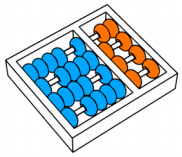

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE CIÊNCIA DA COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIA DE COMPUTAÇÃO

Modelagem e Construção de um Ambiente Computacional para Apoio Automatizado às Atividades de Manutenção de Software Livre Baseado na Extração de Medidas Históricas e Sociais.

Plano de Trabalho do Doutorado

Luiz Alberto Ferreira Gomes
(gomes.luiz@gmail.com)
Candidato

Prof. Dr. Mario Lúcio Côrtes
(cortes@ic.unicamp.br)
Orientador

Resumo

O software ocupa um papel essencial na sociedade moderna. A crescente demanda por novas aplicações ou por manutenções em aplicações softwares já existentes têm imposto enormes desafios à disciplina de Engenharia de Software. Pesquisas e investimentos têm sido realizados em modelos de maturidade, normas, processos e ferramentas computacionais para entregar o software dentro do prazo estabelecido, com os recursos previamente planejados e, principalmente, com uma qualidade aceitável. Percebe-se que se obteve um avanço significativo nesse sentido nas últimas décadas. Entretanto, diversos autores apontam que muitos problemas relacionados com a não só com entrega, mas também com a manutenção e evolução, e a qualidade do software ainda persistem. Em razão da complexidade de vários processos de software e de grandes projetos de software recomenda-se o uso intensivo de ferramentas computacionais para apoiar o Engenheiro de Software na execução de suas atividades relacionadas ao desenvolvimento, à manutenção e à evolução. Uma opção considerada econômica e com qualidade de reconhecida pela indústria é o emprego de ferramentas de software livre para o apoio a essas atividades, não só de produtos de software proprietários, mas também em grandes projetos de software livre. Porém, a heterogeneidade e a falta de interoperabilidade entre a maioria delas dificultam a integração dos dados gerados em uma visão uniforme de todo o ciclo de vida de um produto. O objetivo deste projeto de pesquisa é modelar e construir um ambiente computacional que permita a avaliação e a melhoria da manutenibilidade do produto de software com base na extração de um conjunto de medidas (históricas e sociais), a partir de ferramentas de apoio às atividades de Engenharia de Software, relacionadas à manutenção e evolução de softwares. O foco da pesquisa, em razão da sua importância e da disponibilidade de uma gama enorme de dados, será colocado em projetos de software livre.

Palavras-chaves: manutenibilidade, manutenção e evolução de software, visualização de software e software livre.

1 Introdução

É consenso que as aplicações de software desempenham um papel fundamental no cotidiano das organizações, das pessoas e, de maneira geral, da sociedade moderna como um todo. Muitos negócios ou empreendimentos são fortemente baseados nessas aplicações. O software está presente em uma infinidade de atividades: desde a realização um saque em um caixa eletrônico até o controle de uma aeronave que pousará para um determinado destino. Desse cenário, de clara dependência, emerge a preocupação com a manutenção dessas aplicações – conhecidas como sistemas legados –, corrigindo falhas e com sua evolução, para adicionar ou modificar funcionalidades, para atender a novos requisitos de negócios ou mudanças tecnológicas para que continuem úteis às organizações que as utilizam. Essa preocupação e corroborada por estudos que apontam que entre 65% e 85% do orçamento para a área de tecnologia da informação estão sendo gastos com a manutenção de software (SOMMERVILLE, 2011).

A manutenção e a evolução de software estabelece desafios importantes à indústria e aos pesquisadores da área. Entre os quais, MENS (2005) destaca: (i) preservação e aperfeiçoamento da qualidade de software; (ii) construção de uma plataforma comum para evolução de software; (iii) suporte à evolução de modelos, (iv) suporte aos sistemas com múltiplas linguagens; (v) integração da manutenção e evolução no ciclo de vida; (vi) necessidade de sistemas de versionamento mais adequados; (vii) integração de dados provenientes de diversas fontes; (viii) análise de grandes quantidade de dados; (ix) necessidade de melhores modelos de estimativa e (x) evolução em tempo de execução após implantação.

