UMA ANÁLISE DE ESCALABILIDADE DE MÉTODOS ÁGEIS EM GRANDES PROJETOS COM MÚLTIPLOS TIMES

Luckeciano Carvalho Melo
Instituto Tecnológico de Aeronáutica
H8A, Apto. 113, CTA
12.2280-460 – São José dos Campos / SP
Bolsista PIBIC - CNPq
luckeciano@gmail.com

Adilson Marques da Cunha
Instituto Tecnológico de Aeronáutica
Divisão de Ciência da Computação
Praça Marechal Eduardo Gomes, 50
12.229-900 – São José dos Campos / SP
cunha@ita.br

Cassiano Monteiro
Instituto Tecnológico de Aeronáutica
Rua João Teixeira Neto, 102 – Apto. 702 – Pq. Res. Aquarius
12.246-160 – São José dos Campos / SP
cassianomonteiro@gmail.com

Resumo. *Nos últimos anos, a adoção de Metodologias Ágeis por parte de times de desenvolvimento transformou, de forma disruptiva, a indústria de desenvolvimento de softwares. Contudo, a abordagem ágil clássica possui certas limitações, relacionadas ao seu direcionamento para pequenos times, e por requisitar que todos estes trabalhem juntos. Nesse contexto, surgem os métodos de escalabilidade das práticas ágeis, a fim de solucionar esse motivo de consternação da comunidade ágil. Objetiva-se, portanto, a análise dos principais problemas relacionados com a escalabilidade de modelos ágeis, bem como o estudo de metodologias de destaque dentro de uma perspectiva de estudo do ágil. Com isto, acredita-se que o surgimento de diversos paradigmas de escalabilidade expõe a real necessidade da agilidade em grandes corporações e revela que este problema possui não apenas diversas abordagens como também se apresenta em aberto para novos modelos de gestão pautados no ágil.*

Palavras chave: Metodologias Ágeis, Escalabilidade, Múltiplos Times, Engenharia de Software.

1. Introdução

A adoção de métodos ágeis na indústria de Tecnologias da Informação (TI) e no desenvolvimento de software têm se mostrado bastante vantajosos em relação aos métodos tradicionais de desenvolvimento. Estudos recentes indicam um número crescente de empresas abandonando o desenvolvimento tradicional em prol de métodos ágeis, principalmente por conta dos benefícios trazidos em relação à melhoria de produtividade, maximização de valor e otimização de velocidade de resposta às demandas de mercado.

Métodos Ágeis, como Scrum e XP, designam-se originalmente para o uso em pequenos times, nos quais seus integrantes trabalham no mesmo ambiente em comunicação constante entre si. Atualmente, muitas organizações desenvolvem grandes sistemas com múltiplos times distribuídos em variadas localizações geográficas - e buscam o benefício destes métodos. Logo, faz-se necessário escalar tais práticas (Paasivara, Lassenius, 2011).

Os benefícios da agilidade encontram-se desde a sua capacidade de gestão de mudanças até a melhoria da qualidade do produto e a produtividade do time. Desta forma, a transição para o ágil e o *lean* (Lean Enterprise Institute, 2000) se tornou necessidade para diversos mercados.

Contudo, a abordagem ágil clássica possui certas limitações, relacionadas ao seu direcionamento para pequenos times e por requisitar que todos estes trabalhem juntos. Para pequenos projetos, isto não causa problema. Porém, ao se analisar grandes empresas que buscam agilidade nos seus negócios ou até mesmo *startups* em fase de crescimento, estas limitações impossibilitam a implementação destes métodos da maneira "tradicional".

Nesse contexto, surgem os métodos de escalabilidade das práticas ágeis, a fim de solucionar esse motivo de consternação da comunidade ágil (InfoQ, 2014). Todavia, escalar envolve grandes desafios, como coordenação de múltiplos times, problemas na comunicação de times altamente distribuídos - e até questões mais técnicas, como lacunas na arquitetura de grandes componentes de software e na análise de requisitos. Logo, escalar ágil não se trata apenas de solucionar as limitações das metodologias clássicas, mas também tratar da complexidade de projetos e da estrutura organizacional de empresas.

Voltando-se para o escopo previamente definido, este trabalho de pesquisa visa a análise dos principais problemas relacionados com a escalabilidade de modelos ágeis, bem como o estudo de metodologias de destaque dentro de uma perspectiva de estudo do ágil. A linha de ação torna-se, portanto, reconhecer e fazer um comparativo entre estas

abordagens, a fim de analisar em qual situação cada uma se torna mais vantajosa, bem como apontar quais seus principais desafios de implementação.

2. Materiais e Métodos

Este trabalho dá continuidade à uma pesquisa e análise sobre Metodologias Ágeis. Em um primeiro momento, esta focou na adoção do ágil no desenvolvimento de software *safety-critical* (Melo, Cunha, Monteiro, 2015) e, no seu andamento, foi realizada uma pesquisa acerca destes métodos, descrevendo seus princípios e práticas mais comuns, comparando-os com práticas tradicionais e analisando a sua adoção no mercado de Tecnologia da Informação da atualidade.

No segundo estágio da pesquisa, analisa-se a abordagem ágil sob uma nova perspectiva: sua capacidade de escalabilidade em empresas grandes e/ou em rápido crescimento. Na primeira etapa, realizou-se um levantamento bibliográfico das metodologias de escalabilidade de Métodos Ágeis. Montou-se, então, um cronograma de métodos baseados na relevância para a comunidade ágil e na disponibilidade de material acadêmico para estudo.

Posteriormente, iniciou-se o estudo de grandes abordagens do ágil em escala: SAFe (Leffingwell, 2010); SA@S (Spotify Labs, 2013); DAD (Ambler, Lines, 2012); e LeSS (LeSS Company, 2014), analisando práticas, estrutura organizacional e alguns estudos de caso - recorrendo-se, para cada abordagem, a contatos com seus criadores. Por fim, ao término do estudo das principais metodologias ágeis em escala, identificou-se os principais desafios encontrados no fenômeno de escalar agilidade em organizações e comparou-se as abordagens que cada método propõe para solucioná-los.

Dessa forma, ao fim deste trabalho, dispõe-se de conhecimentos e prática nos aspectos explorados durante a pesquisa. Dentre eles, destacam-se: os princípios e dificuldades da escalabilidade de métodos ágeis; os principais modelos de escalabilidade do ágil; e estrutura, práticas e estudos de caso dos modelos de escalabilidade em destaque (SAFe, SA@S, DAD e LeSS).

3. Contextualização

3.1. Dificuldades de Escalar o Ágil

A motivação inicial desta pesquisa se encontrava no problema do gerenciamento de múltiplos times altamente distribuídos. O desafio, portanto, era analisar a abordagem de metodologias para diversos times e sua integração, bem como características de tamanho e dispersão geográfica.

