Project Name: Yoshi
Project Requirements

Version 3.3

Revision History

Date	Version	Description	Author
2014-11-10	1.0	Initial Draft	Martin Anev and Yuxing Chen
2014-11-25	2.0	"Yoshi Evolved" changes connected with the change of the requirements from Customer's side	Martin Anev
2014-12-01	3.0	After the unsuccessful attempts to run the "Yoshi Evolved" prototype. The requirements and requests have changed.	Martin Anev and Yuxing Chen
2014-12-03	3.1	Font and minor details edited, Section 2.2 Background description, 2.3.1 Community types, 2.3.2 Community Decision Tree and 2.3.3 Community types and Yoshi are added, Functional Requirements, Goals are modified	Martin Anev
2014-12-28	3.2	1.5 Definitions and acronyms, 2.1.1 Interfaces, 2.3 High Level Description of the domain, 2.3.1 Community types, 3.1.2 Yoshi	Martin Anev
2015-01-13	3.3	Adding list of tables and figures. Format edits.	Martin Anev

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1 Purpose of this document

The purpose of this document is to introduce general functionalities of the project delivered by Team Yoshi in the Distributed Software Development course done simultaneously in 'Politecnico di Milano' situated in Milan, Italy and 'Mälardalen University' situated in Västerås, Sweden.

1.2 Document organization

The document is organized as follows:

- Section 1, *Introduction*
- Section 2, *Overall Description*, describes different interfaces and high-level description of the domain
- Section 3, *Goals and Requirements*
- *Section 4, Use cases*

1.3 Intended Audience

The intended audience is:

- The customer of the project
- The supervisors of the project
- Yoshi Team
- All related stakeholders
- Any developer with interest to continue or improve the project

1.4 Scope

The document reveals what the project requirements are and provides background knowledge of the product on which this project is based.

1.5 Definitions and acronyms

1.5.1 Definitions

Basic definitions of the document terms are defined in Table 1.

Keyword	Definitions
Community	Social unit of any size that shares common values.
Open Source Community	Community that develops open source software.
Compute a Community	The action of observing a community, measuring metrics for Social Communities (e.g. collaboration between community members, common projects, list of followers of the members, collaborators, contributors to projects). The outcome of the action is a decision – what type is the observed community.
Visualize a community	The action of observing the output of the decision "What type is a community?". The deliverable is text and images.
Software adaptor	In software engineering, the adapter pattern is a software design pattern that allows the interface of an existing class to be used from another interface.
Eclipse plug-in	Plugins are the smallest deployable and installable software components of 'Eclipse' software developing tool.
Community Decision Tree	Decision tree defining what type is an observed social community based on some characteristics. The Decision Tree is discussed in the document 'Uncovering Latent Social Communities in Software Development'[4].

Table 1. Definitions

1.5.2 Acronyms and abbreviations

Acronym or abbreviation	Definitions
API	Application Programming Interface
G	Goal
FR	Functional Requirement
HTTPS	Hypertext Transfer Protocol Secure (communication protocol)
JSON	Java Script Object Notation (data interchange format)
OSS	Organizational Social Structures
CoP	Community of Practice
KC	Knowledge Communities
SC	Strategic Communities
IC	Informal Communities
LC	Learning Communities
PSC	Problem Solving Communities
NoP	Network of Practice
FN	Formal Networks
IN	Informal Networks
SN	Social Networks
WG	Work Groups
FG	Formal Groups
PT	Project Teams

Table 2. Acronyms and abbreviations

2. Overall description

This section describes the general factors that affect the software product and its requirements, providing background for specifying concrete requirements in the next sections of the document.

2.1 Project description

The project has two aims:

- Modify the existing "Yoshi" product.
- Develop missing components of the product.

A deliverable of the project is software product. Product's name is "Yoshi Vis" and it is dependent on "Yoshi"[1] (Eclipse plug-in), providing a visualization and computation layer on top of "Yoshi". In order to provide the adaptation, proper adaptors should be assigned to the output of "Yoshi". Design and implementation of these adaptors will also be covered during this project.

