Fichamento - TCC 1 - Lista de artigos explorados

Aluno: Gabriel Santana Barroso

Artigos fichados

- Marques, M. *et al*., "Enhancing the Student Learning Experience in Software Engineering Project Courses", IEEE Transactions on Education, vol. 61, no. 1, pp. 63-73, Feb. 2018, doi: 10.1109/TE.2017.2742989.
- Seifermann, S. *et al.*, "Data-Driven Software Architecture for Analyzing Confidentiality", IEEE International Conference on Software Architecture (ICSA), Hamburg, Germany, 2019, pp. 1-10, doi: 10.1109/ICSA.2019.00009.
- Fernandes, E.; Barcelos, T., "Identifying Success Factors in a Legacy Systems Reengineering Project Using Agile Methods", 10th Brazilian Workshop, WBMA 2019 Revised Selected Papers, pp. 101-110, Sep. 2019, doi:10.1007/978-3-030-36701-5_9.
- Poth, A.; *et al.*, "Lean and agile software process improvement in traditional and agile environments", Journal of Software: Evolution and Process. 2019. 31. e1986. doi: 10.1002/smr.1986.
- Chatley, R.; Field, T., "Lean Learning - Applying Lean Techniques to Improve Software Engineering Education", IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET), Buenos Aires, 2017, pp. 117-126, doi: 10.1109/ICSE-SEET.2017.5.
- Soares, M., "Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software", Revista Eletrônica de Sistemas de Informação. 2004. 3. doi: 10.21529/RESI.2004.0301006.
- Moralles, C. *et al.*, "On the Mapping of Underlying Concepts of a Combined Use of Lean and User-Centered Design with Agile Development: The Case Study of the Transformation Process of an IT Company", 10th Brazilian Workshop, WBMA 2019 Revised Selected Papers, pp. 25-40, Sep. 2019, doi: 10.1007/978-3-030-36701-5_3.
- Miranda, R. *et al.*, "Uma Análise do Impacto da Filosofia Ágil do Scrum no Sucesso de Projetos de Software". Anais do XXVII Workshop sobre Educação em Computação, (pp. 389-403). 2019. Porto Alegre: SBC. doi: 10.5753/wei.2019.6645.

Outros artigos explorados

- Abdul Rahman, N. *et al.*, "Lean Manufacturing Case Study with Kanban System Implementation". Procedia Economics and Finance. 2013. 7. 174–180. doi: 10.1016/S2212-5671(13)00232-3.
- Cawley, O. *et al.*, "Lean/Agile Software Development Methodologies in Regulated Environments - State of the Art". Lecture Notes in Business Information Processing. 2010. 65. doi: 10.1007/978-3-642-16416-3_4.

- Wang, X., "The Combination of Agile and Lean in Software Development: An Experience Report Analysis", 2011 Agile Conference, Salt Lake City, UT, 2011, pp. 1-9, doi: 10.1109/AGILE.2011.36.
- Perera, I.; Fernando, M.S.D., "Enhanced agile software development — hybrid paradigm with LEAN practice". 2007. pp. 239 - 244. doi: 10.1109/ICIINFS.2007.4579181.
- Rodríguez, P. *et al.*, "Combining Lean Thinking and Agile Methods for Software Development: A Case Study of a Finnish Provider of Wireless Embedded Systems Detailed", 2014 47th Hawaii International Conference on System Sciences, Waikoloa, HI, 2014, pp. 4770-4779, doi: 10.1109/HICSS.2014.586.
- Edison, H. *et al.*, "Lean Internal Startups for Software Product Innovation in Large Companies: Enablers and Inhibitors". Journal of Systems and Software. 2017. 135. doi: 10.1016/j.jss.2017.09.034.
- Ebert, C. *et al.*, "Lean Software Development". IEEE Softw. 29, 5 (September 2012), pp. 22–25. doi: 10.1109/MS.2012.116

Data-Driven Software Architecture for Analyzing Confidentiality

Seifermann, S.; Heinrich, R.; Reussner, R., "Data-Driven Software Architecture for Analyzing Confidentiality", IEEE International Conference on Software Architecture (ICSA), Hamburg, Germany, 2019, pp. 1-10, doi: 10.1109/ICSA.2019.00009.

1. Fichamento de Conteúdo

O artigo cunha o termo *Data-Driven Software Architecture* (DDSA, ou Arquitetura de Software Orientada a Dados, traduzido do inglês) e propõe uma abordagem diferente ao desenvolvimento de software, que possibilita melhorias de qualidade no âmbito da confidencialidade de dados. A abordagem defende que o comportamento do sistema em relação aos dados deve ser discutido e detalhado junto ao cliente desde a fase de projeto arquitetural, de modo a impactar positivamente todas as fases posteriores e garantir os requisitos de confidencialidade, mitigando gastos futuros com correções e problemas legais e melhorando a aceitação e a satisfação do cliente. A DDSA visa descrever os dados e seu processamento a nível arquitetural. As contribuições do artigo são um modelo para representar a DDSA e uma análise da conformidade da abordagem com os requisitos de confidencialidade. A avaliação da abordagem é feita por meio de estudos de caso, aplicando-a a dezesseis cenários diferentes, pertencentes a quatro classes de equivalência geralmente aplicáveis para análise de direitos de acesso baseados em papéis, dentre os quais quatorze cenários contêm violações de confidencialidade e dois não. A avaliação examina a acurácia da análise de confidencialidade. A análise detectou corretamente todos os problemas de confidencialidade, sem reportar falsos positivos. O benefício da abordagem é possibilitar que arquitetos de software conduzam análises de confidencialidade na fase de projeto, por meio de processamento de dados.

