Demoiselle Signer: an Analysis from Fogel FLOSS Community Framework

Flávio Gomes da Silva Lisboa
UTPFR
Curitiba, Brazil
https://orcid.org/0000-0002-9396-7944

***Abstract*—This paper aimed to make an analysis of community maturity for Demoiselle Signer, a FLOSS produced by Serviço Federal de Processamento de Dados (Serpro). This research was motivated by criticisms to FLOSS production by Brazilian government entities after a period of support policies. The challenge of this paper was to make a community maturity analysis without an appropriated tool to do it. Then, it was necessary to define a community maturity framework for FLOSS project based on Fogel's guidelines to create a successful FLOSS project. From what we have named as Fogel's Framework and with the aid of software called RepoSense, we have analyzed the community maturity of Demoiselle Signer. As it is possible to notice along the paper, the analysis of members is the initial and essential part for investigating the community maturity. From a scale ranging from 0 (zero) to 10, we have determined that Demoiselle Signer has a community maturity equals to 4. This value indicates that Demoiselle Signer is an active community and have some items pointed out by Fogel as desirable features of a FLOSS community infrastructure.**

***Keywords*—capability maturity model; collaborative software; digital signatures; open source software.**

Introduction

FLOSS stands for Free\Libre and Open Source Software. The theme of this paper is the FLOSS community maturity (FCM) and the question of this paper is about the FCM of an specific FLOSS project, the Demoiselle Signer, a brazilian software for generating and validating digital signatures. Demoiselle Signer is one of few FLOSS produced by Serviço Federal de Processamento de Dados (Serpro), the biggest brazilian information technology state-owned company [1].

Oram [2] criticizes the results of more one than decade of FLOSS Brazilian support from government. He said that "results are disappointing". Really, considering the FLOSS production by Brazilian government, few software were produced and for this little production seems there are community around them. This assumption, about lack of community activity, needs to be verified. In this paper, we investigate the community around Demoiselle Signer.

First we introduce the FCM subject and next we talk about Demoiselle Signer.

A. FLOSS Community Maturity

Taurion [3] states that "not every free software project will be successful. Many do not attract community interest and tend to disappear" (our translation). There are some free and open source software projects maintained by only one person and another ones maintained by groups – or communities. For the first case, it is easier to finish the project, because the decision depends only on the creator. For the second case, it is harder to finish the project, because, according to Fogel [4], "as long as there are people somewhere — anywhere — interested in continuing it, it can never be unilaterally shut down". In fact, Fogel considers that even the project maintained by one only developer can survive because if it is FLOSS, its source code must be available and anybody interested to continue the project can do it.

Fogel [4] presents a guide to what he calls a "successful open source project" (he considers open source and free softwares as synonyms). His guide is based on observations and in his own personal experience with several FLOSS projects, mainly for Subversion, to which Fogel was fully dedicated along seven years. "Success" for Fogel is not an indicator that a FLOSS project has won a battle against other project. Indeed, Fogel states that "In the long run, every successful project contributes to the well-being of the overall, worldwide body of free software". Maybe we can consider that a successful FLOSS project in the fogelian concept is a project that keeps being useful for their users and contributors.

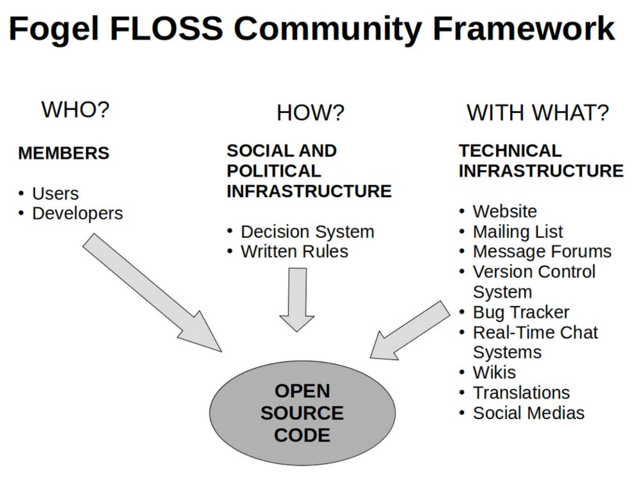

Fig. 1. Fogel FLOSS Community Framework [4].

