REVISTA DE GESTÃO E SECRETARIADO
MANAGEMENT AND ADMINISTRATIVE PROFESSIONAL REVIEW
ISSN: 2178-9010

Revista GeSec
São Paulo, SP, Brasil v. 14, n. 11, p. 20568-20585,2023

DOI: http://doi.org/10.7769/gesec.v14i11.2952

Gerenciamento e planejamento de projetos de software usando metodologias ágeis: um estudo de caso

Software project management and planning using agile methodologies: a case study

Daniela Milagros Quenaya Mendoza[1]

Nélio Domingues Pizzolato[2]

Resumo

Entregar software com qualidade, nos prazos previstos e respeitando os custos é um grande desafio e oportunidade para empresas deste setor. A constante procura pela produtividade e qualidade em projetos de software deu origem a diversos modelos de gerenciamento que procuram auxiliar esta área com metodologias e formas de organização de trabalho. O modelo mais conhecido na área de gerenciamento de software é o "Modelo em Cascata", caracterizado pela rigidez e fases bem definidas nos processos. Uma vez encontrados problemas com os modelos rígidos surgiram como alternativa de gerenciamento os métodos ágeis, caracterizados pela agilidade, iteratividade e adaptabilidade para diversos projetos. O objetivo deste artigo é analisar o problema do gerenciamento de projetos de desenvolvimento de software e propor um modelo de referência considerando a abordagem tradicional e a abordagem ágil. A pesquisa analisa a compatibilidade das metodologias ágeis com um modelo de maturidade nos processos de desenvolvimento de software. Finalmente, a pesquisa apresenta um estudo de caso, o qual permitiu observar as forças e fraquezas dos métodos estudados, concluindo que as práticas ágeis contribuem na melhoria e gerenciamento de projetos de software.

[1] Mestre em Engenharia de Produção, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Gávea, Rio de Janeiro – RJ. E-mail: danielaquenaya@gmail.com Orcid: https://orcid.org/0009-0005-9393-3372
[2] Doutor em Administração de Empresas, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Gávea, Rio de Janeiro – RJ. E-mail: ndp@puc-rio.br

Palavras-chave: Modelo Referência. Gerenciamento de Projetos. Software. Abordagem Ágil.

Abstract

Today, delivering quality software, on time and respecting costs is a key differentiator for companies dedicated to this activity. The constant search for productivity and quality in software project management led to several models that seek to help this area with methods and forms of work organization. The best-known model in software management is the "Waterfall Model", characterized by stiffness and well-defined phases in the process. Once found problems with rigid models have emerged as alternative management of agile methods, characterized by agility, and adaptability to various iterative projects. Given that this paper analyzes the problem of managing software development projects, and proposes a reference model for managing software projects considering the traditional approach of project management as well as the agile approach. The research examines the compatibility of agile methodologies with a maturity model in software development processes. Finally, the research presents a case study, which allowed us to observe the strengths and weaknesses of the methods studied, concluding that such practices contribute to improved and agile project management software.

Keywords: Reference Model. Project Management. Software. Agile Approach.

Introdução

A gestão de produtos software têm evoluído com o passar dos anos. A metodologia tradicional como o Modelo em Cascata foi largamente utilizada, inspirado no gerenciamento da produção na manufatura, já que trabalham com processos claros e definidos e sugerem que o desenvolvimento de software seja feito numa sequência de fases bem claras (Teles, 2004). Estas metodologias são aplicadas principalmente em situações onde os requisitos dos projetos são estáveis.

Na área de desenvolvimento de software, além de existirem projetos com características estáveis e conhecidas, existem projetos com muita variabilidade nos requisitos. Diante disso surgem como alternativa de gerenciamento as metodologias ágeis, que de acordo com Martíns (2007) são metodologias com controle empírico de processo, e são indicadas nas situações onde as entradas do processo são variáveis, e o processo é complexo para produzir

resultados similares. No caso especifico de empresas de pequeno porte dedicadas ao desenvolvimento software, observa-se muitas dificuldades no gerenciamento de projetos, as quais originam adiamentos nas entregas dos projetos, custos elevados, trabalho excessivo, retrabalho, falta de qualidade no software entre outros problemas.

O objetivo deste artigo é estudar as principais metodologias ágeis e tradicionais de gerenciamento de projetos, boas práticas de desenvolvimento de software, de modo tal que se possa apresentar uma proposta de modelo de referência para o gerenciamento de projetos de software para um caso específico, mas que possa ser adaptado em empresas com características similares aquelas do caso estudado em particular.

O artigo está organizado da seguinte maneira. Após esta seção de introdução, na seção 2 são apresentadas as metodologias tradicionais e ágeis de gerenciamento de projetos de software. Na seção 3 é apresentado o modelo de referência proposto. Na seção 4 é apresentado um estudo de caso com uma empresa de pequeno porte, na qual aplicou-se o modelo proposto na seção 3. Na seção 5 são apresentadas as conclusões da pesquisa.

Revisão Bibliográfica

As metodologias tradicionais são conhecidas também como metodologias pesadas ou orientadas à documentação. No passado, o custo de corrigir e fazer alterações era muito alto, já que o acesso aos computadores era limitado e não existiam ferramentas como depuradores e analisadores de código que pudessem ajudar ao desenvolvimento do software. Por isso, o software era planejado e documentado antes de ser implementado (SANTOS e SOARES, 2004). Teles (2004) usa o termo de metodologia tradicional, ou desenvolvimento tradicional, para se referir aos projetos de software que se baseiam no desenvolvimento em cascata.

O modelo em cascata foi proposto na década de 1970 por ROYCE (1970) e surgiu pela necessidade de controlar grandes projetos de software. O modelo em cascata é estruturado, sequenciado e direcionado à documentação em todas as atividades realizadas ao longo do desenvolvimento. Royce (1970) e Teles (2004) descrevem as seis etapas do modelo em cascata, sendo elas: a) análise: a equipe faz o levantamento dos requisitos do projeto e busca compreendê-los detalhadamente; b) desenho: a equipe projeta a arquitetura do sistema considerando a análise; c) implementação: a equipe implementa as diferentes partes de *software*, considerando a análise e o *desenho; d)* teste: a equipe testa o *software* para verificar se o sistema atende às necessidades especificadas pelo usuário; e) implantação: O sistema é colocado em produção e os usuários finais passam a utilizá-lo; f) manutenção: O *software*

poderá sofrer diversas alterações, como correções, inclusão de novas funcionalidades até o final da sua vida.

