SciAgile: Aplicação de Metodologias Ágeis em Experimentos Científicos Baseados em Simulação

Augusto Romeiro[1], Kary Ocaña[2], Marcos Kalinowski[1], and Daniel de Oliveira[1]

[1] Instituto de Computação - Universidade Federal Fluminense (UFF), Niterói - Brasil
{augusto.romeiro,kalinowski,danielcmo}@ic.uff.br
[2] Laboratório Nacional de Computação Científica (LNCC), Petrópolis - Brasil
karyann@lncc.br

Resumo Muitos dos riscos envolvidos no desenvolvimento de *software* tem associação com requisitos voláteis. Tal volatilidade motivou a criação e a adoção de metodologias ágeis, que hoje são uma realidade nas organizações. A volatilidade dos requisitos também pode ser encontrada na pesquisa científica, que cada vez mais depende de complexas simulações computacionais. Tal como *software* comercial, o *software* científico, sua execução e a análise dos resultados precisam ser compreensíveis, passíveis de reprodução e livres de defeitos. Como requisitos de experimentos científicos também são comumente voláteis, resolvemos investigar se as metodologias ágeis podem também ser aplicadas neste contexto. Esse artigo apresenta uma abordagem, denominada SciAgile, que aplica uma série de práticas ágeis no ciclo de vida de experimentos científicos. Para avaliar a proposta, realizamos um estudo experimental que envolveu a aplicação da abordagem SciAgile na gerência de experimentos. Resultados fornecem indícios que a adoção das práticas ágeis propostas pode ajudar a melhorar o processo de experimentação científica *in silico*.

1 Introdução

É de conhecimento comum que o desenvolvimento de *software* é uma atividade complexa e que envolve potenciais riscos, principalmente no que tange o orçamento, o consumo de tempo e a qualidade do produto final. Muitos dos riscos existentes estão diretamente associados a requisitos voláteis [1]. Metodologias de desenvolvimento de *software* tradicionais nem sempre se mostram adequadas em cenários que envolvem requisitos voláteis. Com o intuito de se possuir um processo mais aderente às mudanças de requisitos constantes foram propostas metodologias ágeis [2]. Hoje, o uso de metodologias ágeis e híbridas (orientadas a plano [3], mas que fazem uso de práticas comuns em métodos ágeis) em grandes corporações é uma realidade [4].

Entretanto, não é somente na indústria que *software* é desenvolvido e impactado por mudanças constantes nos requisitos. A área científica tem se tornado cada vez mais dependente do desenvolvimento de abordagens computacionais, principalmente em áreas cujos experimentos dependem fortemente de simulações computacionais complexas. Tais experimentos são denominados experimentos *in silico* [5]. Muitos dos cientistas de diversas áreas, como a biologia e a química, já desenvolvem *software* como parte fundamental de sua pesquisa, seja diretamente

escrevendo programas ou criando novas aplicações por meio da composição de *software* e serviços existentes, seja na forma de um *workflow* científico ou um *script*. Nesses casos, os objetos de análise dos experimentos são usualmente processados por simulações computacionais e analisados via técnicas de visualização e/ou mineração de dados. Tal como os desenvolvedores de *software* comercial, o código desenvolvido (ou o *workflow* modelado), a execução do mesmo bem como a análise dos resultados precisam ser compreensíveis, passíveis de reprodução e livres de defeitos.

A Engenharia de *Software* é capaz de fornecer métodos e técnicas para aprimorar tais experimentos, como ressaltado por Pereira e Travassos [6], e, uma vez que requisitos no experimento científico também são voláteis, cria-se a expectativa de que metodologias ágeis possam também ser aplicadas neste contexto. Ainda, a aplicação de práticas de metodologias ágeis pode ser extrapolada do desenvolvimento de *software* para outras fases do ciclo de vida do experimento científico, como, por exemplo, a análise dos resultados. Alguns trabalhos já foram propostos na literatura para a aplicação de metodologias ágeis no processo de experimentação científica [7,8,9,10]. Entretanto, tais trabalhos cobrem somente a etapa do desenvolvimento do *software* (chamada de composição de acordo com [5]), sem explorar o ciclo de vida do experimento por completo.

Este artigo apresenta uma abordagem, chamada de SciAgile, que aplica uma série de práticas de diferentes metodologias ágeis como Scrum [11], *eXtreme Programming* (XP) [12] e Kanban [13] no ciclo de vida do experimento científico originalmente proposto por Mattoso *et al* [5]. Adicionalmente, um estudo experimental avaliando a abordagem SciAgile é apresentado. Resultados apontam para uma redução no tempo no ciclo de vida do experimento científico e para a aceitação dos usuários.

Este artigo está organizado em 5 Seções, incluindo esta introdução. A Seção 2 apresenta o referencial téórico e os trabalhos relacionados. A Seção 3 apresenta a abordagem proposta enquanto a Seção 4 descreve a estudo experimental da abordagem proposta. Finalmente, a Seção 5 conclui esse artigo e apresenta os trabalhos futuros.

2 Referencial Teórico

2.1 Experimentos Científicos Baseados em Simulação

Nos últimos anos, cientistas têm utilizado complexas simulações computacionais em seus experimentos científicos [14]. Um experimento científico está associado a um conjunto de ações controladas (*i.e.* um protocolo) com objetivo de se atingir um resultado, e, após análise dos resultados do mesmo, ser capaz de confirmar ou refutar uma hipótese. Os principais ambientes que executam experimentos científicos, a saber são: *in vivo, in vitro, in virtuo e in silico* [5]. Na abordagem proposta nesse artigo estamos interessados em experimentos *in silico* (baseados em simulação). Deste ponto do artigo em diante, o termo "experimento científico" será utilizado para referenciar somente experimentos baseados em simulação. Este tipo de experimento é utilizado nos mais diversos domínios científicos desde a biologia até a astronomia [14].

