9.
Conclusão

Este capítulo contextualiza e resume esta tese, apresenta as contribuições científicas das soluções pontuais, e descreve, de forma resumida, os estudos de caso desenvolvidos nos trabalhos relacionados aos quatro desafios para a transparência de software. Além disso, discute os trabalhos relacionados, as contribuições científicas e a avaliação da abordagem integrada, as limitações desta pesquisa bem como sugere trabalhos futuros.

9.1.
Contextualização

O Grupo de Transparência de Software da PUC-Rio (GTS 2011a; GTS 2011b) conduziu um estudo para o entendimento e a definição de transparência no contexto de software. Como resultado desse estudo inicial foi desenvolvido o SIG de Transparência (Leite e Cappelli 2010; Aló 2009), que possui 28 metas flexíveis operacionalizáveis. Adaptando-se o método GQM para um método GQO, obteve-se conjuntos de questões que ajudam a identificar operacionalizações para cada uma das metas flexíveis. Em um trabalho mais recente, o SIG de Transparência e as questões foram expressos como padrões de requisitos e transformados em um Catálogo de Transparência de Software.

Entretanto, não existiam estudos de como aplicar o conhecimento desse catálogo para a construção de softwares transparentes. Abordagens encontradas na literatura para a aplicação de catálogos de requisitos não-funcionais (Chung et al. 2000; Supakkul et al. 2010; Cysneiros et al. 2001) concentram-se em atividades a serem realizadas pelo engenheiro de requisitos. Dada a natureza abstrata e subjetiva da transparência de software, seria necessário permitir que os interessados (e não o engenheiro de requisitos) discutissem os requisitos de transparência apoiados pelo conhecimento do catálogo.

Processos de desenvolvimento dirigidos por modelos, como o Tropos (Bertolini et al. 2007; Bresciani et al. 2004), aplicam o conceito de rastreabilidade entre artefatos para gradualmente baixar o nível de abstração dos artefatos, obtendo, por fim, o código. Porém, mudanças de paradigmas, tecnologias e

PUC-Rio - Certificação Digital Nº 0711310/CA

abstrações ao longo desse desenvolvimento não garantem que os requisitos podem ser facilmente identificados no código, o que dificulta a compreensão dos interessados sobre o que o software faz e por que ele o faz. Assim, torna-se necessário aproximar os requisitos do código ou representá-los no código de forma explícita. Na modelagem intencional, por exemplo, ao menos as metas, metas flexíveis e tarefas dos atores deveriam estar presentes de forma explícita no código.

9.2. Resumo

Transparência é um critério de qualidade crítico para sociedades democráticas modernas. Vários autores, como Holzner&Holzner (Holzner e Holzner 2006) e Lord (Lord 2006), debatem a questão da transparência em relação à sociedade, aos governos e às organizações.

Como o software permeia nossas vidas sociais, a transparência se tornou uma preocupação para softwares operando em domínios públicos, sejam eles *eGovernment*, *eCommerce* ou softwares sociais. Alguns trabalhos, como (Meunier 2008) e (Mercuri 2005), abordam a questão da transparência no contexto de software, mas de forma limitada. Esse trabalhos não oferecem, por exemplo, uma definição para a transparência, tampouco apresentam evidências de que as suas visões sobre a transparência de software são consistentes e coerentes. Dessa forma, a transparência de software está se tornando um critério de qualidade que demanda mais atenção dos cientistas e dos engenheiros de software.

Leite e Cappelli (Leite e Cappelli 2010) afirmam que requisitos de transparência em processos organizacionais estão relacionados a requisitos não-funcionais, tais como: Disponibilidade, Usabilidade, Informatividade, Entendimento e Auditabilidade. Entretanto, requisitos de transparência no contexto do software são especialmente difíceis de serem validados devido à natureza subjetiva dos conceitos envolvidos. Os interessados no software não possuem um entendimento claro do que se entende por transparência de software, pois essa é um conceito abstrato e recente. Além disso, as visões sobre transparência por parte de alguns interessados no software podem ser conflitantes

PUC-Rio - Certificação Digital Nº 0711310/CA

em relação às visões dos outros interessados, uma vez que essas visões são relacionadas às próprias expectativas de cada um sobre o software.

Modelos intencionais são fundamentais para a transparência de software, uma vez que associam aos requisitos às metas e aos critérios de qualidade esperados pelos interessados e que justificam as decisões tomadas. Segundo Leite e Cappelli (Leite e Cappelli 2008), o *framework* i* possui várias características que apóiam a transparência de software, como os *rationales* explícitos das decisões de desenho.

Essa tese propõe o desenvolvimento intencional de software transparente dirigido por requisitos de transparência. Os requisitos de transparência são elicitados com o apoio de um catálogo de padrões de requisitos, relativamente validados pelos interessados através do uso de argumentação e representados em modelos intencionais.

Mantém-se o paradigma intencional durante todo o processo de desenvolvimento, inclusive no código. O uso de um mesmo paradigma permitiu a definição de heurísticas transformacionais que auxiliam a produção do código. O sistema é implementado como um sistema multi-agentes intencional, ou seja, com agentes colaborativos que implementam o modelo *Belief-Desire-Intention* e que são capazes de raciocinar sobre metas e critérios de qualidade.

