Proposal for requirements traceability in distributed agile teams

https://doi.org/10.56238/uniknowindevolp-064

Rafael Augusto Paggi Diomedesse
Course Technology in Systems Analysis and Development – Federal Institute of Mato Grosso – IFMT – São Vicente Campus.
E-mail: rafael_paggi@hotmail.com

Libia de Souza Boss Cunha
Course Technology in Systems Analysis and Development – Federal Institute of Mato Grosso – IFMT – São Vicente Campus.
E-mail: libia.boss@svc.ifmt.edu.br

ABSTRACT
This article aims to present an experience report on the configuration of an automated management tool, Jira Confluence, for documentation and requirements management, throughout the software lifecycle using agile teams, in addition to clarifying concepts about the Traceability of requirements and situate through examples why standardize and centralize them, Especially when we're talking about distributed teams.

Keywords: Scrum, Agile Development, Traceability and Agile Methodology.

1 INTRODUCTION

Nowadays it is possible to observe in a remarkable way that a significant and growing number of companies around the world have been distributing their software development processes as a competitive strategy associated with cost, quality and flexibility [Santos, et al.]. In this context, Scrum presents a great growth over other methodologies, because they have the ability to reduce risks of failures, provides fast deliveries, deals with scope changes easily, in addition to other reasons such as the reduction of bureaucracy of activities and iterative communication between the client and the team [Sabbagh, 2013].

Scrum focuses on people and is suitable for projects where requirements are changeable in the course of development. This methodology has practices and techniques that fit easily in distributed teams, because it has a great adaptability that will be described soon in this article.

The process of elicitation and analysis of requirements in Scrum teams encounters some challenges, such as customer availability, cross-functional team, neglect of non-functional requirements, lack of formal documentation, among other reasons [Soares 2013].

Many factors hinder the process of analyzing requirements in distributed teams, among them: physical separation and time zone. However, distributed agile teams can achieve the productivity of a physically united team [Cohn 2012], but for this to happen the process must be integrated from the requirements gathering. A major barrier in distributed teams is the collection of requirements and the way they are made available to the team.

This paper aims to present an experience report of configuring an automated management tool, *Jira Confluence*, for documentation and requirements management, throughout the software lifecycle using agile teams. The software was used in order to manage the collection, storage and maintenance of the requirements agreed between the *stakeholders* (strategic public), in addition to managing the changes that occurred in the requirements, due to its natural evolution and to track the relationships between requirements and between requirements documents.

The target company of this work, provides software development services to companies in the public and private sectors. Using the agile methodology Scrum and has been in the market for about ten years consolidating with the successful completion of several projects. The configuration of the *Jira Confluence* software in order to meet the requirements management in agile projects carried out by this company arose from the maturation that was obtained through problems visualized in previously developed projects. It is expected to report, contribute and foster the discussion of good work practices regarding the management of requirements carried out in the current market.

2 AGILE SCRUM METHODOLOGY

Due to the presentation of positive results in relation to the use of agile methodologies, more and more *software houses* have been seeking these methodologies for application in software projects. Among the options is Scrum, according to Versionone, [2008] 70% of companies that use some agile methodology opt for Scrum. Scrum is a flexible framework, focused on teamwork and used for the incremental and iterative development of any product [Collins, et al., 2010].

It is considered a great solution for companies looking for fast and functional deliveries, in addition to the detachment of initial documentation. This methodology has three features that are considered by Sutherland [2013] as pillars of it: transparency, inspection and adaptation.

Transparency refers to the knowledge of the significant aspects of the process, which all those responsible must possess [Vieira 2016]. Silva, et al. [2009] reports transparency as the guarantee of clarity of project results for both parties (company and client). Inspection is the continuous performance of monitoring and testing that focus on finding variations of the functionality specification, thus avoiding problems of expectation breakage and refactoring of the same. Cruz [2013] points to inspection as essential and states that sufficient inspections must be made for variations to be found.

