Apoio a Atividades de Análise de Impacto em Desenvolvimento Distribuído de Software

Mayara C. Figueiredo, Cleidson R. B. de Souza
Faculdade de Computação – Universidade Federal do Pará
mayfigueiredo@gmail.com, cdesouza@ufpa.br

Abstract. Traditional software development activities are regarded as more difficult in distributed settings than in collocated ones. One of these activities is change impact analysis. Challenges in impact analysis in distributed contexts are related to communication, collaboration and, especially, information sharing among team members. To properly support this scenario, it is necessary to understand the different ways in which impact analysis activities are performed: from an organizational and from a developer's point of view. The first view is important for managers to understand the impact of the changes in the activities of the team, while the second refers to the developers' strategies to deal with impact. This paper discusses these views and based on them describes a tool to support impact analysis activities in distributed settings. Our tool, called Wolf, automatically generates traceability links, which are used to support impact analysis activities. We adopted an approach based on information visualization to facilitate impact analysis activities among distributed software developers.

Key words: change impact analysis, coordination, communication, distributed software development.

1 Introdução

Atualmente, não é raro encontrar projetos de software cujo desenvolvimento é distribuído globalmente. Neste cenário, a colaboração entre os *stakeholders* se torna ainda mais importante, especialmente entre os engenheiros de software. Além disso, desenvolvimento distribuído de software (DDS) é considerado mais difícil que engenharia de software co-localizada [1]: muitas atividades são mais difíceis de coordenar devido à distribuição, incluindo a análise de impacto, que envolve a identificação do impacto decorrente de uma requisição de mudança.

É possível separar dois aspectos diferentes das atividades de análise de impacto. O primeiro aspecto foca na análise de impacto do ponto de vista organizacional. Essa visão consiste principalmente em identificar os itens e engenheiros de software impactados e oferecer a eles informações sobre a mudança. O segundo ponto de vista é baseado na visão individual dos desenvolvedores (ver [2]). Este relaciona-se com as estratégias destes desenvolvedores para lidar diariamente com o impacto de mudanças no seu trabalho. Essas duas visões são importantes porque são complementares e envolvem aspectos críticos de projetos DDS: comunicação e coordenação [2]. Na visão organizacional, estes aspectos são importantes para difundir informações sobre as

mudanças para as pessoas certas [3], e na visão individual eles são essenciais para o trabalho diário dos engenheiros de software [2].

Este artigo descreve a ferramenta Wolf, que, baseada em informações de rastreabilidade, provê apoio aos dois aspectos das atividades de análise de impacto: organizacional e individual. Mais precisamente, ela facilita o fluxo de informações entre os engenheiros de software, pois identifica automaticamente os desenvolvedores de software que serão potencialmente impactados por uma mudança. Ela também provê informações sobre a localização desses engenheiros, e portanto permite que o gerente avalie o custo de coordenação para realizar esta mudança.

O restante do artigo está organizado da seguinte forma. A seção 2 apresenta uma visão geral sobre análise de impacto, incluindo a visão organizacional e individual. A seção 3 apresenta a ferramenta Wolf que oferece apoio à análise de impacto em ambientes distribuídos. A seção 4 apresenta um exemplo de uso da ferramenta baseado em dados reais. Finalmente, a seção 5 apresenta as conclusões e trabalhos futuros.

2 Análise de Impacto

Mudanças sempre ocorrem em projetos de desenvolvimento de software. Dellen [3] argumenta que problemas em gerenciar requisições de mudanças são uma das razões pelas quais os projetos excedem tempo e custo estimados. Quanto mais mudanças são realizadas, evitar defeitos se torna mais difícil e passível de erro, pois uma mudança sempre traz mudanças adicionais (efeitos em cascata) que podem não ser notadas até tarde no projeto, gerando atrasos, custos extras e problemas de coordenação [4].