Essa tese discute também as questões cruciais para o sucesso da nossa abordagem de desenvolvimento de software transparente, tais como: (i) rastreabilidade requisitos-desenho-código (*forward*) e código-desenho-requisitos (*backward*); (ii) o uso de lógica nebulosa para desenvolver uma máquina de raciocínio para agentes intencionais; (iii) a aplicação de argumentação para a validação relativa de requisitos de transparência através da obtenção de um consenso entre os interessados; e (iv) pré-rastreabilidade colaborativa para modelos intencionais baseada nas interações sociais. Nossas idéias foram validadas através de estudos de caso em diferentes domínios, tal como computação ubíqua e aplicações *Web*.

PUC-Rio - Certificação Digital Nº 0711310/CA

9.3. Contribuições dos Trabalhos Pontuais

Esta seção está dividida em subseções. Cada subseção discute as contribuições relacionadas aos desafios para a operacionalização da transparência de software.

9.3.1. Anexação de Requisitos ao Código

Podemos mencionar como contribuições científicas do nosso trabalho sobre rastreabilidade entre modelos i* e código de SMA intencional em JADEX: (i) o uso de modelos i* tanto como modelo de requisitos quanto como modelo arquitetural, ao invés de utilizá-lo apenas para modelar os requisitos; (ii) as heurísticas que transformam as abstrações do *framework* i* em abstrações do modelo BDI, isto é, de modelos i* para especificações BDI; (iii) as heurísticas que transformam as abstrações do modelo BDI para abstrações da implementação BDI do JADEX, isto é, de especificações BDI para código de um SMA implementado em JADEX; (iv) a representação de metas flexíveis e contribuições de tarefas para as metas flexíveis como crenças; e (v) o uso das heurísticas como rastros, ou seja, a definição de elos de rastreabilidade como arestas origem-destino a partir das condições e aplicações das heurísticas transformacionais.

Uma contribuição também importante dos nossos esforços consiste em evitar a introdução desnecessária de diferentes notações ou diagramas para capturar a intencionalidade dos atores e agentes. Propomos o uso de modelos i* (modelos SD e SR) para representar os requisitos e os detalhes arquiteturais. Afirmamos ainda que modelos i* capturam essas informações e que não existe a necessidade de se introduzir outras notações ou diagramas.

Em relação às heurísticas transformacionais, oferecemos heurísticas para todas as abstrações do *framework* i*, incluindo as abstrações de papéis, posições e dependências estratégicas. As dependências estratégicas entre atores foram traduzidas para protocolos de interação entre agentes.

Finalmente, podemos argumentar, com base nos diversos estudos de caso desenvolvidos e na aplicação incremental de nossa abordagem para o Lattes-Scholar (Capítulo 7), que estamos na direção correta em termos de: (i) reduzir a quantidade necessária de tipos de modelos para especificar os requisitos e o

PUC-Rio - Certificação Digital Nº 0711310/CA

desenho detalhado; (ii) melhorar a capacidade cognitiva dos agentes intencionais através do uso do modelo BDI e de um apoio baseado em lógica nebulosa, e (iii) conduzir o desenvolvimento de um SMA intencional desde os requisitos até o código.

9.3.2. Raciocínio sobre Metas Flexíveis em Tempo de Execução

Oferecer um tratamento formal para a incerteza e a subjetividade intrínsecas às metas flexíveis foi a principal contribuição dos nossos trabalhos relacionados a explicitar metas flexíveis no código dos agentes intencionais. Esse tratamento formal aplicou lógica nebulosa, uma teoria matemática para lidar formalmente com a incerteza, visando representar as metas flexíveis, as contribuições das tarefas para as metas flexíveis e os impactos que propagam através das contribuições quando se seleciona e executa uma tarefa.

As representações relacionadas às metas flexíveis foram utilizadas para desenvolver um simulador de propagações que replica o trabalho dos engenheiros de requisitos ao usarem regras de propagação para analisar modelos intencionais.

Com base no simulador de propagações, foi desenvolvida uma capacidade JADEX que consiste em uma máquina de raciocínio qualitativa para lidar com metas flexíveis em tempo de execução. A máquina de raciocínio seleciona tarefas em tempo de execução levando em consideração suas contribuições para as metas flexíveis. A capacidade JADEX pode ser reutilizada por qualquer agente intencional que necessite de uma análise qualitativa de metas flexíveis em tempo de execução.

Tomar decisões em tempo de execução em detrimento de em tempo de desenho aprimorou nosso SMA intencional, conferindo aos agentes maior autonomia e maior capacidade cognitiva. Tarefas que eram descartadas em tempo de desenho estão agora presentes no código e podem ser os únicos meios de se atingir uma determinada meta no caso de todas as outras alternativas falharem.

