1.
Introdução

Este trabalho apresenta uma abordagem para o desenvolvimento intencional de software transparente baseado em argumentação. Descrevemos nossos trabalhos relacionados aos quatro desafios para a operacionalização da transparência de software e a abordagem que integra nossos esforços. Neste capítulo, expõe-se a motivação deste trabalho, caracteriza-se o problema da operacionalização do conceito de transparência de software, descreve-se o enfoque da solução proposta e define-se a organização da tese.

1.1.
Motivação

Transparência é uma preocupação crítica para as sociedades democráticas modernas. Podemos entender que a transparência é um princípio chave para se estabelecer uma relação de confiança entre os cidadãos e os governos que os representam. De acordo com Holzner&Holzner (Holzner and Holzner 2006), a transparência consiste no "valor social do acesso aberto, público e/ou individual à informação mantida e disponibilizada por centros de autoridade." Percebe-se que os autores enfatizam que o acesso às informações que antes eram consideradas privilegiadas possui um valor social.

Outro autor (Lord 2006), diz: "A transparência é uma condição na qual as informações relacionadas às prioridades, capacidades e comportamentos de poderosas organizações estão amplamente disponíveis para o público mundial". Lord deixa claro o novo poder da sociedade em auditar organizações poderosas, re-equilibrando a distribuição dos poderes em uma sociedade democrática.

Esse interesse da sociedade em relação à transparência pode ser observado pela criação de várias instituições e organizações como a Transparência Internacional (TI 2010) e Transparência Brasil (TB 2010), dos portais de transparência dos governos federal (PTRANSP 2011), estaduais e municipais e de iniciativas como o Freedom of Information Act (FIA 2010) e a lei Sorbanes-Oxley (SOX 2010), entre outros. Segundo Cappelli (Aló 2009), existe uma forte expectativa da sociedade em relação à transparência.

O interesse da sociedade na Transparência de Software já começa a ser percebido, ainda que de forma implícita, pois o software permeia nossas vidas sociais. Softwares que operam em telefones celulares, por exemplo, precisam informar ao cidadão[1] o que estão fazendo, explicar por que é necessário acessar a rede de dados e até mesmo solicitar a autorização do cidadão para acessar determinados recursos, como o GPS. No domínio web, os cidadãos exigem cada vez mais que as políticas de privacidade sejam claras, explicitando exatamente como o site manipulará as informações do cidadão. Nota-se um interesse crescente dos cidadãos em conhecer e até mesmo controlar o que o software faz. Assim, a Transparência de Software está se tornando um critério de qualidade que exige mais atenção dos desenvolvedores de software.

Alguns trabalhos (Meunier 2008) (Mercuri 2005) podem ser encontrados na literatura em relação à transparência de software, de forma explícita. Meunier, por exemplo, relaciona transparência de software à qualidade do software de conter exatamente, e apenas, as funcionalidades descritas pelos requisitos. Assim, seria possível evitar que um software realizasse uma coleta de dados sem autorização do cidadão. Também seriam prevenidos ataques do tipo *backdoor*, que consistem em uma falha de segurança proposital criada por um dos desenvolvedores para que seja explorada posteriormente.

Entretanto, nenhum desses trabalhos se preocupa em definir o que seria a transparência de software. Os autores seguem uma visão pessoal e implícita de transparência. Meunier, por exemplo, relaciona a transparência à lista de funcionalidades do software, mas não discute questões relacionadas à acessibilidade.

Leite e Cappelli (Leite e Cappelli 2010) tentam entender e definir a transparência de software como um requisito não-funcional (RNF) identificando sua relação com outros RNFs bem conhecidos.

De acordo com os autores, a transparência de software é auxiliada pelos RNFs Acessibilidade, Usabilidade, Informatividade, Entendimento e Auditabilidade. Eles propõem um grafo de interdependência de metas flexíveis

[1] Evitamos o uso do termo "usuário", uma vez que o cidadão não apenas usa o software, mas também quer ser informado sobre seu funcionamento e questiona suas ações.