Em muitos cenários, os experimentos científicos são representados por meio do encadeamento de programas. Nesses experimentos, cada programa consome um conjunto de valores de parâmetros e dados e a saída de um programa é normalmente utilizada como entrada para outro programa no fluxo. Esse encadeamento de programas e suas dependências de dados é denominado *workflow* científico. Um programa em um *workflow* é chamado de atividade. Como a gerência dos *workflows* não é uma tarefa trivial, existem sistemas especializados em auxiliar na sua composição, execução e análise: os sistemas de gerência de *workflows* científicos (SGWfC). Existe uma gama de SGWfCs disponíveis [14], como o SciCumulus [15] (que foi usado na avaliação da proposta). Neste artigo, o conceito de experimento científico engloba o conceito de *workflow*, não sendo considerados sinônimos. A execução de um *workflow* pode ser vista como um conjunto de ações controladas de um experimento.

Cada experimento científico possui um ciclo de vida definido. Neste artigo, o ciclo de vida do experimento científico desempenha um papel fundamental, pois ele serve como "guia" para a aplicação de práticas ágeis no contexto do experimento científico. Existem diversas propostas de ciclos de vida de experimentos científicos, mas utilizaremos como base a proposta de Mattoso *et al.* [5]. A Figura 1 apresenta o ciclo de vida de um experimento científico, que consiste n interações que são percorridas pelo cientista diversas vezes durante o experimento. As principais fases do ciclo de vida do experimento podem ser definidas como: a composição, a execução e a análise. De acordo com o modelo proposto por Mattoso *et al.*, cada fase possui um subciclo independente, que é percorrido em momentos distintos do experimento.

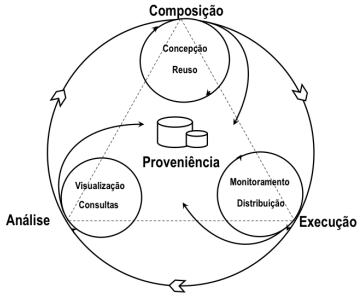


Figura 1. Ciclo de Vida do Experimento Científico de acordo com Mattoso *et al.* [5]

A fase de **Composição** é onde o cientista estrutura e modela todo o experimento e a estrutura dos *workflows* envolvidos no mesmo. Podemos decompor a fase de composição em duas subfases: a concepção e o reuso. A concepção é responsável pela criação do experimento enquanto que o reuso recupera uma definição de experimento existente e a adapta para um novo propósito. A fase de **Execução** é responsável por, a partir de uma especificação de *workflow*, executar os programas consumindo um conjunto de dados. A execução pode ser decomposta em duas subfases: distribuição e monitoramento. A subfase de distribuição está relacionada com a alocação de programas em ambientes computa-

cionais para a execução, enquanto a subfase de monitoramento está relacionada à necessidade de verificar periodicamente o estado atual da execução do *workflow*, uma vez que este pode executar por um longo tempo. Finalmente, a fase de **Análise** é responsável por analisar, validar e gerar conclusões a partir do dados gerados pelas fases de composição e execução.

2.2 Metodologias Ágeis

A proposta principal das metodologias ágeis é fazer com que as pessoas estejam comprometidas com a agilidade e com a qualidade do *software* a ser desenvolvido. Nessas metodologias é valorizada a entrega de um produto funcional e adequado ao que o cliente realmente deseja. Existem diversas metodologias ágeis, como o XP e o SCRUM. Embora cada uma tenha uma característica própria, no que tange suas práticas, as mesmas compartilham características em comum, incluindo o desenvolvimento iterativo e o foco na comunicação interativa e na redução do esforço empregado em artefatos intermediários. Neste artigo utilizaremos como base práticas propostas pelo SCRUM [11], XP [12] e Kanban [13].

O Scrum é uma das metodologias ágeis mais utilizadas no mercado. Ele é baseado no conceito de ciclos ou iterações sucessivas. Tais ciclos são intervalos de tempo bem definidos que são chamados de *sprints*. Cada *sprint* é considerado um pequeno projeto com todas as atividades necessárias para se gerar um produto estável e funcional, mesmo que não seja entregue para seus usuários imediatamente. A execução do *sprint* envolve a realização de cerimônias bem definidas, que são reuniões que acontecem em momentos distintos do *sprint*. Tais reuniões contam com todos os membros do projeto, que podem ser o *Scrum Master* (responsável por fazer com que a equipe se mantenha organizada), o *Product Owner* (garante o valor do produto e o retorno para o cliente) e a equipe de desenvolvimento.

O XP, assim como o Scrum, tem como objetivo principal prover eficiência no desenvolvimento mantendo a qualidade. O XP tem o foco em facilitar a comunicação, a resposta ao usuário e a colaboração entre os membros da equipe. Por fim, o Kanban é um arcabouço que tem como objetivo expor problemas e definir o que realmente deve ser feito no processo. Para isso, o mesmo utiliza um quadro com alguns cartões colados, onde os cartões representam as tarefas e cada cartão transita entre as atividades do processo.