Apesar de existirem várias técnicas e procedimentos para analisar modelos orientados à meta, concentramos nossos esforços na simulação qualitativa – em tempo de execução – de um procedimento comumente utilizado para análise de satisfação. A nossa máquina de raciocínio simula um procedimento bem conhecido e aprovado na comunidade da Engenharia de Software. A teoria

PUC-Rio - Certificação Digital Nº 0711310/CA

matemática de lógica nebulosa foi capaz de nos ajudar na detecção de comportamentos indesejáveis e no "ajuste fino" dos conjuntos nebulosos.

A máquina de raciocínio está disponível como uma capacidade para o JADEX. Esse recurso habilita o agente com uma capacidade de raciocínio maior que a oferecida aos agentes usando a máquina convencional do JADEX, baseada no modelo BDI.

9.3.3. Argumentação sobre Requisitos de Transparência

A principal contribuição do trabalho sobre argumentação está na integração de um *framework* de argumentação (o ACE) com um *framework* intencional de modelagem de requisitos (i*, NFR) visando apoiar a validação dos requisitos. Combinar ACE com i* estendeu o *framework* i* com uma capacidade de argumentação baseada no AIF, mais poderosa do que o operador *claim*. Essa integração é particularmente importante ao lidarmos com requisitos complexos e não bem entendidos, como a transparência.

Outra contribuição relevante dos nossos estudos foi a definição de regras de tradução de SIGs do NFR Framework e de modelos SD e SR do *framework* i* para a linguagem ACE. Em (Jureta et al. 2009), os autores afirmam ser possível validar relativamente qualquer artefato da Engenharia de Requisitos, desde que o mesmo possa ser expresso como proposições. Com as regras de tradução oferecidas nesta tese, torna-se possível validar relativamente qualquer SIG do NFR Framework ou modelo do *framework* i*.

Métodos convencionais de inspeção, como os comparados em (Porter et al. 1995), geralmente envolvem a análise independente de vários revisores. A argumentação entre várias partes força os interessados a justificarem seus argumentos a favor ou contra um desenho e a submeterem seus pontos de vista a um processo de revisão por pares. O grafo de argumentos pode ser formalmente avaliado para checar se existe um consenso.

Outra técnica de validação é a prototipação (Carter 2001). Esse tipo de técnica difere da nossa proposta no tempo da validação. A prototipação apóia a validação dos requisitos após a implementação do protótipo, enquanto a nossa proposta valida relativamente os requisitos em tempo de desenho. Embora a validação em tempo de desenho ocorra antes de qualquer esforço ser expedido na

PUC-Rio - Certificação Digital Nº 0711310/CA

implementação, não existem estudos quantitativos que comparem a eficiência da argumentação e da prototipação na Engenharia de Software.

O processo de argumentação foi aplicado em dois estudos de caso. Os resultados obtidos permitiram-nos concluir que é possível usar um mecanismo formal para rastrear opiniões distintas em uma discussão com a ajuda de uma estrutura formal. Além disso, é possível validar os artefatos de transparência, de acordo com os interessados que participam da discussão, usando o *framework* ACE para manter os rastros entre as proposições e os interessados que as propuseram.

Nosso trabalho também mostrou a aplicabilidade de trabalhos anteriores recentes, como o *framework* ACE e o Catálogo de Transparência de Software.

9.3.4. Pré-Rastreabilidade

A principal contribuição científica da nossa proposta relacionada à pré-rastreabilidade foi anexar o processo de Engenharia de Requisitos ao artefato. Estruturas de Contribuição (Gotel e Finkelstein 1995) permitem elos diretos entre os artefatos e os indivíduos. Entretanto, elas perdem, por exemplo, as dimensões de rastro: *Por que?*, *Quando?*, *Onde?*, *Como?* e *Quanto?*.

O Google (Brin e Page 1998) é baseado em uma representação da Internet – seus *sites*, recursos e elos – como um grafo. O ITrace tenta aplicar a mesma ideia no escopo da Engenharia de Requisitos. O ITrace rastreia a evolução dos artefatos ER de volta às atividades que foram realizadas nas interações sociais. O modelo oferece um ponto de vista inovador no que tange a rastreabilidade de artefatos da ER e, consequentemente, em como o software evolui.

O ITrace é um modelo prático e flexível. Ele pode ser facilmente replicado e usado por qualquer pessoa sem treinamento extensivo. Formalizamos o mínimo necessário para definir uma base comum para todos os usuários do ITrace. Entretanto, os três grafos entrelaçados podem ser estendidos como todo modelo baseado na representação gráfica RichPicture. Não existe uma forma única ou mesmo correta de se construir um modelo ITrace: o projetista pode modelar da forma como desejar, levando em consideração suas necessidades e habilidades pessoais (e.g. criatividade, conhecimento e motivação). O que realmente importa é não perder os rastros entre os artefatos ER, as interações sociais, as metas das

PUC-Rio - Certificação Digital Nº 0711310/CA

interações sociais, as atividades, as técnicas, as fontes de informações e as redes sociais que produzem/evoluem esses artefatos.

