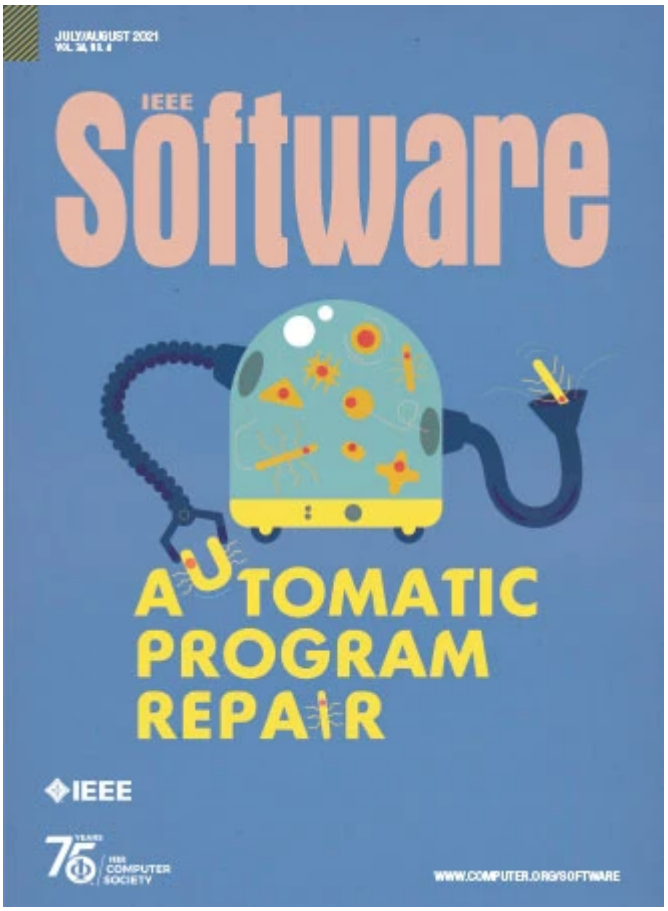
Grupo 6:

IGOR SERODIO BARCELAR
JOÃO PEDRO SILVA DE MELO
ODY JUNIOR DE MELO MENDES
SANDERSON CEZAR AUGUSTO JUNIOR
LUIZ ANTONIO CORREIA COSTA CARDOZO



Sumário

A Watershed Moment for Search-Based Software Engineering

Referência do artigo

I. Ozkaya, "A Watershed Moment for Search-Based Software Engineering" in *IEEE Software*, vol. 38, no. 04, pp. 3-6, 2021.

Resumo do artigo

Qual o contexto?

O artigo trata do assunto *"Search-Based Software Engineering"*, *Engenharia de Software Baseada em Busca*, e destaca a utilidade de sua aplicação, onde determinados problemas podem ser reformulados como problemas de otimização para enfim serem resolvidos utilizando técnicas algorítmicas de busca, de forma automatizada.

Qual o problema?

A problemática surge da necessidade de otimizar o processo de tarefas do cenário de engenharia de software. Geração de casos de teste, design, planejamento de *sprints* e refatoramento são atividades que se resumem a otimização, o que em outras palavras pode se dizer que tais tarefas não são tão diferentes de problemas de busca. No entanto, mostra-se que pontos como a participação do usuário e suas respectivas tarefas ao utilizar SBSE não podem ser esquecidos, afinal a intervenção humana é crucial para critérios de avaliação de algoritmos.

Qual a solução?

Apesar do problema ser um tanto complexo, exatamente por abordar questões que dizem respeito a como uma determinada situação pode e deve ser resolvida, os usos recentes de SBSE em testes automatizados têm-se tornado cada vez mais comuns. Uma das soluções sugeridas foi incorporar o feedback de desenvolvedores para melhorar a robustez e confiabilidade dessa ferramenta. Outra maneira de expandir a utilização do SBSE é resolver pequenos problemas frequentes no cenário de reparo de bugs e testes para abrir portas para a investigação de grandes problemas.

Como avaliou a solução proposta?

Como importante, pois o feedback dos desenvolvedores, tal como suas interações e preferências viria a ser crucial no processo de melhoria da performance do SBSE.

Pontos positivos

- O tema foi abordado de forma compreensível;
- Apesar de um assunto denso, o autor consegue claramente passar sua mensagem;
- As soluções são apresentadas de forma concisa.

Análise crítica

Apesar de já ter imaginado a relevância do tema, não esperava que ainda fosse motivo de tanta discussão, e que ainda há inúmeras dificuldades para tornar o seu uso algo costumeiro. O artigo é curto, mas completo, utilizando uma linguagem acessível para aqueles que são da área de tecnologia.

Agile Systems Engineering

Referência do artigo
C. Ebert and F. Kirschke-Biller, "Agile Systems Engineering" in *IEEE Software*, vol. 38, no. 04, pp. 7-15, 2021.

Resumo do artigo

Qual o contexto?

O artigo trata do assunto *"Agile Systems Engineering"*, *Engenharia de Sistemas Ágeis,* e destaca como o seu uso pode facilitar os processos de desenvolvimento.

Qual o problema?

A problemática surge da necessidade da junção do uso de metodologias ágeis na engenharia de sistemas. Apesar de técnicas ágeis serem essenciais para flexibilidade e eficiência, muitas empresas ao adotar metodologias ágeis focam exclusivamente no que irá ser entregue, e não no processo, o que de certa forma é uma má interpretação de tais metodologias. Consequentemente, os sistemas feitos se tornam demasiadamente complexos, dada a energia colocada ao se focar apenas no que será entregue.

Qual a solução?

Dada a necessidade de incorporar as metodologias ágeis na engenharia de sistemas, se é considerado durante o processo de desenvolvimento quatro dimensões a serem seguidas:

- Comercial: noção da visibilidade do valor comercial dos requisitos; foco na qualidade a fim de evitar imprevistos e retrabalho; planos de desenvolvimento e decisões aceleradas de design; técnicas de regressão sistemática para entregas parciais; reuso de software e hardware;
- Pessoal (*personnel*): modelos de colaboração e cooperação ágil dos times; gerenciamento de conhecimento; mapeamento das funções dos membros do time;
- Processo: metodologia SCRUM adaptada para engenharia de sistemas; consistência entre desenvolvimento e validação de atividades; desenvolvimento orientado por testes; integração contínua;
- Tecnologia: simulação de hardware e software; análise arquitetônica;

Uma vez que essas dimensões são consideradas, é garantido consistência e coerência, independente do produto, sempre facilitando as entregas (deliverables). A mescla da metodologia ágil com a engenharia de sistemas engloba todo o ciclo de vida do produto, não necessariamente apenas as atividades que envolvem desenvolvimento.

