CESAR SCHOOL
Unidade de Educação do Centro de Estudos e Sistemas Avançados do Recife (CESAR)

MESTRADO PROFISSIONAL EM ENGENHARIA DE SOFTWARE

Proposta de Pesquisa

Autor:
Helton Santa Cruz Souto

Orientador:

Linha de Pesquisa:
Qualidade de Software

CONTEÚDO

Apresentação

O documento corrente tem o objetivo de apresentar uma pesquisa a ser realizada durante o Mestrado Profissional em Engenharia de Software, com a finalidade de envolver pontos práticos, atuais e de necessidade de várias empresas de software e, dessa forma, adquirir mais conhecimento para atuação profissional prática de desenvolvimento de software com qualidade, além de contribuir para a área acadêmica com o aprofundamento em conceitos bastante relevantes.

Motivação para a pesquisa

Os sistemas de softwares precisam evoluir pois são criados, geralmente, para refletirem situações do mundo real, o qual está mudando constantemente (GALL et al., 1997). Cada vez mais a população depende do avanço de produtos de softwares pois estes estão presentes em todas as áreas da sociedade como no setor financeiro, entretenimento, saúde, indústria como um todo e vários outros setores. Dessa forma, no funcionamento da sociedade moderna a engenharia de software tem um importante papel (SOMMERVILLE, 2011). O alto dinamismo da sociedade atual junto com as expectativas dos usuários provocam uma intensa pressão para mudanças em sistemas de software que podem se tornar inúteis, obsoletos e cairem em desuso, caso não sofram essa transformação exigida. Portanto, é necessário que os sistemas evoluam de forma a acompanharem esse desenvolvimento progressivo nos seus universos reais específicos.

Na área de evolução de software, Lehman é considerado um dos pioneiros e, por vários anos, realizou estudos em que a partir deles foram formuladas leis que são consideradas como alicerce para a caracterização da evolução do software. Essas leis ficaram conhecidas como leis de Lehman e são oito. As primeiras foram apresentadas no inicio dos anos 70 e as últimas na década de 90. Elas se aplicam aos sistemas do "tipo E", que são aqueles que refletem problemas do mundo real, que evoluem constantemente devido às mudanças causadas por novas funcionalidades ou por problemas encontrados na sua utilização (MENS e DEMEYER, 2008).

Alguns objetivos podem ser alcançados através do estudo da evolução do software, como por exemplo: identificação de boas práticas de engenharia de software ao analisar como grandes sistemas evoluem por tanto tempo (BENNET e RAJLICH, 2000); identificação de padrões colaborativos utilizados entre desenvolvedores (Cataldo e HERBSLEB, 2008); e através da visualização do histórico do software, identificar possíveis problemas no código (D‘AMBROS et al., 2008).

Da mesma forma que acontece com o envelhecimento humano, o software envelhecerá inevitavelmente. Porém, podemos tomar algumas medidas para entender suas causas, retardar esse processo e até mesmo diminuir os efeitos dos impactos causados por ele em determinadas situações. Envelhecimento de software não é um fenômeno novo, mas vem ganhando muita atenção por causa do crescimento econômico da indústria do software e pelo fato de, cada vez mais, o softwre ser uma parte importante do "capital" de muitas empresas de tecnologia. Muitos produtos antigos de software são parte integrante de sistemas maiores e muito utilizados pela sociedade, e o envelhecimento de alguns desses produtos está impedindo a evolução de tais sistemas. Proprietários de novos produtos de software muitas vezes olham para o software antigo com desdém. Eles acreditam que, se o produto tivesse sido desenvolvido utilizando técnicas atuais, o mesmo não estaria causando problemas. Assim como todos ficaremos velhos um dia, o software também envelhecerá.

Então devemos reconhecer que esse fato vai acontecer com nossos produtos e nos prepararmos para isso. Quando a velhice chega, devemos estar preparados para lidar com ela (PARNAS, 1994). Sendo assim, como os sistemas de software estão se ampliando muito rapidamente em tamanho, estrutura, complexidade e funcionalmente (LEHMAN, 1998), é necessário se antecipar às mudanças, porém não é um trabalho fácil, pois existem vários motivos pelos quais os sistemas são modificados (PFLEEGER, 1998). Por isso, é preciso que se faça um bom planejamento para que as métricas desses sistemas também evoluam positivamente.