Quando apresentada a ideia do ITrace, o Professor Berry[2] recordou sobre o "POTLO BOAD TIP" - ***P**roblem **O**f **T**he **L**ack **O**f **B**enefit **O**f **A** **D**ocument **T**o **I**ts **P**roduces*, um problema que pode inibir os esforços de sistematização de requisitos (Berry et al. 2010). A pré-rastreabilidade é frequentemente não considerada pelos engenheiros de requisitos, pois a mesma não beneficia a modelagem de requisitos em si. Procuramos lidar com esse problema provendo pré-rastreabilidade com esforço mínimo. Como nossos modelos são facilmente desenhados à mão durante as interações sociais, nenhum esforço adicional é necessário depois dos encontros.

Nossa proposta vai de encontro às soluções habituais para a pré-rastreabilidade, que se baseiam em meta-modelos rígidos que delineiam os formulários a serem preenchidos em ferramentas de gerência. A implementação da nossa proposta através de uma plataforma *Web* colaborativa também não é usual.

A transformação desses modelos em hipermídia, feita de forma colaborativa, também ajuda a distribuir os esforços, reduzindo o esforço individual. É claro que a organização precisa ter maturidade para entender que vale a pena investir em práticas de Engenharia de Requisitos visando um retorno futuro.

Mai um detalhe, a representação gráfica do ITrace pode tornar-se volumosa uma vez que a evolução do modelo é um grafo que está sempre crescendo. Visando contornar essa limitação, representações gráficas poderiam ser usadas somente para mostrar algumas poucas interações sociais, produzindo um *snapshot* do artefato no dado momento de tempo.

Finalmente, o ITrace pode ainda rastrear outros artefatos da Engenharia de Software estendendo o terceiro grafo *A* (vide Capítulo 6 – Seções 6.1 e 6.2) para considerar e detalhar um artefato específico.

[2] Um encontro com o Professor Daniel M. Berry durante a visita dele ao Departamento de Informática na PUC-Rio (9/6/20011).

PUC-Rio - Certificação Digital Nº 0711310/CA

9.4. Estudos de Caso para os Trabalhos Pontuais

Utilizamos uma série de estudos de caso para os trabalhos pontuais, ou seja, focados em um dos desafios para a transparência de software. O desenvolvimento desses estudos de caso foi fundamental para que pudéssemos observar, na prática, os problemas a serem superados ou contornados.

O primeiro estudo de caso desenvolvido foi uma aplicação ubíqua no domínio de comércio eletrônico. Nesse estudo de caso, desenvolvemos as primeiras heurísticas para anexar os requisitos ao código. Uma primeira representação de metas flexíveis como variáveis nebulosas foi utilizada em uma máquina de raciocínio qualitativa. Porém, as regras nebulosas eram criadas manualmente, uma a uma. Além disso, nem as metas flexíveis, nem as contribuições eram representadas como crenças dos agentes.

Aprimoramos nossas heurísticas em um segundo estudo de caso, um *framework* para Adaptação de Conteúdo no domínio da computação ubíqua. A maioria das abstrações do modelo i* foram cobertas por heurísticas transformacionais, bem como implementou-se as abstrações de papéis e posições como capacidades dos agentes. As dependências estratégicas ainda não possuíam heurísticas relacionadas. As metas flexíveis e as contribuições foram representadas como crenças dos agentes e acessadas pela máquina de raciocínio.

O terceiro estudo de caso diferenciou-se dos dois primeiros, uma vez que tratou-se de um experimento sobre delegação de tarefas entre atores. Um ator podia realizar uma tarefa ou delegá-la a outro ator. Nesse experimento, um agente, representando um ator, escolhia entre executar um plano obtido de uma base de planos ou delegá-lo a outro agente. Dessa forma, um agente inicial, representando um sistema de agendamento de reuniões, delegava tarefas, formando uma comunidade de agentes – um sistema multi-agentes. Nesse estudo de caso, foram definidas as heurísticas transformacionais para as abstrações restantes do *framework* i*. As dependências estratégicas entre atores foram finalmente transformadas em protocolos de interação entre agentes.

A dificuldade em se criar regras nebulosas manualmente, uma a uma, para cada situação de escolha de planos, motivou o quarto estudo de caso, um simulador de propagações. O objetivo desse estudo de caso foi a geração

PUC-Rio - Certificação Digital Nº 0711310/CA

automática das regras nebulosas, de forma padronizada, de acordo com as contribuições dos planos e as metas flexíveis impactadas. Quando um plano era selecionado, o simulador aplicava os impactos gerados pelas regras nebulosas nas metas flexíveis.

O simulador de propagações foi reescrito como uma máquina de raciocínio implementada em uma capacidade JADEX, podendo assim ser reutilizada pelos agentes. Nesse quinto estudo de caso, em um sistema de iluminação inteligente, aplicou-se a máquina de raciocínio qualitativa para metas flexíveis no intuito de selecionar os planos adequados sempre que uma dada meta era ativada. Os impactos nas metas flexíveis passaram a ser comunicados a todos os agentes por *broadcasting*.

