8
Considerações Finais

A atividade de manutenção de sistemas há anos têm recebido atenção de pesquisadores devido aos custos e dificuldades inerentes a tal atividade (Lehman et al. 1997). Númerosas técnicas e processos têm sido propostos com o objetivo de facilitar a manutenção de sistemas de software. Em particular diversos autores têm catalogado exemplos de estruturas de código que representam inadequada modularização do código e que podem contribuir com a deterioração do software em evolução (Fowler et al. 1999, Riel 1996). A refatoração, realizada no contexto da manutenção perfectiva, visa reestruturar o sistema de forma a eliminar essas anomalias de modularidade. A localização dessas anomalias no código, contudo, não é uma atividade simples. Apesar disso, ela é crucial ao processo de refatoração.

Métricas e estratégias de detecção têm sido frequentemente utilizadas para apoiar tal etapa. Entretanto, observa-se que, apesar de muitos problemas de modularidade serem adicionados no decorrer da evolução dos sistemas, a maioria das métricas e estratégias de detecção não são sensíveis ao histórico dos mesmos. Essa limitação pode ser percebida em estratégias de detecção definidas em (Marinescu 2002, Marinescu 2004, Lanza e Marinescu 2006), na definição de diversas métricas para o paradigma orientado a objetos (Chidamber e Kemerer 1994, Henderson-Sellers 1996, Li e Henry 1993, Lorenz e Kidd 1994) e também na maioria das ferramentas que suportam detecção automática de anomalias de código (Together 2009, inCode 2009, inFusion 2009, iPlasma 2009).

Em geral, abordagens de avaliação de código tendem a considerar apenas as características da versão mais recente do código, ou seja, são agnósticas à evolução de propriedades do mesmo. Observamos que, apesar de existirem suspeitas de que informações sobre a evolução do código poderiam contribuir para a localização de potenciais anomalias de modularidade (Ratiu et al. 2004), ainda são poucas as evidências de trabalhos nessa linha de pesquisa. É nesse contexto que estão as contribuições desta dissertação.

Inicialmente, o objetivo deste trabalho era o de apenas propor métricas e estratégias de detecção sensíveis à história que contribuíssem com avaliações

de código. Tínhamos o interesse em investigar se tais recursos poderiam contribuir com a diminuição de falsos positivos e falsos negativos obtidos por estratégias de detecção convencionais. Conforme o andamento de nossa pesquisa, identificamos que o desenvolvimento de uma ferramenta de suporte que possibilitasse à aplicação das estratégias seria imprescindível. Além disso, identificamos que estratégias sensíveis à história talvez também pudessem contribuir com deficiências de outras abordagens de detecção. A abordagem visual utilizando a SourceMiner foi escolhida devido à oportunidade de estudo colaborativo com pesquisadores da UFBA e da UFMG, ambos interessados nessa temática de detecção de anomalias de modularidade a partir de recursos visuais.

A metodologia de trabalho envolveu ampla análise da literatura relacionada a métricas, anomalias de modularidade, estratégias de detecção e ferramentas relacionadas. Além disso, em estágios iniciais ou mais avançados desta pesquisa, tivemos a oportunidade de trocar experiências com pesquisadores de outras instituições da área de manutenção e modularidade de sistemas. São eles os integrantes do estudo exploratório sobre a detecção de anomalias baseada em recursos visuais. Também dispusemos de experiências trocadas com o professor Radu Marinescu, pesquisador da Universidade de Timisoara, na Romênia, e também um dos proponentes das estratégias convencionais de detecção. Necessidades reais de avaliação de código descobertas em projetos envolvendo o LES e o TecGraf também motivaram esta pesquisa.

A conclusão deste trabalho dispôs de evidências iniciais que nos permitem afirmar que estratégias sensíveis à história (1) podem ser um recurso de detecção de elevada precisão e revocação, (2) podem apresentar melhores resultados que os gerados por estratégias de detecção convencionais difundidos pela literatura e, ainda, (3) podem contribuir para minimizar deficiências de abordagens tradicionais de detecção baseadas em recursos visuais do código. Essas afirmativas foram baseadas em estudos realizados ao longo de 16 versões de dois sistemas selecionados para a etapa de avaliação das estratégias. Nossas contribuições são resumidamente enumeradas na seção seguinte. Algumas sugestões de trabalhos que possam dar continuidade a este são listados na Seção 8.2.