A análise de impacto (AI) é a atividade chave para analisar (possíveis) mudanças e identificar os artefatos afetados por elas [5]. O padrão IEEE para manutenção de software [6] diz que esta deve: i) identificar potenciais efeitos em cascata, ii) permitir negociação entre as abordagens sugeridas para as mudanças de software, iii) ser realizada com ajuda da documentação extraída do código fonte, e iv) considerar o histórico de mudanças anteriores, tanto as bem sucedidas quanto as mal-sucedidas. A AI é usada para determinar o escopo de uma requisição de mudança, por exemplo, o conjunto de artefatos que precisam ser modificados para atender a esta requisição, para assim planejar a modificação a ser feita e analisar e justificar o custo/benefício da requisição. Para realizar de forma correta esta atividade é fundamental gerenciar as dependências do projeto, pois é através destas que é possível identificar os itens impactados [4]. Estas dependências são estudadas pela área de pesquisa conhecida como rastreabilidade de software, que é "a habilidade de relacionar artefatos de projetos uns com os outros, com os *stakeholders* que contribuíram para a criação deles e com as razões que explicam suas formas" [7]. Neste artigo, os artefatos que podem ser rastreados são chamados de itens de rastreabilidade e as relações entre estes artefatos são chamadas de *links* de rastreabilidade.

É possível identificar dois aspectos diferentes da AI. O primeiro lida com a AI de um ponto de vista organizacional, enquanto o segundo é baseado em uma visão individual dos desenvolvedores de software sobre as mudanças e seus efeitos [2]. A primeira visão consiste principalmente em *identificar os itens e os engenheiros de software impactados e oferecer aos engenheiros informações sobre a mudança*. Em outras palavras, este aspecto está alinhado com os objetivos gerais da organização. A segunda visão está

relacionada às estratégias dos engenheiros de software para lidar no seu dia-a-dia com o impacto de (possíveis) mudanças nos seus trabalhos, ou seja, está relacionada com os objetivos dos indivíduos que realizam as atividades de AI: os engenheiros de software. As sub-seções seguintes irão detalhar essas duas perspectivas da AI.

2.1 A visão *organizacional* da análise de impacto

A maioria dos estudos sobre AI se concentra nessa visão que tem o foco nos times e organizações [2]. Por exemplo, existem modelos de processo para lidar com mudanças, conforme descrito em [5]. Tais modelos baseiam-se na criação de conjuntos de impacto. O primeiro, chamado Conjunto de Impacto Inicial, contém os itens afetados pela mudança, que são os primeiros identificados, e é formado através da análise da especificação de mudanças e do código e documentação do sistema. Através da análise desses itens, são identificados impactos adicionais que, junto com os itens do Conjunto Inicial, formam o Conjunto Candidato. Como as atividades de AI são um processo iterativo, novos itens impactados podem ser descobertos durante a implementação da mudança (Conjunto de Impacto Descoberto), enquanto alguns itens podem ser descartados por não precisarem ser modificados: Conjunto de Impacto Falso-Positivo.

Idealmente, a AI deveria ser capaz de prever corretamente todos os itens impactados, porém isto é muito difícil. Assim, esta atividade tem por objetivo minimizar os conjuntos de impacto descoberto e falso-positivo. Nesse processo é possível distinguir duas fases: análise e execução [3]. A atividade principal da primeira é calcular o custo e o impacto da mudança, enquanto na segunda é implementar esta mudança com custo efetivo e sem trazer problemas para o projeto [3]. Este artigo e a ferramenta apresentada nele tratam da primeira etapa do processo de AI onde a atividade fundamental é gerenciar as dependências do projeto para tornar possível calcular o custo da mudança.

2.2 A visão *individual* da análise de impacto

Enquanto existem diversos estudos sobre a visão organizacional da AI, estudos que focam na visão individual – onde o foco é nos indivíduos que realizam as atividades de AI – são raros [2]. Essa visão é importante, pois os engenheiros de software têm suas próprias estratégias para realizar AI, e conhecer como eles lidam com estas atividades é útil para entender e oferecer apoio para as mesmas, o que deve proporcionar melhorias para todo o processo de AI.