After inspections and found variations, the adjustment of the deviation outside the acceptable limits should be immediate [Vieira, 2016]. Changes deemed necessary should be allowed and executed in the product *backlog* [Silva, 2017]. The changes necessary for the functionality to return within a limit of acceptable changes should be executed as soon as possible so that future deviations are minimized [Cruz, 2013].

2.1 ROLES AND RESPONSIBILITIES

For the concept of the three pillars to be solid, Scrum consists of a set of small teams that have responsibilities for events that rely on the use of specific artifacts for support and apply rules according to the roles assigned to each team participant [Cruz, 2013]. In the Scrum framework there are 3 roles which are:

- **Product Owner: Product** owner.
- **Scrum Master:** Member of the team responsible for managing the project and leading the *Scrum Meetings*. Despite having special functions vis-à-vis the other members of the project, the Scrum Master does not have authority over other members of the team [Bissi, 2007], [Pedro, et al. 2016] completes by saying that the Scrum Master is the guardian of the Scrum processes.
- **Development Team:** The project development team consists of 3 and a maximum of 9 people with self-management characteristics [Pedro, et al., 2016].

Figure 1 – Scrum Roles

3 AGILE REQUIREMENTS MANAGEMENT

Requirements management deals with the management and control of system need (requirements) including the traceability of requirements and their changes as they evolve [Molinari, 2008]. This management is fundamental for a software to have QA (*Quality Assurance*), because the requirements without proper management, or worse, without any management, directly impact the quality, time and expectation of the functionality delivered.

The requirements gathering has an important role in the construction of a software because it is the beginning of all project activity. One of the reasons for user dissatisfaction is the lack of an adequate requirements survey [Mendonça, 2014].

As stated by Dimes [2014], many *startup* companies find it difficult to deliver functional prototypes on time, because there is no control over specific deadlines for small tasks. In addition, they underestimate large tasks and only focus on them when it is close to the deadline. With this is generated a wear and tear that causes loss of deadlines or delivery of features with unsatisfactory quality. Dimes [2014] concludes by saying that companies that started using Scrum, realized quality and punctuality in addition to more productive developers.

The process of elicitation and analysis of requirements in Scrum teams encounters some challenges, such as customer availability, cross-functional team, neglect of non-functional requirements, lack of formal documentation, among other reasons [Soares 2013].

4 APPLYING REQUIREMENTS MANAGEMENT IN AGILE TEAMS USING JIRA CONFLUENCE

4.1 ABOUT THE COMPANY

The company object of study of this article, operates in the field of Information Technology and is a specialist in Software acting in three business lines: Custom Design, Training and Commercial Automation. It is an Information Technology (IT) company whose mission is to help customers and partners to be more agile and competitive in business through the use of technology.

It has been using the *Scrum framework* in its development process for some time and has been consolidating itself in the maturation of good practices. The project studied in this article works with a distributed Scrum team and used the Remote Team Member distribution pattern.

The Remote Team Member distribution pattern consists of work where a single team member acts in a different location from the rest of the team [STARR, 2012]. Using this distribution line, the project has a requirements analyst separated geographically from the team, with the function of monitoring and collecting information about the needs of the *Product Owner* and is located in São Luiz do Maranhão – MA, while the development team and other professionals involved are at the company's headquarters in Cuiabá – MT.

4.2 JIRA TOOL

JIRA software is a project management tool for agile teams, with support for various methodologies, such as *Scrum*, *Kanban*, or similar. It is developed by the Australian company *Atlassian*, which specializes in collaboration and development tools. From agile whiteboards and reports, the tool allows you to plan, control and manage multiple agile software development projects in one place.

Among the features present in this tool, we can highlight:

- Web *and* mobile *tool*: allows access to information regardless of the location of the participants;
- Easy preparation of the *Backlog*: Allows the creation of tasks, categorizing for example, by user stories and *bugs*, allowing the prioritization of these tasks easily through the drag

and drop feature. In addition, it allows you *to upload* and *download* documents related to tasks;

- *Virtual Kanban : Possibility of creating and customizing* Kanban, *in order to better adapt to your workflow of each project.*
- Integration with *Confluence: Confluence* is a commercial tool for information management and collaboration, known as Wiki, also developed by the company *Atlassian*. The data included in this tool serves as project documentation, such as: requirements management documents, meeting minutes, and other project information.
- Integration with development tools: Possibility to connect with code versioning tools such as *Bitbucket* or *GitHub*, providing teams with traceability on the *backlog*.

