# *Pattern based user interface generation*


## Resumo

O desenvolvimento da *interface* com o utilizador é provavelmente a fase mais importante no desenvolvimento de *software*. A sua qualidade pode muitas vezes definir o seu sucesso junto dos clientes.

Apesar de todas as ferramentas e toda a bibliografia existentes continua a ser difícil construir boas *interfaces*. Como tal faz sentido que se faça um maior aproveitamento dos padrões que já existem, já foram testados e é sabido que têm em atenção as questões de usabilidade.

Este trabalho tem como objetivo estudar e implementar uma ferramenta capaz de interpretar um conjunto de padrões armazenados com um formato específico, liga-los à camada de negócio através de anotações no código fonte e gerar *interfaces* baseadas nas funcionalidades implementadas e no padrão base.


## Contextualização

O desenvolvimento de *software* é normalmente dividido por camadas. O padrão mais utilizado consiste em dividir a aplicação em três camadas, a camada de dados, a camada de negócio e a camada de apresentação. Sendo que a primeira faz a *interfaces* com os dados (pode ser uma base dados ou outra estrutura de armazenamento), a segunda trata toda a lógica da aplicação e a terceira faz a *interface* com o utilizador. A camada de apresentação é muitas vezes a que mais tempo leva ser desenvolvida muito devido à sua complexidade mas isto também quer dizer que existe potencial para melhorar as metodologias aplicadas de forma a agilizar todo o processo. Neste trabalho vamos estudar a geração automática da camada de apresentação.

A geração de *interfaces* deve ser, tal como qualquer outro processo de automação, fortemente baseada em padrões. Assim, os padrões utilizados são tão importantes para o sucesso da ferramenta como os algoritmos que utiliza. No caso particular dos padrões de *interfaces*, a documentação e selecção de padrões é um desafio pois são difíceis de documentar de forma eficiente.

Os padrões de desenho para componentes de *software* orientado a objectos estão bem documentados (1). A sua documentação inclui parâmetros descritivos com nome, motivação, casos de uso ou padrões relacionados e ainda uma especificação estrutural que recorre à linguagem de modelação UML. Esta estratégia de documentação torna os padrões facilmente compreensíveis por humanos e ao mesmo tempo, graças à sua especificação em UML, interpretados por máquinas.

Em (2) são identificados dois tipos de definições de padrões de *interface*. Primeiro os *Descriptive Patterns* e em segundo *Generative Patterns*. Sendo que os primeiros são mais genéricos e descritivos tendo como principal objectivo ser interpretados por humanos. *Descriptive Patterns* podem ser representados recorrendo à linguagem PLML (*Pattern Language Markup Language*). Esta linguagem

serve para lida por humanos e a sua estrutura não é ideal para ser interpretada por autómatos. Os segundos favorecem mais características como expressividade e *generatividade*[1] sendo que o seu principal objectivo é ser interpretado por uma máquina. Um exemplo é o UsiXML. Uma linguagem para especificação de *interfaces* baseada em XML. Os *Generative Patterns* parecem mais interessantes quando o objectivo é gerar *interfaces*

Em (3) é descrito um formato que tira partido das duas linguagens referidas no capítulo acima e foi batizado UsiPXML (*User Interface Pattern Extensible Markup Language*). A estrutura da linguagem está representada na Figura 1 - Estrutura do UsiPXML. A linguagem é dividida em duas camadas. A primeira fornece informação contextual que deve ser interpretada por humanos e para isso faz uso da linguagem PLML descrita acima. A camada inferior fornece informação de implementação fazendo uso da linguagem UsiXML com algumas extensões que permitem atributos estruturais, uso de variáveis e referências a outros padrões. Esta última extensão parece particularmente interessante pois promove a reutilização de código.