De Souza e Redmiles [2] descrevem as estratégias dos engenheiros de software para lidar com as mudanças e como eles lidam com o efeito dessas mudanças em seus trabalhos. Eles definem um *framework* para gerência de impacto que foca nas atividades dos desenvolvedores de software, ou, "o trabalho realizado pelos desenvolvedores de software para minimizar o impacto do seu trabalho no trabalho dos outros e, ao mesmo tempo, o impacto dos outros no seu próprio trabalho". Existem três aspectos principais da gerência de impacto [2]. O primeiro é encontrar o conjunto de pessoas, a *rede de impacto*, que pode afetar o trabalho de um engenheiro de software e que pode ser afetado pelo trabalho dele. O segundo é chamado *forward impact management* e consiste no trabalho realizado por um engenheiro para estimar o impacto do seu trabalho na sua rede de impacto e informar os outros presentes nesta rede sobre isso. E, finalmente, o terceiro

aspecto é chamado de *backward impact management* e é definido como o trabalho realizado pelos engenheiros de software para evitar o impacto do trabalho de outros desenvolvedores no seu próprio trabalho [2]. Como exemplos desses dois aspectos, De Souza e Redmiles citam, entre outros, enviar e-mails com notificações de mudanças e revisões de código com o objetivo de entender o impacto de mudanças na arquitetura do software como exemplo de *forward impact management*, e ler e-mails de notificações e checagem de erros como estratégias de *backward impact management*. Mais detalhes sobre essas estratégias podem ser encontrados em [2].

O ponto principal desta abordagem é que ela reconhece que os engenheiros realizam várias atividades para lidar com mudanças, entre outras palavras, *eles realizam AI diariamente*. A adoção de estratégias pelos engenheiros de software indica que eles estão cientes das dependências no software, uma vez que estas são a base para eles identificarem a rede de impacto e usam as suas estratégias para minimizar o impacto das mudanças no trabalho dos outros. De fato, alguns desenvolvedores entrevistados por De Souza e Redmiles [2] tinham este conhecimento, porém este não foi o caso de todos os entrevistados. Este conhecimento faltava a novatos devido ao seu curto tempo no projeto, ou quando mudanças nos membros das equipes causavam mudanças nas dependências e desenvolvedores associados. Além disso, a falta de percepção (*awareness*) e de fluxo de informações é o maior problema em ambientes distribuídos [1, 4, 8], assim este conhecimento sobre as dependências de software e a rede de impacto se torna ainda mais importante em projetos de software distribuídos.

3 A Ferramenta Wolf

A ferramenta descrita neste trabalho chama-se Wolf. Ela oferece apoio à criação semi-automática dos links de rastreabilidade, e consequentemente oferece apoio a análise de impacto tanto do ponto de vista organizacional quanto do individual, podendo ser usada tanto por gerentes quanto por engenheiros de software. *Links* de rastreabilidade são gerados pela Wolf através de um *parser* da documentação do projeto que deve ser armazenada em um repositório Subversion. Os itens e *links* de rastreabilidade são armazenados em um banco de dados, e essa informação é usada para apoiar a AI. Mais detalhes sobre esta etapa podem ser vistos em [9].

O apoio à AI organizacional e individual pode ser feito graças às duas visões que a ferramenta oferece: na primeira, o gerente pode ter uma visão mais completa do projeto e acessar várias informações sobre o mesmo que são relevantes para ambientes distribuídos, e dessa forma ele pode estimar o impacto levando-as em consideração. Já a segunda visão é direcionada para engenheiros de software e usa os aspectos descritos em [2] para oferecer informações que podem influenciar o trabalho dos mesmos e podem ajudar nas atividades diárias de gerência de impacto.

A ferramenta Wolf foca na identificação automática das dependências entre desenvolvedores através dos *links* de rastreabilidade. Baseado nessa informação, ela gera a rede de impacto usando uma abordagem baseada em redes sociais [11]. Com a identificação automática da rede de impacto é possível oferecer um apoio melhor para as atividades de desenvolvimento de software, especialmente em contextos distribuídos.