O software se "deteriora" gradativamente – em maior ou menor grau de acordo com as técnicas que são empregadas – em razão das inúmeras manutenções sofridas durante o seu ciclo de vida (PRESSMAN, 2006). Tal deterioração contamina negativamente a manutenibilidade do software – medida que, de acordo com o *Software Institute Engineering* (SEI), mede o esforço necessário para realizar modificações na implementação de um software. Embora não exista um entendimento comum a respeito sobre o que seja manutenibilidade ou, mesmo, como medi-la, considera-se que quanto menor o valor dessa medida, maior o esforço e, consequentemente, maior o custo para manter determinada aplicação. KOZIOLEK (2011) acrescenta, ainda, que sistemas de grande porte e de longo de tempo de vida são considerados manuteníveis somente se podem mantidos e evoluídos

eficientemente a um custo aceitável durante todo o ciclo de vida.

Ferramentas computacionais de software livre (em inglês: *Free Software/Open Source* Software (FOSS)) vêm atraindo a atenção de um grande número empresas e profissionais interessados em automatizar diversos processos. O software livre possibilita ao usuário executar, estudar, modificar para atender as suas necessidades, além de distribuir cópias originais ou modificadas do aplicativo sem qualquer ônus (WHEELER, 2015). Esse fato torna o software livre uma opção economicamente viável para empresas que possuem orçamentos restritos ou que não desejem arcar com custos de compra e licenciamento de software. O que é válido para os próprios projetos de software, que tem à sua disposição uma gama razoável de opções, com excelente qualidade técnica, para automatizar diversas atividades do seu processo de software.

Percebe-se, entretanto, a existência de uma lacuna com relação à integração de dados dos diversos ambientes computacionais que fornecem o apoio automatizado à manutenção e evolução do FOSS (PANSANATO; FORTES, 2004), que poderiam, de outra maneira, representar importantes informações que poderiam ser utilizadas efetivamente durante o ciclo de vida desse tipo de software. Entre as características que afetam os esforços necessários para integrá-los destacam-se: a heterogeneidade e a falta de interoperabilidade entre as entre esses ambientes; e o fato de que muitos FOSS relevantes, como o *kernel* do Linux, são desenvolvidos por profissionais de países distintos, com culturas e conhecimentos técnicos muito diferentes(SCIALDONE et al., 2009). Esse tema, em razão do crescimento da preocupação com a qualidade de software e, também, do número de projetos de software livre têm estimulado diversas pesquisas no meio acadêmico e na indústria de software para tratar esse problema.

2 Motivação e Justificativa

A manutenção de software não é uma preocupação recente da indústria. A conferência de engenharia de software, promovida pela Organização do Tratado do Atlântico Norte(OTAN), realizada em 1968, apontou como um dos fatores responsáveis, pelo que ficou conhecido como “a crise de software”, o custo da manutenção de sistemas de software, que naquela época, já eram considerados elevados. Em seu livro clássico sobre engenharia de software, BROOKS (1995) chama a atenção que “o custo total de manutenção de software de médio e grande portes é tipicamente 40 ou mais do custo de desenvolvimento”.

Em razão do impacto econômico, a manutenibilidade (métrica associada aos esforços

de manutenção) tornou-se amplamente aceita como um importante atributo da qualidade de sistemas de software e, com o amadurecimento das práticas de desenvolvimento de software, a manutenibilidade de software converteu-se em uma das maiores preocupações da indústria de software (BROY, 2006).

3 Problema e Hipóteses

Embora existam nas referências da área, muitos princípios, recomendações e padrões sugeridos para aumentar a manutenibilidade de um produto, que, via de regra, podem ser aplicados em todas as etapas de processo de software; quase sempre os impactos ocasionados pela aplicação inadequada ou, mesmo, pela ausência de tais práticas, não são percebidos claramente durante as atividades de desenvolvimento. Levando, em muitos casos, ao aumento do tempo e dos custos da execução dessas atividades em razão do retrabalho durante as atividades de manutenção e evolução.

Em razão da grande quantidade de dados, produzidos por diversos participantes (algumas vezes espalhados geograficamente, como equipes de FOSS), durante o desenvolvimento, manutenção e evolução de um produto; aliado à lacuna de ambientes integrados que possam coletar e processar medidas de manutenção e evolução – em todas as etapas do processo de software. Fornecer uma visão unificada e conveniente a todos os membros equipe de desenvolvimento sobre a manutenibilidade do seu software, para que possam perceber como seu produto está evoluindo em termos de critérios associados à manutenção de software torna-se uma tarefa bastante complexa.