As metodologias ágeis surgiram com o objetivo de melhorar e criar uma alternativa ao processo de desenvolvimento de *software* tradicional (COHEN *et al.*, 2003). A Agile Aliance (2009) descreve as metodologias ágeis como uma forte colaboração entre a equipe de programadores e os clientes; uma comunicação face-a-face (considerada mais eficiente do que documentação escrita); entrega de *software* frequente que agregue valor; equipes com um bom relacionamento e auto organizáveis; e formas de manter cuidadosamente o código e a equipe, de modo tal que modificações drásticas nos requisitos não sejam consideradas uma crise para o projeto em desenvolvimento.

O Manifesto Ágil (2009) destaca como as principais metodologias ágeis de gerenciamento de projetos de software a *Extreme Programming*, *Scrum* e *Feature Drive Development.* A metodologia Scrum foi criada por Ken Schwaber e Jeff Sutherland em 1996, como um método de gerenciamento que aceita que o desenvolvimento de *software* seja imprevisível e aplicável a projetos com muitas mudanças (SCHWABER, 2004). Na Tabela 1 são apresentados os elementos componentes do Scrum.

Sprints	É uma série de iterações bem definidas.
Time-box	E a duração do sprint de duas a quatro semanas.
Sprint Planning Meeting	Realiza-se reunião de planejamento onde o time de desenvolvedores tem contato com o cliente.
Product Owner	E o cliente quem prioriza o trabalho que precisa ser feito.
Execução da *Sprint*	Fazer o *Sprint.*
Daily Meeting	Reuniões diárias rápidas - não mais de 15 minutos de duração.
Sprint Burndown	Gráfico de avanço de tarefas usado nos *Daily Meeting.*
Sprint Review	Reunião de Revisão de trabalho, onde se valida se o objetivo foi atingido.
Sprint Retrospective	Reunião de lições aprendidas

Tabela 1 – Elementos no Scrum
Fonte: Adaptado Kniberg (2007)

O gerenciamento de projetos é a aplicação do conhecimento, habilidades, ferramentas e técnicas às atividades do projeto, a fim de atender aos seus requisitos. O gerenciamento de projetos é realizado através da aplicação e da integração dos seguintes processos de gerenciamento de projetos: Iniciação, Planejamento, Execução, Monitoramento e Controle, e Encerramento.

Metodologias Tradicionais	Metodologias Ágeis
São preditivas: Cada etapa de desenvolvimento do projeto é baseada na etapa anterior, só funciona bem se a concepção do sistema não sofre nenhuma alteração	**São adaptativas**: Partem do princípio de que o conhecimento sobre o problema aumenta à medida que o sistema vai sendo desenvolvido, levando a uma busca constante por melhores soluções.

Orientadas a processos: Processos bem definidos devem ser impostos e executados, para garantir a qualidade do produto resultante.	**Orientadas a pessoas:** Pessoas devem ser tratadas como indivíduos e não como recursos, pois cada pessoa apresenta um ritmo de trabalho único e completamente diferente, que é fruto da sua vivência pessoal
São rígidas: Pressupõem que é possível especificar de antemão todos os requisitos do *software* a ser desenvolvido.	**São flexíveis e iterativas:** Adaptam-se constantemente ao conjunto de requisitos mais atual: a cada iteração, usuários, clientes e desenvolvedores decidem sobre quais características devem ser adicionadas, modificadas e até retiradas do sistema.
São burocráticas: Geram muita sobrecarga no desenvolvimento a ser realizado, comprometendo a velocidade de desenvolvimento e, muitas vezes, o sucesso do projeto.	**Buscam constantemente a simplicidade**: Partem do princípio de que é preferível fazer algo simples de imediato e pagar um pouco mais para alterá-lo depois, se necessário

Tabela 2 – Diferenças entre as metodologias tradicionais e ágeis
Fonte: Adaptado de Magalhães, Vasconcelos e Roullier (2005)

Modelo Proposto

O modelo proposto para realizar a gestão ágil de projetos de software inicia com a geração de uma ideia ou a solicitação de um cliente para desenvolver algum produto de *software* específico. Em seguida, realiza-se uma análise inicial da oportunidade, categorizando os diversos projetos, para que a empresa mantenha uma carteira de projetos, conforme recomendação da PMBOK. Posteriormente procede-se com a avaliação dos projetos da carteira, se faz a seleção, priorização e finalmente se procede à autorização para início do desenvolvimento do projeto (Figura 1).

O gerente de projetos é a pessoa responsável pela realização dos objetivos do projeto (PMBOK, 2008). Magalhães, Vasconcelos e Roullier (2005) descrevem as principais diferenças entre as metodologias ágeis e tradicionais mostradas na Tabela 2.

