Um Processo de Desenvolvimento de Software para Projetos Super Ágeis*

A Software Development Process for Super Agile Projects

Marília M. B. Cerqueira
Universidade do Estado da Bahia
Rua Silveira Martins, 2555, Cabula, Salvador
Brasil
mbcmarilia@gmail.com

Ana Patrícia Magalhães
Universidade do Estado da Bahia / Universidade Salvador
Rua Silveira Martins, 2555, Cabula, Salvador
anapatriciamagalhaes@gmail.com

Hugo Saba
Universidade do Estado da Bahia
Rua Silveira Martins, 2555, Cabula, Salvador
Brasil
hugosaba@gmail.com

Eduardo M. F. Jorge
Universidade do Estado da Bahia
Rua Silveira Martins, 2555, Cabula, Salvador
Brasil
emjorge1974@gmail.com

ABSTRACT

Increasing in the creative economics, growth in the demand of mobile applications, and necessity of rapid product availability to the customer promote arising from projects that have a life cycle of a few days. These type of projects are called super agile. Among them are innovation projects with emphasis in mobile application and elaboration of proof of concept to study the technical viability or others projects developed, for example, by startups. However, the field of super agile software development still is in its initial stages, maybe because it is a recent problem faced by companies and startups. Thereby, this article proposes a development process that meets the features of projects with short life cycles of up to 2 weeks. The proposed process based on market demands, identified in field research, integrates the most current used agile methods and methodologies. An initial evaluation in three organizations showed that the process is adherent to necessities of these companies.

CCS CONCEPTS

• CCS → Software and its engineering → Software creation and management → Software development process management → Software development methods → Agile software development

KEYWORDS

Super agile process; Software development process; Startups, Innovative projects.

https://doi.org/10.1145/3229345.3229399

1 INTRODUÇÃO

Com o surgimento das metodologias ágeis nos anos 90 e a criação do Manifesto Ágil em 2001, repensou-se a forma de planejar e construir software em relação aos processos clássicos de desenvolvimento. As metodologias tradicionais consideradas pesadas, difíceis de se manter no cronograma, e com grande quantidade de documentação gerada, impulsionaram a mudança de pensamento e a criação de metodologias leves de desenvolvimento de software, as metodologias ágeis [26][17][6].

Métodos ágeis atendem a uma diversidade de projetos, mas podem não ser bem adequados para segmentos que constroem softwares em *startups* e empresas inovadoras. Essas empresas buscam formular e testar hipóteses acerca de sua inovação, por se encontrarem num momento de muitas incertezas [15]. As hipóteses precisam ser testadas, com o objetivo de definir o cliente e outras questões da proposição de valor do produto. De acordo com Ries [15], para testá-las, utiliza-se uma versão inicial do que um dia se tornará o produto final, com a finalidade de obter uma rápida colocação do produto no mercado e a coleta do *feedback* do cliente.

Neste contexto, observa-se que muitas *startups* e empresas procuram por processos de desenvolvimento de software que atendam suas demandas de disponibilização rápida de produtos no mercado. Apesar do sucesso de metodologias ágeis, como *Scrum* e *eXtreme Programming* [5] ainda existem questões em aberto para projetos denominados super ágeis, a exemplo do tempo de desenvolvimento, que muitas vezes é inferior a duas semanas. São exemplos de domínios desse tipo de projeto aplicativos *mobile* e elaboração de provas de conceito em projetos de inovação para

estudo de viabilidade técnica ou outros projetos desenvolvidos por *startups* e empresas. Neste contexto, surge a seguinte questão: quais são os elementos necessários para um processo de desenvolvimento de software que dê suporte a projetos super ágeis?

Para contribuir com esse cenário, este artigo analisa o nível de aderência dos métodos, metodologias ou processos ágeis para projetos denominados super ágeis e propõe um processo de desenvolvimento de software que atenda às características de projetos que possuam ciclos curtos chegando a uma duração de tempo de até duas semanas. O processo integra técnicas, práticas, métodos e ferramentas no estado da arte da Engenharia de Software que apoiem projetos nesse domínio.

A metodologia adotada para o objetivo supracitado baseia-se na pesquisa-ação e na avaliação qualitativa. A escolha desses dois métodos se deve a necessidade de coleta de dados através de entrevistas com desenvolvedores que trabalham em projetos super ágeis e a necessidade de incrementar o processo proposto ao longo das etapas de pesquisa (ver mais detalhes na Seção 3.1). O restante do artigo está organizado da seguinte maneira: a Seção 2 apresenta as metodologias, técnicas e ferramentas analisadas aderentes ao desenvolvimento de projetos super ágeis; na Seção 3, é apresentada uma pesquisa de campo com as demandas de empresas super ágeis, detalhado processo proposto e sua avaliação; a Seção 4 apresenta os trabalhos correlatos encontrados; e, na Seção 5, são apresentadas as conclusões.

2 MÉTODOS, TÉCNICAS E FERRAMENTAS PARA PROJETOS SUPER ÁGEIS

No final do século XX e início do século XXI, projetos de desenvolvimento de software enfrentam atrasos em seus cronogramas, aumento no orçamento inicial e falta de qualidade por não seguirem processo algum ou por seguirem um método pesado demais para a demanda que se tinha. Com isso, no início de 2001, um grupo de especialistas representantes de metodologias ágeis e simpatizantes, chamado de Aliança Ágil, preocupado com o rumo que o desenvolvimento de software estava levando, se reuniu para discutir valores e princípios que ajudariam equipes de desenvolvedores do mundo todo. Dessa reunião surgiu o Manifesto Ágil, que prega os seguintes valores:

> Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazê-lo. Com esse trabalho, passamos a valorizar:
>
> Indivíduos e interações mais que processos e ferramentas; Software em funcionamento mais que documentação abrangente;Colaboração com o cliente mais que negociação de contratos;Resposta a mudanças mais que seguir um plano
>
> Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda [10].

