DEPLOYMENT MODEL FOR SOFTWARE PROCESSES IN COLLABORATIVE AND DISTRIBUTED ENVIRONMENTS TYPICAL OF FREE AND OPEN-SOURCE FLOSS COMMUNITIES

Fabian Alonso Lara Vargas[*]
Luis Alberto Esteban Villamizar[**]

Received: 19/10/2022 • Accepted: 16/04/2023
https://doi.org/10.22395/rium.v22n43a2

ABSTRACT

This article presents the development of a model for deploying software processes within the context of free and open-source software FLOSS communities, characterized by the voluntary participation of people geographically distributed in different places and with diverse profiles and interests in a software development project. The Delphi method validated the model, in which 15 people interested in FLOSS communities participated. It was found that the model promotes communication and motivation of the participants in more than 80% and favors the participation of new community members, according to the answers given by the participants. The communication and motivation skills of the members of a FLOSS community are fundamental for the proper development of free software construction projects.

Keywords: FLOSS, software, model, process, open source

[*] MSc. Universidad Pontificia Bolivariana Seccional Monteria, Grupo de Investigaciones en Informática y Tecnologías emergentes ITEM, E-mail: fabian.lara@upb.edu.co. Instituto Universitario de Ingeniería Energética, Universitat Politécnica de València (UPV), Camino de Vera s/n, 46022 Valencia, Spain ORCID https://orcid.org/0000-0001-8246-1852.

[**] MSc. Universidad de Pamplona, CICOM. Ciudadela Universitaria, Pamplona, E-mail: lesteban@unipamplona.edu.co, ORCID https://orcid.org/0000-0002-5724-9136

MODELO DE IMPLEMENTACIÓN DE PROCESOS DE SOFTWARE EN ENTORNOS COLABORATIVOS Y DISTRIBUIDOS TÍPICOS DE COMUNIDADES DE SOFTWARE LIBRE Y DE CÓDIGO ABIERTO (FLOSS)

Este artículo presenta el desarrollo de un modelo para implementar procesos de software en el contexto de comunidades de software libre y de código abierto (FLOSS), caracterizadas por la participación voluntaria de personas distribuidas geográficamente en diferentes lugares y con perfiles e intereses diversos en un proyecto de desarrollo de software. El método Delphi validó el modelo, en el que participaron 15 personas interesadas en las comunidades FLOSS. Se encontró que el modelo promueve la comunicación y motivación de los participantes en más del 80% y favorece la participación de nuevos miembros de la comunidad, según las respuestas dadas por los participantes. Las habilidades de comunicación y motivación de los miembros de una comunidad FLOSS son fundamentales para el adecuado desarrollo de proyectos de construcción de software libre.

Palabras clave: FLOSS, software, modelo, proceso, código abierto

1. INTRODUCCIÓN

With the rise of the FLOSS movement at the beginning of the first decade of the 2000s, there was a first division between free software [1] and open source [2]. While the former was a social movement around a philosophy of freedoms, the latter revolves around a software development methodology and a business model; however, today, these two approaches converge into a single concept, thanks to the common point of the two movements regarding the access to source code as an indispensable element to achieve freedoms (of the free software movement) and the proper development process (of the Open Source movement). This common ground led to the unification of the acronym FLOSS, "Free/Libre Open-Source Software" [3] [4].

Regardless of the origin of these movements, it is clear that it is about software development. Therefore, it is essential to talk about the process that the different communities related to these movements use to obtain software products under the criteria of voluntary cooperation and, in most cases, in a distributed manner, depending on the location of the team members. On the other hand, any process, not only in the FLOSS context but generalized, even to industrial contexts, goes through a series of phases during its life cycle, for which it is necessary to interpret each stage's purpose and product.

In this topic, process modeling and tool integration were the main trends. As a base document for the formulation of the present article, it was found that in [5], the main research topics related to the software process are structured; in [6], a brief history and achievements of software process research are presented, as well as the critical evaluation of the results produced up to that moment. Unterkalmsteiner, in [7], reviews work related to the variety of process modeling, including the representation of modeling elements. Ruiz González in [8] makes a study on the integration of production and management processes in software projects, presenting some integrated tools, known as Software Engineering Environments (SEE), whose objective is to support these processes, leaving as an open problem the integration of the tools in a single process-oriented environment. Matturro [9] proposes managing the knowledge and experiences acquired during a software project as the most valuable asset for organizations in process improvement. Bermon in [10] offers the use of a wiki for the management of PAL process asset libraries. Rolandsson in [11] presents a study of how programmers cope with the coexistence of a commercial industrial production mode (typical of companies) and the cooperative community of the FLOSS context. Ruiz [12] proposes a framework based on the application of model-driven software engineering techniques and the integration of information through linked open data, where he also presents a framework for the deployment and evaluation of software processes.

Alvertis in [13] proposes a methodology to bridge the gap between customers and developers as a vital aspect of a successful development project, suggesting the Cloud Teams platform as a groupware system supporting collaborative software development. Linaker [14] proposes a Contribution Acceptance Process (CAP) model from which companies can adopt contribution strategies that align with product strategies and planning. Ewenike in [15] reviews the development process to assess the factors and gaps that create the need to improve the collaborative software development process in the cloud. Considering the above, it can be concluded that the main issues faced by FLOSS communities are related to collaborative development, communication, quality review, and process clarity.