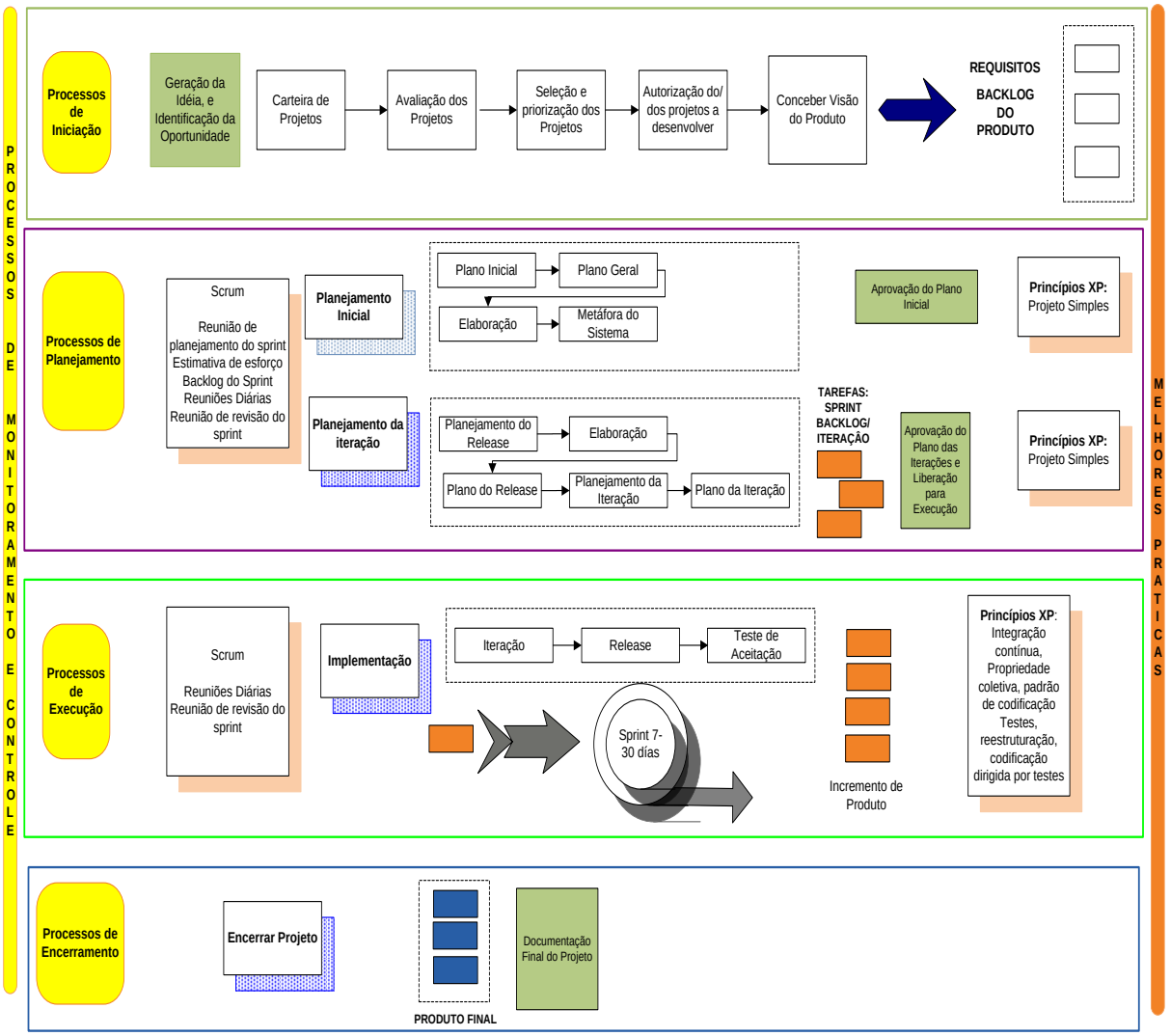

Figura 1 – Modelo de Gerenciamento de Projetos de Software
Fonte: Elaboração própria

Uma vez que o projeto é autorizado, procede-se com a inicialização para declarar a **visão inicial** do produto de *software* e obter a lista de funcionalidades com as premissas e os requisitos principais do projeto. Recomenda-se fazer as seguintes perguntas para declarar a visão inicial do produto: Qual é a visão que o cliente tem do produto a desenvolver? Que visão a equipe tem do produto a desenvolver? Quais são as limitações e restrições do projeto? A empresa tem a equipe certa para desenvolver o projeto? Em paralelo, procede-se com a estimação do orçamento do projeto, assim como os recursos necessários para preparar o ambiente de desenvolvimento (Figura 2).

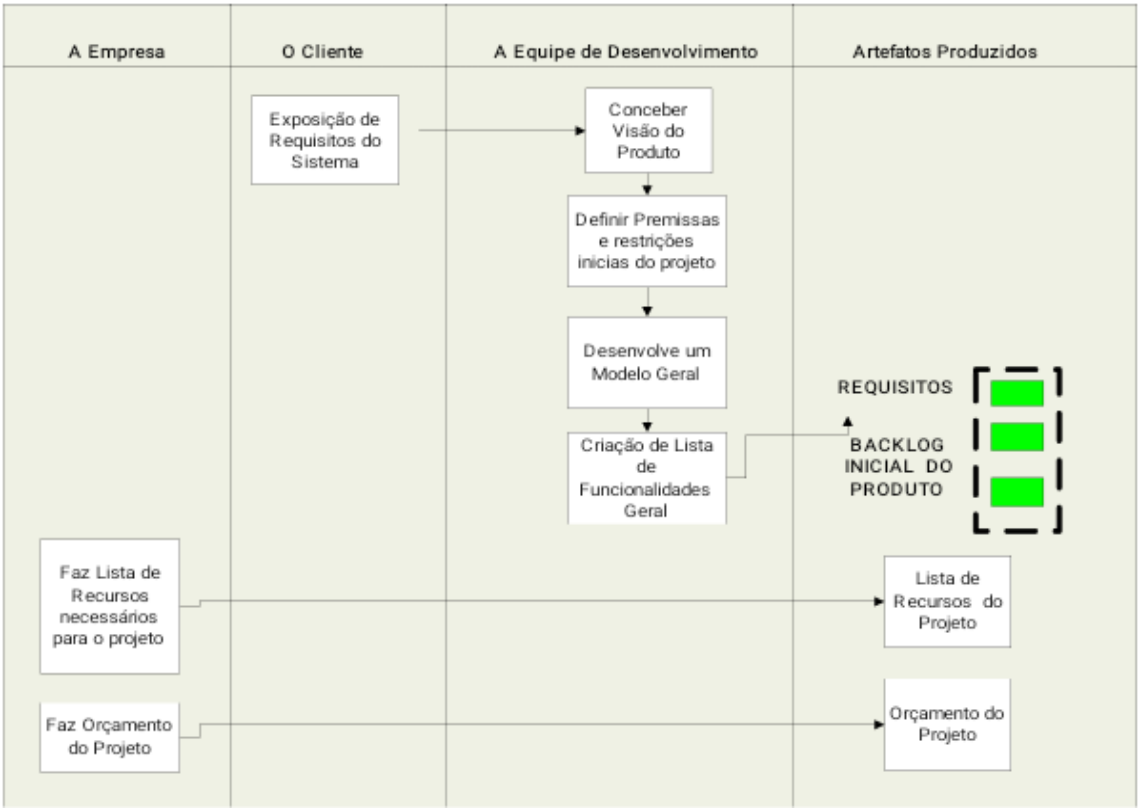

Figura 2 – Visão Inicial do Projeto
Fonte: Elaboração Própria

Em relação à análise de recursos se fará análise do ambiente de desenvolvimento, onde se analisará o ambiente atual disponível e serão listados os diferentes requerimentos necessários para poder desenvolver o projeto. Em seguida, realiza-se a análise da necessidade de pessoal, identificando a equipe necessária para desenvolver o projeto. A Figura 3 detalha os elementos que devem ser definidos nesta etapa.

