A Generator of User Interface Prototypes for the IVY Workbench

João Miguel Araújo
HASLab/INESC TEC &
Dept. Informática/Univ. do Minho
Braga, Portugal
joao_matela96@hotmail.com

Rui Couto
HASLab/INESC TEC &
Dept. Informática/Univ. do Minho
Braga, Portugal
rui.couto@di.uminho.pt

José Creissac Campos
HASLab/INESC TEC &
Dept. Informática/Univ. do Minho
Braga, Portugal
jose.campos@di.uminho.pt

***Resumo*—O IVY Workbench é uma ferramenta que suporta a modelação do comportamento de sistemas interativos e a verificação formal dos mesmos. A ferramenta contém um conjunto de *plugins* que suportam o processo de modelação e análise, incluindo um editor de modelos, um verificador de propriedades e um animador. Este último permite visualizar e interagir com os modelos, mas não suporta associá-los a protótipos representativos do sistemas. A interação com os modelos facilita a sua validação por parte de quem os está a desenvolver. Não facilita, no entanto, a comunicação com os potenciais *clientes* do sistema modelado, para quem um protótipo será um meio mais eficaz de comunicação. Este artigo apresenta o trabalho realizado para remediar esta lacuna no IVY. O artigo detalha a investigação preliminar, decisões arquiteturais e o resultado obtido.**

***Abstract*—The IVY Workbench is a tool that supports the modeling and formal verification of interactive systems. The tool features a set of plugins that support the modeling and analysis process, including a models editor, a properties checker, and a models animator. The latter, allows visualizing and interacting with a model, but does not support associating it with a prototype of the system. Interaction with the model facilitates its validation by modelling experts. It does not, however, facilitate communication with domain experts and users, to whom a prototype would be a more effective means of communication. This article presents the work done to remedy this gap in IVY. The article details the preliminary research carried out, architectural decisions and the obtained end result.**

***Index Terms*—Interaction, Usability, Prototypes, Formal verification, System analysis and design**

I. Introdução

A interface de um programa é um elemento importante na experiência que o utilizador tem com o software, pois constitui o principal método de interação com a lógica do programa. A existência de métodos fiáveis que nos forneçam formas de analisar de modo rigoroso e exaustivo o comportamento da interface torna-se indispensável quando é necessário fornecer garantias sobre a qualidade de um sistema interativo [1].

Uma das soluções mais comuns para comunicar o *design* de uma interface são ferramentas de prototipagem [2]. Em fases iniciais do processo de concepção é comum a utilização de *mockups* (protótipos de baixa fidelidade). São exemplos de ferramentas que suportam este tipo de prototipagem, o Balsamiq Mockups[1], o Pencil Project[2] e o Mockplus[3]. Estas ferramentas permitem editar os mais variados elementos do protótipo da interface e visualizar o aspeto desta, mas fornecem funcionalidades de animação dos protótipos muito limitadas, normalmente restringidas a transições entre os vários ecrãs. Transições estas, definidas de forma estática no protótipo.

O IVY Workbench [3] adopta uma abordagem diferente, focada na modelação e análise formal de sistemas interactivos [1]. Para este propósito, recorre a Modal Action Logic (MAL) [4] para descrever os modelos. A linguagem MAL permite descrever as ações suportadas pela interface e de que modo estas alteram o estado do sistema, com um especial foco nas interações por parte dos utilizadores. Os modelos podem ser validados face a propriedades da interface (por exemplo, existência de *undo*) expressas em Computational Tree Logic (CTL) [5]. O IVY utiliza um compilador que analisa os modelos especificados em MAL e os compila para a linguagem de entrada do *model checker* NuSMV [6], para que seja realizado o passo de verificação formal das propriedades definidas em CTL. Após a verificação, o IVY permite-nos analisar os resultados da verificação das propriedades no modelo especificado (em caso de falha, na forma de comportamentos da interface que não obedecem à propriedade), assim como modificar essas mesmas propriedades e acrescentar novas para verificar cenários adicionais de utilização.

Para auxiliar a análise dos resultados da verificação existe um visualizador de traços (Trace Analyzer) e um animador de modelos (AniMAL). O Trace Analyzer permite visualisar os resultados produzidos pelo NuSMV, quando a verificação das propriedades face ao modelo falha. Utilizando-o, é possível analisar a sequência de ações que demonstra a falsidade da propriedade. Já o AniMAL permite-nos criar a nossa própria sequência de ações, partindo do estado inicial do modelo. Tal é útil para melhor visualizar que ações são possíveis em que estados, prevenindo assim possíveis falhas no modelo.

A Figura 1 apresenta a atual arquitetura do IVY. Nesta, é possível observar a estrutura modular da ferramenta. Ao

[1]https://balsamiq.com/, visitado a 5 de julho de 2019
[2]https://pencil.evolus.vn/, visitado a 5 de julho de 2019
[3]https://www.mockplus.com/, visitado a 5 de julho de 2019

