"Ever tried. Ever failed.

No matter. Try Again. Fail again. Fail better."

Samuel Beckett

Resumo

Este trabalho apresenta uma plataforma, denominada *Integrated Virtual Operator* (IVO), que suporta o desenvolvimento rápido de aplicações móveis sensíveis ao contexto por utilizadores sem nenhum conhecimento de programação. Esta plataforma é composta por um conjunto de ferramentas de composição que fornecem um ambiente de programação visual, onde os utilizadores podem facilmente definir condições de contexto e associá-las a *workflows* com actividades disponíveis na plataforma. É fornecido um *runtime* para os dispositivos móveis, que disponibiliza o suporte necessário para a execução destas aplicações. Desta forma, sempre que o contexto definido ocorrer, o dispositivo móvel do utilizador reproduzirá imediatamente o comportamento desejado, sem necessidade da sua intervenção. As aplicações desenvolvidas usando o IVO podem facilmente ser disponibilizadas a outros utilizadores através de uma plataforma *web* distribuída. É ainda apresentada neste trabalho, a avaliação da plataforma feita ao nível das ferramentas de composição e ao nível do *runtime* que corre nos dispositivos móveis.

Abstract

Rapid Development Platform for Context-aware Applications Based on Mobile Devices

This work presents a platform called *Integrated Virtual Operator* (IVO) that supports the rapid development of context-aware applications by users with no programming skills. This platform is composed by a toolset of composite tools that provide a visual programming environment, where users can easily define a number of context conditions and associate them with a workflow of activities available within the platform. A mobile runtime layer is provided by another component of the system, which offers the necessary support for the execution of such applications. This way, whenever the defined context occurs, the user's mobile device will immediately produce the intended behaviour, with no need for user intervention. The applications developed using the IVO can easily be made available to other users through a distributed web platform. This work also presents, the evaluation of the platform made at the composition tools and at the runtime that runs on mobile devices.

Agradecimentos

Apesar destas linhas não serem retribuição suficiente, o meu primeiro agradecimento vai para a minha família que sempre soube compreender as minhas ausências, umas vezes físicas outras de espírito, a que foi sendo obrigada para a realização deste meu trabalho.

À ILUSTRATOWN e em particular ao Diogo Corrêa Mendes, agradeço a disponibilidade e os meios facultados no sentido de proporcionar as condições que me permitiram desenvolver, testar e validar o sistema.

Aos meus orientadores, Professor António Eduardo Dias e Professora Teresa Romão, agradeço todo o suporte, incentivo e encorajamento dado na realização deste trabalho.

Um agradecimento especial ao Professor Fernando Birra pelos seus valiosos comentários e sugestões, bem como a todos os membros do grupo de Sistemas Multimodais do Centro de Informática e Tecnologias da Informação da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, pela valiosa colaboração na validação do trabalho.

Agradeço ainda à Escola Superior de Tecnologia e Gestão de Portalegre, todas as facilidades que me permitiram gerir o tempo sempre escasso e cumprir com o plano definido.

O meu agradecimento final vai para os amigos que me acompanharam neste trabalho e me facilitaram os meios para o concluir.

Índice

Índice de Tabelas

Índice de Figuras

Abreviaturas e Siglas

API	Application Programming Interface
BPEL	Business Process Execution Language
CDATA	Character Data
ECA	Event-Condition-Action
ECW	Event-Condition-Workflow
EUD	End User Development
EUP	End User Programming
FDD	Feature Driven Development
GAT	Guard-Action-Trigger
GUID	Globally Unique Identifier
HTML	Hyper Text Markup Language
IVO	Integrated Virtual Operator
IVOML	IVO Markup Language
KML	Keyhole Markup Language
MVC	Model View Controller
NFC	Near Field Communication
ORM	Object Role Modeling
OWL	Web Ontology Language
PDA	Personal Digital Assistant
RAD	Rapid Application Development
RDF	Resource Description Framework
REST	Representational State Transfer
SIG	Sistema de Informação Geográfica
TTS	Text-To-Speech
UCD	User Centered Design
UDP	User Datagram Protocol
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

WS-BPEL	Web Services Business Process Execution Language
XML	Extensible Markup Language
XP	Extreme Programming
XSD	XML Schema Definition

Capítulo 1

Introdução

Conteúdo

Resumo

Pretende-se fornecer neste capítulo a motivação subjacente a este trabalho fazendo-se uma introdução aos principais conceitos relativos à computação sensível ao contexto, sendo dado especial foco à problemática do desenvolvimento deste tipo de aplicações. São depois apresentados os objectivos e contribuições deste trabalho para finalizar com a estrutura da dissertação.