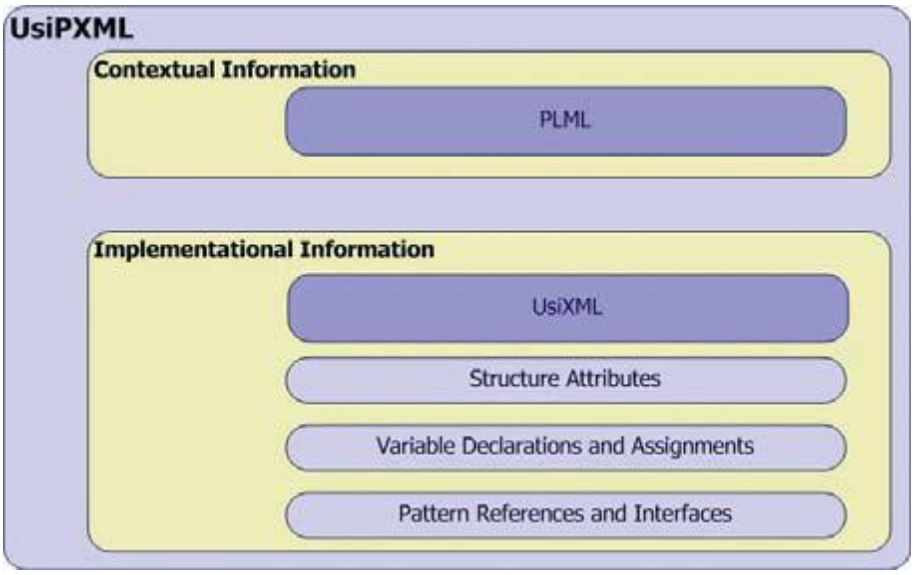


Figura 1 - Estrutura do UsiPXML

As duas últimas linguagens primam pela *generatividade* e parecem ser as mais adequadas para este trabalho sendo que a primeira é mais concisa enquanto a segunda guarda informação numa forma mais estruturada.

## Motivação

A construção de *interfaces* com o utilizador constitui um processo complexo e muito exigente em termos de tempo envolvendo a colaboração entre programadores e *designers* (4). Em muitos casos, envolve até a construção de várias *interfaces* para diferentes dispositivos. Como tal, faz sentido o investimento no desenvolvimento de técnicas de automação deste processo, ou parte dele, de forma a

[1] Capacidade de ser interpretado por uma máquina com o objectivo de gerar código fonte.

tornar esta importante etapa o menos dispendiosa possível. No entanto a maioria das soluções actuais para geração automática de *interfaces* não têm em conta aspectos de usabilidade por não possuírem toda informação necessária para o efeito.

Na conjuntura actual é claro que qualquer forma de criação de *interfaces* (automática ou manual) que não tenha em conta todos os aspectos de usabilidade está destinada ao insucesso pois o factor diferenciador de uma aplicação é cada vez menos tecnológico e prende-se cada vez mais com a forma como interage com o utilizador final.

A informação que não pode ser inferida a partir do código fonte tem que ser fornecida à ferramenta que vai gerar a *interface*. Em casos semelhantes onde é requerida informação adicional para efectuar, de forma automática, uma mudança de paradigma recorreu-se ao uso de anotações (ex. *Hibernate*) que podem tanto ser inseridas no código da aplicação ou em ficheiros à parte com uma sintaxe própria, normalmente baseada em *XML.*

Outro aspecto importante quando se tenta automatizar um processo é o recurso a padrões. Algo de que as soluções actuais para esta temática não estão a tirar o devido partido. Já existem diversos padrões para construção de *interfaces* devidamente documentados e que podem e devem ser usados. Assim um utilizador terá a possibilidade de instruir a ferramenta para recorrer a um determinado padrão de *interface* para uma determinada funcionalidade.

Nos últimos anos as aplicações *Web* têm vindo a ocupar uma fatia importante no negócio do *software* forçando o desenvolvimento de novas tecnologias a um ritmo impressionante. Uma das tecnologias mais usadas na construção de aplicações *Web* é o JEE. Trata-se de uma plataforma que permite desenvolver aplicações na linguagem *Java* destinadas a correr em servidores aplicacionais como *JBoss* ou *Glassfish* e são, normalmente, acedidas a partir de um *browser.* A alta taxa de utilização deve-se tanto à diversidade de *frameworks* existentes e que facilitam o desenvolvimento como à escalabilidade dos servidores aplicacionais que adoptam a plataforma JEE.