Contudo, ao passo que a análise dessas abordagens se desenvolveu, surgiram novos **desafios**, ainda relacionados com os métodos ágeis, para adotar o perfil de agilidade em grandes organizações (Leffingwell, 2007)(Ambler, Lines, 2012):

- **Falta de análise de requisitos e documentos de especificação** - para grandes empresas, a comunicação se torna cada vez mais complicada. Com variados perfis profissionais trabalhando em um mesmo projeto (desenvolvedores, analistas de negócio, gerentes de produto, etc.), faz-se necessário que todos estejam alinhados com a visão do negócio e do produto. Caso contrário, surgirão ambiguidades e mal-entendidos. Logo, surge o problema de gerar requisitos e documentos de especificação eficientes em larga escala – sem a burocracia de extensas e detalhadas documentações;
- **Ambiente físico** - o ambiente ágil é aberto e todos trabalham juntos, sem divisões. Para alguns, este cenário parece caótico e sem supervisão, confrontando, muitas vezes, o estilo de vários círculos empresariais;
- **Conformidade regulatória** – projetos complexos de software podem demandar o seguimento de certas normas específicas de seu domínio;
- **Domínio nas iterações** - uma grande empresa desenvolve suas atividades em uma rapidez maior do que se tomam decisões estratégicas relacionadas a um grande produto ou a própria empresa. Deste modo, ela precisa realizar planejamentos relacionados ao desenvolvimento do produto, a médio e longo prazos. Porém, o ágil confronta a ideia de grandes planejamentos, devido a pouca garantia destes planos maiores;
- **Execução dos processos ágeis e reuniões** - quando se trata de larga escala, muitos dos processos de fácil implementação das metodologias tradicionais se tornam complexos e pouco efetivos. Além disto, a logística de reuniões se torna inviável;
- **Sincronia de times** - quando se trata de apenas um time ágil, a linha do tempo se resume a iterações até o lançamento. Porém, quando se trata de vários times e pretende-se lançar algo dentro de determinado prazo, pode ocorrer dos times se tornarem assíncronos. Isto ocorre devido a fatores relacionados a integração dos componentes de cada time e prejudica a agilidade do sistema como um todo. Basta notar que o componente mais lento guia a data de entrega e, geralmente, não se sabe quem este é, até chegar lá.

Por outro lado, existem desafios que surgem da própria organização empresarial:

- **Processos e políticas para gerenciamento de projetos formalizados e rígidos (disciplina empresarial)** - empresas grandes possuem certos procedimentos engessados e burocráticos que resistem às mudanças e impedem os reais benefícios do ágil.
- **Empresa com estrutura orientada por cargos e não pelo produto** - comum em empresas mais antigas, este problema não foca no desenvolvimento do produto, o que tira a agilidade no diagnóstico dos problemas deste;
- **Comunicação interna** - em uma grande empresa, fora os times de desenvolvimento, existem times de marketing, vendas, bem como pessoas encarregadas de gerir o produto ou a própria empresa. Há a necessidade de manter todas estas pessoas atualizadas a respeito do que se desenvolve, de maneira rápida e ágil; e
- **Modelo de gerenciamento executivo** - métodos ágeis clássicos não possuem uma linha de ação "*top-down*" (de cima para baixo). O desafio, portanto, encontra-se em adaptar o existente e criar ferramental de gestão executiva, desde a prospecção de impedimentos organizacionais até métricas para avaliação de desempenho e progresso.

Adicionalmente, surgem também **desafios** relacionados à parte técnica de criação do produto (Kniberg, Ivarsson, 2012):

- **Arquitetura do software** - trata-se de um ponto bem polêmico e alvo de críticas em algumas metodologias de escalabilidade ágil. A abordagem ágil clássica tem como princípio que a arquitetura deve emergir da implementação dos requisitos e não ser desenhada totalmente com antecedência, e qualquer problema pode ser resolvido com refatoração. Contudo, para projetos volumosos, com muitos componentes, a refatoração se torna muito cara e pouco eficiente. Precisa-se ser mais cauteloso e repensar como lidar com a arquitetura do software;
- **Complexidade de domínio** – escalabilidade também significa construir soluções que possuam alto nível técnico em um domínio específico, necessitando-se, muitas vezes, de conhecimento especializado; e
- **Compartilhamento de conhecimento** - embora não seja uma prática totalmente presente e necessária, empresas de tecnologia bem-sucedidas possuem o princípio de nivelar o conhecimento técnico dos seus times, para evitar inclusive que estes percam muito tempo enfrentando o mesmo problema. Logo, o desafio, neste caso, está em criar uma organização que facilite e engaje os colaboradores no compartilhamento de conhecimento.

3.2. Métodos de Escalabilidade Ágil

Neste trabalho de pesquisa, estudaram-se quatro métodos ou metodologias de destaque em escalabilidade ágil: SAFe; SA@S; DAD; e LeSS. Cada um deles possui particularidades e discrepâncias, devido ao perfil de organização em que se enquadram, e variam de acordo com a complexidade corporativa nas quais se aplicam, conforme descrito a seguir.

3.2.1. SAFe: Scaled Agile Framework

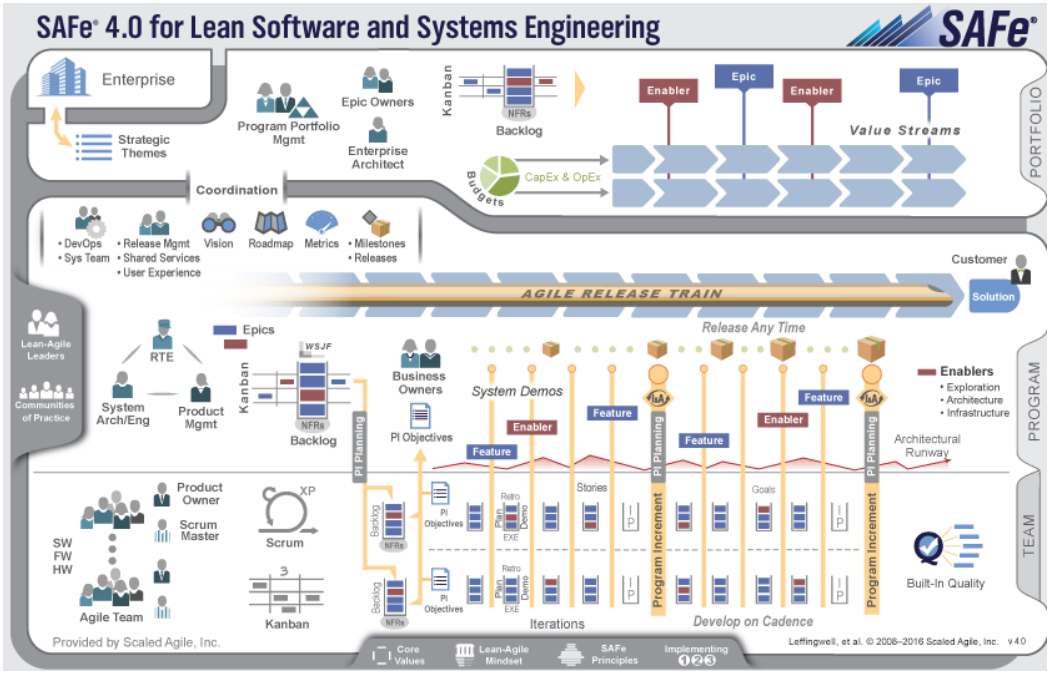

Figura 1: Visão geral do SAFe (Leffingwell, 2010).

O Scaled Agile Framework, conhecido como SAFe, surgiu em 2011 com sua primeira versão e foi desenvolvido por Dean Leffingwell. Trata-se de uma metodologia baseada em experiências bem-sucedidas de adoção do desenvolvimento lean e ágil, em nível de grandes empresas. A Fig. 1 representa uma visão geral do SAFe (Leffingwell, 2010).

O SAFe se estrutura na orientação em três grandes níveis (de time, de programas e de portfólio), com grande atenção para fluxo de valor (Leffingwell, 2011). Esta divisão se relaciona com o papel dos colaboradores dentro do negócio: as posições da base se envolvem mais com o desenvolvimento do produto, enquanto as mais do topo se responsabilizam por gestão de produtos, pessoas e direcionamento estratégico. Tais níveis são descritos a seguir.

No **nível de time**, criam-se equipes entre 7 a 9 membros. Estes definem, constroem e testam *User Stories* em uma série de iterações e *releases*. Em pequenas empresas, existem apenas poucos times. Em grandes, por outro lado, os times trabalham unidos (*pods*), construindo maiores funcionalidades em produtos completos, componentes de arquitetura, subsistemas, etc. O responsável por gerenciar a lista de *User Stories* e as necessidades do time chama-se *Product Owner* (PO).