O Manifesto Ágil foi importante para fortalecer os métodos ágeis, aumentar sua credibilidade enquanto metodologia e aumentar, consequentemente, sua aceitação. As metodologias ágeis, na verdade, trazem um conjunto de técnicas já existentes há anos, como, por exemplo, a técnica de melhoria iterativa existente desde 1975, a qual é utilizada na maioria delas [5]. Muitas de suas técnicas e práticas não têm nada de realmente novo, o que muda de fato é o foco e os valores por trás das metodologias ágeis, como visto anteriormente no Manifesto Ágil.

Algumas das metodologias ágeis mais utilizadas são o *Scrum* [10] e o XP [2]. O *Scrum* possui ciclos de desenvolvimento compostos por iterações, chamados de *sprints*, que duram em média de 2 a 4 semanas. Antes de o sprint acontecer, define-se o que precisa ser implementado e essas informações são guardadas numa lista denominada de *Product Backlog*. Em uma reunião, a *Sprint Planning Meeting*, as funcionalidades que serão implementadas no *sprint* são selecionadas e transferidas para o *Sprint Backlog*. Já a Programação Extrema (XP) proporciona um ambiente de trabalho onde o cliente tem um papel ativo no desenvolvimento do projeto, fazendo-se disponível para qualquer demanda da equipe e sendo considerado parte dela. Antes de cada iteração, todos os participantes do projeto se encontram para planejar, definindo a lista de requisitos, priorizando-os e estimando a entrega. A lista de requisitos do usuário é chamada de histórias do usuário e fica armazenada em cartões de história, podendo ser alterada quando necessário. A cada iteração, novas funcionalidades são acrescentadas ao produto, e o software funcionando é mostrado às partes interessadas. Algumas técnicas importantes propostas pelo XP são o trabalho em pares, o código não proprietário, o refatoramento o ritmo de trabalho sustentável, a integração contínua, entre outros. Para apoiar essas técnicas, algumas ferramentas podem ser utilizadas, tais como repositórios de armazenamento, a exemplo do *Git*, *Subversion*, *Perforce* e *Mercurial* [12]. Adicionalmente, *frameworks* também são adotados para efetuar os testes automatizados, necessários para validar as mudanças feitas no código e apontar possíveis erros de integração para o desenvolvedor, a exemplo dos *frameworks XUnits*, como o *JUnit* para a linguagem Java e *CPPUnit* para a linguagem C++. Ambientes integrados IDE's (*Integrated Development Environment*) possibilitam integrar essas ferramentas através da utilização, por exemplo, de *plug-ins*.

Além do *Scrum* e do XP, o *Design Sprint*, um processo para se testar ideias com o usuário em cinco dias [7] também é interessante para o contexto de projetos super ágeis. Trata-se de um processo que possibilita o *design*, prototipaçãoe teste com usuários em menos de uma semana. Para isso, ele define um cronograma com etapas a serem cumpridas. A primeira etapa consiste em entender as necessidades do usuário e do negócio, e obter conhecimento acerca da tecnologia que se quer utilizar. Na segunda etapa decide-se o que precisa ser melhorado, o que deve ser enfatizado e quais as estratégias a serem seguidas, baseando-se nos resultados da primeira etapa. A terceira etapa é a de divergir, que se caracteriza por explorar as ideias de cada membro da equipe. Na etapa seguinte, a de decisão, as ideias são refinadas e expostas num quadro branco para serem votadas individualmente e anonimamente. Depois, o grupo faz uma discussão para decidir o que de fato será usado na construção do protótipo. A quinta etapa consiste em fazer um protótipo de alta fidelidade que será testado por usuários na etapa seguinte. Este protótipo pode ser um vídeo, um protótipo físico, ou uma simulação, dentre outros. A sexta etapa é a de validação do protótipo com usuários, o que por consequência serve para validar a ideia. Além disso, há a

validação por parte do *stakeholder* chave do projeto, que é a pessoa responsável por alocar recursos e financiar a ideia.

Métodos diversos da Engenharia de Software também são aplicados ao desenvolvimento ágil. De acordo com Kniberg e Skarin [8], pode-se misturar e combinar métodos e técnicas de acordo com o modelo de negócio da organização, utilizando o que melhor se encaixar na cultura da empresa e na natureza dos projetos, sem medo de experimentar e falhar. O *Kanban*, por exemplo, é um método altamente adaptativo [8] [18] devido às poucas limitações impostas que pode ser usado em projetos super ágeis. Neste, utiliza-se normalmente um quadro branco com cartões colados a ele descrevendo atividades. Os cartões são estrategicamente colados em determinadas colunas para poder visualizar o status da tarefa, ou seja, se ela ainda precisa ser feita, se está sendo desenvolvida ou se já foi concluída. Ao visualizar o fluxo de trabalho é possível enxergar gargalos e imaginar possíveis soluções para resolvê-los, evitando perder mais tempo resolvendo problemas e comprometer o cronograma reduzido de projetos curtos, como os projetos super ágeis.

A Tabela 1 traz um resumo dos elementos da Engenharia de Software, que podem ser utilizados no processo de desenvolvimento para projetos super ágeis. Os elementos trazidos nesta tabela foram descritos parcialmente ao longo desta seção e escolhidos pela sua relevância através dos estudos feitos e a partir de uma revisão sistemática elaborada para subsidiar esta pesquisa. Alguns elementos, como *Kanban* e *Design Sprint* foram estudados após sugestões feitas no questionário e nas entrevistas. O primeiro item do processo metodológico, mostrado na Subseção 3.1, traz um pouco mais de clareza quanto à escolha desses elementos.