Em conclusão, existe uma necessidade de melhorar o processo de criação de *interfaces*. Essa necessidade pode ser colmatada por uma ferramenta coma capacidade de automatizar parte do processo. A criação de *interfaces* está restringida por parâmetros de usabilidade muito importantes para o sucesso de uma aplicação. De forma a possibilitar uma ferramenta a ter em conta estas restrições podem ser passados através de anotações alguns parâmetros que façam a ligação entre o código e um padrão de *interface* que já tenha sido desenvolvido tendo em conta a usabilidade. Quanto a tecnologia, as mais importantes nos tempos que correm são as tecnologias viradas para a *Web*. Uma das mais utilizadas é a plataforma JEE mostrando-se assim como um excelente ponto de partida para um projecto deste tipo.

## Objectivos

O principal objectivo deste trabalho é tornar o processo de desenvolvimento de software mais eficiente ao simplificar a fase de criação de *interfaces*.

Assim pretende-se criar uma ferramenta que seja capaz de processar o código da camada de negócio de uma aplicação e, através do recurso a anotações no código fonte, fazer a ligação a um padrão de *interface* previamente seleccionado gerando assim a camada de apresentação para a aplicação.

Para o efeito vai-se tirar partido das linguagens existentes para especificação de *interfaces*, como é o caso do UsiXML e UsiPXML, ambas referidas na contextualização.

## Metodologia

Para este trabalho de mestrado será aplicada uma metodologia para várias fases. Numa primeira fase será feita uma pesquisa bibliográfica sobre informação relevante para o tema, nomeadamente relativa aos assuntos referidos nas secções de contextualização e motivação. O levantamento e síntese do estado da arte será o próximo passo tendo como base a recolha de informação efetuada na primeira fase. A fase seguinte será a conceção da arquitetura da ferramenta a desenvolver seguida pelo desenvolvimento da mesma. Numa fase em que a ferramenta esteja estabilizada deve-se proceder ao desenvolvimento de um caso de estudo que tire partido das funcionalidades e permita tirar conclusões sobre a sua relevância no panorama atual. Por último será feita uma análise dos resultados obtidos.

## Calendarização

A duração da tese será aproximadamente de 10 meses. Mesmo com os objetivos bem definidos as datas apresentadas em baixo são uma aproximação e podem sofrer alterações com o decorrer do tempo. A dissertação será escrita em paralelo com todas as fases.

- **Outubro a Dezembro:** Pesquisa e leitura bibliográfica.
- **Dezembro a Janeiro:** Estudo e resumo da bibliografia selecionada. Elaboração do estado da arte.
- **Fevereiro a Abril:** Conceção da arquitetura da ferramenta a desenvolver.
- **Abril a Julho:** Desenvolvimento da ferramenta propriamente dita.
- **Agosto a Setembro:** Desenvolvimento de um caso de estudo e análise dos resultados obtidos.
- **Setembro a Agosto:** Finalização da escrita da dissertação.

## Bibliografia

1. **Gamma, Erich, et al.** *Design Patterns - Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley, 1994.

2. *Generative Pattern-Based Design of User Interfaces.* **Vanderdonckt, Jean e Simarro, Francisco Montero.** 2010.

3. *Different Kinds of Pattern Support for Interactive Systems.* **Forbrig, Peter e Wolff, Andreas.** 2010.

4. *An annotation tool for enhancing the user interface Generation Process For Services.* **Izquierdo, Paoli, et al.** 2009.

**Universidade do Minho, Braga, 4 de Novembro de 2011**

**O Mestrando,**

_____

André Lopes Barbosa

O Orientador,

_____

António Nestor Ribeiro