Como avaliou a solução proposta?

Necessária, pois a engenharia de sistemas ágeis suporta desenvolvimento contínuo dos requisitos até o momento do produto ser lançado, o que garante confiabilidade, transparência, eficiência e qualidade.

Pontos positivos

- O tema foi abordado de forma compreensível;
- Apesar de um assunto denso, o autor consegue claramente passar sua mensagem;
- As soluções são apresentadas de forma concisa.

Análise crítica

O autor mostra como líderes de times de software normalmente focam apenas no que é para ser entregue e se esquecem completamente do processo que irão utilizar, o que é uma péssima prática. Com o uso da engenharia de sistemas ágeis, flexibilidade e custos são constantemente aprovados e garantem ciclos com menos trabalho, menos retrabalho e um feedback mais rápido, o que é primordial para o sucesso do projeto.

Scalable Requirements: One Size Can Fit All

Referência do artigo
T. Gilb, "Scalable Requirements: One Size Can Fit All" in *IEEE Software*, vol. 38, no. 04, pp. 16-21, 2021.

Resumo do Artigo

Qual o contexto?

O chamado “avô da metodologia Ágil”, Tom Gilb, quer argumentar a favor da necessidade do uso de requisitos no planejamento de desenvolvimento de software atualmente, frente aos questionamentos de estudantes e profissionais. Não apenas a favor do seu uso, mas do uso correto e apropriado para a necessidade do projeto.

Qual o problema?

Na indústria de software são utilizados uma grande gama de métodos ou padrões para especificação de requisitos, mas muitos são empregados em situações que não são as ideais, como um padrão para projetos de pequena escala sendo usado em um projeto de larga escala e vice-versa. Pior, alguns padrões empregados não são ideais para nenhuma situação.

Qual a relevância do problema?

Requisitos são uma parte fundamental e amplamente utilizados em diversas metodologias de desenvolvimento de software na indústria hoje, com isso o levantamento correto e eficiente deles é extremamente importante para evitar atrasos e levar ao sucesso de um projeto de software independente do seu tamanho.

Qual a solução?

O uso da *Planguage*, uma linguagem de especificação de requisitos, resolve os problemas citados e ajuda os desenvolvedores a lidar com os requisitos por ela ser escalável, ou seja, apropriada para projetos de qualquer escala ou complexidade, e também por ser padronizada e adaptável para as necessidades de cada projeto.

Como avaliou a solução proposta?

Eu achei a linguagem interessante, principalmente por lidar com um problema existente na indústria atual. Muitos desenvolvedores não priorizam a fase de requisitos como deveriam, e a *Planguage* facilita e de certa forma guia o desenvolvedor a como realizar o levantamento de maneira correta. A linguagem ser grátis, escalável para qualquer projeto e ter muitas funcionalidades também são pontos importantes para estimular o seu uso.

Pontos Positivos

- Aborda um tema importante na indústria de software atual
- Propõe uma solução notável ao problema
- Explica muito bem a solução, seus usos e funcionamento

Análise crítica

O autor mostra bem o problema que empresas de desenvolvimento de software têm hoje com requisitos, e a consequência disso. Então ele passa a descrever a solução, que é a linguagem Planguage. Para mim é uma solução muito boa para o problema, e o texto explica bem como ela é boa e como usar ela bem. Eu creio que é um texto muito importante para qualquer desenvolvedor profissional dos dias atuais, mesmo se sua empresa não for adotar a linguagem.

Automatic Program Repair

Referência do artigo

C. Le Goues, M. Pradel, A. Roychoudhury and S. Chandra, "Automatic Program Repair" in *IEEE Software*, vol. 38, no. 04, pp. 22-27, 2021.

Resumo do artigo

Qual o contexto?

O artigo trata do assunto *“Automatic Program Repair", “Reparo automático de programas”,* e mostra recentes avanços nos estudos e na prática de ferramentas que auxiliam no reparo automático de software.

Qual o problema?

A problemática surge da necessidade da expansão do uso de ferramentas para o reparo automático de programas. O esforço feito por desenvolvedores buscando bugs e os consertando durante a fase de execução de projetos de software ainda é uma constante. A venda de softwares com defeitos, não importa o quão mínimo sejam, prejudica os negócios do desenvolvedor/empresa, consequentemente afastando possíveis clientes.

Encontrar bugs rapidamente é algo crucial para os desenvolvedores, especialmente em uma época em que a população depende continuamente do uso de software em quaisquer aspectos de suas vidas.

Qual a solução?

A comunidade responsável por pesquisas envolvendo ferramentas de reparo automático de programas tem se dedicado a este problema, especialmente na última década, sempre buscando novas técnicas para consertar bugs identificados por falhas em testes ou *crashes.* Não importa qual o método de encontrar bugs e utilizado, o objetivo, por sua vez solução, é encontrar *patches* que mudem o programa apropriadamente. Para alcançar tal objetivo, as tecnologias a serem utilizadas variam desde o aprendizado de máquina (*machine learning*) à computação evolucionária.

O fato de a maior parte do custo ser associado não com a parte criativa de um determinado projeto mas com o conserto de bugs é um problema, mas não é tão simples quanto se parece de corrigir. Às vezes, o aparecimento de um bug ou determinado comportamento inesperado de um software reside apenas na cabeça do desenvolvedor. O uso das ferramentas mencionadas alteraria o código fonte do software em si ao invés de procurar erros e saber onde eles estão, desta forma ajudando o desenvolvedor a evitar erros e o sistema a se recuperar dos mesmos.

Pontos positivos

- O tema foi abordado de forma compreensível;
- Apesar de um assunto denso, o autor consegue claramente passar sua mensagem;
- As soluções são apresentadas de forma concisa.

