O presente documento apresenta o mapeamento sistemático conduzido, nas fases iniciais da pesquisa que embasou o desenvolvimento de uma ferramenta de apoio à rastreabilidade no contexto de evolução de software.

Sumário

Introdução

Segundo [1] um mapeamento sistemático é projetado para fornecer uma visão geral de uma área de investigação e serve de base para se identificar áreas afins para a condução de revisões sistemáticas.
Um protocolo de mapeamento, por sua vez, é o método através do qual o pesquisador define a formalização de buscas sobre a área de interesse alvo de investigação [2].
Este mapeamento foi aplicado seguindo três etapas: (i) o planejamento onde se elaborou o protocolo com os dados a serem observados, (ii) a execução do mapeamento propriamente dito, e (iii) a apresentação dos resultados. Uma visão geral do processo aplicado no mapeamento pode ser visto na Figura 1.

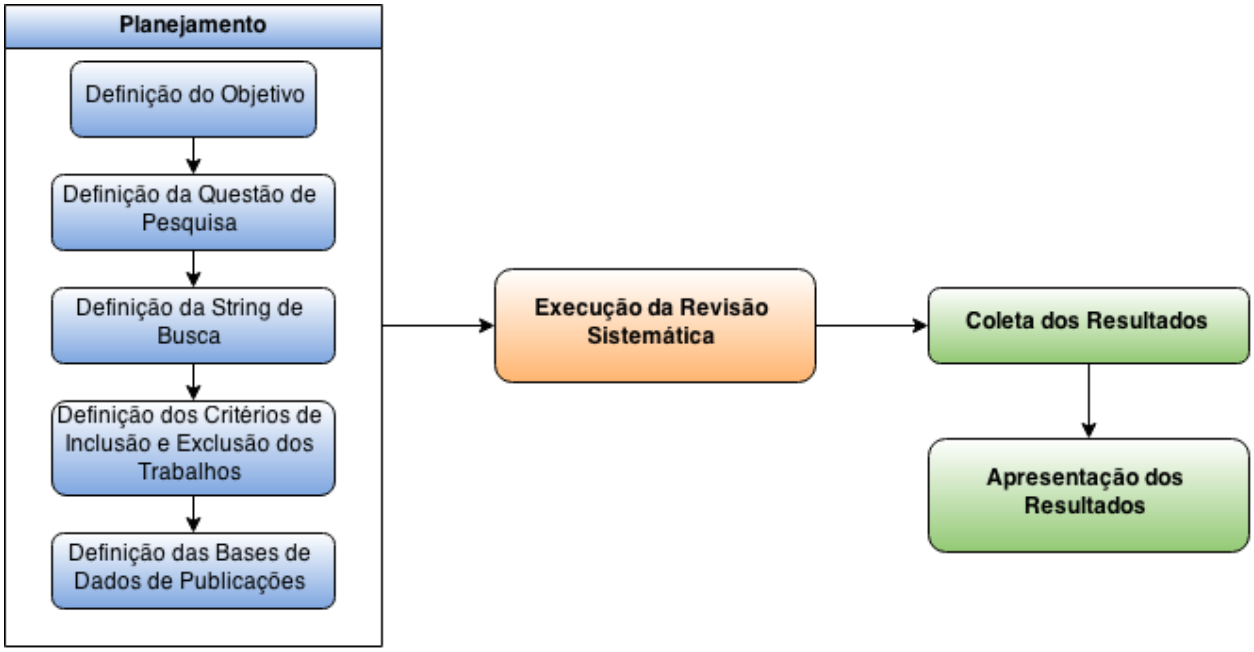

Figura 1: Visão geral do processo de mapeamento

Planejamento do Mapeamento Sistemático

O protocolo desenvolvido foi constituído de objetivo, questão de pesquisa, *string* de busca, bases de dados de publicações e os critérios de inclusão e exclusão.

Objetivo: levantar ferramentas, contempladas pela literatura, que oferecem suporte à geração de rastreabilidade entre código fonte e suas diferentes versões, além dos benefícios obtidos com o uso de dados de rastreabilidade na manutenção enquanto o software evolui.

Questões de Pesquisa: Mediante o objetivo apresentado foram definidas questões de pesquisa, com o objetivo de guiar a execução do mapeamento sistemático. As questões de pesquisa definidas são as seguintes:

(Q1) Quais as ferramentas existentes para promover a rastreabilidade de software entre código fonte e suas diferentes versões?
(Q2) Quais benefícios podem ser obtidos com o uso de dados de rastreabilidade na manutenção enquanto o software evolui?

