. There can also be difficulty in combining subsystems and then testing them.

ii) Top-Down testing

Top-down testing is an approach in which the highest level components are tested first and then, step by step, start working downwards to lower level components. Top-down testing mainly requires the testing team to separate what is important from what is least important, so that the most important modules are addressed first. The top-down approach is similar to a tree in which you would start off by integrating the top level before slowly working your way down, integrating all the components at that level.

The advantage to this approach is that when a prototype is released or shown, most of its main functionalities will already be working. It is also easy to maintain the code and there will be better control in terms of errors, so most of the errors are taken out before moving on to the next stage of testing.

The disadvantage is that it is hard to test the lower level components using test data. Yet, another problem with top-down testing is that lower level modules may not be tested as much as the upper level modules. Because modules at top of structure chart are tested first, starting with the main or control modules, for called modules that are not yet written, it is necessary to use stubs, i.e., simple dummy modules used to avoid linker errors which can result in extra work.

iii) Big Bang Testing

Big bang testing is an extreme case of bottom-up testing where all modules are united tested, then combined together. It can be difficult to fully unit test a module. A more complex driver is usually necessary, which can further complicate the identification of the source of errors. Big bang testing is performed after most or all of the modules are integrated together, forming a nearly complete system. This is very similar to system testing, as this basically has a whole system before the testing begins.

Big bang integration testing is ideal for small system whereas, for big systems, you would have to wait for all the modules to be integrated in order to do big-bang testing: this results in quite a lot of delay before the testing begins. Errors are identified at a very late stage and it is very hard to identify the fault. In big bang testing, it is very difficult to be sure that all testing has been systematically done before product release.

In GOEASY project, a combination of top-down and bottom-up testing approach is used in the continuous integration in order to reap the benefits of both approaches.

3 Components Specification and Integration Testing

This section is divided into two subsections – Section 3.1 and Section 3.2 – and describes detailed specification of components presented in D2.2 (see Figure 3.1) along with their integration and test specification. On one hand, Section 3.1 describes the specification of individual components organized into three as: i) components that are grouped into the GOEASY platform, ii) components that are specific to the ApesMobility application scenario, and iii) components that are specific to AsthmaWatch application scenario. Section 3.2, on the other hand, describes the Integration Test Plan, Test Cases and Acceptance Conditions of user stories that will be implemented by the associated components.

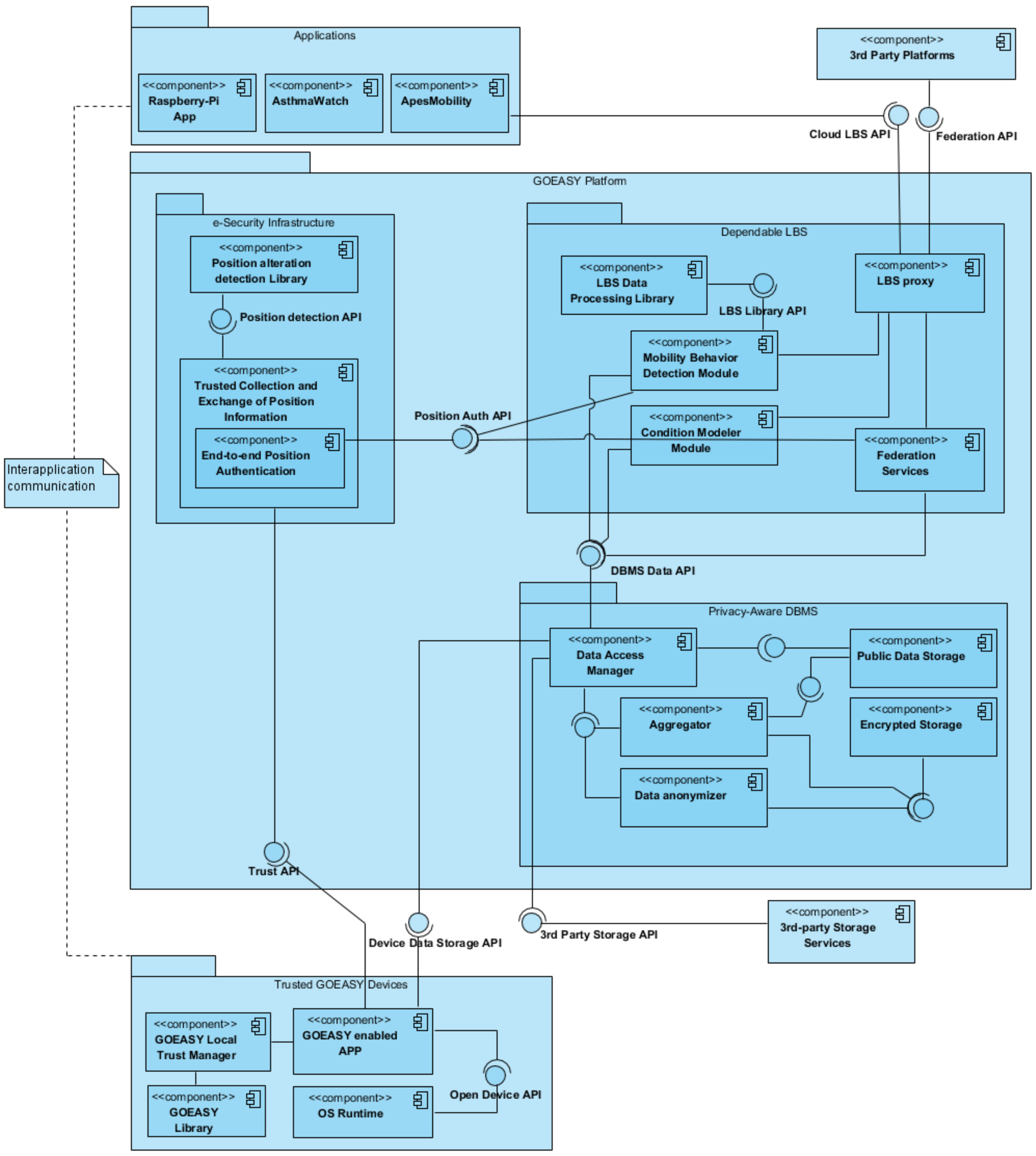
Applications
<<component>> Raspberry-Pi App
<<component>> AsthmaWatch
<<component>> ApesMobility
<<component>> 3rd Party Platforms
Cloud LBS API
Federation API
GOEASY Platform
e-Security Infrastructure
<<component>> Position alteration detection Library
Position detection API
<<component>> Trusted Collection and Exchange of Position Information
<<component>> End-to-end Position Authentication
Position Auth API
Dependable LBS
<<component>> LBS Data Processing Library
LBS Library API
<<component>> LBS proxy
<<component>> Mobility Behavior Detection Module
<<component>> Condition Modeler Module
<<component>> Federation Services
Interapplication communication
DBMS Data API
Privacy-Aware DBMS
<<component>> Data Access Manager
<<component>> Public Data Storage
<<component>> Aggregator
<<component>> Encrypted Storage
<<component>> Data anonymizer
Trust API
Device Data Storage API
3rd Party Storage API
<<component>> 3rd-party Storage Services
Trusted GOEASY Devices
<<component>> GOEASY Local Trust Manager
<<component>> GOEASY enabled APP
<<component>> GOEASY Library
<<component>> OS Runtime
Open Device API