The Fogel's guide seems to be a FLOSS Community Framework (FCM). We understand FCM as an instrument to measure the maturity of a FLOSS project. Figure 1 shows the three main aspects of a FLOSS community, according to Fogel: the members, the social and political infrastructure and the technical infrastructure. All of this is reunited around the source code, the commons for users and developers.

Fogel's Framework can be used to check a FLOSS project has a matured community. We can understand a matured FLOSS community as an active community around their project source code. The proposal of using Fogel's guide as a FLOSS framework is justified by the absence of such a framework. We should not confuse a community maturity model with a process maturity model related to FLOSS. This last type is already dealed, for example, by Pinho [5], who approaches the process of a company to absorbe FLOSS tools for its development lifecycle. Oram [6] also approaches the internalization of FLOSS skills by companies. We also can observe that huge information technology companies, like Amazon, Facebook, Google, Microsoft and Oracle, have internalized FLOSS products and processes, but they keep being companies, not communities.

In August 2022, we have searched for this specific term "FLOSS Community Framework" in some journal databases. We searched in ACM Digital Library, Scielo and CAPES journal database. None of these databases presented results for articles with the searched term in paper title or in the publication title. Of course, these databases there are many research papers with the terms "open source" or "free software", but the search was done for "FLOSS", understanding that both of communities must be considered.

There are also the term "FOSS" (Free and Open Source Software), but Stallman [7] criticizes this acronym because it does not explain that free is about freedom and it emphasizes the term "open source". FLOSS is a clearer acronym for making reference to software projects built by communities organized both around Free Software Foundation concepts and Open Source Initiave concepts. We don't want to discuss the differences between them here.

As we are proposing a framework to measure the maturity of a FLOSS project – or better, the maturity of a community round a FLOSS project, we need a measurement scale. There is no community without members, so this item is required. It doesn't matter how many members are users or developers. It is more important to check the releases of software. If there are recent releases, there are activity. There are people developing and delivering something for someone else to use. As one of the Agile Manifesto principles states: "Working software is the primary measure of progress" [8]. Then we considered the date of releases as a component of our measurement scale. We are building the scale and we don't have previous references, so we are defining a simple criterious which can be reviewed in future works. An active community, for the purpose of analysis, is one that have a release of its software in the current year.

Activity will have as value only zero or one. There is recent activity or not. We can stablish that each one of the items of technical infrastructure can have as value zero or one too. The two items of social and political infrastructure could be dealed as a single item, because it is use having a decision system if the rules are not documented for all community to know. So, we have ten items (Figure 1) worthing zero or one.

From these definitions, we can propose a measurement scale ranging from zero to ten, expressed for CM, as follows:

$$CM = a * \sum_{i=0}^{9} t_i, a, i \in \mathbb{Z}, a=0 \vee a=1$$

Where a is the community activity and t_i is each one of the infrastructure items. An inactive community has zero maturity. An active community has maturity from one to ten.

This is a simple measure, which allows comparisons among different projects regarding to recent activity and the social, political and technical infrastructure items available. The precision of this comparison can be evolved, with a study about each one of this items and its weight in the community maintenance. But for now, it is enough for starting an evaluation, in absence of other frameworks as references.

B. *Demoiselle Signer*

Before talking about Demoiselle Signer, we need to talk briefly about Demoiselle Framework [9]. Demoiselle Frameworks is presented as an "integrator framework" in its homepage (www.frameworkdemoiselle.gov.br). In fact, Demoiselle Framework is a reusable components collection for building Java Enterprise Edition (JEE) applications. JEE [10] is focused on building Java applications for network environments. Demoiselle Framework homepage brings not only information about framework, but other two products: Demoiselle Signer and Demoiselle Behave. The first is presented as a digital certification component for ICP-BRASIL (Brazilian Public Key Infrastructure). The second is presented as a tool for automated functional tests.

The specific page of Demoiselle Signer (Figure 2) presents its features:

- Generation of digital signatures (CAdES and PadES patterns);
- Validation of digital signatures;
- Assembly and validation of ICP-BRASIL chains;
- Time stamp;
- Cryptography.

The page already brings links to documentation, issue tracker and source code. The documentation is signed by ten contributors and it is very detailed. Github hosts the source code repository and provides the issue tracker for Demoiselle Signer. Demoiselle Signer, as well as Demoiselle Framework, is implemented in Java language.