String de busca: foi definida uma única string de busca genérica, dada a seguir:

(traceability OR rastreabilidade) AND software

Bases de dados de publicações: para executar a string de busca foram consideradas seis bases de dados eletrônicas de publicações com acesso livre para alunos da Universidade Federal de Juiz de Fora (UFJF), acessados via periódicos da Capes, os dados das bases encontram-se registrados na Tabela 1, a seguir:

Bases de Publicações
Science@Direct (http://www.sciencedirect.com)
Compendex (http://www.engineeringvillage.com)
IEEE Digital Library (http://ieeexplore.ieee.org)
ACM Digital Library (http://portal.acm.org)
Web of Science (http://www.isiknowledge.com)
Scopus (http://www.scopus.com)

Tabela 1: Bases de dados de publicações

Critérios de Inclusão: para guiar a seleção dos artigos foram definidos critérios de inclusão que se encontram dispostos na Tabela 2.

Critérios de Inclusão
Estudos sobre ferramentas que geram a rastreabilidade de software no código fonte;
Estudos que discutem os efeitos da rastreabilidade no código fonte
Estudos publicados a partir do ano de 2004

Tabela 2: Critérios de Inclusão

Critérios de Exclusão: para guiar a remoção dos artigos que fogem ao foco de interesse foram definidos critérios de exclusão que se encontram dispostos na Tabela 3.

Critérios de Exclusão
Estudos relacionados a código fonte, mas não, ou só superficialmente, à rastreabilidade;
Estudos que não puderem ser acessados completos de forma gratuita pela web;
Estudos que tenham mais do que uma descrição publicadas serão incluídas apenas uma vez considerando a mais detalhada e atualizada versão do estudo;
Estudos que não tratem da rastreabilidade na área de ciência da computação.

Tabela 3: Critérios de Exclusão

Execução do Mapeamento Sistemático

Para auxiliar o processo de seleção dos artigos nesta etapa foi utilizada uma ferramenta chamada StArt [3], trata-se de uma ferramenta para apoio ao processo de revisão sistemática, capaz de gerenciar artigos dividindo-os pelas bases de publicações escolhidos, e através das classificações de aceito, rejeitado, ou aguardando classificação, além disso dentre outras funcionalidades, indica os artigos duplicados e possibilita a exclusão automática destes.

Após a definição do protocolo, a string de busca definida foi executada nas bases de publicações (Tabela 1), sendo inicialmente aplicada ao título dos artigos. Vale lembrar que o mapeamento foi realizado no mês de novembro de 2014.

Dos 423 retornos, uma primeira análise, automática, feita pela ferramenta StArt, classificou 150 registros como duplicados, esta classificação foi verificada e aprovada pelo condutor do mapeamento. Nos 273 restantes observou-se a necessidade de uma nova análise, neste caso manual, em busca de outros resultados duplicados, desta segunda análise foram identificados 118 duplicatas usando como critério de análise a comparação entre os títulos, datas de publicação e as fontes.

A partir dos 154 trabalhos restantes, iniciou-se o processo de análise do resumo de cada trabalho. Da leitura do “abstract” foram selecionados 34 artigos, que após a análise do texto completo, 8 foram os resultantes.

Ao final do processo de execução do mapeamento, foi possível relacionar os resultados selecionados respondendo as questões de pesquisa. A análise dos resultados e demais considerações são feitas na próxima seção.

Resultados

Os artigos resultantes foram categorizados com o intuito de responder as questões de pesquisa, o que é exibido a seguir:

Resultados da Questão de Pesquisa (Q1)