Currently there are tools on the market, similar *to Atlassian Confluence* such as *Trello* and *Redmine*, but we chose to use *Confluence* because the *Git* repository used in the project is *Bitbucket*. *Confluence* has a simple, functional and secure integration with this repository thus ensuring the purpose of this article, end-to-end traceability, from the requirements collected to the versioning of the source code. *Redmine* also has integration with the Bitbucket repository, *but* Jira, *Confluence* and *Bitbucket* are *Atlassian* products*, so the ease of integration between these services is absurd.*

4.3 CONFLUENCE TOOL

Confluence is a software where there is content collaboration, basically it is a repository of information where *Wikis can be created* formally documenting the requirements collected with the PO (Client or Project Owner) informally, thus avoiding wear and tear and inconsistency with the survey made by the requirements analyst. This is a space where the client can access and validate the information contained therein, thus following one of the principles of *Scrum*, the active participation of the Client.

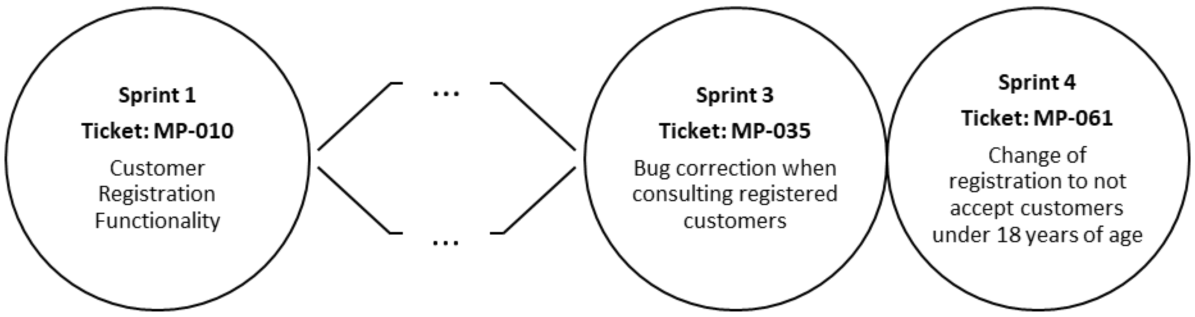
The *Wiki* could store texts, files and images on its document pages, through this feature it is possible to link prototypes that complement the documentation. In some cases, for features that have complex behaviors, written requirements alone are not sufficient for the team to build the functionality within the limits of acceptable changes, and this is where the Software Interaction overview diagram comes in.

4.4 ISSUES ANALYZED IN PREVIOUS PROJECTS

In *Jira*, the project management tool used by the company, each task, or *ticket,* as it is called, receives a unique code that makes it possible to identify it. In the projects previously carried out by the company, the documentation of the tasks was carried out in the *ticket itself*.

Suppose that in a given project, a Customer Registration functionality was developed in Sprint 1, in this way, all the documentation and requirements for performing this task, were described in the Ticket itself, *as shown in Figure 1, the* Ticket MP-010. As the project progressed, improvements and *bugs* needed to be made in this functionality. As we can see in Figure 1, the functionality underwent a *bug* fix in Sprint 3 *(Ticket MP-035) and an improved business rules change* in *Sprint 4 (*Ticket *MP-061).*

Figure 2: Example of the evolution of customer registration functionality


Sprint 1
Ticket: MP-010
Customer Registration Functionality
...
...
Sprint 3
Ticket: MP-035
Bug correction when consulting registered customers
Sprint 4
Ticket: MP-061
Change of registration to not accept customers under 18 years of age

All these documents were recorded in the system, and could be consulted by performing a Sprint filter. But a frequent problem that was noticed in the course of the projects, is that often after the tasks developed, the *Product Owner* or others involved, did not remember some business rules and improvements requested in previous *Sprints*.

