Version and Change Control in Software Configuration Management Using Agile Methodologies: Application Framework in Development of Grain Traceability

Luma Alves Lopes, Denise do Rocio Maciel, Cláudio Roberto Agner, Mônica Cristine Scherer Vaz, Maria Salete Marcon Gomes Vaz

State University of Ponta Grossa (UEPG) - Ponta Grossa, Paraná, Brasil.

E-mails: luma_lopes@outlook.com, dnise_maciel@hotmail.com, claudioagner@hotmail.com, monicacsvaz@yahoo.com.br, salete@uepg.br

Abstract: In the development of a software, documents are generated and may vary throughout the project. Maintaining control of inclusions, alterations and exclusions of the content, helps maintain the software's quality during its development. The Software Configuration Management is the area responsible for component version control involved in the development of the software through the Version Control stage, and to indicate why the component was modified. In this sense, it is necessary to adopt methodologies to assist in the version control process. This article presents the version control using the Agile Methodologies Scrum and XP, applied in the Development of Application Framework of Grain Traceability - RastroGrão. As a result, the approach brought benefits such as the ability to trace documents and have access to all versions of it.

***Keywords*:** Versioning, Scrum, Extreme Programming, Traceability.

1. INTRODUÇÃO

A software is not only a computer program, it is all the documents and configurations necessary for the program to operate correctly. In this sense, a document is just as important as the creation of the program itself, and therefor necessary in planning the process control. For the construction of a software, it is necessary to consider a method with the defined steps and a methodology with its set of recommended practices [1]. A methodology generates documents, and these have version, needing version control.

During the software development, with the production of numerous documents, quality assurance is required, in order to facilitate the organization process of the development, in addition to modification and future development. When the control process has been defined, document planning consist of selecting which artifacts will be submitted to the Software Configuration Management (SCM), the area responsible for controlling the component involved in the software development [2].

With regard to the software, versioning allows you to recover previous versions, in case an error is detected in the current version. Keeping a record of files and the variations of a project allows greater security both for the developer as for the user.

The purpose of this article is to present the stages of Software Configuration Management and Version Control and Change, using the Agile Development

Methodology Scrum [2], in the development of the *RastroGrão* grain Traceability *framework*. In the same scope, it is the *Extreme Programming Agile Methodology* (XP) [2], in coding production.

2. THE RASTROGRÃO FRAMEWORK

Information technology plays an important role for information exchange between the production chain agents. The use of computer systems for assertiveness in decision making on the part of those involved in the management of the production chain is critical, may it be for the improvement of the process or product.

To meet this demand for data integration and availability of information for end clients through the internet, the RastroGrão Framework was specified [3], seeking to assist production process tracking. Through its structures, it is possible to track products, the stages of the production processes of each product, and the data to be tracked in each phase.

According to the defined structure, the process can assist in tracing any kind of grain, since the system can be managed by the user, according to his needs. Since the tracing process is not static and should cover all the production chain agents, the user can personalize the process to suit each need, that is, with the agribusiness rules, with new standards that can appear and with the constant research in the area that could result in new requirements and a Traceability process.

RastroGrão was modeled with five (5) module, as follows: 1) General Register, comprises the User Management which will have access to the framework, Enterprise Management, comprises the entities with structure traceability process and will be the management of data and Management Property, including the places where the grain is produced and where the traceability process parts. 2) Traceability Structure Management, related to product management, phases and attribute that will be part of the library, and can be customized for each company. 3) Customization, module where each company selects the products, phases and attributes, which will be part of the structure to be traced; 4) Data Register, module where the company will insert the production data, effective traceability, 5) Data consultation and generation of QRCode labels, containing the information that the company entered into the database and will be available for consultation for the company and/or the end user.

3. VERSION AND CHANGE CONTROL IN SCM

Version Control is a software engineering practice, organizing the project through component version management. The records of inclusion, alteration and exclusion of functions is maintained. Its application avoids problems such as the loss of project file versions, the difficulties in maintaining various versions of a system, source code modifications and the difficulties in knowing the author responsible for the modification of a project. The Change Control lets you know why certain version of an item of a software configuration was succeeded by another, indicating added, removed or modified features, may be predicted or not in the development plan [2].