Análise crítica

Os autores mostram claramente como as ferramentas de reparo automático são úteis no desenvolvimento de software. Se a expansão destes estudos na área alcançarem níveis mais favoráveis dos que os que já demonstraram, o processo de desenvolvimento teria bem menos problemas para serem investigados e consequentemente mais lucro para o desenvolvedor/empresa em questão.

A Software-Repair Robot Based on Continual Learning

Referência do artigo

B. Baudry, *et al*.,"A Software-Repair Robot Based on Continual Learning" in *IEEE Software*, vol. 38, no. 04, pp. 28-35, 2021.

Resumo do artigo

Qual o contexto?

O artigo trata da utilização de sistemas baseados em treino contínuo para a correção de bugs no desenvolvimento de software.

Qual o problema?

O desenvolvimento de software é um processo complexo que lida com durantes erros e problemas durante seu ciclo de vida. Tais erros podem ter consequências que variam desde a insatisfação do cliente a situações catastróficas como a perda de vidas inocentes. Consertar esses erros manualmente é uma tarefa tediosa que demanda tempo e dinheiro. Como seria possível lidar com isso?

Qual a solução?

Os autores do artigo trazem suas experiências com a construção de um robô que faz o reparo de programas, chamado de *R-Hero*, que executa automaticamente o aprendizado de máquina. O *R-Hero* é um bot de reparo de programas baseado em aprendizado contínuo, com exclusividade para o reparo de erros que podem ser solucionados com a mudança de uma única linha de código. Como o código durante o desenvolvimento de qualquer sistema está sempre a evoluir, é natural que um paradigma como o aprendizado contínuo venha a ser utilizado. A performance geral do bot, ao utilizar o paradigma, evolui com o tempo porque as ferramentas de reparo nunca estacionam no tempo, dado os novos estudos realizados, novos patches e até mesmo problemas. Além disso, o paradigma tem a capacidade de aprender estratégias de reparo que ocorrem em prazos diferentes, de dias a meses e de meses a anos.

O *R-Hero* funciona de forma que armazena seu conhecimento em dois bancos de dados, compostos de patches sintetizados e escritos por humanos. Ele analisa eventos de um sistema de integração contínua, coletando as entregas que podem ter sido ou não um reparo de bug, mas apenas o indício de ter passado em todos os casos de teste é um dado valioso.

Apesar de ser uma ferramenta valiosa no reparo de bugs, o *R-Hero* possui algumas limitações, como o fato de ser exclusivo a mudanças em uma única linha; bugs complexos que necessitam de alteração em múltiplas linhas não podem ser consertados. Também, por tratar-se de um protótipo, ele foi capaz de solucionar 13 de 44002 builds, o que apesar da ideia ambiciosa, é ainda um número muito baixo. Felizmente, os estudos ainda avançam e talvez num futuro próximo mais bots sejam capazes de contribuir para o desenvolvimento de software.

Como avaliou a solução proposta?

Fundamental, pois o reparo de bugs é crucial para diversos fins de projeto, e uma vez utilizando ferramentas automáticas para realizar tais funções, os benefícios seriam inquestionáveis.

Pontos positivos

- O tema foi abordado de forma compreensível;
- Apesar de um assunto denso, o autor consegue claramente passar sua mensagem;
- As soluções são apresentadas de forma concisa.

Análise crítica

Os autores trouxeram uma ideia ambiciosa para a solução de bugs de software. No mais, apesar do pouco que demonstrou em sua capacidade, o *R-Hero* pode vir a ser uma ferramenta indispensável se aprimorada, ou até mesmo inspirar o surgimento de outras que tenham uma funcionalidade semelhante.

A Novel Approach for Search-Based Program Repair

Referência do artigo

L. Trujillo, O. Villanueva and D. Hernandez, "A Novel Approach For Search-Based Program Repair" in *IEEE Software*, vol. 38, no. 04, pp. 36-42, 2021.

Resumo do artigo

Qual o contexto?

O artigo *"A Novel Approach for Search-Based Program Repair", "Uma nova abordagem para reparo de programas baseados em busca",* trata de mudanças nas avaliações de *patches* de correção de bugs e falhas e seus possíveis benefícios ao expandir os seus respectivos espaços de solução.

Qual o problema?

A automação no processo de desenvolvimento de software se tornou um dos maiores desafios para a inteligência artificial e o aprendizado de máquina. Neste processo de automação, técnicas de busca são tecnologias fundamentais, também referenciadas como *SBSE(Search Based Software Engineering),* que incluem técnicas evolucionárias e programação genética. Um exemplo de sistema baseado em programação genética é o *Gen Prog,* que detecta bugs e busca por uma sequência de *patches*, os reparando. Porém, não é uma forma adequada de se aproximar do problema, uma vez que problemas reais podem acabar gerando confusões, e consequentemente erros de compilação sem a produção de um feedback próprio.

Qual a solução?

Uma maneira de superar estes problemas é através de uma solução que foque em qualidade, que fora denominada de *novelty search (NS),* que atua reduzindo o número de *patches* com problemas de compilação e reparando mais bugs. *NS* muda o estilo de busca focando no quanto uma solução é diferente quando se comparada com aquelas já encontradas na busca. Recentemente, a aplicação do *NS* tem sido feita em conjunto com *Gen Prog (GP)* para a resolução de problemas envolvendo aprendizado de máquina. Enquanto o *GP* avalia os casos de teste positivos e negativos, o *NS* avalia a singularidade de cada caso de teste, o que demonstrou ser uma ferramenta indispensável para já que o reparo de bugs foi maior do que aqueles feitos por outras ferramentas comuns. A combinação do *NS* com o *Gen Prog*, portanto, torna-se mais útil que o próprio *Gen Prog,* e mostra que não é o quanto uma exploração de bugs é feita, mas a maneira com que é feita.

Como avaliou a solução proposta?

Excelente, pois mostra como duas perspectivas de ferramentas diferentes, feitas em conjunto, proporcionam mais resultados do que apenas uma, que utiliza um único método.

Pontos positivos

- O tema foi abordado de forma compreensível;
- Apesar de um assunto denso, o autor consegue claramente passar sua mensagem;
- As soluções são apresentadas de forma concisa.

