Bloomen

Blockchains in the new era of participatory media experience

HORIZON 2020

762091 – BLOOMEN - H2020-ICT-2016-2
ICT-19-2017 Media and content convergence

D4.7 Bloomen overall integrated system - 1st cycle

Version:	1.0
Date:	31/05/2019
Authors:	ATC
Type:	Demonstrator
Dissemination level:	Public

Co-funded by the Horizon 2020 programme
of the European Union

Table of Contents

1 Introduction

The overall integrated system of Bloomen refers to the fully functional set of Bloomen modules together with the Bloomen platform which will fulfil the requirements of Bloomen project. The various functional components of Bloomen are brought together and interact with each other in order to cover the needs of a variety of business workflows. In this deliverable we are covering the methodology and the definition of the approach we have used to integrate and validate the various modules of Bloomen into one single system.

The integration is always an important aspect of technical development. In this particular project integration enables the combination and support of three different use cases, for music, photos and streaming video, in varying degree. Core elements, such as smart contracts are available via an extensive API, which simplifies calling features and functionalities where needed.

Finally, a highly important element is learning about how to put blockchain technology to good use. Finding use cases is as important as tackling the limitations of this nascent, though promising technology.

1.1 About this deliverable

This deliverable is the first cycle of a software prototype. In the course of the Bloomen project the deliverable is the specific outcome of Task 4.4 “Overall integration and validation”. The software prototype is accompanied by this short document, which provides a description of the approach we are using for the integration and validation of the system, providing, also, short descriptions of the different components to be integrated.

The next iteration of this deliverable, the second cycle, will be delivered in M30.

1.2 Document Structure

- The document provides a general description of the Integration Plan and Methodology in Section 2;
- Section 3 presents the different components to be integrated.
- Section 4 provides details about the Testing Plan
- Section 5 contains the conclusions and the plans for the next iteration.

2 Integration Plan and Methodology (Overall)

For the integration purposes of the project we are going to follow the Agile Software Development Practices with frequent integration cycles, rapid prototyping and close collaboration between self-organizing, cross-functional teams. Based on agile principles, we are also going to apply Continuous Integration techniques to perform automated building, testing and deployment of the provided modules. For adopting Continuous Integration practices have set-up a development environment containing a set of continuous integration tools.

Continuous integration (CI) comprises practices such as daily builds and additional checks so as to prevent bugs. In order to enable automatic daily builds, Continuous Integration software gathers the whole source code in one place (with different revisions), automates the build process and testing, and provides the latest working executable to anyone involved in the project.

The CI model comprises a set of activities for the process implementation: building the system, running tests, deployment activities, and finally reporting test and deployment results. The practice of Continuous Integration assumes a high degree of tests, which are automated into the software: a facility that can be seen as “self-testing code”, often using a testing framework.

In the scope of Bloomen project, we will try to adopt an Agile development process based on the Kanban methodology. The methodology is based on the assumption that all tasks are described as tickets which are assigned to Bloomen team members and are distributed and collected in various columns of a Kanban board. The initial requirements are described as user stories.

The main integration of each pilot with the Bloomen platform will be achieved through the Bloomen platform API. The API provides all the necessary functions for each use case, which are constantly being updated through a shared spreadsheet, and provides an abstraction layer to the complexity of the lower level blockchain functionalities, wherever these are needed.

The User Management and Authentication modules, and all other global functionalities, will be provided for all use cases through the shared Admin Panel, which is an extension of the Bloomen platform.

3 System Components

The components that need to be integrated are presented in this section. The description of every component is accompanied by the methodology of the integration, the fulfilled requirements and the testing plan.

3.1 Bloomen Wallet

3.1.1 Description and role in Bloomen system

The Bloomen wallet consists of an application for mobile devices that allows interaction with smart contracts hosted in the Alastria Blockchain. The distributed nature of this technology allows us to access functionalities/services offered by smart contracts from any node of the blockchain without having a single point of interaction as in traditional systems.

This component has a specific deliverable within the project (**D4.3 - Bloomen mobile clients - 1st cycle**) where you can find documentation about its functionalities.