For example, in *Sprint* 10, the *Product Owner* questioned the *Scrum Master* regarding the business rule of not allowing customer registration under the age of 18. As many *Sprints* of the accomplishment of this activity *passed, the Scrum Master* sometimes did not remember or even was in doubt regarding the questioned doubt. To confirm these doubts the same should look for all the tickets that changed this functionality already developed, going through all the previous *Sprints* or using search engines of the tool, but this process was not very agile.

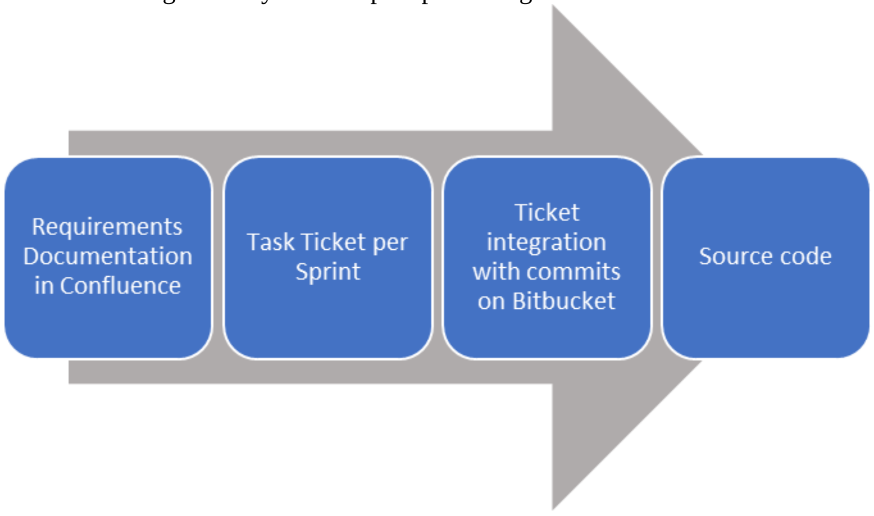
Another difficulty that could also happen was in situations when a new professional entered the project and needed to make a complex improvement in a certain functionality. As the documentation regarding the functionalities were scattered among the *Sprints,* it hindered the process of full understanding of the business rules and changes already implemented, requiring the explanation of a professional who participated in this construction process.

Based on the experience of previous projects, it is perceived that this process of search and confirmation of information was often time-consuming and exhausting, requiring a mechanism that facilitated the traceability of changes that could happen in the functionalities of the software during development.

5 CONFIGURATION AND REQUIREMENTS MANAGEMENT IN JIRA CONFLUENCE

In view of the problems faced in the management of the requirements, we sought the definition of a solution that would meet this reality and that can be summarized in Figure 2 below.

Figure 3. Cycle of steps representing the solution found


Requirements Documentation in Confluence
Task Ticket per Sprint
Ticket integration with commits on Bitbucket
Source code