De forma a comparar as abordagens existentes que aplicam metodologias ágeis no ambiente científico e definir quais práticas seriam utilizadas na SciAgile, foi adaptada uma listagem de práticas ágeis inicialmente proposta por Sletholt *et al.* [16] provenientes do Scrum, XP e Kanban e que poderiam também ser aplicadas em projetos de experimentação científica: (i) Existe um papel dedicado para priorizar as funcionalidades, (ii) Existe um papel de um facilitador do processo de experimentação/desenvolvimento, (iii) Existe uma reunião de planejamento do ciclo para criar uma lista de tarefas, (iv) É realizado um planejamento para estimar as tarefas durante o planejamento do ciclo, (v) Existe um ciclo com tempo definido e produzindo um incremento do produto/entregável,

(vi) Existe compromisso mútuo entre coordenador e time de desenvolvimento no que tange as tarefas do ciclo, (vii) É realizada uma curta reunião diária para resolver os problemas atuais, (viii) Os membros da equipe se voluntariariam para realizar as tarefas, (ix) É elaborado um gráfico de *Burndown* para monitorar o progresso das tarefas do ciclo, (x) É realizada uma reunião para revisar o trabalho que foi completado, (xi) É realizada uma reunião para aprender com os resultados do ciclo anterior, (xii) É realizado um planejamento para gerar incrementos do produto/entregável, (xiii) São escritas *user stories*, (xiv) O time tem um espaço aberto para trabalho, (xv) É definido um ritmo de trabalho, (xvi) A velocidade do projeto é capaz de ser medida, (xvii) As pessoas são conectadas para não haver perda de conhecimento, (xviii) O coordenador deve estar sempre disponível, (xix) O desenvolvimento de programas/métodos e a execução do experimento deve ser padronizada, (xx) Testes devem ser desenvolvidos, (xxi) Toda produção de programas/*workflows* deve ser feita em pares (*pair programming*), (xxii) Cada par deve realizar a integração dos produtos desenvolvidos de cada vez, (xxiii) Deve ser feita a integração contínua, (xxiv) Deve-se configurar um computador/ambiente para realizar a integração, (xxv) É utilizada a propriedade coletiva, (xxvi) É fomentado o uso de arquitetura simples, (xxvii) Escolha metáforas para o projeto, (xxviii) São usadas sessões de *brainstorm* para criar arquiteturas, (xxix) São usados testes de conceito para reduzir os riscos de uma solução (*Spike Solutions*), (xxx) Nenhuma funcionalidade é adicionada no início do projeto sem avaliação da equipe, (xxxi) É realizada a refatoração de código sempre que possível, (xxxii) Todo código deve estar coberto por testes, (xxxiii) Todos os testes unitários devem passar antes de ser entregue um incremento, (xxxiv) Quando é achado um defeito é criado um teste para o mesmo, (xxxv) Testes de aceitação são executados com frequência e a pontuação é publicada, (xxxvi) Visualizar o Fluxo de Trabalho, (xxxvii) Limitar o Trabalho em Progresso, (xxxviii) Medir e Gerenciar o Fluxo,e (xxxix) Melhoria Contínua.

2.3 Trabalhos Relacionados

Para a busca dos trabalhos relacionados realizamos um mapeamento sistemático [17]. Um resumo dos trabalhos diretamente relacionados é apresentado a seguir.

Kane *et al.* [7] apresentam experiências de múltiplos projetos onde métodos ágeis foram aplicados no cenário biomédico. Os autores concluem que métodos ágeis são adequados para área científica. Wood e Kleb [8] apresentam uma tentativa de inclusão de práticas do XP em um projeto de pesquisa da NASA. Das práticas existentes para o XP, a única que não foi utilizada por não haver viabilidade foi a programação em pares. Maxville [9] e Ackroyd *et al.* [10] discutem a aplicação do SCRUM em projetos científicos. Todos os autores relatam experiências positivas com a implementação ágil como a prática incremental, com foco em testes e requisitos.

É importante ressaltar que, apesar de todos os trabalhos afirmarem que a aplicação de metodologias ágeis é interessante, nenhum dos trabalhos utilizou metodologias ágeis em todo ciclo de vida de experimentos científicos. Todos os trabalhos encontrados focaram apenas na etapa do desenvolvimento de *software*.

3 Abordagem Proposta: SciAgile

A abordagem SciAgile consiste na combinação das práticas do Scrum, XP e Kanban adaptadas para serem utilizadas no ciclo de vida de um experimento científico baseado em simulação previamente definido por Mattoso *et al.* [5].

Para decidir quais práticas decritas na Seção 2.3 deveriam ser utilizadas na SciAgile foi consultado um especialista em *workflows* científicos e realizada uma análise conjunta pelos autores. Dentre as práticas anteriormente apresentadas foram selecionadas as seguintes: i, ii, iii, iv, v, vi, vii, viii, ix, x, xi, xii, xiii, xvi, xvii, xviii, xix, xxv, xxvi, xxviii, xxix, xxxi, xxxii, xxxv, xxxvi, xxxvii, xxxviii, xxxix. Algumas foram consideradas opcionais para a abordagem proposta, pois tem benefícios a longo prazo e em processos de tamanho pequeno podem não ser possíveis de serem aplicadas. As práticas iv, ix, xvi, xix, xxxi, xxxv e xxxviii envolvem estimativas e medição de melhorias que só podem ser feitas a longo prazo. No estudo experimental apresentado nesse artigo tais práticas não foram utilizadas. As práticas xiii, xxxii e xxxiii foram adaptadas. Na prática xiii os times deveriam "quebrar"as tarefas grandes do projeto em diversas pequenas tarefas tentando explicar a motivação daquela tarefa em formato de história. Nas práticas xxxii e xxxiii cada tarefa que pudesse possuir algum tipo de teste deveria de fato possuir tal teste, caso não tivesse passado no teste a tarefa não seria considerada pronta.