2. Fichamento Bibliográfico

- A análise de confidencialidade detecta um problema se uma operação de dados contendo um conjunto de papéis R_{op} acessar dados com direitos de acesso R_{data} de modo tal que $R_{op} \cap R_{data} = \{\}$. (página 8)
- Os resultados de modelar os cenários e analisá-los em busca de problemas de confidencialidade podem ser vistos na Tabela II. *Issues Detected* (problemas detectados) significa que a análise detectou um problema de confidencialidade nos cenários elencados. *No Issues Detected* (nenhum problema detectado) significa que a análise não detectou problema algum. (páginas 8 e 9)
- *Internal validity* (validade interna) garante que as relações causais são válidas, i.e. o fator que se espera que exerça influência é o único fator de influência. (página 9)
- *Construct validity* (validade de construção) garante que as métricas colhidas são apropriadas para responder às questões de pesquisa. (página 9)

- *Reliability* (confiabilidade) significa que os resultados não devem depender da condução dos pesquisadores na avaliação. (página 9)

3. Fichamento de Citações

- “Software architects need means for expressing and analyzing confidentiality to ensure compliance. Because of the complexity of modern systems, detecting confidentiality issues manually is not feasible.”
- “There is a wide range of approaches for analyzing confidentiality but most of them only target the implementation phase. The implementation is, however, not sufficient to introduce confidentiality in a cost-efficient way.”
- “There is a wide range of approaches addressing confidentiality. We focus on approaches that use structured system descriptions to carry out automated analyses as described in the survey of Nguyen et al. [10]. We favor automated analyses because they provide strong user guidance and usually require less experience to achieve satisfying results.”
- “Before the analysis, software architects define analysis goals in terms of the logic program or select predefined goals. The goals use characteristics of data that the analysis derives. In this paper, the goal is to match access right characteristics of data and role characteristics of data processing operations.”
- “The benefit of our approach is that software architects can conduct confidentiality analyses in the design phase by means of data processing. We expect this to be more straight forward to specify than control flow oriented descriptions and to be possible in an earlier stage because the required information is already available from the requirements engineering phase. In addition, the model serves as documentation that can be used for communication with other stakeholders.”

Enhancing the Student Learning Experience in Software Engineering Project Courses

Marques, M.; Ochoa, S. F.; Bastarrica, M. C.; Gutierrez, F. J., "Enhancing the Student Learning Experience in Software Engineering Project Courses", IEEE Transactions on Education, vol. 61, no. 1, pp. 63-73, Feb. 2018, doi: 10.1109/TE.2017.2742989.

1. Fichamento de Conteúdo

O artigo propõe um método de monitoramento de formação denominado *Reflexive Weekly Monitoring* (RWM, ou Monitoramento Semanal Reflexivo, traduzido do inglês), para cursos que envolvem processos de software disciplinados e trabalho ligeiramente acoplado, conjunto, com o objetivo de melhorar a experiência de aprendizado e de trabalho em equipe dos estudantes. O RWM usa de autorreflexão e práticas de aprendizado colaborativo para ajudar os estudantes a ter consciência de seu desempenho individual e coletivo (dentro da equipe). O método foi aplicado em um estudo de caso por nove semestres consecutivos e os resultados obtidos indicam que ele é efetivo para melhorar a experiência de aprendizado e de trabalho conjunto no cenário instrucional observado; os estudantes monitorados se demonstraram mais efetivos e coordenados e também desenvolveram um senso maior de pertencimento e satisfação. Todavia, isso não evidencia que atingiram desempenho ou produtividade superiores em relação aos demais estudantes. A relevância do estudo se dá pela falta de suporte ao desenvolvimento de “soft skills”, comum nesses cursos, como o de Ciência da Computação; os cursos aparelham os estudantes para se tornarem engenheiros ou cientistas com todo o fundamento técnico necessário, mas não desenvolvem tanto o fator humano que eles enfrentarão no dia a dia, em seus trabalhos.