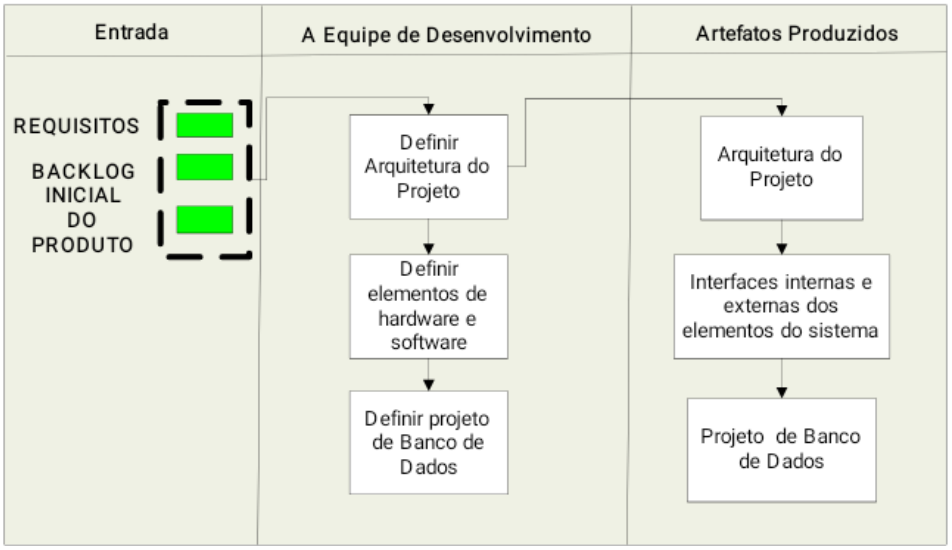

Figura 3 – Arquitetura Inicial do Projeto
Fonte: Elaboração Própria

Uma vez aprovado o projeto e gerada a visão inicial, passa-se à fase do planejamento e, considerando a metodologia XP, o planejamento será dividido em duas etapas: Planejamento Geral e Planejamento Detalhado. Na integração contínua dos requisitos do projeto serão definidos hardware, *software* e as diversas ferramentas necessárias para integrar o projeto. Na Figura 4 são apresentados as atividades da equipe de desenvolvimento e os artefatos que deverão ser produzidos nesta etapa.

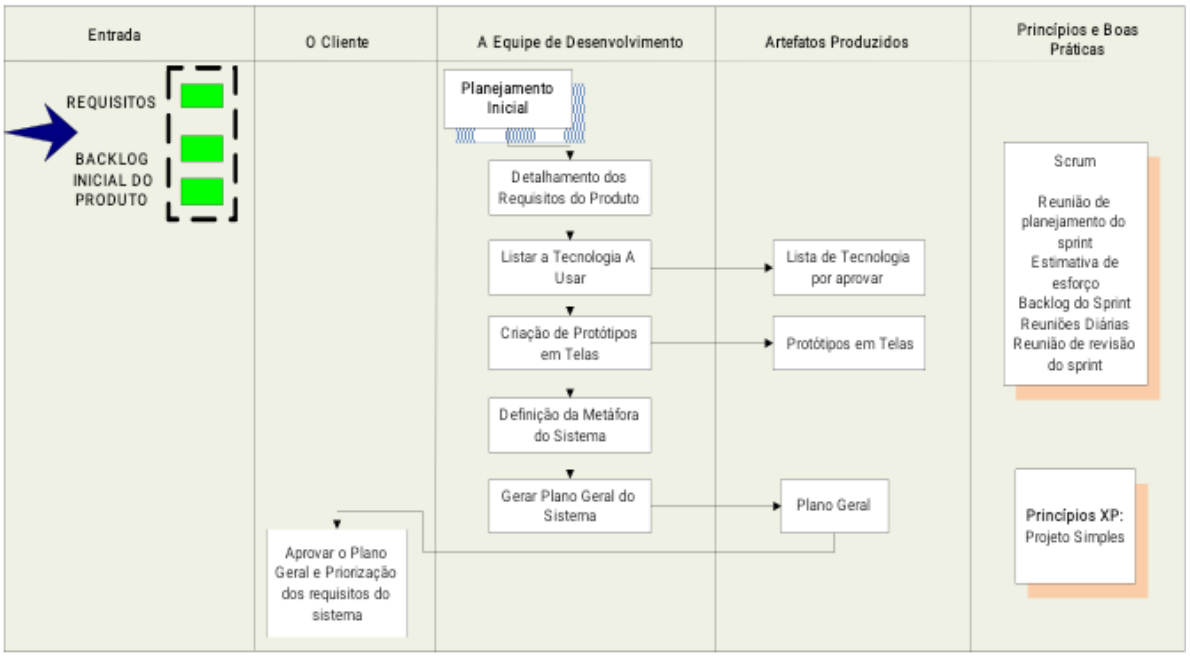

Figura 4 – Planejamento Inicial
Fonte: Elaboração Própria

Na etapa de planejamento a equipe de desenvolvimento trabalha com o cliente para priorizar os requisitos definidos inicialmente. Considerando a metodologia Scrum criam-se as iterações, fazendo uma seleção dos requisitos de projeto mais importantes do *Product Backlog*. Uma vez priorizados os requisitos pelo cliente a equipe de desenvolvimento deve fazer uma estimativa de esforço necessário em tempo para produzir cada um dos requisitos. É importante ressaltar a importância do trabalho em conjunto com o cliente já que considerando tanto *Scrum* quanto XP, é o cliente quem vai exigir quais requisitos serão desenvolvidos em cada iteração de desenvolvimento. Finalmente o gerente de projeto deverá gerar um documento com os nomes das iterações, a lista e estimativas dos requisitos de projeto, e os testes de integração para as iterações a desenvolver.

Procede-se a implantar a lista das iterações definidas na fase do planejamento detalhado. Com ajuda dos princípios XP e *Scrum*, trabalha-se cada iteração, fazendo os

releases, e os testes de aceitação para ir incrementando o projeto de *software* que se está desenvolvendo. A Figura 5 apresenta o fluxo de execução do projeto.