Process life cycle

The process life cycle can be organized into the following phases (see Figure 1): i) the design phase consists of determining the main steps of the process and organizing them into phases, defining precedence relationships, inputs, outputs, tools, and techniques necessary to develop each of the stages constituting the process in question; ii) the initial deployment is aimed at launching a process for the first time in a real context and not an abstract one as in the case of design. This phase includes adequate documentation of the process and effective communication to the people executing it; iii) the continuous improvement phase consists of change control of the process, carried out through continuous monitoring (measurement) of each sub-processes, and designed in terms of effectiveness and efficiency. This phase is expected to be the longest in the life cycle to the extent that the process can continue to be improved without being declared obsolete in the light of technological advances, mainly in the tools and techniques used by the process; iv) and the retirement phase, which consists of declaring the obsolescence of the tools and techniques, so that the continuous improvement processes lose their efficiency and therefore it is more economical to change the entire process. This occurs mainly in processes in other contexts that require machinery or physical tools that become inefficient due to wear and tear and technological evolution. Likely, this retirement stage does not occur in some processes, such as software development, since they do not depend on physical tools, so it may happen that a process through continuous improvement changes radically over time, and therefore, continuous improvement can be seen as a reengineering process, by including the natural changes of tools and techniques to the rhythm of the advance of the technologies used in software development.

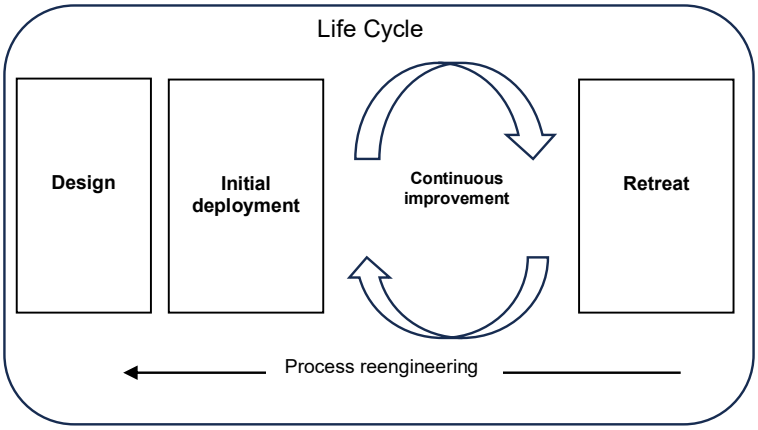

Figure 1. Life Cycle of a Process

Source: own elaboration

Particularly in software development processes, this same process life cycle can be interpreted as the knowledge required to make each stage effective (see Figure 2). The design phase [16] requires an adequate study of existing software life cycle models according to the needs and types of development projects [17], as well as of the various methodologies that have proven to be helpful or that have become standard practice due to effective communication and dissemination processes. It is recommended the study of formally defined standards such as ISO 12207 [18] and IEEE 1074 [19] that give clarity to the scopes of the sub-processes in which processes are typically subdivided, either at the level of a particular project or at the level of an organization engaged in software development.

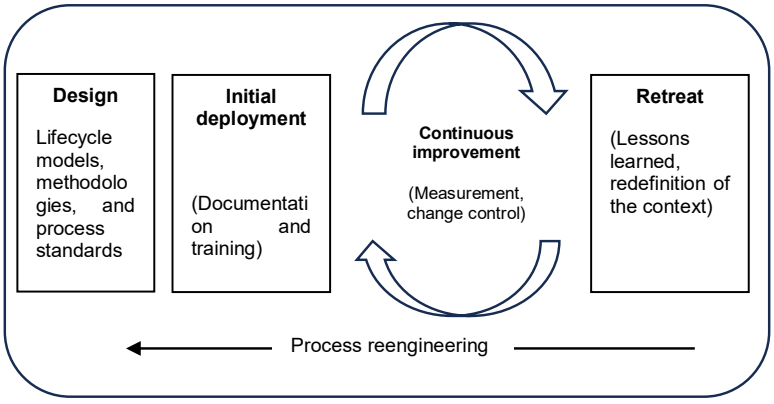

Figure 2. Software development process life cycle.

Source: own elaboration

The initial deployment stage is based on the ability to unambiguously describe each of the process's activities [20], so it is described in terms of roles, activities, tools, and techniques used in each activity. A second factor in this stage is the training of the people who will use the process, thus ensuring the usefulness of the documentation since documenting a process does not imply that people execute the activities as they were defined.

The continuous improvement phase includes evaluating the software process [21], [22] and is related to quality models. In the academic and business environment, the CMMI (Capability Maturity Model Integration) model [23], [24], which has proven to be effective in process improvement at the organizational level, with its descendants TSP (Team Software Process) [25] at the level of project teams, or PSP (Personal Software Process) [26], [27] at the level of individual members of software development, is recognized. Process changes also arise from product quality measurements, so it is necessary to consider models and standards such as McCall, GQM, FURPS, Dromey, GILB, ISO 9126, SQAE, WebQEM, and ISO 25000 (see Table 1). Much of the research links the deployment process to process improvement using the CMMI model.

Table 1. Models and standards for measuring product quality.

Model	Description
McCall	It is considered one of the pioneers in the evaluation of software quality. Its structure is based on three levels (factors, criteria, and metrics) and has eleven base criteria: accuracy, reliability, efficiency, integrity, usability, maintainability, testability, flexibility, portability, reusability, and interoperability.
GQM (Goal Question Metric)	It provides a way to define metrics based on applying some questions related to the project that allow the achievement of previously set goals.
FURPS (Functionality, Usability, Reliability, Performance, and Supportability)	It is a model developed by Hewlett-Packard, whose name comes from the criteria it evaluates Functionality, Usability, Reliability, Performance, and Supportability.
Dromey	It proposes three models for each stage of the development process: requirements model, design model, and implementation quality model.
GILB,	It guides software evaluation based on workability, adaptability, availability, and usability, which are divided into sub-attributes.
ISO 9126	It is based on McCall's model and is organized into four parts: quality model, external metrics, internal metrics, and quality metrics in use. It also includes the evaluation of characteristics such as functionality, reliability, usability, efficiency, maintainability, and portability, for each of which it defines sub-characteristics.
SQAE (Software Quality Assessment Exercise)	Based on Boehm, McCall, Dromey, and ISO 9126, it is oriented to evaluate by third parties not directly involved with the development. It is organized into three layers that allow the evaluation to be taught hierarchically: area, factor, and quality attribute.
WebQEM (Web-site Quality Evaluation method)	It is a website quality evaluation methodology that follows six phases: planning and programming of the quality evaluation definition and specification of quality requirements, definition and implementation of the elementary evaluation definition and implementation of the global evaluation analysis of results, conclusion and documentation, and validation of metrics.
ISO 25000,	Also known as SQuaRE (System and Software Quality Requirements and Evaluation), its purpose is to guide the development with requirements and evaluation of quality attributes, mainly functional adequacy, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability [28].