DEMOISELLE SIGNER

É um componente que provê as funcionalidades de assinatura digital e manipulação de certificados nos padrões da ICP-Brasil.

Demoiselle **Signer**

Versão Final 4.0.0 !

Objetivos:

Prover as funcionalidade necessárias para geração e validação de assinaturas digitais no Padrão Brasileiro
O Padrão Brasileiro para Assinaturas digitais (PBAD) é definido pela ICP-BRASIL

Os Requisitos das Políticas de Assinatura na ICP-Brasil (DOC-ICP-15.03) definem o formato, estrutura e sintaxes que devem ser observadas para a criação de novas políticas de assinatura digital.

Fig. 2. Demoiselle Signer page top.

Demoiselle Signer is used by Assinador Serpro [11], a Java desktop application for signing documents with digital certificate or validating digital signed documents. Assinador Serpro uses Demoiselle Signer, but it is not a FLOSS. There are available free binaries for Assinador Serpro in Serpro website, however the source code is not open.

This was a general view about Demoiselle Signer. In August 2022, we have searched for term "Demoiselle Signer" in the same journal databases where we have searched for FCM: ACM Digital Library, Scielo and CAPES journal database. None of these databases presented results for articles with "Demoiselle Signer" in paper title or in the publication title. So, it seems that a scientific analysis about Demoiselle Signer seems unprecedented until now.

Next we will present the methodology used to analyze the community maturity of Demoiselle Signer.

METHODOLOGY

It is important to remember that we are analyzing the community maturity of Demoiselle Signer. We don't want to discuss the software architecture or the used algorithms. We used Fogel's Framework (FF), presented in introduction, as reference guide to investigate the source code repository and issue tracker of Demoiselle Signer. We made a checklist of FF items and did a descriptive analysis of each item.

We got information from two tools: Insights, available for any project in Github, and RepoSense, a report generator written in Java for viewing activities in git repositories.

Demoiselle Signer Insights page is available in https://github.com/demoiselle/signer/pulse. This tool has information about contributors, community standards, commits, code frequency, software dependencies, branches and forks development. The problem with this page is that it brings data from up a month ago only.

Github provides an API for handling repositories through applications, but it requires implementation or the use of an application which already accesses Github API. For our research, we used a tool named RepoSense. RepoSense is a Java application which analyzes git repositories and generate reports. RepoSense is a FLOSS, distributed under the MIT license. RepoSense documentation is available at https://reposense.org.

We wanted to analyze the activity of Demoiselle Signer since the development start. In Github Insights page, the Contributors item brings an initial date, but it is the creation of repository. We wanted to know when the developers started to push code to repository. For this, we needed to discover the first commit date. First, we recovered the first commit hash with the following command:

```
git rev-list --max-parents=0 HEAD
```

With the hash, we ran other command to get the commit date:

```
git log –stat [commit hash]
```

So, we found that first Demoiselle Signer commit was made on 8th November, 2016. From this information, we ran RepoSense from this date until August 30th 2022. The command executed was this:

```
java -jar RepoSense.jar --repos https://github.com/demoiselle/signer.git --since 8/11/2016 --until 30/08/2022
```

RepoSense has several optional parameters, detailed in its documentation. The parameters --since and --until allow, respectively, to set the initial and final date to be analyzed. The default behavior is to create a report inside a folder named reposense-report, but it is possible to change the output folder with parameter –output.

RepoSense generates a HTML page which can be opened in any web browser. Next, we will discuss the results obtained from this report. We have divided the result presentation in two sections, the first about members and the second about the infrastructures.

MEMBER RESULTS

Before presenting the results about members, it is appropriate to talk about the concept of member for the FLOSS context. Members of a FLOSS comunity can be users or developers. One person can be user and developer of a software at the same time. In a FLOSS project, a developer could be named as a contributor, because the work, for someone interested in the software, is volunteer, not required. Of course, if the software was created by a company, such software should there be full time developers working on it and their work should be required.

Users can install and use freely a FLOSS, but they can't require changes except under the condition of customers. Users can ask for changes, but the developers are not required to make them. Community users need to convince developers that a change is important for them work on it. In most cases, bugs and vulnerabilities reports are quickly executed. New features, however, need be negotiated. If a user wants that a change be made quickly, it is better to hire a developer or company to make it.