Trata-se, portanto, do coração da empresa, criando os componentes, testando-os e entregando valor ao cliente. Compõem os times: desenvolvedores e testadores (de quatro a seis pessoas); um *Scrum Master*; e um *Product Owner*. No dia-a-dia de trabalho, o time é suportado por vários profissionais (especialistas em documentação, banco de dados, QA, gerenciamento de código, TI) de maneira a tornar o time capaz de definir, desenvolver, testar e entregar software no sistema de gerenciamento de configurações do projeto.

Times são organizados para entregar funcionalidades ou componentes. Os times de componentes focam em compartilhar estruturas, subsistemas e componentes de arquitetura. Já times de funcionalidades focam em iniciativas verticais de entrega de valor, voltadas para o usuário.

Em uma empresa grande, tipicamente, juntam-se alguns poucos times (até dez) para cooperar na construção de uma funcionalidade maior, um sistema ou subsistema – no **nível de programa**. Neste nível, os profissionais atuam na organização do desenvolvimento de funcionalidades em larga escala por múltiplos times em um "*Agile Release Train*" (ART) sincronizado. O ART se trata de um conjunto padrão de iterações de prazos e qualidade fixados, mas com escopo variável, gerando incrementos de produtos entregáveis. Para isto, estes profissionais gerenciam os planejamentos de *releases* e mantém a Visão do produto. O conteúdo primário da Visão é um conjunto de funcionalidades priorizadas com objetivo de entregar benefícios aos usuários. Adicionalmente, a Visão deve conter vários requisitos não funcionais para garantir que o sistema cumpra seus objetivos.

Na função de gerenciamento de portfólio (**nível de portfólio**), incluem-se pessoas, times e organizações dedicadas ao investimento na empresa, de acordo com sua estratégia de negócio. Aqui, encontram-se dois novos conceitos: temas de investimento e épicos, que juntos criam a Visão do Portfólio.

Temas de investimento estabelecem os objetivos relativos a investimento para a empresa ou unidade de negócio. Estes temas guiam a Visão para todos os programas e novos épicos são derivados destes temas. O objetivo destes épicos é, por fim, propor valor ao produto, fornecendo diferenciação no mercado e vantagem competitiva. Por outro lado, épicos representam a expressão de mais alto nível da necessidade do cliente. Eles são gerados para entregar valor de um tema de investimento e são identificados, priorizados, estimados e mantidos no portfólio.

3.2.2. SA@S: Scaling Agile @ Spotify

O Scaling Agile @ Spotify, SA@S, trata de uma metodologia de escalabilidade ágil documentada formalmente pela primeira vez em 2012 (Kniberg, Ivarsson, 2012) por Henrik Kniberg e Anders Ivarsson, ambos técnicos de agilidade da *startup* em crescimento Spotify.

O SA@S surgiu, devido ao grande crescimento interno que a empresa passou nos últimos anos. Deste modo, as ferramentas do Scrum – metodologia ágil utilizada pela startup em seu início – estavam pouco efetivas e algumas vezes se tornavam impedimentos (Spotify Labs, 2014). Logo, estas se tornaram opcionais aos times e iniciou-se o desenvolvimento de métodos que entregassem agilidade nas condições de crescimento da organização.

Esta metodologia se destaca por tratar de um perfil de empresa específico, mas que se tornou bastante comum nos últimos anos. Além disto, desenvolveram-se seus princípios de maneira bastante analítica, de acordo com os próprios resultados da empresa. A sua estrutura impressiona, por ser uma organização matricial com foco total em entrega e forte direcionamento para desenvolvimento técnico dos seus colaboradores, encorajando pesquisa e inovação.

A sua estrutura se baseia na formação de esquadrões (*squads)*, tribos (*tribes)*, capitanias (*chapters)* e guildas (*guilds)*, como mostrado na Fig. 2.

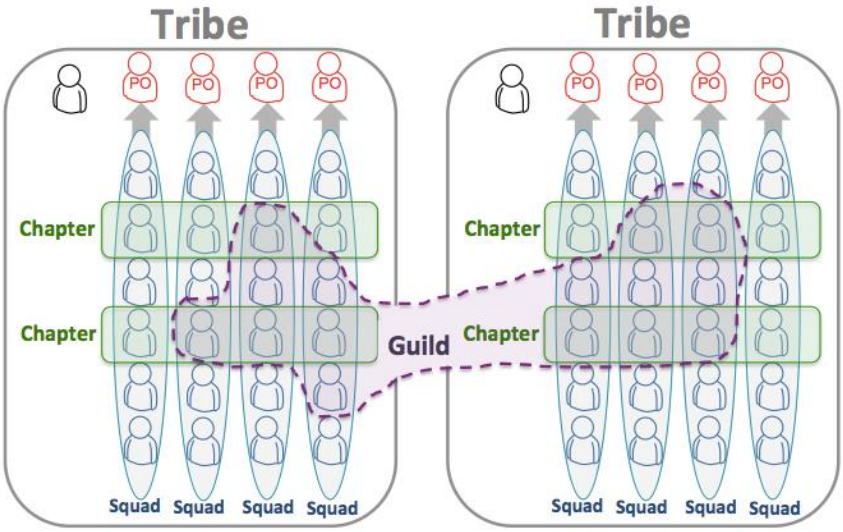

Figura 2: Estrutura organizacional com SA@S.

Um esquadrão assemelha-se com o time Scrum e desenha-se como uma mini *startup*. O time possui todas habilidades e ferramentas necessárias para desenhar, desenvolver, testar, e entregar produto. Este se auto organiza e pode se gerir com Scrum, Kanban ou até uma mescla de ambos. Cada esquadrão tem uma missão de longo prazo, como melhorar o cliente Android, escalar o serviço de *backend*, fornecer soluções de pagamento, entre outros. Encorajam-se esquadrões a aplicarem princípios de Lean Startup, como entrega frequente e validação de aprendizagem.

A fim de promover aprendizado e inovação, cada esquadrão é encorajado a investir 10% do seu tempo em "*hack days*", na pesquisa de novas ideias e para se manter atualizado em relação as novas ferramentas e técnicas que, algumas vezes, causam grandes inovações no produto. Idealmente, cada time é totalmente autônomo, com contado direto aos *stakeholders* e sem dependência com outros esquadrões. Adicionalmente, ocorrem pesquisas trimestrais para cada esquadrão (*health check models*), as quais auxiliam no foco para melhorias e para encontrar qual tipo de suporte organizacional é necessário.

Uma tribo, por sua vez, é uma coleção de esquadrões que trabalham em áreas relacionadas – como music player ou infraestrutura de *backend*. Caracterizam-se tribos como incubadoras dos esquadrões, com um grau justo de autonomia e liberdade. Cada tribo tem como função prover o melhor ambiente para seus esquadrões. Estas tribos encontram-se no mesmo escritório, normalmente lado a lado, e as salas de estar promovem colaboração entre esquadrões.

A capitania, por outro lado, envolve uma pequena família de pessoas que compartilham das mesmas habilidades e trabalham na mesma área de competência, na mesma tribo. Estes se encontram regularmente para discutir sobre a área de *expertise* e os desafios específicos.

Uma guilda é uma "comunidade de interesse" mais orgânica e de grande alcance – um grupo de pessoas que querem compartilhar conhecimento, ferramentas, código e práticas. Enquanto capitanias caracterizam-se por ser locais, guildas geralmente cruzam toda a organização. Exemplos: guilda de tecnologia web, guilda de testes, guilda de *coaches* do ágil.