A visão geral da SciAgile é apresentada na Figura 2. As sub-seções a seguir descrevem as práticas incorporadas em cada uma das fases do ciclo de experimentação.

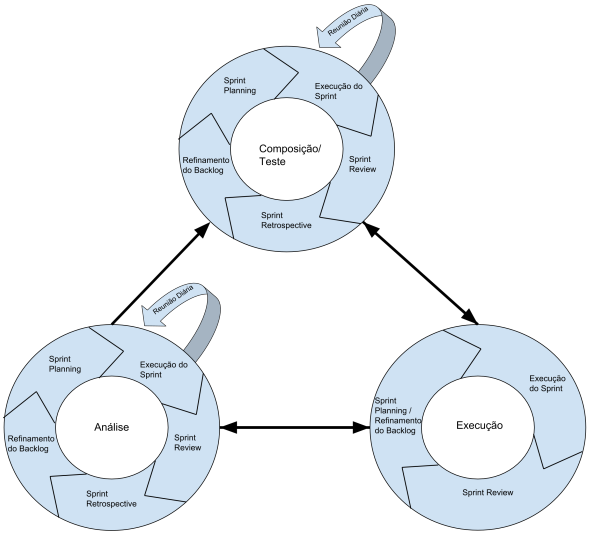


Figura 2. O Ciclo de Vida do Experimento segundo a SciAgile

3.1 Fase da Composição/Teste

A fase de composição contém as sub-fases de *Sprint Planning*, Execução do *Sprint*, *Sprint Review*, *Sprint Retrospective* e Refinamento do *Backlog*. Assim como nas outras fases do ciclo, a duração do *sprint* deve ser definida *a priori*. Em geral, a composição de experimentos é uma atividade incremental onde testes menores são realizados até que todo o experimento tenha sido modelado (*e.g.* várias versões de *workflows*). Dessa forma, é intuitivo que os *sprints* nessa fase sejam curtos (em torno de 1 semana), de forma que cada incremento do experimento seja associado a um *sprint* somente. Além disso, nessa fase é definida a ferramenta que será utilizada para modelar o *workflow* (no contexto desse artigo utilizamos o SGWfC SciCumulus). Além dissso, são definidos o limite de tarefas que podem ser feitas ao mesmo tempo, o *Scrum Master* e o *Product Owner*. O *Scrum Master* é o responsável pelo experimento (biólogo, astrônomo, *etc.*) enquanto o *Product Owner* é o coordenador do laboratório de pesquisa.

O processo de composição é iniciado pela etapa do refinamento do *Backlog*, que é uma reunião onde o time define quais tarefas devem ser incluídas no *Backlog*. Uma adaptação em relação ao ciclo originalmente proposto por Mattoso *et al.* na fase de composição é a incorporação de testes. Nessa fase, toda tarefa que é criada automaticamente gera uma tarefa de teste associada. Para definir um primeiro "rascunho"do experimento (também chamado de *workflow* conceitual) é realizada uma sessão de *brainstorm* buscando a forma mais simples de modelar o *workflow*. Após essa sessão de *brainstorm*, é iniciada a etapa *Sprint Planning*, onde o time define quais tarefas serão feitas ao decorrer do *sprint*, ou seja, quais tarefas irão para o *backlog* e que devem ser entregues ao final do *sprint*.

A etapa de execução do *sprint*, que é a mais longa e ocupa praticamente todo a duração do *sprint*, é onde membros do time se voluntariam para executar as tarefas que estão associadas com a especificação do *workflow*. Nesta etapa são realizadas reuniões diárias para verificar o andamento das tarefas que estão no *backlog* e o que cada membro do time realizou. Além disso, em tais reuniões diárias são esclarecidas as dúvidas com o *Product Owner*. Todos os membros do time podem alterar qualquer parte do *workflow* a qualquer momento. Após a execução do *sprint*, é executada a *Sprint Review*, que é uma reunião onde ocorre a inspeção do incremento que foi feito durante o *sprint*. Esta reunião dura cerca de 15 minutos. Em seguida é realizada a *Sprint Retrospective*, onde é feita uma análise do que pode ser aprendido com o *sprint* anterior e o que pode ser melhorado no processo como um todo. Após essa etapa, caso a composição do experimento ainda não tenha terminado, o ciclo se inicia novamente na sub-fase de Refinamento do *backlog*.

3.2 Fase da Execução

A fase de execução é aquela em que há menos sub-fases definidas, pois a execução de um *workflow* já definido é automática e pode demorar dias ou semanas. Dessa forma, a duração do *sprint* desta fase deve ser diferente da das demais fases do ciclo de vida. De fato, a duração do *sprint* dependerá do *workflow* que está sendo

executado. No estudo experimental deste artigo, os *workflows* modelados não são de larga escala, logo a duração do *workflow* foi definida como 1 dia.

A fase de execução se inicia com uma reunião que inclui a etapa Refinamento do *backlog* e *Sprint Planing*, para criar os casos de testes para o monitoramento e as tarefas de execução do *workflow* que serão alocados ao *sprint*. A sub-fase seguinte é a de Execução do *sprint* onde são efetivamente executadas as tarefas planejadas anteriormente. Após, é realizada a *Sprint Review* onde os resultados dos casos de teste são inspecionados assim como a validade da execução do *workflow*.