As previously mentioned, the retirement process and its subsequent reengineering phase can be considered non-existent in software development processes, in the sense that since it does not depend on physical tools, it can undergo drastic changes as part of continuous improvement; that is, the process initially deployed in an organization can be very different from the one used at a point in the life cycle, as long as continuous improvement guarantees the control of changes in the documentation and an effective process of communication of such changes through training. Therefore, lessons learned and changing technologies are quickly incorporated into the evolution of a sub-process, which can be seen as process reengineering in the long term.

Finally, it is vital to highlight the difference that may exist in the contexts of FLOSS development and proprietary development, in which there is no form of free access to the source code outside the boundaries of an organization that needs the software (and develops it) or of an organization that is dedicated to producing software and that was contracted by the one who needs the product [28]. The difference between these two contexts [19] lies in the voluntary collaboration and the geographical distribution of the team members. The first and most important differentiating factor is related to the attachment of the members to a development team: in the FLOSS context, in most cases, there is no financial compensation for all team members, while in the proprietary context, there is a contract that in one way or another obliges team members to comply with the rules defined by the contractor; As for the second factor, today it can be overcome through the appropriate use of technological tools for teamwork; however, it has been shown that collaborative work in the same physical space results in both process and product quality, as is the case of methodologies such as SCRUM [29], [30].

About the above, the question arises: how to develop a model for deploying software processes within the context of free and open-source software communities FLOSS, which contributes to improving communication, motivation, and participation of the people who make them up?

2. PROPOSED MODEL

The proposed model involves the software deployment phase and the design phase, which must guarantee the usability of the processes defined for a particular community. The proposed model (see Figure 3) presents three indispensable components to ensure adequate deployment of processes in a FLOSS community: constraints, techniques, and tools. The model also includes the particularities of a development context in which collaboration is voluntary, and its members are geographically distributed. It can be adapted to the reality of each FLOSS community under its particularities.

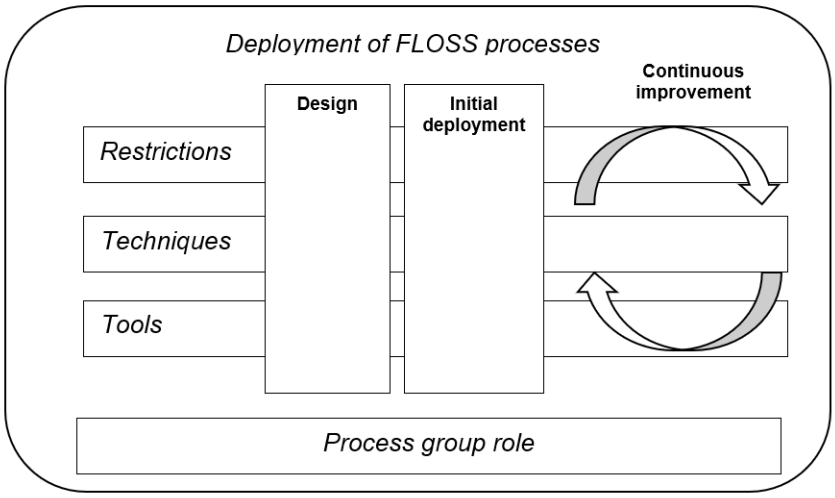

Figure 3. Proposed deployment model

Source: own elaboration

The construction of a solution starts from the design process, which is influenced by constraints, techniques, and tools that can alter this process. Then, we move on to the initial deployment process, conditioned by other restrictions, methods, and tools different from those used in the previous process. In this stage, deliverables are developed and improved until the minimum viable product is achieved, with continuous improvement of tests and corrected codes. It is essential to mention that the group of processes is from the beginning of the process until the desired product is obtained.

2.1 Design

A software process should be seen generally as the whole development process, including the sub-processes into which it can be divided. Therefore, when talking about the initial deployment of the process, it is considered for the first time the integral process whose only product is functional software, which includes the source code, specifications, designs, and documentation, both for the developers and the software users.

In a FLOSS community, there is an initial version known as "Credible Promise," which motivates members to join a project. This initial version of the product did not apply a community development process, which is why the design of the process will be a parallel task to the involvement of the first participants [29]. At this point, it is necessary to form the process group, which may be considered one more role in the overall development process. Its members will design, document, and give permanent support to all the other members in the different roles determined.

2.1.1 Restrictions

The primary constraint of the process design component is the context, considering the characteristics of the people who will use the process. It is necessary to characterize this population, which, although not clearly defined at the beginning of the project, can be predicted depending on the problem domain the project addresses. An essential part of the context is the geographical distribution of its members, for which the process must be supported by appropriate communication tools [28].