Capitanias e guildas surgem da seguinte situação: um testador do esquadrão A enfrenta um problema pelo qual o testador B solucionou na semana anterior. Se todos os testadores pudessem ficar juntos, eles poderiam compartilhar conhecimento e criar ferramentas para beneficiar todos os esquadrões. Se todos os esquadrões fossem totalmente independentes, não se trataria de uma organização só – mas de várias pequenas companhias. Para isto é que existem as capitanias e guildas. Eles são como colas para manter a organização unida, provendo economia em escala sem sacrificar muita autonomia.

A organização proposta por este modelo ágil possui certa formação matricial. Entretanto, diferentemente de organizações que juntam pessoas com habilidade similares em departamentos funcionais e as atribui em projetos, reportando-se a um gerente funcional, o SA@S trata de uma matriz ponderada em direção de entrega.

No SA@S, a dimensão vertical trata de como as pessoas são agrupadas fisicamente e onde passam maior parte do tempo. Trata-se de pessoas com diferentes habilidades colaborando e se auto organizando para entregar produtos. A dimensão horizontal serve para compartilhamentos de conhecimento, de ferramentas e de código. Portanto, enquanto a dimensão vertical trata de "o quê", a horizontal trata de "como".

Esta conformação segue o modelo de "empreendedor e professor". O *Product Owner* atua como "empreendedor", focando em entregar um bom produto, enquanto o líder de capitania atua como "professor", focando em excelência técnica. Esta conformação gera uma tensão "saudável": enquanto o PO procura acelerar, o professor tende a desacelerar e construir coisas corretamente.

3.2.3. DAD: Disciplined Agile Delivery

O DAD – Disciplined Agile Delivery – consiste em um *framework* criado na IBM por Scott Ambler e Mark Lines, que se caracteriza por um perfil híbrido de outros paradigmas de agilidade ao fornecer uma base para escalar soluções ágeis de entrega a nível organizacional. A sua ideia principal concentra-se em construir um paradigma para tomada de decisão a partir dos métodos ágeis tradicionais (Schiffer, 2014).

O *framework* do DAD baseia-se em pequenas iterações, ao longo do ciclo de vida do projeto, como mostrado na Fig. 3. Dividem-se estas iterações em três fases, dependendo do nível de maturidade que este se encontra.

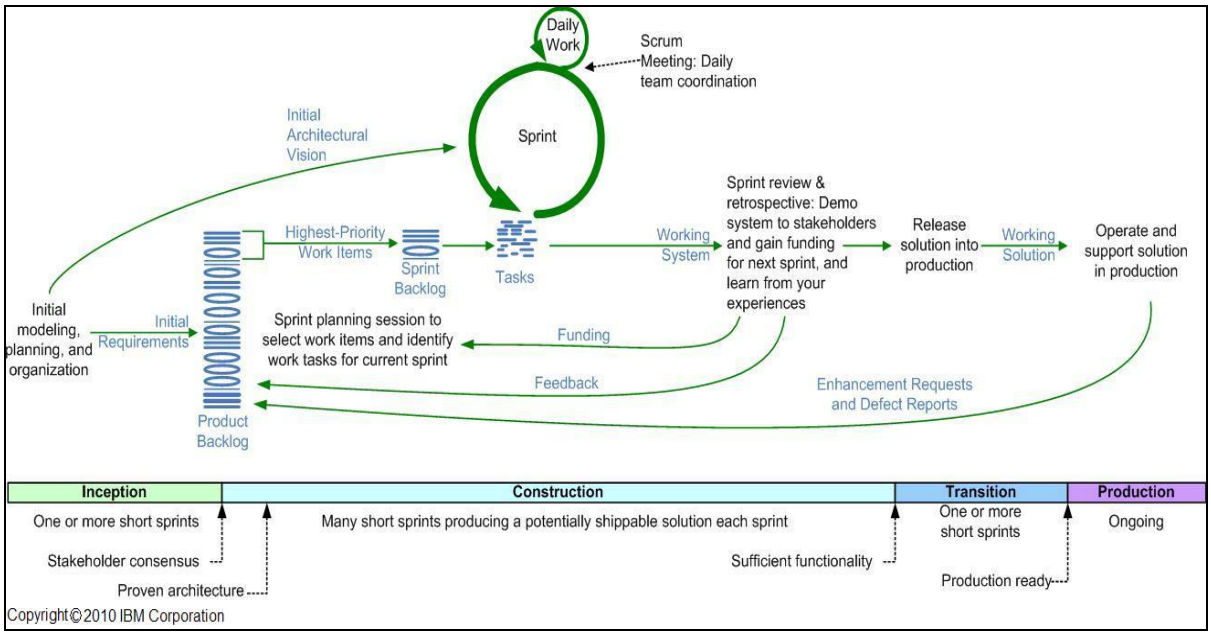

Figura 3: Ciclo de vida de um software no DAD.

A fase de Incepção (*Inception)* trata da iniciação do projeto (análogo a *Sprint* zero do Scrum), executada antes de se começar a construir de fato a solução. O objetivo fundamental desta fase encontra-se em criar uma base sólida para o projeto na qual este pode ser bem-sucedido da maneira mais rápida e barata possível. Para isto, deve-se esclarecer o desafio de negócio a ser resolvido, identificar uma solução técnica viável, planejar a abordagem, construir os times e configurar o ambiente de trabalho e, principalmente, criar uma Visão para o projeto. Adicionalmente, esta fase deve alinhar e gerar consenso relativo com os *stakeholders* a respeito de escopo de projeto, cronograma, riscos e estratégia de arquitetura.

Posterior a esta fase inicial, ocorre a fase de Construção (*Construction*), que parte de provar as conjecturas técnicas formadas anteriormente por meio de código funcional e então desenvolver a solução potencialmente consumível, de maneira iterativa, tratando as necessidades de todos os *stakeholders* envolvidos.

Essa fase assemelha-se bastante com a estratégia das metodologias ágeis tradicionais, e a ela englobam-se as principais práticas técnicas provenientes do Scrum/XP. Por outro lado, o DAD configura-se como método adaptativo, de maneira que várias práticas do desenvolvimento (e dos planejamentos envolvidos) podem ser executadas de diferentes formas – cabe a equipe decidir qual maneira se encaixa melhor ao seu contexto.

Ao final da fase de Construção, bem como ao fim de cada *sprint* ou *release* interna, expõe uma solução atual para os *stakeholders*, a fim de obter *feedback* acerca do que se desenvolveu.

A última fase de um projeto DAD denomina-se Transição (*Transition)*. Diferentemente de projetos sem escala, transitar a solução para produção em grandes projetos torna-se um esforço importante e necessita-se da colaboração de *stakeholders* externos, como especialistas de segurança e DevOps. Nesta fase, desenvolvem-se os últimos testes relacionados ao fim do ciclo de vida e à entrega. Nela, ocorre a migração dos componentes para o ambiente de produção, além da finalização da documentação do software.

A fase de Transição simboliza o reconhecimento das complexidades e interdependências de entregar soluções críticas em grandes organizações. Portanto, o DAD possui uma abordagem disciplinada para gerenciar a entrega da solução, reconhecendo, também, a necessidade de colaborar continuamente com *stakeholders* externos a preparar e minimizar os riscos de entrega.

Ao longo de todo o ciclo de vida de um projeto DAD e em todas as escalas (dia, iteração, *release* ou fase do projeto), orientam-se as atividades por meio do modelo 3C: Coordenar, Colaborar e Concluir. Este ritmo de trabalho objetiva orientar as atividades dentro de um período de tempo. Por exemplo, a nível de fase do projeto, há correspondência entre as fases e o modelo (a incepção como coordenação; Construção como colaboração e transição

como conclusão). Já em nível de iteração, inicia-se com um workshop de planejamento (Incepção), desenvolvem o trabalho de implementação (Construção) e finalizam com demonstração e retrospectiva (Conclusão).

