Práticas Ágeis para o Desenvolvimento de Software Científico

Francisco Alves

2015

Resumo

Conforme a ABNT NBR 6022:2003, o resumo é elemento obrigatório, constituído de uma sequência de frases concisas e objetivas e não de uma simples enumeração de tópicos, não ultrapassando 250 palavras, seguido, logo abaixo, das palavras representativas do conteúdo do trabalho, isto é, palavras-chave e/ou descritores, conforme a NBR 6028. (...) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto.

Palavras-chave: latex. abntex. editoração de texto.

Introdução

A evolução da infra-estrutura computacional disponível para pesquisadores nos mais diversos campos do conhecimento tem gerado o desenvolvimento de duas novas disciplinas intrinsicamente relacionadas que visam utilizar estes recursos para a resolução de problemas, quais sejam, a ciência computacional e a ciência dos dados.

De acordo com PITAC (2005) a ciência computacional pode ser entendida como um campo de estudos interdisciplinar que visa a utilização de recursos computacionais avançados para o entendimento e resolução de problemas complexos. Ao contrário da ciência da computação, que visa o estudo da computação a partir de uma perspectiva científica, o foco da ciência computacional é a resolução de problemas de outras disciplinas científicas, o foco é a aplicação.

Por sua vez, uma das definições mais aceitas de ciência dos dados provém de Cleveland (2014) que utilizou o termo para se referir a uma proposta de expansão das áreas de interesse da estatística com foco na análise de dados. Dentre as áreas de expansão sugeridas encontram-se computação com dados e avaliação de ferramentas.

Estes dois campos estão tão relacionados e vem ganhando tanta força, que tem sido chamados, respectivamente, de terceiro e quartos pilares da ciência, ao lado da teoria e da experimentação. Apesar posições contrárias a inclusão desses campos como pilares da ciência, como aquela de Vardi (2010), é possível encontrar no mínimo dois pontos de unanimidade: 1) a ciência computacional e a ciência dos dados estão proporcionando avanço científico, econômico e social a ponto de serem consideradas uma revolução; 2) cada vez mais pesquisadores irão precisar de habilidades computacionais para aproveitar os novos recursos a disposição para resolução de seus problemas.

Este trabalho irá se debruçar sobre uma das componentes dessa revolução que se faz presente tanto na ciência computacional quanto na ciência de dados. O desenvolvimento de software. Mais especificamente, o tema deste artigo é a utilização de práticas ágeis de engenharia de software por pesquisadores para o desenvolvimento de softwares científicos.

A justificativa para este trabalho deriva da importância destes novos campos para o avanço científico e da constatação de que pesquisadores gastam uma parcela cada vez maior do seu tempo desenvolvendo software sem o treinamento adequado. (WILSON et al., 2014). Isso gera pesquisadores autodidatas que não utilizam práticas de engenharia de software que já se tornaram *mainstream* na indústria. O conselho de Gentzkow e Shapiro (2014) é extremamente válido nesse contexto:

> If you are trying to solve a problem, and there are multi-billion dollar firms whose entire business model depends on solving the same problem, and there are whole courses at your university devoted to how to solve that problem, you might want to figure out what the experts do and see if you can't learn something from it. (GENTZKOW; SHAPIRO, 2014, pg. 5).

Além disso, o processo de desenvolvimento de software científico aparenta ter características similares a aquelas endereçadas pelo manifesto ágil, como responsividade a mudança e colaboração (SLETHOLT et al., 2012), tornando essas práticas especialmente interessantes de serem estudadas para fins de utilização por pesquisadores. Deste modo, o problema de pesquisa desta trabalho pode ser formulado como: Quais práticas ágeis de engenharia de software são adequadas as necessidades dos pesquisadores que precisam desenvolver software no âmbito de sua pesquisa?