2.1.1 Interfaces

- External interfaces – The software product does not provide any external interfaces.
- User interfaces
  - Input - the software product should have graphical user interface for the input from the user. The user is expected to specify the community that should be observed.
  - Output - the software product should provide visual output in HTML page form of text and images. The user should be able to visually understand and recon characteristics of observed communities.
- Hardware interfaces – The software product does not provide any hardware interfaces.
- Software interfaces – The software product should get as input the output of "Yoshi" (which is data.out file).

2.2 Background description

Damian A. Tamburri is a Ph.D. researcher at Vrije Universiteit - Amsterdam. Damian is the customer of Yoshi project. The customer wants a software for supporting social community awareness in open-source. He provided the source code of "Yoshi". "Yoshi" has been built by Alexandra Leta, student in Information Sciences, Vrije Universiteit - Amsterdam for her graduation thesis project. The code was sent without any proper documentation. The team received only the thesis work of Alexandra Leta[5]. Yoshi, is an analytic software for open-source communities, helping different users better understanding the open-source community and getting a good understanding for research and for practice. The team for the project did not succeed in running the prototype. A meeting with the customer was scheduled, but the prototype was still not being able to run. According to the output files, the Yoshi prototype has worked before and it had already made some output in a text form (i.e. 'data.out' file).

2.2.1 Description of the "Yoshi" prototype

Based on observation of the code, "Yoshi" uncovers 6 characteristics to typify an open source-community: Structure, Situatedness, Dispersion, Informality, Formality, and Engagement, described in **part 3.2.1 Decisive characteristics**. The values should determine the community's type and represent key characteristics that describe how the community is active and carrying out its work.

To evaluate these characteristics "Yoshi" should proceed as follows:

1) Establish the value of several indicators for said characteristics.
2) Ascertains that reference thresholds are passed, thus making the characteristics explicit.
3) Based on the characteristics and following an algorithm (from [2]) "Yoshi" should determine community type for an observed community.

By team's observation and interviews with the customer, both parts deduced that the provided prototype was able to perform only part 1) in previous stable versions. From the team is not required to update the "Yoshi" prototype to stable version, but to develop parts 2) and 3) and provide output, based on the data that was computed before from stable versions of part 1). Figure 1 reveals Yoshi's software architecture using input-output control flow diagram.