Análise crítica

Os autores mostram claramente como ainda existe a necessidade de aplicar uma certa robustez em ferramentas de reparo de bugs, mas priorizando o fato de explorar ferramentas já existentes, pois é possível que a combinação de duas ou mais que atuem de formas diferentes, com perspectivas diferentes, venha mostrar-se útil.

On the Introduction of Automatic Program Repair in Bloomberg

Referência do artigo
S. Kirbas, *et al*.,"On The Introduction of Automatic Program Repair in Bloomberg" in *IEEE Software*, vol. 38, no. 04, pp. 43-51, 2021.

Resumo do Artigo

Qual o contexto?

O texto fala que o reparo de *bugs* é uma das tarefas mais importantes no desenvolvimento de software é feita diariamente. Com isso, os autores querem mostrar que um sistema automático de reparo de bugs é fundamental para melhorar e auxiliar essa tarefa.

Qual o problema?

A tarefa de desenvolver software hoje é bem diferente do que era há 10 ou 20 anos atrás. Os sistemas estão cada vez mais complexos, com diversas partes interligadas e muitas vezes sendo desenvolvidos para dispositivos completamente diferentes, além dos sistemas que são considerados críticos em que falhas não podem existir. Frente a isso, a correção de eventuais *bugs* no código é mais significativa do que nunca, e por isso são desenvolvidos sistemas de reparo de *bugs* para sistematizar, melhorar e acelerar a fase de correção de *bugs*.

Qual a relevância do problema?

Sistemas de reparo de *bugs* são uma necessidade da indústria de desenvolvimento de software atual, pois mesmo em projetos de tamanho considerado médio a tarefa de encontrar e corrigir bugs pode ser imensa, e se não for realizada corretamente o projeto será um fracasso. Dessa forma, empresas têm um grande trabalho não apenas de fazer o reparo de *bugs* corretamente, mas de inicialmente escolher um sistema apropriado que possibilite que a tarefa seja com eficiência e precisão.

Qual a solução?

A utilização de um sistema automático de reparo de *bugs*, mais especificamente o Fixie. Esse sistema desenvolvido no Blumberg identifica e sugere como reparar *bugs* automaticamente com alta eficácia. O Fixie auxiliou bastante no desenvolvimento e aliviou os desenvolvedores de muitas horas gastas no trabalho de encontrar e corrigir *bugs*.

Como avaliou a solução proposta?

Eu achei uma solução muito boa, o artigo mostra como importante foi a contribuição do Fixie no desenvolvimento. O foco do sistema era encontrar os bugs mais simples e comuns, que mesmo com essas características continuamente tomavam tempo dos desenvolvedores. Com o Fixie, esses bugs comuns foram rapidamente detectados e os desenvolvedores puderam focar nos bugs mais complexos. Com isso considero que a solução foi um sucesso.

Pontos Positivos

- Aborda um tema importante na indústria de software atual
- Mostra uma solução nova que soluciona boa parte do problema
- Explica muito bem a solução, seus usos e benefícios

Análise crítica

O autor explica bem a importância da automação de reparos de *bugs*, e então dá o exemplo implementado que foi o Fixie. Pra mim tudo foi muito bem explicado, desde a importância de sistemas automáticos de reparos de bugs até a demonstração do sistema Fixie e os seus pontos positivos e negativos. Considerando tudo, para mim o sistema funciona muito bem e o artigo é uma ótima leitura para desenvolvedores.

AI in Software Engineering at Facebook

Referência do artigo

J. Bader, S. Seohyun Kim, F. Sifei Luan, S. Chandra and E. Meijer, "AI in Software Engineering at Facebook" in *IEEE Software*, vol. 38, no. 04, pp. 52-61, 2021.

Resumo do artigo

Qual o contexto?

O artigo discute práticas de Inteligência Artificial estudadas e realizadas por membros do Facebook com o propósito de facilitar o trabalho do desenvolvedor durante a escrita de código.

Qual o problema?

Qualquer desenvolvedor durante a etapa de desenvolvimento se questiona se o seu código está bem feito, se está funcionando ou se está faltando alguma coisa. É comum, durante esta situação, recorrer a sites com uma imensa quantidade de códigos disponíveis, como Github e StackOverflow, seja para encontrar trechos de código já prontos ou para pedir auxílio em seu próprio; claro, pode haver inúmeros resultados, mas supondo que ele ainda encontre o que pesquisou, tudo ocorre de forma favorável. Porém, se a primeira opção não for viável, ele recorre para a segunda e há as seguintes possibilidades:

- Ninguém responder a questão;
- Demorar horas ou dias para a resposta;

Tendo esses dois problemas, como seria possível solucioná-los para tornar o desenvolvimento mais eficiente?

Qual a solução?

Além dos problemas mostrados acima, o desenvolvedor corre o risco de encontrar dois códigos que solucionem seu problema, mas implementados de forma diferente; o seu uso depende do contexto em que for utilizado e se sua eficiência é consideravelmente melhor que o outro. O Facebook constatou que mesmo técnicas simples de aprendizado de máquina (*Machine Learning*) podem remover ineficiências durante a fase de desenvolvimento e evitar com que pesquisas para solucionar um determinado problema tomem boa parte do tempo do desenvolvedor.

A empresa surgiu com a ideia de uma ferramenta intitulada de *"Neural code search" (Busca de Código Neural),* que funciona focando trechos relevantes de códigos dado uma certa linguagem. No final da fase de avaliação, constatou-se que entre 287 questões, a ferramenta conseguiu passar 98 corretamente. Outra ferramenta utilizada chama-se *Aroma*, que retorna trechos de código pronto de diferentes formas, de maneira com que o desenvolvedor possa aprender diferentes estilos de programação.

A última ferramenta, chamada *Get a Fix,* tem um papel peculiar: aprender diferentes formas de solucionar bugs, baseando-se em soluções anteriores, como se fossem de fato construídas por mentes humanas, e as oferece para o desenvolvedor.

Desta forma, o uso dessas ferramentas não beneficia somente a escrita de código, mas todos os estágios do processo de desenvolvimento de software, salvando tempo e oferecendo de certa maneira um aprendizado para o desenvolvedor.