Figure 3.1 Components in GOEASY Functional Architecture (adapted from Deliverable D2.2)

3.1 Specification of Components

3.1.1 Components related to the GOEASY Platform

3.1.1.1 Position alteration detection Library

The aim of this subsystem is to detect and prevent the alteration of the position being collected. As a part of the local trust manager, this subsystem is in charge of the detection of potential spoofing, jamming or manual modification of the positions.

3.1.1.2 End-to-end position authentication

From an end-to-end perspective, and as detailed in D2.2, these are the features that are expected to be part of the authentication system:

- Devices supporting Navigation message will be able to authenticate the signal by using the following three parameters:
    - Data of the reported GNSS message
    - Satellite ID
    - I/NAV message ID
- Devices not supporting Navigation message authentication will require other options to authenticate the position:
    - BQ phones will include a feature to allow apps to double check that the position given by the GNSS system matches with the info provided by the network (see D2.2 figures 13 and 14 for further reference).
    - All the GOEASY apps must include some checks at Android level to ensure that the position cannot be altered manually by end-users:
        - Device is not rooted
        - Bootloader is not unlocked
        - Fake position setting in developer options is disabled

3.1.1.3 Trusted collection and exchange of position information

This module is in charge of guaranteeing the secure and private exchange of position information through the GOEASY platform. This is accomplished in two ways: i) by interacting with GNSS-enabled devices - more specifically with the GOEASY Local trust manager of these- to authenticate them and guarantee they can be trusted when interacting with the GOEASY platform, ii) by overseeing the end-to-end exchange of position data through the platform, from trusted devices to cloud applications or 3rd party platforms or storage services.

3.1.1.4 Data Access Manager

Data access manager performs authentication, authorization and accounting tasks on data storage in the GOEASY platform. This module is central to controlling access to the data, enforcing policies, auditing usage, and providing the information that the requesting client is granted access to.

Authentication identifies the user, by having user credentials before access is granted. Authentication is based on a user having a unique set of criteria for gaining access to the data in the platform. Once the user is authenticated, he/she must gain authorization to do certain tasks. That is, after logging into a system, for example, the user may want to execute commands in order to insert, update or delete data. The authorization process is in charge of determining whether the user has the authority to execute the command. By doing so, authorization enforces policies that apply to a particular user who is authenticated to access the system. The third role of the data access manager is accounting. It measures the usage of the resources by the user, i.e., accounting enables what activities the authenticated user performed.

3.1.1.5 Aggregator

Aggregator protects privacy and security of user data on GOEASY platform. This module uses some threshold to aggregate individual data to hide density of data points in a certain geographic location thereby hiding disclosure of unnecessary details.

3.1.1.6 Data Anonymizer

The data anonymizer module ensures user anonymity by dissociating data from Personally Identifiable Information (PII) through encryption, removal of PII, or "noise" introduction around starting and ending points of routes. This module takes as input, the data that are deemed personally identifiable and performs the aforementioned operations to anonymize the data. The output of the data anonymizer module is the dataset without PII disclosed.

3.1.1.7 Public Data Storage

The public Data Storage components enables storing non PII that will be made publicly available. The data is stored on the cloud and is available for external services. For example, LBS services can have access to this data, analyse it and use it to show their services accordingly.

3.1.1.8 Encrypted Storage

Unlike Public Data Storage (cf. 3.1.1.7), the encrypted storage module enables storage of PII in encrypted form in order to enable mapping of anonymized data to PII for certain services that require private information (e.g., collection of BankoNuts). This storage is a client-side storage (i.e., on the smartphone) so that the user has full control of his/her own private information.

3.1.1.9 GOEASY Local Trust Manager

This component is in charge of local (device-level) trust features and interacts with the platform e-security infrastructure components through the GOEASY enabled App, and more specifically through the Trust API offered by the Trusted Collection and Exchange of Position Information module.

3.1.1.10 GOEASY Library

This library is needed to enable the local trust features and will be updated periodically in order to allow new features to be added to the Local Trust Manager and to improve the way that the GOEASY Enabled Apps are communicated with the GOEASY platform.

3.1.1.11 GOEASY Enabled App

Any application supporting the authentication features described in chapter 3.1.1.2 would be considered GOEASY Enabled Apps, in order to identify applications that are not exposed to the risk of considering the positions as correct that could have been altered.

3.1.1.12 LBS Data Processing Library

Relevant information and resources will be stored in this library supporting the mobility behaviour detection module, for example patterns and information needed to identify a certain travel type.

3.1.1.13 Federation Services

This module is in charge of guaranteeing interoperability with 3rd platforms services. Federation APIs will allow GOEASY to be federated with Internet of Things (IoT) platforms, like FIWARE enablers or

LinkSmart; with Smart City platforms, such as the one developed by the ALMANAC project or the CityPulse one; and with Collective Awareness Platforms, Federation APIs will also be used to leverage third party trust and identity providers.

This module allows services to be combined transparently across different domains, based on common trust agreements, also managing security and privacy issues related to the data flows generated by private data sources. This is done by exploiting techniques to model data flows and authorization mechanisms through descriptive semantic languages, as well as access control methodologies based on open standards and leveraging on dynamic, scalable authentication and authorization mechanisms supporting multi-domain environments.

3.1.1.14 OS Runtime

The device shall be able to provide the necessary information to the GOEASY Enabled Apps to authenticate and collect the positioning information in Runtime with no delay.

3.1.1.15 3rd Party Platforms

This module encompasses Smart City Platforms, IoT platforms and Collective Awareness Platforms (CAPs) that have been already developed and exist in the market, and which are of interest to the GOEASY application domains.

3.1.1.16 3rd-party Storage Services

This module provides storage services to enable integration of 3rd party applications data through 3rd party storage API. In practice, any access to third-party data storage and data navigation such as map data from applications (e.g., Google Maps) is achieved through 3rd party storage services.