Software Configuration Management is the area responsible for controlling versions of the artifacts involved in software development, defining how they should be

modified and identified [1][2]. It is a set of tasks related to Quality Management. The goal is to manage the alterations through the software life cycle and maintain the quality of the project as the configuration evolves, identifying the artifacts that define the software configuration, managing the changes of these artifacts and facilitating the construction of several versions of an application [4].

The configuration management process starts by identifying the components, followed by the Version Control stages, Change Control, Configuration Audit and Status Report [2]. This article aims to address Version control and Alterations, applied to the development of RastroGrão. Therefore it is necessary to understand the concepts of Software Item Configuration, Release, Baseline, Repository and Document metamodel.

The software configuration item is a development component, controlled by the management system, and can be modified only to the definitions in the configuration management development plan. The Release is the distribution of software version or a configuration item for the production environment. Baseline is a brand created by the technical review, featuring a version of the software or a configuration item that will not go to the production environment [2][4].

The repository is where the different versions of a configuration item are maintained and identified. Ideally, the control is automated, but can be done manually. Finally, the Meta Model Document (MMD) document defines the operation of the repository, the storage form of files and information, means of access and visualization, security management, data integrity and the ability to extend the repository to meet future needs [2].

Semantic Versioning is a set of rules and requirements that establishes how version numbers are assigned and incremented is in the form XYZ (Major Release. Minor Release. Correction Release) [5][6].

4. AGILE METHODOLOGY

There are several methods (or processes) for software development, but it is common for the organization to establish its own methodology or adapt any existing according to its reality. Among the main methodologies there is the Agile Development Methodology.

In the Agile methodologies, the key concepts are [7]: Individuals and interactions rather than processes and tools, executable software instead of extensive documentation, customer collaboration instead of negotiating contracts, and quick responses to changes instead of following plans. They are adaptive and work with constant feedback, which allows you to quickly adapt to any changes in requirements.

Within the agile methodology, we have the Scrum method that aims to define a process of interactive and incremental development [8] and Agile Model Extreme Programming, whose focus is on software development [2].

In the Scrum model there are three profiles: the Scrum Master, the Product owner and Scrum Team. The Scrum Master is the project participant who has a better understanding of the model and which will act as a resolver of conflicts, but not characterized as a leader or manager, as the teams in this model are self-organized. The Product owner is the person responsible for the project, indicating which are the

important requirements to be implemented in each development cycle. The Scrum team is the development team, consisting of 6 to 10 people, interacting for the development of the product [2].

The development of the Scrum involves some concepts such as The Sprint planning, Product Backlog, Daily Scrum, Sprint, Sprint Backlog, Sprint Review and Sprint Retrospective. The Sprint Planning is a meeting between the Product Owner and the development team, with the selection of the requirements or user history, called Product Backlog, to be implemented during the development cycle. The Daily Scrum refers to the daily meetings of the development team to control of what has already been implemented in the project.

The Scrum development cycle is called Sprint, lasting on average of two weeks to one month [2]. At the beginning of each Sprint the Sprint Backlog is set, a list of features formed from the Product Backlog will be implemented into the Sprint cycle that begins. At the end of Sprint the Sprint Review is performed to evaluate the product and the Sprint Retrospective to evaluate the work process [2]. The Scrum development phases can be divided into three phases [8]: planning, development and closing.

At the planning stage, the features required are defined by the system, based on the knowledge of the application. It is at this stage that the Sprint Planning takes place. In the second stage, the features are developed, respecting time, quality and requirements, performing the Daily Scrum. Finally, the closing step refers to the product delivery activities. In this step the Sprint Review and the Sprint Retrospective is carried out [8][9].

As an auxiliary tool for Scrum management, we used the method of improving the Kanban processes. It operates incrementally and evolutionary through experimental changes to optimize the software production [10]. Uses a visual control mechanism to track work of development. The process starts with the mapping of ongoing activities, and in sequence the activities to be controlled are selected. The table is divided into columns called States, defined according to the scope. For example, Not Started, Started and Ready [11].

The goal of XP methodology is to develop a system with better quality, produced in less time and in a cost-effective way, based on a set of values, principles and practices. Among the main values it has simplicity, respect, communication, feedback, and courage. Simplicity is related to the fact that the team focuses on the actual features and not on what might be needed. Respect and Communication refer to the relationship between the team and the customer, recommending good quality communication. The feedback should be performed as soon as possible in order to avoid any miscommunication, damage and higher costs. Finally, Courage is related to trust the changing needs [2].