978-1-7281-6378-9/19/$31.00 ©2019 IEEE

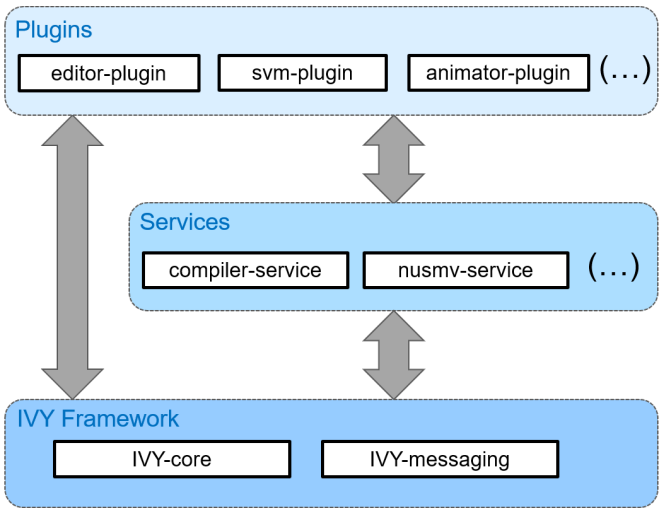

Figura 1. Arquitetura do IVY Workbench (v. 2) [7]

nível mais baixo, o núcleo (IVY Framework) fornece as funcionalidades essenciais, como a infraestrutura da interface com o utilizador e o sistema de comunicação entre *plugins* e serviços. No nível intermédio existem os serviços que fornecem funcionalidades essenciais à própria aplicação. Estes são usados pelos *plugins* e não são expostos ao utilizador. Por último, na camada superior existem os *plugins* que fornecem as ferramentas com que o utilizador irá interagir, como o editor de modelos, ou visualizador de propriedades.

Neste artigo apresentamos um novo *plugin* que adiciona à ferramenta capacidades de prototipagem. A criação dos protótipos deverá passar pela importação de *mockups* gerados em ferramentas de prototipagem de interfaces, sobre os quais iremos mapear os controlos da interface explicitados no modelo. O objetivo é poder animar o protótipo de acordo com a lógica de interação definida no modelo, juntando assim a facilidade de comunicação dos *mockups*, com o rigor do comportamento definido no modelo formal.

O artigo encontra-se estruturado do seguinte modo: na Secção II são estudadas ferramentas similares ao IVY e as suas abordagens à análise e construção de modelos de sistemas interativos; a Secção III define os requisitos do *plugin* a desenvolver; segue-se uma explicação da arquitetura do *plugin* e de decisões tomadas para o seu desenvolvimento, na Secção IV; por fim, será apresentado o *plugin* desenvolvido (Secção V) e um exemplo de aplicação (Secção VI); o artigo termina com uma análise dos resultados obtidos e apontadores para trabalho futuro, na Secção VII.

II. Trabalho relacionado

Para melhor percebermos o alcance do que se pretende implementar é importante referir outras ferramentas de análise da interface, com as suas diferentes abordagens. Nesta secção são analisados as ferramentas CogTool [8] e PVSio-Web [9].

A. CogTool

O CogTool [8] é uma ferramenta de modelação e análise de interfaces focada na análise da eficiência da interação. Para tal, incluí uma arquitectura cognitiva que procura capturar o modo como os utilizadores interagem. Neste momento, a ferramenta não se encontra em desenvolvimento ativo, tendo a última atualização ocorrido em 2014. Apesar disso, considera-se que os princípios e conceitos que esta aplica ainda são relevantes neste contexto.

O principal objetivo da ferramenta é ajudar no desenho de interfaces com o utilizador, para que efetuar tarefas comuns no programa seja intuitivo e rápido. O CogTool segue o seguinte princípio: "Tornar tarefas frequentes fáceis e tarefas menos frequentes possíveis" [8, Secção 1.1]. Para atingir este objetivo, o CogTool aplica modelos preditivos de comportamento humano. A ferramenta permite desenhar a interface e definir as sequências de ações do utilizador para as tarefas pretendidas. O modelo cognitivo incorporado calcula, então, uma estimativa do tempo que o utilizador irá demorar a efetuar essas tarefas. O modelo de previsões implementado é baseado na técnica *Keystroke-Level Modelling* (KLM) [10]. De forma sucinta, esta técnica tem em consideração diversos fatores relacionados com a realização de uma tarefa na interface, tais como o número de teclas a pressionar no teclado, a distância que o cursor irá percorrer e o nível de perícia do utilizador, só para listar alguns, utilizando-os no cálculo do tempo que o utilizador irá demorar a completar a tarefa.

A interface utiliza o conceito de *storyboards* ou *frames*, onde cada *frame* representa a interface num dado momento de interação. Em cada um destes *frames* é possível definir uma imagem, representativa da interface a modelar, e sobrepor controlos sobre ela. A interação com estes controlos permite avançar para o próximo *frame*. Após os *storyboards* criados, é possível demonstrar a execução das tarefas pretendidas, algo semelhante a executar uma ferramenta de *capture-replay*, e posteriormente obter o tempo necessário para executar as tarefas descritas.

Comparando o CogTool com o IVY, a primeira é uma ferramenta que apresenta algumas características interessantes. Enquanto o IVY está orientado para a modelação do comportamento do sistema no seu todo, o CogTool foca-se mais na descrição das tarefas que o utilizador pode efetuar no sistema e na extração de dados de usabilidade a partir destas. Uma das principais desvantagens que o CogTool tem sobre o IVY é estar mais orientado para interfaces gráficas tradicionais e não permite a análise de outro tipo de sistemas. Tal é devido ao modelo cognitivo utilizado, estando a descrição das tarefas limitada aos operadores por este suportados. Apesar deste detalhe, o conceito de *frames* para a representação de ações no sistema e a possibilidade de importação de imagens como fundo para os controlos do protótipo são funcionalidades relevantes a ter em conta durante este projeto.

B. PVSio-Web

O PVSio-Web [9] é uma ferramenta de modelação e análise formal de interfaces, com foco na construção de protótipos.

Explicar o modo de funcionamento da ferramenta e dos princípios em que assenta pode ser feito com base no nome da mesma. A sigla PVS [11] significa *Prototype Verification System*, um demonstrador de teoremas que assenta numa lógica de ordem superior. O PVSio [12] é uma extensão do PVS que facilita a utilização do avaliador de expressões (*ground evaluator*) do demonstrador de teoremas, adicionando um conjunto de facilidades de programação imperativa. Em particular, uma biblioteca de *input/output*. O PVSio-Web, utilizando o PVSio para comunicar com o PVS, suporta a construção e animação de protótipos, recorrendo ao demonstrador de teoremas para dinamicamente calcular o comportamento definido pelos modelos.

