PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 03/16

Técnicas para Aplicação de Agilidade em Arquitetura de Software

Diogo Silveira Mendonça
Arndt von Staa

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 03/16 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena Dezembro, 2016

Técnicas para Aplicação de Agilidade em Arquitetura de Software *

Diogo Silveira Mendonça [1, 2] Arndt von Staa [1]

[1] Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
[2] Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET-RJ)

dmendonca@inf.puc-rio.br, arndt@inf.puc-rio.br

Abstract. Software architecture is an phase that consumes significant time and effort, furthermore its return on investment is achieved only in long term. Hence, this phase is commonly neglected, creating architectural problems. One approach to deal with this situation is applying agile architecture techniques. However, this combination is still an active research area, and is not a consolidated industry practice. Intending to help software engineering practitioners to combine agile and architecture we present some preeminent techniques to create, document and evaluate architecture. We also present a brief review of the problems and the research in this area. Finally, we made a critical analysis of these techniques against the architectural problem factors, identifying which factors are treated by each technique, and also its limitations and possible implementation difficulties.

Keywords: Software Architecture, Agile, Documentation, Evaluation, Architectural Problems

Resumo. Arquitetar software exige significativo esforço e tempo nos projetos de software, além disso seu retorno sobre o investimento só é obtido a longo prazo. Em virtude disto esta fase é muitas vezes negligenciada, levando a problemas arquiteturais que têm um impacto significativo sobre a qualidade do software e sobre seu custo de desenvolvimento e manutenção. Uma abordagem para lidar com esta questão é agilizar a confecção da arquitetura. Apesar desta combinação ser uma área de pesquisa ativa, não é uma prática consolidada na indústria. Com o intuito de ajudar o praticante da engenharia de software a confeccionar arquiteturas com agilidade, apresentamos algumas técnicas ágeis promissoras para a criação, a documentação e a avaliação de arquiteturas de software. Apresentamos também uma revisão curta dos problemas e pesquisas na área. Por fim, fazemos uma avaliação crítica destas técnicas em relação aos fatores que levam a problemas arquiteturais, identificando quais fatores cada técnica trata e suas possíveis limitações e dificuldades.

Palavras-chave: Arquitetura de Software, Agilidade, Documentação, Avaliação, Problemas Arquiteturais

---

* Trabalho patrocinado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq – através da bolsa 141345/2015-2.

Responsável por publicações:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3114-1516 Fax: +55 21 3114-1530
E-mail: bib-di@inf.puc-rio.br

Sumário

1 Introdução

A atividade de arquitetar software é frequentemente negligenciada ao desenvolver e manter software. Um dos motivos para isto ocorrer é que suas atividades demandam significativos tempo e custo no início do projeto do software, sendo que o retorno sobre o investimento delas só é percebido a longo prazo (FOOTE; YODER, 1997). Assim, geralmente devido a pressões de cronograma, são priorizadas atividades de desenvolvimento em detrimento à elaboração da arquitetura do software, seguindo o caminho de menor esforço para lidar com arquitetura. Este caminho leva ao padrão arquitetural conhecido como "*The big ball of mud*" (FOOTE; YODER, 1997), no qual o software é formado por uma selva de código mal estruturado e consequentemente sujeito a diversos problemas de manutenção, como dificuldade de realizar alterações no software sem inserção de novos defeitos.

Além do problema da deterioração da estrutura interna do código, tem-se ainda o esforço despendido em retrabalho inútil. Este corresponde ao esforço que poderia ser deixado de realizar caso se trabalhasse corretamente desde o início. É altamente provável que a ausência de uma especificação suficientemente abrangente e a correspondente arquitetura conduza a inúmeras alterações que acabam sendo registradas como se fossem mudanças de rumo solicitadas pelo cliente. Entretanto, muitas dessas mudanças de rumo são reflexo de uma especificação de requisitos incompleta ou mesmo incorreta. Muitos desses problemas podem ser resolvidos ao arquitetar uma solução, pois essa atividade tem como subproduto o questionamento da especificação de requisitos quanto à sua corretude e completeza, sem induzir excessivo detalhamento da especificação. Infelizmente, não foram encontrados artigos confiáveis tratando de medições relativas às causas das solicitações de alteração e quantas delas poderiam ter sido respondidas ao arquitetar um sistema.

A solução para o problema da negligência em arquitetura requer maturidade e disciplina por parte dos arquitetos do software. Agilizar a confecção da arquitetura é uma ideia promissora para ajudar a reduzir os custos iniciais do projeto, bem como a redução do custo despendido em retrabalho inútil. Apesar da agilidade em atividades de arquitetura de software ser ativamente pesquisada desde 2003 (YANG; LIANG; AVGERIOU, 2016), ainda não é uma prática consolidada na indústria. Acreditamos que um dos fatores que leva a esta situação é a falta de conhecimento a respeito do que é pesquisado e experimentado na área. Com o objetivo de diminuir este desalinhamento entre indústria e academia apresentamos neste trabalho um resumo do mapeamento de literatura apresentado por Yang, Liang e Avgeriou (2016), e quatro trabalhos com viés prático e que tratam da confecção ágil da arquitetura.

O mapeamento sistemático de literatura realizado por (YANG; LIANG; AVGERIOU, 2016) mostrou que as atividades mais pesquisadas sobre o uso de agilidade em conjunto com arquitetura são as de documentação e avaliação da arquitetura. Isto pode ser um indício que estas duas atividades são as mais relevantes para pesquisadores e praticantes. Em qualquer um dos casos merecem ser exploradas mais a fundo, e por este motivo escolhemos trabalhos com viés prático nestas duas áreas. Os trabalhos selecionados sobre a atividade de documentação foram o de Hadar et al. (2013) que elabora um modelo de documentação simplificada da arquitetura para projetos que utilizam metodologias tradicionais e o de Jensen, Platz e Tjørnehøj (2008) que mostra uma proposta de documentação e trabalho de arquitetura integrada ao modelo de processo *extreme programming* (XP). Os trabalhos que selecionamos que tratam da ava-

liação de arquiteturas foram o de Eloranta e Koskimies (2012) que introduz quatro modelos em que a arquitetura é integrada com o Scrum utilizando avaliações baseadas em decisões arquiteturais, e Nord, Brown e Ozkaya (2011) que apresentam um processo de elaboração de arquiteturas utilizando prototipação como mecanismo de avaliação. Finalmente como contribuição realizamos uma avaliação crítica destas técnicas analisando suas forças, fraquezas e possíveis ajustes para adoção pela indústria.

O resto deste trabalho está organizado com segue. Na seção 2 apresentamos o problema de negligência em arquitetura e o padrão no qual ela resulta. Na seção 3 mostramos resumidamente o estado da arte na pesquisa sobre o uso de agilidade em conjunto com arquitetura. Na seção 4 apresentamos os trabalhos relacionados a documentação de arquitetura com agilidade. Na seção 5 mostramos os trabalhos relacionados a avaliação de arquitetura com agilidade. Na seção 6 discutimos os pontos fortes e fracos das metodologias e apresentamos as dificuldades e limitações para implementá-las na indústria. Na Seção 7 concluímos e apresentamos trabalhos futuros.

2 Fatores de Influência em Arquitetura de Software

Foote e Yoder (1997) descreveram o padrão *"The big ball of mud"*, que segundo suas próprias palavras é uma selva de código espaguete, mal estruturada, espalhada, desleixada, amarrada com arame e colada com fita adesiva. Sem a compreensão dos motivos que levaram a este padrão não podemos agir para evitá-lo. Assim apresentamos nesta seção um resumo do trabalho de Foote e Yoder (1997) focando nas forças que influenciam um projeto de software e que podem levá-lo a se tornar uma grande bola de lama, entre elas estão tempo, custo, experiência, habilidade, visibilidade, complexidade, mudanças, escala e fatores organizacionais.

O tempo é sempre uma restrição em projetos de software, seja para limitar o custo ou por motivos competitivos de *time-to-maket* de produtos. Gerentes de projeto estão sempre procurando reduzir esses fatores. Projetar uma boa arquitetura de software consome tempo e seus benefícios são vistos somente a longo prazo, neste cenário as atividades de arquitetar são candidatas a serem encurtadas ou cortadas por não trazerem um benefício imediato para o projeto. Além disto, projetos em fase inicial andam mais rápido caso existam poucas restrições de arquitetura, podendo a presença de uma arquitetura detalhada e, possivelmente, restritiva ser compreendida como um fator negativo em relação a tempo e custo. Por fim uma arquitetura preliminar, feita às pressas, pode ser pior do que não ter nenhuma arquitetura, pois esta engessa o projeto desencorajando a experimentação em busca de soluções novas e melhores.

Despender muito tempo e esforço com a definição da arquitetura e não entregar o projeto não é uma solução aceitável, por outro lado a adequação da arquitetura ao projeto é fundamental para que os custos de manutenção não sejam excessivos. A experiência das pessoas que elaboram a arquitetura é um fator de grande influência para que este requisito seja atendido. Esta experiência pode ser dividida em três tipos, experiência no mercado de trabalho, no domínio do problema e na atividade de arquitetar software. Intuitivamente profissionais com pouca ou nenhuma experiência de desenvolvimento não são as pessoas mais indicadas para projetar arquiteturas, pois esta é uma atividade de alta complexidade e que exige conhecimento de como combinar diversos requisitos em uma solução técnica, o que somente pode ser aprendido na prática. Porém, mesmo profissionais com perfil sênior podem ter dificuldade em projetar arquiteturas para domínios de problemas nos quais não tenham experiência prévia e, mesmo

que tenham, se não forem experientes em projetar arquiteturas terão dificuldade com a tarefa.