A formação de times em DAD possui diferenças em relação as metodologias ágeis tradicionais. No DAD, definem-se os conceitos de cargos, que podem ser primários ou secundários. Os primários englobam o time Scrum tradicional, com a inclusão do *architecture owner*, que trabalha na modelagem técnica do sistema ao prover a melhor solução viável em termos de arquitetura. Além disto, o termo *stakeholder* em DAD engloba vários perfis: *usuários finais*; *principals* (gerentes de negócios), *partners* (equipes de operações); e *insiders* (próprio time de desenvolvimento e fornecedores de serviço técnico para a solução).

Os cargos secundários, por sua vez, aparecem em casos de escalabilidade e geralmente relacionam-se com a complexidade do projeto, compondo-se de especialistas, testadores independentes, experts técnicos e de domínio, e integradores do sistema.

De forma sintética, o time em DAD distribui as responsabilidades dos cargos tradicionais de desenvolvimento para os membros do time ágil que possuem perfil "especialista generalista". A Fig. 4 mostra um exemplo de time DAD para projetos de grande escala, onde necessita-se de todos os cargos definidos na metodologia.

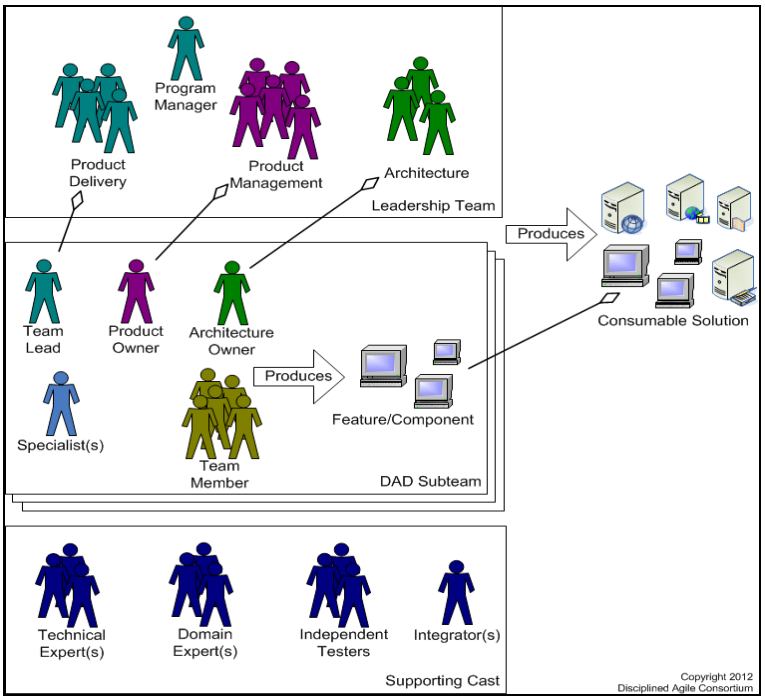

Figura 4: Time DAD para projetos escaláveis.

Por fim, vale ressaltar a maneira com que o DAD trata times dispersos geograficamente. Nesta metodologia, times não colocalizados demandam que o projeto invista mais tempo em planejamento e comunicação interna. Desta forma, deve-se investir mais nas atividades da fase de Incepção, definindo linhas de ação e convenções de desenvolvimento claras. Adicionalmente, deve-se adotar ferramental automatizado para o fluxo de trabalho e informação. Finalmente, deve-se maximizar a independência dos subtimes, bem como adotar responsáveis por compartilhar informações entre estes ("embaixadores").

3.2.4. LeSS: Large-Scale Scrum

O LeSS (Large-Scale Scrum) é uma metodologia de escalabilidade ágil proposta por Craig Larman e Bass Vodde que possui uma ideia bastante simples: aplicar o Scrum regular no desenvolvimento de larga escala (Schiffer, 2014). Isto significa que o papel do LeSS se concentra em aplicar o Scrum da maneira mais genuína possível em grandes projetos, reduzindo-se ao máximo a criação de novas estruturas, cargos, artefatos e eventos.

A metodologia, como uma versão em escala do Scrum tradicional, mantém diversas práticas e ideias desta (Larman, Vodde, 2016): uma única *product backlog;* uma definição de "*done*" para todos os times; um incremento entregável do produto ao final de cada *Sprint*; um único *product owner* (por existir poucos cargos, permite-se ao LeSS escalar o product owner efetivamente); vários times completos e funcionais, além de uma única *sprint* (ou seja, times são sincronizados).

Portanto, LeSS é Scrum: não se trata de uma forma melhorada, mas sim uma maneira de como aplicar os princípios e elementos do Scrum, de forma escalável. A metodologia baseia-se em um processo empírico de controle, inspecionando e adaptando o produto no decorrer do tempo. Transparente, foca em aprendizado e evita desperdícios. O

foco encontra-se inteiramente no produto, centrado no cliente. Desenvolve-se melhoria contínua em direção à perfeição, tanto no produto como nos processos, a partir de um pensamento de melhoria do sistema e *lean*.

A metodologia fornece dois *frameworks* distintos: o Basic LeSS, para times de duas a oito pessoas e o LeSS Huge, com times de mais de oito pessoas e alguns milhares de colaboradores no produto. Os elementos (responsabilidades, artefatos e eventos) do LeSS básico são, essencialmente, os mesmos do Scrum de um único time. As principais diferenças ocorrem em algumas regras e orientações, baseadas no relacionado anteriormente.

Quando a equipe cresce muito, dividir para conquistar torna-se inevitável. No desenvolvimento em larga-escala tradicional, geralmente, a divisão ocorre em pequenos grupos funcionais ou em grupos de componentes (equipe de UI, equipe de administração de dados, etc.). Isto, entretanto, gera grandes perdas, planejamento e coordenação complexa, feedback fraco e pouca aprendizagem. No LeSS Huge, por outro lado, divide-se em áreas de requisitos (ou de preocupações do cliente), refletindo a abordagem centrada ao cliente. Na *product backlog*, atribui-se uma e apenas uma área de requisitos para cada item.

No LeSS Huge, adiciona-se apenas o cargo de *product owner de área*, especializado em uma área de requisitos e foca-se na *backlog* para esta e vários times – nunca apenas um – colaboram para esta área. Cada área funcionará como uma implementação do Basic LeSS, trabalhando paralelamente em uma *Sprint* geral. O *product owner* geral e os de área formam o *Product Owner Team*. Acerca de eventos, este *framework* do LeSS possui os mesmos do Scrum básico, sendo os eventos executados para cada área de requisitos. A Fig. 5 mostra o *framework* do LeSS Huge.

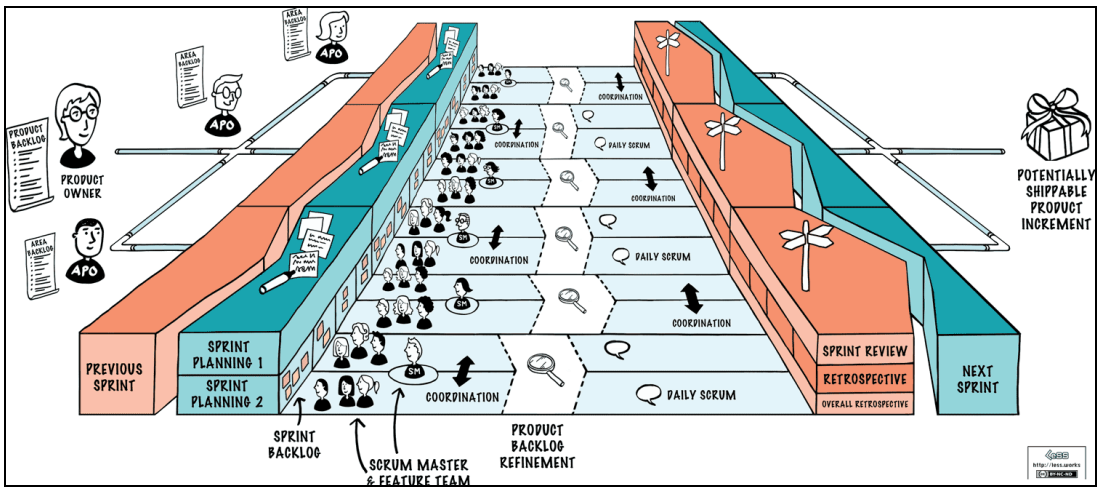

Figura 5: Visão geral do LeSS Huge (Less Company, 2014).