Realizar uma gerência efetiva dos artefatos durante a evolução do software e suas inúmeras mudanças ao longo do tempo é imprescindível, e trata-se de uma das atividades previstas pela Gerência de configuração. A rastreabilidade pode ajudar nesta tarefa. Enquanto a gerência de configuração provê meios de gerenciar, controlar, executar as mudanças e evolução do software, a rastreabilidade ajuda no gerenciamento de dependências entre os artefatos relacionados às diferentes fases do ciclo de vida do desenvolvimento. No entanto, na maioria das vezes essas duas práticas trabalham isoladamente. Mohan et al. [4] apontam dois problemas nas abordagens de gerenciamento de mudanças, o primeiro diz respeito a falta de suporte ao processo de conhecimento, que pode rastrear dependências entre artefatos em diferentes fases do ciclo de vida de desenvolvimento, e o segundo é a falta de suporte para o gerenciamento do produto e do processo de conhecimento num grau mais fino de granularidade. Então desenvolveram um modelo de referência para relacionar a rastreabilidade com o SCM. Numa segunda fase, uma ferramenta foi desenvolvida, denominada *Tracer*, que suporta a aquisição e uso do conhecimento de processo e produto representado no modelo de referência. Esta ferramenta suporta a criação de uma rede de rastreabilidade que representa a associação entre os diversos componentes de conhecimento originados de diferentes ferramentas usadas por desenvolvedores de software. *Tracer* suporta o MS Visual SourceSafe®, uma ferramenta de controle de versão de uso geral. Para tratar o problema de granularidade, Tracer facilita a documentação de mudanças implementadas em elementos específicos dentro das versões de vários artefatos de software e facilita a ligação entre as mudanças efetivadas e a solicitação que as originou. No relacionamento entre as mudanças o foco é dado na relação com os documentos descritivos e modelos de projeto, podendo ser feito em um nível mais fino de granularidade, por exemplo, descrição de um requisito específico ou um caso de uso de um diagrama UML pode ser ligado a um pedido de mudança para versões específicas deste artefato na ferramenta SourceSafe. Os conhecimentos gerados auxiliam na retenção de parte do conhecimento tácito que se perde na rotação de funcionários, além de ser útil em outras decisões de projeto semelhantes, acelerando o processo de decisão. Em essência, este conhecimento pode levar ao desenvolvimento de melhores práticas gerais que

podem ser compartilhadas entre projetos. A ferramenta suporta apenas um sistema de controle de versão.

A evolução do software com foco nos modelos de projeto é assunto de outros trabalhos também. A ferramenta EvolTrack, por exemplo, que captura e comunica, com a mínima intervenção humana, cada contribuição feita para um projeto de software específico. No caso, a contribuição significa qualquer ação, resultando em uma evolução do software. EvolTrack pode ser implantado em configurações distribuídos. A comunicação é feita, usando EvolTrack, para todos os membros da equipe e é alcançado ao mesmo tempo, mostrando a todos o projeto a partir de cada contribuição individual. Na verdade, ele mantém o controle de todos os projetos intermediários gerados até o mais atual, o que permite ao usuário navegar, se necessário, através de toda a história da evolução. Além disso, fornece alguns recursos que aumentam a consciência do que foi alterado de uma evolução para o outro e, com o recurso de zoom, também permite trabalhar com grandes projetos. Usando essa abordagem, um desenvolvedor pode estar ciente do estado atual do software como um todo e pode, adicionalmente, verificar se o projeto atual, também chamado projeto emergente, está evoluindo de acordo com as expectativas da equipe e a orientação do líder, prevenindo problemas causados por mal-entendidos da solução de software esperada. As mudanças são extraídas de repositórios *SVN* ou *CVS* e convertidas em uma representação, indicando, por exemplo, que o desenvolvedor “Bruno” alterou o diagrama de classe acrescentando mais um método, e alterando os parâmetros de outro método, ou seja, alcançando fina granularidade dos dados, principalmente quem realizou as alterações. Assim minimamente, o desenvolvedor que criou o modelo, aquele que alterou o modelo e o líder da equipe devem, receber um aviso de que aquele modelo sofreu alteração. O suporte a repositórios de versão do sistema GIT não foi implementado, ficando como uma tarefa futura.

Analisar repositórios das alterações feitas ao código fonte é foco de outras pesquisas também. A extração de *links* de rastreabilidae através das versões dos diferentes artefatos, e consequentemente do software como um todo, é uma realidade. A mineração das informações contidas nos bancos de dados de versões em busca do chamado “padrão de alteração”, foi a base que sustentou o trabalho de Kagdi et al. [5]. Segundo eles estes padrões, ao indicarem quais artefatos foram alterados juntamente com outros repetidas vezes, é possível inferir a existência de algum tipo de relação entre eles. Para demonstrar sua técnica, a mineração do histórico de versões encontradas em repositórios de software que são mantidos sob ferramentas de controle de versão tais como *Subversion* e *CVS*. Foram usadas duas técnicas de mineração, uma para identificar e analisar conjuntos de artefatos que foram “commitados” juntos, desta forma produzindo os padrões de alteração, e a outra técnica de mineração aplicada para considerar a ordem cronológica entre os diferentes padrões. Assim os padrões recuperados deram a ordem específica na qual os arquivos em um padrão foram alterados. Esta informação ordenada pode ser usada para inferir o tipo, a direção e os artefatos envolvidos na relação de rastreabilidade. Foi possível comprovar a alta precisão na predição dos links caso padrões similares tenham sido encontrados em versões anteriores. Apesar do resultado positivo, o autor salienta que uma análise numa granularidade mais fina, no nível de classe ou método, poderia produzir melhores resultados.