The following paragraphs describe the most important restrictions:

a. Iterative and incremental: FLOSS development contexts are characterized by frequent releases, so their design must include appropriate iterative and incremental development strategies.

b. Design based on roles and not on sub-processes: process documentation notations and tools allow the global software process description. Due to the great variety of profiles in the members of a community, these are self-organized according to their skills, and their linkage is always based on the role they play. Thus, the design of FLOSS processes must be based on each of the necessary roles: developers, designers, validators, documenters, and a new role proposed in this model: process engineer.

c. Flexible: the role of process engineer includes among its responsibilities the measurement and analysis of the processes of each of the roles defined in a community, so the initial design of the process must consist of aspects that facilitate changes without suffering traumas within the participants in the role affected by the change. This could be called process maintainability.

d. Simplicity: minimum necessary and sufficient based on the idea that something is simple when there is nothing to take away. Processes should be the minimum required to achieve the participation of the most significant number of stakeholders.

2.1.2 Techniques

The primary technique in process design is the Process Asset Library (PAL), for which open-access process repositories should be promoted. Just as the source code of a software product is available, the processes of FLOSS communities must be adequately documented and freely accessible.

2.1.3 Tools

The Eclipse Process Framework Composer (EPF Composer) is the primary tool for documenting software processes. With this tool, it is possible to specify the activities

corresponding to each role, the artifacts generated in those activities, and the tools used. In addition, the EPF Composer project has a repository of process libraries that includes documentation for approaches such as OpenUP, Scrum, and XP, which are freely available. These libraries can be a starting point for documenting proprietary processes in a FLOSS community. In addition, tools such as WikiPAL [31] have been proposed as appropriate options for maintaining a process repository.

2.2 Initial Deployment

This model component includes the publication of the process documentation, the training of team members, and the installation and integration of the tools that support the activities for each of the roles defined in the previously designed process.

2.2.1 Restrictions

The primary constraint in this aspect corresponds to the legibility of the process, that is, the ease with which each team member understands the activities of his role and the use of the tools for the construction of the artifacts under his responsibility.

Integration of tools: Repositories and version control have become the primary tools for working in FLOSS communities; however, other tools are required, such as communication tools, forums, bug reporting, testing, and documentation tools, which must be integrated appropriately to achieve that the members of a community consider a single context and that with a single access (login) they can access the tools involved in a given role.

2.2.2 Techniques

The deployment of an initial process is less critical than in the case of code deployment since the availability of the functionality must be guaranteed in real time. For the context of this work, it is suggested that the initial deployment be done by diverting the flow of participation of the most expert users to an instance of the new process. In this way, a beta test of the process can be considered, and, with their feedback, the pertinent adjustments can be made to provide service to one hundred percent of the participants. Training by different means is the primary technique for understanding each member's activities, tools, and techniques within a given role. Thus, the process group must have the corresponding strategies to advise mainly using the tools.

2.2.3 Tools

Collaborative development platforms: Also known as forges. They are mainly version control tools, in which all kinds of digital documents are registered, allowing control not only of the source code but also of requirements documentation,

designs, tests, manuals, and, generally, all kinds of artifacts in a software development process.

Communication between team members: Mailing lists and forums are the primary tools for communication between members of FLOSS communities. New requirements are communicated, solution alternatives are discussed, and coding and testing processes are undertaken through these. This type of tools can also include:

a. Bug Tracking System (BTS), used to register bugs found by users and serve as a tool for defining requirements, planning, and assigning tasks that constitute a collaborative development platform when integrated with version control and communication tools such as mailing lists.

b. Documentation tools: technical manuals (description of designs, requirements, packages, classes, source code, error codes, and their meanings), user manuals, and any other documentation that facilitates the modification, use, and operation of a software product. They constitute a guarantee of the long-term survival of a FLOSS project.

c. Wiki: this type of tool is probably the most versatile for the FLOSS context since it could support user documentation processes, documentation of the process itself, requirements, designs, and even a FLOSS project promotion portal.

2.3 Continuous Improvement

The component is based on managing process changes from measurement processes that detect parts of a process that can be improved.

2.3.1 Restrictions

In the FLOSS communities, the increments in the product are given as the commits are registered, which refer to the confirmations of a set of provisional changes in a permanent way in one or several constituent artifacts of a software product. On these, it is necessary to make the pertinent analysis in search of aspects that can be susceptible to improvement. Therefore, the main restriction for process improvement is given by the artifacts on which information analysis can be performed. Access to commit information within a community should be unlimited for the process group; however, it can be more valuable when it is available to anyone through repositories such as SourceForge Research Data Archive, FLOSSmole, or FLOSSMetrics.

2.3.2 Techniques

Defect characterization facilitates finding opportunities for improvement and allows finding the root cause of failures detected in testing processes. The analysis of reverse commits is fundamental since a large number of inputs in a community need to be incorporated into the product for different reasons, and studying this information allows process improvement.

2.3.3 Tools

Data analysis: Statistical and artificial intelligence tools enable the processing of large volumes of data to find patterns of behavior and identify opportunities to improve a process.

2.4 Role of Process Engineer

This role within a FLOSS community has the permanent mission of identifying, analyzing, and proposing solutions to inefficiencies in the development process, mainly performing the following functions: i) performing permanent analysis on commits to identify aspects for improvement in the process; ii) helping team members with the management of the tools in each of the defined roles; iii) helping to understand and adopt the process defined for the community; and iv) managing process changes, including the integration of tools and training for community members.

Within the capabilities of the people participating in this role, there should be skills very similar to those described for agile workgroups, such as self-organization and collaborative work.