(*Softgoals Interdependency Graph* - SIG) para a Transparência de Software, utilizando o NFR Framework (Chung et al. 2000).

A abordagem inicial do Grupo de Transparência de Software (GTS 2011a; GTS 2011b) foi estender o SIG de Transparência com várias operacionalizações para cada meta flexível. Essa abordagem mostrou-se contra-produtiva, pois somente ofereceu um conjunto fixo de operacionalizações. Optamos, posteriormente, por estender o SIG de Transparência com conjuntos de questões para cada meta flexível relacionada à transparência. Essas questões identificam boas práticas para operacionalizar as metas flexíveis e foram obtidas como resultado de uma adaptação do método *Goal-Question-Metric* (GQM) (Basili 1992) para um método *Goal-Question-Operationalization* (GQO).

O SIG de Transparência e os conjuntos de questões para cada meta flexível foram, recentemente, expressos como padrões de requisitos (Serrano e Leite 2011a) e inseridos em um Catálogo de Transparência de Software (CTS 2011). Esse catálogo facilita a reutilização e a extensão desse conhecimento com novos conjuntos de questões ou de operacionalizações.

A existência desse catálogo expõe algumas questões: Como aplicar esse catálogo? e Como utilizar esse conhecimento na produção de um software que possa ser considerado transparente?

Tradicionalmente, catálogos de requisitos não-funcionais (Chung et al. 2000; Cysneiros et al. 2001) são utilizados por engenheiros de requisitos para introduzir requisitos não-funcionais nos requisitos de um software. Entretanto, a transparência de software é uma meta flexível complexa, relacionada a dezenas de outras metas flexíveis. Diferentemente de metas flexíveis como a Performance, os interessados no software não possuem um entendimento claro do que é a transparência de software, uma vez que a transparência é um conceito abstrato e recente. Além disso, cada interessado possui seu ponto de vista em relação ao que esperar do software em termos de transparência. Os pontos de vista dos interessados podem, inclusive, serem divergentes.

Nesse contexto, surgem ainda outras questões: Como mostrar que a transparência de software está presente nos requisitos, conforme desejado pelos interessados? Como mostrar que, mesmo após todo o processo de desenvolvimento, a transparência de software continua presente no software? Como estabelecer que o software é suficientemente transparente?

1.2. Caracterização do Problema

A transparência de software é uma meta flexível pouco compreendida e subjetiva, dependente do ponto de vista e das expectativas dos interessados. Elicitar e validar a transparência de software não são tarefas simples. Cabe ao cidadão, como um interessado no software, afirmar como deve ser o software transparente. Em última instância, cabe ao cidadão, e não ao engenheiro de software, afirmar se o software é suficientemente transparente.

O próprio processo de desenvolvimento de software deve permitir que seja possível demonstrar que o software produzido segue o que foi definido pelos interessados em termos de transparência. É necessário que se possa anexar os requisitos ao software, ou seja, mostrar aos interessados que um determinado requisito funcional ou não-funcional está presente no código ou que um trecho de código contribui para um dos requisitos. Mesmo as metas flexíveis (ou critérios de qualidade) solicitadas pelos interessados devem estar presentes no software de forma explícita.

Surgem, assim, pelo menos quatro desafios para a operacionalização da transparência de software:

1. Como anexar os requisitos ao software?
2. Como tornar explícitas, no código, as metas flexíveis?
3. Como elicitar o conceito abstrato e subjetivo da transparência de software e como permitir a participação dos interessados no processo de validação dos requisitos de transparência para um determinado software?
4. Como anexar aos artefatos as informações sobre como os próprios artefatos foram produzidos, ou seja, como fazer a pré-rastreabilidade dos artefatos?

Outras perguntas que também precisam ser respondidas:

- É possível aplicar o estado da arte da engenharia de requisitos, a engenharia de requisitos orientada a metas (*Goal-Oriented Requirements Engineering* - GORE)?
- Como anexar requisitos orientados à meta, ou intencionais, ao código?