2. Fichamento Bibliográfico

- *Coordination* (coordenação) é definida como a capacidade dos membros do time de trabalharem juntos para atingir as metas do projeto, e o *sense of belonging* (senso de pertencimento) como a percepção dos membros de que compartilham uma meta em comum. (página 64)
- *Effectiveness* (efetividade) é entendida como a capacidade de focar nos requisitos obrigatórios do projeto de software. (página 64)
- *Productivity* (produtividade) é medida pela quantidade de software útil construída durante o projeto. (página 64)
- O método RWM é liderado por monitores que realizam sessões de monitoramento semanais com um time de desenvolvimento de software para facilitar o entendimento dos membros acerca de seu próprio desempenho e status de projeto. O monitor não é um *coach*, gerente de projetos ou Scrum master. *Coaches* sugerem ações corretivas e melhorias, gerentes de projeto atribuem tarefas e Scrum masters lideram times na aplicação de práticas e ajudam na tomada de decisões; o monitor é um

agente que promove e facilita a autorreflexão pelos membros do time e usa isso para ajudá-los a encontrar suas próprias soluções para seus problemas individuais e de seus times. (página 66)

3. Fichamento de Citações

- "Project-based learning is probably the most valuable strategy for students and instructors, but is considered time-consuming and difficult to implement [8], [24]. When these instructional activities are implemented, students should take advantage of them, although this can represent a challenge for instructors."
- "COMPUTER science programs strive to prepare future software engineers for work in industry, by teaching students core computing concepts that will allow them to become lifelong learners, able to keep pace with innovations in the discipline. Approaches to delivering technical knowledge have usually been well adopted by universities. However, the development of transversal capabilities, such as leadership, teamwork, decision-making, negotiation, and self-reflection, are usually less supported in these programs [1]–[3]. These team's capabilities, also known as "soft skills", not only impact the teams' results but also their work climate [4], since software development also involves several human and social aspects [5]."
- "The results indicate that RWM helps student teams improve their coordination, sense of belonging to a team, and effectiveness, but not necessarily their productivity. Monitored teams tend to be more productive during the first half of the project, but this changes when they realize that they have the project under control. From that point on, students tend to be more speculative, probably adopting the so-called "apprentice attitude" [19], [20], which negatively impacts on the team productivity. This issue requires further research and is planned as part of the future work."
- "The evaluation results have both similarities with, and differences to prior findings reported in the literature. Similar to experiences using coaches [24], [28], [29] or Scrum masters to support student teams [40]–[42], RWM uses monitors to enhance students' learning experience during software projects. However, monitors do not guide nor examine students, but instead act as facilitators for self-reflection by team members. This probably caused the situation where no tension with monitored students was observed during the evaluation process."
- "The RWM method was designed for use in software engineering courses, but this does not prevent its use in other engineering project courses whose students work in teams using a disciplined process to solve a problem, although, the results shown in this study may not be representative of other scenarios."

Identifying Success Factors in a Legacy Systems Reengineering Project Using Agile Methods

Fernandes, E.; Barcelos, T., "Identifying Success Factors in a Legacy Systems Reengineering Project Using Agile Methods", 10th Brazilian Workshop, WBMA 2019 Revised Selected Papers, pp. 101-110, Sep. 2019, doi:10.1007/978-3-030-36701-5_9.

1. Fichamento de Conteúdo

O artigo busca compreender os fatores de sucesso que influenciam o processo de refatoração (reengenharia) de sistemas legado e mostrar como métodos ágeis podem influenciar os resultados. Os autores nos apresentam que custos envolvendo manutenção ou extensão, durante o ciclo de vida de um software, podem acabar excedendo o custo de sua reescrita, o que leva muitas empresas a optar por uma estratégia de reengenharia e que as práticas recomendadas para projetos de tal cunho compartilham muitas características em comum com as práticas ágeis, de acordo com a literatura. Foi usado como base para um estudo de caso um projeto real de uma empresa de desenvolvimento de software de São Paulo, em que fizeram a reengenharia de um sistema para uma aplicação SOA. Os resultados do projeto foram comparados com a percepção do time de desenvolvimento por meio de entrevistas semiestruturadas, com a análise dos artefatos do projeto e as melhores práticas propostas na literatura para entender se os resultados mencionados nesta última seriam confirmados pela prática. Os resultados obtidos reforçam a hipótese de que projetos de reengenharia podem ser mais bem sucedidos quando desenvolvidos sob metodologias ágeis.

2. Fichamento Bibliográfico

- O processo de reengenharia envolve compreender um sistema legado e reconstruir suas funcionalidades de modo a melhorar a qualidade de seus requisitos funcionais e não funcionais. Para atingir os resultados esperados, o sistema legado passa por processos de engenharia reversa da aplicação e reimplementações subsequentes de seus requisitos.
- Buscas na literatura encontraram variações do modelo *Plan, Do, Check and Act* (PDCA, ou Planejar, Verificar, Executar e Atuar) dentre modelos de desenvolvimento de software focados no processo de reengenharia. Esse processo funciona de forma iterativa e parte da compreensão do software legado, da projeção do software esperado, da realização de um estudo de viabilidade e depois passa por ciclos de escolhas e execução de migrações, implementação e técnicas de lançamento (*deploy*).
- Fatores de sucesso identificados por meio de revisão sistemática da literatura foram: competência e *expertise*, suporte executivo, motivação da equipe e do usuário, equipes pequenas, participação do usuário, práticas ágeis, priorização de entregas de funcionalidades chave, quantidade moderada de documentação, forte comunicação, treinamento técnico adequado, testes (unitário, integração, usabilidade etc), design simples, suporte de ferramentas, padrões de codificação bem definidos.