A experiência dos arquitetos tem grande influência na elaboração da arquitetura, contudo após ela estar pronta é entregue as mãos dos desenvolvedores. A menos que os artefatos desenvolvidos sejam monitorados de perto, o que não acontece na maioria das vezes, a implementação será criada de maneira modificada por eles. Programadores têm diferentes competências, predisposições e temperamentos. Enquanto alguns são apaixonados por encontrar boas abstrações e organizar seu código, outros são habilidosos para navegar nos pântanos de código desorganizado e complexo, deixando as boas abstrações de lado. Além disto, a arquitetura de software não é visível pelo usuário final e muitas vezes nem pelos patrocinadores do projeto, provendo um ambiente perfeito para que não haja preocupações com a boa organização do software. Na maioria dos casos somente os desenvolvedores veem a arquitetura implementada, ficando a cargo deles zelar pela qualidade dela.

Muitas vezes os programadores são competentes, mas o problema a resolver é complexo e, por este motivo, inevitavelmente a arquitetura também o será. Aliando isto com as inevitáveis e frequentes solicitações de mudanças durante o desenvolvimento e manutenção do software (BECK, 2000) a arquitetura pode se deteriorar. Caso a arquitetura inicial não se adeque bem às necessidades do domínio, ela não comportará alterações, sendo mais suscetível à deterioração. Alterações também costumam fazer o software crescer em tamanho, e arquiteturas que já não são boas para software pequeno se tornam piores quando esses crescem. Isto ocorre não somente pelo aumento da complexidade mas também pela diferença que existe na gestão de projetos pequenos e grandes, como por exemplo na comunicação que é facilitada em projetos pequenos e em grandes precisa de mecanismos mais formais tais como documentos cuidadosamente elaborados. Se estes mecanismos não forem adaptados para as necessidades de projetos maiores, a arquitetura tem grande chance de ser afetada.

Por fim, fatores organizacionais da empresa desenvolvedora podem influenciar o tratamento da arquitetura. O modo de fazer software da organização faz parte da cultura técnica da organização. Independentemente do grau de maturidade esta cultura estabelece um processo de desenvolvimento padrão, frequentemente de conhecimento tácito. Caso o processo padrão não incentive a criação de arquiteturas, ele será um bom caminho para projetos se tornarem bolas de lama. Além disto, mesmo que o padrão incentive a arquitetura, se for único a ser seguido de forma inflexível em todos os projetos, desestimulará a experimentação de novas soluções, o que pode ser um caminho para problemas arquiteturais.

3 Estado da Arte de Arquitetura com Agilidade

Yang, Liang e Avgeriou (2016) realizaram um mapeamento sistemático de literatura da combinação de agilidade e arquitetura. Para isto os autores buscaram na literatura artigos sobre o uso de práticas ágeis em projetos desenvolvidos com métodos centrados na arquitetura ou que utilizavam arquitetura em projetos desenvolvidos com metodologias ágeis. Foram identificados 54 artigos, sendo estes classificados segundo as atividades gerais e específicas de arquitetura e também segundo os métodos ágeis utilizados. As tabelas 1 e 2 mostram respectivamente as atividades específicas e gerais de arquitetura juntamente com seu significado. A figura 1 mostra o mapeamento dos artigos, onde os números dentro dos círculos representam o identificador de cada artigo.

Tabela 1 – Atividades Específicas de Arquitetura

Atividade	Objetivo
Análise Arquitetural (AA)	Definir os problemas que uma arquitetura precisa resolver. Tem como resultado um conjunto de requisitos relevantes para a arquitetura.
Síntese Arquitetural (AS)	Propor soluções arquiteturais candidatas para atender os requisitos arquiteturais.
Avaliação Arquitetural (AE)	Garantir que as decisões arquiteturais de design realizadas são apropriadas, e medir as arquiteturas propostas contra os requisitos arquiteturais.
Implementação Arquitetural (AI)	Criar o design detalhado realizando a arquitetura.
Manutenção e Evolução Arquitetural (AME)	Corrigir falhas ou implementar novos requisitos na arquitetura.

Tabela 2 – Atividades Gerais de Arquitetura

Atividade	Objetivo
Recuperação Arquitetural (AR)	Extrair a arquitetura atual da implementação do sistema.
Descrição Arquitetural (ADp)	Descrever a arquitetura em um conjunto de elementos arquiteturais (visões).
Compreensão Arquitetural (AU)	Entender os elementos arquiteturais (decisões arquiteturais) e seus relacionamentos no design da arquitetura.
Análise de Impacto Arquitetural (AIA)	Identificar os elementos arquiteturais que são afetados por um cenário de mudança.
Reuso Arquitetural (ARu)	Reutilizar elementos de design de uma arquitetura existente em uma nova.
Refatoração Arquitetural (ARf)	Melhorar a estrutura arquitetural de um sistema sem mudar o seu comportamento externo.