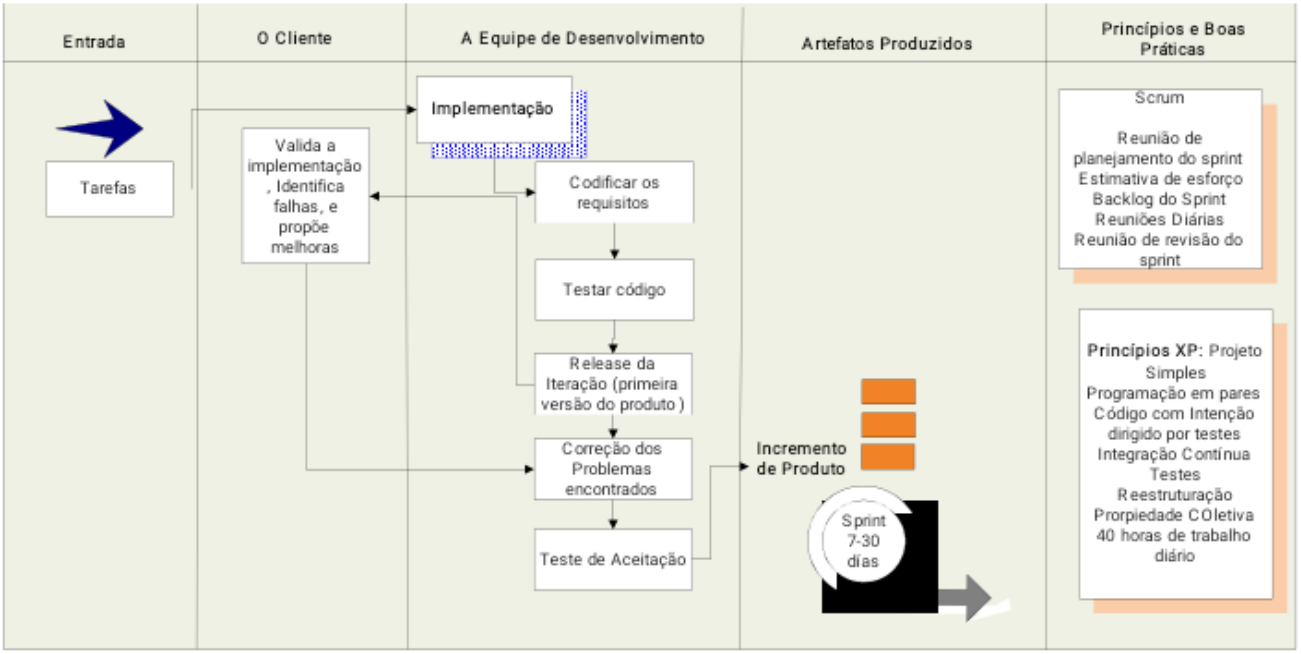

Figura 5 – Execução do Projeto
Fonte: Elaboração Própria

Uma vez terminado o desenvolvimento das iterações e feitas as integrações e testes do projeto procedesse à etapa do encerramento. Teles (2004) afirma que não existe um conjunto de documentos que possam atender as necessidades de todos os projetos em geral, já que cada um dos projetos tem características específicas. No entanto, ele cita um conjunto de documentos que podem ser utilizados para a realidade de diferentes projetos de desenvolvimento de *software*: Estória, Testes de aceitação, Testes de Unidade, Modelo de classes, Modelo de dados, Processo de Negócio, Manual do Usuário, Acompanhamento diário, Acompanhamento do projeto.

Na fase de monitoramento e controle do projeto o gerente deve definir os requisitos que satisfaçam as especificações dadas pelo cliente, definir a equipe com as suas responsabilidades, desenvolver os diversos procedimentos e padrões do projeto, e monitorar o desempenho do projeto no longo da sua duração.

Estudo de Caso

Foi avaliado o trabalho de uma empresa de desenvolvimento de software de pequeno porte, que não utiliza metodologias ágeis, predominando o uso do modelo tradicional em

cascata. Não existe controle sobre o trabalho dos funcionários nos projetos que se desenvolvem, o que ocasiona adiamentos nas entregas, retrabalho, e mudanças nos orçamentos dos projetos. A equipe de desenvolvimento é formada por dez pessoas, um coordenador, que é responsável de direcionar os projetos da empresa e uma equipe de nove desenvolvedores de software. Cada vez que a equipe inicia um novo projeto, o coordenador principal faz o planejamento dos projetos da empresa. São três os tipos de projetos de software desenvolvidos: soluções sob encomenda, soluções pré-formatadas e soluções embarcadas.

Para avaliar o modelo proposto, três pessoas da equipe de desenvolvimento foram entrevistas. Estas pessoas são denominadas por Funcionário A (FA), Funcionário B (FB), e Funcionário C (FC). O tipo de projeto selecionado foi de soluções embarcadas.

A Tabela 2 apresenta a equipe faz uma lista dos diversos projetos a desenvolver. A avaliação dos projetos foi feita por três funcionários A, B, e C. A nota de avaliação para cada projeto foi realizada utilizando uma escala entre 0 a 10, sendo 10 um projeto com maior importância. Para obter uma avaliação agregada dos funcionários, calculou-se a média das três avaliações.

		FA	FB	FC	
Categoria	Projeto	Nota	Nota	Nota	Média
ERP	Dados de recursos humanos (informações)	10	6	7	7,67
	Dados dos tickets	10	6	9	8,33
	Indicadores de desempenho	9	7	8	8,00
Página web	Web site	6	6	6	6,00
	Aplicativos sob encomenda	5	2	8	5,00
	Download de arquivos	7	2	7	5,33
Aplicativos móveis	Aplicativos Iphone	5	2	8	5,00
Analise Econômica	Demo de Análise Econômica de Projetos de E&P	9	9	10	9,33
Gestão de conhecimento	Gestão de arquivos eletrônicos	10	7	7	8,00
Gestão de Riscos	Riscos	10	10	10	10,00
Intranet	Intranet	9	5	10	8,00
Sistemas de Comunicação	Chats	8	5	7	6,67
	E-mail	8	5	7	6,67
	Videoconferência	9	3	8	6,67

Tabela 2 – Lista da carteira de projetos
Fonte: Elaboração Própria

O projeto com maior nota foi “Riscos”, o qual foi selecionado para desenvolvimento. Em seguida, o *SCRUM Master* preparou a primeira visão inicial do projeto. Neste caso específico, o projeto “Riscos” é uma ferramenta de software que visa detectar e controlar

riscos empresariais, considerando a norma ISO 31.000 de Gestão de Riscos. Em seguida, partiu-se para a definição dos requisitos e do *backlog* dos processos do produto, sendo eles: Estabelecer o contexto; Identificação de Riscos; Análise de Riscos; Avaliar Riscos e Tratamento de Riscos.