1.1 Motivação

De acordo com a visão de Mark Weiser para a computação ubíqua, o ser humano deverá ser capaz de viver com os computadores e não apenas de interagir com eles [1]. A ideia básica do paradigma da computação ubíqua é a de que a computação se move do *desktop* para o meio ambiente, operando na periferia da nossa atenção. Na sua essência, este paradigma utiliza pequenos dispositivos, com elevado poder de computação e recursos de rede, perfeitamente integrados no ambiente, envolvendo esta heterogeneidade de dispositivos um elevado grau de interoperabilidade. Estes dispositivos, sendo baratos e abundantes espalhar-se-iam por toda a parte (pelas nossas roupas, mobília, paredes e até pelo nosso corpo), e estariam ligados em rede

partilhando dados e tornando a nossa vida mais agradável [2]. Greenfield descreve este paradigma de interacção como *"information processing dissolving in behavior"* [3].

Os avanços verificados na nanotecnologia têm permitido que cada vez mais recursos sejam adicionados a estes pequenos dispositivos possibilitando uma maior consciência do mundo dinâmico de que são parte. Pister introduziu o termo *"smart dust"*, referindo-se a pequenos sensores do tamanho de grãos de arroz que monitorizam tudo, agindo como terminações nervosas do planeta [4]. Neste mundo, o utilizador não faz mais parte do sistema (uma vez que a interacção entre o sistema e o utilizador é reduzida), tornando-se o supervisor deste, agindo apenas quando estritamente necessário [5].

Contudo, e apesar da enorme evolução tecnológica verificada desde que Mark Weiser introduziu o conceito da computação ubíqua, ainda estamos longe de dispor dos dispositivos tal como visionados por Weiser, sendo actualmente o telemóvel considerado o primeiro computador verdadeiramente ubíquo [6]. Na realidade, o telemóvel acompanha sempre os utilizadores, ajudando-os a manterem-se em contacto uns com os outros e a gerir as tarefas do dia-a-dia. Com o surgimento e a crescente utilização de dispositivos móveis como os *notebooks*, PDAs, *smartphones* e mais recentemente os *tablets*, a computação ubíqua tem-se tornado cada vez mais uma área de interesse para os investigadores. Neste trabalho, a abordagem foi a de usar dispositivos móveis tais como *smartphones* e *tablets*, como dispositivos de interacção ubíqua. A grande adopção do telemóvel no mundo inteiro permite uma ampla utilização do sistema. Estes dispositivos fornecem actualmente um conjunto grande de possibilidades de detecção, comunicação e interface com o utilizador, havendo uma grande procura do mercado por estes tipos de dispositivos [7], tornando-os ideais para o desenvolvimento de sistemas ubíquos. Embora o uso de *smartphones* e *tablets* limite a possibilidades de sensibilidade do contexto aos sensores disponíveis neste tipo de dispositivos, por outro lado facilita o uso generalizado deste tipo de aplicações, com nenhuma ou pouca configuração. No entanto, esta desvantagem tende a desaparecer uma vez que estes dispositivos cada vez mais incluem um número maior de sensores. O uso de *Near Field Communication* (NFC) [8] em dispositivos móveis abre um novo espectro de utilização de aplicações sensíveis ao contexto, o que só era possível anteriormente através de *hardware* e de uma infra-estrutura adicional. As últimas versões do Android e do iPhone já suportam NFC existindo actualmente uma grande expectativa acerca da sua utilização nas mais diversas situações do dia-a-dia.