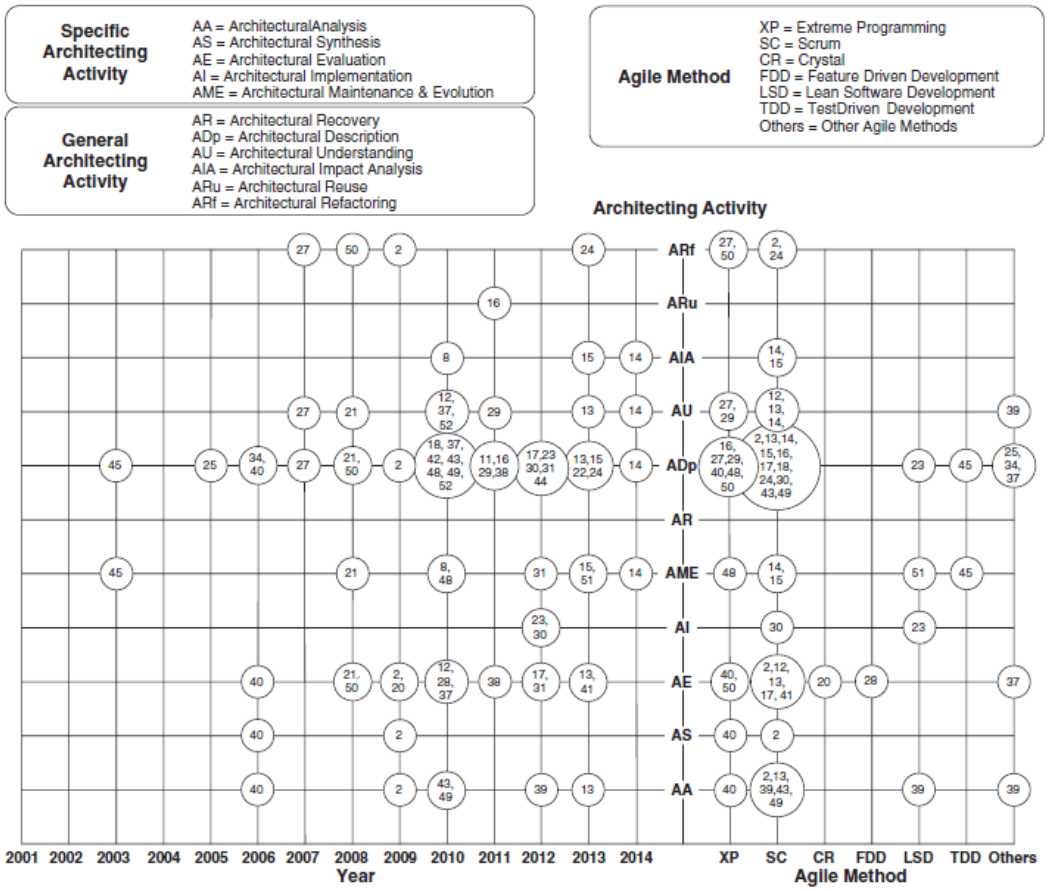

Figura 1 – Mapeamento dos artigos sobre a combinação de agilidade e arquitetura

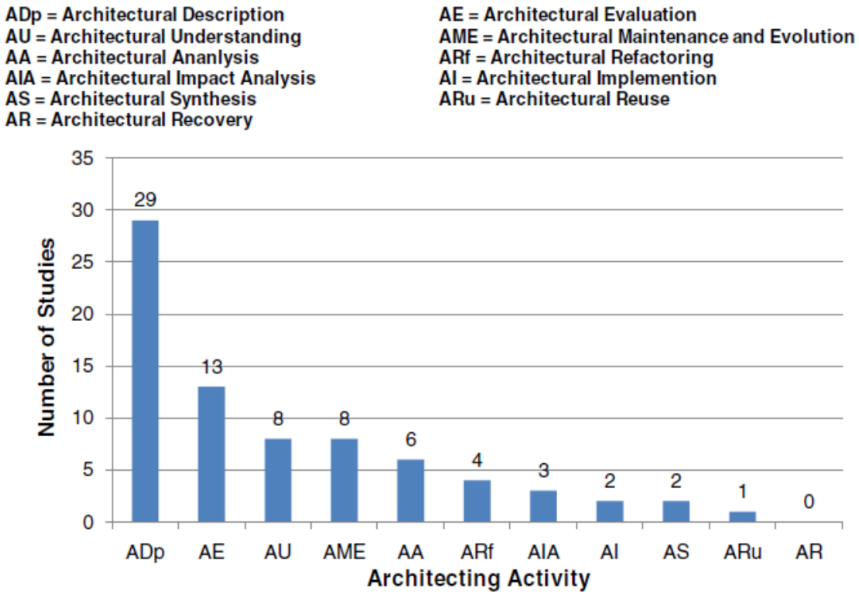

Figura 2 – Quantidade de artigos publicados por atividade de arquitetura

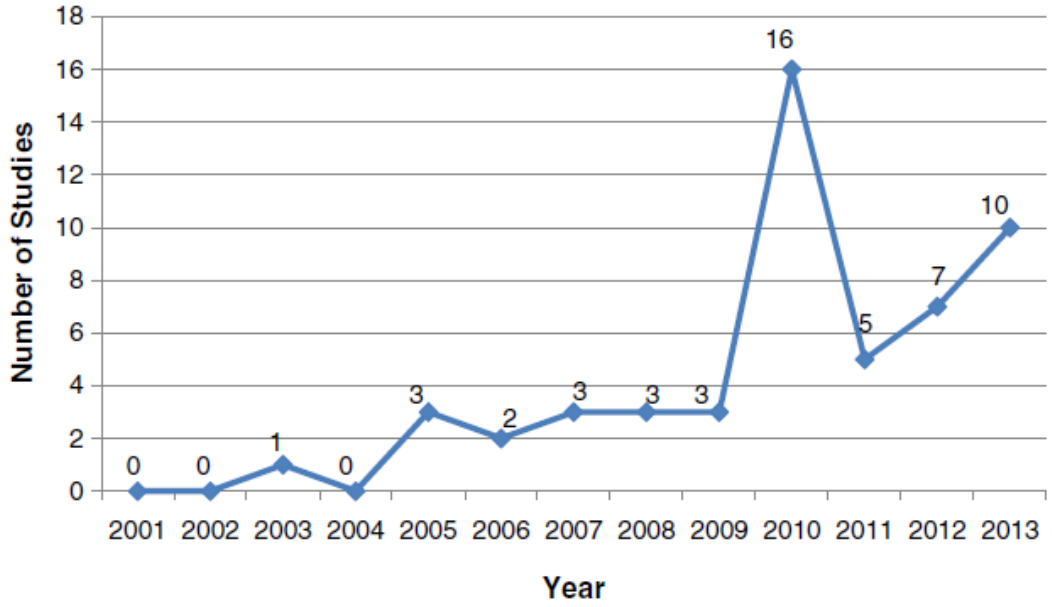

Figura 3 – Quantidade de artigos publicados por ano

Na figura 1 e 2 podemos observar que a atividade que concentra maior quantidade de artigos é a descrição de arquitetura, e o método ágil mais pesquisado é o Scrum. Entretanto, é preciso ressaltar que Scrum não é um processo de desenvolvimento, mas, sim, um processo de gestão do desenvolvimento. Podemos observar também que a combinação da descrição de arquitetura com Scrum apresenta grande concentração de estudos e algumas atividades como recuperação e reuso arquitetura foram muito pouco pesquisadas em combinação com agilidade. As tendências apresentadas não são surpreendentes uma vez que o Scrum é o método de gestão de projetos ágeis mais utilizado, não se preocupando com o desenho do software a ser desenvolvido. A estatística também é coerente com o fato que a descrição de arquitetura sempre foi colocada em segundo plano pelos métodos ágeis.

A figura 2 mostra a quantidade de estudos por atividade de arquitetura, nela observamos que após a descrição e avaliação de arquitetura as atividades mais pesquisadas são as de compreensão, manutenção e evolução. A figura 3 mostra a quantidade de artigos publicados por ano, nela observamos um crescimento da quantidade de estudos tendo seu pico em 2010, quando na conferência XP o tema agilidade e arquitetura figurou na segunda posição do top 10 das questões de pesquisa mais quentes na comunidade ágil (FREUDENBERG; SHARP, 2010). Após este ano houve uma queda considerável e uma retomada do crescimento nos três anos seguintes.

A revisão sistemática pesquisou como arquitetar em desenvolvimento ágil identificando 43 práticas no total, existindo pouca convergência entre elas nos artigos. As práticas reportadas com maior frequência foram: abordagem iterativa para criação da arquitetura, o uso de arquitetura de referência, análise de impacto de mudança, métodos de avaliação de arquitetura iterativos e incrementais, conectar a arquitetura com o código-fonte e elaboração preliminar de arquitetura (iteração zero). Na seção 5 apresentamos em mais detalhes abordagens iterativas e incrementais para elaboração e avaliação de arquitetura em projetos ágeis com Scrum.

Da mesma forma a revisão sistemática apresentou as práticas de desenvolvimento ágil que foram aplicadas com arquitetura em projetos centrados na arquitetura, sendo identificadas 41 práticas, e, havendo uma maior convergência entre elas nos artigos. As principais práticas identificadas foram: *backlog*, desenvolvimento iterativo e incremen-

tal, realizar somente o trabalho suficiente, *sprints*, arquitetura ágil e integração contínua. Na seção 4 mostramos como é possível elaborar um documento de arquitetura com somente a informação suficiente, e, na seção 5, apresentamos uma abordagem baseada em prototipagem para elaboração e avaliação iterativa da arquitetura.

Os benefícios da utilização de arquitetura com agilidade foram divididos em duas classes, os benefícios gerais que não dependem de uma abordagem de arquitetura específica, método ágil ou ferramenta aplicada, e os benefícios específicos que podem depender destes. Os benefícios gerais mais citados nos artigos foram a melhoria dos atributos de qualidade do sistema, ajudando a tratá-los de uma maneira sistemática e antecipada, e a maior facilidade de evolução contínua da arquitetura, provendo flexibilidade a ela. Já os benefícios específicos mais citados foram a facilitação do design de arquitetura, melhorias na documentação, qualidade da arquitetura, qualidade do sistema, e nos processos de decisão e avaliação da arquitetura. Por outro lado os custos envolvidos na aplicação de arquitetura com agilidade precisam ser melhor estudados.

Para obter os benefícios da utilização de arquitetura com agilidade alguns desafios são enfrentados. Um dos principais desafios é o conflito de paradigmas entre arquitetura, que estabelece planos a serem seguidos, e agilidade, que trata de flexibilidade e mudanças contínuas. Isto leva ao questionamento de o quanto de arquitetura é necessário para construir um sistema com qualidade e flexibilidade. Outro desafio que merece destaque é o conflito entre documentação extensiva dos métodos tradicionais de arquitetar contra pouca ou nenhuma nos métodos ágeis. Outros desafios no desenvolvimento ágil são a ausência de um papel definido para arquitetura, a não utilização de métodos de avaliação da mesma e a evaporação do conhecimento arquitetural, visto que ele só é transferido tacitamente. Por fim, outra dificuldade é a ausência de técnicas bem definidas com ferramentas de suporte para aplicação da combinação de arquitetura com agilidade.

O engenheiro de software consciente dos desafios da combinação de arquitetura com agilidade deve ter atenção aos fatores de sucesso para a sua aplicação. A figura 4 mostra a quantidade de artigos que citam cada fator de sucesso. Temos como destaques os seguintes fatores: a comunicação da arquitetura para os envolvidos no projeto, principalmente quando se está utilizando desenvolvimento incremental; a seleção apropriada do esforço para elaboração da documentação da arquitetura, juntamente com a sua complexidade, dimensão e a consciência de sua importância; a clara definição de quem comunica e para quem as decisões arquiteturais, juntamente com a tamanho e experiência das equipes; e por fim o papel do arquiteto juntamente com suas responsabilidade, conhecimentos e experiência são fundamentais para o sucesso da aplicação da combinação de arquitetura com agilidade.

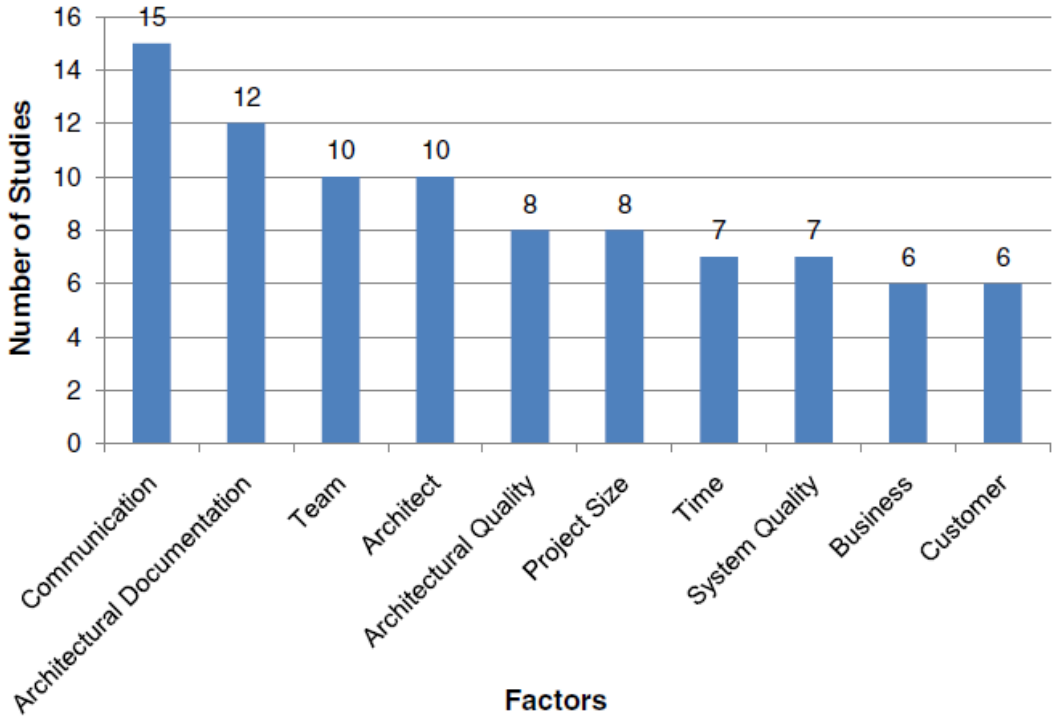

Figura 4 – Número de estudos que citam cada fator de sucesso

As ferramentas amplamente disponíveis identificadas no estudo para suportar a combinação de agilidade com arquitetura foram o quadro branco, *wiki*, planilha eletrônica, e *flip chart*. Cada uma dela serviu para propósitos diferentes nos artigos, o quadro banco e o *flip chart* ajudaram a armazenar e comunicar informação arquitetural em projetos ágeis, já o *wiki* foi utilizado para documentação e a planilha eletrônica para registrar e mapear as preocupações arquiteturais em relação a fase do projeto que seriam abordadas. Também foram identificadas no estudo algumas ferramentas que não estão disponíveis para o uso livre, como arquivos XML para armazenamento de informações arquiteturais, sistemas de reputação que avaliam a contribuição de forma colaborativa dos desenvolvedores para a documentação da arquitetura e uma ferramenta específica chamada de ferramenta de especificação abstrata que apoia a elaboração de um documento de arquitetura integrado a uma ferramenta de modelagem. Abordaremos melhor esta última ferramenta na seção 4.