The following methodology illustrates an example of the proposed model:

Design

- Socialize the credible promise in the FLOSS community.
- Invite community members to participate in the construction of the process group.
- Characterize people by their profile and geographic location of the process group
- To develop the appropriate communication tools for the process community
- Determine the roles of the process group
- Determine processes for change management and maintainability in such a way that they are not complex to implement
- Build open-access repositories

Initial deployment

- Include publication of process documentation, training of team members, and installation and integration of tools.
- Analyze the constraints for the process group to understand the activities of their role and the use of the tools for the construction of the artifacts under their responsibility.
- Determine communications, forums, bug reporting, testing, documentation tools.
- Analyze the work of the most expert users in an instance of the new process.
- Determine the tools for collaborative work
- Socialize documentation tools in the process group.
- Analyze the management of the commits to the process group and the open access group.
- Characterize the defects and their cause
- Analyze artificial intelligence tools to find error patterns
- Detect new opportunities for improvement in the process group

3. DISCUSSIONS FOR THE VALIDATION OF THE PROPOSED MODEL

Expert judgment is used in this work to validate the hypothesis:¿ Is it possible to formalize the development process in a FLOSS community in a way that motivates the participation of novice volunteers in this type of community (variable level of formalization of the software process)?

3.1 Instruments

The following instrument was designed to measure the suitability or not of using the model in a FLOSS community, according to the experts' criteria. The first part of the instrument collects information on the experience level of each invited to fill out the instrument. The second part corresponds to the convenience of using the model.

1. Participant's information:
   a) Full name
   b) Level of education
   c) Occupation

d) To what degree do you know how a FLOSS community works (basic, medium, high, very high)?
e) Have you participated in a FLOSS community? if yes, which one? in what role?

2. About the proposed model:
   a) The use of the proposed model will motivate the attachment of novice members in a FLOSS community.
   b) The use of the proposed model will motivate the participation of members who have previous experience in FLOSS communities.
   c) The formalization of a process through SPEM, as proposed by the model, will create dissatisfaction in the members of a FLOSS work team.
   d) A critical process when working with virtual teams is that of communications. The proposed model facilitates the communication process in a FLOSS community.
   e) Please leave here any observations and comments you consider relevant regarding the evaluated model (optional).

This second part of the survey uses a Likert scale. It is named after its author, Rensis Likert, and is a psychometric scale commonly used in surveys in which the level of agreement or disagreement with a statement is specified for the response options:

1. Strongly disagree
2. Disagree
3. Neither agree nor disagree
4. Agree
5. Strongly agree

4. ANALYSIS OF RESULTS

For the application of the instrument, the experts used a summary of the proposed model and the online instrument, which was also presented in printed format to know the questions in advance without the need to visit the digital version. Fifteen expert participations were obtained with the following characterization.

Current profession: The experts reported a varied profile, among which are: systems engineering student, systems engineer, academic secretary, development leader, teacher, systems engineer, computer engineer, electronic engineer, business intelligence consultant, head of information and communication technology center, information technology office manager, information security officer.

Highest level of academic training obtained so far. Most participants have a professional degree, which allows us to deduce that their concepts are based on the theory they acquired in their undergraduate academic program rather than their experience in FLOSS communities (see Figure 4).

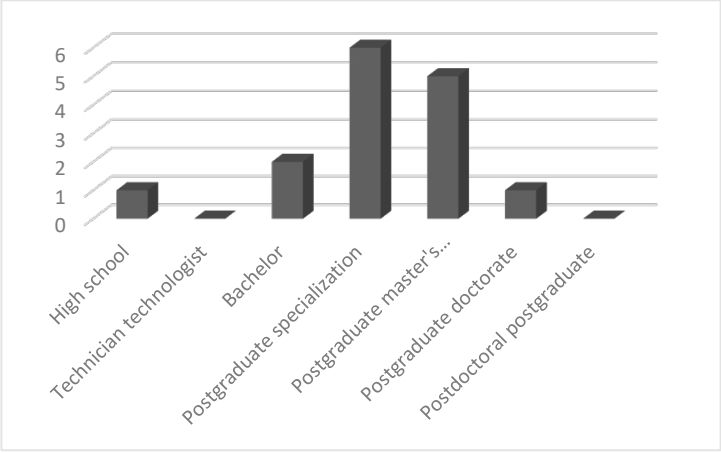

Figure 4. Academic level of participants

Source: own elaboration

¿To what degree do you know how a FLOSS community works? Most of the participants have a basic knowledge of FLOSS communities, which only partially guarantees the viability of the application of the model, so it is necessary to get concepts from members involved in FLOSS development processes (Figure 5).

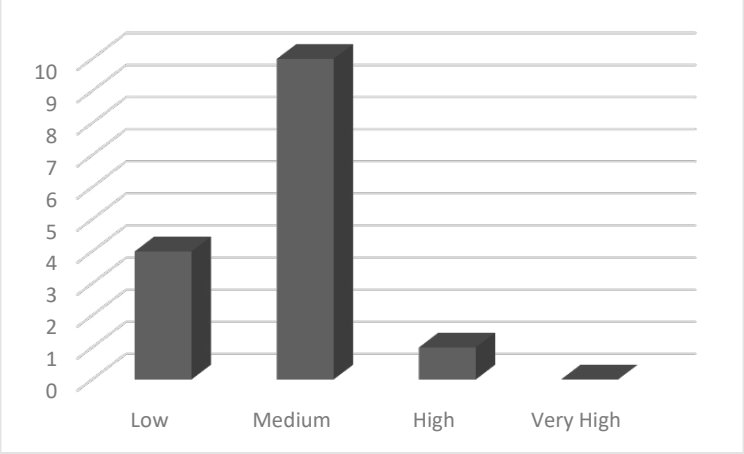

Figure 5. Participants' knowledge of FLOSS communities.

Source: own elaboration

¿Have you participated in a FLOSS community? The low participation in FLOSS communities coincides with the low knowledge about the internal development processes in this type of community (Figure 6).