Logo após, realizou-se a definição da equipe à ser alocada ao projeto (Tabela 3). Neste caso, o funcionário A, que possui conhecimento da Norma da ISO foi alocado como líder do projeto, para ser o *Scrum Master*.

Equipe	**Papel desempenhado**	**Responsável**
TI	Técnico	FC
TI	Acompanhador	FB
TI	Facilitador	FA
TI	Arquiteto	FC
Cliente	Contador de histórias	FA
Cliente	Aceitantes	FA
Cliente	Proprietário do produto	FA
Cliente	Planejadores	FA
Cliente	Chefe	FA

Tabela 3 – Descrição Genérica do Projeto Riscos
Fonte: Elaboração Própria

Uma vez aprovado o projeto e gerada a visão inicial passa-se à fase do planejamento e, como a metodologia proposta considera a metodologia XP, o planejamento será dividido em duas etapas: Planejamento Geral e Planejamento Detalhado. Na etapa de planejamento inicial o FA (Scrum Master), apresentou o *Backlog* do Produto (visão inicial) e toda a equipe fez uma estimativa inicial em tempos, considerando as prioridades da primeira iteração dos processos (Tabela 4). Identificaram-se os responsáveis por cada processo.

Processo	**Tempo em horas**	**Responsável**
Estabelecer o contexto	52	FC
Identificação de Riscos	44	FB
Análise de Riscos	20	FA
Avaliar Riscos	20	FB
Tratamento de Riscos	20	FA

Tabela 4 – Planejamento geral do Projeto Riscos
Fonte: Elaboração Própria

Uma vez identificados os processos, fez-se uma reunião para definir as diversas tarefas que seriam desenvolvidas por cada processo da primeira iteração, se identificou os itens com maior prioridade para desenvolver, os itens que se podiam desenvolver em paralelo, e as horas de trabalho para cada um dos itens identificados, assim como os responsáveis por cada processo. A Tabela 5 resume a reunião final de planejamento que a equipe teve, onde se

considera: Prioridade mais importante como 1, Prioridade média 2, e Prioridade baixa 3. Para o caso específico a equipe não considerou tarefas de trabalho em paralelo.

O projeto Riscos foi desenvolvido em iterações, como é proposto nas metodologias estudadas, já que esta abordagem permite ter um melhor controle dos requisitos do sistema, e facilita conhecer o que acontece durante o desenvolvimento do projeto. A execução do projeto foi dividida em *sprints* ou iterações com reuniões diárias, semanais e mensais. Cada dia à primeira hora se avaliou o trabalho do dia anterior, refinando os requisitos e definindo as tarefas à serem realizadas no dia, numa reunião de aproximadamente 10 minutos. Cada sexta-feira foi realizada uma reunião semanal com duração entre 30 a 45 minutos, na qual se avaliava a iteração semanal e se definiam os requisitos a desenvolver para as semanas posteriores. No final de mês se avaliou o *sprint* mensal, onde a primeira iteração do projeto deveria estar concluída.

Para o caso do projeto em desenvolvimento Riscos, foram consideradas as seguintes práticas de desenvolvimento ágeis de XP:

- Comunicação com o contador de estórias (requisitos), neste caso com o FA, já que ele é quem atua como cliente.
- Programação em pares, já que um programador ajuda ao outro a manter o foco na tarefa, e se ajudam a fazer *brainstorm* e melhorar os processos. Com a programação em pares no projeto, tanto FB e FC se ajudaram mutuamente a não perder o foco das práticas de gerenciamento da equipe.
- Foram aplicados os testes antes da programação, com o objetivo de localizar os erros o mais cedo possível.
- *Refactoring*, com o objetivo de melhorar o programa e poder evoluir com segurança. O *refactoring* ajudou a melhorar o código-fonte.
- Padrões para nomenclatura dos artefatos em desenvolvimento, e as variáveis, banco de dados e classes. Em relação a esta prática a equipe definiu algumas regras de nomenclatura.
- Cada palavra de uma variável tinha que começar com letra maiúscula.
- Os nomes, tanto das variáveis, tabelas, classes, arquivos e outros devem ser compostos por substantivo e se é necessário por um complemento mais.
- Todas as classes tinham que antepor o prefixo “c”, exemplo: “cRisco”
- Inspeção do código-fonte, com o objetivo de garantir a qualidade do projeto e detectar os defeitos no programa. Para o caso em estudo a inspeção do código foi feita para as partes mais complexas do projeto.

Processo 1: Estabelecer o contexto	Prioridade	Tempo em horas	Responsável
Pesquisar sobre Ribbons	1	8	FC
Implementação das tabelas do organograma	1	4	FC
Implementação das tabelas dos usuários	1	4	FC
Controle para organograma	3	8	FB
Programar UI para usuários	2	8	FB
Tabelas para os parâmetros	2	8	FB
Controles para os ingressos de parâmetros	3	8	FB
Normalização dos parâmetros	2	4	FA
Processo 2: Identificação de Riscos	**Prioridade**	**Tempo em horas**	**Responsável**
Ribbons	1	4	FC
Mostrar organograma	1	4	FC
Adicionar objetivo	1	4	FB
Adicionar risco	1	4	FB
Mostrar todos os riscos	2	4	FB
Modificar riscos	1	4	FB
Eliminar riscos	1	4	FB
Gerar vistas personalizadas	2	8	FB
Exportar dados do Excel	1	8	FC
Processo 3: Análise de Riscos	**Prioridade**	**Tempo em horas**	**Responsável**
Ribbons	1	4	FA
Vistas dos riscos	2	4	FA
Gráficos do porta folio	1	4	FA
Controles personalizados	3	8	FA
Processo 4: Avaliar Riscos	**Prioridade**	**Tempo em horas**	**Responsável**
Filtrar Riscos	1	4	FB
Gráficos personalizados	1	8	FB
Parâmetros de curvas	1	4	FB
Parâmetros de Limites	1	4	FB
Processo 5: Tratamento de Riscos	**Prioridade**	**Tempo em horas**	**Responsável**
Aceitar o risco	1	4	FA
Evitar o risco	1	4	FA
Reduzir o risco	1	4	FA
Compartilhar o risco	1	4	FA
Gerar documento Word dos informes dos riscos	1	4	FA
	Total	**156**	