3.1 Hipóteses Levantadas

- **H1**: Existe uma grande quantidade de medidas históricas e sociais associadas à manutenção de software que não são correlacionadas pelas equipes de desenvolvimento em razão de serem provenientes de diversas fontes e possuírem diversos formatos.
- **H2**: Existe a possibilidade de extrair medidas de manutenção a partir de mídias sociais e de ferramentas de colaboração**.**

4 Objetivos e Contribuições Esperadas

O objetivo geral deste projeto de pesquisa é construir uma infraestrutura computacional (modelos, processos e ferramentas) que permita a extração, o processamento, a visualização e a melhoria da manutenibilidade de um produto de software baseada em medidas históricas e

sociais. Em decorrência da pesquisa realizada, vislumbram-se, ainda, os seguintes objetivos específicos e contribuições:

- Especificação de metamodelos que possibilitem a descrição de medidas associadas a manutenibilidade em diversos artefatos gerados em todas as etapas do processo de software.
- Elaboração de uma arquitetura de software baseada nos metamodelos, mencionados acima, em que facilite a extração de medidas históricas e sociais;
- Catalogação de processos, métodos e ferramentas utilizados para a manutenção e evolução de FOSS.
- Construção de um sistema tutor inteligente que oriente a equipe de desenvolvimento nas ações de melhoria da manutenibilidade do seu produto.
- Avaliação da manutenibilidade de grandes projetos de software libre a partir das informações geradas pelo ambiente computacional proposto..

5 Linhas de Pesquisa

Esta seção descreva algumas áreas de pesquisas que serão objeto do estudo ou, potencialmente, oferecerão técnicas que serão utilizadas para se atingir os objetivos estabelecidos no trabalho.

5.1 Aprendizado de Máquina

Aprendizado de máquina é um subcampo da inteligência artificial dedicado ao desenvolvimento de algoritmos e técnicas que permitam ao computador "aprender", isto é, que permitam ao computador aperfeiçoar seu desempenho em alguma tarefa. Enquanto que na inteligência artificial existem dois tipos de raciocínio – o indutivo, que extrai regras e padrões de grande conjuntos de dados, e o dedutivo – o aprendizado de máquina só se preocupa com o indutivo. No contexto desta pesquisa, as técnicas de aprendizado de máquina poderiam ser empregadas para a construção de modelos de previsão de manutenibilidade com base nas medidas levantadas para manutenção de software livre.

5.2 Mineração de Dados

A mineração de dados é o processo não trivial de encontrar em grandes volumes de dados, padrões que sejam válidos, inéditos, potencialmente úteis e, por último, compreensíveis. Para alcançar esse objetivo, algumas técnicas são usualmente aplicadas (ELSMARI, 2005):

1. **Regras de associação**: envolve a descoberta de associações entre conjuntos de itens de tal maneira que a presença de um conjunto de itens implica a presença de outro conjunto de itens.
2. **Hierarquia de classificação**: envolve o tratamento de um conjunto de eventos ou transações para a criação de uma hierarquia de classes.
3. **Padrões sequenciais**: procura-se detectar associações entre itens ao longo do tempo. Os itens são associados em uma linha do tempo de forma que a presença de um conjunto de itens implicará a presença de outro conjunto de itens no futuro.
4. **Padrões de séries temporais**: baseia-se em similaridades que podem ser encontradas em posições em uma série temporal.
5. **Agrupamentos**: baseia-se no fato de que uma dada população de eventos ou novos itens podem ser segmentados em um conjunto de elementos similares.

5.3 Manutenção de Software

SOMMERVILLE (2011) define a manutenção de software como a modificação do produto após ter sido colocado em uso. Essas mudanças podem ser motivadas pela correção de erros de codificação, pela correção de erros de projeto ou de especificação e pela necessidade da inclusão de novos requisitos. Ele enumera três tipos de manutenção diferentes:

1. **Correção de defeitos:** envolve a correção de erros de programação, erros de projeto ou erros de especificação. Em termos custos, os erros de programação são os mais baratos e os erros de especificação os mais caros de se corrigir.
2. **Adaptação ambiental:** engloba modificações realizadas em razão da alteração do hardware, do sistema operacional ou de algum outro componente do ambiente em questão.
3. **Adição de funcionalidade:** esse tipo de manutenção é ocasionado pela mudança de requisitos de software por causa das mudanças organizacionais ou de negócios.