O PVSio-Web utiliza um cliente baseado em tecnologias Web, mais especificamente em NodeJS, para abstrair a lógica do PVSio [9]. O construtor de protótipos da interface gráfica interage com o servidor que contém os processos do PVSio e PVS através de WebSockets. Este modelo de funcionamento permite definir protótipos, importando imagens e definindo as áreas de interatividade nessas imagens, diretamente a partir da interface gráfica do PVSio-Web. Estas áreas ficam definidas por cima do protótipo como *overlays* e a cada uma são associados atributos ou ações do modelo. Este modo de construção dos protótipos assemelha-se ao oferecido pelo CogTool, mas com um foco maior na modelação do comportamento de sistemas em vez da extração de dados de usabilidade. Por este motivo, torna-se num dos exemplos mais relevantes a ter em conta no desenvolvimento de um *plugin* de prototipagem para o IVY, e preconiza um dos caminhos pelo qual se poderá enveredar no desenho da solução a adotar.

Relativamente ao IVY, o PVSio-web apresenta a desvantagem de recorrer a um demonstrador de teoremas, o que torna o processo de verificação menos automático e mais exigente em termos de conhecimentos de métodos formais. Combinar as capacidades de verificação mais automatizada (através de *model checking*) do IVY, com as capacidades de prototipagem do PVSio-web significaria obter o melhor das duas abordagens.

III. Requisitos

As representações gráficas utilizadas no analisador de comportamentos e no animador de modelos do IVY focam-se na representação dos modelos (as suas ações e atributos – ver Figura 2). Embora sejam úteis para análise, não favorecem a comunicação do *design* da interface a não-peritos. Tal seria melhor conseguido com uma representação focada no *design* da interface (ou seja, com protótipos da interface), tal como referido acima. Na verdade, já anteriormente se procurou adicionar ao IVY capacidades de prototipagem [13]. No entanto, a abordagem seguida era de difícil manutenção e utilização, uma vez que forçava a que o protótipo fosse definido no próprio IVY. Pretende-se, agora, uma abordagem mais flexível e de mais fácil integração com o processo de conceção e desenvolvimento de sistemas interativos.

Esta nova funcionalidade do IVY deverá permitir importar o protótipo de um sistema interativo, sob a forma de *mockups*,

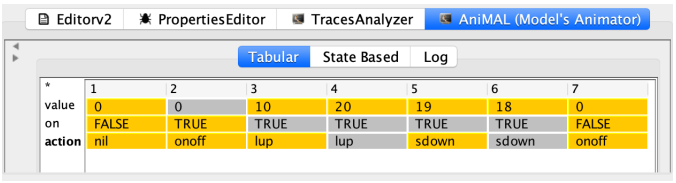

Figura 2. Representação tabular do comportamento de um modelo simples com dois atributos (value e on)

e estabelecer uma mapeamento entre estes e o modelo formal do sistema, descrito no IVY. Tendo em consideração estes objetivos, podemos formular o seguinte conjunto de requisitos funcionais para o *plugin* a desenvolver:

1) o plugin deverá suportar a importação de um ou mais ficheiros que descrevam os *mockups* da interface desejada;
2) os ficheiros importados devem estar num formato aberto e suportado por ferramentas de prototipagem para facilitar o uso do *plugin*;
3) deverá ser possível a identificação/especificação de áreas de interatividade dentro dos *mockups* importados;
4) deverá ser possível mapear essas áreas para atributos e ações do modelo, criando um protótipo interativo;
5) o *plugin* deverá suportar vários tipos de interações com os elementos interativos do protótipo;
6) deverá ser possível animar o protótipo para visualizar o comportamento do sistema modelado;
7) deverá ser possível guardar o estado do animador e posteriormente restaurá-lo e retomar o trabalho.

IV. Conceção do Plugin

Esta secção descreve as decisões tomadas no desenvolvimento do *plugin*.

A. Importação do Protótipo

Como definido acima, os requisitos para o novo *plugin* especificam que o ficheiro que descreve o protótipo deve estar num formato aberto (i.e. independente da plataforma onde foi criado e de livre utilização) e de fácil criação. Tal especificação levou à identificação de dois tipos de solução, e dois caminhos para o desenvolvimento desta nova funcionalidade:

- utilização de imagens *raster* como base para a definição dos protótipos (cf. CogTool e PVSio-web)
- utilização de formatos estruturados como base para a definição dos protótipos

Ou seja, a diferença está entre ter o *mockup* representado numa imagem como PNG ou JPEG (tal como as ferramentas da secção anterior), ou em algum formato produzido por ferramentas de edição de protótipos, nos quais é possível ter acesso à estrutura do *mockup*.

1) Utilização de imagens raster*:* A importação de imagens *raster* para o IVY deverá passar por escolher uma imagem representativa da interface do sistema modelado. Depois, será necessário definir os controlos sobre a imagem, representativos do comportamento definido no modelo.

A principal vantagem deste método é que a criação e importação da imagem são tarefas que se realizam facilmente. É possível utilizar qualquer tipo de editor de imagens, como o Adobe Photoshop[1] ou o Gimp[2] para a criação da dita imagem. Não há necessidade de analisar ou alterar o ficheiro que será importado. No entanto, a falta de estrutura neste tipo de ficheiro pode representar uma das suas maiores desvantagens. A incapacidade de distinguir elementos específicos implica que todo o esforço de associação destes com os componentes do modelo da interface será da responsabilidade dos utilizadores. Torna-se impossível delegar algum ou todo o esforço de identificação de elementos interativos do *mockup* no *plugin* ou, adicionalmente, evoluir este conceito e automatizar o processo de mapear os controlos do *mockup* em atributos e ações do modelo.