Tabela 1: Resumo dos elementos da Engenharia de Software aderentes a projetos super ágeis

Elemento	Característica	Classificação
Scrum [10][25]	Caracteriza-se por ciclos de desenvolvimento com iterações *(Sprints*), *Sprint Backlog*, e reunião diária (*Daily Scrum*).	Metodologia Ágil
XP [2]	Usa integração contínua, o desenvolvimento é de forma iterativa, a programação é feita em pares.	Metodologia Ágil
Reuso [22] e Linha de Produto de *Software* [13]	Reutiliza-se componentes já existentes na criação de novos produtos, poupando tempo de desenvolvimento e garantindo a confiabilidade e qualidade por esses componentes já terem sidos testados anteriormente [13].	Processo e técnica Respectivamente
Integração continua [12]	Automatiza o processo de integração do novo código com o que já foi produzido anteriormente, incluindo a utilização de testes automatizados.	Prática
Corrente Crítica	Trabalha direto nas restrições para tentar impedir atrasos no cronograma, com possível realocação de recursos.	Método
Planejamento em Ondas Sucessivas [9]	Útil para se planejar em curto prazo, porque utiliza uma elaboração progressiva [9].	Técnica
Prototipação [11]	Útil na elaboração de prova de conceito, obtenção de *feedback* de forma mais rápida dos possíveis clientes.	Técnica
Kanban [3]	Quadro *Kanban* útil no planejamento e acompanhamento das tarefas.	Método
Design Sprint [7]	Auxilia no processo de concepção da ideia e prototipação do futuro produto.	Método

3 PROCESSO SUPER ÁGIL

O processo de desenvolvimento de *software* para projetos super ágeis proposto neste artigo utilizou como base os elementos da Engenharia de Software elencados na Tabela 1 e os resultados de uma pesquisa de campo realizada em empresas que trabalham com este tipo de projeto.

3.1 Processo Metodológico

Para a elaboração do processo, a metodologia foi conduzida em seis etapas de forma iterativa:

1. A primeira consistiu no estudo das principais características, técnicas e práticas de Engenharia de Software que podem ser aplicadas no desenvolvimento de projetos super ágeis. Estudos sobre diferentes metodologias ágeis, suas práticas e características foram feitos. Foi dado um enfoque no método *Scrum* por sua grande aceitação e uso pela comunidade de desenvolvimento de *software*, conforme visto na pesquisa feita pela VERSIONONE[24], na qual 56% de pessoas afirmam utilizar este método dentre 3.925 profissionais de TI entrevistados. Estudos sobre o método *Kanban* e o processo *Design Sprint* também ocorreram após sugestões feitas no questionário e nas entrevistas das etapas seguintes. É válido lembrar que essa metodologia é cíclica, podendo acontecer de etapas ocorrerem novamente.
2. Na segunda etapa, foi aplicado um questionário com 9 empresas que trabalham com desenvolvimento de *software*, em *startups* e empresas, no domínio de aplicativos móveis e elaboração de provas de conceito em projetos de inovação. A partir das respostas desse questionário, alguns participantes foram selecionados para entrevistas a fim de ter um aprofundamento em algumas questões referentes ao

desenvolvimento de *software* onde trabalham. Três pessoas de diferentes contextos e ambientes de trabalho do domínio pesquisado foram entrevistadas. Com essas entrevistas, buscou-se levantar dados sobre as demandas e necessidades que essas empresas e *startups* têm ao realizar projetos de domínio super ágil.

3. A terceira etapa consistiu em analisar o nível de aderência das metodologias e métodos para o domínio de projetos super ágeis. Através da pesquisa feita na segunda etapa e dos estudos sobre métodos ágeis realizados na primeira, foi possível analisar quais métodos, técnicas e práticas dessas metodologias podem ser reutilizados e adaptados para um processo de desenvolvimento de *software* que atenda as demandas e necessidades de projetos super ágeis.
4. Na quarta etapa, um processo de desenvolvimento de *software* para projetos super ágeis foi elaborado, baseando-se nas demandas e necessidades identificadas nas entrevistas e nos estudos sobre metodologias ágeis e Engenharia de Software. A partir das percepções e análises feitas com as informações obtidas pelas entrevistas e pelas interações com os entrevistados, juntamente com o que foi realizado na primeira etapa, buscou-se propor um processo que possa auxiliar no desenvolvimento de projetos super ágeis, atendendo as necessidades e demandas apresentadas.
5. Na quinta etapa, o processo para projetos super ágeis, criado na etapa anterior, foi mostrado para os entrevistados, a fim de se obter um *feedback* para descobrir se aquele processo atende as necessidades deles. A partir deste *feedback*, procurou-se fazer uma avaliação qualitativa de todo o processo.
6. A sexta e última etapa é caracterizada por se fazer mudanças e adaptações no processo, a partir do *feedback* obtido na quarta etapa e da avaliação qualitativa realizada. E por fim, uma avaliação de todo o processo é feita.

3.2 Pesquisa de Campo

O objetivo principal da pesquisa de campo foi identificar como é realizado o desenvolvimento de *software* em empresas que trabalham com sistemas super ágeis, ou seja, quais as técnicas e métodos da Engenharia de Software utilizados por essas empresas ao longo do desenvolvimento de um sistema.

A pesquisa foi realizada através da aplicação de um questionário a nove empresas e *startups* que trabalham com desenvolvimento de aplicativos móveis e projetos de inovação, seguida de entrevistas em algumas destas empresas. O questionário compreende 9 perguntas relacionadas aos métodos e técnicas empregadas ao longo do desenvolvimento. Foi utilizada a escala Likert[20], com cinco níveis para medir o nível de aceitação das respostas do questionário. Adicionalmente foram feitas algumas perguntas subjetivas para coletar sugestões de métodos e técnicas relevantes ao contexto de projetos super ágeis.