Apesar do *workflow* já ter sido modelado na fase de composição, são realizados testes para validar se a execução está de fato correta, para caso haja algum defeito que a torne inválida, a mesma seja interrompida e o experimento retornado para a fase de Composição. Esses testes são realizados, pois muitos *workflows* podem ser executados em ambientes de nuvem e podem levar a custos proibitivos, caso sejam realizados com defeitos.

3.3 Fase da Análise

A fase de Análise segue uma estrutura bem similar a fase de composição, porém o objetivo agora é estudar e analisar os resultados gerados pela fase de execução. Na fase da análise a duração do *sprint* é a mesma da fase de Composição. A fase de análise se inicia pela reunião de refinamento do *backlog* onde são definidas as tarefas a serem alocadas para o time. Em seguida é realizada a *Sprint planing* onde são definidas quais tarefas que se encontram no *backlog* serão alocadas ao *sprint*. Ao longo do *sprint* o time deve se voluntariar para executar as tarefas referentes à análise. Ao final do *sprint* é realizada uma reunião de *Sprint Review* para rever os dados analisados e uma reunião de *Sprint Retrospective* com objetivo de melhorar o processo. Uma vez finalizada a fase de análise, a equipe verifica se a hipótese foi confirmada ou refutade e pode ser necessário reiniciar todo o ciclo do experimento.

4 Estudo Experimental

A seguir apresentamos os detalhes do estudo experimental realizado, que segue os preceitos sugeridos por Wohlin *et al.* [18] para experimentos controlados.

4.1 Objetivo

Baseamos a definição do objetivo no *template* do método GQM (*Goal Question Metric*) [19], conforme apresentado a seguir: **Analisar** o uso da abordagem SciAgile em projetos científicos, quando comparada à condução *ad-hoc* desses projetos **Para o propósito de** caracterizar. **Com respeito a** tempo de execução, ganhos gerais, organização, comunicação, facilidade de uso e utilidade percebida. **Do ponto de vista dos** pesquisadores. **No do contexto de** Pesquisadores iniciantes conduzindo projetos científicos utilizando o SciAgile, quando comparado à condução do projeto sem o uso da abordagem.

Com base neste objetivo, buscamos responder as seguintes questões de pesquisa no contexto do processo científico: **Q1**: As práticas ágeis propostas são capazes de melhorar o tempo decorrido? **Q2**: De maneira geral, as práticas ágeis propostas são capazes de ajudar no processo? **Q2**: As práticas ágeis propostas são capazes de melhorar a organização do processo? **Q3**: As práticas ágeis propostas são capazes de melhorar a comunicação? **Q5**: As práticas ágeis propostas são consideradas fáceis de utilizar? **Q6**: As práticas ágeis propostas são consideradas úteis?

4.2 Planejamento do Experimento

O estudo experimental apresentado nessa seção foi projetado para ser realizada de forma *off-line*, pois não há o monitoramento constante de todas as atividades dos participantes envolvidos no experimento. Os participantes são estudantes de pós-graduação. Os problemas escolhidos são problemas científicos reais e a generalidade do estudo é específica.

As hipótese associadas com questão **Q1** encontram-se listadas abaixo. As demais questões serão respondidas com base em uma análise qualitativa da percepção dos participantes, sem testes de hipótese.

- $\mathbf{H_0}$: Não há ganho ou perda em tempo com a aplicação das práticas ágeis propostas
- $\mathbf{H_A}$: Há ganho em tempo com a aplicação das práticas ágeis propostas

Com base no objetivo e nas questões foi possível definir as variáveis independentes e dependentes do estudo experimental. As variáveis independentes são: (i) uso da abordagem e (ii) composição dos times. As variáveis dependentes são: (i) o tempo de desenvolvimento do projeto e (ii) a percepção dos participantes em relação a ganhos gerais, organização, comunicação, facilidade de uso e utilidade percebida.

O projeto foi organizado da seguinte forma: (i) um fator (abordagem utilizada) (ii) dois tratamentos (execução com uso da SciAgile e execução *ad-hoc*) e (iii) dois objetos de estudo (dois projetos científicos). Os participantes foram separados em 10 times. A formação dos times utilizou os princípios de *blocking* [18] (por nível de experiência), *random assignment* (os participantes de mesma experiência foram atribuídos aleatoriamente entre os times) e *balancing* (times de tamanhos próximos). O projeto para a condução do estudo experimental faz uso de um *crossed-design*.

O termo "Grupo"neste trabalho se refere a um conjunto de times de 3 ou 4 participantes. Antes da avaliação, foi aplicado um *workflow* "piloto"para todos os participantes envolvidos no estudo experimental. Assim, todos os participantes tiveram algum aprendizado sobre como realizar as tarefas de experimentação científica envolvidas no nosso estudo.

Na primeira etapa do estudo foi aplicado o *workflow* científico "A"para todos os times, onde o Grupo 1 contém 5 times que utilizaram a SciAgile e o Grupo 2 contém 5 times que utilizaram a abordagem *ad-hoc*. Na segunda etapa do estudo experimental foi aplicado o *workflow* científico "B"invertendo o tratamento aplicado aos grupos.

A instrumentação é composta pelos seguintes artefatos: (i) Termo de consentimento; (ii) Questionário de caracterização; (iii) Descrição dos *workflows* a serem realizados e (iv) Documento de Suporte à abordagem SciAgile e (v) Questionário de pós-avaliação para análise qualitativa. Os documentos utilizados na avaliação podem ser acessados em https://github.com/UFFeScience/SciAgile.