It is difficult to identify users of a FLOSS. One essential point of a FLOSS is the freedom of user. There is no control about the use of instances of a FLOSS, as there is for proprietary software, whose companies control numbers of copies for their customers. A way to estimate the number of users is the number of downloads of the software. Github API provides this information, but only for the software owners. It seems that this information is not important for Demoiselle Signer owners, because it doesn't appear in the project website or Github repository page. We requested this information by opening an issue, but nobody answered.

Assinador Serpro seems to be a very useful software for Brazilian users interested to sign documents digitally. It could be used for estimating the use of Demoiselle Signer. However, its page also doesn't present any download amount information. There are clues of Assinador Serpro use from search engines. From a search by Google, we found references about Assinador Serpro by Administration Secretary of Bahia State government, Espírito Santo State Court of Justice and Santa Catarina State Court of Justice. But it's accidental information, from mentions on support pages.

Although it is hard to know exactly how many users Demoiselle Signer community has, we can know some of them, excluding the developers from the list of issue authors. Then, we need first to discover the Demoiselle Signer developers. Github informs that there are 13 contributors. Table I shows the commit amount of each contributor and the proportion to total number of commits. We named the contributors as core developers and we will explain the reason next. But it is important to notice that more than half of commits were made by only two developers: esaito and juliancesar.

TABLE I
COMMITS BY CORE DEVELOPER IN MASTER BRANCH

Core Developer	Commits	% master
esaito	360	47.43
juliancesar	119	15.68
joserenecampa	17	2.24
kyriosdata	13	1.71
FabianoK	10	1.32
laubstein	5	0.66
botelhojp	5	0.66
dependabot [bot]	3	0.40
crivano	3	0.40
denisfalqueto	3	0.40
renatodantas	1	0.13
monticelli91	1	0.13
IgorMartinez	1	0.13
Total	**541**	**71.28**

It is important to clarify that there is bot (dependabot) counted as a core developer. This bot updates the Demoiselle Signer dependencies based on the POM file, a configuration file of Maven, a Java dependency manager.

By proximity of names, we can identify that 6 of 10 documentation authors are core developers.

It is interesting to observe a divergence between total of commits from core developers and the total number of commits (for branch master) in August 30th 2022. The first number is 541 and the second is 759. Where are the developers of the other 218 commits?

One answer possible is that the other developers are authors of pull requests. When we search for pull requests, we discover that there is code from other people, beyond these 13 people who Github calls contributors. For example, in 24th June, 2022 there is a commit by pramimpo user who is not a project contributor. Actually, what Github calls contributor is a core developer, someone who can integrate changes to repository. For a FLOSS community, everyone who contributes to software with an artifact is a developer, but for Github the word contributor is restricted to core developer.

However, we couldn't find all the non-core developers. From the pull requests list we got to find three developers (Table II).

TABLE II
COMMITS BY NOT CORE DEVELOPERS IN MASTER BRANCH

Pull Requester (Non-core Developer)	Commits	% master
pramimpo	1	0.13
tuliomoreira77	3	0.40
seuerick	1	0.13
Total	**4**	**0.66**

When we analyze the commits, we can find sometimes commits from users who don't have accounts in Github. Maybe these users have removed their accounts. So, we don't have information about all the Demoiselle Signer developers, including the past developers.

The number of commits indicate how many times a developer has changed a software. But it does not provide the amplitude of changes. RepoSense reports provided this information. So we got the numbers of changed lines of Table III. We kept the core developers in the same order of Table II to show that lines of code are an independent variable of the number of commits. RepoSense also showed some developers that Github Insight doesn´t show. They were added to end of the table. So, we have 16 people who worked as developers for Demoiselle Signer (we are excluding the bot).

TABLE III
LINES OF CODE BY DEVELOPER IN MASTER BRANCH

Developer	Lines of Code	% master
esaito	48452	79.09
juliancesar	5589	9.12
joserenecampa	250	0.41
kyriosdata	4495	7.34
FabianoK	980	1.60
laubstein	181	0.30
botelhojp	19	0.03
dependabot [bot]	2	0.00
crivano	258	0.42
denisfalqueto	288	0.47
renatodantas	1	0.00
monticelli91	1	0.00
IgorMartinez	1	0.00
80621732915	720	1.18
assinador	7	0.01
erick	14	0.02
Fábio Nogueira de Lucena	3	0.00
Total	**61260**	**100.00**

You can notice that proportion of changes by esaito in comparison with the second bigger contributor is greater than proportion of commits. The esaito developer is responsible for almost 80% of changed lines. The kyriosdata developer, who is the fourth is number of commits, is the third in changed lines. On other hand, the joserenecampa user, who is the third in number of commits is the eighth in changed lines.