Como avaliou a solução proposta?

Relevante, pois se tais ferramentas alcançarem a maioria do público desenvolvedor, maiores resultados serão conquistados independente do projeto, grande ou pequeno porte, sendo realizado.

Pontos positivos

- O tema foi abordado de forma compreensível;
- Apesar de um assunto denso, o autor consegue claramente passar sua mensagem;
- As soluções são apresentadas de forma concisa.

Análise crítica

Os autores mostram uma vastidão de alternativas para o que antes eram práticas manuais. Quanto mais ferramentas de automação forem inseridas no cenário de desenvolvimento de software, mais benefícios serão recebidos, tanto pros desenvolvedores quanto para o cliente e público em geral.

Why and How Your Traceability Should Evolve

Referência do artigo
R. Wohlrab, P. Pelliccione, A. Shahrokni and E. Knauss, "Why and How Your Traceability Should Evolve: Insights From an Automotive Supplier" in *IEEE Software*, vol. 38, no. 04, pp. 62-70, 2021.

Resumo do Artigo

Qual o contexto?

A maioria dos sistemas de gestão de rastreabilidade não evoluem conforme o progresso do projeto, dessa forma os autores querem mostrar como ela deve e pode acompanhar esse progresso em vista de como o desenvolvimento é feito atualmente, de maneira incremental e dinâmica.

Qual o problema?

A gestão de rastreabilidade de artefatos em projetos de software é uma necessidade no desenvolvimento de software hoje, para o controle e gestão do código e dos requisitos. Por isso é fundamental ter um sistema capaz de fazer essa gestão, a questão é que de acordo com o artigo, a maioria destes sistemas utilizados atualmente não evoluem conforme o progresso do projeto, um grande problema uma vez que o desenvolvimento de software é geralmente feito de maneira incremental e dinâmica.

Qual a relevância do problema?

O problema da gestão de rastreabilidade é altamente relevante para o desenvolvimento de software moderno nas mais diversas escalas, pois faz parte dos padrões e metodologias mais utilizadas para garantir o sucesso de projetos de software.

Qual a solução?

Maior flexibilidade e capacidade de evolução do sistema de gestão de rastreabilidade. Para este fim, o autor propõe um ciclo de 4 passos: 1 - levantamento de requisitos e reuniões de planejamento, 2 - avaliar os dados levantados, 3 - design dos artefatos, 4 - implementação do sistema.

Como avaliou a solução proposta?

Eu achei a solução boa, o sistema é bem elaborado e deve resultar em maior chance de sucesso dos projetos que o utilizarem.

Pontos Positivos

- Aborda um tema importante na indústria de software atual
- Detalha um bom sistema para resolver o problema
- Explica muito bem a solução, seus usos e funcionamento

Análise crítica

O autor mostra bem o problema que empresas de desenvolvimento de software têm hoje com gestão de rastreabilidade e os problemas que uma má gestão pode causar. Então ele passa a descrever a solução, que é utilizar um sistema de gestão de rastreabilidade capaz de evoluir com o projeto. Para mim é uma solução muito boa para o problema, e o texto explica bem como ela é boa e como usar ela bem. Eu creio que é um texto interessante para qualquer desenvolvedor.

A Hitchhiker’s Guide to Model-Driven Engineering for Data-Centric Systems

Referência do artigo

B. Combemale, *et al*.,"A Hitchhiker's Guide to Model-Driven Engineering for Data-Centric Systems" in *IEEE Software*, vol. 38, no. 04, pp. 71-84, 2021.

Resumo do artigo

Qual o contexto?

O artigo trata de enfatizar e ilustrar a complementaridade e dualidade de modelos e dados em sistemas sociotécnicos, desmistificando formas de como ambos podem ser utilizados sinergicamente com o framework *MODA (Models and Data)*.

Qual o problema?

Um amplo espectro dos domínios de aplicação faz uso de largos e heterogêneos volumes de dados. O sucesso recente da inteligência artificial, e particularmente do aprendizado de máquina, amplifica a relevância de dados no desenvolvimento, manutenção, evolução e gerenciamento de execução em sistemas construídos com técnicas de engenharia orientada a modelos. As aplicações envolvidas são normalmente relacionadas a transporte público, saúde e sistemas de emergência, que em outras palavras podem ser consideradas sociotécnicas. Como seria possível integrar modelos heterogêneos e seus respectivos dados durante todo o ciclo de vida de sistemas sociotécnicos?

Qual a solução?

A solução proposta seria utilizar um framework denominado *MODA (models and data),* que suporta o ciclo de vida de um sistema sociotécnico e tem o propósito de lidar com um um espectro enorme de stakeholders e outros grupos. O framework funciona de forma que provê uma visão de como integrar as funções desempenhadas pelos modelos (prescritivos, preditivos, descritivos), enfatizando as ações para integrá-los.

Utilizando o framework *MODA*, espera-se clarificar as respectivas funções de modelos e fonte de dados, organizar e comparar processos de engenharia e tecnologias para suportar escolhas críticas envolvendo grandes sistemas e, por fim, iniciar novas pesquisas com o propósito de integrar diferentes modelos e fontes de dados. Os dados processados pelo framework passam pelos modelos prescritivos (documentam o sistema a ser construído), preditivos (prevê informações que não podem ser medidas) e descritivos (documentam aspectos passados ou presentes do sistema em estudo) para adaptar o sistema a lidar com dados em evolução.

Como avaliou a solução proposta?

Importante, pois o framework pode ser utilizado para organizar, explicar e comparar processos complexos de engenharia, desenvolvimento de software e ciclos de vida de sistemas, enquanto criando pesquisas para solucionar maiores problemas.

Pontos negativos

- O tema foi algo um tanto complexo de se entender, dado a maneira com que o artigo foi estruturado;
- Solução dada de uma maneira confusa;

Análise crítica

O artigo mostra como um framework pode ser utilizado para integrar dados e modelos em sistemas sociotécnicos. No mais, a maneira com que as informações são passadas deixam a desejar e tornam a leitura um tanto confusa.