O sexto estudo de caso concentrou-se na argumentação e no uso do *framework* ACE. Nesse estudo de caso, discutiu-se a clareza das abstrações do modelo i*, mais especificamente o impasse sobre a semântica dos elos meios-fim. Esse impasse é bem conhecido na comunidade de Engenharia de Requisitos e resume-se a uma falta de consenso sobre a semântica da seguinte situação: quando uma meta pode ser atingida por várias tarefas representadas no modelo i*, deve-se escolher somente uma (semântica OU exclusivo - XOR) ou pode-se selecionar várias ao mesmo tempo? Utilizou-se de grafos de argumentos e dos algoritmos do *framework* ACE para identificar impasses e consensos.

O último estudo de caso para os trabalhos pontuais foi modelar as reuniões do Grupo de Transparência de Software através de modelos de pré-rastreabilidade ITrace. Nessas reuniões, padrões de requisitos foram criados ou evoluídos, visando à construção de um Catálogo de Transparência de Software. Os modelos ITrace ofereciam a pré-rastreabilidade aos artefatos.

9.5. Contribuições Relacionadas à Abordagem Integrada

A abordagem integrada proposta é um dos primeiros estudos sobre como desenvolver software transparente. A abordagem apresenta atividades em diferentes níveis de abstração, guiando o trabalho dos engenheiros de requisitos e de software ao longo do desenvolvimento.

PUC-Rio - Certificação Digital Nº 0711310/CA

As atividades propostas aplicam os trabalhos desenvolvidos, apresentados nos Capítulos 3 a 6. Esses trabalhos produzem artefatos que ajudam os engenheiros a enfrentar os desafios relacionados à transparência de software. Alguns dos artefatos produzidos são grafos de argumentos, requisitos de transparência, elos de rastreabilidade, modelos de pré-rastreabilidade e *templates* de código, ou seja, a arquitetura e a "casca" dos agentes intencionais.

O software produzido pela abordagem integrada possui, explicitamente, os requisitos e os requisitos de transparência discutidos pelos interessados. Mesmo os requisitos não-funcionais, ou critérios de qualidade, estão presentes explicitamente no código como metas flexíveis que auxiliam a escolha de planos pelos agentes. Para isso, são utilizados elos de rastreabilidade na forma de pares origem-destino. Esses pares são obtidos a partir das condições e das regras de aplicação das heurísticas transformacionais. Diferentemente de abordagens que geram rastros a partir de sistemas legados ou artefatos já produzidos, nossa abordagem baseia-se em rastreabilidade por construção, ou seja, os artefatos são criados juntamente com seus elos de rastreabilidade. Entretanto, uma padronização de nomes é necessária, o que exige disciplina dos engenheiros de software.

A abordagem oferece uma forma organizada e integrada de se aplicar os trabalhos pontuais. Além disso, a abordagem especifica um processo iterativo, onde os artefatos são evoluídos após discussões com os interessados. As atividades Armazenar e Consultar formam esse ciclo de retroalimentação.

Outra contribuição da abordagem integrada é o uso de um *Wiki* para armazenar os artefatos e disponibilizá-los a todos os participantes.

9.6. Estudo de Caso para a Abordagem Integrada

A construção do Lattes-Scholar como um software transparente foi utilizada como estudo de caso para a abordagem integrada. O Lattes-Scholar é uma aplicação de interesse da comunidade de pesquisadores brasileiros disponível na Internet e também pode ser utilizado em bancas de concursos públicos e em processos de seleção em empresas. O Lattes-Scholar une processos de busca no

PUC-Rio - Certificação Digital Nº 0711310/CA

Lattes e no Google Scholar, manipulando informações oriundas de fontes externas.

No decorrer da prototipação do experimento, isto é, nas suas primeiras aplicações para solidificar o processo, notamos que o uso da argumentação na elicitação de requisitos de transparência possibilita aos interessados não só expressarem seus pontos de vista como também solidificá-los. Por exemplo, em uma reunião que discutiu a Disponibilidade, a Portabilidade e a Publicidade do Lattes-Scholar (critérios de qualidade relacionados ao software ser acessível), existia praticamente um consenso inicial entre os interessados de que a simples natureza do Lattes-Scholar como um serviço *Web* seria suficiente, uma vez que: (i) em termos de Disponibilidade, o software estaria disponível para todos os interessados via *Web*; (ii) em termos de Portabilidade, o software seria compatível com vários dispositivos e *browsers*, e (iii) em termos de Publicidade, o software já teria sua divulgação garantida apenas por estar na *Web*.

Ao longo da reunião, vários argumentos foram desconstruindo essa visão consensual inicial, uma vez que (i) em termos de Disponibilidade, o servidor onde o software estava hospedado estaria sujeito a falhas, à manutenção e a problemas em outros servidores, como *proxys* e/ou *firewall*; (ii) em termos de Portabilidade, a linguagem de programação utilizada na implementação do Lattes-Scholar poderia torná-lo incompatível com alguns *browsers*, e (iii) em termos de Publicidade, era imprescindível que o Lattes-Scholar fosse divulgado para a comunidade acadêmica, além de constar nos resultados das buscas efetuadas no Google Scholar. Um novo entendimento foi surgindo entre os interessados de que algumas restrições seriam necessárias para a transparência do software, tais como: (i) a escolha de um servidor *web* confiável, com um período mínimo de *downtime*, e (ii) o uso de uma linguagem de programação compatível com os *browsers* mais difundidos.