8.1 Contribuições

A identificação de mecanismos para avaliação de código e identificação de candidatos à refatoração têm se apresentado uma área emergente de pesquisa em engenharia de software. Neste contexto, este trabalho apresenta as seguintes

contribuições:

- O trabalho apresenta o estado da arte e análise reflexiva de recursos existentes para detecção de anomalias, tais como métricas e estratégias de detecção convencionais. Além disso, para capturar requisitos desejáveis de uma ferramenta de detecção e identificar limitações de ferramentas existentes, apresentamos uma visão geral de ferramentas de suporte ao mecanismo de estratégia de detecção destacando algumas de suas vantagens e desvantagens. Todos esses recursos foram considerados no Capítulo 3 desta dissertação.
- Um conjunto de métricas sensíveis à história é proposto. Essas métricas possibilitam medições do comportamento evolutivo de propriedades do código e tornam possível a composição de estratégias de detecção sensíveis à história. Todas as métricas foram apresentadas através de elementos como definição textual, expressão matemática, exemplos de cálculo e relevância de uso. Essas métricas foram consideradas no Capítulo 4.
- Um conjunto de estratégias de detecção sensíveis à história é apresentado. Essas estratégias podem ser utilizadas para capturar módulos do código com o mesmo comportamento evolutivo de uma dada propriedade, por exemplo, detecções do tipo Coesão Estritamente Decrescente, Acoplamento Estritamente Crescente, Tamanho Constante, dentre outros. Além disso, elas também podem ser utilizadas para identificar anomalias clássicas de código como *God Class* , *Shotgun Surgery* e *Divergente Change* (Riel 1996, Fowler et al. 1999). Essas estratégias foram apresentadas no Capítulo 5.
- Uma ferramenta de medição e avaliação foi desenvolvida. Ela suporta todas às métricas sensíveis à história proposta neste trabalho e a geração de gráficos de evolução das propriedades do código. Além disso, ela possibilita a especificação e aplicação de diferentes estratégias de detecção convencionais e sensíveis à história. Através da utilização de uma linguagem específica de domínio, o usuário pode especificar declarativamente as estratégias de seu interesse. Essa ferramenta foi apresentada no Capítulo 6.
- Estudos empíricos foram realizados para avaliar as possíveis contribuições de estratégias de detecção sensíveis à história. Os resultados de estratégias foram avaliados quanto a precisão e revocação. Tais resultados foram também comparados com os resultados de estratégias convencionais e de uma abordagem de detecção baseada em recursos visuais (Carneiro et al. 2010a). Tais estudos foram apresentados no Capítulo 7.

O estudo apresentado neste trabalho fornece evidências iniciais da real importância de se considerar informações sensíveis à história na detecção de anomalias de modularidade de código. Vale destacar que grande parte das contribuições desta pesquisa também foi reconhecida por membros da comunidade científica de engenharia de software dos quais obtivemos aceitação de 100% dos artigos submetidos (Mara et al. 2010a, Mara et al. 2010b, Carneiro et al. 2010a).

8.2 Trabalhos Futuros

As contribuições apresentadas representam um primeiro esforço sobre a utilização de informações sensíveis à história para a detecção de anomalias de modularidade de código. É possível enumerar alguns trabalhos que poderiam ser realizados para complementar o estudo iniciado:

Avaliações

- Realização de novas avaliações utilizando sistemas com históricos maiores que os do Mobile Media e Health Watcher. O Health Watcher foi o sistema selecionado para a avaliação com o maior número de versões (10 versões). Contudo, esse era um sistema que sofria basicamente alterações estruturais entre as versões. Seria interessante avaliar os resultados das estratégias em sistemas com um grande número de versões e com características evolutivas diferentes das observadas no Health Watcher e Mobile Media. Tal fato possibilitaria análises e comparações sobre os efeitos relacionados às características evolutivas de cada sistema nos resultados das estratégias sensíveis à história.
- Também poderiam ser realizadas avaliações utilizando sistemas com características de evolução semelhantes às do Mobile Media, mas que não tivessem sido implementados como linhas de produto. Como as estratégias SHs apresentaram-se bastante eficazes neste sistema, isso possibilitaria discussões relacionados aos possíveis impactos de termos utilizado uma linha de produto na avaliação. Outras análises e investigações que poderiam ser feitas nesse contexto, por exemplo, seria se as métricas que avaliam linhas de produto deveriam ou não ser diferentes, e assim por diante.