A Figura 1 mostra a tela principal da Wolf com quatro diferentes regiões marcadas. A região 1 contém os comandos principais da interface, na região 2 o usuário pode escolher os itens do projeto que possivelmente serão modificados e pode visualizar o impacto da

mudança através da árvore de impacto. A região 4 contém o grafo de dependências dos itens de projeto enquanto que a região 3 traz informações adicionais sobre os itens ou autores. A região 2 possui 3 painéis. O primeiro (“Item Types”) é um filtro para os itens de projeto por seus tipos. O segundo painel (“Items”) mostra os itens do tipo escolhido no primeiro painel. Nesta área, o usuário pode escolher os itens que deseja ver o impacto se uma mudança ocorrer. O último painel (“Impacted Items”) mostra os itens que serão impactados se o item escolhido for modificado. Os itens impactados são mostrados em uma árvore que especifica todos os itens que dependem de cada item selecionado. A árvore é composta por *check boxes* que aparecem todos selecionados, mas o usuário pode desmarcar qualquer um sempre que for necessário.

Na região 1 existe o painel “Filters” onde o usuário pode escolher se ele deseja ver ou não no grafo os nós correspondentes aos autores dos itens. O usuário também pode escolher ver apenas os nós impactados no grafo. Esta funcionalidade é útil quando o projeto possui muitos itens, pois nesse caso, o grafo se torna muito denso. Esta é uma maneira de tornar a visualização mais efetiva. Ainda na região 1 existe também um botão (“Generate Report”) para gerar o relatório de AI. Essa funcionalidade está presente apenas na visão geral: somente o gerente pode gerar este relatório e é necessário efetuar o login para acessar esta funcionalidade. Um assunto importante relacionado à AI é o fato de os *stakeholders* impactados serem informados disso tarde ou, no pior caso, não serem informados [2]. Para oferecer uma solução para este aspecto, quando o gerente gera o relatório de impacto, um e-mail pode ser automaticamente enviado para os engenheiros impactados, os quais também são identificados. Com isso, todos os envolvidos em uma mudança são informados sobre a mesma.

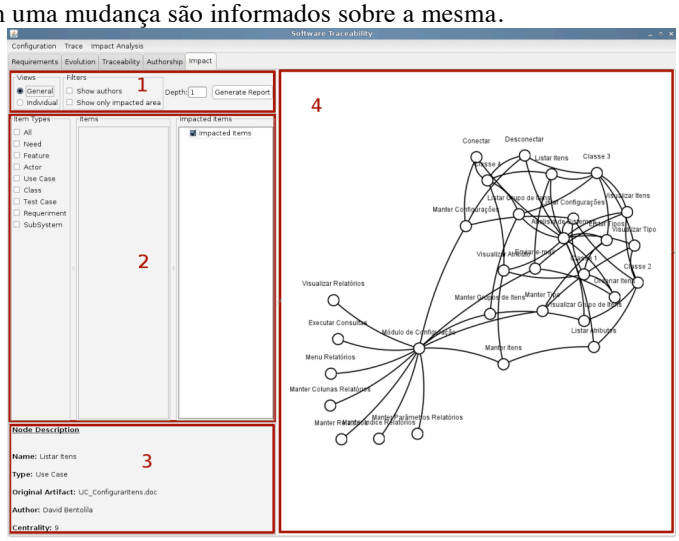

Fig. 1. Regiões da interface principal.

Por fim, a região 3 mostra informações sobre um nó quando o usuário o seleciona. Quando o nó é um item, estas informações são nome, tipo, artefatos originais, autores e centralidade (uma medida de redes sociais) [11]. Quando o nó é um autor, as informações são nome, papel, localização, fuso-horário e centralidade. Essas informações sobre os autores são úteis especialmente em contextos distribuídos [1].

3.1 Visualização do impacto

Para visualizar o impacto devido a uma modificação em um item basta o usuário selecioná-lo na lista de itens do painel "Items" (Figura 1). Quando isso acontece, o grafo na região 4 (Figura 1) é automaticamente atualizado. Na Figura 2, o item "Visualizar Atributo" está selecionado, por isso está em vermelho. Os itens em laranja são os impactados por uma modificação no item vermelho.