3.1.2 Components related to Application Scenarios – ApesMobility

3.1.2.1 ApesMobility

ApesMobility module accepts a series of positions and sensors data fed by the Trusted GOEASY device. It then performs a process of assembling the positions and sensors data into a single user activity, which is sent to the Cloud LBS API to be analysed. The Cloud LBS API returns the result of the analysis as a categorization of the citizen activity.

If the Detection Module has categorized the activity as a sustainable one (e.g. journey by bike) the users is given rewarding points via the dedicated UX and integration with the greenApes platforms. Furthermore, if the user has given consent to the sharing of his/her activities on a public DB, the Cloud LBS API is also used to store the information on the Database Management System (DBMS) The app will be available on Android and will run on GOEASY trusted devices but also on other smartphones to allow comparative analysis.

3.1.2.2 Mobility Behaviour Detection Module

Mobility behaviour detection module takes as input a citizen activity defined by a series of positions (with attached metadata), fed through the Cloud LBS API. The Module then uses the LBS Processing Library to interpret the incoming information and categorize the citizen activity. This produces and returns the activity, now categorized, to the caller application (Apes Mobility) via the Cloud LBS API.

3.1.3 Components related to Application Scenarios – AsthmaWatch

3.1.3.1 Raspberry-Pi App

The sensor system (Raspberry Pi App), continuously collects air pollution data linked to the location where it is collected. This data is transmitted regularly and anonymously stored in the GOEASY platform. The app will interface with air pollution sensors attached to the Raspberry PI gateway. This gateway will be Galileo enabled.

3.1.3.2 AsthmaWatch

AsthmaWatch is the app that is to be used by end-users/patients that have a health condition. They will be using the app to help them manage their daily activities to avoid areas with low air quality. The app will use the Cloud based LBS API (see below) to retrieve current and forecasted air quality at the user’s current location, or at a location specified by the user. The app will run on a smart phone which might be Galileo enabled but not necessarily.

3.1.3.3 Condition Modeler Module

The module is in charge of processing specific criteria (e.g. air pollution) to calculate levels of air pollution for specific areas. Furthermore, considering the calculated conditions, this module must be able to calculate the “healthiest” route given 2 geographical points (departure and destination), possibly integrating data from 3rd party applications.

3.1.3.4 LBS Proxy

Manages requests from the GOEASY applications and from 3rd party platforms federated with the GOEASY platform and forwards them to the specific LBS. This module implements the Cloud Based API. In order to enable the creation and growth of a third party ecosystem we will define standardized, easy-to-use and open APIs for developers. We intend to base our APIs on the Open Geospatial Consortium (OGC) SensorThings format, which defines a data model for exchange of IoT data. The API will provide a high level semantic layer for app developers to use. Different existing ontologies will be evaluated to foster use of common concepts.

3.2 Integration Test Plan, Test Cases and Acceptance Conditions

The integration test plan has the purpose of describing the tests needed to ensure that all of the components of the GOEASY system are properly integrated and function as intended, i.e., to ensure that the unit-tested modules interact correctly. As discussed in Section 2 “Continuous Integration and Testing”, below are some test cases that were created for each of the components listed in D2.2, in order to show exactly how the components should be tested, what test cases should be used and the related acceptance conditions.

Prior to performing integration test specification, it is important to match the user stories identified in D2.1 with associated components from D2.2. Doing so will make sure that when the component integration testing is performed, the requirements in the user story are met. The relationships between user stories and components that implement them are shown in *Appendix A: GOEASY Platform Requirements with Associated Components*, *Appendix B: ApesMobility Requirements with Associated Components* and *Appendix C: AsthmaWatch Requirements with Associated Components.*

For every component identified for GOEASY in Deliverable D2.2, Section 3.2.1, 3.2.2 and 3.2.3 show the test plans vis-à-vis, i) component to be tested, ii) purpose of the test, iii) test cases and procedure, iv) acceptance conditions as well as v) associated components.

3.2.1 Integration Testing of GOEASY Platform

Table 3.1 Test objectives, procedures and acceptance conditions for GOEASY Platform stories

User Story ID	Test Objective	Test Case and Procedure	Acceptance Condition
GOEAS-45	Prove that the exchanged position of a generic user is not spoofed and is securely exchanged through the GOEASY platform Check successful authentication is made Prove that a fake position data fed into the platform is recognized as such by implementing the GOEASY trusted mechanisms	Input: Real GNNS data of a generic user, Fake GNSS data generated and fed into the GOEASY platform Process: Authenticate a generic user according to the implemented algorithm of the Security Framework, Record and exchange user's position information, Check the authenticity of the GNSS data, Provide feedback to the developer Output: A flag that shows the signal is authentic or not	SW developer is able to discriminate between spoofed and real position data
GOEAS-46	1. Protect unauthorized access to the data	Input: wrong access credentials Process: provide wrong credential and try to access the data Output: reject access to data and report the attempt	unauthorized access should be rejected
	2. Allow authorized access to the data	Input: correct access credentials Process: Provide correct credential and try access to the data Output: grant access to the data	authorized access should be granted
GOEAS-47	1. transmit data to GOEASY platform	Input: data to be transmitted, target destination where the data needs to be transmitted to Process: select data to transmit, select target to transmit to, transmit Output: data transmission to GOEASY platform completed	data must be transmitted to GOEASY platform
	2. avail data to be accessed by LBS services	Input: data to be available to LBS, LBS service that needs the data Process: selecting and using the data in LBS services Output: LBS is able to use the data, confirmation or notification returned	data on the GOEASY platform must be available to LBS services
GOEAS-48	Enable data aggregation	Input: Data to be aggregated, Rules and thresholds for data aggregation Process: Select data to be aggregated, select the rules for aggregation, Perform data aggregation using selected rules Output: aggregated data	Data beyond a certain threshold must be aggregated
GOEAS-49	avail data to be accessed by LBS services	Input: data to be made available to the application, application that needs the data	data on the GOEASY platform must be