3. Fichamento de Citações

- “Legacy software systems are maintained as long as their maintenance and evolution costs outweigh the replacement or rewriting costs. The cost of maintaining or extending a system can increase due to several factors, but we can highlight architectural, documentation, design or granularity problems of its components [1, 2]. All of these factors over time can lead to a high cost of maintenance or impossibility to implement extensions and, therefore, lead to the decision to abandon the system, rewrite it or reengineer it.”
- “The reengineering process involves understanding a legacy system and redeploying its functionality in order to improve the quality of functional and non-functional requirements [1]. The reasons for adopting a system reengineering approach can be diverse as time/cost to create a new system, knowledge added to the existing product, adherence to the company’s business, among others [2]. Thus, the system reengineering process aims to rebuild the system in a new form to make its maintenance and extension costs more sustainable.”
- “Melo *et al.* [7] and Mazuco [8] identified through structured interviews the most widely used agile methods in the Brazilian software industry and listed Scrum and a mixed version of Scrum and XP as the most commonly used methodologies and daily meeting, unit testing, sprint planning, product backlog and release planning as the most adopted practices. In the mentioned works there is also consensus that the adoption of agile practices brings improvements such as increased productivity, better adaptability to changes, improved team communication and increased quality of delivered software.”
- “The case study was chosen as the research strategy; according to Wholin et al. [10], this strategy is suitable for studying phenomena in real contexts and also for confronting the obtained results with earlier studies or theories. The same authors argue that analysis of multiple sources of information is important to ensure the validity of the results. Hence, for project analysis, a triangulation strategy [11] was used to confront literature data, documental analysis of project artifacts (Jira, Git, etc.) and semi-structured interviews with those involved in the development phase.”
- “The reengineering project was considered successful from the customer’s point of view, as identified through testimonials given to the development team and mentioned in the interviews. The reengineered system provided better performance and more stability. From the development team’s point of view, the application now has better code quality and easier maintenance and development of new features. For the company, the developed product is aligned with its strategy and meets the needs of its customer. The project results helps to reinforce the hypothesis that reengineering projects can be more successful when developed using agile methodologies.”

Lean and agile software process improvement in traditional and agile environments

Poth, A.; Sasabe, S.; Mas, A.; Mesquida, A., (2018) "Lean and agile software process improvement in traditional and agile environments", Journal of Software: Evolution and Process. 31. e1986. doi: 10.1002/smr.1986.

1. Fichamento de Conteúdo

O artigo busca analisar as abordagens *Lean* e *Agile SPI* (*Software Process Improvement*) em ambientes de desenvolvimento tradicionais e ágeis e demonstrar que em ambos os cenários é possível buscar essas abordagens. Para tal, os autores retomam as raízes do *Lean* e passam por sua história, desde seu surgimento, que remonta à década de 1950, no Japão, com estudos sobre qualidade de produto. Em seguida, são pontuados os princípios chave do *Lean Software Development* e depois revisitam o Manifesto Ágil. Essa construção permite ao leitor perceber as conexões existentes e as convergências de ideias existentes desde o surgimento do *Lean* e seus processos de qualidade e de melhoria e seus princípios. Então, há uma análise de SPI em ambientes tradicionais e ambientes ágeis e são discutidas as abordagens para cada um dos ambientes. A conclusão obtida é de que depende do ambiente e de como a abordagem SPI ou *Lean* serão implementadas; toda uma análise deve ser feita anteriormente, buscando identificar em qual realidade dos tópicos anteriormente citados o ambiente de desenvolvimento da empresa e a cultura dela se encaixam melhor e até mesmo se é possível ou viável fazer transições para o modelo proposto pelo *Lean*, bastante diferente do tradicional.

2. Fichamento Bibliográfico

- Agilidade pode ser definida como a habilidade de criar e responder às mudanças para criar valor num ambiente de negócios turbulento. (página 4)
- Adaptabilidade é a característica chave de qualquer abordagem ágil, mais importante até mesmo que previsibilidade, que é a base da abordagem tradicional de software. (página 4)
- Algumas características que definem empresas tradicionais: estrutura fortemente hierarquizada, comunicação vertical, tomada de decisão por parte dos gerentes sem consultar os subordinados, falta de flexibilidade a mudanças. (página 4)

3. Fichamento de Citações

- “Lean principles are well established in different sectors such as lean manufacturing, logistics, or construction. Agile principles are mostly used in the IT sector. Agile software development methods have close relations with Lean principles. The possibility to integrate these Agile and Lean principles in process‐dependent (established) environments like medical devices, automotive, space, and other

industries where safety is a critical issue needs to be carefully analyzed. Moreover, and regarding the importance of people aspects in modern SPI approaches, the working culture and mindset of the people are changing over time with, for example, the entrance of millennials into the workforce."