3.1.2 Integration with Bloomen platform

This component is integrated with the other modules through Smart Contracts deployed in the blockchain.

3.1.3 Integration Methodology

Ethereum JSON RPC API.

3.1.4 Requirements fulfilled

The requirements fulfilled can be found in Table 1.

ID (from D2.2)	Name	Fulfilled
RNF-GT-4	Privacy by design Data Storage	X
RNF-GT-8	User Digital Wallet	X
RNF-GT-9	Anonymous Personalization	X
RF-WTV-1	User Registration	X
RF-WTV-2	Create Account	X
RF-WTV-3	Create Wallet	X
RF-WTV-4	View Wallet	X
RF-WTV-7	Purchase Product	X

Table 1: Requirements fulfilled for Bloomen Wallet

3.1.5 Testing Plan

Manual testing scripts

3.2 Kendraio App

3.2.1 Description and role in Bloomen system

The Kendraio App provides a framework for prototyping copyright management workflows within the Bloomen project. As such the application intends to extend features and functionalities, based on the Bloomen core architecture. The framework will also be developed to be used as a management portal for accessing Bloomen administrative features.

3.2.2 Integration with Bloomen platform

The Kendraio App is a standalone app that integrates via the Bloomen API

3.2.3 Integration Methodology

The Kendraio App uses Adapters to provide integration with external systems and services. The current iteration of the Kendraio App uses the Bloomen REST API via HTTP connection. Integration with blockchain components via web3 may be required for the functionality to be developed as part of the management portal.

3.2.4 Requirements fulfilled

The requirements fulfilled can be found in Table 2.

ID	Name	Fulfilled
1	Tag created content	X
2	Define usage terms	To be fulfilled in the final iteration
3	Bloomen API integration	X
4	Copyright management workflows	Prototyped
5	Asset management	X
6	Integration with third party APIs	Partially fulfilled (to be completed in the final iteration)

Table 2: Requirements fulfilled for Kendraio App

3.2.5 Testing Plan

The testing plan for the Kendraio App aims to ensure correct operation of the application by using manual testing scripts.

3.3 REST API

3.3.1 Description and role in Bloomen system

The Bloomen platform API is a RESTful API that allows the different pilots to use basic functionality for each of their cases, such as authentication management, file uploads, etc., and gain access to blockchain operations. The API achieves this without having to interact with the blockchain infrastructure and general complexity, thus making blockchain operations as simple as an HTTP API call. The role of the Bloomen platform API is to glue together the different functionalities for the already created pilots, as well as potential future pilots created by the blockchain or any other communities interested in secure transparent transactions. Further the API removes all the sophisticated knowledge otherwise required by developers to use blockchain technologies. The API replaces these needs through simple and easily customized components. This approach opens the door to fill as many use cases as possible, thus creating a Bloomen ecosystem of applications.

3.3.2 Integration with Bloomen platform

The Bloomen platform API is at the center of the integration effort, as it provides an interface for all the basic functionalities that the pilots require for them to work. Each pilot uses the parts of the Bloomen platform API that it needs for basic functionalities, but is also free to use whatever else backend, or blockchain, it chooses to, to complete its purpose. If pilots need direct access to the smart contract and blockchain functionalities, they can use different tools from the API for direct integration with the blockchain, such as the web3 library.

3.3.3 Integration Methodology

For the integration purposes of the Bloomen platform we follow the Agile Software Development Practices with frequent integration cycles, rapid prototyping and close collaboration between self-organizing, cross-functional teams. Based on agile principles, we are also applying Continuous Integration techniques for performing automated building, testing and deployment of the provided modules. For adopting Continuous Integration practices we are going to setup a development environment containing a set of continuous integration tools such as the nyc testing framework, Jenkins, etc.

3.3.4 Requirements fulfilled

The requirements fulfilled can be found in Table 3.