		Process: selecting and using the data in the application Output: application is able to use the data, confirmation or notification returned	available to the application
GOEAS-50	1. Provide aggregated data, 2. Protect PII	1: Input: Request for aggregated data, Process: try reading aggregated data Output: show aggregated data 2: Input: Request PII, Request non-PII Process: try reading PII Output: refuse showing PII	Developer should access aggregated data, PII should not be accessed
GOEAS-51	Test the performance when we have many simultaneous request for different locations	Input: number of requests, multiple locations Process: perform multiple requests simultaneously for different locations and measure performance Output: response time	The response time should be within acceptable range
GOEAS-53	Verify the reliability of the system in guessing the actual mobility mode; The system should assign points	Input: The test user performs more than 100 mobility "actions". The "score" for each mobility mode is defined. Process: the system (composed by ApesMobility plus GOEASY) analyses the collected data and passes on the points Output: the type of mobility mode and reliability of guess, points are assigned correctly to the user	The mobility mode detection shall be correct more than 90% of the times
GOEAS-54	Verify the reliability of the system in guessing the actual mobility mode; The system should assign points	Input: The test user performs over 100 mobility "actions"; the "score" for each mobility mode is defined; Process: the system (composed by ApesMobility + GOEASY) analyses the collected data and passes on the points Output: the type of mobility mode and reliability of guess, points are assigned correctly to the user.	The mobility mode detection shall be correct more than 90% of the times

3.2.2 Integration Testing of Scenarios – ApesMobility

Table 3.2 Test objectives, procedures and acceptance conditions for ApesMobility stories

User Story ID	Test Obejctive	Test Case and Procedure	Acceptance Condition
GOEAS-20	The user shall have the possibility of selecting if and when tracking of his/her position are allowed	Input: the user has installed the app Process: the user can explicitly choose if he/she wants to enable tracking and detection of position by the AM app (GOEAS-62); the user can pre-set automatic tracking in specific and repeated time-frames (GOEAS-62); the user can change this setting via the AM app settings (GOEAS-62) at any time; the user can select via the UI a limited time frame for tracking his/her journey (GOEAS-63); the user can start/stop tracking ad moments of choice (GOEAS-56); the user can manually check in at POI (GOEAS-13) Output: the tracking of a user position is performed only when the user desires to	the AM app detects the activities of the user only if and when the user has chosen to (acceptance tests passed for GOEAS-62, GOEAS-63, GOEAS-56, GOEAS-64, GOEAS-13)
GOEAS-55	Enable user to share performed activity as a public post on GA platform Scenario A: User activity log is only on the GA platform Scenario B: User activity log is on AM app	Input: List of performed activity, Share button on the UI of GA platform (scenario A), Share button is on the UI of AM app (scenario B) Process: Select activity to share, Share performed activity by clicking the share button, Maybe adding more details (e.g., picture), submit Output: Activity and information attached shall be visible on the GA platform	• Only the activity (and all details) explicitly shared by the user is visible on the GA platform • Members on the platform are able to see the shared activity
GOEAS-56	Enable the user to manually start/stop detection mode in specific moments (e.g. beyond the scheduled time frames provided by User Story GOEAS-64)	Input: The user starts the detection mode by pressing the "start" button on the UI of the AM app Process: The user performs the activity (s)he wants to detect; The user ends the detection mode by pressing the "stop" button on the UI of the AM app Output: The recorded activity is analysed by the AM & GOEASY libraries	• The detection starts when triggered by the user • The detection ends when requested by the user • The recorded activity is correctly analysed by AM/GOEASY
GOEAS-57	The users must be notified for activities	Input: The automatic detection performed by the app under	Once the automatic

	detected in background mode Pre-requisites: • the user has allowed AM notifications on his/her device • the user is accessing the service under User Story GOEAS-63 or GOEAS-64	GOEAS-63 or GOEAS-64 has ended Process: The user checks the phone display Output: The notification confirming the activity detection appears on the phone	detection has ended, the notification appears on the user's phone within minutes
GOEAS-10	The user who has activated the detection of his/her activity shall be able to be rewarded for completing it Pre-requisites: the user is accessing the service under User Story GOEAS-56, GOEAS-63 or GOEAS-64	Input: The detection of a user activity has been completed (GOEAS-55, GOEAS-63 or GOEAS-64) Process: AM (via the GOEASY platform) certifies and classifies the user activity as "rewardable" (e.g. a green trip by bus/bike) Output: Rewarding points are made available to the user as made visible in the UI of AM app and/or in the UI of GA platform (Depending on scenario presented in GOEAS-55). This output provides the Input for GOEAS-9	Once the user has completed a "rewardable activity" the UI should display such activity and the corresponding amount of rewarding points in the "BankoNuts Log" (either on AM app or GA platform).
GOEAS-13	The user shall be able to receive rewarding points for manually "checking in" in a Point of Interest. Pre-requisites: ▪ the user is accessing the service having fulfilled GOEAS-20 ▪ the user is located in proximity of a Point of Interest (pre-defined via the GOEASY platform)	Input: The platform administrator has defined an area in proximity of a chosen point of interest and an associated amount of rewarding points for visiting it; The user is located within such area with his/her smartphone. Process: The user clicks on the "Check-in" button on the UI of the AM app. And awaits detection feedback. Output: The user receives an on-screen message confirming the successful check in; the user also finds the "check-in" activity in the UI of "BankoNuts log" in the GA platform (depending on scenario presented in GOEAS-55). This output provides the Input for GOEAS-9	• The user visualizes the successful check in on the UI of the AM app. • The UI of "BankoNuts log" section on the GA platform (or of the AM app, depending on scenario presented in GOEAS-55) displays the check in and associated rewarding points
GOEAS-9	Enable user to confirm (or not) the claiming of rewarding points for an activity detected.	Input: The user has completed the detection of 2 or more activities under User Story GOEAS-56, GOEAS-63, GOEAS-64 or GOEAS-13; The activities are visible on the activity log Process: The user opens the activity log on the GA platform (or on the AM app depending on the scenario presented under GOEAS-55) and	The GA platform shall: • assign rewarding points for the confirmed activity (activity1) • assign no points for the cancelled activity (activity2)

		a) confirms one of the detected activities (activity1) b) cancels another one of the detected activity (activity2) Output: • Activity 1: the GA platform registers the activity in its backend and assigns points to the user • Activity 2: the GA platform cancels the activity from the UI and the user receives no rewarding points for it	• activity2 is no longer visible on the UI of the activity log
GOEAS-14	Organisations (e.g. city administrations) shall be able to publish on the GA platform some "challenges" to engage citizens Prerequisite: • an admin interface is created for organizations who want to engage users	Input: a challenge is defined for end-uses (e.g. users get 500 bonus rewarding points use local public transport on a given day); the "test-organisation" has access to an admin interface Process: the tester accesses the "test-organisation" admin account; the tester creates a challenge via the interface (e.g. "use public transport on September 15") including an image, a description of the expected activity to complete the challenge and the amount of rewarding points attached; the tester publishes the challenge for a chosen audience (e.g. a subgroup of users who are resident in a city of choice) Output: the challenge is visible on the greenApes platform for the targeted audience	• the "test-organisation" can access the admin account • the "test-organisation" can create a challenge on the admin and publish it on the GA platform • the "end-user" can easily find the challenge on the GA platform • the "end-users" can see the challenge description and image(s) included by the "test-organisation"
GOEAS-15	Citizens shall be able to join special "challenges" created by organisations Prerequisite: • a challenge has been created under GOEAS-14	Input: a challenge is visualised on the UI of the GA platform, including image and description (GOEAS-14) Process: the user finds the challenge and its description on the UI of the gA platform; the user selects the challenge by clicking on the "join" button; the user performs the activity requested by the challenge (e.g. uses public transport in the given time frame) and enables the AM mobility app to detect such activity (GOEAS-56, GOEAS-63 , GOEAS-64 or GOEAS-13) Output: the user is notified by the GA platform that the challenge has been completed; the user receives the	• the user is notified by the GA platform that the challenge has been completed • the user receives the rewarding points connected to the challenge