A estrutura organizacional e a formação dos times no LeSS caracterizam-se, mais uma vez, pela simplicidade. Não existe gerência de produto ou grupos de suporte (gerenciamento de configurações, suporte para integração contínua). Organizações de LeSS preferem expandir a responsabilidade dos times existentes. A estrutura organizacional de uma equipe LeSS é mostrada na Fig. 6.

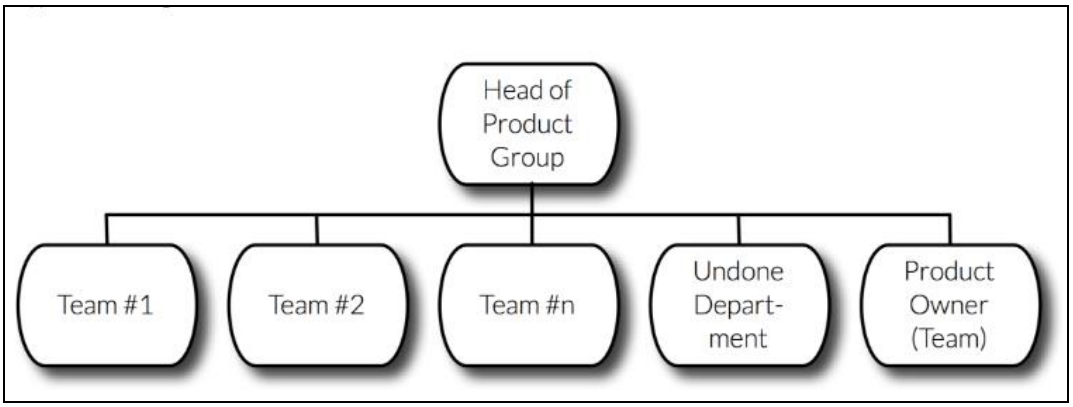

Figura 6: Estrutura de uma organização com LeSS (LeSS Company, 2014).

O *Head* basicamente caracteriza-se como um gerente hierárquico de todos os times. A sua função se concentra em remover obstáculos e melhorar os times. O departamento de *Undone* idealmente não deve existir, principalmente quando se trata de grupos menores que se utilizam do *framework* do LeSS. Este departamento relaciona-se com a incapacidade do grupo de criar incrementos potencialmente entregáveis em cada sprint, necessitando, assim, de suporte externo. Como exemplos, tem-se departamento de testes, QA, arquitetura, entre outros.

Em LeSS, organizam-se os colaboradores em times de funcionalidades. Um time de funcionalidades caracteriza-se por ser de longa duração, interfuncional e intercomponente, idealmente colocalizado, trabalhando em funcionalidades completamente centradas no cliente e compostas por "generalistas especialistas". Este difere de um time de componentes, cujas características se assemelham a uma abordagem mais tradicional.

Além disso, o LeSS possui a característica de dar grande autonomia para os times, o que também os leva a ter grandes responsabilidades. Esta característica se configura como uma maneira a descentralizar a tomada de decisão e, assim, diminuir a necessidade de cargos de gerenciamento dentro da organização, facilitando e simplificando a escalabilidade do framework.

Para finalizar, vale revelar que o LeSS real não começa com uma mudança no Scrum, mas sim na organização. O LeSS precisa ser simples. Existe a tendência de se adicionar cargos, artefatos e processos ao se escalar o desenvolvimento, o que deve ser evitado para manter a simplicidade, e a metodologia consegue isto. Trata-se do Scrum em escala, simplesmente, e não o Scrum como um bloco de construção. Isto significa que o LeSS busca como desenvolver o Scrum em grande escala, sem reinventar novos paradigmas.

4. Resultados e Discussão

A análise da adoção de métodos de escalabilidade ágil no mercado tem como base o 10th State of Agile Survey (VersionOne, 2016). Este relatório fornece as principais atualizações acerca das mudanças na adoção destes paradigmas em organizações. Nesta pesquisa, referente ao ano de 2015, focou-se bastante no tema da escalabilidade, atestando-se, mais uma vez, o valor deste tema na comunidade que pesquisa o ágil, atualmente.

Esta décima edição do relatório reforça, mais uma vez, os resultados dos trabalhos anteriores desta pesquisa (Melo, Cunha, Monteiro, 2015). Comparados ao relatório do ano anterior (VersionOne, 2015), os números mais elevados de empresas adeptas aos métodos ágeis, em conjunto com os números também maiores de empresas pouco experientes em práticas ágeis demonstram a entrada de novas empresas no mercado ágil, reafirmando o período de transição atual. Além disto, o Scrum continua a ser a metodologia ágil mais utilizada para projetos sem escalabilidade.

Ao se tratar de escalabilidade, como foco desta pesquisa, o relatório fornece dois resultados de destaque. Primeiramente, observa-se que cerca de 82% das empresas participantes possuem algum nível de distribuição dos seus times ágeis, contra apenas 35% de três anos antes. Isto mostra quão importante se tornou adotar maneiras de gerenciar times altamente distribuídos – um grande desafio para as metodologias tradicionais de agilidade.

O segundo ponto a se destacar – e que levará às comparações finais desta pesquisa – trata do uso de abordagens escaláveis nas empresas participantes. Apresentam-se os resultados na Fig. 7.

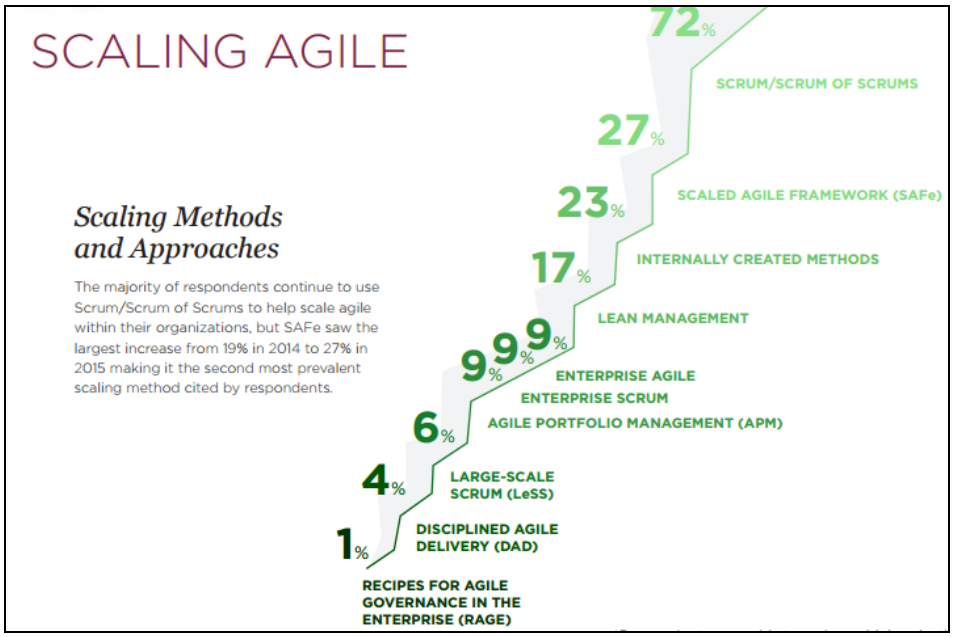

Figura 7: Adoção dos métodos de escalabilidade ágil no mercado (VersionOne, 2016).