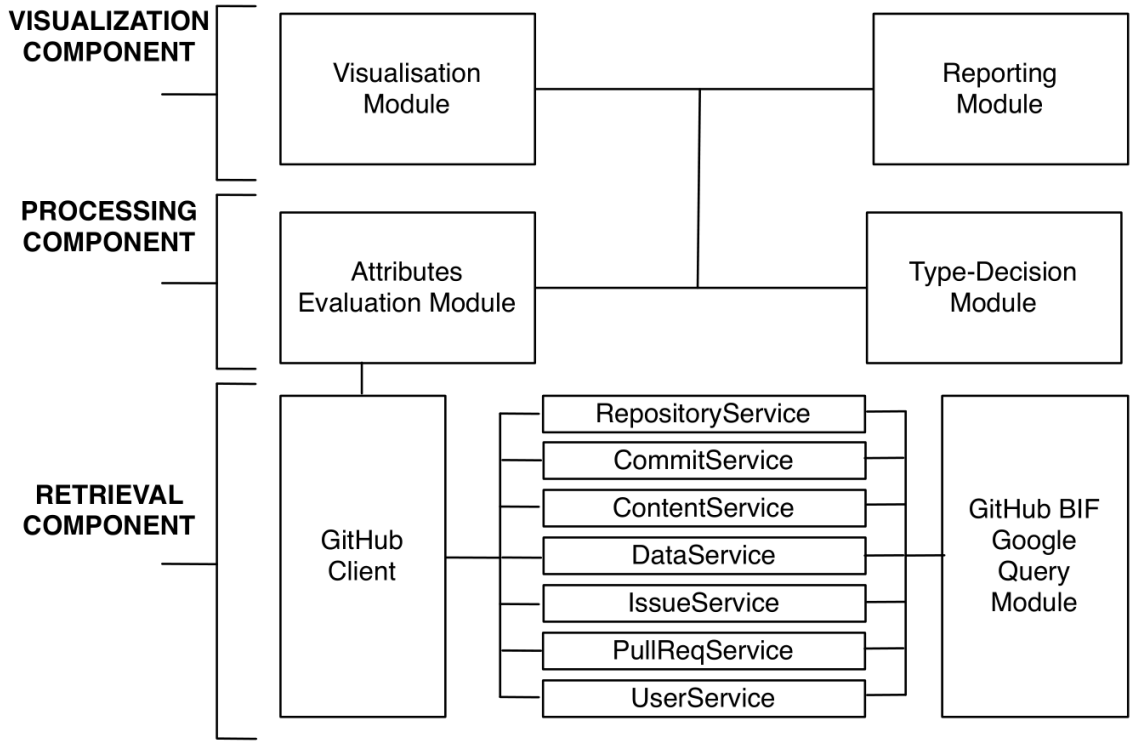

Figure 1. Yoshi high-level architecture

First, an information retrieval component is responsible for retrieving data with which YOSHI can perform its functions. This component retrieves data from two data sources: source code management systems and issue trackers from GitHub. The service classes (see middle part of Fig. 1)[1] provided by this API are used for retrieving information about the file structure of the repository, the members contributing to the project, issues associated to a repositories and their current status.
On top of the first layer the customer requires to be build the processing and visualization component. In order to achieve this, proper formatting of the current output should be done.

2.3 High level description of the domain

The software product that will be delivered is “Yoshi Vis”. “Yoshi vis” is containing the Processing and Visualization layer of the “Yoshi” project. “Yoshi” and “Yoshi Vis” mutual coexistence intend to help people to measure, compute and study social community types by visualizing metrics and characteristics of the observed community. For instance, it will allow users to measure social and organizational characteristics of open-source communities (e.g. structure, formality, informality) measured by different metrics (e.g. collaboration between community members, common projects, list of followers of the members, collaborators, contributors to projects). The possible types of communities are: Community of Practice (CoP); Knowledge Communities (KC); Strategic Communities (SC); Informal Communities (IC); Learning Communities (LC); Problem Solving Communities (PSC); Network Of Practice (NoP); Formal Networks (FN); Informal Networks (IN); Social Networks (SN); Work Groups (WG); Formal Groups (FG); Project Teams (PT). [3] For more information on the domain, refer to the Project Requirements Document [6] and the references [2][3][4].

2.3.1 Community Types

In this section are presented (quoted and paraphrased from [3]) key characteristics of the different types of communities:

2.3.1.1 Communities

All communities are made for sharing. For example, in Communities of Practice (CoPs), people sit in the same location, sharing a practice or passion. In Strategic Communities (SCs), people share experience and knowledge with the aim of achieving strategic business advantage for the corporate sponsors. Here follow detailed definitions.