- "High motivation of the function teams is one of the important success factors of the agile transition. People are often motivated when they have an opportunity to create their solution ideas by themselves and feel a successful experience with their solution. The function teams know about details of their responsible work at everyday workplace and understand possible solution ideas for improving their performance."
- "To apply the Lean/Agile approach in a traditional environment has been usual in recent times. However, companies have had to carry out an important transformation to adapt their way of working to this new approach. Even though there exist many guidelines and frameworks for agile adoption, organizations have problems with the selection of the most convenient method and with the general initiation of the Agile Transformation Process. Each organization as a whole, and each project being implemented in the organization, have different circumstances. Therefore, Agile Transformation Process is hard to standardize and offers a unique framework suitable for all the potential cases."

Lean Learning - Applying Lean Techniques to Improve Software Engineering Education

Chatley, R.; Field, T., "Lean Learning - Applying Lean Techniques to Improve Software Engineering Education", IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET), Buenos Aires, 2017, pp. 117-126, doi: 10.1109/ICSE-SEET.2017.5.

1. Fichamento de Conteúdo

O artigo relata um estudo de caso que exibe como foi construído um programa acadêmico no Imperial College London para reduzir a distância entre o que os estudantes aprendiam em sala e o que aprendiam e era demandado deles na realidade profissional, no mercado de trabalho (considerando a dificuldade de se construir um currículo base que acompanhe as práticas e exigências da indústria), de modo a prover aos estudantes habilidades e conhecimentos para carreiras industriais de engenharia de software. São dados detalhes da estrutura seguida e da evolução do programa, que é focado em ferramentas, técnicas e problemas que fazem parte do dia a dia de um desenvolvedor profissional atuando em um time moderno. O alinhamento dos métodos de ensino com os princípios do *lean software* possibilitou o fornecimento de experiências de aprendizado de alta qualidade. As contribuições do artigo são como lições aprendidas da implementação desse programa e podem servir como recomendações para outras instituições que estejam buscando evoluir seus métodos de ensino.

2. Fichamento Bibliográfico

- *Extreme Programming* (XP) é um dos métodos ágeis originais e inclui técnicas de gerenciamento de projetos bem como práticas técnicas para suportar a entrega ágil e confiável de software. (página 2)
- Scrum se concentra mais nos métodos de gerenciamento de projeto e não versa especificamente sobre como construir software. (página 2)
- Ambos XP and Scrum são focados em entregar software de forma iterativa e incremental em ciclos de duração igual e fixa. (página 2)
- Kanban é um outro método ágil, mais recente e inspirado pelo modelo de manufatura dos japoneses, especialmente o modelo toyotista, que preza por um fluxo contínuo de trabalho; é baseado nos princípios do *lean manufacturing*, focado em eliminar desperdícios (perdas) para aumentar taxa de entrega, apesar de não ser executado num ciclo de iterações com durações regulares. (página 2)
- *Continuous delivery* (entrega contínua) se trata de uma prática moderna e amplamente adotada, em que toda mudança individual a um fragmento de software deveria produzir um incremento de produto potencialmente empacotável e com o uso de automação, o tamanho desses pacotes de

mudança pode se tornar pequeno, de modo a reduzir o tempo de entrega sem perdas de rigor ou qualidade. (página 2)

3. Fichamento de Citações

- “In order to provide training in the types of software engineering methods and practices that are used in industrial development projects, many universities and other higher education institutions are striving to bring modern industrial software development techniques into the classroom. Keeping pace with rapid changes in industrial practice has required changes in the way software engineering is taught.”
- “Properly administered, the application of lean principles to content delivery and to assessment reduces the burden on both students and instructors, and we believe that at the same time this can enhance the learning experience.”
- “Classes in research-led universities are almost always taught by academics, but few academics have personal experience of developing software in an industrial environment. While many academics, particularly computer scientists, do write software as part of their research work, the way in which these development projects are carried out is normally not representative of the way that projects are run in industrial settings. ”
- “Over the past five years, we have refined our courses in a series of iterations. In order to discuss the approaches that we have tried, we borrow some vocabulary from [10]. This gives us three useful terms to describe different types of learning experience. The first is transmission, which describes the classic lecture situation. An expert holds a body of knowledge and tries to transmit it to a hopefully attentive audience. This is typically a one way interaction between one teacher and many learners. The second is apprenticeship, which refers to a learning experience focussed on the development of skills rather than theoretical knowledge, most likely through kinaesthetic learning and practical exercises. You can imagine this in a setting like a cookery class, where each student can practise a recipe repeatedly until they have mastered a dish. The third is developmental, which describes a personalised learning experience without a set curriculum. It focuses on taking the learner from where they are to somewhere more advanced, in a particular direction depending on their strengths and weaknesses. This sort of individual tuition works well in a situation like a piano lesson, but it is hard to replicate it with a lecture class of 150 students.”

Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software

Soares, M., "Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software", Revista Eletrônica de Sistemas de Informação. 2018. 3. doi: 10.21529/RESI.2004.0301006.