In XP, the rules related to coding are [2]: Customer always available; Code written according to the standard; Write unit test; All the code is produced in pairs; Code integration; Frequent integration with an exclusive computer; Possession of collective code.

5. MATERIALS AND METHODS

Version control was applied on the encoding and on the documentation. In the case of documentation versioning, it was applied at two levels of detail. The first level controls

the creation of item configuration, through the standardization of nomenclature, the second level, controls the content of each item.

To perform the coding project version control, we used the tool Git, integrated with Netbeans, and the remote repository Github. For control of documentation we used Dropbox as a repository of other documents relating to the development of the project. Both the content stored on GitHub and the documents of Dropbox, were shared with all the members of the development team. The choice of the Github was based on studies [12], where the systems are analyzed, according to functionality and usability. In this study, the SVN system presents greater credibility, followed by Git/Github. For ease of installation and for being a distributed system, we decided to choose the Github. The choice of Dropbox was based on its rapid synchronization between the internet and local network.

For the versioning process applied in the development of the grain traceability framework, we used the agile Scrum and XP methodologies [2][8]. Scrum was implemented in the management model, while the XP was used in the coding of the system. Diagram (Figure 1) illustrates the process adopted for the Version Control and modification.

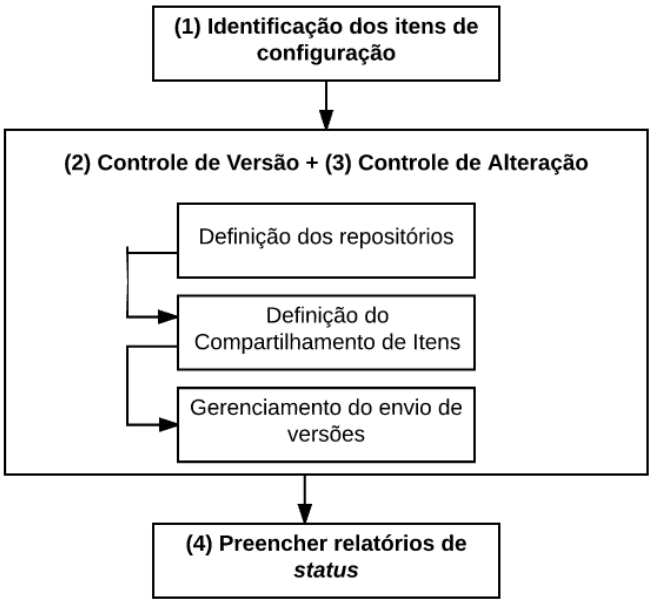


Figure 1 - Diagram of applied process versioning.

The process applied to the implementation of the Framework RastroGrão is a simplified version of the model management of software configuration [2][4], using four steps: identification of Configuration Items, Version Control, Change Control and Filling of Status Reports. Version Control and Change Control were applied simultaneously.

5.1 Identification of Configuration Items

From the identification of configuration items, the definition of which artifacts will be controlled in the versioning and which are the rules for the implementation of this process was performed. In the project we chose to version configuration items: Use Cases, build delivery, Minute Meetings, Project Opening, project closure, change requests and document meta model and the encoding.

According to the purpose and need for control, the rules for management of the versions were divided into version configuration items that do not relate to specific versions, version configuration items that relate to specific versions and version Internal Content of configuration items. Not all components defined as configuration items have been versioned.

The versioning of documents and coding of the system follows numbering formed by four sequences: system version, revision, correction and construction such as, 1.0.0.0 is the initial version, where it is the creation of the environment with the initial settings for development. The numbering system used in the project presents a level higher than the one proposed in [5]. This last level indicates the delivery of a Release.

The first sequence refers to the version of the software, this numbering is incremented each time the stages predicted in the initial scope are overcome. The second sequence defines the revision, it indicates an increase of a use or adding new functionality not predicted. The third result indicates the correction, and occurs when changes are requested due to review not agreeing with the initial proposal. This occurs because it is the product of programming errors. Finally the construction that symbolizes the availability of Review to the Product owner.

For performing new versioning, follow the following rules: The version of the system will be incremented only with a new project; The Review will be incremented for each case of use included in the project; The correction will be incremented for each bug/error fixed in the system; The build will be incremented for each version available for testing and/or production.