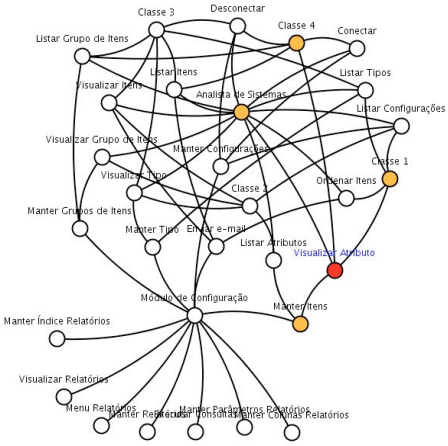

Fig. 2. Utilizando cores para visualizar informações sobre impacto.

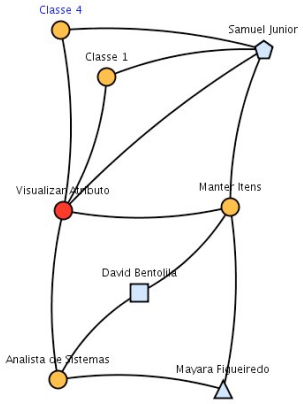

Fig. 3. Grafo com autores e apenas nós impactados.

A Figura 3 apresenta um grafo representando o mesmo impacto da Figura 2, mas apresentando os nós dos autores e apenas os nós impactados. Os nós dos autores são sempre representados em azul e possuem formas diferentes de acordo com a localização deles. Essa diferenciação por formas é útil em ambientes distribuídos, pois permite uma rápida visão de como a distância pode influenciar nas atividades de AI. Por exemplo, o requisito "Visualizar atributo" criado por "David Bentolila" será modificado e deve impactar os itens "Classe1", "Classe4", "Manter Itens" e "Analista de Sistemas". Os desenvolvedores impactados devido a esta mudança são "Mayara Figueiredo" e "Samuel Junior", os engenheiros associados aos artefatos impactados. Pelo grafo é possível identificar que eles estão em locais diferentes de David, uma vez que as formas dos seus nós são diferentes. Através desta informação, a Wolf permite que o gerente, ou desenvolvedor, identifique a complexidade de uma mudança: quantos artefatos e desenvolvedores serão potencialmente afetados e onde estes estão localizados, sugerindo assim uma maneira de inferir o custo de coordenação da mudança.

3.2 Visão individual

A descrição da seção anterior usou como base a visão organizacional da Wolf, mas se um engenheiro de software desejar acessar a visão individual, ele precisará apenas escolher esta opção no painel "Views" na Figura 1. Seu *login* e senha da ferramenta serão usados para customizar a ferramenta para cada engenheiro, pois a interface muda para apresentar informações específicas para cada desenvolvedor, como por exemplo,

identificando os itens do engenheiro que ele estiver utilizando no momento. Para ser mais específico, a primeira diferença está na visualização do grafo: os itens que pertencem ao desenvolvedor que fez o login aparecem em lilás. Outra diferença é a lista de itens que mostra apenas os itens pertencentes ao usuário que efetuou o login. Dessa forma, o usuário pode escolher apenas os seus itens para ver o impacto que pode causar se modificá-los e também quem será impactado. Em outras palavras, a Wolf identifica a rede de impacto de cada desenvolvedor de software.

A diferença da visualização dos itens impactados nas duas visões está nas cores dos nós do grafo na visão individual. Os itens do usuário aparecem em lilás e os nós dos autores continuam em azul. Quando um item é selecionado, ele continua mudando para vermelho, mas os itens impactados por ele apresentam algumas diferenças: os itens impactados que pertencem a outros engenheiros de software continuam em laranja, mas aqueles que pertencem ao usuário que fez o login passam a ser da cor rosa (item "Módulo Configuração" na Figura 4).

