MAC5779 - Engenharia de Software Experimental

Modelo para estruturação da pesquisa

Versão 1.0 Prof. Marco A. Gerosa

Profa. Claudia Melo

Nome: José Teodoro da Silva

Preencha **sucintamente** os itens abaixo. Pode ser que alguns dos itens não se apliquem, principalmente em pesquisas exploratórias, pesquisa-ação ou levantamentos. Nesses casos, justifique.

Partes deste documento foram baseadas no modelo de relatório de acompanhamento discente da UNIRIO e no livro "Metodologia da Pesquisa para Ciência da Computação" (Wazlawick, 2009).

1. Contexto: fatores de influência

Quais são os fatores externos a esta pesquisa que a restringem ou a direcionam? Ex: ela é integrante a um projeto maior, ou você já tem um interesse fixo em uma tecnologia/abordagem (solução em busca de problema). É bom explicitar essas informações, pois elas acabam restringindo/direcionando a pesquisa.

Esta pesquisa é a extensão de um trabalho que teve por objetivo caracterizar os desenvolvedores chaves em uma release do projeto Apache Ant [9].

Quais são as oportunidades que você gostaria de aproveitar nesta pesquisa? Ex: acesso a um grande volume de dados, acesso a empresas, acesso a sujeitos etc.

Aproveitar o acesso ao grande volume de dados existente nos projetos da Apache Software Foundation. Existem dados públicos sobre o histórico do desenvolvimento, sobre as mensagens trocadas entre os desenvolvedores e sobre as *issues* que representam as atividades dos projetos hospedados neste repositório.

2. Tema

Tema (uma sentença sem verbo) – qual o assunto que você está investigando?

Avaliação dos métodos existentes na literatura para a identificação de desenvolvedores-chave em projetos de software livre.

3. O problema

Definição do problema (uma frase). (às vezes o problema é derivado de limitações dos métodos/ferramentas/tecnologias relacionados ao tema em questão). Lembre-se de caracterizar como um problema de pesquisa (gera conhecimento) e não como um problema técnico (que pode ser resolvido com a aplicação de uma técnica ou tecnologia já disponível).

A literatura indica a existência de diferentes métodos para encontrar desenvolvedores-chave de um projeto. Entretanto, a escolha de um método ou a comparação entre eles nem sempre é discutida. Isto torna difícil a tarefa de escolha de um dos métodos já existentes.

Evidências do problema (se possível baseadas na literatura ou em dados coletados) (coloque a resposta em itens). Algumas vezes, não se encontra evidências facilmente na literatura e torna-se necessário conduzir um estudo preliminar para gerar essas evidências (estudo primário ou secundário).

- Crowston destaca a necessidade da realização de uma comparação entre os vários métodos de identificação de membros chaves utilizando uma grande quantidade de projetos [3]
- Estudos são realizados sobre o conjunto dos desenvolvedores chaves sem identificar ou apontar na literatura uma razão para escolha do método de identificação desse conjunto[4],[1],[8],[14],[13] ,[5];
- Lakhani aponta a dificuldade de generalizar o método de identificação de desenvolvedores devido às diferentes características das comunidades de cada projeto open source [7];

O problema em questão existe apenas em uma situação em particular (localizado) ou é generalizável? Por quê?

O problema exige uma análise das características de cada método e das características dos projetos alvos do estudo.

Qual a relevância do problema? Porque é um problema interessante de ser investigado?

Alguns estudos caracterizam a divisão em camadas para representar os papéis existentes em comunidades de software livre. Entre eles, o modelo OnionPatch [6], descreve esses papéis como sendo “camadas de uma cebola”. As partes mais externas representam desenvolvedores menos experientes no projeto. À medida que esses desenvolvedores participam do projeto, eles podem migrar para papéis mais centrais no modelo (e no projeto). No entanto, estes limites entre as camadas do modelo OnionPatch não são intuitivos[3]. Não é prático utilizar os papéis existentes nos *sites* e na documentação do projeto para identificação dos desenvolvedores-chave. Em muitos casos, os desenvolvedores já possuem as características e atribuições antes mesmo de serem formalmente reconhecidos como tal [3]. A relação entre o grupo de desenvolvedores-chave e periferia (camadas mais externas) pode diagnosticar a maturidade do time que desenvolve o software [3]. Além disso, os desenvolvedores-chave são importantes para manter a comunidade unida [11] e são eles que tomam decisões importantes que podem definir o sucesso do projeto [12]. Terceiro (2010) encontrou evidências da influência dos desenvolvedores-chave para manutenção da integridade do projeto e destaca a importância de manter um grupo saudável para realizar essa tarefa [10].