O resultado desta pesquisa manteve as tendências do ano anterior. O Scrum (e a prática do Scrum of Scrums) permanecem liderando a abordagem de escalabilidade ágil, com um pequeno crescimento. Isto mostra que a maioria das empresas fazem uso da metodologia ágil tradicional para gerir grandes projetos – muito provavelmente devido a simplicidade e facilidade de aplicação deste frente aos novos métodos. Embora a prática do Scrum of Scrums, por sua

vez, apresente eficiência, não se trata de um processo de escalabilidade propriamente dito, mas sim uma forma de Scrum replicado para grandes projetos (Levison, 2008). Estudou-se o Scrum tradicional na pesquisa anterior (Melo, Cunha, Monteiro, 2015).

O maior crescimento, entretanto, resultou do SAFe, que agora ocupa a segunda posição de método escalável, de acordo com as empresas participantes. Baseado nas referências utilizadas nesta pesquisa, pode-se afirmar que o SAFe apresenta a estrutura mais consistente para uma eficiente mudança organizacional e cultural das organizações, mostrando-se a mais preparada para uma implementação bilateral e que apresenta abordagens para englobar todas as áreas de uma empresa, não apenas a relacionada ao desenvolvimento do produto. Além disto, esta metodologia melhor apontou os principais desafios de escalabilidade, bem como apresentou as abordagens mais completas para solucioná-las.

O terceiro lugar foi ocupado pela categoria de "métodos criados internamente". Este resultado revela que muitas empresas ainda gerenciam grandes projetos por meio de suas próprias convicções, o que pode ser bastante arriscado. Por outro lado, existem métodos desta categoria bem-sucedidos, como se pode exemplificar por meio do SA@S. Este método possui bastante destaque no tipo de empresa em que atua, e pode-se reconhecê-lo devido a sua organização estrutural inovadora e sua eficiência nos métodos de compartilhamento de conhecimento e de comunicação interna. Além disto, este paradigma destoa bastante dos outros três pesquisados, devido à natureza da empresa na qual se emprega: enquanto os outros se preocupam em desconstruir a organização tradicional com a agilidade, o SA@S foca em gerir a evolução de uma *startup* em crescimento acelerado. O SA@S não aparece na pesquisa por ser um método específico de uma empresa (Spotify), mas que vem crescendo em adoção em várias empresas, incluse no Brasil (Qulture Rocks, 2016).

A segunda metodologia escalável de maior crescimento nos resultados foi o LeSS, ultrapassando o DAD. Este crescimento tem como razão a simplicidade do método e no seu objetivo claro de não criar novas estruturas que possam tornar o Scrum tradicional complexo, mas sim torná-lo uma abordagem plausível e eficiente para grandes projetos e organizações. O problema, entretanto, concentra-se no fato de que o LeSS "sobrecarrega" seus times, ou seja, os delega diversas responsabilidades, a fim de não precisar criar estas novas estruturas. Isto, porém, torna sua adoção inicial mais complicada.

Por último, analisa-se o DAD. Este, percentualmente, ficou estático nas pesquisas. Em toda bibliografia analisada, este método se mostrou interessante por sua natureza adaptativa: não existe uma forma de implementar o DAD, mas sim diversas opções para realizar cada atividade. Contudo, isto a torna bastante complexa, extensa e detalhada, tornando sua adoção e ensino difícil. Além disto, esta metodologia mostrou-se ainda muito ligada a concepções tradicionais, principalmente em termos de arquitetura de software e composição de times.

Em linhas gerais, todos as metodologias estudadas herdam bastante dos métodos ágeis tradicionais, principalmente quando se trata de cargos, artefatos e eventos. Além disto, englobou-se as práticas técnicas em todos os casos – o que mostra que as boas técnicas de Engenharia de Software provenientes do XP têm grande reconhecimento e eficiência na gestão de projetos desta natureza. Vale destacar que, nesta pesquisa, destacou-se os métodos citados tanto pelo fato de que estes apresentaram uma tendência de crescimento nas pesquisas em relação ao ano anterior (no caso do SAFe e LeSS), quanto pela disponibilidade de material de pesquisa.

Por fim, em anexo, este trabalho apresenta uma tabela comparativa que relaciona os principais desafios de escalabilidade ágil (destacados anteriormente na Contextualização) com a abordagem de cada uma das metodologias estudadas. Dividiu-se a tabela em quatro problemas gerais de escalabilidade dos métodos ágeis. Em cada uma destas, enumerou-se os desafios inerentes a cada problema. Em cada desafio, descreveu-se a solução de cada um dos métodos estudados. Vale observar que a escolha da solução depende do contexto ao qual os times estão inseridos e quais seus principais problemas. Em linhas gerais, deve-se escolher de acordo com as principais vantagens e desvantagens de cada metodologia, explicados anteriormente.

5. Conclusão

A escalabilidade de metodologias ágeis tem possibilitado com que grandes empresas e *startups*, em fase de crescimento, possam adequar seus processos aos padrões da cultura *lean*-ágil, quer no desenvolvimento do produto, quer no gerenciamento em nível executivo.

Esta pesquisa focou na prospecção e análise de metodologias para escalabilidade do ágil em grandes organizações, a fim de compará-las e destacar as principais características e problemas de cada uma, provendo conhecimento a respeito dos principais empecilhos que estas enfrentam e quais das suas práticas existentes se encontram em busca de criar agilidade, no contexto de grandes organizações. Acredita-se que, em alguns anos, os métodos pesquisados cresçam e liderem a lista de métodos de escalabilidade apresentada anteriormente.

A partir das referências utilizadas como base de pesquisa, pode-se concluir que a escalabilidade ágil se tornou, nos últimos anos, um tema de grande discussão, devido aos benefícios que a agilidade leva para os pequenos times. Contudo, a adoção destes paradigmas de gestão em grandes empresas possui grandes desafios, de maneira que se propõem diversas abordagens para melhor solucioná-los.

O surgimento de diversos paradigmas de escalabilidade expõe a real necessidade da agilidade em grandes corporações e revela que este problema possui não apenas diversas abordagens como também se apresenta em aberto para novos modelos de gestão pautados no ágil.

6. Agradecimentos

Agradeço às pessoas ligadas diretamente a mim, pelo apoio durante este período do projeto. Ao Prof. Dr. Adilson Marques da Cunha e ao M.C. Cassiano Monteiro, pela orientação durante o trabalho. Foram essas pessoas que me deram a oportunidade e o direcionamento necessários para que eu pudesse aprender sobre Metodologias Ágeis, sua cultura e principais práticas. Agradeço também ao CNPq que, vinculado ao Ministério da Ciência e Tecnologia (MCT), apoia a pesquisa brasileira e contribui diretamente para a formação de jovens pesquisadores, investindo e promovendo o aumento da produção de conhecimento e gerando novas oportunidades para universitários desejosos em iniciar uma vida de pesquisa e desenvolvimento nas diversas áreas do conhecimento.

Agradeço, por fim, àquelas pessoas que fazem o mundo se superar a cada dia.

7. Referências

9th Anual State of Agile Survey. 2014, VersionOne, Inc. All Rights Reserved.

10th Anual State of Agile Survey. 2015, VersionOne, Inc. All Rights Reserved.

Ambler, S., Lines, M. “Disciplined Agile Delivery: A Practitioner’s Guide to Agile Software Delivery in the Enterprise”. IBM Press, 2012.Agile Manifesto. Disponível em http://www.agilemanifesto.org. Acesso em 26 de agosto de 2015.