Applying Emotional Team Coaching to Software Development

Referência do artigo
E. Marcos, R. Hens, T. Puebla and J. Vara, "Applying Emotional Team Coaching to Software Development" in *IEEE Software*, vol. 38, no. 04, pp. 85-93, 2021.

Resumo do Artigo

Qual o contexto?

O mundo do desenvolvimento de *software*, é por muitas vezes retratado como solitário e socialmente reservado, por isso, a autora do artigo propõe um dos campos de conhecimento externo que pode ser incluído num time é o da psicologia.

Qual o problema?

Foi observado pela equipe autora do estudo que muitas vezes a falta de uma interação social feita de maneira correta pode levar a atrasos no desenvolvimento.

Qual a relevância do problema?

Alta relevância, pois uma equipe mais motivada e melhor engajada é melhor não só para o bem do projeto como também para a saúde mental dos programadores envolvidos.

Qual a solução?

A solução apresentada pela equipe autora é a adição de mais um integrante na equipe, um "*Coach* emocional", para auxiliar com a motivação e a comunicação interpessoal, minimizando falhas de comunicação e possíveis atritos entre os membros do time.

Como avaliou a solução proposta?

É inovativo, embora financeiramente taxativo, a adição de uma pessoa em específico para estabelecer canais de comunicação claros e prezar pela saúde mental dos envolvidos no projeto.

Pontos Positivos

- Promove um ambiente de trabalho mais saudável e produtivo;
- É menos estressante saber que possíveis conflitos e falhas na comunicação podem ser analisadas de um ponto de vista externo;
- Diminui a pressão no *Scrum Master* para a gerência do time;

Análise crítica

A mente humana é uma coisa bastante complexa e no contexto de desenvolvimento de *software* ter um profissional no time para ajudar a motivar e guiar os esforços em conjunto com o *Scrum Master* pode ser uma ótima ferramenta na direção correta em criar um ambiente de desenvolvimento saudável.

Choosing a Chatbot Development Tool

Referência do artigo

S. Perez-Soler, S. Juarez-Puerta, E. Guerra and J. de Lara, "Choosing a Chatbot Development Tool" in *IEEE Software*, vol. 38, no. 04, pp. 94-103, 2021.

Resumo do artigo

Qual o contexto?

O artigo foca em avaliar programas *chatbots* e discuti-los, focando em seus aspectos técnicos e gerenciais.

Qual o problema?

Chatbots são interfaces de usuário conversativas que tem se tornado populares nos últimos anos, dado suas funcionalidades de acesso a diferentes tipos de serviços. Porém, escolher uma ferramenta específica para o desenvolvimento de *chatbot* é algo difícil. Quais aspectos deveriam ser considerados no design de um *chatbot*?

Qual a solução?

Ao invés de considerar aspectos funcionais, de integração, análise ou garantia de qualidade, os autores trabalham em focar em fatores técnicos como *"quer-se acessar as nossas informações em inglês e espanhol",* e gerenciais como *"os desenvolvedores não sabem criar chatbots"* para ajudar desenvolvedores e gerentes a escolher as melhores ferramentas.

São consideradas ferramentas que, durante a entrada *(input)*, sejam capazes de reconhecer frases de entrada esperadas, utilizando expressões regulares ou padrões definidos, através de *NLP (Natural Language Processing),* um sub ramo da inteligência artificial capaz de entender as línguas humanas naturais. Soluções baseadas em NLP podem identificar certos parâmetros em frases de entrada, diferentes tipos de linguagem (inglês, espanhol) e até mesmo entradas de voz. Outro ponto importante é a organização conversativa da ferramenta: maioria das plataformas e frameworks salvam valores de conversas no sistema para usos futuros, mas aquelas que conseguem identificar padrões entre frases para criar uma nova, de forma concisa, também são indispensáveis. Por fim, um dos aspectos mais importantes a serem considerados é a segurança, especialmente se o chatbot lida com os dados pessoais dos usuários. Apesar de ser responsabilidade do desenvolvedor, existem ferramentas específicas que podem adicionar uma camada extra de segurança.

De uma perspectiva gerencial, os fatores a serem considerados normalmente dizem respeito ao preço da ferramenta, qual sua utilidade, quais idiomas a ferramenta pode trabalhar e se novas versões do mesmo chatbot podem surgir. Quanto à expansão de idiomas, existem ferramentas que graças ao NLP podem aprender novas línguas treinando através de pequenas frases.

Como avaliou a solução proposta?

Relevante, pois a escolha de uma determinada ferramenta implica na satisfação do cliente e na reputação de uma determinada empresa.

Pontos positivos

- O tema foi abordado de forma compreensível;
- Apesar de um assunto denso, o autor consegue claramente passar sua mensagem;
- As soluções são apresentadas de forma concisa.

Análise crítica

É um tema interessante o qual eu não tinha tanta proximidade, portanto não fazia ideia de como funcionava a elicitação de requisitos para o funcionamento de um determinado chatbot. Em grandes empresas, é crucial que determinados aspectos sejam incluídos no chatbot para evitar futuros problemas.

Rapid Yet Robust Continuous Delivery of Software for Disaster Management Scenarios

Referência do artigo

R. Guntha, S. Rao, H. Muccini and M. Vinodini Ramesh, "Rapid Yet Robust Continuous Delivery of Software for Disaster Management Scenarios" in *IEEE Software*, vol. 38, no. 04, pp. 104-113, 2021.

Resumo do Artigo

Qual o contexto?

O artigo aborda a criação de um novo sistema ágil inspirado no modelo DevOps, o R-DevOps, o R significando Rapid ou Rápido em potuguês, com o objetivo de desenvolver um aplicativo o mais rápido possível para salvar vidas durante uma enchente na Índia.

Qual o problema?

Como as linhas de ajuda do governo estavam frequentemente ocupadas, os operadores dos "call-centers" não conseguiam conectar facilmente as vítimas aos socorristas. O exército, por sua vez, só aceitava pedidos de resgate com coordenadas de GPS e muitas das vítimas precisavam de orientação detalhada para obter sua localização. A falta de fluxo de informações precisas e oportunas entre as vítimas e as agências de socorro fez com que surgisse a necessidade de uma solução em software, visto que não foi encontrado nenhum app público que atendesse a demanda. Outro problema encontrado é que se o desenvolvimento natural (DevOps) fosse usado, o aplicativo poderia demorar meses para ser entregue.