During the versioning of a spreadsheet of Version Control, it is filled out in Scrum (Figure 2). This spreadsheet performs the function of the Product Backlog, where the first four columns refer to the number sequences used, using the pattern of semantic versioning, they promote the Version Control. The other columns, Comment, Date, Hours and the Responsible provide the Change Control. In Figure 2, the inclusion of cases of use of the Access Management, Manage Rules and Manage Permission through an increase of the Revision column.

Version	Revision	Correction	Construction	Commentary	Date	Hours	Responsible
1	0	0	0	Environment Initialization	2016/04/19	2:00	Member x
1	0	0	1	System Authentication	2016/04/08	1:00	Member x
1	1	0	1	Manage Access	2016/04/20	3:00	Member x
1	2	0	1	Manage Rule	2016/04/21	3:00	Member x

Figure 2 - Example of a Version Control Table filled out.

In order to implement Version Control and Change, it was necessary to establish how items would be managed. We defined the rules for: versioning configuration items of documentation that do not relate to a specific version, versioning configuration items of documentation that relate to a specific version, versioning internal content of configuration items of documentation, versioning codification configuration items.

More details on the Component Identification applied in this work can be found in [13]

5.1.1 Versioning configuration items of documentation that do not relate to a specific version

Applied to configuration items associated with more than one, or none, version of the software. Understand the configuration items: Meeting Minutes and deliveries of the build. The minutes of the meeting shall specify the details of the Sprint Review; the supply of build are documents submitted in the Sprint Review to the Product Owners; the documents should specify the details on the Delivery.

The control of these configuration items is performed in the nomenclature, which follows the rule "File Type + "." + date + "." + hour". The symbols used to represent the type of file were AT meeting minutes, SC to request correction, SM for Change Request and EC for build delivery. At this stage the Change Control is not applied, since the configuration items shall not be subject to change. If the event occurs, the configuration items prior to modification should be replaced. In Figure 3, an example of versioning applied to nomenclature with date is presented. The file type is "AC", the date is "31.05.2016" and the time "07.00".

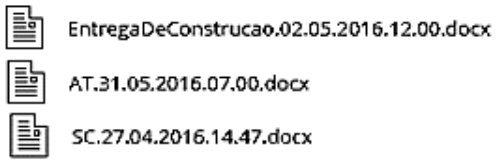


Figure 3 - Example of versioning applied to nomenclature with data.

5.1.2 Versioning configuration items of documentation that relate to specific version

This versioning is applied to configuration items connected with only one version of the software. In this case the version is more representative than the date. When applied to the configuration item, Use Case, the rule of the nomenclature is "File Type+"." + "Version Number". The symbol used to represent the type Use Case was UC. The control of such configuration item changes occurs through the Version Control spreadsheet (Figure 2). In Figure 4, an example of versioning applied. The file type is "UC", the release is "1.0.0.1".

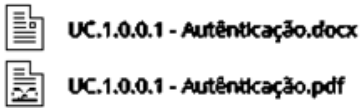
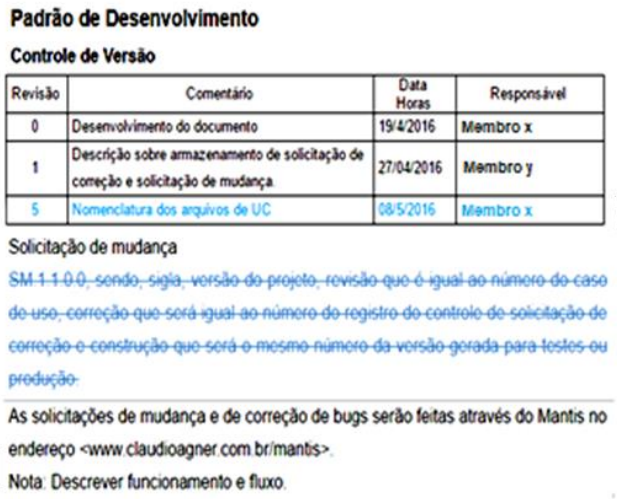


Figure 4 - Example of versioning applied to nomenclature with version.

5.1.3 Versioning Internal Content of Configuration Item Documentation

This versioning is used in conjunction with the versioning cited above, applied to the Configuration Items: Meta-model Document, System Tutorial, Tutorial for Development Requirements and Use Cases. This versioning is implemented in components with need for more strict control and in a scenario with simultaneous interaction of multiple agents. With each series of changes made in the document, the author points them out with a blue color, and performs the fills out the table "Version Control", available at the top of the document.