ID (from D2.2)	Name	Fulfilled
RF-PH-1	Upload image	X
RF-PH-2	Set image price	X
RF-PH-3	Browse images	X
RF-PH-4	Search for images	X
RF-PH-5	Filter images	To be fulfilled in 2nd iteration
RF-PH-6	Buy image rights	X
RF-PH-7	Give image rights	To be fulfilled in 2nd iteration
RF-PH-8	Vote contributor	X
RF-PH-9	View analytics	X
RNF-PH-1	Set privacy settings	X
RNF-PH-2	Workflow efficiency	X
RNF-PH-3	Reliability of payments	X

Table 3: Requirements fulfilled for REST API

3.3.5 Testing Plan

The basis of the Bloomen platform API testing is unit testing the different controllers for each route, as well as every functionality and service that can be tested, including the contracts and interactions with the contracts. Since the smart contract functionality is also tested through the individual route and service unit tests, this kind of testing could also be considered integration testing between the Express server and the Smart Contract and blockchain infrastructure.

For unit testing the application we have used the mocha and chai libraries as well as the tools provided by the truffle suite, to test the contracts individually. In addition, with these frameworks, we use nyc CLI tool to keep track of test coverage and keep testing relevant, after making changes to the code.

3.4 Contracts

3.4.1 Description and role in Bloomen system

“Smart contracts” are a big innovation of blockchain technologies. The promise here is to reduce the manual work of setting up and the even more complex work of checking the actual transactions versus the fine print agreed on in an analog contract. This is why, in the course of Bloomen smart contracts are considered an important technology element.

Two smart contracts have been built and deployed in the context of Bloomen System, the PRM Smart Contract and the PRM Token. The PRM Smart Contract is considered responsible for the implementation of all the methods required in order an external web service to interact with the blockchain as well as the data stored within. As a matter of fact, it means that this contract manages the sale and purchase of photos along with their ownership rights. On the other hand, PRM Token is an example ERC223 that interacts with the PRM Smart Contract to fulfill all the payment procedure of the blockchain transactions.

3.4.2 Integration with Bloomen platform

Both contracts are written in Solidity and are deployed on Quorum instance running inside Alastria ecosystem. All the communication between blockchain and other components is implemented in JavaScript through web3.js library. Although it is highly recommended to use the Bloomen API by all external applications, there is an ability to interact directly with the blockchain by using web3.js or a similar library. Developers are not encouraged to do so, but, nevertheless, it is supported by the platform.

More details about the smart contracts can be found in the open-source GitHub repository: https://github.com/atcilab/Smart-contract.

3.4.3 Integration Methodology

For the integration purposes of the contracts in this project, the Agile Software Development Practices were followed. This methodology requires frequent integration cycles, rapid prototyping and close collaboration between self-organizing, cross-functional teams. In particular, Kanban methodology was adopted in this component. The methodology is based on the assumption that all tasks are described as tickets which are assigned to Bloomen team members and are distributed in the various columns of a Kanban board. The initial requirements are described as user stories that typically follow the following template: As a < type of user >, I want < some goal > so that < some reason > e.g. "As a consumer, I want to search uploaded photos so that I can find the right one.". The lifecycle of development was split in 2-weeks iterations, named Sprints. Each Sprint, contains a board where tasks were organized between "TODO", "In Progress", "To Test" and "Done" columns.

3.4.4 Requirements fulfilled

The requirements fulfilled can be found in Table 4.

ID (from D2.2)	Name	Fulfilled
RF-PH-1	Upload image	X
RF-PH-2	Set image price	X
RF-PH-3	Browse images	X
RF-PH-4	Search for images	X
RF-PH-5	Filter images	To be fulfilled in 2nd iteration
RF-PH-6	Buy image rights	X

RF-PH-7	Give image rights	To be fulfilled in 2nd iteration
RF-PH-8	Vote contributor	X
RF-PH-9	View analytics	X
RNF-PH-1	Set privacy settings	X
RNF-PH-2	Workflow efficiency	X
RNF-PH-3	Reliability of payments	X

Table 4: Requirements fulfilled for Contracts

3.4.5 Testing Plan

The testing plan for the contracts aims to ensure that the produced output will meet all defined requirements regarding stability, performance, safety and business needs. The areas of software testing covered by Bloomen contract are the Unit testing and Beta software testing. In particular, Unit testing was conducted by using the Truffle environment that offers a dedicated testing framework. The Truffle testing framework offers an automated process while ensuring that a fresh set of contracts is used during each process. An additional benefit is the ability to review contracts while coding in Solidity, throughout the development stage. In order to track bugs affecting the business functionality, we used the Atlassian Jira platform. As mentioned earlier, the software development process followed the agile methodology based on sprints. The development and testing teams worked closely together with the domain experts collaborate through Jira to monitor the progress of development. Any issues or malfunctions were reported and addressed, in order to maintain a high product quality.