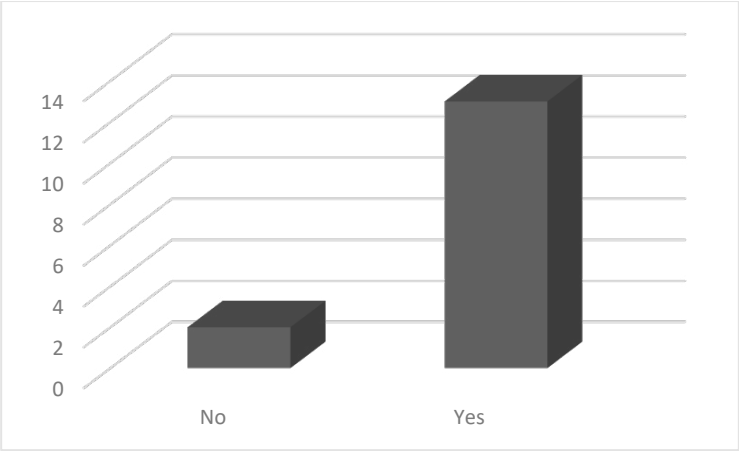

Figure 6. About the participation of experts in FLOSS Communities.

Source: own elaboration

Considering the evaluation of the proposed model by the experts, the analysis of the answers to the questions posed in the instrument is presented below:

¿Using the proposed model will motivate the linkage of novice members in a FLOSS community ?(Figure 7). The implementation of the model is feasible since the success of a FLOSS community depends mainly on the number of participants. Although several decades have passed since the movement started, many people interested in participating in a community for the first time find an obstacle in understanding the internal process, even more so when they do not have computer science as a profession.

¿Using the proposed model will motivate the participation of members with previous experience in FLOSS communities ?(Figure 8). According to the results, it is feasible to implement the model since it does not prevent users with previous experiences in FLOSS communities from perceiving the documentation of a process as a loss of freedom in their way of developing or participating in the community.

The formalization of a process through SPEM, ¿ as proposed by the model, will create discomfort in members with experience in FLOSS teamwork? (Figure 9). The documentation of processes through symbolic notations facilitates their understanding, regardless of the language; thus, experienced members will have the opportunity to assimilate changes in a strategy quickly and will not see in this notation an imposition that hinders the freedom of participation in a community.

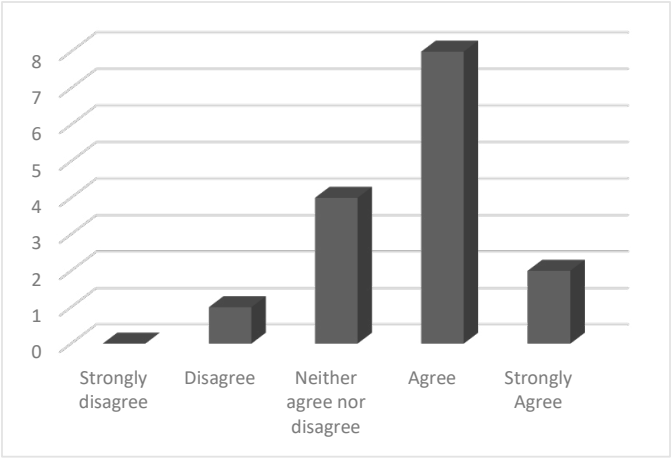

Figure 7. The model will encourage the participation of novices in a FLOSS Community.

Source: own elaboration

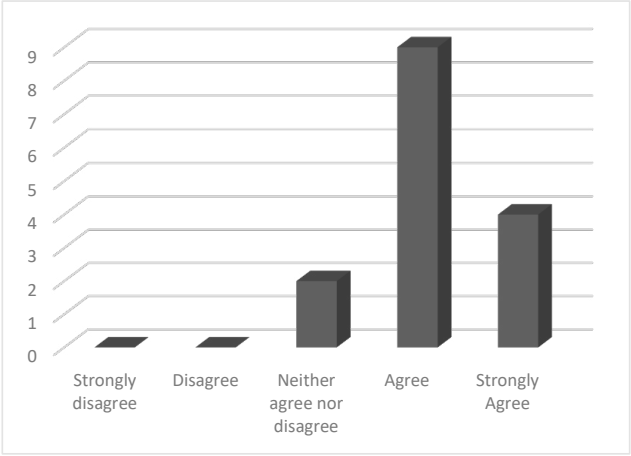

Figure 8. The model will encourage the participation of expert members in a FLOSS Community.

Source: own elaboration

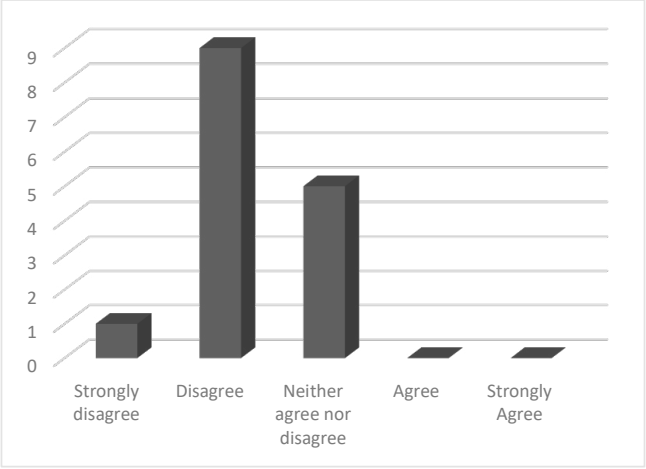

Figure 9. Formalization with SPEM will create dissatisfaction among the members of a FLOSS community.

Source: own elaboration

A critical process when working with virtual teams is that of communications.¿The proposed model facilitates the communication process in a FLOSS community ? (Figure 10). Most participants agree that it facilitates communication within a development process. It is highly probable that if all team members have the same understanding, their communication channels will be determined.