- Community of Practice (CoP) – The key differentiating attribute for CoP is situated sharing, or learning/sharing of a common practice (i.e., the attribute "situatedness" is a differentiator to identify a CoP). A CoP consists of groups of people who share a concern, a set of problems, or a passion about a topic, and who deepen their knowledge and expertise in this area by interacting frequently and in the same geolocation. As such, CoPs serve as scaf-folding for organizational learning in one specific practice
- Knowledge Communities (KC). The key differentiating attribute for KC is the visibility of its contents and results produced (i.e., the attribute "Visibility" is a differentiator to identify a KC). Essentially KCs are groups of people with a shared passion to create, use, and share new knowledge for tangible business purposes (e.g.,increased sales, increased product offer, clients profiling, etc.). The main difference with other types is that KCs are expected (by the corporate sponsors) to produce actionable knowledge (knowledge which can be put to immediate action, e.g., best practices, standards, methodologies, approaches, problem-solving-patterns, etc.) into a specific business area.
- Strategic Communities (SC). The key differentiating attribute for SCs is the contract value (i.e., the attribute "Contract Value" is a discriminator to identify an SC) that should be maintained (e.g., by generating ad hoc best practices) or generated (e.g., by analysing strategic market sections ripe for growth). In software engineering, for example, SCs are commonly associated with mission-critical systems that need 24/7 availability. Strategic communities are formalized structures that consist usually of a limited number of experts within a single organization. These share a common, work related interest into producing unexpected ideas to achieve strategic advantage. These communities are intentionally created by the organization to achieve certain business goals
- Informal Communities (IC). The key differentiating attribute for ICs is the high degree of member engagement (i.e., the attribute "Member Engagement" is a discriminator to identify an IC). An example in software engineering can be seen in the Agile Movement, whose success is decided explicitly by the contributions of its members (i.e., their engagement in actively disseminating and using "agile" practices). ICs are usually sets of people part of an organization, with a common interest, often closely dependent on their practice. They interact informally, usually across unbound distances, frequently on a common history or culture (e.g., shared ideas, experience etc.). The main difference they have with all communities (with the exception of NoPs) is that their localization is necessarily dispersed so that the community can reach a wider "audience".
- Learning Communities (LC). What differentiates this particular type from others is its explicit goal of incrementing, tracking and maintaining the organizational culture of an organization (the attribute "Organizational Culture" is differentiator when identifying an LC). The nature of discussed topics makes it extremely different from other communities: most other communities have either personal or organizational goals, while LCs have both and pursue them equally.
- Problem Solving Communities (PSC). The key differentiating attribute for PSCs is their goal, that of solving a specific problem in the scope of an organization or corporate sponsor (i.e., the attribute "Organizational Goal" is a differentiator for PSCs). The problem-solving network is a distributed network of practice which meets the criteria of an expert group. The network provides resources in terms of help-desk functions where participants of the network support other colleagues by giving them special advice as regards particular business problems. In addition, participating in this kind of network ensures collaborative learning among the participants of the network.

2.3.1.2 Networks

Networks suggest the presence of digital or technological support tools, however, we found this explicitly only for NoPs. All network types are used by an organization to increase reachability, either through formal means (in FNs) or through informal ones (in INs) or through customized forms of boundary spanning (e.g., in NoPs). Here follow detailed definitions.

- Networks of Practice(NoP). The one differentiating attribute for NoPs is informality in geolocalized practice (i.e., the attribute "geodispersion" is a differentiator to identify an NoP). In global software engineering many (virtual) teams collaborate together through the Internet across time zones, with specific networks (e.g., Virtual Private Networks) and with strong management and governance policies in place.

NoP comprises a larger, geographically dispersed group of participants engaged in a shared practice or common topic of interest [...] CoPs and NoPs share the characteristics of being emergent and self-organizing, and the participants create communication linkages inside and between organizations that provide an "invisible" net existing beside the formal organizational hierarchy.

- Informal Networks (IN). The key differentiating attribute for INs is the type of interaction that binds its members (i.e., the attribute "members interaction" is a differentiator to identify an IN). Informal networks have existed in software engineering since its very beginning: all people participating within any software process collaborate and interoperate within a web of social ties that could be defined as an IN. As compared to the other types, INs can be seen as looser networks of ties between individuals that happen to come in contact in the same context. The driving force of the informal network is the strength of these ties between members.
- Formal Networks (FN). In formal networks memberships and interaction dynamics are explicitly "made" formal by corporate sponsors (i.e., the attribute "Membership Official Status" is differentiator when identifying an FN). Conversely, although formally acknowledged by corporate sponsors, FGs are (commonly) informal in nature, and are grouped for governance or action. Within FNs, members are rigorously selected and prescribed. They are forcibly acknowledged by management of the network itself. Direction is carried out according to corporate strategy and its mission is to follow this strategy.
- Social Networks (SN). In social sciences, the concept of social networks is often used interchangeably with OSSs. Since every OSS type can be defined in terms of SNs, there is no distinctive difference with other types; rather, SNs can be seen as a super-type for all OSSs. "To identify the presence of an SN (or OSS) it is sufficient to be able to split the structure of an observable set of organizational (i.e., structure of social ties and interactions in the large [Johnsen 1985]) and microstructure (i.e., structure of social ties and interactions at the single social-agent level [Fershtman and Gandal 2008])"