Em grandes empresas preocupadas com a qualidade de seus serviços e produtos, que seguem padrões e normas para estabelecer esta qualidade interna, a rastreabilidade serve como importante parâmetro de controle. Pequenas e médias empresas também devem atentar para a qualidade onde a rastreabilidade pode e deve ser usada. APIS [6] é um ambiente implementado numa empresa Austríaca de pequeno porte, que com técnicas simples, gera rastreabilidade entre artefatos. A empresa já adotava algumas técnicas que favoreceram a implantação do sistema proposto, por exemplo, os desenvolvedores usam os ids dos artefatos armazenados em comentários para explicar as mudanças sofridas. Além de convenções de nomenclatura, e um repositório central *CVS*. Foi identificado que a implantação do sistema traria significativas melhorias, como a compreensão do software, a análise de impacto das mudanças, e facilidade em se relacionar com sistemas legados. APIS seguiu então duas premissas funcionar como um grande banco de dados (*data warehouse*) aproveitando as técnicas já implantadas, todos os dados já gerados e mantidos pela empresa de modo a adequar aos processos já existentes. A segunda premissa baseou-se na utilização dos links de rastreabilidade através de um sistema de busca simples com suporte a consulta (query), cuja interface permite consultar informações de rastreamento e navegar entre os requisitos, documentação, tabelas de banco de dados, o código fonte, versão *logs*, e testes de unidade. Nas versões de *log* é possível, com a utilização de hyperlinks, navegar até o código implementado. No suporte ao código fonte é possível verificar o código originado por um requisito. Apesar do avanço já alcançado com o uso da ferramenta, no período de publicação do estudo ela não estava finalizada deixando algumas lacunas em aberto. No entanto o autor expõe as lições aprendidas como o uso de ferramentas acompanhadas por processos apropriados, a introdução e evolução gradativa da cultura de rastreabilidade na empresa.

Em [7] é apresentado um algoritmo de geração automática de rastreabilidade. Este algoritmo que determina links entre uma requisição de mudança e entidades de código fonte, a partir de dados do movimento ocular. Sua abordagem consiste em gravar em vídeo, as sessões dos desenvolvedores tratando as solicitações e num segundo momento, os dados gerados são passados ao algoritmo apresentado resultando finalmente em uma entidade de ligação entre a solicitação de mudança e o que efetivamente foi alterado durante o período de resolução. O processo implica que a ferramenta de captura de vídeo seja inicializada, durante o processo de resolução do caso, o movimento dos olhos do desenvolvedor e o tempo de permanência fixado numa dada posição X,Y da tela eram monitorados por uma ferramenta que gerava um arquivo condensando estes dados que eram arquivados. Este mapeamento era feito para todos os artefatos tratados pelo desenvolvedor conseguindo um nível de granularidade bem fino. Em seguida, o algoritmo proposto acessava os dados arquivados e os processava comparando o código fonte depois da alteração com as posições no arquivo nas quais o desenvolvedor fixou mais o olhar. Segundo o autor, são necessárias melhorias na precisão mas os resultados são promissores. Para alcançar o objetivo do trabalho foi necessário o uso de um software especial e específico para a captura do movimento dos olhos. Além disso, houve a necessidade de várias sessões para a coleta das informações.