Um tema importante de investigação abordado em computação ubíqua é a sensibilidade ao contexto, em que as aplicações se adaptam dinamicamente às mudanças nas actividades do

utilizador e do ambiente [9]. O contexto é uma situação particular do utilizador, e pode incluir, entre outros, a localização, as actividades que estão a ser realizadas, a data e a hora, e as pessoas ou equipamentos próximos. A computação sensível ao contexto envolve a detecção dessas situações e o fornecimento de informação e serviços adequados ao utilizador. Os trabalhos pioneiros de investigação em computação ubíqua, como [1, 10–12], abordam o tema da sensibilidade ao contexto e demonstraram o potencial das aplicações sensíveis ao contexto. Alguns trabalhos de investigação concentram-se no desenvolvimento de *frameworks*, *toolkits* e infra-estruturas de suporte aos programadores na construção deste tipo de aplicações. Outros projectos usam técnicas de programação pelo utilizador final, de forma a capacitá-los para a prototipagem de aplicações sensíveis ao contexto [9, 13–14]. Outros ainda fornecem ferramentas de autor que suportam a criação de aplicações sensíveis ao contexto por especialistas de domínio sendo contudo incapazes de dar uma resposta mais geral [15–17]. Verifica-se assim, faltar o suporte ao utilizador final na criação completa de aplicações sensíveis ao contexto de âmbito mais genérico utilizando dispositivos móveis como dispositivos de interacção ubíqua.

Neste trabalho é apresentada a plataforma IVO (*Integrated Virtual Operator*), a qual permite uma utilização mais ampla que os projectos referidos anteriormente possibilitando a construção de aplicações para diversos fins. O IVO permite aos utilizadores finais a criação e a disponibilização de aplicações sensíveis ao contexto sem a necessidade de escrever qualquer linha de código, usando dispositivos móveis do tipo *smartphones* e *tablets* como dispositivos de interacção ubíqua. A plataforma é composta por ferramentas de construção que permitem aos utilizadores finais definirem contextos que servem de *triggers* para o início de *workflows* que determinam as acções que devem ser realizadas sempre que o contexto se verificar segundo um modelo de funcionamento que foi chamado de evento-condição-workflow (*event-condition-workflow* ou ECW).

1.2 Objectivos e Contribuições

Este trabalho pretende contribuir para a área da computação sensível ao contexto, propondo, implementando e avaliando uma plataforma de uso genérico que facilita o desenvolvimento rápido de aplicações sensíveis ao contexto usando como dispositivo de interacção ubíquo, dispositivos móveis como *smartphones* e *tablets*. Esta plataforma fornece um conjunto de ferramentas que permite a pessoas sem conhecimentos de programação criar aplicações sensíveis ao contexto e disponibilizá-las a outros utilizadores através de um ambiente *web*.

As principais contribuições deste trabalho são:

1. Uma **arquitectura de referência** para a plataforma;

2. Uma **metodologia para o desenvolvimento** das aplicações sensíveis ao contexto criadas com a plataforma;
3. Uma **especificação de uma linguagem de marcação para definição das aplicações – IVO Markup Language ou simplesmente IVOML**;
4. Uma **ferramenta de autor** em ambiente *web* que permite aos utilizadores finais desenvolverem e disponibilizarem as aplicações;
5. Um **add-on para o Calendário do Microsoft Outlook** que usa os compromissos marcados na agenda do utilizador como base para a criação de contextos temporais;
6. Um **add-on para os Contactos do Microsoft Outlook** que usa os contactos do utilizador como base para a criação de contextos de proximidade;
7. Uma **API REST** que permite a comunicação com os dispositivos móveis de forma a sincronizar as aplicações criadas;
8. Uma **framework para o runtime dos dispositivos móveis** o qual permitirá correr as aplicações desenvolvidas através da ferramenta de autor;
9. Uma implementação do **runtime para a plataforma Android** que serviu de base para validação e avaliação deste trabalho;
10. Um **conjunto de aplicações** perfeitamente funcionais que foram desenvolvidas com a plataforma e que serviram de base para testes e avaliação:
    a) Um guia turístico para a zona de Belém em Lisboa;
    b) Um guia turístico para Barcelona (obras de Gaudí em Barcelona);
    c) Um guia turístico para o distrito de Portalegre;
    d) Uma aplicação do tipo *peddy-paper* para o campus da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa;
    e) Uma aplicação na área vitívinicola para automatizar processos usados por técnicos agrícolas na visita a fornecedores de uva.
11. As seguintes **publicações em conferências internacionais** de grande impacto:

    Realinho, V., Dias, A.E. and Romão, T. (2011) Testing the Usability of a Platform for Rapid Development of Mobile Context-Aware Applications. *In Proceedings of Human-Computer Interaction - INTERACT 2011*, pp. 521-536 (Lisbon, Portugal, September 5-9, 2011), Springer Berlin/Heidelberg. DOI=10.1007/978-3-642-23765-2_36

    Realinho, V., Romão, T., Birra, F. and Dias, A.E. (2011) Building Mobile Context-aware Applications for Leisure and Entertainment. *In Proceedings of 8th International Conference on Advances in Computer Entertainment Technology - ACE 2011* (Lisbon, Portugal, November 8-11, 2011), ACM Press. DOI=10.1145/2071423.2071459

Realinho, V., Romão, T., Birra, F. and Dias, A.E. (2011) Rapid Development of Mobile Context-aware Applications with IVO. *In Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology - ACE 2011* (Lisbon, Portugal, November 8-11, 2011), ACM Press. DOI=10.1145/2071423.2071532

Graça, S., Oliveira, J.F. and **Realinho, V.** (2012) WorldPlus: An Augmented Reality Application with Georeferenced content for smartphones - the Android example. *In Proceedings of 20th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision - WSCG 2012* (Pilsen, Czech Republic, June 25-28, 2012), Journal of WSCG 2012.

Realinho, V., Romão, T. and Dias, A.E. (2012) An Event-Driven Workflow Framework to Develop Context-Aware Mobile Applications. *In Proceedings of Mobile and Ubiquitous Multimedia - MUM 2012* (Ulm, Germany, December 4-6, 2012), ACM Press. DOI= 10.1145/2406367.2406395.

Está ainda em preparação um artigo a submeter a uma revista.

1.3 Estrutura da Dissertação

Além do presente capítulo, este trabalho está organizado em mais três partes genéricas: fundamentos, desenvolvimento e a última relativa à avaliação e conclusões.