2) Utilização de formatos estruturados: A importação de formatos estruturados significa ter um *mockup* da interface com uma estrutura acessível. Isto permitir-nos-á atribuir o comportamento do modelo a certas partes da estrutura interna do *mockup* importado, como um botão ou um *dropdown*. Este nível de integração mais profunda com o *mockup* importado permite uma maior facilidade e automatização no mapeamento entre os elementos na interface e os elementos no modelo.

A utilização de formatos estruturados, permite também uma melhor integração do processo de modelação e análise com os processos de conceção de interfaces tradicionais. Com efeito, a criação de *mockups* é uma actividade comum na conceção de uma interface, existindo várias ferramentas disponíveis para os gerar. Todas estas ferramentas oferecem um modo semelhante de criação de *mockups* de interfaces gráficas. Disponibilizam *canvas* onde se podem inserir os vários *widgets* correspondentes aos elementos da interface. Em cada *widget* é possível personalizar propriedades individuais, como tamanho e cor, para tornar o *mockup* uma representação fiel do sistema pretendido.

Considerando os pontos acima, optou-se por utilizar esta última abordagem como base para a criação dos protótipos.

B. Formatos de Importação

Face ao decidido acima, tornou-se necessário decidir qual ou quais os formatos a suportar no processo de importação. Para tal, foi realizada uma análise aos formatos de exportação de algumas ferramentas de criação de *mockups* de interfaces com o utilizador. Foram escolhidas as seguintes ferramentas: Balsamiq Mockups[3], Pencil Project[4] e Mockplus[5].

Os programas de prototipagem mencionados permitem a exportação nos seguintes tipos de formatos: SVG (*Scalable Vector Graphics*); XML (*Extensible Markup Language*); JSON (*JavaScript Object Notation*); HTML (*Hypertext Markup Language*); PNG (*Portable Network Graphics*); JPEG (*Joint Photographic Experts Group*); PDF (*Portable Document Format*); ODT (*Open Document Format for Office Applications*).

Formatos como PNG e JPEG podem desde logo ser excluídos do processo de seleção, pois encaixam na categoria de imagens *raster*. Com os formatos PDF e ODT existe a possibilidade de extração de informação, mas com limitações. Das ferramentas analisadas, apenas o Pencil consegue exportar para estes formatos. Acresce que, mesmo assim, a exportação é feita na forma de imagens *raster* embebidas nos ficheiros PDF ou ODT. Assim, podemos descartar estes formatos da lista de possibilidades.

Depois, temos os formatos de *markup* XML e HTML. O primeiro é gerado pelo Balsamiq Mockups, enquanto o segundo é gerado tanto pelo Balsamiq como pelo Mockplus. O HTML gerado pelos dois não contém nenhuma informação estrutural do protótipo, pelo que pode ser ignorado. Já o XML contém esta informação, mas tem a peculiaridade de não ser um formato diretamente exportado pelos programas mencionados. Tal significa que só se pode obter um XML no Balsamiq Mockups se se exportar para HTML e no Mockplus se se exportar para o formato ZIP (onde o ficheiro se encontrará nas diretorias geradas). No entanto, há um detalhe que inviabiliza o uso do formato: o esquema XML (*XML Schema*) do ficheiro varia de acordo com o programa que o cria. Caso se optasse por utilizar o XML gerado por um destes programas, seria necessário limitar as opções disponíveis para criação de protótipos. Utilizar-se-ia exclusivamente o programa que exportasse com a estrutura pretendida. Tendo este aspeto em consideração, pode-se também descartar o XML.

No caso do formato JSON, este só é gerado pelo Balsamiq Mockups, e contém um formato próprio do programa, pelo que se verifica o mesmo problema mencionado para o formato XML.

Por fim, resta o formato SVG [14], uma norma desenvolvida pela W3C, com a primeira versão lançada em 2001, que continua a ser iterada até hoje, estando uma nova versão a ser preparada (versão 2.0[6]). Este formato foi desenvolvido para solucionar um dos principais problemas dos formatos de imagens mais comuns utilizados na internet: a escalabilidade. Tomando uma imagem no formato JPG, por exemplo, quando se aumenta a escala desta para além do especificado, começamos a notar a degradação da qualidade da imagem. Isto deve-se ao facto que este tipo de formatos apenas guardam um conjunto fixo de pixeis. Já com o SVG, guarda-se um conjunto fixo de formas, que depois são desenhadas pelo programa de visualização. Esta solução permite manter a definição da imagem, independentemente do seu tamanho.

A especificação das formas que compõem o SVG é realizada no formato XML. Isto permite a visualização em qualquer editor de texto, ou seja, facilita a extração de informação. Com a especificação das formas também não teremos problemas na distinção entre os vários elementos do protótipo, já que podemos adicionar um identificador único a cada elemento. O

[1]https://www.adobe.com/products/photoshop.html, visitado a 4 de julho de 2019

[2]https://www.gimp.org/, visitado a 4 de julho de 2019

[3]https://balsamiq.com/, visitado a 5 de julho de 2019

[4]https://pencil.evolus.vn/, visitado a 5 de julho de 2019

[5]https://www.mockplus.com/, visitado a 5 de julho de 2019

[6]https://www.w3.org/TR/2018/CR-SVG2-20181004/, visitado em 5 de julho de 2019

SVG suporta ainda camadas (*layers*), possibilitando a especificação de todas as variações do protótipo num só ficheiro. Para além do mais, o formato tem suporte nativo para animações por scripting, utilizando *Synchronized Multimedia Integration Language* (SMIL), ECMAScript ou até mesmo Javascript, e programação por eventos (cf. o evento `onClick`), acrescentando à riqueza de funcionalidades ao dispor.