Resultados da Questão de Pesquisa (Q2)

Em recente estudo [8], Javed e Zdun apresentaram uma revisão sistemática da literatura realizada com o objetivo de descobrir abordagens e ferramentas existentes para a rastreabilidade entre arquitetura de software e o código fonte. Após a execução da revisão, os resultados foram divididos em seis categorias principais para distinguir as diferentes abordagens e ferramentas encontradas. As categorias foram as seguintes: (i) abordagens de rastreabilidade entre arquitetura de software e código fonte, (ii) nível de automação das abordagens, (iii) tipo das relações de rastreabilidade, (iv) a granularidade suportada, (v) a direção da rastreabilidade e (vi) representação das informações de rastreabilidade. Como resultado, propôs um esquema de classificação baseado nos vários aspectos da rastreabilidade, traz ainda importantes conceitos e uma percepção direta das estratégias de solução para tratar a rastreabilidade de software além de um diagnóstico deste cenário de pesquisa em franca expansão.

A rastreabilidade pode ser associada a todos os artefatos gerados ao longo do ciclo de vida do software, desde as fases iniciais de levantamento dos requisitos do software. Após o software ser entregue inicia a fase de manutenção. Nesse sentido, Jaber et al. [10] mostram através de um experimento que quando se utiliza a rastreabilidade durante o processo de manutenção, o resultado é cerca de 86% mais preciso.

Outro experimento a fim de averiguar se os desenvolvedores se beneficiariam com a ajuda da rastrebailidade durante a navegação pelo código fonte durante as atividades de manutenção, levaram, Mader e Egyed[11] a desenvolverem um experimento no qual 52 indivíduos deveriam realizar tarefas reais de manutenção em dado projeto. Para metade dos indivíduos, foi fornecido também um suporte de navegação automatizada baseado em informações de rastreabilidade com a capacidade de mostrar onde um requisito afetado pela mudança foi implementado, enquanto a outra metade de indivíduos, não pode contar com este suporte. Uma tarefa de mudança consiste em três atividades, inicialmente entender a requisição de mudança, em seguida procurar no código fonte os pontos relevantes para realizar a mudança e, por fim, implementar a alteração. A busca no código pelos pontos a alterar (segunda atividade), poderia se beneficiar muito com a navegação automática baseada em rastreabilidade. Os resultados mostraram que aqueles desenvolvedores cuja navegação em rastreabilidade foi oferecida, desconsideraram o uso da navegação convencional, e o novo tipo de navegação proposta teve impacto profundo no desempenho, qualidade, e no fluxo de trabalho de como eram trabalhadas as solicitações de mudança.

REFERÊNCIAS

[1] Kitchenham, Barbara A. and Charters, S. Guidelines for performing systematic literature reviews in software engineering. Technical report, EBSE Technical Report EBSE-2007-01, 2007.

[2] KITCHENHAM, Barbara. Procedures for performing systematic reviews.Keele, UK, Keele University, v. 33, p. 1-26, 2004.

[3] StArt: http://lapes.dc.ufscar.br/tools/start_tool. Acessado em 30 de dezembro de 2014.

[4] Mohan, K., Xu, P., Cao, L., & Ramesh, B. 2008. Improving change management in software development: Integrating traceability and software configuration management. *Decision Support Systems*, *45*(4), 922-936.

[5] Kagdi, H., Maletic, J. I., & Sharif, B. 2007. Mining software repositories for traceability links. In *Program Comprehension, 2007. ICPC'07. 15th IEEE International Conference on* (pp. 145-154). IEEE.

[6] NEUMULLER, Christian; GRUNBACHER, Paul. Automating software traceability in very small companies: A case study and lessons learne. In:Automated Software Engineering, 2006. ASE'06. 21st IEEE/ACM International Conference on. IEEE, 2006. p. 145-156.

[7] Walters, B., Shaffer, T., Sharif, B., & Kagdi, H. 2014. Capturing software traceability links from developers' eye gazes. In *Proceedings of the 22nd International Conference on Program Comprehension* (pp. 201-204). ACM.

[8] Javed, Muhammad Atif, and Uwe Zdun. "A systematic literature review of traceability approaches between software architecture and source code."*Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2014.

[9] de Cepêda, R. D. S. V., Magdaleno, A. M., Murta, L. G. P., & Werner, C. M. L. 2010. EvolTrack: improving design evolution awareness in software development. *Journal of the Brazilian Computer Society*, *16*(2), 117-131.

[10] JABER, Khaled; SHARIF, Bonita; LIU, Chang. A Study on the Effect of Traceability Links in Software Maintenance. Access, IEEE, v. 1, p. 726-741, 2013.

[11] MADER, Patrick; EGYED, Alexander. Do software engineers benefit from source code navigation with traceability?--An experiment in software change management. In: Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering. IEEE Computer Society, 2011. p. 444-447.