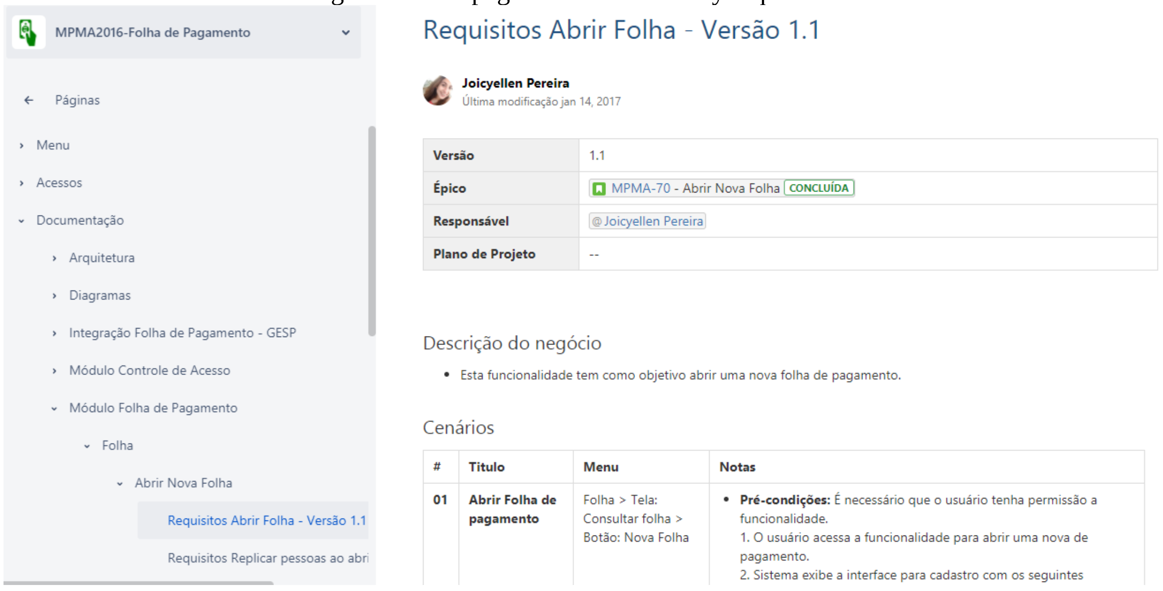
5.1 DOCUMENTATION OF REQUIREMENTS IN *CONFLUENCE*

To centralize and formalize the requirements collected, we created a specific space for the documentation of the project in *Confluence*, documenting the functionalities of the hierarchical form present in the menu of the developed system. Suppose the flow of functionality in the system is: Payroll>Open New Sheet. When a new member joins the project and they need to read the documentation of the functionality, in *Clonfluence* will be available a documentation page of the same with the versions of the functionality, as we can see in Figure 3.

Each feature receives one page per change version of the feature, to manage the changes made. The first implementation of this feature receives a page with version number, as we can see in Figure 3: Open New Sheet 1.0 and Open Sheet 1.1, where version 1.1 is the page responsible for maintaining the latest changes to this functionality.

Another advantage that we can observe is the linking of *the Ticket responsible for the existing improvements in version 1.1*, in this case the Ticket is the MPMA-70 and in which Sprint this Ticket *was contemplated, thus facilitating the integration of new members and fast traceability of Documentation by Activity.*

Figure 4. Wiki page with functionality requirements


MPMA2016-Folha de Pagamento

← Páginas

- Menu
- Acessos
- Documentação
  - Arquitetura
  - Diagramas
  - Integração Folha de Pagamento - GESP
  - Módulo Controle de Acesso
  - Módulo Folha de Pagamento
    - Folha
      - Abrir Nova Folha
        - Requisitos Abrir Folha - Versão 1.1
        - Requisitos Replicar pessoas ao abri

Requisitos Abrir Folha - Versão 1.1

Joicyellen Pereira
Última modificação jan 14, 2017

Versão	1.1
Épico	MPMA-70 - Abrir Nova Folha CONCLUÍDA
Responsável	@Joicyellen Pereira
Plano de Projeto	--

Descrição do negócio

- Esta funcionalidade tem como objetivo abrir uma nova folha de pagamento.