Por que você acredita que o problema ainda não foi resolvido adequadamente?

Os trabalhos evidenciam a importância da identificação dos desenvolvedores-chave, mas essa identificação é efetuada de modo arbitrário em muitos estudos. Não foram encontradas análises sobre as características dos métodos em função das características dos projetos. Essa

análise pode facilitar a escolha de um método mais adequado dependendo do projeto objeto do estudo.

Quais são os principais autores/artigos que discutem ou tentaram resolver o problema (separe os que apenas discutem, os que tentam resolver e o que fazem ambos)?

- Amrit discute sobre as formas de se encontrar o membros chave mas não avalia as características de cada método de acordo com as características de cada projeto. O autor também não executa os métodos em um escopo maior de projetos para comparar seus resultados [2].
- Oliva et al apresenta um estudo exploratório que compara diferentes métodos para caracterização dos desenvolvedores chaves, porém essa análise foi efetuada em apenas uma release do projeto Apache Ant [9].

O problema no contexto em questão é similar a problemas encontrados em outros contextos?

O problema é similar à identificação de papeis para membros de redes sociais.

Caso o problema seja relacionado à aplicação da computação em uma área específica, o que essa área específica tem de diferente das outras? Ex: se você está atuando em Informática na Educação, e propõe uma nova abordagem para construção de ambientes colaborativos, o que impede/dificulta o uso de técnicas convencionais de Engenharia de Software e te motiva a propor algo específico de Informática na Educação.

Não se aplica.

4. A proposta

(se o estudo for qualitativo exploratório esse item não se aplica)

Quais são as possíveis alternativas de solução ao problema?

Uma comparação dos requisitos e dos resultados provenientes dos vários métodos existentes para identificação dos desenvolvedores-chave sobre um grande número de projetos de software livre.

Qual a sua proposta de solução? Informe a hipótese/visão/abordagem a ser avaliada na pesquisa. Formule a hipótese a ser avaliada seguindo o modelo: SE (solução_proposta) ENTÃO (dados_esperados que indicam a resolução do problema). Pode ser que a solução proposta inclua o desenvolvimento de algum sistema ou protótipo. Para os dados esperados, deixe claro quais são as métricas.

Esse estudo pode auxiliar pesquisadores na área que utilizem os desenvolvedores-chave para estudo dos fenômenos que permeiam o processo de desenvolvimento de software livre. Os métodos existentes para identificação de desenvolvedores-chave possuem diferentes características e requisitos. O levantamento dessas características aliado à comparação de seus resultados possibilita a escolha consciente do método que melhor se ajuste ao contexto a ser analisado em pesquisas futuras, tanto em complexidade quanto em restrições dos dados.

De modo geral, os estudos realizados escolheram arbitrariamente o método a ser utilizado. A execução e avaliação dos resultados desses vários métodos sobre um grande número de projetos open source pode evidenciar as fraquezas, pontos fortes, pontos fracos e requisitos de cada método.

Hipótese: Se, dentre os vários métodos de identificação de desenvolvedores-chave, forem identificados métodos simples e melhor ajustados ao contexto, que possuam resultados semelhantes ou melhores que métodos mais complexos, então será possível escolher, dentre os método avaliados, aquele que melhor se adapta ao contexto em que será aplicado.

Os resultados serão mensurados do seguinte modo:

- A complexidade dos métodos será medida pela disponibilidade de recursos necessários e pela dificuldade de execução;

- A viabilidade de cada método será avaliada de acordo com as características do projeto que podem invalidar os resultados obtidos;

- Cada método apresenta como resultado um conjunto de desenvolvedores chaves. Para determinar a similaridade entre métodos será computada a distância euclidiana entre esses conjuntos.