O objetivo geral deste trabalho é identificar quais práticas ágeis de engenharia de software podem ser utilizadas por pesquisadores para o desenvolvimento de softwares científicos. Para tanto, podem ser listados como objetivos específicos deste trabalho:

- Documentar as práticas de engenharia de software caracterizadas pela agilidade

- Avaliar a similaridade entre o contexto de trabalho de engenheiros de software e pesquisadores

- Identificar quais práticas podem ser utilizadas com poucas adaptações

- Identificar práticas que não podem ser utilizadas tendo em vista a diferença entre os contextos

Pode-se encontrar na literatura especializada incontáveis e absolutamente diversas classificações que se aplicam a metodologia. No presente trabalho, quatro critérios de classificação serão adotados: aquele em relação a natureza, em relação aos objetivos gerais da pesquisa, em relação a abordagem empregada – se qualitativa ou quantitativa, e aquele em relação aos métodos empregados.

Em relação a natureza trata-se de uma pesquisa aplicada que visa gerar soluções para problemas específicos relacionados a aplicação de práticas de engenharia de software para a pesquisa científica. A abordagem aplicada será qualitativa caracterizada pelo aprofundamento nas questões subjetivas do fenômeno em detrimento da produção de medidas quantitativas. Quanto aos objetivos será uma pesquisa exploratória tendo em vista que seu objetivo é buscar familiaridade com problemas pouco conhecidos. Quanto aos procedimentos técnicos será uma pesquisa bibliográfica especialmente por meio de

artigos científicos que descrevam as práticas de engenharia de software e o contexto do desenvolvimento de softwares científicos.

Revisão da Literatura

Desenvolvimento de Software Científico

O objetivo desta seção é caracterizar o processo de desenvolvimento de software científico, com foco especial nas particularidades que existem em função do produto final (ie: software científico) e dos indivíduos envolvidos (ie: pesquisadores) neste processo.

O software científico pode ser entendido como aquele desenvolvido por pesquisadores com o objetivo de resolução de um problema em sua área de conhecimento de interesse. Apesar de simples, essa definição tem encontrado respaldo na literatura. Sletholt et al. (2012) define software científico como sendo aquele desenvolvido por cientistas e para cientistas, seja para a resolução de sistemas de equações matemáticas ou para geração de simulações. Heaton e Carver (2015) segue a mesma linha ao afirmar que software científico é aquele desenvolvido por cientistas e engenheiros para substituir ou expandir a experimentação física.

De maneira geral, um processo de desenvolvimento de software pode ser entendido como um conjunto de atividades relacionadas que levam a produção de um software. No caso da produção de software científico, a regra parece ser a ausência de um processo formal, com a realização de tarefas ad-hoc que geram o resultado desejado. Além da ausência de um processo formal, algumas das peculiaridades já documentadas na literatura dizem respeito ao levantamento de requisitos; a ausência de testes; a forma de relacionamento com o cliente; e ao treinamento recebido pelos responsáveis por desenvolver softwares científicos.

Em relação ao levantamento de requisitos, duas características principais se destacam. Em primeiro lugar, os requisitos de um software científico aparentam ser especialmente mais indefinidos no início de um projeto em comparação com softwares tradicionais. É usual que o problema de pesquisa de um cientista evolua ao longo de sua pesquisa, e, consequentemente, os requisitos acompanham essa transformação. Apesar de requisitos que emergem ao longo do tempo também existirem em contextos de softwares tradicionais, é razoável supor que ao menos o problema é mais bem definido e irá mudar menos em contextos tradicionais do que no científico. Além disso, uma segunda característica que explica as particularidades do levantamento de requisitos de softwares científicos deriva do fato de que os desenvolvedores, em grande medida cientistas, possuem vasto domínio de negócio. Isso implica que os requisitos podem ser levantados de forma extremamente genérica, tendo em vista que o indíviduo responsável pela implementação possui conhecimento suficiente para conceber o que deveria ser implementado em primeiro lugar. Nas palavras de um pesquisador que estava acostumado a desenvolver software e depois a especificar:

> So all I told one person [the developer] was: I want you to find a way of doing a ... fast graph matching problem, in an interface that is easy to use and shows you everything you need to know on the interface, and that's all I said. And he went away for a year and came back and here is the system. (SEGAL, 2008).