Os participantes devem representar uma população de cientistas envolvidos em um experimento científico. O questionário de caracterização deve ser entregue a cada um dos participantes para permitir identificar sua experiência prática relacionada a execução de tais *workflows* e sua formação acadêmica. Esse questionário é fundamental para a divisão equilibrada dos participantes entre os dez times de acordo com a sua experiência em: (i) Modelagem de *workflows* científicos, (ii) Análise de *workflows* científicos e (iii) experiência na utilização de métodos ágeis em projetos de qualquer área.

4.3 Ameaças à Validade do Experimento

As ameaças à validade identificadas, bem como ações para mitigar seu impacto sobre os resultados do estudo, são apresentadas nessa subseção.

Uma das ameaças à validade interna está relacionada ao aprendizado dos participantes. Para minimizar esse impacto, antes de iniciar o estudo experimental, um projeto piloto foi executado com a modelagem e execução de um *workflow* da área de bioinformática. Outra ameaça de validade interna é a relação dos resultados com a seleção dos participantes do estudo. Para evitar esta ameaça, o estudo envolve a divisão dos participantes em times equilibrados em relação à sua experiência. Uma ameaça à validade externa é o fato do estudo envolver estudantes e a representatividade dos projetos científicos selecionados. Para minimizar estas ameaças, os participantes foram divididos em 10 times diferentes e foram utilizados 2 *workflows* diferentes e representativos. Uma das possíveis ameaças à validade de construção é a interferência dos *workflows* escolhidos e da sua complexidade no resultado final. Para mitigar este problema, o experimento foi realizado utilizando o *crossed design*. Adicionalmente, cada tarefa foi bastante detalhada e os participantes puderam tirar as dúvidas ao longo do estudo experimental. Uma possível ameaça à validade de conclusão é o tamanho e a diversidade da amostra. Esta ameaça está relacionado com a utilização de uma amostra de 31 participantes, que é um número bastante razoável considerando que se trata do estudo de uma abordagem que cobre todo o ciclo de vida do processo científico.

4.4 Operação do Estudo Experimental

O estudo experimental foi conduzido com 31 participantes, todos estudantes de pós-graduação do Instituto de Computação da UFF. Os *workflows* tiveram a mesma quantidade de tempo para serem entregues. Ao todo a execução do estudo durou cerca de 3 meses. Os *workflows* que foram utilizados no estudo foram escolhidos da primeira e da terceira edições do *Provenance Challenge* (http://twiki.ipaw.info/). A primeira etapa do estudo experimental envolveu a implementação do *workflow* AIR que tem como objetivo criar *"brain atlases"*

a partir de imagens anatômicas de alta resolução. A segunda etapa envolveu a implementação do *workflow* Pan-STARRS que tem como objetivo mapear áreas do céu. Em cada etapa os times tiveram o mesmo prazo de 3 semanas para a implementação do *workflow*. Foi solicitado que todos os times medissem o tempo levado para a implementação de cada atividade do *workflow*. Durante a implementação dos *workflows* coletamos os dados quantitativos referente ao tempo gasto nos projetos e os participantes responderam um questionário de perguntas de natureza qualitativa.

4.5 Análise e Interpretação dos Resultados

Em relação à questão **Q1** (tempo), a Figura 3 apresenta o tempo necessário para cada time finalizar o projeto utilizando cada uma das abordagens SciAgile e *ad-hoc*. Os *workflows* são representados na Figura 3 pelas letras "A"(AIR) e "B"(Pan-STARRS). Diante destes resultados utilizamos a média para comparar o tempo entre os times. Podemos perceber que há um ganho de tempo (aprox. 23%) quando a SciAgile é utilizada. Ao realizar os testes estatísticos usando o método *Mann-Whitney* para os *workflows* "A"e "B"o resultado foi um *p-value* de 0.53174603 e 0.21031746, respectivamente e o nível de significância utilizado foi 0,05. Embora estes resultados não sejam estatisticamente significativos e sejam necessários mais dados (a tarefa precisava ser feita em time para avaliar características das práticas ágeis e contamos com somente 10 times), o ganho observado na média provê indícios preliminares de que há uma melhora no tempo utilizando o SciAgile. Entretanto, a partir desses resultados não podemos refutar a hipótese nula $\mathbf{H_0}$.

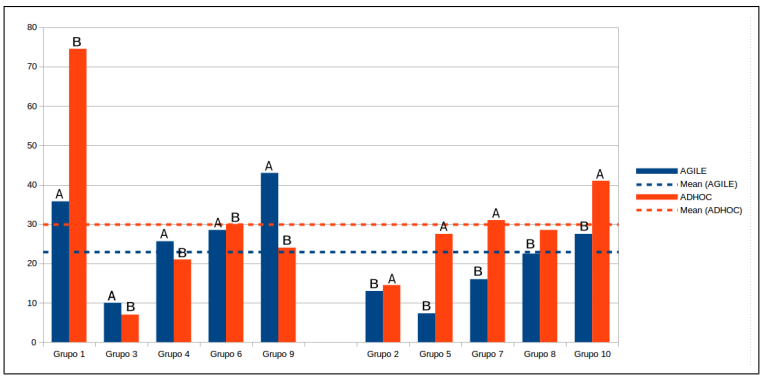
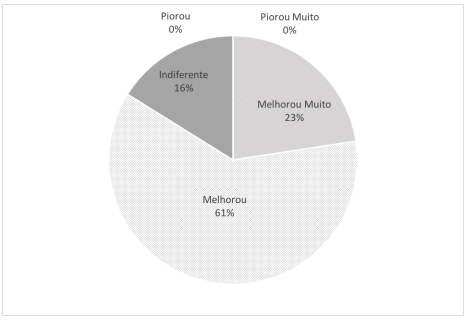
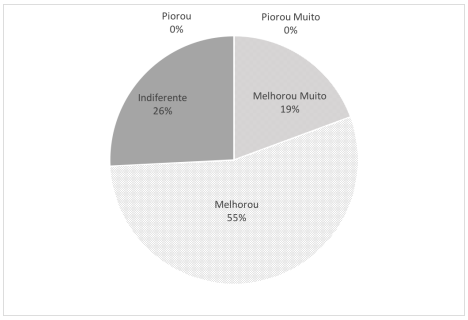