1. Fichamento de Conteúdo

O artigo busca analisar as diferenças entre o desenvolvimento de software com metodologias ágeis em relação ao desenvolvimento com metodologias tradicionais. Em particular, são apontadas as características mais marcantes e as práticas das metodologias ágeis denominadas *Extreme Programming* (XP) e Scrum. O artigo também contém comparações com as metodologias tradicionais (como a de modelo em Cascata), buscando enfatizar que as metodologias ágeis são baseadas em pessoas e não em processos e planejamentos. Posteriormente, são levantadas e apresentadas as principais vantagens e desvantagens do XP e do Scrum. Também são apresentados alguns resultados empíricos do uso de metodologias ágeis. O texto aborda também os conceitos do Manifesto Ágil e menciona rapidamente outras metodologias ágeis que surgiram com esse novo modelo de produção de software, apesar de não se aprofundar nelas. No contexto em que o artigo foi escrito, as metodologias ágeis ainda estavam "em sua infância", como o próprio autor diz, mas ele chega à conclusão de que elas já eram, desde então, efetivas, promissoras.

2. Fichamento Bibliográfico

- O termo "metodologias ágeis" se tornou popular em 2001, quando especialistas em processos de desenvolvimento de software representando diversos métodos (Scrum, XP, Crystal etc.) estabeleceram princípios comuns, compartilhados por todos esses métodos. O resultado foi a criação da "Aliança Ágil" e do "Manifesto Ágil" (*Agile Manifesto*). (página 3)
- Os conceitos chave do Manifesto Ágil são:

  > "Indivíduos e interações ao invés de processos e ferramentas.
  > Software executável ao invés de documentação.
  > Colaboração do cliente ao invés de negociação de contratos.
  > Respostas rápidas a mudanças ao invés de seguir planos." (página 3)
- Características marcantes do XP em relação às demais metodologias: *feedbacks* constantes, abordagem incremental, incentivo à comunicação interpessoal e à cooperação próxima.
- Características marcantes do Scrum em relação às demais metodologias: é organizado em ciclos iterativos de duração fixa (sprints), tem reuniões diárias de atualização de status da sprint, deve ser executado idealmente em times de até 10 pessoas. (página 5)

3. Fichamento de Citações

- “A maioria das metodologias ágeis nada possuem de novo [2]. O que as diferencia das metodologias tradicionais são o enfoque e os valores. A idéia das metodologias ágeis é o enfoque nas pessoas e não em processos ou algoritmos. Além disso, existe a preocupação de gastar menos tempo com documentação e mais com a implementação. Uma característica das metodologias ágeis é que elas são adaptativas ao invés de serem preditivas. Com isso, elas se adaptam a novos fatores decorrentes do desenvolvimento do projeto, ao invés de procurar analisar previamente tudo o que pode acontecer no decorrer do desenvolvimento.”
- “Para ser realmente considerada ágil a metodologia deve aceitar a mudança ao invés de tentar prever o futuro. O problema não é a mudança em si, mesmo porque ela ocorrerá de qualquer forma. O problema é como receber, avaliar e responder às mudanças. Enquanto as metodologias ágeis variam em termos de práticas e ênfases, elas compartilham algumas características, como desenvolvimento iterativo e incremental, comunicação e redução de produtos intermediários, como documentação extensiva. Desta forma existem maiores possibilidades de atender aos requisitos do cliente, que muitas vezes são mutáveis. Dentre as várias metodologias ágeis existentes, as mais conhecidas são a Extreme Programming [3] e a Scrum [4].”
- “A maioria das regras da XP causa polêmica à primeira vista e muitas não fazem sentido se aplicadas isoladamente. É a sinergia de seu conjunto que sustenta o sucesso de XP, encabeçando uma verdadeira revolução no desenvolvimento de software. A XP enfatiza o desenvolvimento rápido do projeto e visa garantir a satisfação do cliente, além de favorecer o cumprimento das estimativas. As regras, práticas e valores da XP proporcionam um agradável ambiente de desenvolvimento de software para os seus seguidores, que são conduzidos por quatro valores: comunicação, simplicidade, feedback e coragem [3]. A finalidade do princípio de comunicação é manter o melhor relacionamento possível entre clientes e desenvolvedores, preferindo conversas pessoais a outros meios de comunicação. A comunicação entre os desenvolvedores e o gerente do projeto também é encorajada. A simplicidade visa permitir a criação de código simples que não deve possuir funções desnecessárias. Por código simples entende-se implementar o software com o menor número possível de classes e métodos. Outra idéia importante da simplicidade é procurar implementar apenas requisitos atuais, evitando-se adicionar funcionalidades que podem ser importantes no futuro. A aposta da XP é que é melhor fazer algo simples hoje e pagar um pouco mais amanhã para fazer modificações necessárias do que implementar algo complicado hoje que talvez não venha a ser usado, sempre considerando que requisitos são mutáveis. A prática do feedback constante significa que o programador terá informações constantes do código e do cliente.”

On the Mapping of Underlying Concepts of a Combined Use of Lean and User-Centered Design with Agile Development: The Case Study of the Transformation Process of an IT Company

Moralles, C.; Vaccaro, M.; Zorzetti, M.; Pereira, E.; Trindade, C.; Prauchner, B.; Marczak, S.; Bastos, R., (2019) "On the Mapping of Underlying Concepts of a Combined Use of Lean and User-Centered Design with Agile Development: The Case Study of the Transformation Process of an IT Company", 10th Brazilian Workshop, WBMA 2019 Revised Selected Papers, pp. 25-40, Sep. 2019, doi: 10.1007/978-3-030-36701-5_3.