Estratégias de Detecção

- Um possível estudo futuro seria investigar se informações provenientes de ferramentas de gerência de configuração, ou SCM's[1], poderiam contribuir com a detecção desses problemas. Poderia ser feita, por exemplo a mineração de informações como: classes que frequentemente são alteradas juntas ou requisitos que quando alterados repercurtem em impactos em muitas distintas classes, etc. Tal abordagem, sobre a utilização de informações de SCM's, parece interessante e viável, uma vez que análise desses dados já têm sido utilizados em diversos estudos relacionados à evolução de sistemas (Mens e Demeyer 2001). Além disso, já é possível encontrar algumas pesquisas nessa direção, como (Girba et al. 2004).
- Definição e avaliação de estratégias sensíveis à história para outros tipos de anomalias de modularidade apresentados em (Fowler et al. 1999).

Ferramenta Hist-Inspect

- Esta versão da Hist-Inspect não conta com o cálculo de métricas de convencionais, como explicado no Capítulo 6. Uma possibilidade de trabalho seria estender a ferramenta para que ela também efetue os cálculos de métricas convencionais e não apenas os de métricas sensíveis à história.
- Gráficos com outros tipos de informações poderiam ser úteis em avaliações sensíveis à história de código. Exemplos: (1) gráfico do tipo versão vs. número de anomalias; (2) gráfico do tipo versão vs. valor de métrica com cada linha de evolução representando uma entidade. O gráfico (1) pode ajudar a avaliar se a incidência de anomalias em uma entidade cresceu, diminuiu, permaneceu estável e assim por diante. Isso forneceria uma visão panorâmica das anomalias existentes ao longo das versões do sistema ou de um módulo, o que poderia servir como possível indicador de depreciação do código ao longo da evolução do mesmo. Já a relevância do gráfico (2) está relacionada a necessidades de avaliar possíveis discrepâncias no comportamento evolutivo de duas ou mais entidades. Nesse caso, é possível verificar para uma mesma métrica, como se comportam diferentes entidades do sistema. A observação de diferenças de comportamento muito acentuadas entre as entidades talvez poderiam indicar algum tipo de anormalidade vigente.
- Alteração do mecanismo de implementação das métricas sensíveis à história. A especificação das métricas sensíveis à história atualmente

[1]Do inglês, *Software Configuration Management*

é feita usando classes Java. Entretanto, no futuro elas poderiam ser implementadas utilizando uma linguagem Script (como Java Script, Rubby ou Lua) ou uma nova DSL. Tal fato, possibilitaria a criação e obtenção de resultados de novas métricas SHs sem a necessidade de recompilar todo o código, como atualmente é feito na criação de novas estratégias de detecção.

- Refatoração do módulo de detecção. Atualmente, não existe uma interface gráfica para utilização das funcionalidades relacionadas à execução de estratégias de detecção. Esse módulo, por enquanto, é disponibilizado através de uma interface de linha de comando. Ele poderia ser migrado para a plataforma do Eclipse (Eclipse 2010), possibilitando a aplicação de estratégias no mesmo ambiente de programação dos desenvolvedores. Tal fato facilita a associação das entidades detectadas como anômalas e localização de suas respectivas implementações para possíveis refatorações.
- Integração com ferramentas de gerência de configuração. Ao invés das métricas convencionais serem fornecidas através de arquivos armazenados na máquina do usuário, o cálculo dessas métricas convencionais ao longo das versões poderiam estar armazenadas em um repositório. A ferramenta, nesse caso, deveria ser capaz de se comunicar com o repositório para recuperar tais arquivos de métricas coletadas ao longo de cada versão do sistema. Na versão em que a ferramenta já fosse capaz de realizar os cálculos das métricas convencionais, ela poderia acessar o repositório para efetuar o cálculo de métricas convencionais em cada nova versão do sistema armazenada no repositório.