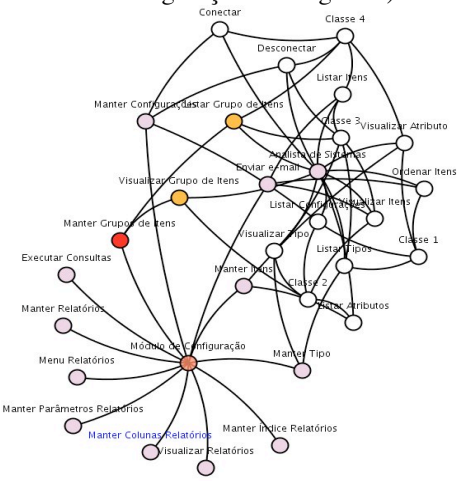

Fig. 4. Impacto na visão individual.

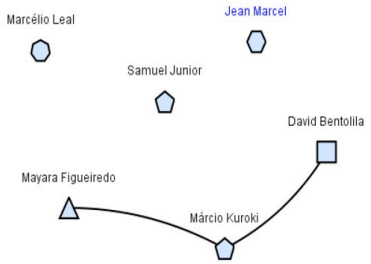

Fig. 5. Um exemplo de rede de impacto.

Na visão individual é possível ver a rede de impacto do usuário para os itens selecionados. Esta rede é gerada baseada na técnica descrita por Cataldo et al [10] que consiste em representar as relações de dependência a partir de operações com matrizes: a primeira, chamada "*Task Assignments*", é uma matriz desenvolvedor por tarefa (um 1 em uma célula *ij* representa que o desenvolvedor *i* está designado para a tarefa *j*). A segunda matriz, "*Task Dependencies*", é uma matriz quadrada onde uma célula *ij* indica que a tarefa *i* e a tarefa *j* são relacionadas [10]. Na Wolf, a matriz "*Task Assignments*" será desenvolvedor por artefato, indicando em qual artefato cada engenheiro está trabalhando. Esta informação é retirada automaticamente do repositório Subversion usando as informações da conta do desenvolvedor. A matriz "*Task Dependencies*", artefato por artefato, será gerada baseada nas informações de rastreabilidade. O próximo passo é multiplicar as duas matrizes, obtendo uma matriz desenvolvedor por tarefa que representa o conjunto de artefatos que um desenvolvedor deve estar ciente pois estes estão conectados aos seus artefatos. Por fim, multiplica-se o resultado da primeira multiplicação pela transposta da "*Task Assignments*", obtendo uma matriz desenvolvedor por desenvolvedor onde uma célula *ij* indica que um desenvolvedor *i* trabalha em

artefatos que dependem dos artefatos do desenvolvedor *j*. Isso indica que estes 2 desenvolvedores precisam coordenar as suas atividades para evitar impacto.

Para gerar a rede de impacto, estas matrizes não contém todos os artefatos do projeto, mas apenas aqueles escolhidos pelo usuário e os impactados por eles, assim o usuário pode ver as dependências entre desenvolvedores relacionadas ao conjunto de itens selecionados. A Figura 5 mostra uma rede de impacto gerada pela ferramenta.

A última diferença entre as visões organizacional e individual é a funcionalidade para enviar e-mails. Na visão individual isto é usado para facilitar a comunicação entre os engenheiros de software. Ao usar esta funcionalidade, o engenheiro pode escolher para qual dos engenheiros impactados quer enviar um e-mail. Essa funcionalidade provê o contato inicial entre os engenheiros, algo que pode ser problemático em DDS [1].

A identificação da rede de impacto ajuda na comunicação entre os engenheiros de software, pois identifica com quem o usuário deve coordenar seu trabalho devido às dependências do mesmo. Os aspectos de *forward* e *backward impact management* também são apoiados pela Wolf. O primeiro, o trabalho de avaliar o impacto do trabalho do engenheiro na sua rede de impacto e informar os outros desenvolvedores dessa rede sobre isso, recebe apoio de forma clara, havendo até mesmo uma forma de informar os outros engenheiros por e-mail. O segundo aspecto, o que engenheiros de software fazem para evitar o impacto do trabalho de outros no seu, recebe apoio pela identificação dos itens e pessoas que podem afetar seu trabalho, pois, com esta informação, ele saberá quais itens precisa ficar ciente para evitar impacto no seu trabalho e poderá se comunicar com os desenvolvedores responsáveis por eles para evitar impactos. Esta melhora na comunicação é muito importante em DDS, especialmente em uma atividade tão colaborativa como gerência de mudanças e a AI associada a ela.