1. Fichamento de Conteúdo

O artigo investiga a convergência de três metodologias (*Lean*, *Agile Development* e *User-Centered Design*) para a formação de uma nova abordagem para desenvolvimento de software, gerando vantagem competitiva e aproximando e atendendo melhor aos usuários e clientes finais. A abordagem proposta pelos autores é nova e pouco explorada, portanto esse artigo cumpre a função de iniciar as investigações rumo ao desenvolvimento de um modelo de maturidade para acelerar a transição do uso de metodologias ágeis para o uso de abordagens combinadas. O objetivo do artigo é identificar quais são os conceitos subjacentes envolvidos no uso de desenvolvimento ágil, *lean* e design centrado no usuário e como podem ser combinados entre si para atingir um modelo único e consolidado de conceitos que poderão futuramente ser usados como esqueleto para a criação do modelo de maturidade anteriormente mencionado. Foram realizados múltiplos estudos de caso com dois times de desenvolvimento de software de uma empresa; suas práticas foram observadas e comparadas ao corpus teórico das três metodologias que são alvo do estudo.

2. Fichamento Bibliográfico

- *Lean* é uma metodologia que tem o objetivo de aumentar a qualidade e eliminar desperdícios dos processos produtivos, promovendo redução de custos, aumento de velocidade de produção e maior agregação de valor para o cliente. Aplicável a organizações diversos setores, vem sendo cada vez mais adotada. (página 28)
- *Extreme Programming* (XP) se trata de uma metodologia de desenvolvimento ágil de software, nascida nos Estados Unidos, na década de 90. Surgiu para possibilitar a criação de sistemas de melhor qualidade, que são produzidos em menos tempo e de forma mais econômica que o habitual. Suas entregas são cíclicas, incrementais e iterativas. Esses objetivos são alcançados graças a um sucinto conjunto de valores, princípios e práticas, que diferem da forma tradicional de se desenvolver software, tais como desenvolvimento orientado a testes, programação em pares e revisão de código. (página 28)

- *User-centered design* (UCD ou Design Centrado no Usuário) é um processo de design iterativo, em que os designers focam nos usuários e suas necessidades a cada fase do processo de design. Os times de design envolvem os usuários ao longo do processo de design por meio de uma variedade de pesquisas e técnicas de design, para que sejam capazes de criar produtos com alta usabilidade e acessibilidades para os usuários. (página 28)

3. Fichamento de Citações

- "We used 3 data sources: a questionnaire to collect the participants' profile (name, role, responsibilities, and time working in IT and at ORG); observations to learn about their day to day activities; and focus group sessions to gather information on their perceptions about the transformation, the training experience, the benefits and challenges of the Pivotal Labs approach; and to discuss the concept mapping between literature and what we observed them doing in practice."
- "Altogether, we performed six focus group sessions that lasted in average 1 hour with the 8 members that worked in the USA. Their profiles are shown in Table 1. Meetings were voice recorded and transcribed for thematic analysis [3,6,18]. Of those six meetings, we used two sessions for each approach. We first presented them the concepts from their practice in order to clarify whether we comprehended them correctly and then we presented the concepts from literature in order to identify the completeness of our observations from practice. By discussing the literature, team members could present us with concepts that we might have missed or misunderstood. We considered the work of Kent Beck [1,2] as literature for XP; Lean Startup [13] and Lean Software Development [12] for Lean; and the work of Norman [11], Brown [4], and Salah, Paige, and Cairns [15] for UCD. We based our definition of literature on existing Pivotal work and an initial observation of the teams."
- "From the comparison between the results from literature and our case study with the two teams that have been undergoing this transformation process for about 6 months, we found that the teams' use of Pivotal Labs is mostly aligned with the literature, but differs in some aspects, namely:
    - All decisions are based on experiments, disregarding the intuition of experts;
    - Lack of leaders, since the team inspires itself and shares decision making equally;
    - There is an Anchor role, that bridges the understanding between business and engineering;
    - Not all UCD techniques are used, but the teams are constantly seeking out to use new ones that might benefit their case."

Uma Análise do Impacto da Filosofia Ágil do Scrum no Sucesso de Projetos de Software

Miranda, R.; Araújo, D.; Portela, C.; Lopes, A., "Uma Análise do Impacto da Filosofia Ágil do Scrum no Sucesso de Projetos de Software". Anais do XXVII Workshop sobre Educação em Computação, (pp. 389-403). 2019. Porto Alegre: SBC. doi: 10.5753/wei.2019.6645.