Figura 3. Tempo de execução do estudo experimental para cada workflow

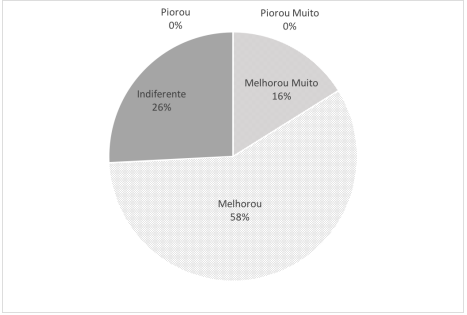
A Figura 4 e a Figura 5 apresentam as respostas qualitativas do questionário pós-tarefa. Em relação a questão de pesquisa **Q2** (melhoria no processo), as Figuras 4a, 4b, 4c e 4d permitem observar que a maioria dos participantes respondeu que o uso da SciAgile, de suas entregas incrementais (84%), sessões de revisão (74%), sessões de planejamento (74%) e de maneira geral (71%) melhorou ou melhorou muito o processo. O participante que mencionou que piorou



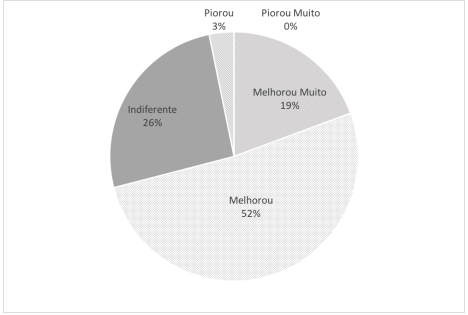
(a) O uso de entregas incrementais melhorou o processo?



(b) O uso de sessões de revisão melhorou o processo?



(c) O uso de sessões de planejamento melhorou o processo?



(d) A aplicação das práticas ágeis melhorou de forma geral o processo?

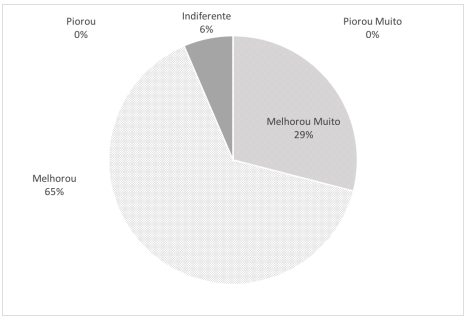
Figura 4. Percepção sobre a melhoria do processo científico

o processo alegou que não gosta de trabalhar em grupo e as reuniões não eram produtivas para ele.

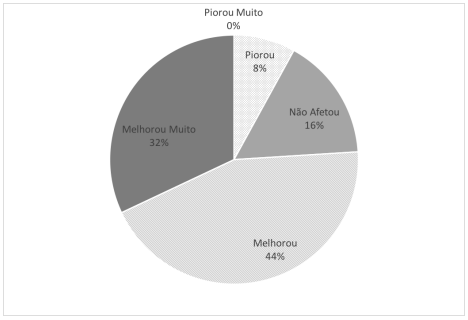
Em relação à questão **Q3** (melhoria da organização do projeto, Figura 5a), 94% dos participantes alegaram que o uso da SciAgile melhorou ou melhorou muito a organização dos participantes no projeto. Os participantes alegaram que o uso da SciAgile define tarefas exatas para cada membro em cada *sprint*, o que faz com que o grupo se organize melhor.

Em relação à questão **Q4** (comunicação do projeto, Figura 5b), 76% dos participantes alegaram que o uso da SciAgile melhorou ou melhorou muito a comunicação dos participantes. Os participantes alegaram que o uso da SciAgile define prazos bem definidos, além das tarefas de cada participante do grupo. O participante que alegou que houve uma piora com a SciAgile alegou que não gosta de trabalhar em grupo.

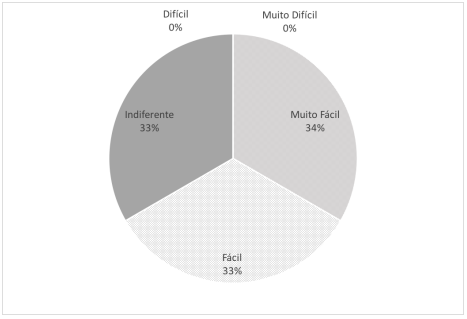
Em relação à questão **Q5** (simplicidade da utilização, Figura 5c), 67% dos participantes alegaram que o uso da SciAgile foi simples ou muito simples, enquanto que 33% alegaram que o uso foi indiferente. Os participantes alegaram que a SciAgile é uma abordagem simples e baseada em práticas ágeis já conhecidas, apoiadas por ferramentas simples e intuitivas.