O porte das empresas entrevistadas foi definido pela quantidade de empregados que ela possui de acordo com o SEBRAE [16], onde uma pequena empresa é composta de 10 a 49 funcionários, uma empresa de médio porte possui 50 a 99 empregados e uma grande empresa possui mais de 100 funcionários. Desta forma, na pesquisa feita, tem-se 4 empresas consideradas de médio ou grande porte, 3 a 4 empresas de pequeno porte e 1 a 2 microempresas (Figura 1). Vale considerar que é difícil definir uma *startup* pelo seu número de funcionários, porque sua definição se deve mais ao objetivo possuído pelas pessoas que a compõem. Com isso, uma *startup* é caracterizada por um grupo de pessoas geralmente pequeno que está em busca de um modelo de negócios, o qual possa ser escalável e repetido várias vezes, e que trabalha em um cenário de bastante incertezas [19].

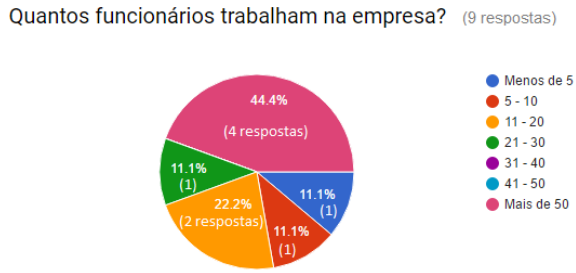

Figura1:Porte das empresas entrevistadas.

A análise dos dados constatou que todas as empresas participantes utilizam algum método ágil no desenvolvimento de seus projetos e a maioria destas utilizam *Scrum* e/ou XP (Figura 2).

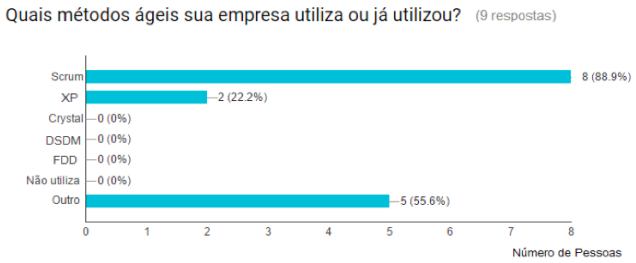

Figura2:Utilização de métodos ágeis nas empresas.

Em relação ao tempo de desenvolvimento de um projeto, 44,4% dos participantes (4 empresas) desenvolvem em média seus projetos em 1 a 2 semanas, enquanto 33,3% (3 empresas) levam de 2 a 4 meses. Uma empresa relatou gastar em média 1 mês por projeto e outra empresa não se manifestou, justificando que o foco de sua organização é em produto e não em projeto. Quando considerado o tempo médio de desenvolvimento de uma versão, 66,7% (6 empresas) responderam 1 a 2 semanas, 11,1% (1

empresa) optou pela alternativa de 1 semana, 11,1% assinalou que leva de 2 a 3 dias, e 11,1% respondeu 1 mês. Com isso, é possível observar, no âmbito de tempo, que a natureza dos projetos dessas empresas se assemelha ao perfil de projetos curtos denominados de super ágeis.

Quanto à importância das técnicas de integração contínua, corrente crítica e ondas sucessivas, algumas pessoas deixaram em branco estas perguntas por falta de conhecimento prévio do que são. Com isso, 75% (6 empresas) consideram a integração contínua muito importante, e 25% (2 empresas) apenas importante. Três empresas consideraram ondas sucessivas como importante ou muito importante, as demais se mantiveram neutras.

Por fim, uma questão subjetiva possibilitou que o participante da pesquisa escrevesse sugestões de técnicas ou métodos que possam ser relevantes para os projetos super ágeis. Das 9 empresas que participaram, 6 responderam a esta questão, Dentre as sugestões estão o método *Kanban*, o *DesignSprint*, *Dual TrackScrum*, *Test DrivenDevelopment* e prototipação.

Vale ressaltar também que durante as entrevistas realizadas após a aplicação do questionário os desenvolvedores destas empresas não se mostraram satisfeitos com as metodologias atuais utilizadas e destacaram a importância de integrar a estas o uso de *frameworks* e padrões de projeto, desenvolvimento baseado em componentes e linhas de produto de *software* como técnicas relevantes para o contexto de projetos super ágeis.

Através da coleta de dados foi possível perceber a importância que os métodos ágeis, como *Scrum* e XP, têm no desenvolvimento de *software* para essas empresas ou *startup*, assim como a relevância de alguns métodos e técnicas, como a integração contínua, ondas sucessivas, *Kanban* e *Design Sprint*.

3.3 Detalhamento do Processo

O processo de desenvolvimento de *software* para projetos super ágeis, objeto desta pesquisa, é composto por três etapas mostradas na Figura 3: concepção da ideia, reuso e busca por conhecimento, e execução. Ele se baseia nas metodologias ágeis *Scrum* e XP, no método *Kanban* e no processo *Design Sprint*. Além disso, o processo também engloba práticas e técnicas da Engenharia de Software, como: integração contínua, planejamento em ondas sucessivas, reuso, prototipação e corrente crítica.

A primeira etapa do processo se baseia no *Design Sprint*. Inicialmente, os integrantes da equipe irão compartilhar suas dúvidas, conhecimentos e questões sobre o problema. Para entender melhor o contexto em que o projeto ou produto está inserido, é importante defini-lo, seja através de *storyboard*, *persona* ou outra ferramenta e técnica. Além disso, a equipe pode escolher pesquisar de 3 a 5 projetos semelhantes a fim de obter inspiração, por exemplo, aplicativos de gamificação quando se busca fazer algo similar. Depois, é preciso decidir o ponto do projeto que será trabalhado. Para isso, cada um escreverá em pequenos cartões possíveis soluções. Estes cartões são agrupados de forma que faça sentido para o projeto em que se está trabalhando. Cada membro da equipe possui um determinado número de votos e escolhe o grupo de ideias que mais se interessou, sendo que pessoas da equipe com maior poder de decisão têm direito a mais votos. Com isso, já é possível definir quem é o público alvo e, caso a equipe opte por fazer teste com usuários no final dessa etapa, pode começar a recrutar pessoas. Além disso, cada membro do time esboça ideias de possíveis soluções no papel e, no final, esses papéis são recolhidos para não pertencer a ninguém. As ideias são expostas como se fossem cartazes, cada membro escolhe em quais irá votar e os que têm maior poder de decisão têm mais votos novamente.