		rewarding points connected to challenge completion	
GOEAS-60	Citizens shall be notified when new challenges are available Prerequisite: • the user has allowed notifications from the AM app and GA platform	Input: a challenge has been created on the GA platform, including image and description (GOEAS-14) Process: The user checks the phone display; The user opens the notification Output: The notification announcing the newly available challenge is received by the user; Users who open the notification are directed to a page with detailed information about the challenge on the GA platform	• the user receives the notification • by opening the notifications users are directed to the page of the GA platform describing the challenge (GOEAS-14)
GOEAS-58	Citizens shall easily find information about active challenges and POI connected to rewarding points	Input: a challenge has been created on the GA platform, including image and description (GOEAS-14); The platform administrator has defined an area in proximity of a chosen point of interest and an associated amount of rewarding points for visiting it Process: The user can: receive and open a notification (GOEAS-60); navigate in the GA platform and looks for a list of open challenges; navigate in the AM app and see on the map a list of POI where he/she can earn rewarding points Output: the user finds information about open challenges and join them (GOEAS-15); the user can explore a map with nearby POI associated to rewarding points (and earn point for visiting them (GOEAS-13)	• the user can find information about active challenges on the GA platform • the user can take part in the challenge and be rewarded for it (GOEAS-15) • the user can find available POI activated in the system in the AM app • the user can earn points for manually checking in the proximity of POI (GOEAS-13)
GOEAS-61	Citizens shall be notified with information about registered locations/events in their area Prerequisite: ▪ the user has allowed notifications from the AM app and GA platform	Input: New POI and events are made available in the user's city, New POI and events are made available in other regions/Areas Process: the user checks the phone display Output: the user receives a notification about the POI and events happening in his/her city	• the user receives a notification about the POI and events happening in his/her city • the user does not receive a notification about the POI and events happening in other

			areas/regions
GOEAS-62	Citizens shall be able to turn on/off location tracking for the AM app	Input: the user has installed the ApesMobility app and • CASE A denied tracking consent while onoboarding the app • CASE B allowed tracking consent while onoboarding the app Process: • CASE A: the user goes to the app settings and turns on tracking option • CASE B: the user goes to the app settings and turns off tracking options Output: • CASE A: the user can start detecting his/her activities and receive rewarding points • CASE B: no activities can be detected by the user until the user re-allows tracking permissions	• CASE A: the user can successfully detect activities under GOEAS-56, GOEAS-63 , GOEAS-64 and GOEAS-13 • CASE B: the user receives a warning message if trying to start/enable the detection of his/her activities redirecting to the AM app settings to grant tracking permissions
GOEAS-63	Citizens shall be able to activate tracking for a limited time frame when starting the detection of activities (e.g., under GOEAS-56)	Input: the user is about to initiate an activity he would like to detect and be rewarded for; the user starts the detection mode by pressing the "start" button on the UI of the AM app Process: the user selects one of the options presented by the AM app concerning the duration of the tracking phase (e.g. "track me for 1 hour /3 hours /24 hours /until I manually stop the tracking") Output: the tracking is active for the selected time frame but not any longer	• the detection takes place for the duration selected by the user • at the end of the time-frame detection automatically stops
GOEAS-64	Citizens shall be able to activate tracking within time-frames of choice across different periods of time/days, to receive rewarding points with less effort but limiting the tracking carried out by the AM app to those specific time-frames (to preserve battery life and increase privacy perception)	Input: the user has recurring mobility habits and wants to repeat detection at specific hours across several days Process: the user enters the "schedule tracking" feature; the user selects the starting time for automatic-tracking and the closing time for such tracking mode (e.g. start at 7am end at 9am); the user selects the week-days in which such automatic tracking shall be activated (e.g. Mon-Tue-Wed) Output: the settings are saved and the app and tracking	• the tracking is automatically activated during the time frames of choice • detection is not activated (if not manually under GOEAS-56 or GOEAS-13) in other time frames

		happens only in time-frames of choice	
GOEAS-65	Citizens who are sharing anonymous tracked information with third-party organisations (e.g. the mobility planning department of a city) shall not be exposed to privacy risks because of accurate positioning of starting and ending points of their journeys E.g. if a citizen lives in a sparsely populated area his/her commuting routes could allow an easy reconstruction of the citizen identity)	Input: the user has tracked a journey; the user during the on-boarding has given consent to sharing his anonymised journeys with a database/heatmap of a local organisation Process: no action is required by the user (the AM app sends the information to the database via the GOEASY platform) Output: the registered journey are sent to the database with "noise" being introduced for both starting and ending points	• the journey stored in the database does not allow accurate (e.g. greater than 250m) positioning of the starting and ending points

3.2.3 Integration Testing of Scenarios – AsthmaWatch

Table 3.3 Test objectives, procedures and acceptance conditions for AsthmaWatch stories

User Story ID	Test Objective	Test Case and Procedure	Acceptance Condition
GOEAS-24	1. Enable user to select starting and ending point, and potentially time of the day 2. The system displays alternative routes (e.g., fastest route, route with least pollution, optimal route) 3. How long the route takes 4. Data Quality: Average exposure to air pollution	Input: input coordinates on map; time of the day Process: Calculate 3 different routes (fastest route, best route at a given time of the day and optimal route with average polution); observe the behaviour of the system Output: highlighted fastest route; route with time on it; route with average pollution Data Quality: Input: Predicted air quality Process: check it with reference sources Output: accuracy of predicting pollution	Fastest route should be comparable with other benchmark systems Optimal and best route could be compared with reference sources to see if the result makes sense Data quality: accuracy of prediction should be at 80%
GOEAS-25	Using support standard format to enable importing fastest routes from other applications, e.g., Google Maps, RunKeeper	Input: File format that states which 3rd party application the map should come from Process: Parse file, verify on map, and import Output: imported route showed in the map	imported route should look same on our map as it was in the 3rd party application
GOEAS-26	After starting routes, if new conditions	Input: current route, new air pollution data, threshold/rule	Conditions on the route