A única ferramenta de prototipagem analisada que permite exportar para SVG é o Pencil. No entanto, como o formato utiliza um esquema XML normalizado [14, Section 2.2], a abordagem não fica restringida ao *output* e estrutura gerados por um único programa. Qualquer programa de edição de SVG poderá ser utilizado na criação dos *mockups*. Em particular, torna-se possível utilizar um editor à nossa escolha para efetuar quaisquer pequenos ajustes necessários.

Ponderando todos os formatos e a análise efetuada sobre estes, concluímos que o SVG se apresenta como o formato mais adequado às nossas necessidades. A Tabela I resume os pontos abordados.

Tabela I
RESUMO DOS FORMATOS DE EXPORTAÇÃO

Formato	Programa	Estru. Exp.	Sintaxe Uni.
SVG	Pencil	Sim	Sim
XML	Balsamiq, Mockplus	Sim	Não
HTML	Balsamiq, Mockplus	Sim[1]	Sim
JSON	Balsamiq	Sim	Não
PNG	Pencil, Balsamiq, Mockplus	Não	-
JPEG	Mockplus	Não	-
PDF	Pencil	Sim[1]	Não
ODT	Pencil	Sim[1]	Não

[1]Formato com pouca informação associada.

C. Bibliotecas

Tal como todos os componentes da ferramenta IVY, o novo *plugin* de prototipagem foi desenvolvido em Java. Nativamente, a linguagem suporta a manipulação de dados XML (necessária para identificação dos elementos no SVG) e ainda oferece a possibilidade de desenhar gráficos, utilizando a biblioteca Java2D. No entanto, para facilitar o trabalho de desenvolvimento, decidiu-se investigar bibliotecas de manipulação de SVGs.

Para efetuar a análise, definiu-se um conjunto de critérios (ver primeira coluna da Tabela II) que se podem dividir em quatro grandes categorias:

- suporte à **visualização** de SVG;
- suporte à **edição** de SVG;
- suporte à **animação** de SVG;
- qualidade da **documentação** disponível.

Foram analisadas três bibliotecas: Batik[1], SVG Salamander[2] e JFreeSVG[3]. Com base na análise (ver Tabela II), a biblioteca Batik foi selecionada para implementar o *plugin*.

[1]https://xmlgraphics.apache.org/batik/, visitado em 5 de julho de 2019

[2]https://github.com/blackears/svgSalamander, visitado em 5 de julho de 2019

[3]http://www.jfree.org/jfreesvg/, visitado em 5 de julho de 2019

Tabela II
COMPARAÇÃO DAS BIBLIOTECAS SVG

Critério de análise	Batik	SVG Salamander	JFreeSVG
Acesso à árvore DOM	✓	✓	-
Geração de SVG	✓	✓	✓
Adição de elementos	✓	✓	✓
Edição de elementos	✓	✓	✓
Remoção de elementos	✓	✓	-
Visualização	✓	✓	-
Eventos/Animação	✓	✓	-
Scripting	✓	-	-
Documentação	✓	-	✓

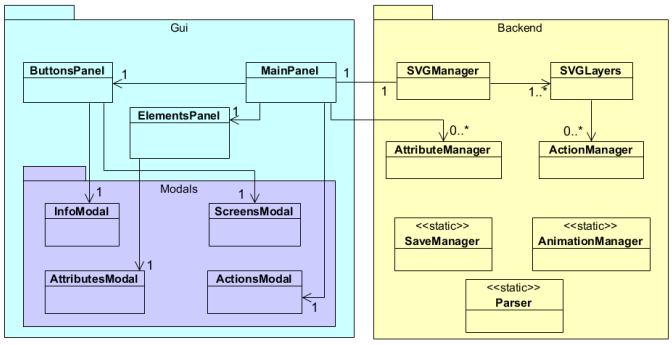

Figura 3. Estrutura do *plugin*

V. O *plugin* "PROTOTYPER"

A. Arquitectura

O *plugin* desenvolvido fornece tanto uma interface em que o utilizador pode definir e interagir com os protótipos, como um *backend* que processa os SVGs importados e gere a comunicação com o modelo. Assim, o código foi dividido em dois pacotes principais. De um lado, está o pacote denominado `Gui`, responsável por atualizar a interface, apresentar os diálogos necessários e mostrar informação sobre o estado do SVG e dos seus elementos. Do outro lado, está a infraestrutura que suporta toda a operação, denominada `Backend`. Este pacote é responsável pelo processamento do SVG e armazenamento de todas as estruturas de dados. Sempre que o utilizador efetua uma ação na interface, esta efetua um pedido ao `Backend`, que devolve a informação necessária.

O diagrama de classes do *plugin* apresentado na Figura 3 ilustra o que foi descrito. À esquerda encontra-se o pacote `Gui`, que contém as classes que gerem os elementos da interface do *plugin*. Dentro deste pode ainda observar-se o pacote `Modals` que, tal como o nome indica, contém todas as classes pertinentes aos diálogos modais, com os quais o utilizador interage para introduzir dados. À direita está o pacote `Backend`, descrito atrás.