As lições aprendidas mais citadas nos artigos revisados falam sobre o projeto, decisões, documentação de arquitetura, o arquiteto, o time de desenvolvimento e a abordagem de combinação de arquitetura e agilidade. Tais lições enunciam que é melhor focar no projeto de arquitetura o quanto antes, preferencialmente em iterações iniciais. Além disto as partes interessadas devem decidir quando congelar a arquitetura para que as entregas sejam realizadas, e também devem estar conscientes da quantidade de débito técnico acumulado. As decisões arquiteturais importantes devem ser tomadas de forma antecipada e documentadas explicitamente antes da implementação. Tanto a documentação quanto a modelagem arquitetural funcionam melhor se forem feitas de forma iterativa e incremental com o cliente envolvido, tendo como meta a qualidade da arquitetura, que deve ser simples, vital, transparente e flexível. Além disto, o processo de desenvolvimento deve incluir técnicas leves e de integração contínua, pois estas reduzem a complexidade e riscos consideravelmente. O arquiteto precisa ter o seu papel e responsabilidades claramente definidos no desenvolvimento ágil, sendo parte do time de desenvolvimento e envolvendo este na comunicação e no processo de

tomada de decisões arquiteturais. Por fim, é necessário considerar e comunicar o potencial conflito entre as abordagens de agilidade e arquitetura, pois a equipe de desenvolvimento pode ter dificuldade caso não tenha orientação prévia.

Apesar da combinação de agilidade e arquitetura ser ativamente pesquisada existem muitas questões em aberto, representando oportunidades de pesquisa. Entre elas podemos destacar que nenhum estudo foi encontrado sobre a recuperação de arquitetura em projetos de desenvolvimento ágil. Além disto, existe uma falta de orientação para a utilização de práticas ágeis com arquitetura e ausência de ferramentas dedicadas para suportar tal combinação. Adicionalmente, a literatura não apresenta orientações de como utilizar os fatores de sucesso identificados para facilitar tal combinação, e por fim os desafios apresentados anteriormente nesta seção representam potenciais problemas de pesquisa.

Nesta seção apresentamos uma visão geral dos principais avanços na combinação de agilidade com arquitetura, no entanto para o praticante da engenharia de software é importante conhecer exemplos de como estes avanços estão sendo aplicados na prática. Para atender esta necessidade nas duas próximas seções apresentaremos estudos selecionados sobre a documentação e avaliação de arquitetura com agilidade.

4 Descrição de Arquitetura com Agilidade

A descrição de arquitetura com agilidade, conforme mostrado na seção anterior, foi a atividade de arquitetura de software mais pesquisada. Dentre os estudos nesta atividade selecionamos dois como exemplos que podem ajudar o praticante da engenharia de software, o trabalho de Hadar et al. (2013) que apresenta um modelo de documentação ágil da arquitetura para projetos que utilizam metodologias tradicionais e o de Jensen, Platz e Tjørnehøj (2008) que mostra uma proposta de documentação e organização do trabalho de arquitetura integrado ao *extreme programming* (XP).

4.1 Descrição de Arquitetura com Agilidade em Projetos Tradicionais

Hadar et al. (2013) apresentaram um trabalho feito em parceria com uma grande empresa de desenvolvimento visando estudar os problemas de documentação da arquitetura e propor uma forma documentá-la utilizando somente as informações estritamente necessárias. Para tal, foram realizadas entrevistas, observações e aplicados questionários com os arquitetos responsáveis por elaborar e revisar as arquiteturas, sendo os dados qualitativos sumarizados utilizando teoria fundamentada em dados *(grounded theory)*. As principais descobertas foram sobre a criação, atualização e compreensibilidade dos documentos. No estudo 52% dos arquitetos reportaram que era difícil criar os documentos e 62% que a tarefa era trabalhosa, sendo necessário despender muito esforço e tempo nela. Além disto, nem todos os arquitetos entendiam a importância de manter o documento de arquitetura atualizado, sendo seguido um processo de atualização *ad hoc*. Sobre a compreensibilidade 48% dos arquitetos reportaram que era difícil encontrar informações no documento, e 42% que era difícil de ler e seguir documentos de projetos anteriores, tendo como consequência pouco reuso arquitetural entre os projetos.

Dado este cenário foram propostos um documento de arquitetura simplificado chamado de documento de especificação de arquitetura abstrata, e uma ferramenta para ajudar a preenche-lo chamada de ferramenta de especificação abstrata. O modelo do documento prevê as seguintes seções: visão geral do produto, objetivos do produto

para as próximas liberações, visão geral da arquitetura do produto e mudanças durante os *sprints*, requisitos fundamentais, acrônimos e definições.

A seção de visão geral do produto é uma descrição resumida de até 100 palavras contendo os objetivos do produto e valor para o negócio, o problema que o produto resolve, tipos de empresa/usuários alvo e links para documentação relevante ou produtos similares. A seção de descrição dos objetivos para a próxima liberação é limitada também a 100 palavras e deve conter os entregáveis planejados e links para documentação relevantes. Já a seção de visão geral da arquitetura do produto apresenta a descrição da versão atual contendo as alterações antecipadas para as próximas liberações. Esta seção é limitada a 1000 palavras contendo os principais componentes utilizados na versão atual, alterações planejadas nos componentes existentes para próximas versões, novos componentes que serão utilizados nas próximas versões e modelos de arquitetura relacionados. A seção de requisitos fundamentais relaciona em formato de tabela os objetivos do projeto com o impacto de arquitetura nos requisitos não funcionais, contendo a cobertura da versão atual, ou seja, os atributos de qualidade o produto suportava no passado, e, a cobertura planejada para a próxima versão. Por fim, a seção de acrônimos e definições contém uma breve descrição de cada um, e, se for necessário, links para locais que contenham maiores explicações.

Com o objetivo de facilitar e restringir o preenchimento do documento foi desenvolvida a ferramenta de especificação abstrata, que é integrada a uma ferramenta de modelagem. Nesta o arquiteto somente preenche campos em formulários específicos enquanto cria o projeto e modelos de arquitetura, sendo o documento gerado pela ferramenta. Esta provê também *checklists* que ajudam os arquitetos a coordenarem suas atividades.

Foi observado pela aplicação do modelo de documento e ferramenta o aumento da compreensibilidade do documento de arquitetura, uma vez que ele é mais curto, focado e estruturado segundo um *template* definido. Este resultado pode ser associado a terem sido movidas informações desnecessárias para o entendimento da arquitetura para outros documentos como a especificação de requisitos, e, também, a ferramenta ter restringido os arquitetos a colocarem somente as informações necessárias no documento. Outros resultados são o aumento da consistência do documento por conta da aplicação de *checklists*, e a facilitação da localização de informação sobre arquitetura, uma vez que os links no documento provem acesso a informações externas incluindo a base de componentes de arquitetura da empresa.

Apesar da proposta de documento ser promissora existem algumas questões para que o praticante da engenharia de software possa utilizá-la. Primeiramente a ferramenta desenvolvida não está disponível de forma livre, e foi reportado no artigo que ela suportava a criação do documento, mas não sua atualização. Além disto, não foi apresentado um exemplo concreto de documento de arquitetura, deixando o leitor sem uma visão concreta do seu funcionamento. Como sugestão para o praticante da engenharia de software imaginamos que seja possível utilizar ferramentas como *wiki*, *markdown* e sistemas de controle de verão para implementar uma ferramenta similar de forma simplificada, sendo possível a integração desta com ferramentas de modelagem através de plug-ins. Mesmo se não for possível desenvolver uma ferramenta, ter um modelo de documento de arquitetura simplificado e que restringe o seu conteúdo ajuda a agilizar a elaboração e compreensão do mesmo, sendo uma prática simples e eficiente para a aplicação de agilidade na arquitetura de projetos tradicionais.

4.2 Descrição de Arquitetura em Projetos Ágeis

Jensen, Platz e Tjørnehøj (2008) realizaram um estudo sobre a utilização de histórias de desenvolvedor para gerenciar a evolução da arquitetura em um projeto XP (*extreme programming*). Histórias de desenvolvedor são relatos curtos que descrevem propriedades unitárias ou mudanças nelas visíveis ao desenvolvedor de software. Um exemplo de história de desenvolvedor é mostrado na figura 5, onde é reportada a necessidade de separação de duas partes de código-fonte sendo também apresentado o tempo estimado e realizado da tarefa.