As atividades de manutenção mais caras (em termos de custos e esforços), segundo (SOMMERVILLE, 2011), não aquelas relacionadas à correção de defeitos, mas sim aquelas associadas à adaptação do software, para que o ele possa lidar com novos ambientes operacionais ou para atender a novos requisitos.

5.4 Visualização de Software

A visualização de software é uso de recursos tipográficos, gráficos, animações e cinematografia com técnicas modernas de interação homem-computador e de computação gráfica com o intuito de facilitar o entendimento humano e o uso efetivo do software de computador (DIEHL, 2007).

6 Trabalhos Relacionados

Esta seção descreve alguns trabalhos de pesquisa relacionados ao trabalho proposto neste documento.

6.1 HPMAS

O *Hierarchical Multidimensional Assessment Model* (HPMAS) é uma ferramenta computacional desenvolvida pela equipe de engenheiros de manutenção da empresa HP com o objetivo de automatizar a avaliação da manutenibilidade de seus produtos de software e, com essa informação, tomar decisões mais adequadas sobre a manutenção ou, mesmo, sobre a substituição de determinado produto (OMAN; HAGEMEISTER, 1992). O modelo em que a ferramenta é baseada enxerga a manutenibilidade de software como uma hierarquia estruturada de atributos do código fonte.

6.2 ConQAT

BROY (2006) propõe um modelo baseado em duas dimensões para avaliação da manutenibilidade de software: atividades de manutenção (por exemplo: análise, implementação, teste ou implantação); e em característica do sistema (por exemplo: documentação), da organização(por exemplo: habilidades) e do ambiente operacional(por exemplo: ferramentas de desenvolvimento). Sendo que cada dimensão pode ser subdividida até o nível de folha o atômico (quando não é possível mais dividi-la). Uma vez que a organização tenha definido as atividades e características relevantes para o seu contexto, ela pode relacionar as duas dimensões levando a perceber quais características têm impactos em quais atividades de manutenção. Nesse trabalho, os autores sugerem que a abordagem proposta possa auxiliar na identificação de critérios de qualidades relevantes e o julgamento (avaliação) de suas interdependências. O processo de controle de qualidade resultante pode ser reforçado por meio medições apoiadas por ferramentas automatizadas, bem como inspeções manuais. Uma limitação da abordagem proposta para equipes de software livres é a ausência de uma entidade central que possa definir, para cada projeto, os critérios e a atividades relevantes para sua situação.

7 Cronograma Proposto

O projeto de pesquisa foi subdividido nas seguintes etapas:

(1) Cumprimento dos créditos exigidos pelo programa de doutorado cursando três disciplinas básicas oferecidas pelo IC-UNICAMP;

(2) Levantamento de material bibliográfico sobre o estado da arte e temas relacionados ao trabalho proposto. Essa atividade será realizada de forma mais enfática durante a etapa do exame de qualificação, porém será estendida a todo o período de desenvolvimento do projeto, de forma a fornecer informações que possam contribuir para o mesmo;

(3) Realização de exame de qualificação, constituído da escrita de monografia referente ao tema do projeto, revisão do plano de trabalho e apresentação da mesma a uma banca julgadora;

(4) Realização de exame de proficiência em língua inglesa, através de exame certificado pela Comissão de Pós-graduação;

(5) Estudo, levantamento e escolhas das métricas de manutenibilidade que podem ser extraídas nas etapas do processo de software;

(6) Estudo das arquiteturas das ferramentas mais proeminentemente utilizadas no desenvolvimento de FOSS;

(7) Estudo das principais técnicas relacionadas à área de visualização de software;

(8) Projeto do metamodelo para coletar e integrar as métricas definidas no item (5);

(9) Projeto de ferramentas de apoio com base no modelo citado no item(8). Esse projeto deverá contemplar a utilização de padrões de projeto e padrões arquiteturais modernos.