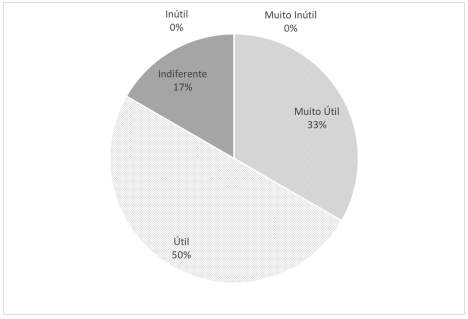
(a) A aplicação das práticas ágeis propostas melhorou a organização?



(b) A aplicação das práticas ágeis propostas melhorou a comunicação?



(c) A aplicação das práticas ágeis propostas foi simples?



(d) A aplicação das práticas ágeis propostas foi útil?

Figura 5. Percepção da Organização, Comunicação, Facilidade e Utilidade.

Por fim, em relação à questão **Q6** (utilidade percebida, Figura 5d), 83% dos participantes alegaram que o uso da SciAgile foi útil ou muito útil, enquanto que 17% alegaram que o uso foi indiferente. Os participantes alegaram que a SciAgile é uma abordagem útil, pois permite o acompanhamento da evolução do projeto com mais facilidade.

5 Considerações Finais

Esse artigo apresentou a SciAgile, uma abordagem que aplica práticas de metologias ágeis no ciclo de vida de um experimento científico baseado em simulação. Foi realizado um estudo experimental para avaliar a utilização de SciAgile em cenários reais. Foram escolhidos *workflows* considerados *benchmarks* e a avaliação apresentou resultados que fornecem indícios de que seu uso, quando comparado ao utilizar uma metodologia *ad-hoc*, pode fazer com que experimentos científicos sejam elaborados em menos tempo e com que sejam percebidas melhorias em seu planejamento, organização e grau de satisfação dos usuários. A disponibilização da SciAgile permite a sua utilização por organizações e parceiros de pesquisa. Trabalhos futuros incluem a realização de novos estudos, em maior escala e com

equipes maiores formadas por pesquisadores experientes em diferentes áreas do conhecimento, ajudando para reforçar a validade externa e de conclusão.

Referências

1. D. Méndez Fernández, S. Wagner, M. Kalinowski, et al., Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice, Empirical Software Engineering (doi:10.1007/s10664-016-9451-7) (2016) 1–41.
2. P. Abrahamsson, J. Warsta, M. T. Siponen, J. Ronkainen, New directions on agile methods: A comparative analysis, in: Proceedings of the 25th International Conference on Software Engineering, ICSE '03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 244–254.
3. B. Boehm, R. Turner, Management challenges to implementing agile processes in traditional development organizations, IEEE Softw. 22 (5) (2005) 30–39.
4. G. Theocharis, M. Kuhrmann, J. Münch, P. Diebold, Is water-scrum-fall reality? on the use of agile and traditional development practices, in: Proceedings of the 16th PROFES, Springer-Verlag New York, Inc., 2015, pp. 149–166.
5. M. Mattoso, C. Werner, G. H. Travassos, V. Braganholo, E. Ogasawara, D. Oliveira, et al., Towards supporting the life cycle of large scale scientific experiments, IJBPIM 5 (1) (2010) 79+.
6. W. Pereira, G. H. Travassos, Apoio na concepção de workflow científico abstrato para estudos in virtuo e in silico em engenharia de software, in: Proc. of the 2009 ESELAW, 2009, pp. 16–28.
7. D. W. Kane, M. M. Hohman, E. G. Cerami, M. W. McCormick, K. F. Kuhlman, J. A. Byrd, Agile methods in biomedical software development: a multi-site experience report, BMC Bioinformatics 7 (1) (2006) 273.
8. W. A. Wood, W. L. Kleb, Exploring xp for scientific research, IEEE software 20 (3) (2003) 30–36.
9. V. Maxville, Preparing scientists for scalable software development, in: Proc. of the Workshop on Soft. Eng. for Comp. Science and Eng., 2009, pp. 80–85.
10. K. S. Ackroyd, S. H. Kinder, G. R. Mant, M. C. Miller, C. A. Ramsdale, P. C. Stephenson, Scientific software development at a research facility, IEEE software 25 (4) (2008) 44.
11. K. Schwaber, M. Beedle, Agile Software Development with Scrum, 1st Edition, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.
12. K. Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.
13. J. Boeg, Priming Kanban: A 10 step guide to optimizing flow in your software delivery system, Trifork, 2011.
14. T. Hey, S. Tansley, K. Tolle (Eds.), The Fourth Paradigm: Data-Intensive Scientific Discovery, Microsoft Research, Redmond, Washington, 2009.
15. D. de Oliveira, E. Ogasawara, F. Baião, M. Mattoso, Scicumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows, in: 3rd International Conference on Cloud Computing, 2010, pp. 378–385.
16. M. T. Sletholt, J. E. Hannay, D. Pfahl, H. P. Langtangen, What do we know about scientific software development's agile practices?, Computing in Science Engineering 14 (2) (2012) 24–37. doi:10.1109/MCSE.2011.113.
17. A. Romeiro, D. de Oliveira, Um mapeamento sistemático sobre o uso de metodologias Ágeis no processo de experimentação científica, in: BreSci, 2014, pp. 378–385.
18. C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in software engineering, Springer Science & Business Media, 2012.
19. V. R. Basili, Software modeling and measurement: the goal/question/metric paradigm.