- Um processo de desenvolvimento dirigido por modelos é suficiente para a produção de um software transparente?
- Qual tecnologia/linguagem utilizar para implementar o software, de modo a evitar a quebra do paradigma intencional durante o desenvolvimento?

1.3. Enfoque da Solução

A solução adotada se baseou, em um primeiro momento, em abordar cada um dos quatro desafios para a transparência de software através do desenvolvimento de estudos de caso. A experiência adquirida em cada um deles possibilitou que fossem propostas soluções pontuais, focadas nos problemas encontrados.

Assim, inicialmente foram desenvolvidos trabalhos relacionados a anexar os requisitos intencionais ao software através de heurísticas transformacionais, de forma a se obter um processo dirigido por modelos. O enfoque de todos esses trabalhos foi a rastreabilidade entre os artefatos e entre os elementos dos artefatos, considerando as heurísticas transformacionais como os elos, ou o rastro. Mais detalhes no Capítulo 3, Anexação de Requisitos ao Código.

Em seguida, nossos estudos foram centrados em construir uma máquina de raciocínio qualitativa para capacitar o software a raciocinar em termos de metas flexíveis em tempo de execução. O enfoque desses estudos foi representar formalmente a subjetividade e a incerteza inerentes às metas flexíveis e ao raciocínio qualitativo. Mais detalhes no Capítulo 4, Raciocínio sobre Metas Flexíveis em Tempo de Execução.

Uma vez que os requisitos intencionais e as metas flexíveis estavam presentes no código de forma explícita, concentramos nossos esforços nos requisitos de transparência. Assim, propusemos soluções para a elicitação e validação de requisitos de transparência em um processo que permitiu a participação dos interessados e que se apoiou no Catálogo de Transparência de Software. Mais detalhes no Capítulo 5, Argumentação sobre Requisitos de Transparência.

Encerrando a série de trabalhos desenvolvidos para abordar cada um dos quatro desafios, concentramos-nos na pré-rastreabilidade dos requisitos de transparência e dos outros artefatos produzidos ou evoluídos. O enfoque desse estudo foi anexar aos artefatos os processos da engenharia de software, bem como a rede social investigada, as fontes de informação utilizadas, os recursos necessários, entre outros. Mais detalhes no Capítulo 6, Pré-Rastreabilidade.

Por último, foi proposta uma abordagem que integrou as soluções pontuais. As soluções foram agrupadas e representadas como atividades. Essas atividades utilizam ao menos duas dessas soluções e podem ser incorporadas a processos de desenvolvimento de software já existentes. A abordagem foi aplicada em um estudo de caso, o Lattes-Scholar (Lattes-Scholar 2011a; Lattes-Scholar 2011b). Mais detalhes no Capítulo 7 e no Apêndice A.

A aplicação da abordagem foi avaliada de forma quantitativa e qualitativa, através de questionários preenchidos pelos participantes em reuniões semanais. A abordagem integrada foi avaliada ainda por construção, demonstrando que os requisitos de transparência estão presentes explicitamente no código dos agentes. Finalmente, e visando complementar o processo de avaliação, também foi realizada uma breve análise da transparência de duas versões anteriores do Lattes-Scholar.

1.4. Organização da Tese

Esta tese está organizada em capítulos. O Capítulo 2 introduz alguns conceitos e apoios tecnológicos que serviram de base para a nossa pesquisa. Os Capítulos 3 a 6 apresentam nossos esforços em relação aos quatro desafios para a operacionalização da transparência de software, respectivamente: a rastreabilidade entre os requisitos e o código, uma máquina de raciocínio qualitativa para metas flexíveis, argumentação sobre requisitos de transparência e pré-rastreabilidade. O Capítulo 7 descreve uma abordagem que integra os trabalhos apresentados nos Capítulos 3 a 6. O Capítulo 8 apresenta a avaliação da abordagem proposta. Por último, o Capítulo 9 expõe as conclusões e os trabalhos futuros.