Por que você acredita que a sua proposta resolverá o problema?

A proposta pode evidenciar os melhores cenários para cada método e uma comparação entre as dificuldades de aplicação de cada um.

Falseamento: Há possibilidade da solução proposta NÃO resolver o problema? Por quê? Deve existir a dúvida se a hipótese é verdadeira ou falsa, caso contrário não existiria um bom motivo para realizar a pesquisa.

Existe a possibilidade da proposta não resolver o problema se não ficarem claros os cenários onde cada método se encaixa. Neste caso, a contribuição do trabalho é a comparação entre os vários métodos para evidenciar os métodos mais simples que alcancem resultados próximos gerados por métodos mais complexos.

A solução que você está propondo já foi tentada por outras pessoas para resolver o mesmo problema? Qual o diferencial da sua proposta?

Um estudo exploratório foi executado por Oliva et al, porém, a comparação dos métodos foi realizada somente em apenas uma *release* do projeto ANT [9].

Amrit comparou o impacto do deslocamento de core-periferia em oito projetos de software livre. Apesar de indicar que a análise deste deslocamento possa ser utilizada como um indicativo para a saúde de um projeto de software, o autor indica a necessidade de aperfeiçoar o estudo comparando diferentes métodos de definição de core-periferia em projetos com características diferentes e considerando uma maior quantidade de projetos [2].

A solução que você está propondo já foi usada por você ou por outras pessoas para resolver problemas de outra área?

Amrit explorou o impacto do uso de estruturas sócio-técnicas de redes para representar core-periferia em projetos de software livre. O trabalho apontou como trabalho futuro a necessidade da criação de um ranking para comparação dos métodos [2].

Este trabalho utilizará a distância euclidiana para comparar os resultados obtidos por cada método comparado. Serão executados testes dos métodos em vários projetos do repositório. Um ranking será proposto para pontuar os métodos levando em consideração seu contexto, dados, dificuldade de criação das redes entre outras características dos projetos.

5. Objetivo

Objetivo geral da pesquisa: (uma única sentença) Lembre-se de delimitar bem o escopo de seu estudo. Que tipos de situações você vai considerar? Que tipo de desenvolvedores/usuários/pessoas você vai considerar? (é necessário limitar o escopo de investigação para a pesquisa se tornar viável). Essas informações devem estar claras no objetivo geral. Lembre-se também que o objetivo deve gerar conhecimento para a área de *computação e informática* e não (apenas) na área de aplicação.

Comparar os vários métodos de identificação de desenvolvedores-chave utilizando o ecossistema dos projetos Java da Apache Software Foundation.

Objetivos específicos da pesquisa: (itens) Os objetivos específicos devem ser subprodutos necessários para atingir o objetivo geral. Os objetivos específicos devem ter resultados finais mensuráveis.

- Identificar os requisitos, dificuldades, pontos fortes e fracos de cada método no contexto dos projetos Java da Apache Software Foundation;
- Comparar os dados necessários e a disponibilidade dos mesmos para a execução de cada método;
- Comparar as listas de desenvolvedores resultantes de cada um dos métodos executados sobre um conjunto de projetos Java da Apache Software Foundation;

6. Contribuições

Contribuições científicas da pesquisa

- Quantificar o *trade-off* (dificuldade de execução *versus* resultados obtidos) e levantar as limitações, pontos fortes e fracos de cada um dos métodos existentes na literatura.

Contribuições tecnológicas da pesquisa

- Construção de um software que possibilite a comparação em larga escala dos resultados de diferentes métodos de identificação de desenvolvedores-chave.

Outras contribuições da pesquisa (diretas e indiretas)

- Não se aplica.

Resultados esperados (o que poderá mudar no mundo após a conclusão da sua pesquisa)

- Futuros trabalhos que analisam as características de desenvolvedores-chave podem utilizar os resultados deste trabalho para escolher o método que seja mais condizente com o contexto analisado e evitar métodos mais complexos que possam apresentar resultados inferiores aos métodos simples.