Qual a relevância do problema?

Desastres naturais são imprevisíveis e implacáveis, logo ter um aplicativo que liga facilmente as vítimas de tal desastre com os prestadores de socorro é algo muito importante e relevante, principalmente para países e regiões que sofrem constantemente com a fúria da natureza.

Qual a solução?

A solução usada para entregar um aplicativo o mais rápido possível, com a menor quantidade de bugs e com uma interface amigável ao usuário foi criado e usado um novo método de desenvolvimento de software, o R-DevOps, onde o objetivo principal é ter o aplicativo rodando.

Para atingir esse objetivo, foi invertida a lógica de “Testar para Lançar” para “Lançar para Testar”, com o aplicativo sendo lançado na Google Play, no auge de desenvolvimento, chegaram a fazer duas atualizações por dia. O monitoramento do aplicativo após o lançamento das atualizações passa a ser a prioridade número um e os ciclos de trabalho ou sprints passam a ser extremamente curtas, horas em alguns casos.

Como avaliou a solução proposta?

Como inovativa, tendo em vista que sempre deve haver melhoramentos na maneira de desenvolver software.

Pontos Positivos

- O novo método de desenvolvimento é mais rápido que o normal;
- Em situações críticas, permite que o aplicativo chegue mais rápido ao público;
- Como a arquitetura tem que ser decidida de antemão, permite um desenvolvimento mais estável;

Pontos Negativos

- Quanto mais rápido o aplicativo fica pronto, maior a chance de aparecerem bugs;
- Alta necessidade de manutenção pós lançamento e alta dependência de feedback dos usuários;
- Com os ciclos de desenvolvimento sendo super curtos, os programadores devem estar envolvidos com o desenvolvimento por muito tempo durante o dia, aumentando a possibilidade de "*burnouts"*.

Análise crítica

Como o foco do aplicativo desenvolvido é salvar vidas, ele deve ser feito com a maior rapidez possível, logo o novo método de desenvolvimento proposto é extremamente útil nesse cenário, e só nesse cenário. Embora o aplicativo tenha obtido sucesso estando disponível em vários países como a ferramenta principal de resgate e, de acordo com os dados fornecidos pelo autor, o Google Play não identificou nenhuma instância de falhar ou crash do aplicativo, foi irresponsável colocar vidas humanas nas mãos de um aplicativo não testado.

Actionable Analytics: Stop Telling Me What It Is; Please Tell Me What to Do!

Referência do artigo

C. Tantithamthavorn, J. Jiarpakdee and J. Grundy, "Actionable Analytics: Stop Telling Me What It Is; Please Tell Me What To Do" in *IEEE Software*, vol. 38, no. 04, pp. 115-120, 2021.

Resumo do artigo

Qual o contexto?

O artigo trata de análises de software, discutindo como péssimas decisões levam a um alto custo e baixa reputação e quais alternativas e ferramentas podem ser utilizadas para tomar uma boa decisão.

Qual o problema?

O sucesso de projetos de software depende de decisões complexas, como quais atividades devem ser realizadas primeiro, quem deve ser encarregado por tais atividades e se o software é de qualidade e confiável. A questão é, numa análise de software, como e por quê isso deve ser feito?

Qual a solução?

Enquanto que a análise de software ajuda organizações de software a destilar insights e dá suporte a tomada de decisões, ainda existem barreiras para a adoção de tais análises. É comum que os desenvolvedores não entendam as predições feitas com a análise de dados, o que os leva a questionamentos do tipo “por que esta pessoa se encaixa nesta tarefa?”, o que dificulta a própria prática da análise. Além disso, decisões injustificadas de análise de dados podem ser catastróficas para o negócio de uma organização.

Em uma pesquisa realizada pelos autores, constatou-se que 91% dos estudos de predição de defeitos apenas focam nas predições em si, sem considerar uma explicação para as mesmas. Portanto, a explicação e contestação da análise de dados é a solução para avaliar certos pontos no desenvolvimento. A utilização de suítes de aprendizado de máquina para fazer predições e ainda explicá-las é indispensável: salva tempo e custo.

Como avaliou a solução proposta?

Relevante, pois com a utilização de machine learning para prever defeitos na análise de dados é possível investir o tempo, que antes seria ocupado pelo desenvolvedor fazendo essas predições, em outras atividades; além do mais, é uma forma mais eficaz e segura de trazer explicações para tais predições.

Pontos positivos

- O tema foi abordado de forma compreensível;
- Apesar de um assunto denso, o autor consegue claramente passar sua mensagem;
- As soluções são apresentadas de forma concisa.

Análise crítica

O artigo serve bem para ilustrar como no cenário de desenvolvimento de software pontos que têm maior relevância são meramente esquecidos, ou não tem tanta atenção quanto deveriam. Ao recorrer a ferramentas de inteligência artificial, deslizes humanos podem ser cada vez mais evitados.

Automatic Program Repair

Referência do artigo

J. Carver, R. Colomo-Palacios, X. Larrucea and M. Staron, "Automatic Program Repair" in *IEEE Software*, vol. 38, no. 04, pp. 122-124, 2021.

Resumo do artigo

Qual o contexto?

O artigo trata do assunto *“Automatic Program Repair", “Reparo automático de programas”,* onde mostra diversos estudos na área.

Qual o problema?

Como o próprio nome sugere, o reparo automático de programas engloba uma suíte de tecnologias que automaticamente repara bugs e outras vulnerabilidades em softwares. Os diferentes estudos mostrados no artigo mostram vantagens e até mesmo possíveis problemas no uso de tais tecnologias.

Qual a solução?

Em um dos estudos são analisados trechos de código que produzem saídas corretas, mas falham em um determinado ponto além da suíte de testes. Com ferramentas de reparo automático e utilizando anti-padrões para improvisar a performance, mostraram-se resultados favoráveis com a diminuição de falhas além dos testes, sugerindo a aplicação dessa técnica em ferramentas de reparo automático.