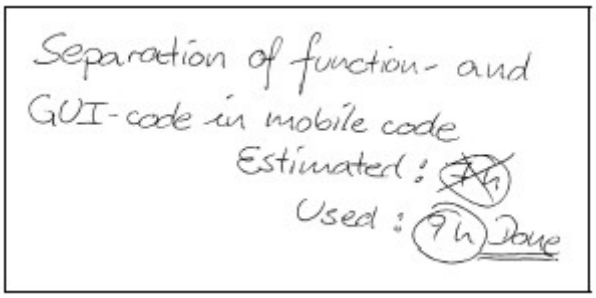

Figura 5 – Exemplo de História de Desenvolvedor

A inserção de histórias de desenvolvedor no modelo de processo XP ocorreram nas atividades de identificação dos problemas arquiteturais, que podem ocorrer a qualquer momento gerando uma nova história de desenvolvedor, e na criação de uma nova etapa chamada de jogo arquitetural. O jogo arquitetural visa selecionar quais histórias de desenvolvedor serão realizadas no *sprint*, sendo uma reunião de aproximadamente duas horas realizada a cada iteração antes ou após o jogo do planejamento. A figura 6 mostra o funcionamento do jogo arquitetural, primeiro os desenvolvedores exploram e elaboram as histórias se baseando na experiência prévia e em ideias que surgiram durante o trabalho, após isto as histórias são exibidas para o cliente identificar o valor para o negócio de cada história e selecionar quais serão desenvolvidas.

As hipóteses do estudo foram que histórias de desenvolvedor poderiam aumentar o compartilhamento e personificação do conhecimento da arquitetura pelos desenvolvedores e também aumentar a visibilidade do cliente sobre mudanças na arquitetura. O estudo foi realizado criando uma equipe XP com seis desenvolvedores do quinto período do curso de ciência da computação, mas utilizado o projeto e cliente industriais. Foram realizadas no experimento três interações de uma semana com os desenvolvedores trabalhando tempo parcial em um ambiente físico configurado para a aplicação do XP. Os dados foram coletados e analisados através de observação diária do pesquisador, aplicação de questionários respondidos três vezes por interação, filmagem do jogo arquitetural e análise do sistema de controle de versão.

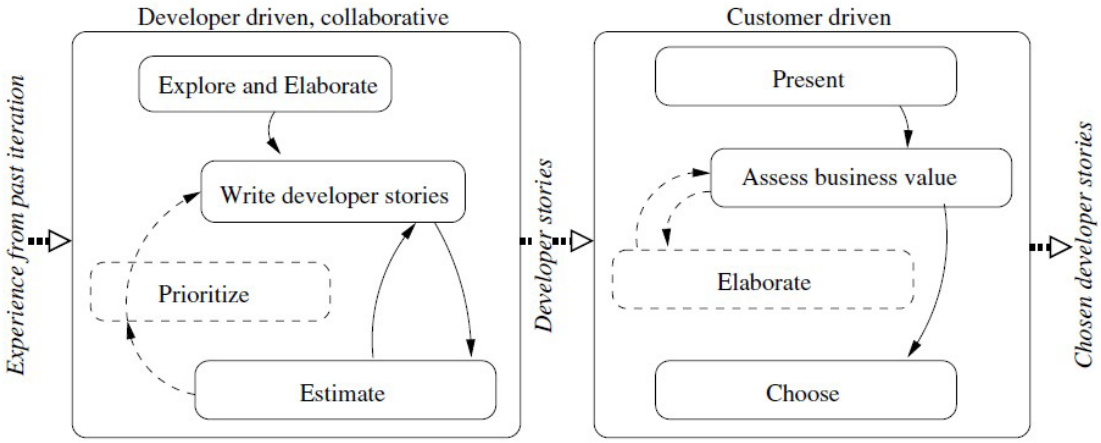

Figura 6 – Dinâmica do Jogo Arquitetural

Os resultados mostraram qualitativamente que histórias de desenvolvedor ajudam a compartilhar o conhecimento de arquitetura por comunicação osmótica entre desenvolvedores e cliente. Nem sempre o cliente consegue acessar facilmente o valor para o negócio de mudanças arquiteturais, precisando de ajuda dos desenvolvedores nesta tarefa. Além disto, as histórias de desenvolvedor ajudaram ambos a focarem em requisitos não funcionais e arquiteturais, criando uma ocasião disciplina para elaboração da arquitetura e aumentando a visibilidade das necessidades de alteração arquitetural. Por outro lado, não foi possível concluir no estudo nenhum aspecto sobre a qualidade da arquitetura produzida, pois não foi utilizado um grupo de controle no experimento. Mesmo assim foi reportado que as refatorações arquiteturais diminuíram o acoplamento e aumentaram a coesão do software, sendo também facilmente rastreáveis para histórias de desenvolvedor.

Uma questão interessantes de ser explorada pelo praticante da engenharia de software é a combinação de histórias de desenvolvedor com outras metodologias de gestão como o Scrum, ou até mesmo em projetos tradicionais, pois é um método leve não só de documentação de arquitetura mas também de gestão da evolução dela.

5 Avaliação de Arquitetura com Agilidade

A avaliação de arquitetura com agilidade foi identificada no mapeamento sistemático de literatura (YANG; LIANG; AVGERIOU, 2016) como a atividade de arquitetura com segundo maior número de artigos. Nesta seção trazemos dois artigos relacionados ao tema, no primeiro Eloranta e Koskimies (2012) apresentam uma forma de avaliação leve de arquitetura que é introduzida no Scrum, e no segundo Nord, Brown e Ozkaya (2011) apresentam uma metodologia de elaboração e avaliação de arquitetura utilizando prototipagem.

5.1 Avaliação de Arquitetura Integrada ao Scrum

Eloranta e Koskimies (2012) apresentaram a inclusão no Scrum do método de avaliação de arquitetura chamado *decision centric architecture review* (DCAR) (VAN HEESCH et al., 2014). O DCAR é um método leve e incremental que realiza a avaliação das decisões arquiteturais levando em consideração as forças de influência e relacionamentos entre as decisões. Este método pode ser dividido em três grandes etapas, primeiramente são identificadas as forças que influenciaram as decisões durante a etapa de design,

em seguida são identificadas as decisões específicas juntamente com quais forças as influenciaram e como as decisões se relacionam, e, por fim, as decisões são documentadas e avaliadas se são boas, aceitáveis ou precisam ser reconsideradas. A figura 7 mostra um exemplo de diagrama de relacionamento entre as decisões arquiteturais, sendo que nesta cada decisão é uma elipse e os relacionamentos são mostrados por setas. A figura 8 mostra um exemplo de documentação de decisão, juntamente com o ponto de vista dos revisores que podem escolher serem favoráveis à decisão (verde), aceita-la (amarelo) ou recomendar reconsideração (vermelho).

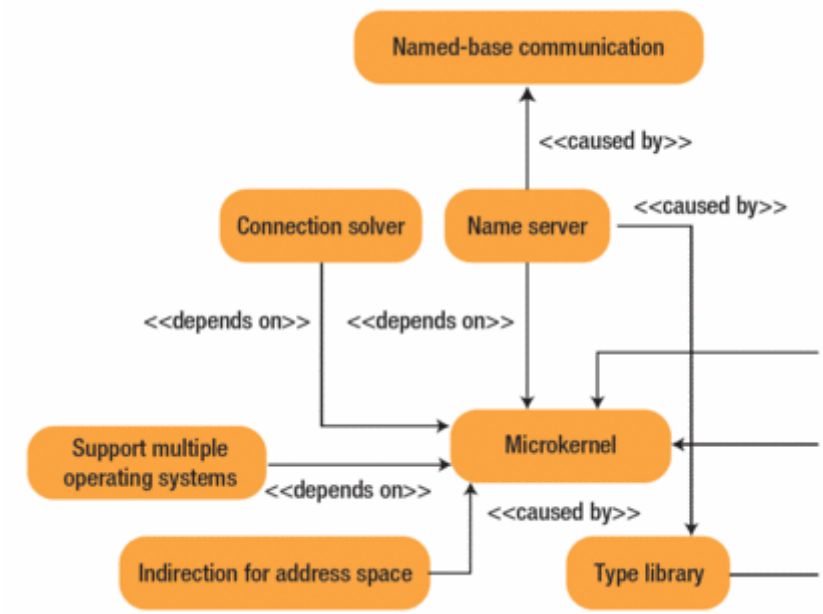

Figura 7 – Diagrama de Relacionamento entre Decisões (VAN HEESCH et al., 2014)

Além do DCAR Eloranta e Koskimies (2012) incluem no Scrum a utilização de gestão do conhecimento arquitetural (VAN VLIET, 2008) para evitar o evaporação do mesmo, e a realização de avaliação utilizando o método *architecture tradeoff analysis method* (ATAM) (KAZMAN; KLEIN; CLEMENTS, 2000) em situações onde uma análise mais robusta e detalhada é necessária. As modificações no Scrum foram incorporadas levando em consideração quatro modelos identificadas na indústria de organização das atividades de arquitetura em conjunto com este método de gestão, são eles: *big up front design*, *sprint zero*, *in-sprints* e time separado de arquitetura.