3.5 Anonymisation Module

3.5.1 Description and role in Bloomen system

The Anonymization Module offers privacy preserving techniques, personalization and identity management over blockchain enabled media delivery platforms.

It focuses on the distribution and decentralization of the identity management functionality including anonymity and personalization functionalities as well. The main prototype of this module is described in deliverables D3.4 - Anonymous personalization services - 1st cycle and D3.5 - Anonymous personalization services - 2nd cycles.

3.5.2 Integration with Bloomen platform

The integration with the Bloomen platform is simplified by using the Bloomen API, enabling connections between different modules and across current and future use cases.

3.5.3 Integration Methodology

The Anonymization Module is communicating with the different components of Bloomen Platform through the Bloomen API. For example, the communication with smart contracts deployed on Quorum blockchain platform is based on Bloomen API.

3.5.4 Requirements fulfilled

The requirements fulfilled can be found in Table 5.

ID (from D2.2)	Name	Fulfilled
RNF-MUS-1	Privacy Policies	Partially covered The system must allow the application of policies to access and share data and ensure the privacy of users, with special care for users' personal data. The final policies are going to be implemented at the last iteration of the document after having received input from the pilots.
RNF-GT-1	Privacy by design Logic	X
RNF-GT-2	Privacy by design Data Anonymization	X

Table 5: Requirements fulfilled for Anonymization module

3.5.5 Testing Plan

A first step in the testing process of the integration-connection of Anonymization service with Bloomen Platform, is the use of the Anonymization Service in the WebTV pilot. Additionally, our aim is to ensure that the produced output will meet all defined requirements, which are previously presented.

3.6 Management portal

3.6.1 Description and role in Bloomen system

The Management Portal provides access to an admin user interface for common functionality across the whole Bloomen platform. Requirements that are related to administration of the system or management of the pilots are to be implemented here.

3.6.2 Integration with Bloomen platform

The implementation of the management portal relies on integration with the Bloomen platform by connecting into the Bloomen API. It builds on top of already implemented functionality to provide a user interface for management of the Bloomen platform.

3.6.3 Integration Methodology

All functionality will be present and address in the API.

3.6.4 Requirements fulfilled

The requirements fulfilled can be found in Table 6.

ID (from D2.2)	Name	Fulfilled
RF-MUS-1	Asset Registration Manager	partially fulfilled
RF-MUS-2	Batch Registration Manager	partially fulfilled
RF-MUS-3	Asset Search	partially fulfilled
RF-MUS-4	Assets Explorer	partially fulfilled
RF-MUS-5	Changelog	partially fulfilled
RF-MUS-6	Notification System	partially fulfilled
RF-MUS-7	Duplicate Search	partially fulfilled
RF-MUS-8	Core Metadata Edition	partially fulfilled
RF-MUS-9	Edition Request Management	partially fulfilled
RF-MUS-10	Link Assets	partially fulfilled
RF-MUS-11	Merge Assets	partially fulfilled
RF-MUS-12	Merging Request Management	partially fulfilled

Table 6: Requirements fulfilled for Management Portal

3.6.5 Testing Plan

Manual testing scripts

3.7 Decentralized Rights Management Tool

3.7.1 Description and role in Bloomen system

This component in its first iteration allows users to create copyright claims in a decentralized manner. The basic idea is to have a shared ledger between partners that raises up claim conflicts once are detected by a smart contract.

As we said this first iteration is more a technology demonstrator rather than final solution component. This component has more documentation in other deliverable within the project (**D3.2-Demo1**) where you can find information about its functionalities.