In Figure 5 an example of versioning is presented, applied to the configuration item "MMD". It is verified in the column review that implemented the numbering system to a level as versioning control. The other columns provide change control to the content of the software configuration items (SCI).



Padrão de Desenvolvimento

Controle de Versão

Revisão	Comentário	Data Horas	Responsável
0	Desenvolvimento do documento	19/4/2016	Membro x
1	Descrição sobre armazenamento de solicitação de correção e solicitação de mudança	27/04/2016	Membro y
5	Nomenclatura dos arquivos de UC	08/5/2016	Membro x

Solicitação de mudança

SM.1.1.0.0, sendo, sigla, versão do projeto, revisão que é igual ao número do caso de uso, correção que será igual ao número do registro do controle de solicitação de correção e construção que será o mesmo número da versão gerada para testes ou produção.

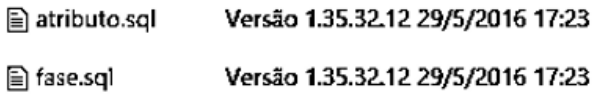
As solicitações de mudança e de correção de bugs serão feitas através do Mantis no endereço <www.claudioagner.com.br/mantis>.

Nota: Descrever funcionamento e fluxo.

Figure 5 - Example of versioning applied to Configuration Item content.

5.1.4 Versioning codification configuration items

This versioning is applied to encoding configuration items, and are components connected with only one version of the software; and the date of availability is as important as the version. The classification follows the standard "Version" + " "+ "Version Number" + " " + "date"+ " "+ "time". In Figure 6, is an example of versioning applied to coding is presented. The version number is "1.35.32.12", the date is "29/5/2016", the time is "17:23".



atributo.sql Versão 1.35.32.12 29/5/2016 17:23

fase.sql Versão 1.35.32.12 29/5/2016 17:23

Figure 6 - Example of versioning applied to coding.

Unlike previous versioning, where the configuration items are stored in the repository Dropbox, for the storage of coding the repository Github is used. Filling out versioning data occurs in the Step Version Sending of version control in the management software configuration, the change control over the encoding is described in Product Backlog (Figure 2), where the encoding can be linked to business logic or indicate change requests in the review.

These requests were controlled through the application Mantis Bug Tracker, where the method of process improvement Kanban is used to manage it, within the agile development methodology Scrum. The frame was replaced by the digital interface of the Mantis application, and the steps of control that were adopted were "Assigned Requests", "Requests Not Assigned", "Requests reported ", "Resolved Requests", "Recently Modified Requests" and "Monitored Requests". From the time of the request, an integral part of the Scrum Team performs the analysis and in case of acceptance binds the request to a new version broken down in the Sprint Backlog (Figure 2).

5.2 Version Control and Change

Version Control is accomplished through the tasks of Repository Definition, Item Sharing Definition and managing the version sending. The Change Control was applied during the whole software process, indicating the reasons that led to increasing versions.

In the definition stage of the repositories, we opted to use Github for control of the encoding and Dropbox for control of other files. In the definition stage of sharing items, the control was manual, the part that makes changes in a document or coding, should analyze the changes made by others in order to avoid conflicts with what has been done. In the stage of managing versions sending, the sending was only performed to the repository only if the file is free of defects and based on standards of the adopted nomenclature. At this moment Change Control of documents is performed.

5.3 Filling out status reports

Filling reports is the last step in the process of managing Software Configuration. Throughout the implementation process of the framework RastroGrão, the documents were filled out manually: Version control, where the baselines for the project are defined, the RastroGrão timeline, where you register the hours spent in each Sprint Backlog, the meetings timeline, detailing the hours spent in meetings and the delivery report for every construction.

6. RESULTS

For performing the version control and change, it is necessary for the definition of the repositories. In the Framework RastroGrão project we chose to define two repositories (Figure 7). To host the documentation we used Repository I made available via the Dropbox, and to host the encoding we used Repository II, through Github.