Formulário de Inscrição do bolsista PIBIC 2015/2016.

Hastie, S. “Compare Agile Scaling”. Disponível em: www.infoq.com/news/2014/07/compare-agile-scaling. Acesso em 25/02/2016.

Kniberg, H., Ivarsson, Anders. “Scaling Agile @ Spotify with Tribes, Squads, Chapters & Guilds”, 2012.

Larman, C. Vodde, B. “Large-Scale Scrum: More with LeSS”. Addison-Wesley Professional. 1st Edition, 2016.

Lean Enterprise Institute. “What is Lean?”. Disponível em: http://www.lean.org/WhatsLean. Acesso em 03/03/2016.

Leffingwell, D. “Agile Software Requirements”. Reading, MA: Addison-Wesley Professional, 2011.

Leffingwell, D. “Scaling Software Agility: Best Practices for Large Enterprises”. Reading, MA: Addison-Wesley Professional, 2007.

Leffingwell, D. Scaled Agile Framework. Disponível em: www.scaledagileframework.com. Acesso em 25/02/2016.

Levison, M. “Scrum of Scrums – Problemas e Valores”. Disponível em: https://www.infoq.com/br/news/2008/12/scrum-of-scrums. Acesso em 06/10/2016.

Melo, L. C., Cunha, A. M., Monteiro, C. B. A. L. “Uma Análise da Utilização de Metodologias Ágeis Envolvendo o Desenvolvimento de Software Safety-Critical”: XXI Encontro de Iniciação Científica e Pós-Graduação do ITA – XXI ENCITA, 2015.

Paasivara, M., Lassenius, C. “Scaling Scrum in a Large Distributed Project”: International Symposium on Empirical Software Engineering and Measurement, 2011.

Qulture Rocks. “Como a Spotify organiza seus times de produto”. Acesso em: 06/10/2016.

Schiffer, B.: “Comparing Ways to Scale Agile”. Agile Product and Product Managers. Melbourne, 28/10/2014.

Simons, M.: Distributed Agile Development and the Death of Distance. Sourcing and Vendor Relationships 5 (4) Cutter Consortium, 2006.

Spotify Labs. Disponível em: labs.spotify.com. Acesso em 25/02/2016.

Spotify Labs. “Spotify Engineering Culture – Part 1”. Retirado de labs.spotify.com/2014/03/27/spotify-engineering-culture-part-1. Acesso em 25/02/2016.

The LeSS Company B.V. “Large-Scale Scrum: More with LeSS”. Retirado de http://less.works. Acesso em 18/07/2016.

8. Anexo: Comparação de Métodos de Escalabilidade

Neste anexo, apresentam-se tabelas comparativas dos métodos de escalabilidade estudados, acerca dos principais desafios encontrados para aplicação das metodologias ágeis tradicionais em organizações e projetos de alta complexidade. Vale destacar que os espaços deixados em branco significam que, a partir das referências utilizadas nesta pesquisa, não foi encontrada nenhuma abordagem relevante para o desafio em questão.

Tabela 1: Comparação acerca dos principais desafios relacionados a formação dos times.

Desafio	SAFe	SA@S	DAD	LeSS
Estrutura de times	Estrutura análoga ao Scrum com equipes de suporte na interface.	Estrutura matricial ponderada em direção à entrega.	Formação de cargos primários e secundários que englobam as responsabilidades de cargos tradicionais.	Estrutura análoga ao Scrum, incluindo o cargo de product owner de área.
Integração de times	Criação de times de coordenação (nível de programa).	Formação de "tribos" que trabalham em áreas relacionadas.	Criação de um time de liderança formado pelos team leaders, product owners e architecture owners, para coordenação técnica, do projeto e dos requisitos.	Ocorre por intermédio do Product Owner Team formado pelos product owners de área e do Scrum of Scrums.
Gerenciamento de times distribuídos	Criação de cargos de interface para times remotos; rotação de líders (Shuttle Advocacy), criação de uma infraestrutura comunicacional eficiente [9].		Forte investimento na fase de Incepção, com definição de linhas de ação e conveções claras; maximização da independência de times e nomeação de "embaixadores"	

Tabela 2: Comparação acerca dos principais desafios relacionados a adoção da agilidade.

Desafio	SAFe	SA@S	DAD	LeSS
Sincronismo e domínio das iterações	Adoção do "Agile Release Train" (ART), reuniões de sincronização e Scrum of Scrums.	Reuniões de sincronização diárias e Scrum of Scrums; Adoção de Release Trains e Feature Toggles.	Reuniões de sincronização.	Embora tenha como princípio que só exista um Sprint para todas as equipes, não se encontrou métodos de sincronismo entre equipes.
Processos ágeis (artefatos e eventos)	Herança do Scrum para times de desenvolvimento. Criação de processos para gerenciamento executivo e de times de marketing e vendas.	Herda diretamente do Scrum.	Híbrido de diversas metodologias ágeis, adaptando-se a quem adota.	Processos análogos ao Scrum tradicional, com pequenas mudanças no Sprint planning, Sprint review/retrospective e PBR.

Tabela 3: Comparação acerca dos principais desafios relacionados a complexidade técnica dos projetos.

Desafio	SAFe	SA@S	DAD	LeSS
Arquitetura de Software	Gera uma "trilha de arquitetura", para construí-la e fazer prototipagens. Demanda-se, também, times focados em arquitetura e infraestrutura de componentes.	A adota o cargo de System Owner, que se preocupa com questões técnicas e de arquitetura para cada sistema. Adicionalmente, existe o arquiteto chefe, que coordena o trabalho em alto nível entre os múltiplos sistemas do projeto.	No DAD, toda peça da arquitetura deve ser pensada e provada antes do desenvolvimento da solução. As referências utilizadas fornecem diversos detalhes a respeito da modelagem da arquitetura do software e como abordá-la.	Logo, encoraja um design emergente por uma cultura de desenvolvimento de "jardinagem", com pequenos e rápidos ciclos de feedback, ao invés de "arquitetar"; desenvolvimento de técnicas de design flexível: TDD, refatoração.
Compartilhamento de conhecimento	Código coletivo, revisões de código; acredita-se que a comunicação ágil interna já possibilita isso. Para casos de times distribuídos, adota-se o Shuttle Advocacy.	Criação de capitanias e guildas para compartilhar conhecimento em áreas de interesse, desafios técnicos, ferramentas, práticas, entre outros.	Código coletivo, revisões de código; especialistas envolvidos exercem papel de coach.	Adoção de Comunidades de Prática: Grupos formados por pessoas que trabalham nos mesmos componentes ao mesmo e tempo e portanto precisam saber um do outro para fazer perguntas e revisar o código.

Tabela 4: Comparação acerca dos principais desafios relacionados a própria organização empresarial.

Desafio	**SAFe**	**SA@S**	**DAD**	**LeSS**
Disciplina empresarial e cultura corporativa	Criação de um gerenciamento executivo "top-down" para quebrar empecilhos de agilidade; adoção de modelos para equipes fora do time de desenvolvimento.		Estratégias colaborativas que focam no direcionamento ágil, não apenas no time de desenvolvimento, como também em equipes de suporte.	Embora afirme que a implementação de uma abordagem ágil deva acontecer em toda organização de forma bilateral, não se encontrou uma abordagem concreta.
Gerenciamento executivo	Criação de métricas de progresso e consequente avaliação quantitativa da organização; adoção de canais de comunicação e coaching que acelerem a implementação de agilidade.		Criação de estratégias de direcionamento ágil a partir das filosofias do DAD e que focam em seus milestones; métricas de progresso e consequente avaliação quantitativa da organização.	