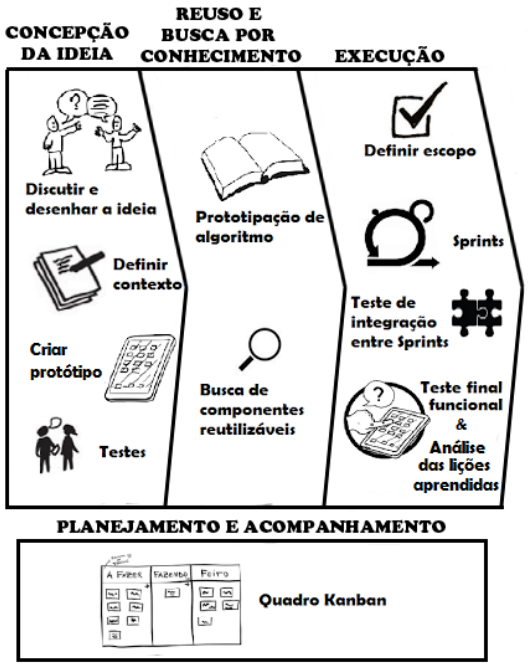

Figura3:Processo de desenvolvimento de software para projetos super ágeis.

Após a escolha da ideia ou ideias, o próximo passo é a prototipação. A depender do projeto ou produto a equipe faz a escolha de criar um protótipo de baixa fidelidade, no papel, por exemplo, ou de fazer um protótipo de alta fidelidade. Este utiliza ferramentas de *software* para sua criação, como a Marvel[11], por exemplo, ferramenta utilizada na criação de telas para Web e aplicativos móveis. É válido ressaltar a importância de se fazer um protótipo de alta fidelidade caso a equipe deseje fazer testes com usuário, porque este protótipo é que será mostrado e testado pelos usuários nesta primeira etapa do processo. Por último, testes serão feitos para avaliar o protótipo, podendo ser testes de usabilidade, navegabilidade, acessibilidade e outros. Os testes com usuário são importantes quando se busca obter um *feedback* rápido da ideia que se procura validar e para responder questões de proposição de valor do produto, economizando tempo e recursos.

A segunda etapa é a fase de reuso e busca por conhecimento pela equipe. É uma etapa que em alguns momentos se confunde com a primeira etapa, ou seja, às vezes a busca de conhecimento e componentes reutilizáveis acontece na etapa de concepção da

ideia. Nesta etapa, fazem-se estudos em pares, como na programação em par do XP, buscando em fóruns, sites, discutindo com pessoas da área melhores soluções para determinadas funcionalidades ou atividades. De acordo com Williams et al. [27], duas pessoas trabalhando juntas costumam pensar em mais soluções para um problema e se ajudar quando há a necessidade de análise e *design*, diminuindo as chances de se persistir em uma má ideia. Com isso, a troca de ideias e experiências, o debate e o *feedback*, possibilitados pelo trabalho em par, mostra-se importante nesta etapa. Além disso, procuram-se também no repositório ou biblioteca de componentes reutilizáveis, artefatos que possam ser usados novamente no desenvolvimento do novo produto. Assim, o tempo de desenvolvimento inicialmente é alto, mas com o aumento no número de diferentes produtos criados há uma considerável diminuição no tempo de desenvolvimento e até certa estabilidade. Essa diminuição de tempo de desenvolvimento é o que se almeja ao se ter projetos super ágeis, porque, com estes projetos, é preciso encontrar um equilíbrio entre um reduzido *time-to-market* (tempo de colocação do produto no mercado) e produzir produtos de qualidade.

A etapa I e II foram pensadas para durar até 3 dias, disponibilizando assim um período de cerca de 11 dias para a etapa de execução do produto, devido a problemas ou gargalos que podem aparecer ao longo da etapa III.

A terceira e última etapa, baseada em *Scrum*, define um escopo mínimo a partir do protótipo feito na etapa I e o quebra em funcionalidades de *software* que irão alimentar o *backlog* dos *sprints*. O protótipo pode trazer mais funcionalidades do que realmente será implementado inicialmente, pelo fato de que o *design* pode ser rápido e fácil, e consequentemente englobar mais características, surgindo, assim, questões de como alinhar o *design* com o desenvolvimento. É definido também um cronograma com as grandes entregas para se ter um planejamento macro do projeto. Essas entregas serão baseadas no *ProductBacklog* definido anteriormente. Os *sprints* têm duração de 2 dias, com reuniões diárias para discutir o que será feito no dia e o que foi feito no dia anterior, revisão do *sprint*, e retrospectiva do *sprint*, se necessário. Utiliza-se integração contínua nos *sprints*, técnica herdada do XP e também são feitos testes de integração automatizados entre os *sprints*. Ao final da etapa, é feito um teste final funcional, podendo ser feito também testes em ambientes reais para se obter um *feedback* mais rápido. Em paralelo a estes testes, é feita também uma análise das lições aprendidas e das estratégias adotadas que proporcionaram resultados positivos no desenvolvimento do projeto, com o objetivo de alimentar a base de conhecimento utilizada na etapa II.