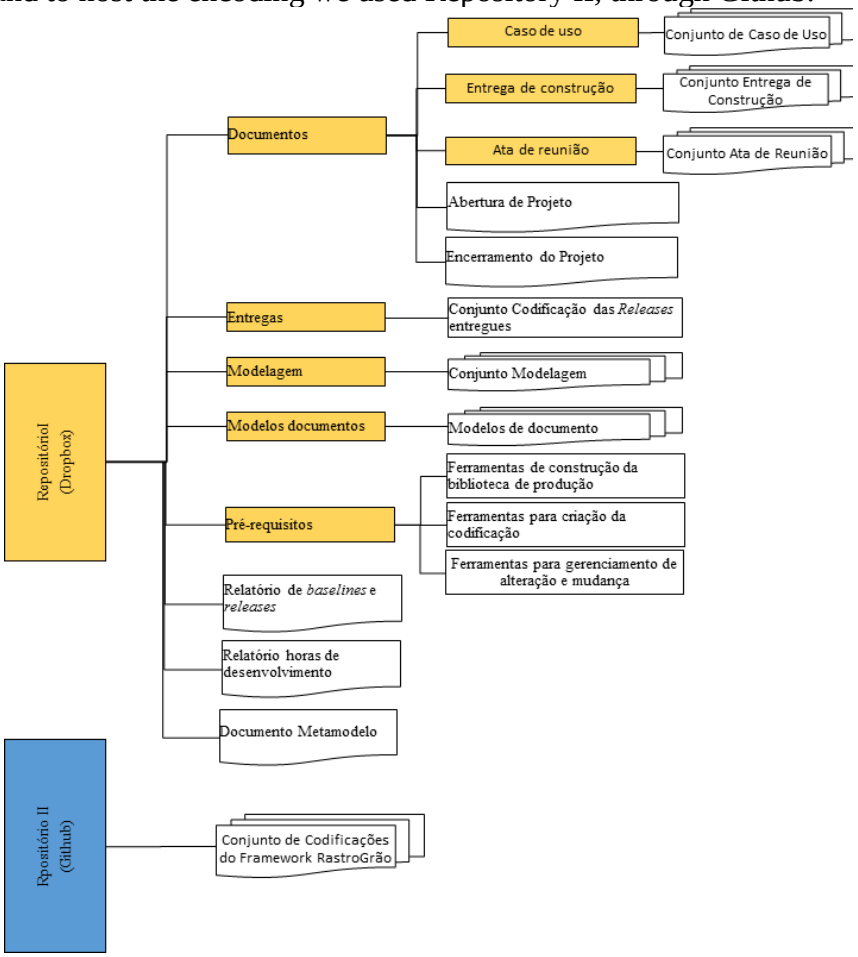


Figure 7 - Dropbox Repository Definition.

Figure 7 shows the final definition of the structure, however, there have been changes since the beginning of the project, due to the need to group the items of software configuration "Use Case Set", "Construction delivery set" and "Meeting Minute Set", in folders. Besides this, the project included requests for changes carried out through documents that were previously set at the opening of the project, however this model could compromise storage and management. For this reason, the Mantis was used, presenting a benefit to automating the process of managing requests for correction and changes. In counterpart, it is necessary that a member of the Scrum team examines the requests, in order to set the corresponding version.

Regarding the layout of Scrum, the first step of Sprint suggests a planning meeting, Sprint Planning, with the development team, Scrum Team, and with the customer, Product Owner, in order to set goals to be achieved. As the objective was to implement the framework of grain traceability, the list of the Product Backlog was generated from use cases extracted from [3], and based on this specification defined in the Sprint Planning and five Sprint Backlogs.

In the implementation stage, the team held meetings to monitor the progress of the project development, as Scrum suggests, however it was decided to hold weekly meetings and not on a daily basis. Still, following the method Scrum held the Sprint Review and the Sprint Retrospective, along with the deliveries of releases. The project included implementing XP in every step of coding, but due to the limitations of human resources and development period, some policies have not been implemented. Among them, coding in pairs and the definition of unit tests, before coding. All Members had access to the code, but it was agreed that only some could perform modification on it.

The Policies XP applied, were keeping the customer always available, with contacts in person and through a virtual environment, standardization of codes, offering them in the configuration item MMD, integration of one code at a time, frequently and with computer exclusively for this purpose, on the GitHub repository.