2.3.1.3 Groups

Groups are tightly knit sets of people or agencies that pursue an organizational goal. Cohesion in groups can be an activation mechanism (to increase engagement, as in WGs) or a way to govern people more efficiently (e.g., in FGs). Here follow detailed definitions

- Workgroups (WG). The key differentiating attribute for WGs is the cohesion of their members: they need to work in a tightly bound and enthusiastic manner to ensure success of the WG (i.e., the attribute "Members Cohesion" is a differentiator to identify a WG). Differentiating it from other types is its level of granularity: PTs act on specific projects (i.e., domain-specific, well-specified problems, with a clear set of goals and complementary sets of skills) while KCs focus on specific business areas and goals. A WG has a wider agenda and uses experts with similar skills and interests to tackle strategic issues
- Formal Groups (FG). The key differentiating attribute for FGs is in their governance practices, which must be declared upon creation of the formal group (i.e., the attribute "Governance" is determinant to identify this type). Literature refers explicitly to formal groups as sets of project teams with a particular mission (or also, groups of people from which project teams are picked). FGs are comprised of people which are explicitly grouped by corporations to act on (or by means of) them (e.g., governing employees or ease their job or practice by grouping them in areas of interest). Each group has a single organizational goal, called mission (governing boards are groups of executives whose mission is to devise and apply governance practices successfully). In comparison to Formal Networks, they seldom rely on networking technologies, on the contrary, they are local in nature.

2.3.1.4 Teams

Teams are specifically assembled sets of people with a diversified and complementary set of skills. They always pursue an organizational goal with clear-cut procedures and activities. Finally, all project teams exhibit a longevity which is bound to a project or product. Here follow detailed definitions.

- Project Teams - the key differentiating attribute for PTs is their longevity, tied to a specific project (i.e., the attribute "longevity" is a differentiator to identify PTs). PTs are temporary organizations or project groups within firms [that] consist of people, most of whom have not met before, who have to engage in swift socialization and carry out a pre-specified task within set limits as to time and costs. Moreover, they comprise a mix of individuals with highly specialized competence making it difficult to establish shared understandings or a common knowledge base. PTs are made by people with complementary skills who work together to achieve a common purpose for which they are accountable.

2.3.2 Decisive Characteristics

The followings are six key organisational, operational and socio-technical characteristics for open-source communities: **Structure** (Is there a non-trivial structure within the observed people?), **Situatedness** (Do all community members exhibit the same location?), **Dispersion** (Do all community members exhibit a different location?), **Informality** (Do community members require informal status evaluation?), **Formality** (Do all community members require formal status evaluation? – mutually exclusive with Informality), **Engagement** (Does the community effectiveness depend on people' engagement?). How the characteristics are represented by the observed metrics is an issue to be resolved. A Community Decision Tree has been defined in [4] and is represented on Figure 2. For Open-source communities (and respectfully for the project) is related only the right-hand side of the tree, divided by Dispersion (i.e. CoP, IN,FN, NoP, IC).