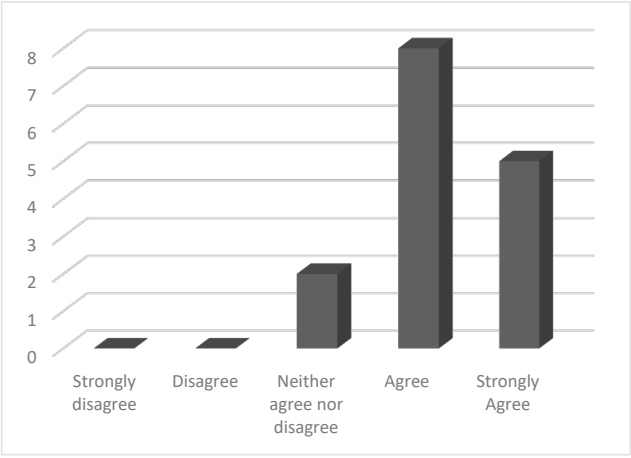

Figure 10. The model will facilitate the communication process

Source: own elaboration

At the end of the instrument, comments, and observations each expert considered pertinent concerning the model evaluated were allowed to be recorded. These are the comments recorded:

- "The model can be adapted to changes."

- "The characterization should have a minimum number of people to take into account in the model; in turn, there is no clarity on the tools to be used for integrating collaborative work tools and platforms is very open options."

- "In the proposed model, it needs to be clarified how the activities are classified according to their priority, the estimated time to be completed, what dependencies it has on other activities, and how they are monitored according to the individual assigned to it.

It should be clarified that, for this point, the model does not focus on estimating the activities and monitoring of each individual in the FLOSS community since this orientation is given by the group of experts and the process engineer of each free software community."

- "It is adequate and relevant as the roles within the community are defined well."

- "The proposed model defines some roles of the work team. The initial software product is known as Credible Promise. The design proposes the conformation of a group of processes with members designing, documenting, and providing permanent support to all the other members of the different roles."

- "The model allows designing and documenting software development processes in a graphical and standard-compliant manner considering the roles defined in the tasks and activities, work products or artifacts, tools, and processes."

- "It can be supported with DevOps"

- "I consider it would be useful to briefly describe the current process to better understand the proposed model's advantages, comparing these two."

The comments generally reflect the feasibility of applying the model since some focus on evaluating the summary document, not the model itself. The negative comments refer to the process design rather than the deployment, so the model does not describe the specific development activities. Instead, it proposes a process design phase in which the particular activities and the roles, tools, and artifacts in SPEM notation should be specified.

CONCLUSIONS

The software process deployment model proposed for FLOSS communities goes beyond the limits of deployment, so it was necessary to incorporate a process design

phase in recognition of the particularities concerning business processes, where the participants in the work team have total control. Thus, the voluntary nature of FLOSS development requires that the process be agreed upon among the initial community members without overloading it with activities that discourage the participation of a wide variety of profiles interested in a particular FLOSS project.

The validation process shows the feasibility of using the model; however, the academic and theoretical nature of the evaluations does not guarantee the success of applying the model in a specific context. In any case, knowing a process in advance motivates the voluntary participation of stakeholders in a FLOSS project.

It is important to note that the model encourages communication and motivation of participants by obtaining more than 80% favorability for these aspects. Likewise, the participation of new members is strongly encouraged, with a favorability of more than 60% of those surveyed. Therefore, the development of FLOSS communities could benefit significantly from using the model since commitment, motivation, communication, and knowledge are human capabilities that encourage the construction of new inventions and technological challenges.

Future research could be developed to analyze the impact of artificial intelligence on the growth of FLOSS communities and the development of agile methodologies within them.

REFERENCES

[1] R. Stallman, "LA DEFINICION DE SOFTWARE LIBRE," *COMMUNIARS*, vol. 3, 2020, Accessed: Dec. 02, 2022. [Online]. Available: https://revistascientificas.us.es/index.php/Communiars/article/view/12773

[2] E. Raymond, "The cathedral and the bazaar," *Knowledge, Technology & Policy*, vol. 12, no. 3, pp. 23–49, 1999, doi: 10.1007/s12130-999-1026-0.

[3] E. S. Raymond *et al.*, "Twenty Years of Berkeley Unix: From AT&T-Owned to Freely Redistributable OPENSOURCES Voices from the Open-Source Revolution," 1999. Accessed: Dec. 02, 2022. [Online]. Available: https://www.oreilly.com/openbook/opensources/book/

[4] Y. Puma Enríquez, W. López Abanto, Y. Mamani Laura, D. Lozano Flores, and J. A. Nuñez Muñoz, "Uso de software libre y de código abierto para la identificación de lineamientos estructurales y realce de estructuras geológicas," *Revista del Instituto de investigación de la Facultad de minas, metalurgia y ciencias geográficas*, vol. 24, no. 48, 2021, doi: 10.15381/iigeo.v24i48.20414.

[5] E. C. Forrester, "A Process Research Framework: The International Process Research Consortium," 2006. [Online]. Available: https://api.semanticscholar.org/CorpusID:107917513

[6] A. Fuggetta, "ICSE '00: Proceedings of the Conference on The Future of Software Engineering," New York, NY, USA: Association for Computing Machinery, 2000.

[7] M. Unterkalmsteiner, T. Gorschek, A. Islam, C. Cheng, R. Permadi, and R. Feldt, "Evaluation and Measurement of Software Process Improvement—A Systematic Literature Review," *IEEE Transactions on Software Engineering*, vol. 38, pp. 398–424, Jul. 2011, doi: 10.1109/TSE.2011.26.

[8] G. Canfora and F. Ruiz González, "Procesos Software características, tecnología y entornos," *Revista de la Asociación de Técnicos de Informática*, 2004.