3.7.2 Integration with Bloomen platform

This component (in this first iteration) is integrated with the other modules through Smart Contracts deployed in the blockchain.

3.7.3 Integration Methodology

Ethereum JSON RPC API.

3.7.4 Requirements fulfilled

The requirements fulfilled can be found in Table 7.

ID (from D2.2)	Name	Fulfilled
RF-MUS-3	Asset Search	Partially fulfilled in 1st iteration
RF-MUS-13	Claiming	Partially fulfilled in 1st iteration
RF-MUS-14	Claims Explorer	Partially fulfilled in 1st iteration
RF-MUS-15	Conflict Resolution	Partially fulfilled in 1st iteration

Table 7: Requirements fulfilled for then Decentralized Rights Management Tool

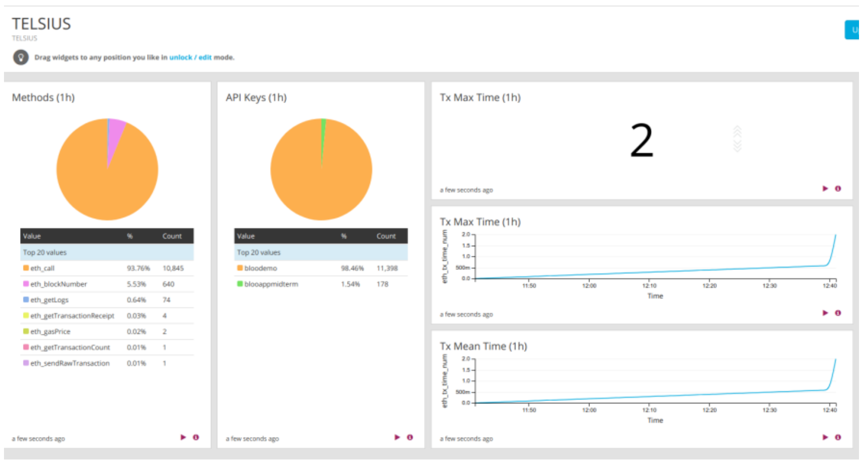
3.7.5 Testing Plan

Manual testing scripts

3.8 BaaS Portal

3.8.1 Description and role in Bloomen system

This component is a platform that allows us to control and monitorize the access to Quorum nodes offered by Bloomen. In this first iteration we only offer access to mainnet (Telsius), testnet (Arrakis) and show the utilization of both on a per net dashboard.



Value	%	Count
Top 20 values		
eth_call	93.76%	10,845
eth_blockNumber	5.53%	640
eth_getLogs	0.64%	74
eth_getTransactionReceipt	0.03%	4
eth_gasPrice	0.02%	2
eth_getTransactionCount	0.01%	1
eth_sendRawTransaction	0.01%	1

Value	%	Count
Top 20 values		
bloodemo	98.46%	11,398
blooappmidterm	1.54%	178

Figure 1: Telsius Dashboard

3.8.2 Integration with Bloomen platform

Not required.

3.8.3 Integration Methodology

Ethereum JSON RPC API.

3.8.4 Requirements fulfilled

This component is a transversal solution to the entire platform and is not related with concrete requirements.

3.8.5 Testing Plan

Manual testing scripts.

4 Testing Plan

4.1 Software Testing Types

"Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results"[1]. There are several testing approaches, each one targeting different aspects of software: feature testing, black-box testing, regression testing, component testing are some of these, each aiming at different objectives.
In addition to those testing approaches, testing procedure is divided into four main testing levels:

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

Throughout the testing lifecycle, the levels above are repeated as individual steps in an iterative manner until the software is fully accepted by the target users.

4.1.1 Unit Testing

"In computer programming, unit testing is a method by which individual units of source code are tested to determine if they are fit for use. A unit is the smallest testable part of an application." [2]

We describe below how this approach is used in Bloomen (Figure 2).

This is an iterative process where each iteration begins by writing code within the local development environment and ends by committing code changes to Github repository. So, unit testing practically starts in each user's local environment of is performed twice before the code is committed on Github code versioning repository.