(10) Implementação de ferramentas de apoio com base no projeto desenvolvido nos itens (8) e (9);

(11) Validação das ferramentas implementadas por meio de testes utilizando estudos de caso. Uma das possibilidades seria utilizar como estudo de caso o Kernel do Linux que é considerado representante importante de FOSS;

(12) Submissão de resultados do trabalho em formato de artigos em eventos ou revistas de nível internacional da área de conhecimento em questão;

(13) Redação da tese de doutorado;

(14) Defesa da tese de doutorado;

A Tabela 1 apresenta o cronograma planejado para a execução das atividades definidas no parágrafo anterior.

Tabela 1: Cronograma Proposto.

Atividade	2016	2017		2018		2019		2020
	2º Semestre	1º Semestre	2º Semestre	1º. Semestre	2º. Semestre	1º. Semestre	2º. Semestre	1º. Semestre
(1)	X							
(2)	X	X	X	X				
(3)			X					
(4)		X						
(5)	X	X						
(6)	X	X						
(7)		X	X					
(8)			X	X				
(9)			X	X				
(10)				X	X	X		
(11)					X	X		
(12)		X	X	X	X	X	X	
(13)				X	X	X	X	
(14)								X

8 Referências Bibliográficas

BROOKS, F. P. **The mythical man-month: essays on software engineering**. Anniversary ed ed. Reading, Mass: Addison-Wesley Pub. Co, 1995.

BROY, M.; DEISSENBOECK, F.; PIZKA, M. **Demystifying Maintainability**Proceedings of the 2006 International Workshop on Software Quality. **Anais**...: WoSQ '06.New York, NY, USA: ACM, 2006Disponível em: <http://doi.acm.org/10.1145/1137702.1137708>. Acesso em: 18 ago. 2015

KOZIOLEK, H. **Sustainability Evaluation of Software Architectures: A Systematic Review**Proceedings of the Joint ACM SIGSOFT Conference – QoSA and ACM SIGSOFT Symposium – ISARCS on Quality of Software Architectures – QoSA and Architecting Critical Systems – ISARCS. **Anais**...: QoSA-ISARCS '11.New York, NY, USA: ACM, 2011Disponível em: <http://doi.acm.org/10.1145/2000259.2000263>. Acesso em: 24 set. 2015

MENS, T. et al. **Challenges in Software Evolution**IEEE, 2005Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1572302>. Acesso em: 6 out. 2015

OMAN, P.; HAGEMEISTER, J. **Metrics for assessing a software system's maintainability**IEEE Comput. Soc. Press, 1992Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=242525>. Acesso em: 2 nov. 2015

PANSANATO, L. T. .; FORTES, R. P. . **Uma Análise de Metadados para o Acesso Unificado Às Informações dos Repositórios de Ferramentas de Software Livre**. In: II CONGRESSO DE TECNOLOGIAS PARA GESTÃO DE DADOS E METADADOS DO CONE SUL. Ponta Grossa: Universidade Tuiuti, 2004

SCIALDONE, M. J. et al. Group Maintenance Behaviors of Core and Peripherial Members of Free/Libre Open Source Software Teams. In: BOLDYREFF, C. et al. (Eds.). . **Open Source Ecosystems: Diverse Communities Interacting**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. v. 299p. 298–309.

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

WHEELER, D. **Why Open Source Software / Free Software (OSS/FS, FOSS, or FLOSS)? Look at the Numbers!** Disponível em: <http://www.dwheeler.com/oss_fs_why.html>. Acesso em: 6 out. 2015.

9 Bibliografia

BOEHM, B. A View of 20th and 21st Century Software Engineering. *Proceeding of the 28th international conference on Software engineering.* Shanghai, China, May 2006, pp. 12-29.

LARMAN, C. **Utilizando UML e padrões : uma introdução à análise e ao projeto orientados a objetos e ao processo unificado**. 2. ed. Porto Alegre: Bookman, 2004.

OSSHER, H.; HARRISON, W.; TARR, P. **Software engineering tools and environments: a roadmap**ACM Press, 2000. Disponível em: <http://portal.acm.org/citation.cfm?doid=336512.336569>. Acesso em: 6 out. 2015