	exceed the threshold, the route will be recalculated from the current position.	Process: Comparing the new air pollution level on the route with the threshold Output: Recommendation of recalculated route that avoids potential asthma attack	should never be beyond the threshold; should comply with the rules, e.g., user should not stay in the condition for more than a certain duration.
GOEAS-27	1. Enable user to select starting and ending point, and potentially time of the day 2. Select any of the routes to see details: • How long the route takes • level of air pollution along the route	For both test objectives 1 and 2 Input: input coordinates on map, generated routes, time of the day Process: Calculate 3 different routes (fastest route, best route at a given time of the day and optimal route with average pollution); Obtain the time, pollution, etc values for the chosen route and display them on a graph Output: highlighted routes; graph of route details on each route	Routes should be highlighted Every route should show route details on graph
GOEAS-28	Using support standard format to enable exporting selected route to other applications, e.g., Google Maps, RunKeeper, for navigation purpose Using supported standard format to enable importing selected route to other applications, e.g., Google Maps, RunKeeper	Input: the generated route in the correct file format Process: export from the application; import into the 3rd party application Output: exported route showed in the map of 3rd party application	• exported route should look same on our map as it was in the 3rd party application
GOEAS-29	Provide UI to enable entering health data such as lung capacity related information	Input: health data, time the data is entered, location Process: enter the data manually check if it is stored correctly Output: data is entered and stored	stored data should be same as entered data
GOEAS-30	Provide device connectivity enable entering health data such as lung capacity related information	Input: device Process: use the device to enter the data and check if it is stored correctly Output: data is entered and stored	stored data should be same as data is shown on the device
GOEAS-31	Passed test for 29 and 30	Test GOEAS-29 and GOEAS-30 combined	Both 29 and 30 should pass acceptance criteria
GOEAS-32	Provide feedback after the measurements	Inputs: Test of either 29 or 30	Same as 29 or 30

	are stored (i.e., 29 and 30)	Process: carry out test 29 or 30, check the feedback Output: Feedback is displayed to the user	Feedback should be displayed and checked with reference sources and should be relevant in 80% of the cases
GOEAS-33	Display the graph with historical values collected through the device or manually	Input: Selected health parameter, Selected time period Process: Fetch historical values of the selected health parameter for the selected period; Generate graph Output: Graph showing historical values	All the historical values are rendered correctly in a graph
GOEAS-35	Enable Zooming Enable entering/selecting location Getting current location	Input: Map and finger position Process: get coordinates of bounding rectangle on the phone; get the data in this bounding rec; update the display Output: The view is updated according to user's interaction, e.g. more or less of the area is shown (zoom in/out) or entered location is visible in the map. Conditions are also added (based on calculation in GOEAS-16 and contained links)	The map should enable the interactions Map should be updated Map should be updated within accepted response time
GOEAS-36	Enable linking to multiple persons in the contact list of the phone to the app	Input: List of contact persons Process: Select contact persons; add contact person in emergecy contacts list Output: Multiple emergency contact persons are recorded	Multiple (more than 1) emergency contact persons should be recorded

3.3 Process of designing integration tests

The process of integration test is depicted in Figure 3.2 Integration testing process. As shown in the figure, the testing process starts by extracting test specifications from the test design, which is also the basis for the specifications of the testing environment requirements. Then the test environment is set up according to the specifications of test environment requirements. Once the test environment is ready, the actual test execution will take place for the given test specification, producing the test results. The test results will then be evaluated. If there are no issues in the test result, the test specification is deemed successful and it is closed. If, however, the test result has issues, the incident is reported, the test is deemed unsuccessful, and the testing process starts all over again. This procedure continues until all the specifications defined in the test design are tested and the test results are all successful.

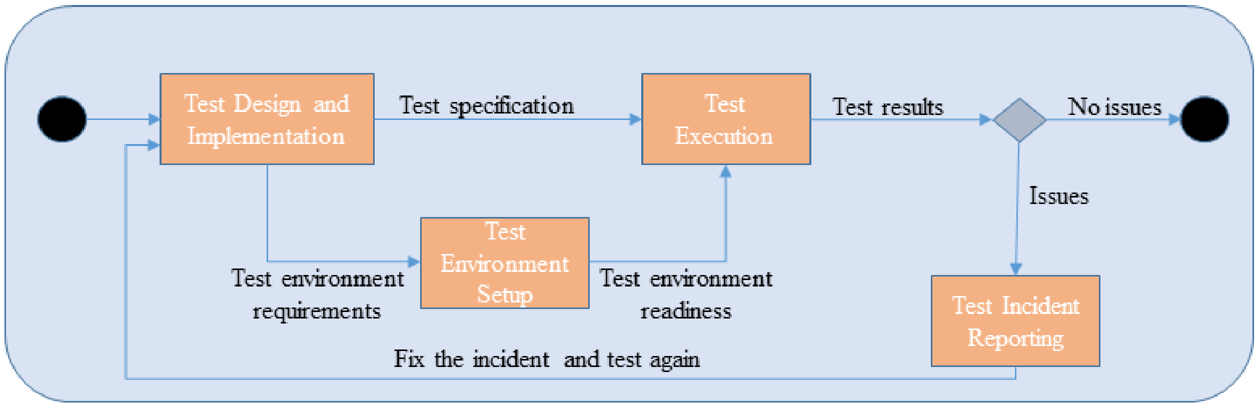
Test Design and Implementation
Test specification
Test Execution
Test results
No issues
Issues
Test environment requirements
Test Environment Setup
Test environment readiness
Test Incident Reporting
Fix the incident and test again

Figure 3.2 Integration testing process

4 Integration Plan

The current section describes the approach to integrate different components from the previous section. Some of the components can be integrated in parallel. Others cannot, therefore an integration order needs to be defined in order for the integration test to be performed. Several steps are the preconditions of a successful integration. The main principle of the whole integration work can be summarized as: testing and integrating small pieces before integrating them in the whole environment.

4.1 Version Control

The foundation of every integration of complex software system starts with a proper version control. In GOEASY, this will be done with help of a common repository. A common repository needs to be setup and kept running already so that developers synchronize their work and system versions are tracked. The repository will be based on Git, a quite popular and tested software. Every GOEASY partner needs to have its own section inside the Git repository. A global write and read access is provided to all partners. Partners should use the repository and avoid creating their own private and untracked modules. This is true especially for common things like interfaces, documentation and distributions.