Then, after the code is committed to the public repository, Jenkins detects the update and runs all build and test goals or waits until the nightly build is scheduled to execute all the build/test goals, as well as the SonarQube analysis task. By keeping the integration team alerted on all errors or failing test cases, we have accomplished a more centralised monitoring of nightly build results; the team had then the responsibility to communicate the arisen issues to the module owners, together with some other useful information (logs, configuration data, error messages etc.).

[1] Hetzel, William C., The Complete Guide to Software Testing, 2nd ed. Publication info: Wellesley, Mass. : QED Information Sciences, 1988. ISBN: 0894352423.Physical description: ix, 280 p. : ill ; 24 cm.

[2] http://en.wikipedia.org/wiki/Unit_testing

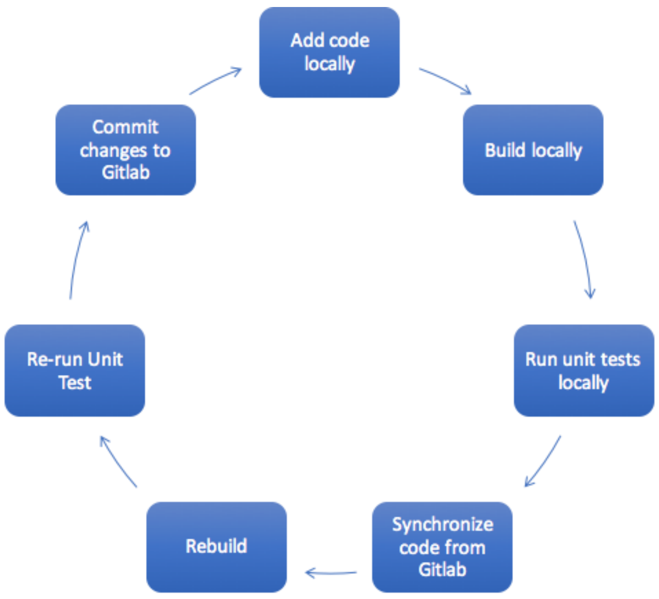


Figure 2: Iterative Process of testing

4.1.2 Integration Testing

The integration testing will be performed by the integration team with the close collaboration of each feature owner. This will take place manually at first by using mockup objects or test scenarios. For the execution of integration testing it is important to have clear instructions from the module providers with steps to reproduce in order to successfully test the service.

4.1.3 System Testing

System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing. The system testing will be performed by the integration team and will cover basic functionality to ensure that the system works as expected.

4.1.4 Acceptance Testing

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing. The customers in the case of Bloomen platform are the real-world users. Therefore, the evaluation of the system will be carried out by the end-user group of the Bloomen project (compiled by the Pilots).

4.1.5 Stress/Performance Testing

Stress testing aims to evaluate how system behaves under unfavorable conditions. Testing is conducted beyond the limits of the specifications. The approach falls under the class of black box testing.

Performance testing is used to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing.

When the product will go on production and the scale of the data will grow, the resource utilization (Memory RAM and CPU) of each server will be monitored through the specific tools. For example:

Visual VM

This is a visual tool to monitor the resource consumption (RAM and CPU) of the JVM on a certain server. On our case, this might not be used due to the fact that we are adopting a clustered solution. Alternatively, for Linux machines there is Gnome System Monitor that visualizes on real time, the utilization of memory and CPU.

Apache JMeter

This powerful tool by Apache can be used after integrating all RESTful web services of the system, to measure their capacity (throughput, latency etc.). With this tool, a scenario is defined and a number of simultaneous requests are sent to the service needed. In this way, we can measure the performance and the capacity of the deployed HTTP services.

Web Performance Load Tester

Web Performance Load Tester is a tool that can be used on the UI to measure its performance. With this, it is possible to record a use case (by using a simple browser) and playback on different loads (number of users). The results are exposed in diagrams indicating certain measures over time: Number of failures, bandwidth consumption, page completion rate, page duration.