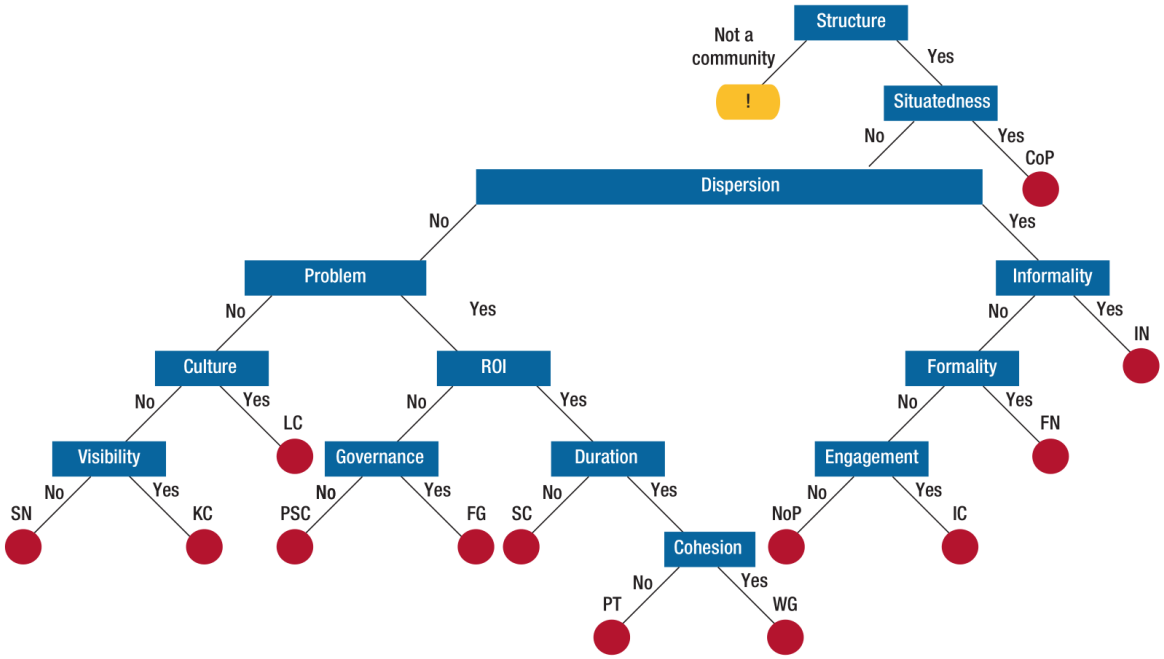

Figure 2. Community Decision Tree

2.3.3 Community Types and Yoshi

The "Yoshi" prototype has no documentation that could be appended to this document. Ideas of how it works and which communities it could measure should be discovered in the process of work, by its code and behavior. Up to date neither the team nor the client succeed in running the "Yoshi" prototype, "Yoshi Vis" is extending the output of previous stable versions of "Yoshi" as well as future versions that could provide the same output.
From theoretical point of view every community shall be uniquely defined by the algorithm (representing the Community Decision Tree – Figure 2) [2], but in practice the customer said "It is normal that a structured set of people (i.e. community) could blend in more than one types. To establish this, mind the precedence of the characteristics." The algorithm used for the project, presented in **part 3.1.2**, has been done together with the customer and it defines only the right part of the Community Decision Tree divided by the attribute Dispersion. The reason for this limitation is the data provided from "Yoshi" prototype. "Yoshi" and "Yoshi Vis" can establish the following types: CoP, IN,FN, NoP, IC.

2.3.4 General functionalities

"Yoshi" and "Yoshi Vis" should provide general functionalities for managing:

- Retrieval
  "Yoshi" is responsible for connecting to social communities and to gather metrics from their repositories. This component retrieves data from two data sources: source code management systems and issue trackers.
- Processing
  "Yoshi Vis" is responsible for evaluating metrics using data available from the retrieval component and compute community typification [3] by matching the type with the decisive attributes.
- Visualization
  "Yoshi Vis" is responsible for visualizing the metrics (using a text form) and the community type on a decision tree. [4]

3. Goals and Requirements

This section describes the goals and functional requirements that affect the software, providing background for specifying concrete use cases which are presented in the next part.

3.1 Goals

In this section are discussed the general goals of the project, the goals that the project has for modifying 'Yoshi' and the goals that the project has for building 'Yoshi Vis'

3.1.1 General

The following are the general goals of both "Yoshi" and "Yoshi Vis" components.
[G0] Allow users to observe community by gathering organizational characteristics.
[G1] Allow users to measure community social and organizational characteristics.
[G2] Allow users to study type and characteristics against performance metrics.
[G3] Allow users to understand the characteristics of an observed community.

3.1.1.1 Yoshi

The following are the project's goals for modifying "Yoshi".
[G4] Format the current output of "Yoshi".