Um detalhe importante a esclarecer tem a ver com a forma como o *plugin* comunica que com os restantes componentes da ferramenta IVY. O *plugin* foi implementado na versão 2 do IVY [7]. Esta versão implementa um mecanismo de *publish-subscribe*. O *plugin* utiliza três canais de comunicação. Um em

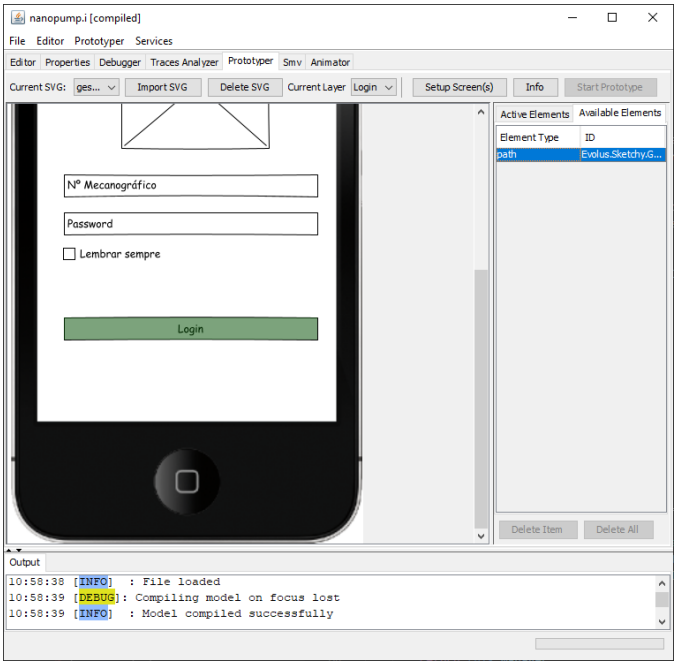

Figura 4. Interface do *plugin* Prototyper

que é publicada informação sobre o modelo e que é subscrito pelo `MainPanel` (ver Figura 3). Outros dois, para envio e receção de informação sobre a animação do modelo, são utilizados no pacote `Backend`, pelo `AnimationManager` e suportam o envio de informação sobre qual a ação do modelo a executar (em resposta a uma ação do utilizador) e receção do estado do modelo resultante da execução dessa ação.

B. Funcionalidades

A interface do novo *plugin* tem dois modos de funcionamento: o modo de edição e o modo de animação. A utilização começa sempre no modo de edição. Neste é possível preparar a animação, gerindo a ligação entre o protótipo e o modelo e ajustando os vários parâmetros. Depois de tudo estar configurado é possível iniciar o modo de animação, onde é possível interagir com o protótipo definido.

Ao selecionar o "Prototyper" dos vários separadores existentes no IVY, a interface da Figura 4 irá aparecer, correspondente ao modo de edição. A interface está dividida em três áreas principais, a partir das quais se pode aceder as funcionalidades disponibilizadas. A maior área, à esquerda, contém o SVG carregado e atualmente selecionado (neste caso é possível ver parte de um protótipo já carregado). É possível selecionar elementos do mesmo para inspeção e configuração (associação a atributos ou acções do modelo).

Para gerir os vários SVGs importados existe o painel superior que alberga todos os controlos essenciais do *plugin*. Existem botões para importar ou eliminar SVGs, controlos para selecionar entre os SVGs importados e as suas possíveis camadas e ainda botões de informação e de início da animação. Ao clicar-se neste último, é iniciado o modo de animação. Por último, e não menos importante, o painel lateral à direita tem duas tabelas, selecionáveis por separadores. A tabela "Available Elements" lista os elementos interativos disponíveis no SVG para configuração e a tabela "Active Elements" lista elementos já configurados pelo utilizador.

Como já foi mencionado, é possível mapear elementos do SVG para atributos e acções no modelo. Estes elementos passam a componentes interativos do protótipo. Ao carregar o SVG, o *plugin* determina quais os elementos que podem ser mapeados (ou seja, que componentes são interativos). A partir desse momento, o utilizador pode selecioná-los, ou diretamente no SVG clicando neles, ou pelo painel lateral, e associá-los a ações ou atributos do modelo. Quando se selecionam elementos numa das tabelas do painel lateral, estes são sinalizados no SVG através de um sombreado.

Elementos associados a ações do modelo serão depois utilizados no modo de animação para suportar a interação entre o utilizador e o protótipo. É possível definir dois tipos de interação para estes elementos: com o clique do rato ou com o pressionar de uma tecla no teclado.

Os atributos do modelo podem ser utilizados de dois modos: serem associados a elementos textuais do SVG, de modo a que o seu valor seja apresentado na interface durante a animação; ou (no caso dos atributos booleanos) serem utilizados para determinar qual a camada do *mockup* que deve ser apresentada em cada momento.

Em modo de animação é apresentado o *mockup*, podendo o utilizador interagir com o mesmo. O comportamento do protótipo é então regido pela simulação do modelo no NuSMV, sendo o *plugin* responsável pelo mapeamento de ações do utilizador para ações do modelo e pela apresentação dos atributos do estado do modelo.

Finalmente, é possível guardar o protótipo e as associações realizadas e mais tarde reiniciar o trabalho sem ter de realizar as mesmas associações sempre que se abra o *plugin*.

Na sua forma atual, o Prototyper cumpre os requisitos definidos inicialmente. Suporta a importação de um ou mais *mockups*, permite a especificação de áreas de interatividade para associá-las a ações e atributos do modelo, utiliza um formato aberto e de fácil criação para representar os protótipos, assim como suporta a animação dos protótipos (melhor detalhada na secção a seguir) e guardar ou restaurar o trabalho efetuado.

VI. Exemplo de aplicação

Com o objetivo de ilustrar a utilização do *plugin* desenvolvido, apresenta-se agora um exemplo de utilização do mesmo para obtenção de um protótipo de uma bomba de infusão. Trata-se de um dispositivo responsável pela injeção intravenosa de medicamentos ou fluídos em pacientes.

A. O modelo

Consideremos que existe já um modelo da bomba de infusão em causa (ver Listagem 1). Por uma questão de facilidade de apresentação no contexto do artigo, o modelo captura simplesmente a manipulação da quantidade de fluído a ser injetado.