Tabela 5 – Planejamento detalhado da Equipe
Fonte: Elaboração Própria

Para conseguir executar e monitorar o avanço do trabalho do projeto foi necessário implantar um sistema de controle de tarefas para a equipe de trabalho. Para o caso específico a empresa teve que criar uma ferramenta de *software* chamada *LoginX*. Com esta ferramenta foi possível controlar o trabalho dos funcionários, já que *LoginX* registra as tarefas e as horas empregadas em desenvolver cada tarefa ou iteração do planejamento. O *Scrum Master* controlou cada uma das tarefas da equipe com ajuda de uma planilha que resumia os dados gerados pela ferramenta.

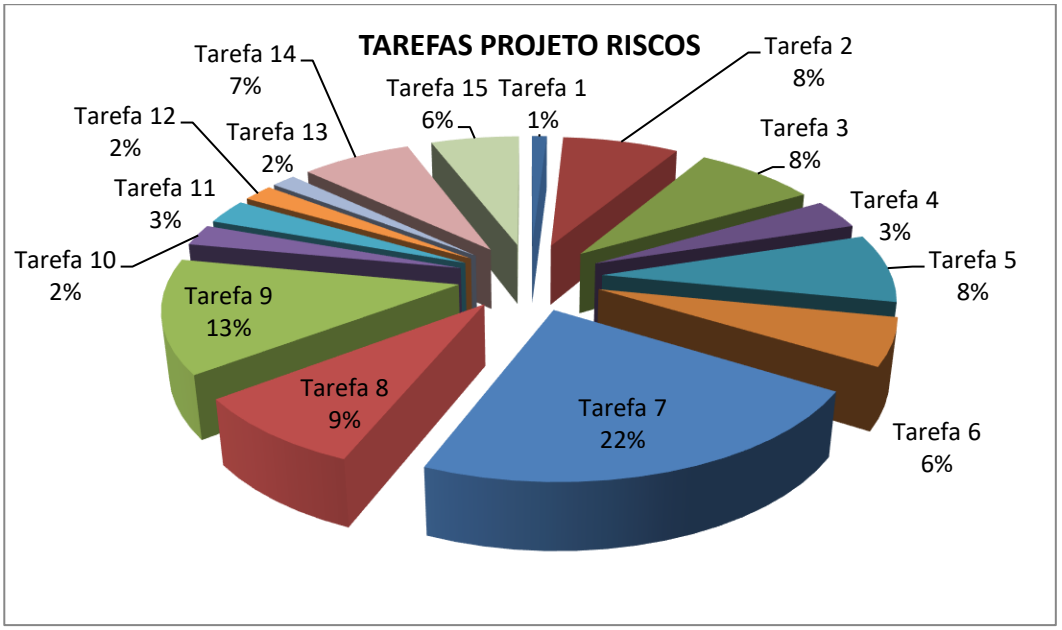

Figura 6 – Previsto vs Realizado no Projeto Riscos
Fonte: Elaboração Própria

A Figuras 6 descreve o trabalho final nas três iterações realizadas para desenvolver o projeto, fazendo uma comparação entre o previsto no planejamento e a execução do projeto por funcionário e pela equipe como um todo. Com o controle realizado foi possível calcular o tempo que foi empregado no desenvolvimento de cada tarefa, assim como os custos por cada módulo programado.

Na etapa de encerramento foi feita a reunião final do projeto com o objetivo de apresentar o produto final e identificar todas as contribuições ao usar uma metodologia de trabalho no momento de desenvolver um projeto de desenvolvimento de software.

De acordo com a análise feita pela equipe que participou no projeto o uso da metodologia de trabalho contribuiu na melhora do trabalho em equipe. Segundo o coordenador de projeto "FA", a aprendizagem da equipe de trabalho nas diversas práticas de desenvolvimento de software XP, e desenvolvimento enxuto de software ajudou a não desviar atividades no desenvolvimento de software e focar o trabalho unicamente nas entregas úteis para o projeto em desenvolvimento.

O funcionário "FB" afirmou que foi útil obter a base de conhecimento gerada ao longo da elaboração do projeto, já que ficaram registrados diversos dados úteis do projeto desenvolvido, de fácil uso para próximos projetos pela equipe de trabalho. O funcionário "FC" afirmou que o trabalhar com iterações pequenas ajuda ao entendimento completo do projeto por parte de toda a equipe de desenvolvimento. Finalmente o coordenador do projeto afirmou que houve uma melhora nas práticas de trabalho, cumprimento de prazo, e melhoria de qualidade no produto de software desenvolvido, já que comparando com o trabalho de projetos

anteriores, na empresa nunca houve cumprimento de prazos dos projetos, e existia muito retrabalho pela falta de comunicação, problema que foi superado com o uso das metodologias ágeis já que contribuíram notavelmente ao compartilhamento de informação na equipe de trabalho.

Conclusões

O problema do gerenciamento de projetos de *software* é na prática complexo e, por esse motivo a metodologia proposta deverá adaptar-se às diversas realidades das empresas, fazendo alterações necessárias, considerando características próprias de cada empresa e cada projeto em desenvolvimento. Hoje em dia muitas empresas dedicadas a desenvolver *software* optam por trabalhar com metodologias ágeis por serem menos burocráticas do que as tradicionais e pela adaptabilidade das metodologias ágeis: as entregas pequenas e rápidas proporcionam um desenvolvimento com respostas rápidas, facilidade de mudanças e de adaptação, em função do tipo de projeto de *software* em desenvolvimento.

O modelo de referência proposto incrementou a produtividade da equipe estudada, já que influenciou claramente no envolvimento, comunicação, colaboração e disciplina da equipe de desenvolvimento de software. Em relação às práticas adotadas, as práticas XP influenciaram na melhoria da qualidade do projeto comparando com projetos anteriores desenvolvidos na empresa em estudo. O *Scrum* permitiu que a equipe foque seu trabalho em entregas úteis aos clientes, e ajudou a produzir só o que era necessário.