1. Fichamento de Conteúdo

O artigo discute os impactos da filosofia das metodologias ágeis no sucesso de projetos de software, apresentando, no ambiente acadêmico, um olhar sobre os valores humanos e filosóficos envolvidos durante o trabalho em equipe no gerenciamento ágil de projetos e apresenta uma análise de aprendizagem de alunos em projetos de software acadêmico que seguem práticas e valores do Scrum. Particularmente, busca fazer uma análise qualitativa do impacto do comprometimento, comunicação e trabalho em equipe no resultado desses projetos de software através de um estudo de caso. Os resultados obtidos pela pesquisa, analisados e relatados com base nos pressupostos teóricos do manifesto ágil, elaborado por Beck *et al.* (2001), sugerem que a absorção desses valores pelas equipes impacta diretamente no seguimento das práticas do Scrum e consequentemente, no sucesso do projeto.

2. Fichamento Bibliográfico

- Desenvolvimento ágil é um termo utilizado para expressar um conjunto de métodos e práticas baseados nos valores e princípios contidos no Manifesto Ágil. Existem convergências entre estes métodos, como, por exemplo, organização em equipes pequenas, menos hierarquia, ciclos curtos, iterativos e incrementais, foco na entrega de valor, entre outros. O Scrum destaca-se porque abrange ser mais que uma metodologia, se caracterizando como um *framework*. (páginas 2 e 3)
- Valores adicionais do Scrum, como o trabalho em equipe, comprometimento e comunicação foram criados tendo como referência os valores do Manifesto Ágil. A equipe Scrum é pequena, desta forma, seus membros devem se comunicar frequentemente para buscar realizar as tarefas dentro do prazo. Trabalhar em equipe é fundamental para o sucesso do projeto; a responsabilidade sobre o trabalho não é individual, mas coletiva. (página 3)
- O comprometimento diz respeito a estar vinculado com o trabalho e com as pessoas da sua equipe por vontade própria, unidos em busca de uma meta compartilhada, estar empenhado em realizar as atividades e entregar os produtos de trabalho no prazo, atingindo estas metas. (página 3)
- Scrum não define práticas e passos a serem seguidos criteriosamente. Há a descrição de apenas três papéis (*Product Owner*, Equipe Scrum e *Scrum Master*), seis eventos (*Sprint* e reuniões de *Sprint Planning*, *Daily Scrum*, *Sprint Review* e *Sprint Retrospective*) e três artefatos (*Product Backlog*, *Sprint Backlog* e o Incremento do Produto). (página 3)

3. Fichamento de Citações

- “Esta pesquisa objetiva analisar a aprendizagem dos alunos em projetos de software acadêmico que seguem as práticas e valores do framework Scrum. Em particular, buscase analisar qualitativamente o impacto do comprometimento, comunicação e trabalho em equipe no resultado desses projetos de software através de um estudo de caso.”
- “A escolha do Scrum é baseada na definição de Schwaber e Sutherland (2013, p. 3), os quais caracteriza-o como um arcabouço de recursos e modelos para gerenciamento de projetos, podendo empregar outros processos ou técnicas, adaptando-o à realidade da equipe e/ou projeto. Estes projetos, nesse caso de estudo, possuem as limitações de experiência dos estudantes no contexto acadêmico e do tempo de duração das disciplinas.”
- “Pessoas são diferentes e interagir uma com as outras em sintonia para atingir um objetivo em comum requer grande experiência, motivação e orientação. Conhecer os valores e princípios filosóficos da metodologia ágil Scrum é diferente de assimilá-los e aplicá-los no dia a dia. Jeff Sutherland (2014), em seu livro “Scrum: A Arte de fazer o dobro do trabalho na metade do tempo”, apresenta o Scrum de uma forma mais abrangente, não somente focado na área de software, mas como ponto importante na parte de gestão de pessoas para a melhoria do processo como um todo.”
- “A imersão de metodologias de desenvolvimento ágil em ambientes acadêmicos vem sendo empregada como forma de estimular a difusão ampla no aprendizado em salas de aula, permitindo desta forma um aprofundamento maior nas disciplinas que trabalham com o tema, como Engenharia de Software (Silva et al. 2016).”
- “Os resultados parciais desta pesquisa sugerem que a absorção desses valores pelas equipes impacta diretamente no seguimento das práticas do Scrum e, consequentemente, no sucesso do projeto. Os valores trabalho em equipe, comprometimento e comunicação são cruciais e indissociáveis para o desenvolvimento do projeto e para a produção do valor de negócio, evidenciado pelo sucesso das equipes 3 e 4, com êxito na absorção desses valores, e pelo insucesso das equipes 1 e 2, com falha no processo de absorção.”
- “O caminho do sucesso em um projeto é sem dúvida o comprometimento. Uma equipe comunicativa é capaz de vencer obstáculos e as limitações individuais. O foco e felicidade é essencial e evidenciamos no tópico Papel, onde transpareceu as inquietudes de cada aluno. A falta de empatia e a necessidade de transpor o fracasso são os maiores danos que se pode ter dentro de uma equipe de projeto, evidenciado na pergunta sobre a contribuição dos demais membros para o resultado final, amenizado pelas perguntas auto avaliativas, onde confronta-se a realidade de aceitar o próprio fracasso. Não à toa, a metodologia ágil requer tempo, dedicação e experiência por parte da equipe e seus membros.”