Dada a natureza aberta da discussão, os interessados também puderam chegar a um consenso sobre a possível criação de um padrão-problema (Serrano e Leite 2011a; Supakkul et al. 2010) para capturar problemas relacionados à disponibilidade de um software na *Web* e outro padrão-problema para capturar problemas de portabilidade devido à incompatibilidade entre algumas linguagens de programação e alguns *browsers*. Esse fato sugere que foi positivo manter as discussões como discussões abertas ao longo do estudo de caso.

PUC-Rio - Certificação Digital Nº 0711310/CA

Nessas primeiras aplicações do experimento, a própria discussão sobre Disponibilidade permitiu que fosse percebida uma discrepância entre o entendimento dos interessados sobre Operabilidade, um critério de qualidade também relacionado à transparência. Alguns interessados defendiam que sob o ponto de vista de um cidadão, um software não disponível implica em um software não operacional. Usar a argumentação permitiu que um consenso fosse obtido, entre os interessados, de que a definição de Operabilidade no CTS não estava clara. O uso da argumentação forneceu, assim, um rastro para a evolução do CTS.

Em relação ao uso do quadro branco pelo mediador para anotar os principais argumentos levantados na discussão, percebemos que em uma dada reunião aproximadamente 20 argumentos foram anotados. Entretanto, analisando posteriormente o vídeo dessa reunião, deduzimos que ao menos cem argumentos foram colocados para sustentar esses 20 argumentos principais. Essa observação permitiu-nos concluir que uma gravação em áudio ou em vídeo é imprescindível para que argumentos não sejam esquecidos e que não se perca a base, ou o rastro, para os argumentos principais.

Observamos também que em situações de impasse entre os interessados, o mediador pode – e deve – desempenhar um papel fundamental para dar prosseguimento à discussão. Nessas primeiras reuniões, utilizamos com sucesso uma estratégia de sugestão de possíveis cenários "e se..." (*what if*), apresentando situações hipotéticas sobre as funcionalidades do Lattes-Scholar para instigar novos argumentos dos interessados.

Assistindo a filmagem da primeira aplicação do experimento em uma reunião, percebemos algumas distorções do que se entende como um consenso entre as partes. Essas distorções são intrínsecas da argumentação em uma discussão livre e originaram-se, principalmente, da subordinação ou intimidação dos interessados por terceiros. Argumentos levantados por doutores usualmente não eram contestados por doutorandos ou mestrandos. Além disso, alunos que fazem parte do grupo há menos tempo só expunham seus argumentos e pontos de vista quando diretamente indagados pelo mediador. Em alguns momentos do vídeo, a discussão tomava a forma de um debate, onde um interessado tentava "vencer" a discussão, "derrotando" os argumentos do "adversário". Essa atitude por parte de alguns interessados pode mostrar-se contra-produtiva, uma vez que a

PUC-Rio - Certificação Digital Nº 0711310/CA

palavra final é, normalmente, do interessado que defende seu ponto de vista com mais vigor. Sugerimos, dessa forma, que as reuniões sejam complementadas com entrevistas individuais com os interessados menos participativos.

Uma última observação sugere que haja um limite de tempo na discussão, fato notório em reuniões de discussão. Portanto, na condução da discussão é preciso estar bem claro o papel do mediador para que se evitem tais situações. O fato observado foi basicamente um impasse entre interessados com pontos de vista divergentes, o que levou à distração dos outros interessados. Isso conduziu-nos a usar a heurística de que o mediador deve ter atenção ao nível de interesse de todos os presentes, sobre o risco de que o experimento deteriore-se em função do desinteresse dos participantes diante do andamento da discussão.

9.7. Avaliação da Abordagem Integrada

O Capítulo 8 apresenta a avaliação da abordagem integrada. Avaliamos a aplicação da abordagem de forma quantitativa e qualitativa. A avaliação quantitativa baseou-se em uma escala de qualidade para as discussões, graduada de Péssimo a Excelente. Os participantes, em cada reunião, assinalaram em um questionário suas impressões sobre a qualidade da discussão. Percebemos, analisando os questionários e os vídeos das reuniões, que os interessados menos participativos tendem a atribuir uma qualidade menor à discussão. Usualmente, essa menor participação devia-se ao problema da subordinação/intimidação ou à falta de vontade de se indispor com um colega.

A avaliação qualitativa da aplicação da abordagem baseou-se em uma pergunta no questionário em relação à impressão do interessado sobre a discussão. Comparando a resposta discursiva do interessado com a categoria de qualidade assinalada, notaram-se algumas diferenças na expectativa dos interessados em relação à discussão. Alguns interessados tenderam a assinalar graus de qualidade mais altos quando vários pontos foram discutidos; outros assinalaram graus de qualidade mais altos quando alguns pontos foram discutidos mais profundamente. De um modo geral, pode-se observar nos vídeos das reuniões que esses dois interesses, discussão mais ampla e discussão mais profunda, tendem a se equilibrar em discussões abertas.