Quais os principais veículos para publicar os resultados desta pesquisa

- O trabalho será utilizado com o objetivo de adquirir o título de mestre em Ciência da Computação e será parte integrante do acervo do Instituto de Matemática e Estatística da Universidade de São Paulo. O trabalho também pode ser publicado em eventos na área de Engenharia de Software.

7. Desafios

Enumere as dificuldades/desafios encontrados ou a serem encontrados na pesquisa

- O levantamento dos pontos fracos e fortes de cada método empregado;
- A obtenção dos requisitos necessários para a execução de cada método;
- O esforço de execução dos métodos que exigirem mais recursos para que se alcance os resultados.

8. Pesquisa bibliográfica

Em termos conceituais, o que você precisará estudar para esta pesquisa?

- Os vários métodos existentes para a identificação de desenvolvedores chaves;
- Metodologia para comparação e avaliação entre métodos e resultados.

Quais são as queries de busca a serem utilizadas para obtenção dos resultados relevantes para sua pesquisa? Informe o objetivo para cada query (ex: obtenção de trabalhos relacionados, obtenção de evidências do problema etc.)

- "((core*) OR (leader*) OR (key) OR (peripheral)) AND (developer)": obtenção de trabalhos relacionados e métodos de identificação de desenvolvedores chaves.
- Além dos trabalhos relacionados pela query de busca também serão levados em consideração aqueles que já fazem parte do referencial teórico do artigo publicado sobre a caracterização dos desenvolvedores chaves [9] que foi a motivação inicial desse trabalho.

Plano de leituras. Quais serão os artigos/livros que você irá ler em ordem?

- Contributor Turnover in Libre Software Projects;

- Exploring the impact of socio-technical core-periphery structures in open source software development;
- Detecting coordination problems in collaborative software development environments;
- Computing core/periphery structures and permutation tests for social relations data;
- Dependencies in geographically distributed software development: overcoming the limits of modularity;
- Analysis of activity in the open source software development community;
- How do committees invent?
- The social structure of open source software development teams;
- Free/Libre open-source software development: What we know and what we do not know;
- Core and periphery in Free/Libre and Open Source software team communications;
- Analysing Software Repositories to Understand Software Evolution;
- Socialization in an open source software community: A socio-technical analysis;
- Shared leadership in the Apache project;
- Communication in open source software development mailing lists;
- Making peripheral participation legitimate: reader engagement experiments in wikipedia;
- Mining version histories to verify the learning process of Legitimate Peripheral Participants;
- Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study;
- The onion patch: migration in open source ecosystems;
- Coordination in collective intelligence: the role of team structure and task interdependence;
- Evolution of developer social network and its impact on bug fixing process;
- The core and the periphery in distributed and self-organizing innovation systems;
- The true role of active communicators: an empirical study of Jazz core developers;
- Understanding the role of core developers in open source software development;
- Leadership in online creative collaboration;
- Socio-technical developer networks: should we trust our measurements?
- A case study of open source software development: the Apache server;
- Two case studies of open source software development: Apache and Mozilla;
- Evolution patterns of open-source software systems and communities;
- The onion has cancer: some social network analysis visualizations of open source project communication;
- Characterizing key developers: a case study with apache ant;
- Evolution of the core team of developers in libre software projects;
- Socio-technical interaction networks in free/open source software development processes;
- Understanding the process of participating in open source communities;
- Seeking the source: software source code as a social and technical artifact;
- An Empirical Study on the Structural Complexity Introduced by Core and Peripheral Developers in Free Software Projects;
- Self-organization versus hierarchy in open-source social networks;
- Mining task-based social networks to explore collaboration in software teams;

- A topological analysis of the open souce software development community;
- Exploration of the open source software community;
- An empirical study on identifying core developers using network analysis;
- Network analysis of OSS evolution: an empirical study on ArgoUML project;
- What make long term contributors: willingness and opportunity in OSS community;
- Growth of newcomer competence: challenges of globalization;
- Models of core/periphery structures;
- The Mythical Man-Month: Essays on Software Engineering;
- The hidden power of social networks: Understanding how work really gets done in organizations;
- On the relationship between software dependencies and coordination: field studies and tool support;
- The development of social network analysis;
- Collective intelligence in action;

Quais são os especialistas ou companhias (que vivenciam o problema, utilizam o objetivo de estudo em questão ou são potenciais usuárias dos resultados desta pesquisa) que podem ser consultados para acompanhar a pesquisa?