O processo de teste de software científico também apresenta diferenças marcantes em relação ao processo de teste de softwares tradicionais. Em primeiro lugar, uma das diferenças que será explorada adiante também ocorre no contexto de testes, qual seja,

pesquisadores não são, via de regra, treinados em como testar software. Além disso, a característica exploratória de diversos softwares científicos faz com que seja difícil definir, *a priori*, qual o comportamento esperado de um programa. Por fim, o teste de algoritmos númericos de estimação apresentam complexidades inerentes a forma como computadores armazenam números flutuantes. As passagens abaixo capturam a essência de testes em um ambiente de software científico:

> Testing is of the cursory nature which would enrage a software engineer ('does the software do what I expect it to do with inputs of the type I would expect to use? If I don't have any real expectations of the output, does the software behave in a way I really would not expect given inputs of the type I would expect to use?'). (SEGAL, 2008).

Em relação ao relacionamento com os clientes, o *modus operandi* tradicional de desenvolvimento de software científico é que um pesquisador que possui familiaridade com o desenvolvimento de software, provavelmente em virtude de contato durante a elaboração da sua tese de doutorado, é responsável por desenvolver os softwares de um grupo de pesquisa ou de um laboratório. Ou seja, o desenvolvedor não só possui conhecimento da área de negócio, ele trabalha diariamente com os demais clientes.

A última grande diferença é relativa ao treinamento (ou falta dele) que os responsáveis pelo desenvolvimento de software científico usualmente possuem. Enquanto no caso do desenvolvimento de software tradicionais os responsáveis usualmente possuem treinamento específico em ciência da computação e engenharia de software, a realidade de pesquisadores é que eles nunca são treinados em como implementar software. Wilson et al. (2014) observa que estudos recentes mostram que cientistas tipicamente gastam 30% ou mais do seu tempo desenvolvendo software mas 90% ou mais são autodidatas, e, portanto, nunca foram expostos a práticas básicas de engenharia de software como controle de versão, revisão de código, teste unitário e automação de tarefas.

Em relação ao treinamento de cientistas, merece destaque o Software Carpentry <https://software-carpentry.org/>, iniciativa que visa treinar cientistas em habilidades computacionais básicas que permitem um ganho de eficiência por parte de pesquisadores envolvidos em computação científica.

Práticas Ágeis de Desenvolvimento de Software

As características apresentadas na última seção objetivaram mostrar as particularidades do processo de desenvolvimento de software científico que devem ser reconhecidas por qualquer prática de engenharia de software que pretenda ser efetiva no referido contexto. O objetivo desta seção é apresentar um conjunto de práticas que aparentam se moldar especialmente bem as necessidades de pesquisadores. A essas deu-se o nome de práticas ágeis.

O movimento de agilidade teve início formal com a publicação do manifesto ágil em 2001. Na ocasião, um grupo de 17 profissionais se reuniram com o objetivo de encontrar similaridades entre os processos de desenvolvimento de software advogados por cada um dos participantes. Todos as metodologias se apresentavam como uma alternativa aos processos cascata de desenvolvimento bastantes difundidos na época. A integra do manifesto é

> We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

> Individuals and interactions over processes and tools Working software over comprehensive documentation Customer collaboration over contract negotiation Responding to change over following a plan
>
> That is, while there is value in the items on the right, we value the items on the left more.

Além do manifesto, também foram publicados doze princípios que dão sustenção aos valores expostos pelos agilistas. No entanto, o objetivo deste trabalho é debruçar-se sobre as práticas ágeis, entendidas como as atividades concretas realizadas no dia a dia para desenvolvimento e gestão de projetos de software de maneira consistente com o manifesto ágil. A justificativa para essa abordagem deriva do entendimento que pesquisadores não irão se interessar pelo movimento ágil como um todo a menos que sejam capazes de observar melhorias concretas no seu dia a dia advindas das práticas aqui expostas.

As práticas ágeis listadas abaixo foram pesquisadas no Guia de Práticas Ágeis <http://guide.agilealliance.org/> publicado pela *Agile Alliance*. Quando necessário, foram pesquisados outras referências bibliográficas para melhor caracterização da mesma, bem como a forma pela qual as especificidades do software científico são bem atendidas. As práticas foram agrupadas de acordo com as principais particularidades existentes no processo de desenvolvimento de software científico conforme identificado na seção , além de um tópico adicional destinado a agrupar práticas que podem ser consideradas como boas práticas em engenharia de software em geral não necessariamente associadas ao movimento ágil.