4 Exemplo de utilização

A ferramenta Wolf foi testada com dados de um projeto real de desenvolvimento de software para verificar seu comportamento e a geração das redes sociais. O projeto será chamado de MBL por questões de confidencialidade e era responsável por desenvolver uma aplicação para ambientes móveis. Os desenvolvedores foram divididos em grupos, como interface de usuário (UI), sincronização (SYNC), instalação (INST), etc. Além disso, este projeto era desenvolvido em 5 locais diferentes: Raleigh, EUA; Westford, EUA; Pequim, China; Shanghai, China e Taipei, Taiwan. Mais especificamente, o design das interfaces de usuário e a validação foram realizados por 6 pessoas e a implementação foi feita em todos os outros locais. A equipe de qualidade estava dividia entre Estados Unidos (em Westford) e China (em Pequim). A coordenação principal do projeto e o gerente estavam localizados em Westford, onde os dados foram coletados.

Os dados foram coletados através de análise de documentação e de 17 entrevistas semi-estruturadas com membros de todas as equipes distribuídas por todos os locais. Informação geográfica é indicada pela forma (círculo, quadrado, etc) do item no grafo. Mais detalhes podem ser vistos em [12]. Como não se pôde ter acesso aos artefatos nem ao sistema de controle de versões da empresa, o banco de dados da ferramenta Wolf foi preenchido de acordo com as informações coletadas através das análises e entrevistas ao invés de utilizar o mecanismo de rastreabilidade semi-automática disponível na Wolf. A Figura 6 mostra o grafo resultante dos dados deste projeto. O item “User interface

guidelines" está selecionado e seu impacto pode ser visualizado (itens "User interface Specification C", "User Interface Specification B" e "User Interface Specification A").

O item "User Interface Guidelines" foi desenvolvido pelo engenheiro de software chamado de MBL-12. Se este desenvolvedor desejasse ver a rede de impacto para este seu item, bastaria ele acessar a funcionalidade de visualizar a rede de impacto da Wolf. Tal funcionalidade retornaria a imagem da Figura 7, mostrando que os desenvolvedores impactados seriam MBL-19, MBL-01 e MBL-02, todos eles desenvolvedores de interfaces com o usuário. A interpretação da informação é correta, uma vez que uma mudança nas *guidelines* das interfaces com o usuário impactaria em todos os artefatos de interfaces de usuário e, logo, impactaria os desenvolvedores dos mesmos.

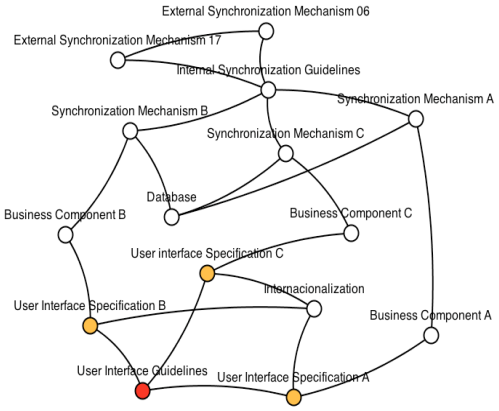

Fig. 6. Exemplo de grafo com dados reais.

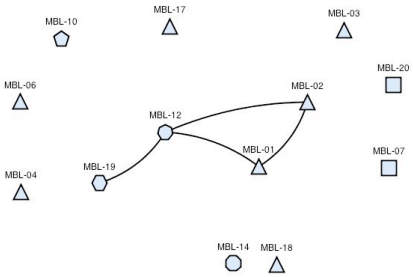

Fig. 7. Rede de impacto para o item "User Interface Guidelines".