Name	Redundancy of controllers			
Problem	The application should run even if the server fails			
Solution or description of decision	The system is deployed to two servers: one is active, the other one is inactive. The active server provides all system services, while the passive one is running in the background. When the active server fails, the inactive server becomes active. During the switch over, the active server tries to update the passive one to make sure that it has the same data and status. Both servers have an identical software configuration. This solution follows the *Redundant Functionality Pattern*.			
Considered alternative solutions	Apply the *Redundancy Switch Pattern*: Both servers are active; external logic is used to decide which output is actually used in the control. In this case, cyclic data copying could be avoided. However, applying this solution would require major modifications to the system. Even though availability would be increased, it would also cause additional costs. The customers are not prepared for paying more for higher availability. Additionally, the external logic component could become a potential single point of failure. Therefore, this alternative was discarded.			
Forces in favor of decision	• Easier to implement than the alternative solution • Scales easily to versions where redundancy is not used • No additional costs			
Forces against the decision	• Slower switch over time than the alternative would have • Hard to offer higher availability than the current 99.99%			
Outcome	Green	Yellow	Yellow	Red
Rationale for outcome	Current solution seems to be ok.	I am concerned about the slow switch over time.	Widely accepted solution. Availability might become a problem in the future.	We should really reconsider this decision, as the next release is likely to have higher availability requirements.

Figura 8 – Documentação de Decisão e sua Avaliação (VAN HEESCH et al., 2014)

A organização do Scrum no modelo *big up front design*, mostrada na figura 9, prevê a elaboração da arquitetura de forma preliminar podendo levar um prazo de seis semanas a seis meses, sendo realizada por uma equipe de arquitetura separa da de desenvolvimento. Após elaborada a versão inicial da arquitetura ela é avaliada utilizando o ATAM. As informações geradas nesta avaliação e decisões arquiteturais são armazenadas em uma base de conhecimento de arquitetura (AKB) que é utilizada posteriormente para gestão e geração da documentação. Em seguida, os *sprints* são realizados e ao término de cada *sprint* as decisões de arquitetura tomadas são avaliadas utilizando o DCAR, permitindo que a arquitetura evolua sendo avaliada por um método leve e incremental. Já o modelo *sprint zero* difere do *big up front design* somente pela elaboração da arquitetura inicial ser realizada em um *sprint* normal, gerando uma arquitetura de forma rápida porém incompleta.

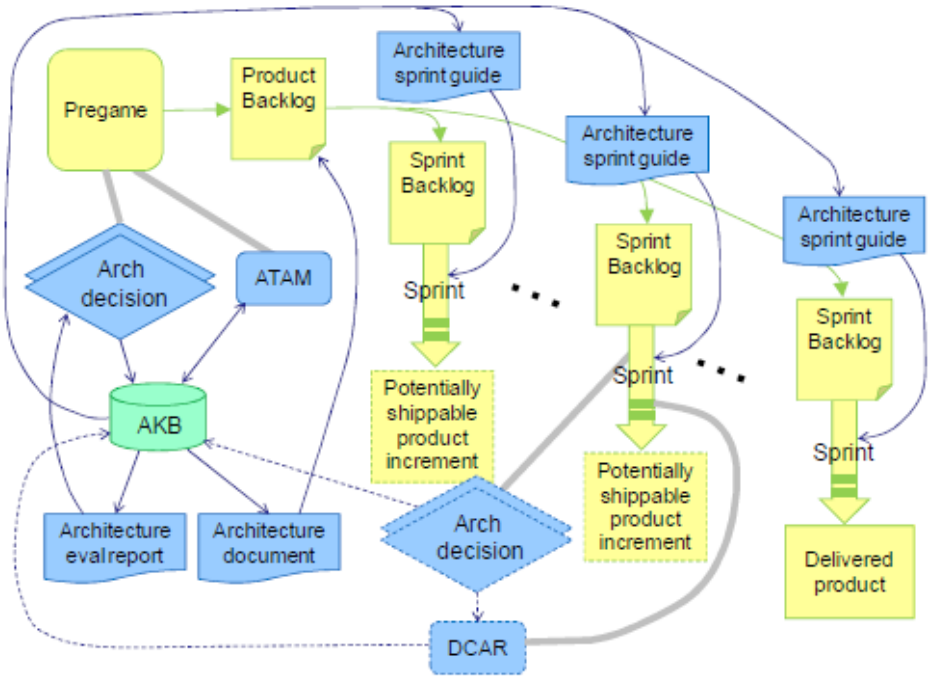

Figura 9 – Organização de atividades de arquitetura com Scrum no modelo *big up front design*

O modelo i*n-sprints* não considera a elaboração de uma arquitetura inicial e nem utilização do ATAM, mas somente a elaboração da arquitetura conforme cada *sprint* é realizado sendo posteriormente avaliada utilizando o DCAR. Este modelo presume que a arquitetura é elaborada por membros da equipe de desenvolvimento, mas pode ser adaptada para suportar uma equipe de desenvolvimento separada. Os modelos *in-sprints* e de equipe de desenvolvimento separada são mostradas nas figuras 10 e 11 respectivamente.

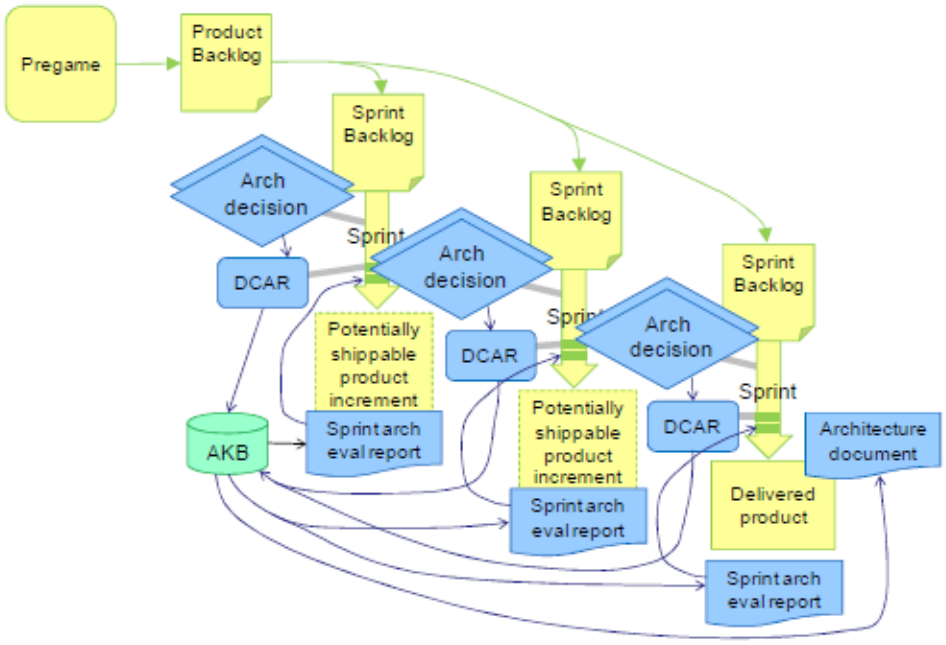

Figura 10 - Organização de atividades de arquitetura com Scrum no modelo *in-sprints*

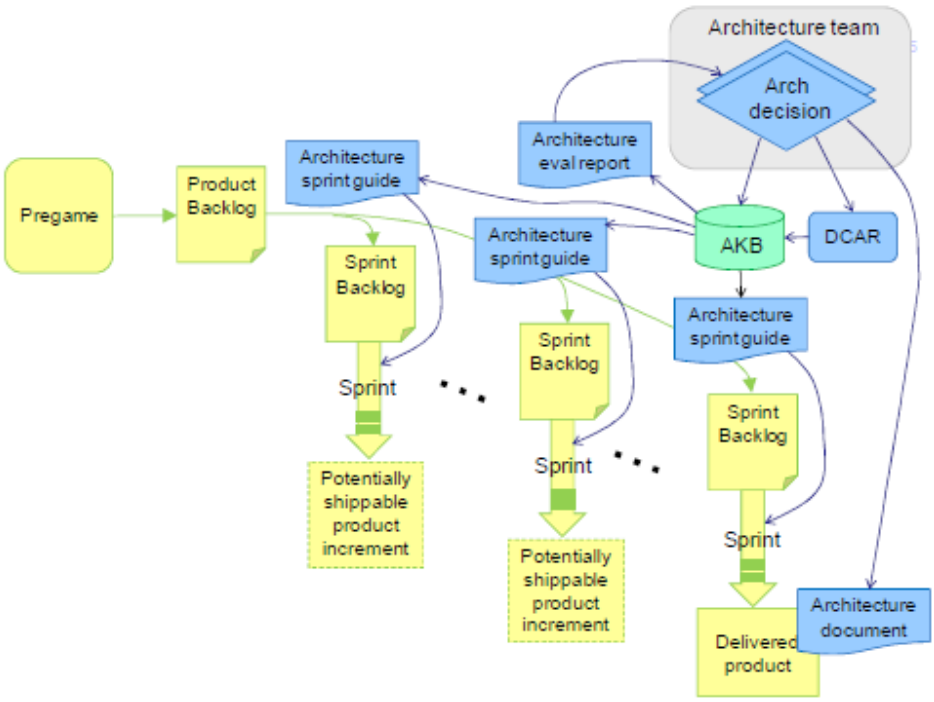

Figura 11 - Organização de atividades de arquitetura com Scrum no modelo de equipe de desenvolvimento separada

Apesar de terem sido reportados quatro modelos de utilização de arquitetura com o Scrum estes são somente conceituais, não tendo sido experimentados na prática. Por conta disto, o praticante da engenharia de software deve estar atento a possíveis inconsistências ao utilizá-los. Em nossa avaliação o modelo *sprint zero* deve prover pouca informação arquitetural no *sprint* inicial para aplicação do ATAM, ou mesmo, este pode consumir muito tempo atrasando *sprints* posteriores. Neste caso, parece mais adequado utilizar o DCAR como método de avaliação desde o início, o que faria o modelo *sprint zero* ser idêntico ao *in-sprints*. Apesar destas questões a utilização de métodos de avaliação de arquiteturas leves e incrementais como o DCAR possibilita a evolução disciplinada da arquitetura em projetos com gestão ágil.