PUC-Rio - Certificação Digital Nº 0711310/CA

Avaliamos a abordagem através da demonstração da presença explícita dos requisitos de transparência no código do SMA, ou seja, nos ADFs e nos planos. Os requisitos de transparência, modelados intencionalmente através do *framework* i*, são compostos por atores, metas, metas flexíveis, tarefas, recursos e dependências estratégicas. Mostramos que essas abstrações estão presentes de forma explícita no código da aplicação.

Analisamos a transparência de duas versões anteriores do Lattes-Scholar, mais especificamente a versão na linguagem Lua e a versão SMA comportamental na plataforma JADE. Exemplificamos nossa análise com trechos de código dessas duas versões bem como discutimos os ganhos em se aplicar a nossa proposta de desenvolvimento em detrimento das propostas aplicadas nas versões anteriores.

9.8. Esclarecimento das Questões Levantadas na Introdução

Ao longo da Introdução dessa tese, Capítulo 1, algumas questões foram levantadas. Essas questões estavam relacionadas a dúvidas ainda não esclarecidas sobre como se obter a Transparência de Software ou como a mesma seria afetada pelo Processo de Desenvolvimento de Software (PDS). Essa seção visa esclarecer as questões com base nos resultados e na experiência adquiridos ao longo do desenvolimento dessa tese.

Na Seção 1.1 - Motivação, cinco questões foram levantadas. As duas primeiras, pertinentes ao Catálogo de Transparência de Software, eram: “Como aplicar esse catálogo?” e “Como utilizar esse conhecimento na produção de um software que possa ser considerado transparente?”. Essas duas questões foram respondidas implicitamente no Capítulo 5 – Argumentação sobre Requisitos de Transparência. O catálogo é aplicado como material de apoio para a argumentação dos interessados, como um guia, ou checklist, sobre assuntos que devem ser discutidos. Esse conhecimento (como os padrões questão e alternativa), portanto, acaba por embasar os requisitos de transparência do software em desenvolvimento, definidos pelos próprios interessados.

A terceira questão, “Como mostrar que a transparência de software está presente nos requisitos, conforme desejado pelos interessados?”, também foi respondida no Capítulo 5. A transparência de software está presente nos requisitos

PUC-Rio - Certificação Digital Nº 0711310/CA

na forma de requisitos de transparência. A condição de aceitabilidade dos requisitos de transparência mostra que a transparência do software, como entendida pelos interessados, foi relativamente validada, isto é, houve um acordo entre as partes interessadas (*multi-party agreement*) de que aqueles requisitos estão corretos.

As duas outras questões levantadas na Seção 1.1, "Como mostrar que, mesmo após todo o processo de desenvolvimento, a transparência de software continua presente no software?" e "Como estabelecer que o software é suficientemente transparente?", foram respondidas implicitamente nos Capítulos 3 e 7 – Anexação de Requisitos ao Código e Abordagem Integrada para o Desenvolvimento de Software Transparente, respectivamente. A transparência de software continua presente no software, pois o processo de desenvolvimento é dirigido por modelos e orientado à meta. Assim, as heurísticas transformacionais de desenho e de implementação garantem que os requisitos de transparência elicitados nas discussões entre os interessados não se perderam ao longo do PDS e estão presentes no código do software. O PDS dirigido por modelos também garante que as mesmas metas estão presentes desde os mais altos níveis de abstração até os níveis mais baixos. O software pode ser dito suficientemente transparente, uma vez que se garante que os requisitos de transparência relativamente validados estão presentes.

A Seção 1.2 – Caracterização do Problema levanta quatro questões. A primeira, "É possível aplicar o estado da arte da engenharia de requisitos, a engenharia de requisitos orientada a metas (Goal-Oriented Requirements Engineering - GORE)?", foi respondida claramente nos Capítulos 3 e 5, Anexação de Requisitos ao Código e Argumentação sobre Requisitos de Transparência, respectivamente. Não apenas foi possível aplicar GORE, como pretendido; a orientação à meta apoiou a Transparência de Software, uma vez que trouxe a dimensão "porquê" da informação (o Why do 5W2H).

A segunda questão levantada na Seção 1.2, "Como anexar requisitos orientados à meta, ou intencionais, ao código?" foi explicitamente respondida pelo Capítulo 3 – Anexação de Requisitos ao Código. O objetivo desse capítulo é descrever como se faz essa anexação dos requisitos orientados à meta ao código.

A terceira questão levantada na Seção 1.2, "Um processo de desenvolvimento dirigido por modelos é suficiente para a produção de um

PUC-Rio - Certificação Digital Nº 0711310/CA

software transparente?", não foi claramente respondida ao longo dessa tese. A resposta a essa questão é sutil, e pode ser percebida pela existência dos Capítulos 4, 5, 6 e 7. A resposta, de forma clara, é "não, um processo dirigido por modelos não é suficiente". A transparência de software exigiu outras soluções para desafios alheios à anexação de requisitos através de um processo dirigido por modelos como, por exemplo, a validação relativa de requisitos de transparência através de argumentação.