By proximity of names, we can identify that 2 of the additional developers (4 in total) are documentation authors (erick and Fábio Nogueira de Lucena). So, we have 8 documentation authors who are developers too and 2 documentation authors who didn't contribute with code. But, according to Pressman and Maxim [12], documentation is also software. So, we can consider that we have 18 software developers in Demoiselle Signer community (16 source code developers and 2 documenters).

Since we know the developers, we can analyze the Github issue tracker and select issue authors who are not developers. Doing this, we found 46 users who submitted 54 issues from 327. 84% of the issues were opened by developers. From 54 issues by users, 40 were closed until August 30th 2022.

Most of the users opened only one issue. The most active users in issue submission, with more than one submitted issue, are showed in Table IV.

TABLE IV
MOST ACTIVE USERS IN ISSUE TRACKER

User	Open issues	Closed Issues	Total
bcfreitas		3	3
estevaocm		3	3
fbtlopes		2	2
HelloWar75	3		3
lferreirad1		2	2

So we have discovered that Demoiselle Signer has 18 developers, of whom 12 are active (they appear in Github homepage), and at least 46 users. Since we have concluded the presentation of member data, next we will present the results about the infrastructures of Demoiselle Signer.

Infrastructures Results

A. *Social and Political Infrastructure*

This item is composed by decision system and written rules. The Demoiselle Signer page in Demoiselle Framework website has information about the use of component, about how to report a bug and how to contribute with code. But there is no information about how the decisions are made. It is not explicit if Demoiselle Signer governance is made by a benevolent dictator or through consensus-based democracy [4].

However, it is possible to notice that esaito developer has protagonism in opening issues and commiting changes. The esaito's behavior suggests that this person is a benevolent dictator. But it lacks a documentation that let it clear.

B. *Technical Infrastructure*

Demoiselle Signer has a website, or better, has a page inside a website. This page has a link for user documentation and basic guidelines for contributors.

We have not found references about a mailing list in Demoiselle Signer page or in the Github source code repository.

We have not found references about forums in Demoiselle Signer page or in the Github source code repository.

Github provides git as version control system for Demoiselle Signer.

Github provides an issue tracker for Demoiselle Signer. About that, it is important to say that Demoiselle Signer issue tracker has a specific tag for discussion. It shows that Demoiselle Signer issue tracker doesn't work only a place to request changes, but to talk about the project. Until August 30th 2022, there were 25 discussion issues. 7 were opened and 18 were closed.

We have not found references about chats in Demoiselle Signer page or in the Github source code repository.

We found a wikipage in Demoiselle Signer Github source code repository. This wikipage documents the several types of digital signature handled by component.

Demoiselle Signer website and documentation are available only in Portuguese. It is comprehensible that there are no expectations for contributions outside of Brazil because the component validates digital certificates based on a Brazilian public key infrastructure.

We have not found references about social media in Demoiselle Signer page or in the Github source code repository. We also have not found social media accounts with the name "Demoiselle Signer" using search engines. However, we discovered some posts of esaito developer in Facebook, Twitter and StackOverflow about Demoiselle Signer.

Conclusion

Remember we have proposed an expression to measure the community maturity considering the Fogel's Framework items.

$$CM = a * \sum_{i=0}^{9} t_i$$

As we have defined, *a* represents the community activity. An active community, as we have defined it, is a community with a release of its software in the current year. Demoiselle Signer had 37 releases until August 30th 2022. The last release was published on May 4th 2022, therefore *a* is equal to 1. Table V shows the values for each one of analyzed items of social, political and technical infrastructures.