Listagem 1. Modelo da bomba de infusão em MAL

```
1  defines
2    MAXV = 25
3  types
4    TDisplayValue = 0..MAXV
5
6  interactor main
7    attributes
8      value: TDisplayValue
9      on: boolean
10   actions
11     onoff sup sdown lup ldown
12   axioms
13     [] !on & value = 0
14     [onoff] on' = !on & value'=0
15     per(sup) -> value<MAXV & on
16     [sup] value' = value + 1 & on'
17     per(sdown) -> value>0 & on
18     [sdown] value' = value - 1 & on'
19     per(lup) -> value <= (MAXV-10) & on
20     [lup] value' = value + 10 & on'
21     per(ldown) -> value >= 10 & on
22     [ldown] value' = value - 10 & on'
```

Exemplos de modelos realistas podem ser encontrados em, por exemplo, [15], [16].

O modelo tem especificados atributos, ações e axiomas. Nas linhas 8 e 9 estão declarados dois atributos:

- `on` - do tipo `boolean`, pode ter o valor de verdadeiro ou falso. Este atributo é utilizado para indicar se a bomba de infusão se encontra ligada, ou não.
- `value` - pode ter um valor numérico, no intervalo de 0 a 25. Este atributo guarda a quantidade de fluído a ser infundido pela bomba de infusão quando está ligada.

Na linha 11 são apresentadas as ações do modelo:

- `onoff` - liga ou desliga a bomba de infusão se estiver desligada ou ligada, respetivamente.
- `sup` - incrementa o fluído a ser infundido em 1 unidade.
- `sdown` - decrementa o fluído a ser infundido em 1 unidade.
- `lup` - incrementa o fluído a ser infundido em 10 unidades.
- `ldown` - decrementa o fluído a ser infundido em 10 unidade.

Por fim, existem os axiomas, visíveis entre as linhas 13 e 22, que definem o comportamento do modelo com base nas ações e atributos mencionados atrás.

B. *O* Mockup

Na Figura 5 apresenta-se um *mockup* criado no Pencil como representação visual do modelo descrito. A bomba de infusão pode estar desligada ou ligada. Estes dois estados são representados pelas duas imagens na figura. No topo encontra-se a bomba desligada e em baixo a bomba ligada (note-se o ecrã iluminado). No SVG exportado estas duas figuras estão descritas como duas camadas. Os botões de controlo são comuns aos dois estados, e representam todas as ações descritas anteriormente. A distinção entre botões que aumentam ou

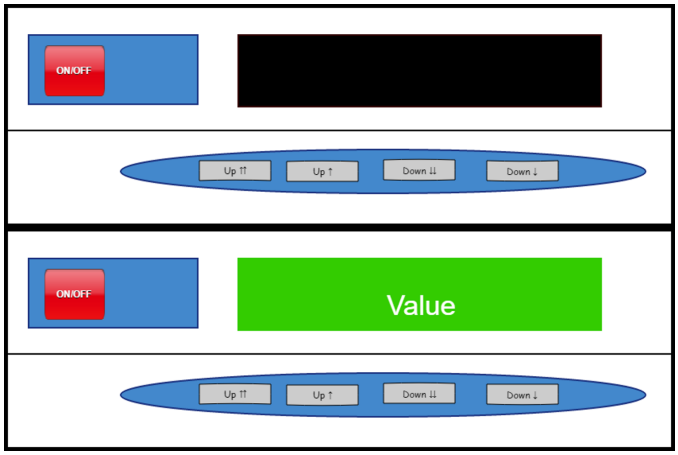

Figura 5. *Mockups* da bomba de infusão

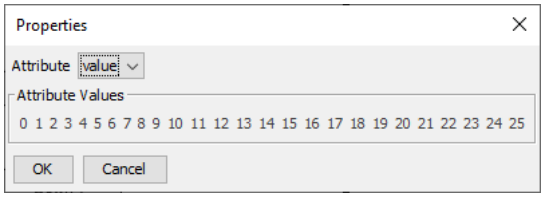

Figura 6. Selecção do atributo a associar

decrementam o fluído a ser injetado em 1 ou 10 unidades é realizada pelo número de setas a acompanhar o rótulo. Os botões "UP↑"/"DOWN↓" aumentam/decrementam o valor em 1 unidade, respetivamente. Os botões "UP↑↑"/"DOWN↓↓" aumentam/decrementam o valor em 10 unidades.

C. *Construção do protótipo*

Utilizando as funcionalidades fornecidas pelo *plugin*, realizou-se a associação entre as ações e atributos do modelo e os elementos existentes no *mockup*. A visibilidade de cada uma das camadas do *mockup* foi definida com recurso ao atributo `on`, uma vez que este modela o facto de o dispositivo estar ligado ou desligado. O valor `false` do atributo (dispositivo desligado) foi associado à imagem superior da Figura 5, o valor `true` (dispositivo ligado) à imagem inferior.

A *label* existente no *mockup* relativo ao estado ligado (com o texto "Value" na Figura 5) foi associada ao atributo `value`, para deste modo ser apresentado o valor programado na bomba em cada momento. Na Figura 6 é mostrado o formulário em que esta configuração é efectuada. O formulário é apresentado após selecção de um elemento textual no *mockup* e permite seleccionar um atributo do modelo, que lhe ficará associado.

Os botões presentes nos *mockup* foram também associados ao modelo. Neste caso, a ações. O botão com label "ON/OFF" foi, naturalmente, associado à ação `onoff`. Isso foi feito quer na camada correspondente ao estado desligado, quer na camada correspondente ao estado ligado. O resultado é, então, que premir o botão faz disparar a ação `onoff` no modelo, o que, ao alterar o valor do atributo `on`, muda a camada apresentada, da que representa o dispositivo desligado, para

Figura 7. Animação em execução

a que o representa ligado e vice-versa. Os restantes quatro botões foram apenas configurados na camada correspondente ao estado ligado: "UP↑" (ação `sup`), "DOWN↓" (ação `sdown`), "UP↑↑" (ação `lup`) e "DOWN↓↓" (ação `ldown`).