O uso dos padrões de nomenclatura gerou grandes benefícios no projeto desenvolvido, já que facilitou o trabalho do *refactoring*, e ajudou a fornecer informações rápidas e consistentes do código em desenvolvimento e melhorou a estética, aparência, e fácil utilização do código-fonte. Cabe ressaltar que existem algumas práticas ágeis com as quais não é fácil lidar como as reuniões diárias, já que, dependendo da quantidade do trabalho da equipe, às vezes é difícil deter o trabalho e fazer as reuniões.

O modelo de referência ajudou a fornecer conceitos e procedimentos padronizados a serem seguidos pela empresa na sua gestão dos projetos de software, é incentivou na criação de uma ferramenta de auxílio inspirada em conceitos da prática de gerenciamento *Scrum* que ajudou a controlar automaticamente o trabalho da equipe, os tempos trabalhados em cada iteração e processos, e custos de trabalho dos desenvolvedores.

Referências

Agile Alliance. (2010). Manifesto for Agile Software Development. Disponível em: http://www.agilealliance.org/.

Agile Alliance. (2010). "What is Scrum." Disponível em: http://www.controlchaos.com.

Astels, D. (2002). Extreme Programming: Guia Prático. Rio de Janeiro: Campus.

Beck, K. (1999). Extreme Programming Explained: Embrace Change. Reading, Mass,. Addison-Wesley.

Cohen, D., et al. (2003). Agile Software Development: A DACS State of Art Report. Disponível em: www.thedacs.com/thecs/agile.

Daflon, L. R. (2004). Um Framework para a Representação e Análise de Processos de Software, Dissertação de Mestrado, Departamento de Informática, PUC-Rio, Rio de Janeiro.

Feature-Driven Development Web Site. Disponível em: http://www.featuredrivendevelopment.com/.

Ferreira, E. (2007). Um Modelo de Gerenciamento de Projetos Baseado nas Metodologias Ágeis de Desenvolvimento de Software e nos Princípios da Produção Enxuta, Dissertação de Mestrado, Escola Politécnica, Universidade de São Paulo.

Fiorini, S. (1988). Engenharia de Software com CMM. Rio de Janeiro: Brasport.

Fowler, M. (2010). Continuous Integration. Disponível em: http://www.martinfowler.com.

Goldman, A., Kon, J., Silva, P. J., & Yoder, J. W. (2004). Being Extreme in the Classroom: Experiences Teaching XP. Journal of the Brazilian Computer Society, 1-18.

Guinato. Produção e Competitividade: Aplicações e Inovações. Ed: Adiel T. de Almeida & Fernando M.C. Souza, Edit da UFPE, Recife, 2000.

Humphrey, W. S. (1995). A Discipline for Software Engineering. Reading, MA: Addison-Wesley.

Isnard, M. J., Jr., Cierco, A. A., Rocha, A. V., Mota, E. B., & Leusin, S. (2008). Gestão da Qualidade. Rio de Janeiro: Editora FGV.

Isotton Neto, E. (2008). Scrumming: Ferramenta Educacional para Ensino de Práticas do Scrum. Dissertação de Mestrado em Informática, Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul.

Johnson, J. (2001). Micro Projects Cause Constant Change. The Standish Group International.

Kerzner, H. (2002). Gestão de Projetos: As Melhores Práticas. Porto Alegre: Bookman.

Kniberg, H. (2007). Scrum and XP from the Trenches.

Manifesto Ágil. Disponível em: http://www.manifestoagil.com.br/principios.html.

Martins, J. (2007). Técnicas Para Gerenciamento de Projetos de Software. Rio de Janeiro: Brasport.

Mnkandla, E., & Dwolatzky, B. (2006). Defining Agile Software Quality Assurance. In: International Conference on Software Engineering Advances (ICSEA06), IEEE Computer.

Nascimento, G. (2008). Um Modelo de Referência para o Desenvolvimento Ágil de Software. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação, USP-São Carlos.

Nocêra, R. de J. (2009). Gerenciamento de Projetos: Teoria e Prática. Santo André, SP.: Ed. do Autor.

Pellegrinelli, S., & Bowman, C. (1994). Implementing Strategy Through Projects. Long Range Planning, 27(4).

Poppendieck, M., & Poppendieck, T. (2003). Engenharia Enxuta de Software. Disponível em: http://www.poppendieck.com/.

PMI - SP. (2010). Gerenciamento de Projetos. Disponível em: http://www.pmi.org.br.

Project Management Institute. (2009). Relatório Principal – Perspectiva Geral Estudo de Benchmarking em Gerenciamento de Projetos – Chapters Brasileiros.

Proqualiti. (2005). Qualidade na Produção de Software: O Gerenciamento de Projetos de Software Desenvolvidos à Luz das Metodologias Ágeis: Uma Visão Comparativa.

Project Management Institute - PMI. (2008). A Guide to the Project Management Body of Knowledge (PMBOK, GUIDE), Fourth Edition.

Santos Soares, M. (2004). Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software. Dissertação de Mestrado, Universidade Presidente Antonio Carlos.

Royce, W. W. (1970). Managing the Development of Large Software Systems: Concepts and Techniques. Proc Wescon, 1-8.

Schwaber, K. (2004). Agile Project Management with Scrum.

Software Engineering Institute (SEI). (2006). CMMI for Development, Software Engineering, University of Carnegie Mellon. Disponível em: http://www.sei.cmu.edu/cmmi.

Teles, V. (2004). Extreme Programming: Aprenda como Encantar seus Usuários Desenvolvendo Software com Agilidade e Alta Qualidade. São Paulo: Novatec Editora.

The Standish Group. (1994). The Chaos Report. Disponível em: http://www.standishgroup.com.

The Standish Group. (2006). The Chaos Report. Disponível em: http://www.standishgroup.com.

Vargas, R. V. (2005). Gerenciamento de Projetos: Estabelecendo Diferenciais Competitivos. Rio de Janeiro: Brasport.

Womack, J. P., Jones, D. T., & Roos, D. (1992). A Máquina que Mudou o Mundo. Campus. 5a Edição. Rio de Janeiro.

Submetido em: 23.10.2023

Aceito em: 21.11.2023