O planejamento e acompanhamento do processo de desenvolvimento são feitos através do *Kanban*. O quadro *Kanban* é utilizado como um meio de visualizar o fluxo de desenvolvimento do *software*. As cartas coladas a ele trazem as tarefas que precisam ser feitas, a assinatura do responsável e um código para que se tenha acesso a mais informações sobre aquela carta em algum sistema. As colunas representam etapas do fluxo de desenvolvimento. Cada etapa possui um número para limitar o trabalho em andamento (WIP) de cada coluna. O quadro *Kanban* é baseado no planejamento em ondas sucessivas e traz atividades que a equipe possa enxergar até o momento e, a partir do resultado dela, novas tarefas serão criadas. O gerenciamento do projeto é feito através dos cartões presos no quadro *Kanban*, os quais possuem diferentes cores para alertar o estágio de criticidade das atividades (verde, amarelo e vermelho). Cada pessoa avalia em que estágio está a atividade do cartão que é responsável, sendo uma avaliação pessoal em relação ao todo do tempo ao invés de uma avaliação micro do tempo. Se o desenvolvimento do *software* estiver nas etapas de concepção e busca por conhecimento, cada indivíduo avalia o nível de criticidade de suas tarefas considerando o tempo de 3 dias que tem para realizar as atividades dessas etapas. Caso esteja na etapa de execução, cada membro considera os 2 dias de duração de cada *sprint* para avaliar o nível de criticidade de suas atividades. Sendo assim, cartão verde significa que a tarefa está fluindo dentro do esperado, amarelo para tarefas que estão começando a ter problemas, e vermelho para atividades que estão tendo problemas para serem realizadas respeitando o tempo que se tem para aquela etapa ou fase. É válido ressaltar que o tempo definido para cada etapa ou para os *sprints* pode ser modificado de acordo com as necessidades de cada projeto, respeitando o tempo de 2 semanas de duração para projetos super ágeis, alvo deste processo de desenvolvimento de *software*.

Assim, tem-se um processo de desenvolvimento de *software* para projetos super ágeis, que duram até 2 semanas, respeitando a necessidade de rápida colocação no mercado e qualidade do produto, para se obter *feedback* dos usuários e respostas em relação a validação de hipótese.

3.4 Avaliação do Processo

Processos de desenvolvimento de *software* são inerentemente complexos, pois envolvem um elevado número de atividades executadas por pessoas com diferentes expertises. Como consequência, avaliações de processos de *softwares* são geralmente realizadas em etapas que se complementam [21]. A avaliação deste processo se iniciou através da análise de especialistas. Três empresas que trabalham com projetos de natureza super ágil foram selecionadas e receberam o processo para que seus desenvolvedores realizassem uma análise com base em sua experiência em projetos já desenvolvidos anteriormente na empresa. Em seguida, essas pessoas foram entrevistadas e forneceram *feedback* em relação a cada uma das etapas do processo.

A primeira empresa entrevistada discordou do uso de *storyboard* e sugeriu a inclusão de uma atividade de mais alto nível, como definir contexto, para deixar a critério da equipe do projeto decidir qual técnica utilizar. Essa sugestão foi acatada e o processo foi modificado. Ainda em relação à primeira etapa do processo, a empresa defendeu que ao invés de separar as atividades de teste com usuário ou interno, e testes (usabilidade, navegabilidade etc), a figura do processo deveria indicar apenas o nome “Teste”. O detalhamento do tipo de testes disponíveis e a opção de como fazê-lo depende do projeto e deveria estar no

detalhamento do processo. Adicionalmente, a atividade de se fazer um protótipo de alta fidelidade foi defendida pela empresa.

Em relação à etapa II, a empresa considerou importante ter uma etapa voltada para o reuso e busca por conhecimento. Ressaltou também que muitas vezes, no desenvolvimento de seus produtos, as etapas I e II aconteceram em paralelo, devido às demandas e necessidades para a criação do produto. Essa sugestão foi detalhada e incluída no processo. Sobre a etapa III, de execução, a empresa destacou que o protótipo pode englobar mais funcionalidades do que inicialmente será implementado nesta etapa, pois o *design* pode ser mais fácil e rápido de se fazer. Isso pode gerar problemas em como alinhar o desenvolvimento com o *design*. Além disso, sugeriu criar a possibilidade de se fazer teste real com pessoas para se obter *feedback*, o que passou a fazer parte da descrição do processo como algo opcional.

A segunda empresa entrevistada defendeu o uso da prototipação na primeira etapa do processo, mas com a possibilidade da equipe poder escolher entre alta e baixa fidelidade, pois acreditam que o nível de fidelidade do protótipo depende da demanda do projeto. Com isso o processo foi modificado mudando o nome da atividade de prototipação que antes se chamava de "protótipo de alta fidelidade" para "criar protótipo". Já em relação à etapa III, foi sugerido explicitar na imagem do processo que os *sprints* acontecem mais de uma vez, ou seja, são um ciclo. Neste caso, não consideramos essa modificação necessária, porque isso já é descrito no processo e a imagem utilizada para representar os *sprints* foi baseada na própria figura do *Scrum* que sugere a visualização como ciclos.

No *feedback* da terceira empresa foi sugerido o estudo de aplicações correlatas como fonte de inspiração para a concepção de ideias. Com isso, na descrição dessa etapa do processo, foi acrescentada a opção de se pesquisar aplicações semelhantes a que se quer desenvolver. Assim como a primeira empresa entrevistada, esta concordou que a etapa I e II podem se confundir ao longo do desenvolvimento do projeto, acontecendo em paralelo. A empresa concordou com o uso do quadro *Kanban* para um planejamento micro do tempo e sugeriu ter um planejamento macro, definindo quanto tempo cada etapa e os *sprints* vão durar, além das grandes entregas. Por isso, na descrição da terceira etapa do processo, foi adicionado um cronograma com grandes entregas para se ter um planejamento macro do projeto. Além disso, também foi sugerido que a etapa III, de execução, contemple, em paralelo ao teste final funcional, as atividades de lições aprendidas e análise das estratégias para alimentar a base de conhecimento. Dessa forma, foi colocada na imagem do processo a atividade de análise das lições aprendidas.