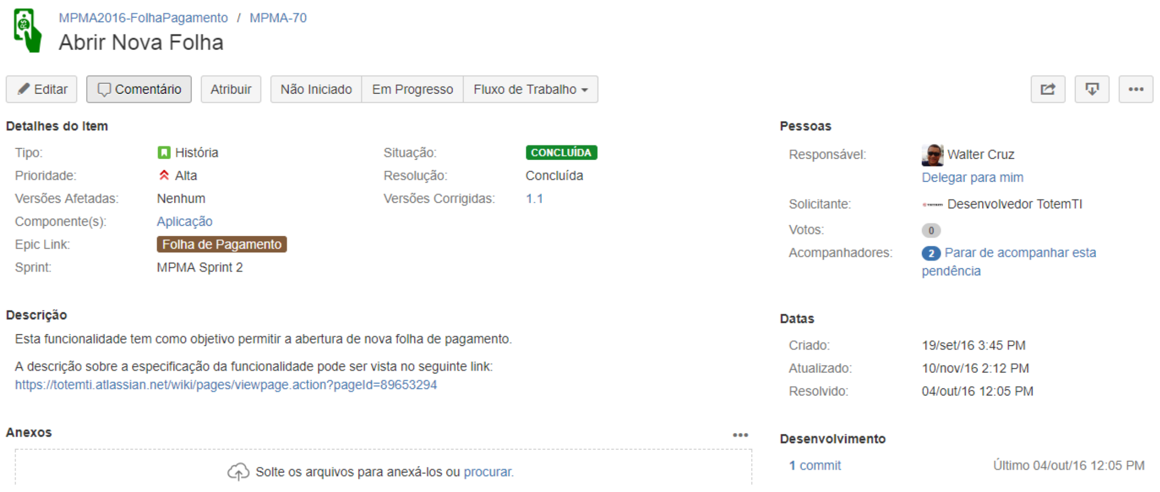
Cenários

#	Titulo	Menu	Notas
01	**Abrir Folha de pagamento**	Folha > Tela: Consultar folha > Botão: Nova Folha	• **Pré-condições:** É necessário que o usuário tenha permissão a funcionalidade. 1. O usuário acessa a funcionalidade para abrir uma nova de pagamento. 2. Sistema exibe a interface para cadastro com os seguintes

5.2 *TASK TICKET* BY SPRINT

The *tickets* of each task have the same *link* of the requirements, where the scenarios, functional prototypes, data diagrams, business rules and other information necessary for the development of the task are described. As we can analyze in Figure 4.

Figure 5. Integration of requirements between the Jira tool and the Wiki


MPMA2016-FolhaPagamento / MPMA-70
Abrir Nova Folha

Detalhes do Item

Tipo:	História	Situação:	CONCLUÍDA
Prioridade:	Alta	Resolução:	Concluída
Versões Afetadas:	Nenhum	Versões Corrigidas:	1.1
Componente(s):	Aplicação		
Epic Link:	Folha de Pagamento		
Sprint:	MPMA Sprint 2		

Descrição

Esta funcionalidade tem como objetivo permitir a abertura de nova folha de pagamento.

A descrição sobre a especificação da funcionalidade pode ser vista no seguinte link:
https://totemti.atlassian.net/wiki/pages/viewpage.action?pageId=89653294

Anexos

Solte os arquivos para anexá-los ou procurar.

Pessoas

Responsável:	Walter Cruz Delegar para mim
Solicitante:	Desenvolvedor TotemTI
Votos:	0
Acompanhadores:	2 Parar de acompanhar esta pendência

Datas

Criado:	19/set/16 3:45 PM
Atualizado:	10/nov/16 2:12 PM
Resolvido:	04/out/16 12:05 PM

Desenvolvimento

1 commit   Último 04/out/16 12:05 PM

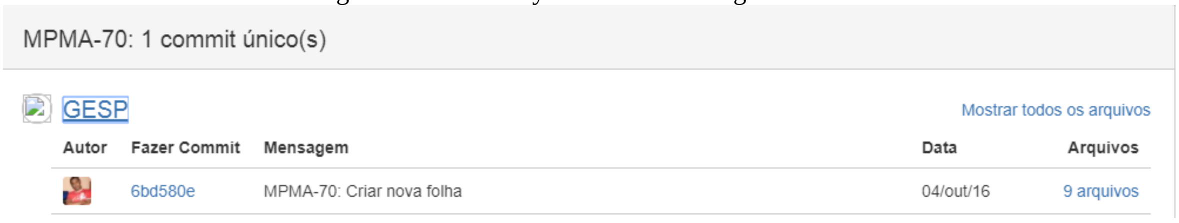
Note that in the "Description" field is the link to the page of the requirements needed to develop the functionality, in addition we can observe that the versioning of Wiki requirements, by default have the same "Fix Versions" (Version). We can verify through Figure 4, useful information about the activity: the member of the Scrum team that was responsible for the activity, how long this member took to complete the task, how many *commits* he uploaded to the repository and in which *Sprint* this functionality was contemplated.