A última questão levantada pela Seção 1.2, "Qual tecnologia/linguagem utilizar para implementar o software, de modo a evitar a quebra do paradigma intencional durante o desenvolvimento?", admite diversas respostas. De forma simplista, qualquer tecnologia/linguagem que apóie a construção de software orientado à metas evitaria a quebra do paradigma intencional. Porém, observando vários critérios como a orientação a atores, a portabilidade, a atividade da comunidade e a adoção de políticas de software livre e código aberto, entre outros, nos levaram a optar por utilizar o sistema multi-agentes JADE e seu add-on para suporte à intecionalidade JADEX, ambos construídos na linguagem orientada a objetos JAVA.

9.9. Limitações

Nossa abordagem foi aplicada em um grupo de interessados que já possuíam certo conhecimento sobre transparência de software, pois participaram da construção do Catálogo de Transparência de Software. Não foi possível determinar o quanto o catálogo contribuiria para a discussão caso o conjunto de interessados fosse composto apenas por leigos em transparência de software.

As reuniões sobre a transparência do Lattes-Scholar junto ao Grupo de Transparência de Software ao longo de dois meses conseguiram discutir sete metas flexíveis relacionadas à transparência de software, de um total de vinte e oito. Embora as metas flexíveis tenham sido selecionadas de modo que pelo menos uma meta flexível de cada grupo (Acessibilidade, Usabilidade, Informatividade, Entendimento e Auditabilidade) fosse discutida, não pudemos observar uma aplicação completa da abordagem. Não houve tempo hábil para, por

PUC-Rio - Certificação Digital Nº 0711310/CA

exemplo, conduzir um experimento que envolvesse discussões de metas flexíveis antagônicas, como Simplicidade e Detalhamento ou Concisão e Completeza.

Modelamos os requisitos de transparência através de modelos intencionais, o estado da arte da Engenharia de Requisitos. Entretanto, para manter o mesmo paradigma ao longo do desenvolvimento, bem como a distribuição das metas entre atores, foi necessário limitar a tecnologia utilizada para implementar o software a SMAs intencionais. Para a implementação do estudo de caso, o Lattes-Scholar, foi necessária, por exemplo, uma arquitetura híbrida páginas *web*/SMA.

Outra possível limitação da nossa abordagem é o uso de argumentação para elicitação e validação relativa de requisitos de transparência. Embora a argumentação tenha sido uma peça chave para permitir a participação dos interessados nos processos de elicitação e validação dos requisitos de transparência, o uso de argumentação na Engenharia de Requisitos (Jureta et al., 2009) é uma proposta recente. Ainda não existem estudos sobre o custo de se aplicar argumentação e de se analisar grafos de argumentos. Não existem também comparações com outras técnicas de elicitação ou de validação.

9.10. Trabalhos Futuros

Nessa seção apresentamos algumas sugestões de temas para trabalhos futuros para a continuidade dos estudos.

A primeira possibilidade, e a com aplicação mais direta, seria o desenvolvimento de uma ferramenta CASE semi-automatizada que implementasse as heurísticas transformacionais e utilizasse os elos de rastreabilidade origem-destino.

Outra possibilidade que envolve ferramentas CASE seria a inclusão do Simulador de Propagações em uma ferramenta de modelagem de requisitos intencionais para a análise dos modelos.

Como foi exposto na seção 9.11, um possível trabalho futuro seria conduzir um experimento que envolvesse discussões de metas flexíveis antagônicas, como Simplicidade e Detalhamento ou Concisão e Completeza. A comparação da aplicação da abordagem com estudos de caso que envolvessem outros domínios seria outra possibilidade. Também seria possível repetir o

PUC-Rio - Certificação Digital Nº 0711310/CA

processo de argumentação com operacionalizações já inseridas no catálogo e avaliar o que isso modificaria no processo de argumentação.

Na área de Gerência de Conhecimento, seria possível utilizar ontologias para descrever os modelos intencionais e as dependências estratégicas entre os atores, de forma que esse conhecimento pudesse ser repassado a agentes que quisessem interagir com o sistema. As ontologias também ajudariam a definir exatamente o que está sendo enviado nas mensagens e qual sua semântica.

Na área da Inteligência Artificial, o trabalho futuro, aparentemente mais promissor, seria estender a máquina de raciocínio qualitativa para uma máquina de *planning*. O agente não selecionaria o plano mais adequado naquele instante; selecionaria o plano que, em conjunto com planos selecionados em outros momentos, resultasse em uma maior satisfação das metas flexíveis de mais alto nível.

Em relação ao modelo de pré-rastreabilidade ITrace, um possível trabalho futuro seria estendê-lo com a representação de modelos de interação na camada intermediária. Outra possibilidade seria investigar como o uso de *crowd tagging* de modelos ITrace na plataforma colaborativa contribui para o refino de buscas em um conjunto de modelos ITrace.

Finalmente, um outro trabalho futuro possível é continuar com a evolução das interfaces gráficas do Lattes-Scholar (Apêndice A), através da discussão de metas flexíveis relacionadas à Usabilidade, à Informatividade e ao Entendimento.

PUC-Rio - Certificação Digital Nº 0711310/CA