Requisitos

Backlog

O *backlog* é uma lista com itens que representa todo o trabalho que deve ou pode ser realizado em um dado projeto. Ele apresenta uma característica evolutiva por natureza, ao representar tudo o que o software pode vir a ser com base nas definições advindas dos stakeholders (clientes, usuários, desenvolvedores, etc). A característica evolutiva advém do fato de que a qualquer momento do tempo novos itens podem ser inseridos ou removidos do *backlog* com base em novas prioridades e requisitos. Não existe um formato físico pré-definido para o backlog e nem um nível de granularidade para os itens a ele pertencentes.

Design Simples

O termo *Design* Simples faz referência ao fato de que a estratégia de definição do *design* do software deve ser baseado no reconhecimento de que essa atividade é recorrente,

A team adopting the "simple design"practice bases its software design strategy on the following principles:

design is an ongoing activity, which includes refactoring and heuristics such as YAGNI

design quality is evaluated based on the rules of code simplicity

all design elements such as "design patterns", plugin-based architectures, etc. are seen as having costs as well as benefits, and design costs must be justified;

design decisions should be deferred until the "last responsible moment", so as to collect as much information as possible on the benefits of the chosen option before incurring its costs.

another common term is "emergent design", emphasizing that good global design can result from consistently paying attention to the local qualities of code structure (by analogy with the processes studied by complexity theorists where purely local rules reliably give rise to consistent global properties)

Testes

Testes Unitários

TDD

Relacionamento com Cliente

Time

Reunião Diária

Práticas Fundamentais

Controle de Versão

Desenvolvimento Iterativo e Incremental

Build Automatizado

Treinamento

Programação em Pares

Considerações finais

Este trabalho visou identificar práticas ágeis de engenharia de software que podem ser utilizadas por pesquisadores para o desenvolvimento de softwares científicos.

A caracterização do processo de desenvolvimento de software científico

Agile Practices for Scientific Software Development

Francisco Alves

2015

Abstract

According to ABNT NBR 6022:2003, an abstract in foreign language is a back matter mandatory element.

Keywords: latex. abntex.

Referências

CLEVELAND, W. S. Data science: An action plan for expanding the technical areas of the field of statistics. *Statistical Analysis and Data Mining*, v. 7, n. 6, p. 414–417, 2014. ISSN 1932-1872. Disponível em: <http://onlinelibrary.wiley.com/doi/10.1002/sam.11239/abstract>. Citado na página 1.

GENTZKOW, M.; SHAPIRO, J. M. *Code and Data for the Social Sciences: A Practitioner's Guide*. 2014. Disponível em: <http://faculty.chicagobooth.edu/matthew.gentzkow/research/CodeAndData.pdf>. Citado na página 2.

HEATON, D.; CARVER, J. C. Claims about the use of software engineering practices in science: A systematic literature review. *Information and Software Technology*, v. 67, p. 207–219, nov. 2015. ISSN 0950-5849. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0950584915001342>. Citado na página 3.

PITAC. *Computational Science: Ensuring America's Competitiveness*. 2005. Acesso em: 20 ago. 2015. Disponível em: <https://www.nitrd.gov/pitac/reports/20050609_computational/computational.pdf>. Citado na página 1.

SEGAL, J. Models of Scientific Software Development. In: *First International Workshop on Software Engineering for Computational Science and Engineering (SECSE08)*. [s.n.], 2008. Disponível em: <http://www.cse.msstate.edu/~SECSE08/Papers/Segal.pdf>. Citado 2 vezes nas páginas 3 e 4.

SLETHOLT, M. T. et al. What do we know about scientific software development's agile practices? *Computing in Science & Engineering*, v. 14, n. 2, p. 24–37, March-April 2012. Citado 2 vezes nas páginas 2 e 3.

VARDI, M. Y. Science has only two legs. *Communications of the ACM*, v. 53, n. 9, p. 5–5, 2010. ISSN 00010782. Disponível em: <http://portal.acm.org/citation.cfm?doid=1810891.1810892>. Citado na página 1.

WILSON, G. et al. Best Practices for Scientific Computing. *PLoS Biol*, v. 12, n. 1, p. e1001745, jan. 2014. Disponível em: <http://dx.doi.org/10.1371/journal.pbio.1001745>. Citado 2 vezes nas páginas 2 e 4.