Por outro lado, em outro estudo, uma solução para evitar erros relacionados a strings em determinadas linguagens de programação, o uso de ferramentas de reparo automático deve ser feito em diferentes estágios, mas de forma apropriada. Em alguns casos, apenas uma linha de código pode solucionar todo o problema, algo que pode ser feito manualmente.

Como avaliou a solução proposta?

Cada estudo apresenta diferentes tipos de solução, mas todas elas, aparentemente, contribuem de forma positiva para o uso de ferramentas de reparo automático, o que salva tempo e trabalho durante o desenvolvimento de software.

Pontos negativos

- O tema foi algo um tanto complexo de se entender, dado a maneira com que o artigo foi estruturado;
- Nem todo estudo no artigo em si sugere ou implica uma solução;

Análise crítica

Os estudos mostram pontos positivos e negativos do uso de técnicas de reparo automático, mas a maneira com que o artigo em si foi elaborado deixa a desejar. Não foi uma leitura fácil, e não necessariamente tem a ver com o idioma, mas com a maneira que as informações são passadas.

Why Is It Getting Harder to Apply Software Architecture?

Referência do artigo

G. Fairbanks, "Why Is It Getting Harder To Apply Software Architecture?" in *IEEE Software*, vol. 38, no. 04, pp. 126-129, 2021.

Resumo do artigo

Qual o contexto?

O artigo discute o porquê de estar ficando mais difícil de aplicar arquitetura de software quando se comparado ao século passado. Apesar de no cenário de desenvolvimento existirem profissionais que entendem perfeitamente o estilo arquitetural perfeitamente, há a deficiência na hora de aplicá-lo.

Qual o problema?

Uma das razões de existir tamanha dificuldade na aplicação da arquitetura de software é o fato de a maioria dos desenvolvedores operarem em processos iterativos que focam toda sua atenção no que é novo, e que demandam respostas rápidas. Isso decorre desde o início do século 21, com mudanças que impactaram o cenário de desenvolvimento, sendo a primeira delas a mudança de atenção: de design para gestão.

Antigamente, atividades relacionadas à engenharia como escolha de objetos, modelagem de problema, alocação de responsabilidades e quaisquer atividades relacionadas a padrões de projeto eram o meio de comunicação para o design. Isto mudou para atividades de gestão, como reuniões de projeto, interações com negócios e principalmente o uso de iterações. Apesar de o método iterativo funcionar melhor que o método de cascata, nos anos 90 a maioria dos times de desenvolvimento ainda utilizavam o último.

A segunda mudança se deu ao fato da inclusão da automação, minimização do trabalho e redução de ciclos. Tendo isto em mente, percebe-se que o método cascata foca no sistema como um todo, ao contrário do iterativo, que foca apenas no que é novo.

Qual a solução?

As mudanças implicaram em projetos que se iniciam com uma velocidade absurda de ideias, mas que eventualmente essas ideias entram em conflito, o que resulta no abandono do projeto. Apenas adicionar novas features em um projeto, sem consolidar as ideias com o código, acaba resultando em prejuízo para os desenvolvedores. Dito isto, manter o método iterativo é primordial, desde que sejam incluídas novas formas de avaliação do sistema. Desenvolvedores que não compreendem a teoria por trás de um programa tendem a fazer péssimas escolhas durante a fase de desenvolvimento.

Como avaliou a solução proposta?

Fundamental, pois mesmo em pequenos projetos o design de sistema como um todo é descartado para a adição de novas features, o que compromete a interação entre os membros e o desenvolvimento. Continuar com o método iterativo, mantendo o foco no design de sistema como objetivo principal e adição de novas features como secundário, mas ainda como um ponto crucial a ser realizado, ajudaria na grande dificuldade que é aplicar técnicas arquiteturais no sistema.

Pontos positivos

- O tema foi abordado de forma compreensível;
- Apesar de um assunto denso, o autor consegue claramente passar sua mensagem;
- As soluções são apresentadas de forma concisa.

Análise crítica

O autor trouxe um problema bastante recorrente, que é o desespero pela adição de novas features em um projeto. Se uma equipe foca, prioritariamente, no resultado e esquece todo o processo de design de sistema, o desfecho nem sempre é favorável. Como dizem, saber escrever código não significa ser um bom desenvolvedor.

Tug Grall on Redis

Referência do artigo

A. Manchale, "Tug Grall On Redis" in *IEEE Software*, vol. 38, no. 04, pp. 130-132, 2021.

Resumo do artigo

Qual o contexto?

O artigo trata de uma entrevista com o gerente técnico de marketing na empresa *Redis Ltd.*, Tug Grall, e discute o *Redis (Remote Dictionary Server)* - armazenamento de estrutura de dados em memória - e como o mesmo funciona.

O que é abordado pelo entrevistado?

O entrevistado é questionado sobre o que é o *Redis* e como ele pode ser utilizado. Basicamente, o *Redis* funciona como um NOSQL (banco de dados não relacional) para casos de uso que requerem velocidade, escalabilidade e flexibilidade, tendo iniciado como uma ferramenta open source, onde a comunidade adicionou diversas novas funcionalidades. O *Redis* mantém os dados armazenados em memória, ao invés de no disco, e em banco de dados, o que garante que operações sejam feitas em 1ms ou até menos.

O *Redis* demonstra ser um sistema de mensagens durável, exatamente por operar em memória: apesar de ser armazenado em memória, há ainda a opção de ser salvo no disco. Logo, o usuário escolhe onde e quando salvar, se adequando às suas necessidades. *Redis* também pode ser utilizado como um banco de dados.

Como utilidades não tradicionais, dado o fato de ser simples e fácil de usar, o *Redis* possui módulos que proporcionam a consulta de índices na pesquisa de tipos, algo que apenas desenvolvedores mais avançados conseguem realizar.

Pontos positivos

- O entrevistador priorizou as perguntas mais importantes;
- O entrevistado conseguiu responder as perguntas de forma compreensível e detalhada;

Análise crítica

A entrevista trouxe uma abordagem sobre o *Redis*, que até então eu não fazia ideia de sua existência. O fato de ser usado como um banco de dados em memória com durabilidade opcional, que oferece suporte a diversos tipos de estruturas de dados e ser open source o torna uma ferramenta atrativa, e não é atoa que é o banco de dados de valores-chave mais popular do mundo.