3.1.1.2 Yoshi Vis

The following are the project's goals for building "Yoshi Vis".
[G5] Define the type of the observed community based on the metrics.
[G6] Provide output with the defined social types and motivate the decision.

3.1.1.3 Description of the Goals

- [G0] Allow users to observe community by gathering organizational characteristics. This is a general goal for "Yoshi" and "Yoshi Vis". As "Yoshi" is not in a stable version, it cannot achieve the goal within the domain of the project. The project's aim is not to achieve [G0], but to fulfil the functional requirements, which will lead to fulfilments of the other Goals (G[1],...,G[6]). The team is not required to update "Yoshi" to runnable version.

- **[G1]** Allow users to measure community social and organizational characteristics. This goal is one of the main purposes of the project. The users should be able to measure the organizational characteristics gathered from the retrieval component.

- **[G2]** Allow users to study type and characteristics against performance metrics. The users should be able to understand what are the correlations between the type of the observed community and the gathered metrics.

- **[G3]** Allow users to understand the characteristics of an observed community.
  The characteristics of the observed community should be expressed in a comprehensible way, every value should be mattered to its metric.

- **[G4]** Format the current output.
  As the current output is in violation with Goal 3, the customer requires that the output should be changed, in order to be able to understand what the "Yoshi" prototype is giving as output.

- **[G5]** Define the type of the observed community based on the metrics. This is the essential part of the system. This goal defines the purpose of the system.

3.2 Requirements

In this section are discussed the functional requirements of the project. Functional requirements for the project related to "Yoshi"

3.2.1.1 Functional requirements for the project related to "Yoshi"

[FR1] The current output of Yoshi should be prepared for adaptation.

3.2.1.2 Functional requirements for the project related to "Yoshi Vis"

[FR2] The product should be able to provide proper adapter for the formatted output of Yoshi.
[FR3] The product should be able to take decision what type is an observed social community based on the received metrics from "Yoshi".
[FR4] The product should be able to visualize as output the formatted output of Yoshi.
[FR5] The product should be able to visualize as output the evaluated decision of the community type in text form.
[FR6] The product should be able to visualize as output motivation of the taken decision.
[FR7] The product should be able to visualize as output the community type on the Community Decision Tree in graphic form.

3.2.1.3 Description of the Functional Requirements

- **[FR1]** The current output of Yoshi should be prepared for adaptation.

This FR should be fulfilled, in order to satisfy G4. The output should be re-formatted in JSON standards format.

- **[FR2]** The product should be able to provide proper adapter for the formatted output of Yoshi.

This FR should be fulfilled, in order to satisfy G4, G5. The input of the "Yoshi Vis" should follow the JSON standards, in order to be able to connect to the output of "Yoshi"

- **[FR3]** The product should be able to take decision what type is an observed social community based on the received metrics from "Yoshi".

This FR should be fulfilled, in order to satisfy G5. The type should be at least one from part 2.3.1 Community Types.

- **[FR4]** The product should be able to visualize as output the formatted output of Yoshi.

This FR should be fulfilled, in order to satisfy G1,G2, G3.

- **[FR5]** The product should be able to visualize as output the evaluated decision of the type in text form.

This FR should be fulfilled, in order to satisfy G3, G5.

- **[FR6]** The product should be able to visualize as output motivation of the taken decision.

This FR should be fulfilled, in order to satisfy G2.

- **[FR7]** The product should be able to visualize as output the type on the Community Decision Tree in graphic form.

This FR should be fulfilled, in order to satisfy G3, G5.

4. Use Cases

4.1 Actors

4.1.1 User

A person that has wants to examine, measure, compute (define) and study a social community. “Yoshi vis” is an abstractive upper layer of “Yoshi”. Hence all the actors of “Yoshi” are actors of “Yoshi Vis”.