- Outros integrantes do grupo LAPPESC: Gustavo Ansaldi Oliva, Igor Steinmacher e Igor Wiese que trabalham com análises de redes em diferentes contextos, tais como: *callgraph* para analisar dependências em arquivos, redes para estudar novatos, e rede sócio-técnicas para predição de bugs.
- Marcelo Serrano Zanetti, membro do grupo ETHz da universidade de Zurique. O grupo tem diversos trabalhos com análise de redes sociais e análises de comunidades.
- Pesquisadores de Engenharia de software que utilizam os grupos de desenvolvedores chaves em suas pesquisas.

Referências:

[1] Chintan Amrit and Jos Van Hillegersberg. Detecting coordination problems in collaborative software development environments. *Information Systems Management*, 25(1):57–70, 2008.

[2] Chintan Amrit and Jos van Hillegersberg. Exploring the impact of socio-technical core-periphery structures in open source software development. *Journal of Information Technology*, 25(2):216–229, 2010.

[3] Kevin Crowston, Kangning Wei, Qing Li, and James Howison. Core and periphery in free/libre and open source software team communications. In *System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on*, volume 6, pages 118a–118a. IEEE, 2006.

[4] Anja Guzzi, Alberto Bacchelli, Michele Lanza, Martin Pinzger, and Arie van Deursen. Communication in open source software development mailing lists. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 277–286, Piscataway, NJ, USA, 2013. IEEE Press.

[5] Chris Jensen and Walt Scacchi. Role migration and advancement processes in ossd projects: A comparative case study. In *Proceedings of the 29th international conference on Software Engineering*, ICSE '07, pages 364–374, Washington, DC, USA, 2007. IEEE Computer Society.

[6] Corey Jergensen, Anita Sarma, and Patrick Wagstrom. The onion patch: migration in open source ecosystems. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, ESEC/FSE '11, pages 70–80, New York, NY, USA, 2011. ACM.

[7] Karim R Lakhani. *The core and the periphery in distributed and self-organizing innovation systems*. PhD thesis, Massachusetts Institute of Technology, 2006.

[8] Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki Nishinaka, Kouichi Kishida, and Yunwen Ye. Evolution patterns of open-source software systems and communities. In *Proceedings of the international workshop on Principles of software evolution*, pages 76–85. ACM, 2002.

[9] Gustavo A Oliva, Francisco W Santana, Kleverton CM de Oliveira, Cleidson RB de Souza, and Marco A Gerosa. Characterizing key developers: a case study with apache ant. In *Collaboration and Technology*, pages 97–112. Springer, 2012.

[10] Antonio Terceiro, Luiz Romario Rios, and Christina Chavez. An empirical study on the structural complexity introduced by core and peripheral developers in free software projects. In *Proceedings of the 2010 Brazilian Symposium on Software Engineering*, SBES '10, pages 21–29, Washington, DC, USA, 2010. IEEE Computer Society.

[11] Jin Xu, Yongqin Gao, Scott Christley, and Gregory Madey. A topological analysis of the open souce software development community. In *System Sciences, 2005. HICSS'05.*

Proceedings of the 38th Annual Hawaii International Conference on, pages 198a–198a. IEEE, 2005.

[12] Wen Zhang, Ye Yang, and Qing Wang. An empirical study on identifying core developers using network analysis. In *Proceedings of the 2nd international workshop on Evidential assessment of software technologies*, EAST '12, pages 43–48, New York, NY, USA, 2012. ACM.

[13] Minghui Zhou and Audris Mockus. Growth of newcomer competence: challenges of globalization. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, FoSER '10, pages 443–448, New York, NY, USA, 2010. ACM.

[14] Minghui Zhou and Audris Mockus. What make long term contributors: willingness and opportunity in oss community. In *Proceedings of the 2012 International Conference on Software Engineering*, ICSE 2012, pages 518–528, Piscataway, NJ, USA, 2012. IEEE Press.