4.2 Interface Schema Compatibility

All major changes in the interface should be marked with help of interface versions. It is good idea to specify the version number inside a schema. Clients can check the version and easily drop incompatible payloads etc. Beside the versioning of schemas used by interfaces, a validation of syntax helps a lot during the integration phase. The validation can be done before actual integration with remote services. If this is skipped an additional complexity is added to the already complex integration environment.

The validation issue is true for all schema formats, regardless of type being used. Some schema frameworks enforce the syntax compatibility by validating the documents. This is true for eXtensible Markup Language (XML) but not necessary for JavaScript Object Notation (JSON). Since the JSON format is crystalizing as the main communication format between the integration of components, we have to tackle validation issues. The possible ways of tackling the problem are:

i) No validation – involves sending payloads and hoping the other side does not crash. Clearly, although it is requires no extra effort to do validation, this is a very bad approach and should be avoided if possible.
ii) Manual validation – involves running an offline/online validator over the payload document. Web based schema Validators like jsonschemavalidator.net is an example of an online validator.
iii) Automatic validation – is similar to mechanisms provided by XML frameworks where validation is performed automatically with minimal intervention. Automatic validation is the best approach except that it is not feasible because of extra implementation effort.

The manual validation is easy to do if a JSON schema is specified. Once such schema is specified, easy-to-use validators can be used to test the compatibility of a given JSON document. It is strongly suggested to use such an approach, and specify and finalize schemas in between integration of different components. Stable and well defined schemas enforce reliability of the whole system. This pays off in regard to integration and bug fixing efforts.

4.3 Debug and Maintenance Messages

Apparently, all integration clusters and modules will produce errors. This is an inevitable fact for all complex systems. Debugging an isolated piece of code is relatively easy. In a distributed heterogeneous network of components the debugging task gets complex. Such systems can consists of thousands of instances running different kinds of software systems. Sometimes they run even identical software but behave differently. Tracking issues like this can be done with the help of exhaustive messaging. In the beginning of the integration, such messages help with debugging, while later on they guide the maintenance of the system. There is no strict difference between those. One can consider the first more specific, and the second more generic. The formal definition of such a message can be very simple. For example, for a subsystem it could look like:

```
{
        TimeStamp: UTC,
        SubSystemID: SomeID,
        Message: “real debug message!”,
        ErrorCode: 404
        DebugLevel: MAINTENANCE
}
```

The messages should be sent to the cloud and stored there. Further diagnostics and problem mitigation processes can be applied thereafter. All modules in every integration cluster need to implement such mechanism. Even if a given module can be tested in isolation and the module is perfect, the later deployment in a production environment requires maintenance output.

Therefore it is important to integrate such messaging system from the beginning on. First it serves during the test and debugging phase, sending lots of logs to the cloud. During the final deployment, only the most important messages are sent to the cloud. One can compare the different levels to the log levels in different log frameworks. Messaging level from every module should be configurable with values like TRACE, DEBUG, INFO, MAINTENANCE etc.

5 Conclusion and Next Steps

The current deliverable presents an integration framework specification for the overall GOEASY system. The deliverable bases itself on D2.1 for scenarios and D2.2 for components and architecture in order to provide component specifications for the overall integration of the system. Furthermore, the deliverable provides test and integration plans for these components and their interaction. As the integration follows Continuous Integration philosophy, small changes should be integrated as soon as possible to receive early feedback about possible problems. To keep on refining the user stories, components and their specifications, regular integration meetings are suggested to ensure a better integration of the overall system.

While the user stories, components and their specifications, integration and testing techniques are specified in this deliverable, they are subject to further refinement in the upcoming deliverables. For example, components and/or user stories that belong to the GOEASY platform could move to the specific applications, i.e., ApesMobility or AsthmaWatch or vice versa. Hence, the update in these aspects will be documented in deliverables D5.3, D5.5 and D5.7 among others.

Acronyms

Acronym	Explanation
API	Application Programming Interface
CAPS	Collective Awareness Platforms
DB	Database
DBMS	Database Management System
GNSS	Global Navigation Satellite System
IoT	Internet of Things
JSON	JavaScript Object Notation
LBS	Location-Based Services
OGC	Open Geospatial Consortium
PII	Personally Identifiable Information
XML	eXtensible Markup Language

List of figures

List of tables

References

IEEE 1074-1997 -- IEEE Standard for Developing a Software Project Life Cycle Process. *https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1665059*

IEEE 1012-2012 - IEEE Standard for System and Software Verification and Validation

ISO/IEC., *Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models,* Technical Report ISO/IEC 25010:2011(E), 2011.

IEEE 29119-1-2013, Software and systems engineering Software testing - Part 1:Concepts and definitions

IEEE 29119-2-2013, Software and systems engineering Software testing -Part 2:Test processes

IEEE 29119-3-2013, Software and systems engineering Software testing -Part 3:Test documentation

Handbook of Software Quality Assurance, Gordon Schulmeyer, Artech House, 2007

JSON schema – http://json-schema.org

Appendix A: GOEASY Platform Requirements with Associated Components

User Story ID	Description	Associated Component/s
GOEAS-67	As a citizen, I want to be able to manage permissions of my data in order to protect my privacy.	ApesMobility, LBS Proxy
GOEAS-66	As a SW application developer, I want to be able to anonymize data to hide details that endanger personal identifiable information.	Data Access Manager, Data Anonymizer , Encrypted Storage, Federation Services, LBS Proxy
GOEAS-54	As a SW application developer, I want to be able to detect specific mobility behavior of users without a data connection in order to reduce data and battery consumption of the mobile device.	ApesMobility, GOEASY Enabled App, LBS Proxy, Mobility Behaviour Detection Module, OS Runtime, GOEASY Local Trust Manager
GOEAS-53	As a SW application developer, I want to certify specific mobility behavior of users in order to reward them for sustainable behavior.	ApesMobility, Data Access Manager, GOEASY Enabled App, LBS Data Processing Library, LBS Proxy, Mobility Behaviour Detection Module, Trusted collection and exchange of position information
GOEAS-52	As a SW application developer, I want to know about conditions regarding air pollution based on location data in order to inform users about possible dangers.	AsthmaWatch, Condition Modeler Module, Data Access Manager, GOEASY Enabled App, LBS Proxy
GOEAS-51	As a SW application developer, I want to make use of GOEASY's location-based services to enhance my application for its users.	LBS Data Processing Library, LBS Proxy
GOEAS-50	As a SW application developer, I want to access aggregated data in order to ensure to not deal with personal data.	Aggregator, Data Access Manager, Federation Services, LBS Proxy
GOEAS-49	As a SW application developer, I want to access data from the GOEASY platform to make use of it in my application.	Data Access Manager, Federation Services, LBS Proxy, GOEASY Enabled App
GOEAS-48	As a SW application developer, I want to be able to aggregate data to hide details that endanger personal identifiable information.	Aggregator, Data Access Manager, Encrypted Storage, Federation Services, LBS Proxy, Public Data Storage
GOEAS-47	As a SW application developer, I want to transmit data to the GOEASY platform to make it available for location-based services.	Aggregator, ApesMobility, Data Access Manager, Data Anonymizer , Federation Services, GOEASY Enabled App, LBS Proxy, Public Data Storage, Raspberry-Pi App, GOEASY Library
GOEAS-46	As a GOEASY developer, I want to be able to control who has access to stored data in order to protect data from unauthorized access.	Data Access Manager
GOEAS-45	As a SW application developer, I want to get an	GOEASY Library, GOEASY Local Trust Manager,