7. CONCLUSION AND PROSPECTS FOR FUTURE WORK

To keep the components organized and manageable, it becomes necessary for adequate documentation and versioning. When it comes to coding, the project developed ended with a total of 11 releases. Of these releases, five correspond to the delivery of a module from the system while the other six match the change request.

Applying the Version Control and Change has brought benefits such as greater independence for the team and greater security, since in case of a problem the various SCI can be returned to a previous version. It is important to be aware that the process requires maturity from the members of the team.

The Scrum method allowed greater control over the development process, through incremental deliveries, allowing you to check the functionality of the specifications determined by the Product Owner. The XP was efficient, especially in relation to the incremental deliveries, made available to the development team, for future developments, by applying the coding in pairs in order for greater productivity. For the versioning process, some aspects identified need of improvement, as shown in Table 1, and are suggested as prospects for future work.

Table 1 - Proposals for improving the process of versioning.

Item	Current	Improvement
01	The nomenclature of SCI is based, sometimes, on the date of creation and on the system of semantic versioning. In the MMD this system is described for each item, which causes doubt in members of the project generating errors in the nomenclature.	Keeping the versioning system changing in the way the rules are described in the SCI MMD. One solution is to define the rules as sections and internally enter the information of the documents to which these rules apply. In addition to facilitate the interpretation, it must a reduction on the load of documentation.
02	The rules of coding nomenclature are not standardized with the rules of the nomenclature applied in the documentation. The fact occurred because of the need to version the baselines.	To solve the problems described and others that may arise due to changes in how to implement versioning, one solution is to define a team member as responsible for managing the repository.
03	The relationship of dependency used is done through the configuration item MMD. This system does not allow you to view the relationship between the different versions of the same item and as does not bind to semantic versioning.	As a solution, fit in the identification stage of Configuration Items to implement the step definition of relationships of SCI suggested in [2]. It must allow linking the items of software configuration between themselves and between the semantic versioning of the project.
04	The current model follows suggestions from the Kanban to compose the meeting minutes. The fact that this information is internal to the documents creates difficulties of viewing them.	Keep the information in the Meeting Minute documents, but use the board for the process to be viewed with greater ease.

ACKNOWLEDGEMENTS

The authors thanks to Comissão de Aperfeiçoamento de Pessoal do Nível Superior – CAPES for the financial support.

REFERÊNCIAS

[1] SOMMERVILLE, I. **Engenharia de Software**. Addison Wesley, 2003.

[2] WAZLAWICK R.S. **Engenharia de Software**: Conceitos e Práticas. Rio de Janeiro: Elsevier, 2013.

[3] VAZ, M. C. S. **Especificação de um Framework para Rastreabilidade da Cadeia Produtiva de Grãos**. 87f. Dissertação (Computação Aplicada) - Universidade Estadual de Ponta Grossa, PR. 2014.

[4] PRESSMAN, R.S. **Engenharia de Software**: Uma abordagem Professional. São Paulo: AMGH, 2011.

[5] PRESTON-WERNER, T. **Semantic Versioning 2.0.0**. Available in: http://www.semver.org/. Access: 01 fev. 2017.

[6] RAEMAEKERS, S et al. **Semantic versioning versus breaking changes**: A study of the Maven Repository. 4th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM 2014). Victoria, Canada, 28-29, 2014.

[7] BECK, K. et al. **Manifesto for Agile Software Development**. Available in: http://agilemanifesto.org/. Access: 01 fev. 2017.

[8] SCHWABER, K. **Agile Software Development with SCRUM**. Prentice Hall, 2002.

[9] BISSI, W. **Scrum** - Metodologia de Desenvolvimento Ágil. Campo Dig., Campo Mourão, v.2, n.1,p.3-6, 2007.

[10] STELLMAN, A., Greene, J. **Learning agile**: Understanding scrum, XP, lean, and kanban. O'Reilly Media, Inc, 2014.

[11] KNIBERG,H., SKARIN M. **Kanban e Scrum obtendo o melhor de ambos**. InfoQ, 2009.

[12] Palestino, C.M.C. **Estudo de Tecnologias de Controle de Versões de Software**. 72 p, Trabalho de Conclusão de Curso (Gestão da Informação) - Universidade Federal do Paraná, 2015.

[13] Maciel, D. R. et al. **Component Identification in Software Configuration Management Applied to the Development of Framework Traceability**. International Journal of Engineering Research & Science (IJOER). v.3, n.1, p. 1-8, 2017.