ENGENHARIA REVERSA DE PADRÕES DE INTERAÇÃO

Clara Raquel da Costa e Silva Sacramento

Dissertação realizado sob a orientação da *Prof. Ana Paiva*
na *Faculdade de Engenharia da Universidade do Porto*

1. Contexto

As aplicações Web (*web apps*) estão a ganhar cada vez mais importância. Devido à sua estabilidade, existe uma tendência cada vez maior para mover aplicações para a Web, exemplos sendo as aplicações de correio eletrónico e de *software* de escritório da Google. As *web apps* estão neste momento a realizar funções que dantes só as aplicações de área de trabalho realizavam.

Apesar da pertinência que as *web apps* têm na comunidade, elas ainda sofrem de falta de convenções e padrões, ao contrário de aplicações de área de trabalho e móveis: a mesma tarefa pode ser implementada de maneiras diferentes, que torna a criação de teste automáticos difícil de concretizar e impede a reutilização de testes. Por exemplo, uma falha numa autenticação de sessão (*login*) pode desencadear uma mensagem de erro, mas algumas implementações simplesmente apagam as credenciais inseridas.

Interfaces gráficas de utilizador *GUIs* de todos os tipos estão povoadas de comportamentos recorrentes que variam ligeiramente entre si; por exemplo, pesquisa de conteúdos e autenticação de sessão. Esses comportamentos (padrões) chamam-se padrões de interface de utilizador (*padrões UI*), e são soluções recorrentes para problemas comuns. Devido ao seu uso bastante difundido, estes padrões permitem aos utilizadores um sentido de familiaridade e conforto quando usam *web apps*.

É de notar que, enquanto os *padrões UI* são familiares, a sua implementação pode variar significativamente; no entanto é possível definir estratégias de teste genéricas e re-usáveis (padrões de teste de interface de utilizador - *UITP*) para testar esses padrões. Isto requer um processo de configuração, de modo a adaptar os testes a diferentes aplicações.

Uma grande parte do esforço em testes baseados em modelo é relacionado com a criação do modelo. Além disso, o modelo pode ser uma fonte de erros conceptuais, introduzidos sem saber pelo testador. Esses problemas podem ser reduzidos ao gerar modelos automaticamente através de engenharia reversa.

2. Motivação

Esta dissertação pretende continuar o trabalho realizado na ferramenta PARADIGM-RE, uma componente do projeto PBGT (Teste de GUIs Baseado em Padrões) responsável por extrair parte do modelo da aplicação Web a testar através de engenharia reversa. Isto é feito através do desenvolvimento de uma ferramenta completamente nova para extrair um modelo parcial da aplicação Web.

3. Objetivos

Os objetivos principais para esta dissertação são: melhorar o processo de engenharia reversa e de dedução de *padrões UI* existente, eliminar a necessidade de interação de um utilizador, e automatizar a construção do modelo; ou em outras palavras, explorar uma aplicação Web independentemente / automaticamente, deduzir *padrões UI* das suas páginas, e finalmente produzir um modelo com os *padrões UITP* que definam estratégias de teste para cada um dos *padrões UI* encontrados. Isto é feito para acelerar o processo de construção do modelo, poupar trabalho ao testador, e mitigar o número de erros conceptuais introduzidos no modelo.

4. Descrição do Trabalho

Esta dissertação apresenta uma abordagem de engenharia reversa dinâmica que pretende extrair parte do modelo de uma aplicação Web existente através da identificação de *padrões UI* do utilizador. Esta abordagem explora automaticamente uma aplicação Web, regista informação relacionada com a interação, analisa a informação guardada, transforma-a em símbolos (*tokens*), e deduz os *padrões UI* existentes através de análise sintática. Após o modelo extraído ser complementado com informação adicional e validado, o modelo está pronto a ser usado como entrada para o projeto PBGT para testar a aplicação Web analisada.

4.1. Visão Global do projeto PBGT

Como já foi mencionado, o foco desta dissertação é a componente de engenharia reversa PARADIGM-RE do projeto de investigação PBGT. O objetivo deste projeto é desenvolver uma abordagem de teste de GUIs baseado em modelo na forma de uma ferramenta de suporte a testes, que possa ser usado em contexto industrial.

A ferramenta PBGT tem cinco componentes principais: **PARADIGM**, uma DSL (*Linguagem específica ao Domínio*) para definir modelos de teste a GUIs baseados em UITPs; **PARADIGM-RE**, uma ferramenta de engenharia reversa de *web apps* cujo propósito é extrair *padrões UI* de páginas Web sem acesso ao seu código fonte, e usar essa informação para gerar um modelo de testes escrito em PARADIGM; **PARADIGM-ME**, um

ambiente de modelação e testes, construído para suportar a criação de modelos de teste; **PARADIGM-TG**, uma ferramenta de geração de testes automática que gera casos de teste a partir de modelos de teste definidos em PARADIGM, de acordo com o critério de cobertura especificado pelo testador; e finalmente, uma ferramenta de execução de testes, **PARADIGM-TE**, que executa os casos de teste, analisa a sua cobertura com uma ferramenta de análise de cobertura (**PARADIGM-COV**),e retorna relatórios de execução detalhados.