TABLE V
SOCIAL, POLITICAL AND TECHNCIAL ITEMS

Item	**Value**
Decision System and Written Rules	0
Website	1
Mailing List	0
Message Forums	0
Version Control System	1
Bug Tracker	1
Real-Time Chat Systems	0
Wikis	1
Translations	0
Social Medias	0
Sum	**4**

So, for Demoiselle Signer, CM = 1 * 4 = 4. As we have said in the introduction, this number, alone, only indicates if a community is active or not. We don't have enough elements to say if a maturity value greater than zero is "good" or "bad". Of course, a CM = 10 probably indicates a community that meets the expectations of Fogel, but our current expression for CM doesn't indicate the maturity of each item. At this moment, we have an instrument for comparison. Using this approach for other communities, we can say if a community is more or less mature than Demoiselle Signer community.

We can observe that Demoiselle Signer community has the minimal infrastructure of communication. The discussion occurs inside the issues, when it occurs. We are not sure about it, but it seems that community is governed by a benevolent dictator, the esaito developer. He is responsible for the most of issues, commits and changed lines. In addition, there is no documentation about governance rules.

As we have said in the introduction, few software were produced by Brazilian government and it seemed that there were no communities around them. In this paper, we have investigated Demoiselle Signer and we have verified that there is an active community around it, although this community is not complete according to Fogel's Framework.

Future Works

We recognize that this investigation is an initial step for the building of a robust maturity community framework. Fogel's Framework is a starting point and it already has contributed to think about the aspects which should be considered for analyzing a FLOSS community. We think that next research steps could be:

- To apply this paper's methodology for other FLOSS projects produced by Brazilian government, for comparison and exploration of Brazilian FLOSS production by government entities;
- To apply this paper's methodology for other FLOSS projects produced by Brazilian non-government entities to compare the maturity among Brazilian FLOSS projects in general.
- Compare maturity of FLOSS projects produced by government entities with FLOSS projects produced by non-government entities;
- From the comparisons, to make refinements to analyzed items of Fogel's Framework, like definition of weights, and to discover indicators.

Possibly, it will be necessary to make deeper sociotechnical analysis for Demoiselle Signer and other FLOSS projects for discovering additional elements to be analyzed from community interaction study cases.

References

[1] LISBOA, Flávio Gomes da Silva. BEATRIZ, Marilene Zazula. *Use and Production of FLOSS in Brazilian Government: an Wide Survey*. https://sol.sbc.org.br/index.php/latinoware/article/view/19899. Accessed on 30 aug. 2022.

[2] ORAM, Andy. *Getting Started with InnerSource*. Sebastopol: O'Reilly Media 2015.

[3] TAURION, Cezar. *Software Livre: Potencialides e Modelos de Negócio*. Rio de Janeiro: Brasport, 2004.

[4] FOGEL, Karl. *Producing Open Source Software: How to Run a Successful Free Software Project*. Sebastopol: O'Reilly Media, 2005.

[5] PINHO, Viviane Dias Malheiros de. "*Uma contribuição para a melhoria colaborativa e distribuída de processos de software"*. http://repositorio.icmc.usp.br//handle/RIICMC/4803. Accessed on 26 aug. 2022.

[6] ORAM, Andy. *Getting Started with InnerSource*. Sebastopol: O'Reilly Media 2015.

[7] STALLMAN, Richard. *FLOSS and FOSS*. https://www.gnu.org/philosophy/floss-and-foss.htmlAccessed on 26 aug. 2022.

[8] BECK, Kent. *et al*. *Manifesto for Agile Software Development*. https://www.gnu.org/philosophy/floss-and-foss.html. Accessed on 26 aug. 2022.

[9] LISBOA, Flávio Gomes da Silva. *Framework Demoiselle: Controvérsias de uma Comunidade Fabricada para um Software Livre e Público*. https://www.cos.ufrj.br/shialc/content/docs/books/Memorias_VSHIALC_2018.pdfAccessed on 26 aug. 2022.

[10] ORACLE. *Java EE at a Glance*. https://www.oracle.com/br/java/technologies/java-ee-glance.html. Accessed on 30 aug. 2022.

[11] SERVIÇO FEDERAL DE PROCESSAMENTO DE DADOS. *Assinador Serpro*. https://www.serpro.gov.br/links-fixos-superiores/assinador-digital/assinador-serpro. Accessed on 30 aug. 2022.

[12] PRESSMAN, Roger S. MAXIM, Bruce R.. *Software Software Engineering: a Practitioner's Approach*. 9.ed. New York: McGraw-Hill, 2020.