Após cada entrevista, o processo foi modificado e, finalmente, após todos os ajustes solicitados, os entrevistados foram novamente apresentados ao processo final e afirmaram que aplicariam nas suas empresas.

A validação realizada foi importante para geração de uma versão inicial do processo de desenvolvimento proposto. Para diminuir as ameaças à avaliação, utilizou-se empresas que trabalham com projetos super ágeis e cujos participantes tem experiência com as técnicas abordadas. Contudo, a avaliação não foi formalmente definida com base em estratégias de experimentação [28]. Desta forma, visando diminuir as ameaças à validade externa, como segunda estratégia de avaliação, planeja-se um estudo de caso para aplicação do processo no desenvolvimento de um projeto real nas empresas.

4 TRABALHOS CORRELATOS

Existem diversos trabalhos que propõe variações de metodologias ágeis, a exemplo de Carvalho et al. [4] aplicada ao desenvolvimento de serviços e Trindade e Lucena [23] aplicada ao controle de rastreabilidade de requisitos. Contudo, poucas são as iniciativas relacionadas ao desenvolvimento de projetos super ágeis.

Em 2004, Abrahamsson et al. [1] teve a iniciativa de propor um processo de desenvolvimento ágil para aplicações móveis, chamado de Mobile-D, compatível com equipes com menos de dez pessoas, trabalhando num mesmo espaço e com o objetivo de entregar uma aplicação totalmente funcional em menos de dez semanas. Essa abordagem foi dividida em cinco iterações, sendo que cada iteração possuía dias de planejar, trabalhar e de entrega. Práticas como integração contínua, programação em par, métricas e melhoria de processos de *software* com métodos ágeis também foram usadas. Na mesma direção, em 2008, Rahimian e Ramsin [14] apresentam uma abordagem de *Design* de Metodologia Híbrida para criar um novo método ágil, voltado para o desenvolvimento de sistemas de *software* para dispositivos móveis. Trata-se de um *framework* que tem como atividades a geração da ideia e a criação de um protótipo de alta ou baixa fidelidade. O processo proposto neste artigo se diferencia do proposto por Abrahamsson et al. [1] por considerar a iteração dentro da etapa e não configura toda a etapa. Além disso, o processo aqui proposto não se limita ao desenvolvimento de aplicativos móveis, se enquadra também em outros cenários, como outros tipos de projetos de inovação ou de prova de conceito. Nenhum dos dois processos citados aborda projetos super ágeis, com ciclos de vida de no máximo duas semanas.

O único artigo encontrado que cita a terminologia super ágil é o de Shelley [17], que traz uma nova técnica de desenvolvimento de *software* escalável e previsível, baseada em práticas de desenvolvimento ágil, utilizada por uma companhia de *software* austríaca. Esta empresa lida com entretenimento, esporte e serviços de apostas online, tendo a necessidade de rápida resposta a mudanças em relação a oportunidades comerciais. Mudanças no método *Scrum* foram introduzidas e chamadas de práticas super ágeis, como permitir que mudanças fossem feitas no meio de um *sprint*, variar o tamanho de um *sprint* de acordo com as histórias exigidas, produzir e comunicar novas estimativas para o *sprint* diariamente, dentre outras.

Assim como em Shelley [17], o processo, objeto desta pesquisa, também utiliza uma adaptação do *Scrum*, a qual é aplicada na etapa de execução. O processo propõe mudanças na duração de um *sprint*, mas, ao invés de variar o tamanho como Shelley [17], utiliza *sprints* de até 2 dias, sendo que um *sprint* no *Scrum* dura de 2 a 4 semanas.

5 CONCLUSÃO

É notória a necessidade de um método de desenvolvimento de *software* que atenda à demanda de desenvolvimento de *software* super ágil a exemplo das vivenciadas por *startups*. Nesta direção, este artigo propõe um processo de desenvolvimento para sistemas super ágeis com base tanto nas técnicas e metodologias ágeis atuais quanto nas demandas das empresas inseridas neste contexto. Desta forma, pode contribuir para aumentar a qualidade do produto de *software* geral, pois além de selecionar práticas relevantes para projetos super ágeis, as organiza de forma sistemática conduzindo a sua utilização.

O processo foi construído a partir do estudo de técnicas, métodos e ferramentas de desenvolvimento de *software*. Além disso, uma pesquisa de campo com profissionais da área foi essencial para se conhecer mais sobre as necessidades das empresas e *startups*, como também para se obter informações sobre métodos de desenvolvimento como o *Kanban* e o *Design Sprint*, pouco explorados antes na fase de estudos. Após a concepção do processo, ele passou por uma avaliação qualitativa inicial realizada através de entrevistas em empresas desenvolvedoras de *software* e se mostrou aderente às necessidades destas empresas. Os entrevistados se mostraram positivos e afirmaram que aplicariam o processo no contexto de suas empresas ou *startups*. Atualmente, está sendo definido um estudo de caso para ser replicado em algumas destas empresas no desenvolvimento de projetos reais. Com base nos resultados deste estudo de caso, pretende-se evoluir o processo e implementá-lo em um ambiente automatizado que facilite a sua utilização.

REFERÊNCIAS

[1] Pekka Abrahamsson, Antti Hanhineva, Hanna Hulkko, Tuomas Ihme, Juho Jäälinoja, Mikko Korkala, Juha Koskela, Pekka Kyllönen, and Outi Salo. 2004. Mobile-D: an agile approach for mobile application development. In *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications (OOPSLA '04).* ACM, New York, NY, USA, 174-175. DOI: http://dx.doi.org/10.1145/1028664.1028736

[2] Kent Beck and Cynthia Andres. 2004. *Extreme Programming Explained: Embrace Change* (2nd Edition). Addison-Wesley Professional.