4.2. Padrões de interface e Padrões de Teste de Interface

Um padrão de teste de interface define uma estratégia de teste para testar um padrão de interface específico, e que é definido formalmente por um conjunto de objetivos de teste (para configuração pelo testador) com a forma $< Goal; V; A; C; P >$. *Goal* é o identificador do teste. *V* é um conjunto de pares [*variável*, *dados de entrada*] que relaciona variáveis envolvidas no teste com os respetivos dados de entrada. *A* é a sequência de ações a tomar durante a execução do teste. *C* é o conjunto de condições a verificar durante a execução de testes (por exemplo, "verifica se se mantém na mesma página"). *P* é uma expressão booleana, que serve como pré-condição e define o conjunto de estados em que é possível executar o teste.

Os *padrões UI* definidos na linguagem PARADIGM são: **Login** (dois campos de entrada de texto (um normal para inserir email ou nome de utilizador, e um cifrado para a *password*) e um botão de submissão, e opcionalmente uma caixa de seleção para "lembrar-se" do utilizador); **Find** (um ou mais campos de entrada de texto, onde o utilizador insere palavras-chave para pesquisar, e um botão de submissão); **Sort** (ordenação de uma lista de dados por um atributo comum (preço, nome, relevância, etc.) de acordo com um critério definido (ascendente / descendente, alfabético, etc.)); **Master Detail** (este padrão está presente quando a seleção de um elemento de um conjunto (chamado *master*)) resulta na filtragem ou modificação de outro conjunto relacionado (chamado *detail*)); **Input** (este padrão é qualquer elemento em que se possa inserir texto); e **Call** (este padrão é qualquer elemento onde um clique desencadeia um procedimento que causa uma mudança de página).

4.3. Ferramenta Anterior

A abordagem desenvolvida nesta dissertação pretende melhorar o processo desenvolvido na ferramenta anterior [1] desenvolvido na ferramenta PARADIGM-RE. Em particular pretende a sua completa automatização. A ferramenta anterior precisava da intervenção do utilizador para interagir com a aplicação Web a ser analisada de modo a guardar os traços de interação e daí prosseguir. A ferramenta extraía informação da interação do utilizador com a aplicação, analisava a informação, produzia métricas (como rácio total das linhas de código de todas as páginas, comprimento de todas as páginas visitadas, rácio de páginas subsequentes) e usava essas métricas juntamente com a informação da interação do utilizador para deduzir *padrões UI* via um conjunto de regras heurísticas.

4.4. Arquitectura

A abordagem pode ser dividida em três componentes: **WebsiteExplorer**, **LogProcessor**, e **PatternInferrer**. A componente **WebsiteExplorer** carrega configurações de utilizador (que conseguem modificar quase todas as componentes da ferramenta), interage automaticamente com a aplicação e produz um ficheiro contendo o traço de execução com todas as ações tomadas. A componente **LogProcessor** é um analisador de texto, que analisa o ficheiro anterior, lê cada linha, procura por palavras-chave importantes, usa-as para identificar a ação contida na linha, e produz um ficheiro de traços de execução processado. Finalmente, a componente **PatternInferrer** analisa o último ficheiro, identifica os *padrões UI*, a sua localização na página e quaisquer parâmetros existentes, e produz um ficheiro PARADIGM com os resultados, sendo as únicas exceções os padrões **Menu** e **MasterDetail**. Dado que não se pode contar que o explorador interaja com todos os elementos que pertencem a esses padrões em cada página, elementos que pertencem a esses padrões são encontrados por análise do código fonte da página e da extração de todos os elementos que obedeçam a um conjunto de regras (que podem ser mudadas através de configurações de utilizador e carregadas para a ferramenta). Os padrões identificáveis por esta ferramenta são todos os padrões de interface definidos la linguagem PARADIGM mais o padrão **Menu** (uma coleção de ligações de navegação a outras páginas Web dentro de uma página Web, geralmente organizados numa estrutura em árvore) que vai ser incluído na linguagem PARADIGM no futuro.

5. Conclusões

Esta dissertação pretende continuar o trabalho realizado na ferramenta PARADIGM-RE, uma ferramenta responsável por extrair um modelo parcial de uma aplicação Web da própria aplicação através de engenharia reversa.

A ferramenta anterior precisava que o utilizador interagisse com a aplicação para a explorar, registava a informação associada, analisava a informação guardada da interação, código fonte e URLs das páginas visitadas, e deduzia padrões através de um conjunto de regras heurísticas; a ferramenta atual é totalmente automática, requer apenas um ficheiro com o traço de execução, e deduz padrões através da análise sintática do resultado da análise lexical do ficheiro de traço de execução e, para alguns padrões, da análise do código fonte da página.

Referências

[1] Miguel Nabuco, Ana CR Paiva, Rui Camacho, e Joao Pascoal Faria. Inferring ui patterns with inductive logic programming. Em *Information Systems and Technologies (CISTI), 2013 8th Iberian Conference on*, páginas 1–5. IEEE, 2013.