D. Animação do protótipo

Na Figura 7 pode-se observar o modo de animação do Prototyper em execução, com a ligação entre os elementos interativos do *mockup* e o modelo já configurada. Neste modo o ecrã principal (à esquerda) apresenta a animação no estado atual de execução e o painel lateral (à direita) informação sobre as ações que é possível executar (na parte superior) e o valor do estado que resultará da ação seleccionada (na parte inferior). Na sequência demonstrada na Figura 7, podemos observar que inicialmente a bomba de infusão se encontra desligada. Após ligar a bomba com o botão "ON/OFF", utilizam-se os botões "UP↑" e "UP↑↑" para incrementar o valor a infundir em uma unidade, primeiro, e em dez unidades, depois, resultando num valor final a infundir de 11.

VII. Conclusão

O IVY Workbench é uma ferramenta que permite a modelação e análise do comportamento de sistemas interativos complexos. No entanto, até à data, não possuía forma de representar estes modelos de uma forma gráfica, simples de interpretar por quem não seja perito em modelação de software. Este artigo descreve a concepção de desenvolvimento de um novo *plugin* para a ferramenta, que tem como objetivo colmatar esta lacuna.

Após a análise de algumas ferramentas de modelação e análise de sistemas interativos, foram definidos os requisitos para o novo *plugin* (o Prototyper) e discutidas as principais decisões relativas à sua conceção. Por fim, descreveu-se o *plugin* desenvolvido e foi apresentado um caso de estudo para demonstrar a sua utilização. É possível, neste momento, dizer que os requisitos definidos na Secção III foram cumpridos: o Prototyper suporta a importação de protótipos em formato SVG (um formato aberto), permite o mapeamento entre o protótipo e o modelo, assim como a possibilidade de animação dos protótipos de acordo com a lógica definida no modelo.

No futuro pretendemos adicionar suporte para *scripting* no SVG, permitindo deste modo a definição de *widgets*. Deste modo, aumenta-se a interatividade dos protótipos, ao mesmo tempo que se simplifica o seu desenvolvimento. Está também em consideração estudar a geração de modelos a partir dos *mockups*.

Acknowledgment

Este trabalho é financiado por fundos nacionais através da FCT – Fundação para a Ciência e a Tecnologia, I.P., no âmbito do projeto: UID/EEA/50014/2019.

Referências

[1] J. Campos, "High assurance interactive computing systems," in *HCI Engineering: Charting the Way towards Methods and Tools for Advanced Interactive Systems*, J. Ziegler, J. C. Campos, and L. Nigay, Eds., 2014, pp. 39–42.

[2] M. Beaudouin-Lafon and W. Mackay, "Prototyping tools and techniques," in *The Human-computer Interaction Handbook*, J. A. Jacko and A. Sears, Eds. L.E.A., 2003, ch. 52, pp. 1006–1031.

[3] J. C. Campos and M. D. Harrison, "Interaction engineering using the ivy tool," in *ACM Symposium on Engineering Interactive Computing Systems (EICS 2009)*. ACM, 2009, pp. 35–44.

[4] M. Ryan, J. Fiadeiro, and T. Maibaum, "Sharing actions and attributes in modal action logic," in *Theoretical Aspects of Computer Software*, ser. Lecture Notes in Computer Science, vol. 526. Springer, 1991, pp. 569–593.

[5] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Trans. Program. Lang. Syst.*, no. 2, pp. 244–263, 4 1996.

[6] A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "NuSMV 2: An opensource tool for symbolic model checking," *CAV '02: Proceedings of the 14th International Conference on Computer Aided Verification*, no. 3-4, pp. 359–364, 2002.

[7] R. Couto and J. Campos, "IVY 2 - a model-based analysis tool," in *The 11th ACM SIGCHI Symposium on Engineering Interactive Computing Systems - EICS 2019*. ACM, 2019, pp. 5:1–5:6.

[8] B. E. John, *CogTool User Guide*, 2nd ed., IBM T. J. Watson Research Center, 2012.

[9] P. Oladimeji, P. Masci, P. Curzon, and H. Thimbleby, "PVSio-web: a tool for rapid prototyping device user interfaces in PVS," *Electronic Communications of the EASST*, vol. 69, 2014.

[10] S. K. Card, T. P. Moran, and A. Newell, "The keystroke-level model for user performance time with interactive systems," *Commun. ACM*, no. 15, pp. 396–410, 7 1980.

[11] S. Owre, N. Shankar, J. M. Rushby, and D. W. J. Stringer-Calver, *PVS System Guide*, 2nd ed., SRI International, 2001.

[12] C. A. Muñoz, "Rapid prototyping in PVS," National Inst. of Aerospace Research, Technical Report NASA/CR-2003-212418, NIA-2003-03, 2003.

[13] N. Guerreiro, S. Mendes, V. Pinheiro, and J. C. Campos, "AniMAL - a user interface prototyper and animator for MAL interactor models," in *Interacção 2008 - Actas da 3a. Conferência Nacional em Interacção Pessoa-Máquina*. GPCG, 2008, pp. 93–102.

[14] E. Dahlström, P. Dengler, A. Grasso, C. Lilley, C. McCormack, D. Schepers, and J. Watt, "Scalable vector graphics (SVG) 1.1 (second edition)," W3C, W3C Recommendation, August 2011.

[15] J. Campos and M. Harrison, "Modelling and analysing the interactive behaviour of an infusion pump," *Electronic Communications of the EASST*, vol. 45, 2011.

[16] M. Harrison, L. Freitas, M. Drinnan, J. Campos, P. Masci, C. di Maria, and M. Whitaker, "Formal techniques in the safety analysis of software components of a new dialysis machine," *Science of Computer Programming*, vol. 175, pp. 17–34, April 2019.