[3] Kleber Bernardo. 2014. Kanban: Do início ao fim! 2014. Retrieved September 2, 2016 from http://www.culturaagil. com.br/kanban-do-inicio-ao-fim/.

[4] Felipe Carvalho, Leonardo Azevedo and Gleison Santos. 2014. A Method for Service Agile Construction. In *X Simpósio Brasileiro de Sistemas de Informação* SBSI, Londrina, PR, BR.

[5] David Cohen, Mikael Lindvall and Patrícia Costa. 2004. An introduction to agile methods. In *Advances in Computers Volume 62*, Vol. 62 (2004), pp. 1-66, DOI: https://doi.org/10.1016/S0065-2458(03)62001-2

[6] Torgeir Dingsøyr, Sridhar Nerur, VenuGopal Balijepally, and Nils Brede Moe. 2012. A decade of agile methodologies. J. Syst. Softw. 85, 6 (June 2012), 1213-1221. DOI: http://dx.doi.org/10.1016/j.jss.2012.02.033

[7] Google. Design Sprint Methods, playbook for startups and designers. 2015. Retrieved June 10, 2016 from https://developers.google.com/design-sprint/downloads/DesignSprintMethods.pdf .

[8] Henrik Kniberg and Mattias Skarin. 2009. Kanban and Scrum, making the most of both. Retrieved June 12, 2016 from https://www.infoq.com/minibooks/kanban-scrum-minibook .

[9] J. J. Kolb. 2013. Planejamento em ondas Sucessivas. Retrieved September 11, 2016 from http://jkolb.com.br/ planejamento-em-ondas-sucessivas/ .

[10] Robert C. Martin and Micah Martin. 2011. *Princípios, Padrões e Práticas Ágeis em C#* (1st Edition). Porto Alegre: Bookman. ISBN 9788577808427.

[11] Marvel. Marvel: Simple design, prototyping and collaboration.2016. Retrieved October 11, 2016 from https://marvelapp.com/.

[12] Mathias Meyer. 2014. Continuous Integration and Its Tools. In *IEEE Software*, v. 31, n. 3, p. 14–16, 2014. DOI: 10.1109/MS.2014.58

[13] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. 2005. *Software Product Line Engineering: Foundations, Principles and Techniques.* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[14] V. Rahimian and R. Ramsin. 2008. Designing an agile methodology for mobile software development: A hybrid method engineering approach. In *Second International Conference on Research Challenges in Information Science*, Marrakech, 2008, pp. 337-342.

[15] Eric Ries. 2011. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Business*. Crown Business, USA.

[16] SEBRAE. 2007. Critérios de Classificação de Empresas: MEI - ME - EPP. Retrieved May 14, 2016 from http://www.sebrae-sc.com.br/leis/default.asp?vcdtexto=4154>.2007 .

[17] C. C. Shelley. 2011. *The Evolution of a Super Agile Software Development Capability*. Oxford Software Engineering, Oxfordshire, England.

[18] D. V. S. Silva, F. A. O. Santos e P. S. Neto. 2012. Os benefícios do uso de Kanban na gerência de projetos de manutenção de software. In *VIII Simpósio Brasileiro de Sistemas de Informação - Trilhas Técnicas.* SBSI, São Paulo, SP.

[19] F. R. Silva, R. S. Pinto and G. k. Akabane. 2014. Startups de empresas de base tecnológica e a computação na nuvem: a viabilização de um modelo mais dinâmico. In *IX Workshop de Pós-Graduação e Pesquisa do Centro Paula Souza*. ISSN 2175-1897.

[20] T. S. Silva. 2009. *Método Nihei: conjunto de atividades para utilização e avaliação de uma ferramenta de webquest com web semântica em sala de aula*. Universidade Católica do Salvador, Salvador, BA.

[21] Ian Sommerville. 2010. Software Engineering (9th ed.). Addison-Wesley Publishing Company, USA

[22] Wouter Spoelstra, Maria Iacob, and Marten van Sinderen. 2011. Software reuse in agile development organizations: a conceptual management tool. In *Proceedings of the 2011 ACM Symposium on Applied Computing (SAC '11)*. ACM, New York, NY, USA, 315-322. DOI=http://dx.doi.org/10.1145/1982185.1982255

[23] Gabriela Trindade and Márcia Lucena. 2016. Rastreabilidade de Requisitos em Metodologias Ágeis: um Estudo Exploratório. In *XII Brazilian Symposium on Information Systems*. SBSI, Florianópolis, SC.

[24] VERSIONONE. 2016. The 10th annual state of agile report. Retrieved March 30, 2016 from http://www.agile247.pl/wp-content/uploads/2016/04/VersionOne-10th-Annual-State-of-Agile-Report.pdf.

[25] D. Vieira. 2014. *Scrum: A Metodologia Ágil Explicada de forma Definitiva*. Retrieved September 20, 2016 from http://www.mindmaster.com.br/scrum/.

[26] Laurie Williams. 2012. What agile teams think of agile principles. Commun. ACM 55, 4 (April 2012), 71-76. DOI: https://doi.org/10.1145/2133806.2133823.

[27] Laurie Williams, Robert R. Kessler, Ward Cunningham, and Ron Jeffries. 2000. Strengthening the Case for Pair Programming. IEEE Softw. 17, 4 (July 2000), 19-25. DOI=http://dx.doi.org/10.1109/52.854064 .

[28] C. Wohlin, P. Runeson, M. Host., M.C. Ohlsson, B. Regnell, A. Wesslen. 2012. Experimentation in Software Engineering. Springer, New York (2012). DOI: 10.1007/978-3-642-29044-2.