[9] M. Matturro and S. Vázquez, "Modelo para la gestión de conocimiento y la experiencia integrada a las prácticas y procesos de desarrollo de software," *Universidad Politécnica de Madrid*, 2010.

[10] L. Bermón Angarita, "Librería de activos para la gestión del conocimiento sobre procesos software: PAL-Wiki," Nov. 2010, Accessed: Jul. 28, 2023. [Online]. Available: https://e-archivo.uc3m.es/handle/10016/10231

[11] B. Rolandsson, M. Bergquist, and J. Ljungberg, "Open Source in the Firm: Opening Up Professional Practices of Software Development," *Res Policy*, vol. 40, pp. 576–587, Jul. 2011, doi: 10.1016/j.respol.2010.11.003.

[12] I. Ruiz Rube, "Un framework para el despliegue y evaluación de procesos software," *Universidad de Cádiz,* 2013.

[13] I. Alvertis *et al.*, "User Involvement in Software Development Processes," *Procedia Comput Sci*, vol. 97, pp. 73–83, Jan. 2016, doi: 10.1016/J.PROCS.2016.08.282.

[14] J. Linåker, H. Munir, K. Wnuk, and C. E. Mols, "Motivating the contributions: An Open Innovation perspective on what to share as Open-Source Software," *Journal of Systems and Software*, vol. 135, pp. 17–36, Jan. 2018, doi: 10.1016/J.JSS.2017.09.032.

[15] S. Ewenike, E. Benkhelifa, and C. Chibelushi, "Cloud Based Collaborative Software Development: A Review, Gap Analysis and Future Directions," Jul. 2017, pp. 901–909. doi: 10.1109/AICCSA.2017.220.

[16] S. Acuña and X. Ferre, "Software Process Modelling," Dec. 2001, pp. 237–242. doi: 10.1142/9789812389718_0011.

[17] P. Mukala, A. Cerone, and F. Turini, "An abstract state machine (ASM) representation of learning process in FLOSS communities," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015. doi: 10.1007/978-3-319-15201-1_15.

[18] B. Mutafelija and H. Stromberg, *Process Improvement with CMMI® v1.2 and ISO Standards*. Auerbach Publications, 2008. doi: 10.1201/9781420052848.

[19] J. Castro and S. Acuña, "Differences between Traditional and Open-Source Development Activities," in *Product-Focused Software Process Improvement*, 2012, pp. 131–144. doi: 10.1007/978-3-642-31063-8_11.

[20] M. Alshakhouri, J. Buchan, and S. G. MacDonell, "Synchronized visualisation of software process and product artefacts: Concept, design and prototype implementation," *Inf Softw Technol*, vol. 98, pp. 131–145, 2018, doi: https://doi.org/10.1016/j.infsof.2018.01.008.

[21] I. Ruiz, R. Director, J. Manuel, and D. Beardo, "Un framework para el despliegue y evaluación de procesos software," 2013. Accessed: Dec. 02, 2022. [Online]. Available: https://rodin.uca.es/bitstream/handle/10498/15725/Ph.D.%20Iv%C3%A1n%20Ruiz-Rube.pdf?sequence=1&isAllowed=y

[22] D. Spinellis *et al.*, "Evaluating the Quality of Open-Source Software," *Electron Notes Theor Comput Sci*, vol. 233, pp. 5–28, 2009, doi: https://doi.org/10.1016/j.entcs.2009.02.058.

[23] "Capability Maturity Model ® Integration (CMMI SM), Version 1.1 Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1) Continuous Representation Improving processes for better products," 2002. Accessed: Dec. 02, 2022. [Online]. Available: https://resources.sei.cmu.edu/asset_files/technicalreport/2002_005_001_14039.pdf

[24] C. Product Development Team, "CMMI for Systems Engineering/Software Engineering, Version 1.02, Continuous Representation (CMMI-SE/SW, V1.02, Continuous)," 2000. [Online]. Available: http://www.sei.cmu.edu/publications/pubweb.html

[25] N. Davis and J. Mullaney, "The Team Software Process SM (TSP SM) in Practice: A Summary of Recent Results," 2003. Accessed: Dec. 02, 2022. [Online]. Available: The Team Software Process SM (TSP SM) in Practice: A Summary of Recent Results," 2003

[26] W. S. Humphrey, "Personal Software Process (PSP)," in *Encyclopedia of Software Engineering*, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2002. doi: 10.1002/0471028959.sof238.

[27] M. Azzeh, A. B. Nassif, Y. Elsheikh, and L. Angelis, "On the value of project productivity for early effort estimation," *Sci Comput Program*, vol. 219, p. 102819, 2022, doi: https://doi.org/10.1016/j.scico.2022.102819.

[28] F. Zambrano, N. Patiño, and F.-J. Pino-Correa, "Apoyando el despliegue de procesos en el contexto de las pequeñas organizaciones software," *Revista Científica*, vol. 43, no. 1, 2021, doi: 10.14483/23448350.18351.

[29] J. Gabriel, "Pequeñas empresas de software libre (floss) en la argentina," *XVII Congreso Latino-Iberoamericano de Gestión Tecnológica - ALTEC*, 2017, Accessed: Dec. 02, 2022. [Online]. Available: https://repositorio.altecasociacion.org/handle/20.500.13048/1551

[30] E. R. B. Cutler, J. Gothe, and A. Crosby, "Design Microprotests," *M/C Journal*, vol. 21, no. 3, Aug. 2018, doi: 10.5204/mcj.1421.

[31] L. Bermón Angarita, "Librería de activos para la gestión del conocimiento sobre procesos software: PAL-Wiki," Nov. 2010, Accessed: Dec. 02, 2022. [Online]. Available: https://e-archivo.uc3m.es/handle/10016/10231