5 Conclusões e trabalhos futuros

Este artigo apresentou a Wolf, uma ferramenta para apoio às atividades de análise de impacto que abrange tanto a visão organizacional quanto a individual. A Wolf se torna especialmente útil em DDS, pois oferece apoio à identificação automática dos desenvolvedores impactados por uma mudança e facilita a comunicação e coordenação destes desenvolvedores. A presença das duas visões é um diferencial da Wolf com relação a outras ferramentas, pois a visão individual é um aspecto não explorado [2].

A Wolf usa informações de um repositório Subversion para identificar a autoria dos artefatos e então considera não apenas o processo de software definido, mas também a real divisão do trabalho. Isso é um aspecto importante devido à grande variedade de pessoas que podem estar envolvidas em uma modificação [8] e a maioria das ferramentas que oferecem algum apoio à análise de impacto considera apenas as informações do processo de software. Além disso, essas ferramentas não possuem a geração dos links (tanto de rastreabilidade entre os itens quanto entre autores e itens) de forma semi-automática. Essa automatização de parte do processo de análise de impacto que a Wolf apresenta é importante, pois agiliza esta atividade e a torna menos cansativa e dispendiosa. Em DDS, essa automatização é ainda mais interessante, pois a

coordenação de processos globalmente distribuídos pode ser facilitada quando parte da tarefa é feita automaticamente [Dellen, 2000].

Como trabalho futuro pretende-se avaliar a ferramenta Wolf em outros projetos distribuídos visando identificar aspectos que precisam ser modificados na ferramenta. Além disso, seria interessante implementar uma análise de redes sociais [11], tanto na rede de impacto quanto na própria "rede" dos itens de rastreabilidade. Tal análise forneceria ainda mais subsídios para a análise de impacto em contextos distribuídos.

Agradecimentos

Esta pesquisa tem financiamento do CNPq através do Edital Universal 2008 processo 473220/2008-3, da FAPESPA através do Edital Universal N.° 003/2008 e da IBM através de um IBM UIMA Innovation Award.

Referências

1. Herbsleb, J.D, Mockus, A., Finholt, T.A., and Grinter, R.E., "An Empirical Study of Global Software Development: Distance and Speed", *International Conference on Software Engineering*, 2001, pp.81-90
2. De Souza, C.R.B., Redmiles, D. F., "An Empirical Study of Software Developers Management of Dependencies and Changes", *International Conference on Software Engineering*, 2008, pp. 241-250.
3. Dellen, B., "Change Impact Analysis Support for Software Development Processes", Ph.D. Thesis, Universität Kaiserslautern, Aachen-Germany, 2000
4. Mockus, A., Herbsleb, J.D., "Expertise Browser: A Quantitative Approach to Identifying Expertise", *International Conference on Software Engineering*, Orlando, USA, 2002.
5. Bohner, S.A., "Software Change Impacts - An Evolving Perspective", *Proceedings of the International Conference on Software Maintenance*, 2002, pp. 263- 272
6. IEEE, "IEEE Standard for Software Maintenance", 1992.
7. Spanoudakis, G., Zisman, A., "Software Traceability: A Roadmap", 2004.
8. Kwan, i.; Marczak, S.; Damian, D. "Viewing Project Collaborators Who Work on Interrelated Requirements". Poster Proc. of IEEE Requirements Engineering Conference, 2007
9. Leal, M., Figueiredo, M.C., De Souza, C.R.B., "Uma abordagem semi-automática para a manutenção de links de rastreabilidade", Workshop on Requirements Engineering, Barcelona, Spain, 2008, pp. 47-58
10. Cataldo, M., Wagstrom, P.A., Herbsleb, J.D., Carley, K.M., "Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools", *20th Conference on Computer Supported Cooperative Work*, Alberta, Canada, 2006
11. Wasserman, S.; K. Faust. "Social Network Analysis: Methods and Applications. Structural Analysis in the Social Sciences". 1994, Cambridge, UK: Cambridge University Press.
12. De Souza, C.R.B., T. Hildenbrand, and D. Redmiles. Towards Visualization and Analysis of Traceability Relationships in Distributed and Offshore Software Development Projects. in Software Engineering Approaches for Offshore and Outsourced Development. 2007.