O envelhecimento do software pode ocorrer de duas formas distintas segundo (PARNAS, 1994): ausência de mudança e presença de mudança. A primeira ocorre quando não ocorrem mudanças no sistema para refletir aos novos requisitos de negócio, e a segunda é provocada pela forma de como tais mudanças são realizadas, muitas vezes de forma não planejada, acarretando a geração de novos erros e diminuindo a sua manutenibilidade devido a má qualidade de código.

Segundo (JÚNIOR, 2012), surgiram muitas propostas de definição para evolução e manutenção de software. No entanto, todas apontam que a manutenção direciona para mudanças corretivas e evolutivas no produto de software. A primeira definição de Lehman para manutenção de software foi a de qualquer modificação realizada, para correção de erros ou para ajustes que reflitam mudanças do ambiente após a instalação inicial do sistema no ambiente de produção. A fase mais dispendiosa de um software é a manutenção, consistindo em cerca de 90% do custo de todo seu ciclo de vida (LI et al., 2010).

Rotineiramente, a manutenção de software é deixada em segundo plano pois é considerada, por grande parte das pessoas envolvidas no desenvolvimento, como algo que só gera custos na construção do software e, devido às restrições de custo e prazo, a qualidade do software é comprometida em detrimento de alguma outra urgência que surge durante o processo. Esse tipo de fenômeno é chamado atualmente de Dívida Técnica.

A metáfora da dívida técnica foi definida por (CUNNINGHAM, 1992) como uma analogia ao débito financeiro. A partir do momento em que a qualidade do sistema é comprometida, uma dívida é adquirida. E essa dívida deverá ser paga em algum momento futuro. Quanto mais tempo essa dívida demorar a ser paga, juros serão acrescidos a ela em forma de custo, esforço e tempo, que dificultarão a realização de manutenções futuras

no software. A decisão será do projeto de permanecer nesse aumento progressivo da dívida ou, de alguma outra forma, pagá-la realizando atividades de refatoração de código por exemplo.

Em algum momento a dívida pode ser necessária, porém se não for bem gerenciada, a tendência é que surjam vários problemas futuros como custos altos com manutenção e diminuição de produtividade (BROWN, et al., 2010), que acarreta negativamente a saúde financeira de um projeto.

Esse tema tem demonstrado muita relevância nas empresas de desenvolvimento de software e também na comunidade científica. Essa pesquisa tem o objetivo de investigar e analisar em vários projetos de softwares reais a qualidade de código após mudanças ocorridas nesses sistemas no decorrer de várias versões instaladas ni ambiente de produção.

Descrição do problema a ser resolvido

Devido às várias mudanças com diversas características que ocorrem nos sistemas, provenientes de correção ou evolução de funcionalidades, o código é modificado constantemente por várias pessoas ao mesmo tempo, muitas delas com pouquíssima experiência no mercado, sem seguir boas práticas de programação e, muitas vezes, inseridas em ambientes de muita pressão que existem nas empresas atualmente. Esse tipo de situação leva a sugerir que, com o decorrer das entregas das várias versões desenvolvidas, a qualidade do código será muito afetada.

O foco principal dessa proposta é analisar a evolução da dívida técnica em produtos de softwares proprietários desenvolvidos em JAVA e descrever as relações existentes entre as características das solicitações de mudanças, das equipes envolvidas e os impactos gerados na qualidade do código, ou seja, verificar quais principais características de uma demanda influenciam mais ou menos na evolução da dívida técnica de um produto. Para isso, serão realizados estudos experimentais envolvendo dados históricos dos códigos fontes e documentações de entregas de versões anteriores de sistemas mantidos pela empresa pública DATAPREV.

Metodologia

Com o objetivo de analisar o problema proposto, será seguida a seguinte metodologia:

1. Levantamento bibliográfico envolvendo os principais conceitos relacionados a evolução e manutenção de software;

2. Levantamento bibliográfico a cerca de dívida técnica com o objetivo de entender de forma mais aprofundada seus conceitos e estudos experimentais já realizados;

3. Levantar ferramentas e bibliotecas que possam facilitar e agilizar as atividades de análise estática de código dos produtos envolvidos;

4. Definir quais ferramentas serão utilizadas e quais tipos de dados elas necessitam;

5. Levantar os projetos que serão utilizados como fonte de dados para avaliação da dívida técnica em relação a qualidade do código junto a DATAPREV/PB;