5.3 TICKET INTEGRATION WITH *COMMITS* IN BITBUCKET

Bitbucket is a project hosting service, where through a G-it repository, the source codes are archived and versioned in an organized way and through Jira it is possible to track the *commits* related to the *Tickets* of the tasks developed.

For this to happen, the team member simply uploads their *commit* to the repository with the default name of the project, accompanied by the corresponding *Ticket* number . In the example used, this nomenclature would be MPMA-70, as shown in Figure 5.

Figure 6. Traceability of *commits* through Jira


MPMA-70: 1 commit único(s)

GESP

Mostrar todos os arquivos

Autor	Fazer Commit	Mensagem	Data	Arquivos
	6bd580e	MPMA-70: Criar nova folha	04/out/16	9 arquivos

5.4 RESULTS ACHIEVED

After identifying a deficiency in the tracking of requirements and in the form of versionaries, we implemented a standard using the tools mentioned during the article, gaining agility in the tracking of requirements.

Previously, the requirements were fixed in the activities themselves, thus generating a decentralization of information. In this way, if a new team member or the *Product Owner* himself needed to visualize a business rule, a search would have to be made for the description of each *Ticket* until he found the necessary information. With the centralization and organization of the requirements in pages structured in a hierarchical way, agility was gained in the process of searching for information because, once the page containing the requirements is available, all those involved (including *the Product Owner*) will be able to view it centrally and solve the business doubts about the functionality. In addition to being able to check more information, such as *the Tickets* that are linked to the task, which Sprint was performed this *Ticket* and the member responsible for developing the functionality.

Through the formalization of requirements, a transparent and agile understanding of the requirements and processes was obtained, adding value to the business and the development environment, increasing the productivity and quality of the product delivered to the customer, as well as avoiding inconsistency between the parties. These benefits were maximized by the fact that the project analyzed in the study was characterized as a distributed team and enabled the broad traceability of requirements for it.

6 CONCLUSION

This proposal aimed to enhance the integration and traceability between Scrum processes, from requirements gathering to source code.

With the growth of the use of the Scrum methodology, we can see that support tools have emerged that have a fundamental role in the distributed teams, because through them, it was possible to versioning and maintain a traceability of the documents as is done with the source code in a *git repository.*

One of the great challenges encountered was to keep the pillars of Scrum intact even with the geographical distribution of the parties involved, in addition to ensuring the interaction of those involved in the project through the tools. With the versioning and standardization of the requirements, we were able to maintain the consistency of the information, the rapid tracking of information about certain functionality and with this we achieved the objective of reducing costs and amplifying efficiency and quality from the beginning until the completion of the project.

REFERENCES

Bissi, Wilson. SCRUM - METODOLOGIA DE DESENVOLVIMENTO ÁGIL 2007. Disponivel em < http://revistas.bvs-vet.org.br/campodigital/article/view/30944/33947>. Acessado 06/09/2017.

Collins, Eliane F. et al. Experiência em Automação do Processo de Testes em Ambiente Ágil com SCRUM e ferramentas OpenSource. Disponível em < http://www.lbd.dcc.ufmg.br/colecoes/sbqs/2010/RL4_eliane_collins.pdf> Acessado 03/09/2017.

Cruz, Fábio. Scrum e PMBOK unidos no Gerenciamento de Projetos, 1 ed. Rio de Janeiro, Brasport Livros e Multimídia Ltda 2013.

Dimes, Troy. Scrum Essencial, Babelcube inc 2014.

Lima, Eleandro Lopes de, et al. SCRUM: UMA DAS METODOLOGIAS ÁGEIS MAIS USADAS DO MUNDO. Disponível em <http://revistapensar.com.br/tecnologia/pasta_upload/artigos/a64.pdf>. Acessado 15/10/2017.

Machado, Marcos, et al. SCRUM – Método Ágil: uma mudança cultural na Gestão de Projetos de Desenvolvimento de Software 2009. Disponível em <http://uniesp.provisorio.ws/fagu/revista/downloads/edicao12009/Artigo_5_Prof_Marcos.pdf>. Acessado 05/10/2017.