	authentication service to make sure the location data is not spoofed.	Position alteration detection, Trusted collection and exchange of position information

Appendix B: ApesMobility Requirements with Associated Components

User Story ID	Description	Associated Component/s
GOEAS-65	As a citizen, I want to be able to delete identifiable location data to protect my privacy.	ApesMobility
GOEAS-64	As a citizen, I want to be able to automatically enable and disable location tracking on a regular basis to protect my privacy.	ApesMobility
GOEAS-63	As a citizen, I want to be able to enable location tracking only for a specific time frame to protect my privacy.	ApesMobility
GOEAS-62	As a citizen, I want to be able to en-/disable location tracking manually to protect my privacy.	ApesMobility, OS Runtime, GOEASY Local Trust Manager, GOEASY Library, GOEASY Enabled App
GOEAS-61	As a citizen, I want to subscribe to specific areas or topics only in order to get relevant information only.	ApesMobility
GOEAS-60	As a citizen, I want to be notified about new challenges in order to improve my sustainable behavior.	ApesMobility
GOEAS-59	As a citizen, I want to see additional information about registered locations/ events in order to decide whether to go there or not.	ApesMobility
GOEAS-58	As a citizen, I want to be able to find registered POIs and challenges in order to collect additional rewards by check-in (and share).	ApesMobility
GOEAS-57	As a citizen, I want to be notified about a detected, certified activity in order to not miss additional rewards.	ApesMobility
GOEAS-56	As a citizen, I want to be able to trigger detection of mobility behavior manually too!!	ApesMobility, GOEASY Enabled App, GOEASY Library, GOEASY Local Trust Manager, OS Runtime
GOEAS-55	As a citizen, I want to be able to share my certified activities in order to motivate fellow citizens.	ApesMobility
GOEAS-20	As a citizen, I want to have full control of	ApesMobility

	the location tracking in order to protect my privacy.	
GOEAS-15	As a citizen, I want to join challenges to get motivated and to support the community.	ApesMobility
GOEAS-14	As an organization, I want to offer challenges for users in order to familiarize them with sustainable topics.	ApesMobility
GOEAS-13	As a citizen, I want to get my location certified to get additional rewards.	ApesMobility, GOEASY Enabled App, LBS Data Processing Library, Trusted collection and exchange of position information, , GOEASY Library, GOEASY Local Trust Manager, OS Runtime
GOEAS-12	As a citizen, I want to share my location to inspire others and/or get additionally rewarded.	ApesMobility, GOEASY Enabled App
GOEAS-11	As a citizen I want to join activities shared by others to get inspired for sustainable behavior.	ApesMobility
GOEAS-10	As a citizen, I want to get my activity certified to get additional rewards.	ApesMobility, GOEASY Enabled App, LBS Data Processing Library, Mobility Behaviour Detection Module
GOEAS-9	As a citizen, I want to be able to confirm my certified activity to be additionally rewarded.	ApesMobility, GOEASY Enabled App, Mobility Behaviour Detection Module, GOEASY Library, GOEASY Local Trust Manager, OS Runtime

Appendix C: AsthmaWatch Requirements with Associated Components

User Story ID	Description	Associated Component/s
GOEAS-44	As a user I want to be able to determine the order in which contacts are contacted to prioritize based on criteria.	AsthmaWatch
GOEAS-43	As a user I want to be able to create and update templates for text messages to inform people faster about my current situation.	AsthmaWatch
GOEAS-42	As a user I want to be able to select text messages to send to all emergency contacts in case I cancelled a call in order to let them know about my current situation.	AsthmaWatch
GOEAS-41	As a user I want to be able to cancel a call at any time to avoid unnecessary attempts of contacts.	AsthmaWatch
GOEAS-40	As a user I want to view all outgoing calls in the app due to an activated alarm to know who was informed and who I should update about my situation now.	AsthmaWatch
GOEAS-39	As a user I want my phone to play an alerting sound when I activated the alarm to get immediate help by people around me.	AsthmaWatch

GOEAS-38	As a user I want to inform my emergency contacts about my position when I activated the alarm to get help as fast as possible.	AsthmaWatch
GOEAS-37	As a user I want to be able to activate an alarm in case of an asthma attack in order to get immediate help.	AsthmaWatch
GOEAS-36	As a user I want to be able to specify multiple persons as emergency contacts to increase the probability to get help.	AsthmaWatch
GOEAS-35	As a user I want to control the map by known interaction patterns to get the required information as fast and easy as possible.	AsthmaWatch
GOEAS-33	As a user I want to be able to view historical data in order to know about the development of my health.	AsthmaWatch
GOEAS-32	As a user I want to get immediate feedback after adding health information in order to react in time on changes.	AsthmaWatch
GOEAS-31	As a user I want to be able to enter health information to the application in order to keep track of my health.	AsthmaWatch
GOEAS-30	As a user I want to be able to add health information automatically by a connected device to keep track of my health.	AsthmaWatch
GOEAS-29	As a user I want to be able to enter health information manually to keep track of my health.	AsthmaWatch
GOEAS-28	As a user I want to be able to export a picked route in order to start navigation with a 3rd party app.	AsthmaWatch
GOEAS-27	As a user I want to be able to view details of any suggested route in order to avoid an asthma attack.	AsthmaWatch
GOEAS-26	As a user I want to get alternative routes suggested automatically if conditions along my preferred route exceed my rules in order to avoid an asthma attack.	AsthmaWatch, Condition Modeler Module
GOEAS-25	As a user I want to be able to import a route from 3rd party app(s) in order to reduce redundant tasks (defining the same route in different applications).	AsthmaWatch
GOEAS-24	As a user I want to be able to define a preferred route in order to get advised whether to take it or not.	AsthmaWatch