4.2 Use cases

4.2.1 Use case diagram

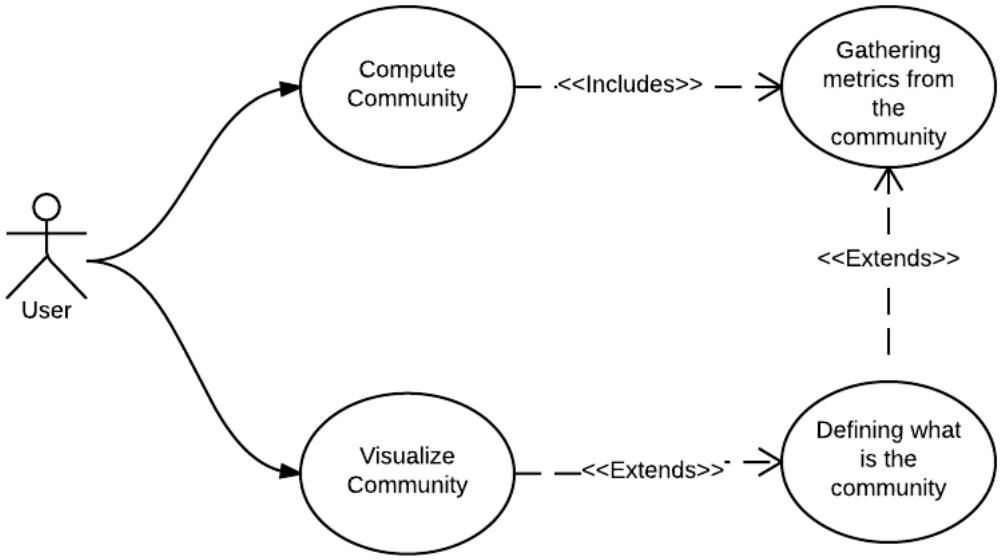

Figure 3. General Use Case Diagram

4.2.1.1 Description of the general use-case

- A user can use “Yoshi vis” to compute the community by gathering the metrics.
  A user wants to find out how the characteristics of the community related to his software development. The “Yoshi vis” system is holding some data of community’s characteristics and can use different kinds of metrics to compute the community. The user opens the “Yoshi vis” system to compute the community. In the end, the user gathers the computation result.

- A user can use “Yoshi vis” to visualize community as well as define the type of the community by gathering the metrics
  A user wants to find out how the characteristics of the community related to his software development and finds that s\he’s not able to understand the digit result. The “Yoshi vis” system is holding some data of community’s characteristics and can use different kinds of metrics to compute the community. This user opens the “Yoshi vis” system to compute the community and visualize the result so that s/he can intuitively understand the characteristics of the community.

4.2.1.2 Description of use-case

In this section is discussed the detailed description with sequence diagram to the various use-cases related to the main use-case diagram defined above.
For simplicity and readability, here are presented sequence diagram of the more complex use cases and the more simple ones are only described in textual form.

Name	Compute the community
Actors	User
Entry conditions	The user select specific community to compute
Flow of events	1. The user starts the system 2. System provides choices for community 3. The user selects the community to compute 4. The user clicks “compute community” 5. The system shows the result
Exit conditions	The user quits the activity
Exceptions	No exception

Table 3. Use Case: Compute the community

Name	Visualize computed characteristics
Actors	User
Entry conditions	After the system computed the system shows the result page
Flow of events	1. The user starts the system 2. System provides choices for visualizing already computed results 3. The user selects the desired option 3.1 The user clicks “visualize the result” 4. The system visualizes the result
Exit conditions	The user quits the activity
Exceptions	No exception

Table 4. Use Case: Visualize computed characteristics

References:

[1] Yoshi - https://github.com/maelstromdat/YOSHI

[2] '*"Let me measure my self!" Said Open-Source'* - D.A. Tamburri, E. di Nitto, P.Lago

[3] 'Organizational Social Structures for Software Engineering' - D.A. Tamburri, P. Lago, H.V. Vliet

[4] 'Uncovering Latent Social Communities in Software Development' – D.A. Tamburri, P. Lago, H.V. Vliet

[5] 'Discovering Open-Source Community Types: An Automated Approach' – A. Leta