MARÇAL, Ana Sofia Cysneiros, et al. Estendendo o SCRUM segundo as Áreas de Processos de Gerenciamento de Projetos do CMMI 2007. Disponível em <https://inf.ufes.br/~monalessa/PaginaMonalessa-NEMO/ES_Mestrado/Artigos/EstendendoScrumSegundoAreasDeGerenciaNoCMMI.pdf>. Acessado 05/10/2017.

Mendonça, (2012) Ricardo Augusto Ribeiro de. Levantamento de requisitos no desenvolvimento ágil de software. Disponivel em < http://www.cpgls.pucgoias.edu.br/7mostra/Artigos/AGRARIAS%20EXATAS%20E%20DA%20TERRA/Levantamento%20de%20requisitos%20no%20desenvolvimento%20%C3%A1gil%20de%20software.pdf >. Acessado 20/10/2017.

Molinari, Leonardo. Testes funcionais de software. Visual Books 2008.

Pedro, Denis et al. O guia Passo-a-Passo para implantar o Scrum em seu próprio projeto. Disponivel em < http://www.mindmaster.com.br/e-book-como-implantar-scrum/>. Acessado 13/10/2017.

Sabbagh, Rafael. Scrum Gestão Ágil para Projetos de Sucesso, Casa do código 2013.

Santos, Alinne C Corrêa et al. Experiência Acadêmica de uma Fábrica de Software.

Star, David. Scrum distribuído. Disponivel em < https://msdn.microsoft.com/pt-br/library/jj620910(v=vs.120).aspx#bkmk_degrees>. Acessado 20/10/2017.

Silva, Daisy Eliana dos Santos, et al. METODOLOGIAS ÁGEIS PARA O DESENVOLVIMENTO DE SOFTWARE: APLICAÇÃO E O USO DA METODOLOGIA SCRUM EM CONTRASTE AO MODELO TRADICIONAL DE GERENCIAMENTO DE PROJETOS 2013. Disponivel em < http://revistas.ung.br/index.php/computacaoaplicada/article/view/1408/1194>. Acessado 04/09/2017.

Silva, Edson. Scrum e TFS: uma abordagem prática, 1 ed. Rio de Janeiro, Brasport Livros e Multimídia Ltda 2017.

Silva, F. G. et al. Uma análise das Metodologias Ágeis FDD e Scrum sob a Perspectiva do Modelo de Qualidade MPS.BR 2009. Disponível em < https://www.scientiaplena.org.br/sp/article/view/678>. Acessado 04/09/2017.

Soares, Michel dos Santos. Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software. Disponível em < http://www.periodicosibepes.org.br/index.php/reinfo/article/view/146> Acessado 03/09/2017.

Silva, Ricardo Pereira e. Como Modelar com UML 2. Editora Visual Books 2009.

Silva, Vinicius Bernardo, et al. Experiências de ensino a distância do gerenciamento ágil de projetos com Scrum e apoio de uma ferramenta para gerência de histórias de usuário. Disponível em < http://seer.upf.br/index.php/rbca/article/view/5614>. Acessado 20/10/2017.

Sutherland, Jeff et al. Guia do Scrum 2013. Disponivel em < https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf>. Acessado 05/09/2017.

Utilizando Scrum no Desenvolvimento de Software. Disponivel em < https://s3.amazonaws.com/academia.edu.documents/5851220/wbma2010.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1507877080&Signature=mmq2%2B42KAFfVT7VDtL99xqYSFYE%3D&response-content-disposition=inline%3B%20filename%3DConducting_an_Architecture_Group_in_a_Mu.pdf#page=93>. Acessado 13/10/2017.

Versionone (2008) “The State of Agile Development Survey Results”. Disponível em <http://www.versionone.com/pdf/3rdAnnualStateOfAgile_FullDataReport.pdf>. Acessado: 04/09/2017.

Vieira, Wagner Leite. PROPOSTA DE UMA ARQUITETURA PARA APLICAÇÕES DE APOIO ÀS ATIVIDADES DO SCRUM BASEADA EM SERVIÇOS 2016.