5.2 Avaliação de Arquitetura Incremental com Prototipagem

Nord, Brown e Ozkaya (2011) apresentam uma abordagem interativa para desenvolvimento e avaliação da arquitetura de software utilizando prototipagem e feedback. Nesta abordagem a cada interação a arquitetura é desenvolvida mesmo de forma incompleta e colocada em funcionamento parcialmente para obter feedback e continuar sua evolução, por fim é aplicado o ATAM (KAZMAN; KLEIN; CLEMENTS, 2000) para que se tenha uma avaliação mais formal da arquitetura.

A figura 12 apresenta a dinâmica do processo iterativo, nesta as setas mostram o fluxo de informação entre os envolvidos e as seções verticais iterações. Podemos observar que a primeira iteração começa com a elaboração por conta dos *stakeholders* dos direcionadores de arquitetura, que ao final são entregues aos arquitetos. Em seguida é realizada uma atividade de encontrar problemas, onde é criado um design candidato com os cenários de atributos de qualidade significativos com objetivo de estudar a arquitetura. Após isto, os problemas identificados são passados tanto para os *stakeholders* quanto para os desenvolvedores, e enquanto o time de arquitetura especifica em detalhes as áreas bem conhecidas os desenvolvedores criam protótipos das áreas problemáticas para ajudar no seu entendimento. Na iteração seguinte com os resultados do es-

força de prototipação os arquitetos especificam as partes restantes da arquitetura, enquanto isto os desenvolvedores criam uma funcionalidade utilizando o esqueleto parcial criado com as partes bem conhecidas da arquitetura. Ao final desta iteração é gerada uma primeira release para os *stakeholders*, e os arquitetos entregam o resto do design para os desenvolvedores. Na iteração 4 é realizada a atividade de avaliação da arquitetura utilizando o ATAM. Com esta avaliação são descobertos novos riscos e a arquitetura é refinada para mitigá-los. Enquanto isto os desenvolvedores implementam uma segunda funcionalidade utilizando o esqueleto completo da arquitetura, gerando assim uma segunda release. Por fim, por conta do resultado da atividade de avaliação na iteração 5 são realizados ajustes tanto na arquitetura quanto no sistema. A partir desta iteração o desenvolvimento segue gerando releases com a arquitetura ajustada.

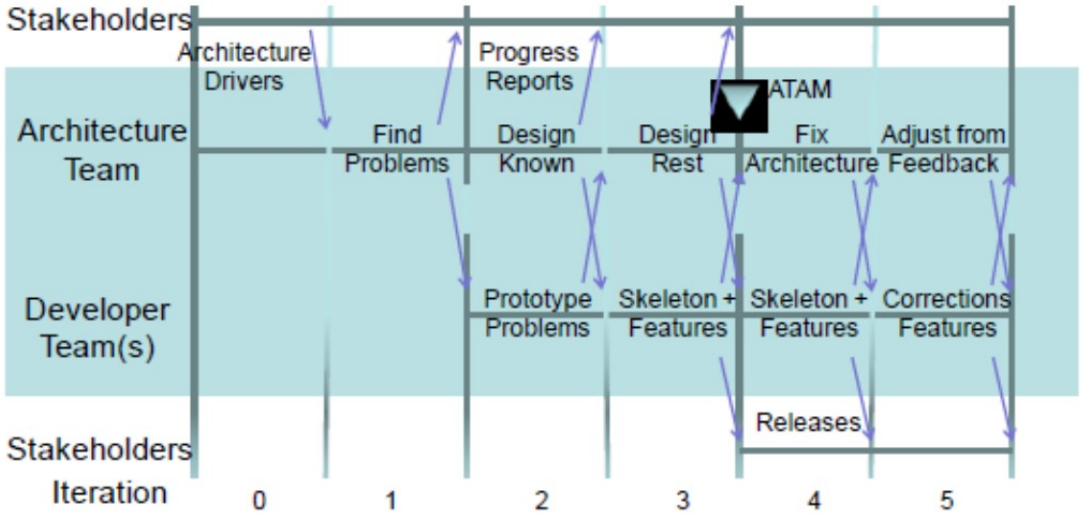

Figura 12 – Processo de Evolução e Avaliação de Arquitetura Incremental (NORD; BROWN; OZKAYA, 2011)

O método apresentado faz uso da informação disponível em cada interação para tanto arquitetar quanto desenvolver, avançando ambas juntas e ajustando o que for necessário. O ATAM é utilizado como método de avaliação no momento que a arquitetura está com um nível de informação suficiente, mitigando os riscos da mesma. Esta metodologia é simples, mas precisa de coordenação entre os times para funcionar adequadamente, sendo aplicável somente a projetos com desenvolvimento centrado na arquitetura onde existe proximidade entre as equipes de desenvolvimento e arquitetura.

Um framework similar ao apresentado por Nord, Brown e Ozakaya (2011) contudo mais geral é introduzido por Nord, Ozkaya e Kruchten (2014). Neste os autores questionam não se arquitetura é importante em métodos ágeis, mas sim quando e com que frequência ela necessária. A metáfora do zíper, mostrada na figura 13, ilustra como esta combinação pode ser realizada. Conforme os requisitos são desenvolvidos e refinados os arquitetos identificam e extraem requisitos arquiteturais significantes (em vermelho), requisitos funcionais (em verde), e dependências entre eles. Pequenas interações são utilizadas para projetar, construir e testar poucos elementos arquiteturais e as funcionalidades que dependem deles. O que for projetado na arquitetura é validado não por testes abstratos, mas sim por um embrião de produto das funcionalidades reais (NORD; OZKAYA; KRUCHTEN, 2014). As iterações de arquitetura e desenvolvimento se intercalam, formando o zíper, mas sem restrições de que iterações tenham somente atividades de um dos tipos. Assim, este framework permite o avanço flexível de arqui-

tetura e desenvolvimento em pequenas partes, com avaliação da arquitetura sem grandes demoras pelas funcionalidades reais implementadas.

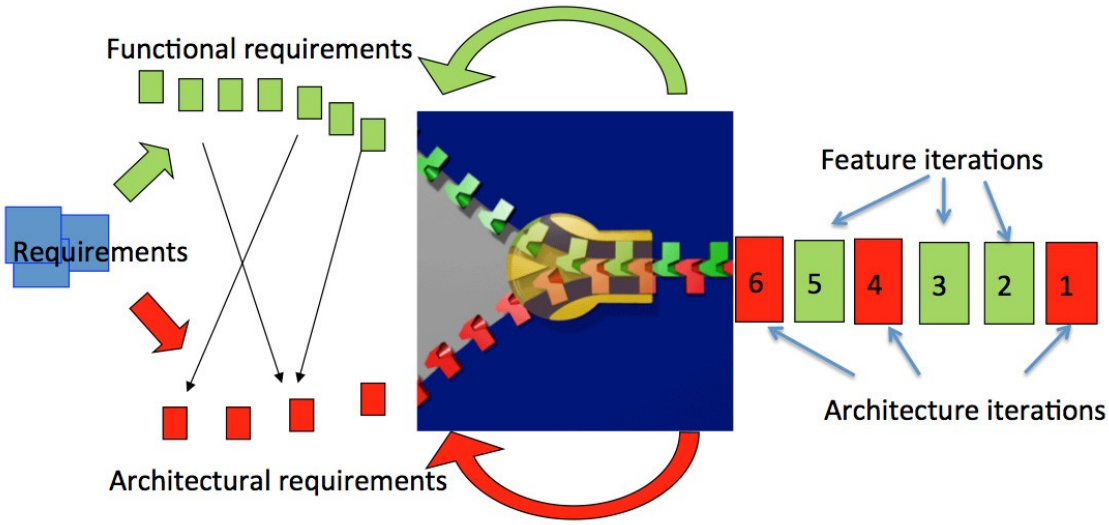

Figura 13 – Metáfora do zíper para combinação de interações funcionais e arquiteturais. (NORD; OZKAYA; KRUCHTEN, 2014)

6 Discussão

Apresentamos na seção 2 os diversos fatores que influenciam o trabalho de arquitetura em um projeto de software, entre eles estão tempo, custo, experiência, habilidade, visibilidade, complexidade, mudanças, escala e fatores organizacionais. Com sempre na engenharia de software não há bala de prata, e portanto combinação de arquitetura e agilidade não trata todos estes fatores. Por outro lado, agilidade tem grande potencial para redução e diluição do tempo e custo de arquitetar, assim como para aumentar a visibilidade, flexibilidade e adequação a mudanças da arquitetura. Já fatores de habilidade, complexidade, escala e fatores organizacionais são abordados superficialmente nos estudos. Um dos possíveis motivos para isto é não ser foco da agilidade tratá-los, ou talvez, a pouca exploração destes fatores em pesquisas.

O mapeamento sistemático apresentado na seção 3 mostrou que existe uma dispersão grande da pesquisa de arquitetura com agilidade, porém as técnicas podem ser separadas em projetos ágeis onde são introduzidas atividades de arquitetura e projetos centrados em arquitetura onde são introduzidos práticas de agilidade. Dentro deste contexto as atividades de arquitetura mais pesquisadas foram a documentação e avaliação de arquitetura. Acreditamos que isto ocorreu devido as abordagens de documentação em projetos tradicionais e ágeis serem ambas inadequadas, sendo excessivamente detalhista em projetos centrados na arquitetura, gerando documentos enormes e complexos, e em projetos ágeis a documentação é justamente o oposto, ou seja, quase inexistente. Já a atividade de avaliação de arquitetura é crítica para não comprometer o projeto com decisões inadequadas, contudo o método mais popular de avaliação chamado ATAM (KAZMAN; KLEIN; CLEMENTS, 2000) exige grande esforço e custo para ser executado. Os métodos ágeis geralmente não possuem um evento de avaliação de arquitetura, mas sim em refatorações desta o quanto for necessário. Consequentemente, caso a arquitetura inicial se distancie da ideal os custos de refatoração serão altos.