6. Definir e levantar todos os artefatos que serão necessários na investigação;

7. Preparar as informações colhidas dos produtos cronologicamente de forma que se possa analisar separadamente toda dívida técnica de código de cada produto de acordo com suas evoluções de versões para que se possa realizar a análise comparativa com os possíveis influenciadores de aumento ou diminuição da dívida, como características das demandas e dos desenvolvedores;

8. Escolher participantes que validem as indicações de tempo de resolução de cada tipo ou categoria de dívida técnica levantada pelas ferramentas que serão utilizadas;

9. Propor um modelo de estimativa da evolução da dívida técnica de código a partir de características específicas de uma solicitação de mudança e características da equipe de desenvolvedores.

Conclusões e considerações finais

Acredito que a execução dessa proposta seja de plena relevância para os profissionais da área de engenharia de software, podendo ser uma referência para gestores de projetos que desejem se planejar melhor em relação ao cumprimento de prazos sem perder a qualidade de código e para a comunidade em geral envolvida com desenvolvimento de projetos, que trará uma disseminação maior dos conceitos de dívida técnica e qualidade de código.

Essa pesquisa incentiva a produção de várias pesquisas futuras, considerando que qualidade de software, de maneira geral, é sempre um objetivo a ser alcançado na indústria de software.

Referências bibliográficas

ARAÚJO, M. A. P.; TRAVASSOS, G. H. **Em busca de um Framework para estudos experimentais em evolução de software**. In: WORKSHOP DE MANUTENÇÃO DE SOFTWARE MODERNA, 3, 2006, Vila Velha, Anais... Vila Velha: WMSWM, 2006.

BENNET K.; RAJLICH V. **Software Maintenance and Evolution**: A roadmap. In: Proceedings of the Conference on The Future of Software Engineering, Limerick, Ireland, 2000.

BROWN, N.; CAI, Y.; GUO, Y.; KAZMAN, R.; KIM, M.; KRUCHTEN, P.; LIM, E.; MACCORMACK, A.; NORD, R.; OZKAYA, I.; SANGWAN, R.; SEAMAN, C.; SULLIVAN, K.; ZAZWORKA, N. **Managing technical debt in software-reliant systems**. In: Proceedings of the FSE/SDP workshop on Future of software engineering research, 2010.

CATALDO, M.; HERBSLEB, J. D. **Communication networks in geographically distributed software development.** In: Proceedings of the 2008 ACM conference on Computer supported cooperative work, 2008.

CUNNINGHAM, W. **The WyCash Portfolio Management System**. In: Conference on Object-Oriented Programming
Systems, Languages, and Applications (OOPSLA'92). Vancouver, 1992.

D'AMBROS, M.; LANZA, M.; ROBBES, **R. Evaluating defect prediction approaches: a benchmark and an extensive comparison.** Empirical Software Engineering, v.17, 2012.

GALL, H.; JAZAYERI, M.; KLOSCH, R. R.; TRAUSMUTH, G. **Software Evolution Observations Based on Product Release History**. Proceedings of Internacional Conference on Software Maintenance (ICSM´97), 1997.

JÚNIOR, A. L. O. C. **Métricas como Ferramenta de Auxílio para o Gerenciamento de Dívida Técnica em Produtos de Software**. Recife, PE: UFPE, 2012.

KEMERER, C. F.; SLAUGHTER, S. **An empirical approach to studying software evolution**. In: IEEE Transactions on Software Engineering, vol. 25, 1999.

LEHMAN,M.M. **Programs, life cycles, and laws of software evolution**. In: Proceedings of the IEEE, 1980.

LEHMAN, M. M. **Software's future: Managing Evolution**. In: IEEE Software, vol. 15, 1998.

LI, J.; STÅLHANE, T.; KRISTIANSEN, M. W.; CONRADI, R. **Cost drivers of software corrective maintenance: An empirical study in two companies**. In: IEEE International Conference on Software Maintenance, 2010.

MENS, T; DEMEYER, S. **Software Evolution**. Berlin: Springer-Verlag, 2008.

PARNAS, D.L. **Software Aging**. In: Proceedings of 16th International Conference on Software Engineering, Sorrento, 1994.

PFLEEGER, S. L. **The Nature of System Change**. In: IEEE Software, vol. 15, 1998.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson, 2011.