4.1.6 Penetration Testing

A penetration test which aims to:

- Identify and gain a thorough understanding of any security vulnerabilities that may be exploitable from the internet.
- Assess the possibility of gaining unauthorized access via the external network.
- Identify the appropriate technical or other measures that are required in order to eliminate or mitigate possible vulnerabilities, weaknesses or technical flaws

In the scope of Bloomen, we mostly rely on cryptographic based security provided by the blockchain implementation in addition to the JWT-powered authentication mechanism. Although we are not going to perform an intensive penetration testing we need to remain security-aware until the end of the project and to be prepared for a proper penetration testing if the product goes to commercial stage.

4.2 Testing Approach and Release Management Policy

4.2.1 Installation and Validation Instructions

When submitting a module, the module owner must provide to the integration team all the necessary instructions in order to successfully install and test the module. The instructions must be given in the format of clear steps to reproduce in order to install and verify successful installation. This might also include manual integration tests with given mock input and expected output. The module owner must provide full support to the integration team for at least one week after delivering the module, until the successful installation of it is confirmed. Only after that, the module can be considered as successfully submitted.

4.2.2 Bug Tracking - Ticketing Flow

All problems are going to be reported in a Jira ticketing system. It is important to maintain common ticketing rules between the development teams and the testing team into the Bloomen project. These rules have to be agreed during the early stages of the project and can be changed throughout the evolution of the system. The most important aspect of these rules is to maintain a common flow for the tickets. This ticket flow implies that every ticket is opened from any tester (user), developer or integrator that detects an error in application functionality.

Ticket Priority:

- Critical (priority-1): system breaks
- High (priority-2): feature not working
- Medium (priority-3): bug which doesn’t break core functionality of any feature
- Trivial (priority-4): new feature or enhancement

By default, when the ticket is opened, it is assigned to a developer with the status “new”. Then, the assigned person has to verify that the ticket is within his/her area of responsibility and accept the ticket. From this point, the developer starts investigating the solution to the problem reported. If the result of the investigation proves that the error belongs to a part of the software out of his/her responsibility, the ticket owner can re-assign it to another developer for further investigation. In the end the ticket has to be resolved, a report has to be produced and the responsible person can close it as: “fixed”, “invalid”, “no plan to fix”, “duplicate”, “cannot reproduce error” or “rejected”. This is diagrammatically described in the following UML state diagram. New feature requests (user stories) will come from and be prioritised by the product owner. From then, the planning of these new features will be handled following the process described in 3.4.

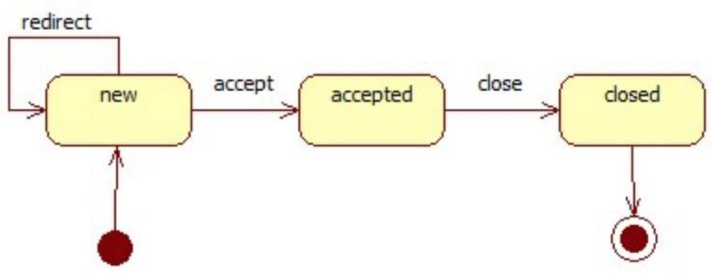


Figure 3: Ticketing Flow

4.2.3 Release Management Policy

It is essential that the whole consortium agrees on a common release management policy which covers the launching of new release. We have to declare a stable release prior to any major event (evaluation, review, milestone etc.).

Some rules could apply to changes of modules and module deliveries. For instance, a set of guidelines could be:

- 10 working days before major event (evaluation, review) we freeze development - meaning that we stop committing in mainline or delivering new versions of modules. Regarding code, any changes can be performed in code branches.

Exceptions (commits allowed only in these cases):

- Changes for tickets with priority-1 (anytime)
- Changes for tickets with priority-2 (up to 5 working days before event) with approval by project + technical coordinator.

We can call a stable release only if:

- System is up and running without any outage for at least 5 days.
- No priority-1 or priority-2 tickets exist for this release.
- No more than 5 priority-3 tickets exist for this release.

5 Conclusions-Next Steps

At this stage, we are covering the first version of Bloomen system. We are currently at a good level of integration, with several modules interacting successfully with each other in an integrated environment. In the next periods we need to focus on fully integrating all core modules and cover all business requirements of the project. Moreover, we need to focus on testing and ensure a stable and efficient system to be offered to end-users.