As técnicas apresentadas na seção 4 mostram criar a documentação de arquitetura de forma ágil tanto em projetos centrados na arquitetura quanto com metodologias

ágeis. No primeiro estudo é apresentado um modelo de documentação simplificado que visa mover tudo que não for sobre arquitetura para outros documentos e provendo links externos, além disto é limitada a quantidade de informação textual em cada seção. Neste estudo os principais fatores tratados são a redução de custo e tempo para produzir o documento de arquitetura, assim como a melhoria da visibilidade e redução da complexidade da arquitetura. Porém, claramente para implantar este tipo de abordagem é necessária uma ferramenta de apoio ou inspeções para garantir que as restrições no documento seja seguidas. Infelizmente a ferramenta apresentada no artigo não está disponível. Um questionamento que fica sobre o modelo de documento, visto que não foi apresentado um exemplo concreto, é se seria possível explicar diversas visões e perspectivas arquiteturais (ROZANSKI; WOODS, 2011) com somente 1000 palavras para descrição textual, mas sem limites de modelos, ou mesmo, descrever a visão geral do produto com 100 palavras, visto que em sistemas com muitos *stakeholders* somente listá-los já pode ultrapassar este limite.

O segundo estudo sobre documentação de arquitetura inclui histórias de desenvolvedor para organizar e documentar arquitetura no modelo de processo XP. Neste os fatores tratados foram o aumento da visibilidade, adaptação a mudanças, e podemos considerar uma redução no tempo e custo de manutenção a longo prazo como resultado de arquitetar disciplinadamente. Apesar de documentar arquitetura no formato de histórias ser melhor do que não ter documentação fica um questionamento sobre se são suficientes, visto que não incluem muitos elementos importantes como modelos ou explicações sobre decisões arquiteturais. Com isto, acreditamos que apesar da atividade de documentação de arquitetura ser a mais pesquisada em conjunto com a agilidade parece ainda não estar em um nível de maturidade elevada, necessitando de continuidade de pesquisa.

Na atividade de avaliação de arquitetura apresentamos um trabalho sobre inclusão de avaliação arquitetural leve no Scrum, e outro sobre elaboração e avaliação arquitetural de forma incremental com prototipagem. No primeiro os fatores de influência abordados foram a redução do tempo e custo das avaliações e melhor adaptação a mudanças pela aplicação do DCAR, que mostrou ser uma técnica leve e incremental. Porém esta não leva em consideração cenários de risco da arquitetura, mas sim as decisões em relação as forças de influência, então caso as decisões tenham sido tomadas sem levar em consideração algum cenário de risco de nada a avaliação poderá melhorá-las. Também vale ressaltar que as propostas apresentadas de integração do Scrum com o método de avaliação não foram experimentadas na prática, sendo somente conceituais. No segundo estudo foram considerados os mesmo fatores do primeiro, porém sendo abordados somente durante a elaboração da versão inicial da arquitetura. Vale ressaltar também que o método de avaliação apresentado precisa um sincronismo grande entre as equipes de desenvolvimento e arquitetura, levando ao questionamento se seria possível aplicá-lo em projetos grandes ou em empresas que não trabalham com desenvolvimento ágil.

7 Conclusões e Trabalhos Futuros

Arquitetar tem grande influência no sucesso dos projetos de software. Porém, consome tempo significativo no início do projeto e os benefícios só são percebidos a longo prazo. Assim a arquitetura é negligenciadas levando ao padrão arquitetural *"big ball of mud"*(FOOTE; YODER, 1997), ou seja, a completa desordem. Uma das soluções propostas para atenuar o efeito dos altos custos iniciais de arquitetar é a aplicação de agilidade, contudo há uma desorientação do praticantes da engenharia de software em como

efetivamente colocar esta proposta em prática. Com o objetivo de ajudá-los apresentamos neste trabalho um resumo do estado da arte sobre o tema juntamente com artigos que mostram propostas de técnicas de aplicação da combinação de agilidade com arquitetura. Adicionalmente discutimos quais os fatores de impacto que levam a grande bola de lama são tratados por cada técnicas e possíveis restrições para sua utilização.

O estado da arte (YANG; LIANG; AVGERIOU, 2016) mostrou uma dispersão grande da pesquisa de agilidade entre diversas atividades de arquitetura, havendo destaque em números de artigos nas atividades de documentação e avaliação de arquitetura. Por este motivo, revisamos técnicas práticas nas duas atividades aplicáveis tanto a projetos tradicionais centrados na arquitetura quanto a projetos ágeis.

O artigo revisado sobre documentação ágil da arquitetura para projetos tradicionais se baseia na simplificação do documento de arquitetura (HADAR et al., 2013), padronizando e limitando tamanho das seções e também criando links para outros documentos. Esta abordagem ajuda a reduzir os custos e o tempo para elaboração da arquitetura, porém não trata questões relativas a mudanças na documentação e exige o uso de ferramentas para apoiar a elaboração do documento. Por outro lado, para projetos ágeis foi apresentada a técnica de histórias de desenvolvedor (JENSEN; PLATZ; TJØRNEHØJ, 2008) que documentam as questões relativas a arquitetura de maneira ágil, ajudando a dar visibilidade para esta e a tratar mudanças. Porém, a aplicação desta técnica só reduz os custos e tempo das atividades no longo prazo, e não podemos afirmar se histórias de desenvolvedores são adequadas para documentação de arquitetura, visto que não contém modelos e decisões sobre esta.

Na parte de avaliação arquitetural o primeiro trabalho selecionado foi sobre a avaliação de arquitetura em conjunto com o Scrum (ELORANTA; KOSKIMIES, 2012) utilizando o método de avaliação baseado em decisões arquiteturais chamado DCAR (VAN HEESCH et al., 2014). Este trabalho por utilizar um processo incremental mostra como reduzir o tempo e custo de atividades de avaliação e torna-las adaptáveis a mudanças. Por outro lado, não foca nos cenários de risco para arquitetura, deixando esta tarefa por conta do arquiteto. O último trabalho apresentou uma metodologia de elaboração e avaliação de arquitetura iterativa utilizando prototipagem (NORD; BROWN; OZKAYA, 2011). Neste os fatores levados em consideração foram redução de tempo, custo e adaptação a mudanças durante a fase de elaboração da arquitetura. Esta técnica utiliza também o ATAM (KAZMAN; KLEIN; CLEMENTS, 2000) como método de avaliação mais formal, contudo ela exige grande sincronização entre equipe de desenvolvimento e arquitetura, deixando em dúvida sua aplicabilidade em projetos grandes ou em equipes com pouco contato.

Por fim, esperamos ter ajudado o praticante da engenharia de software a conhecer algumas técnicas de combinação de agilidade com arquitetura. Como trabalho futuro sugerimos a implementação destas técnicas na prática e seu refinamento para que elas passem de propostas abstratas para relatórios mais completos e com avaliações quantitativas de como elas agem sobre os fatores de influência na arquitetura. Outra sugestão de trabalho futuro é a combinação de técnicas para formar metodologias ágeis completas de trabalho de arquitetura, levando em consideração não somente uma, mas um conjunto completo de atividades de arquitetura.

Referências Bibliográficas

BECK, K. **Extreme programming explained: embrace change**. [s.l.] addison-wesley professional, 2000.

ELORANTA, V.-P.; KOSKIMIES, K. **Aligning architecture knowledge management with Scrum**. Proceedings of the WICSA/ECSA 2012 Companion Volume. **Anais**...2012
FOOTE, B.; YODER, J. Big ball of mud. **Pattern languages of program design**, v. 4, p. 654–692, 1997.
FREUDENBERG, S.; SHARP, H. The Top 10 Burning Research Questions from Practitioners. **IEEE Software**, v. 27, n. 5, p. 8–9, 2010.
HADAR, I. et al. **Less is more: Architecture documentation for agile development**. Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th International Workshop on. **Anais**...2013
JENSEN, R. N.; PLATZ, N.; TJØRNEHØJ, G. Developer Stories: Improving Architecture in Agile Practice. In: **Software and Data Technologies**. [s.l.] Springer, 2008. p. 172–184.
KAZMAN, R.; KLEIN, M.; CLEMENTS, P. **ATAM: Method for architecture evaluation**. [s.l: s.n.].
NORD, R. L.; BROWN, N.; OZKAYA, I. **Architecting with just enough information**. Proceedings of the 6th International Workshop on SHAring and Reusing Architectural Knowledge. **Anais**...2011
NORD, R. L.; OZKAYA, I.; KRUCHTEN, P. **Agile in distress: Architecture to the rescue**. International Conference on Agile Software Development. **Anais**...2014
ROZANSKI, N.; WOODS, E. **Software systems architecture: working with stakeholders using viewpoints and perspectives**Addison-Wesley Professional, , 2011.
VAN HEESCH, U. et al. Decision-centric architecture reviews. **IEEE software**, v. 31, n. 1, p. 69–76, 2014.
VAN VLIET, H. **Software architecture knowledge management**. 19th Australian Conference on Software Engineering (ASWEC 2008). **Anais**...2008
YANG, C.; LIANG, P.; AVGERIOU, P. A systematic mapping study on the combination of software architecture and agile development. **Journal of Systems and Software**, v. 111, p. 157–184, 2016.