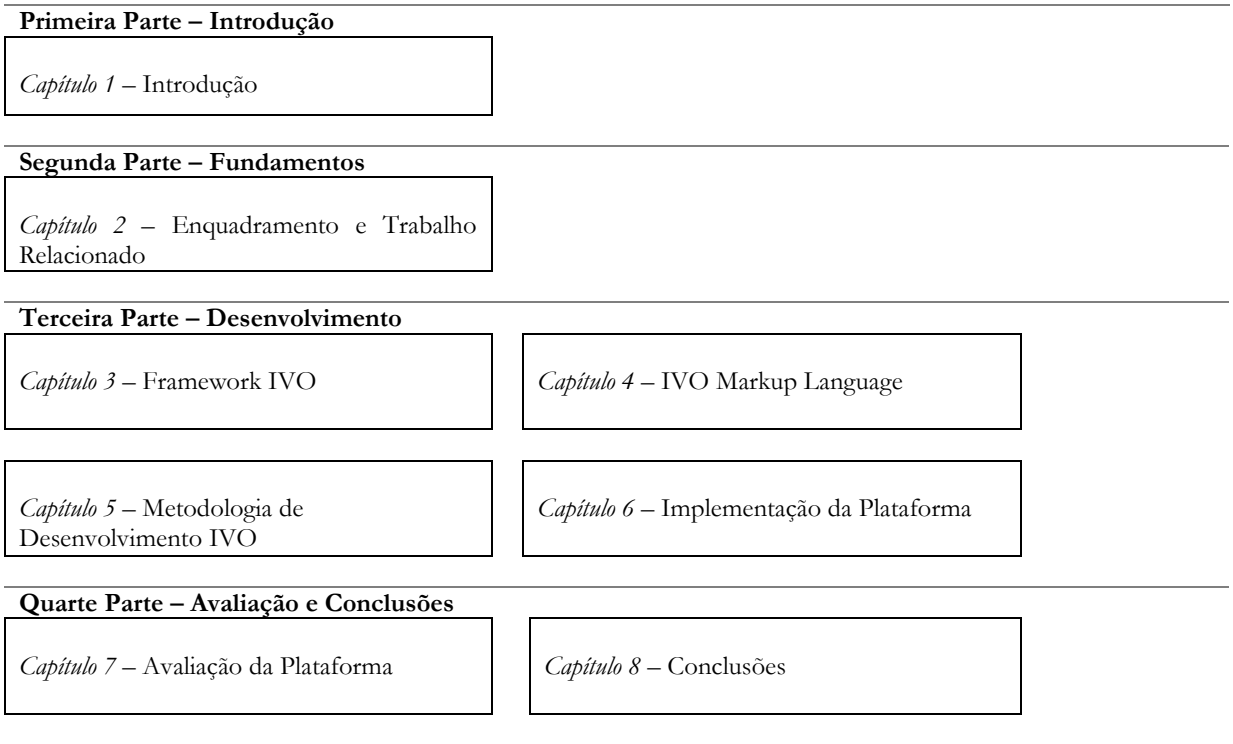


Figura 1.1. Estrutura da dissertação

A segunda parte, de fundamentos do problema, é composta pelo Capítulo 2 e nela se apresentam conceitos básicos directos ou indirectamente relacionados com a computação sensível ao contexto, bem como a análise de trabalhos relacionados.

A terceira parte consiste nos Capítulos 3, 4, 5 e 6 os quais têm como objectivo apresentar a plataforma para o desenvolvimento rápido de aplicações sensíveis ao contexto. No Capítulo 3 é descrita a *framework* IVO e no Capítulo 4 a linguagem de marcação de suporte à *framework*. O Capítulo 5 descreve a metodologia típica de desenvolvimento de uma aplicação através da plataforma IVO e o Capítulo 6 descreve a implementação da *framework*.

Finalmente, na última parte, composta pelo Capítulo 7 e pelo Capítulo 8, apresenta-se a avaliação da plataforma e a consolidação do trabalho, bem como algumas notas sobre trabalho futuro.

Capítulo 2

Enquadramento e Trabalho Relacionado

Conteúdo

Resumo

Este capítulo fornece uma pesquisa sobre os temas mais relevantes deste trabalho nomeadamente a questão da computação ubíqua, do contexto e da sensibilidade ao contexto incluindo as arquitecturas típicas das aplicações sensíveis ao contexto e os modelos de representação do mesmo. É ainda abordada a questão da sensibilidade ao contexto nos *workflows*, bem como o desenvolvimento pelo utilizador final.

2.1 Computação Ubíqua

A computação ubíqua é um paradigma de interacção humano-computador em que a tecnologia é integrada de forma transparente nos ambientes físicos para auxiliar as pessoas na realização das suas tarefas diárias de forma contínua e omnipresente [1]. Weiser, enquanto cientista chefe dos laboratórios da Xerox, realizou vários estudos para poder compreender como é que as pessoas trabalhavam e quais as ferramentas que usavam. Percebeu que a melhor utilização de uma ferramenta acontece quando o utilizador que a usa não percebe que a está a utilizar, isto é, a sua

utilização é transparente para o utilizador que dessa forma pode centrar a atenção no trabalho que está a executar. O termo ubíquo é assim usado, para exprimir a ideia de que os computadores e a computação estão presentes em qualquer lugar embutidos no ambiente que nos rodeia. Por outras palavras, o objectivo da computação ubíqua é mover os computadores do foco central de atenção dos utilizadores para um mundo invisível, onde são usados subconscientemente, para aumentar a eficiência das ferramentas e meios de comunicação existentes.

> *"The most profound technologies are those that disappear: they weave themselves into fabric of everyday life until are indistinguishable from it."*
>
> *Mark Weiser*

Na sua essência, este paradigma, por vezes confundido com computação pervasiva, ambiente inteligente, ou mais recentemente por *everyware* [3, 18], utiliza dispositivos pequenos, baratos e robustos com capacidade de computação e ligação em rede perfeitamente integrados no ambiente, envolvendo esta heterogeneidade de dispositivos um elevado grau de interoperabilidade. Castells no livro *The Rise of the Network Society* [19] prevê a existência de um mundo povoado de dispositivos minúsculos que se interligam "como os pigmentos da pintura de um quadro" num ambiente de computação ubíqua. Estes dispositivos permitirão tornar a execução das actividades mais rápida e fácil tornando-as proactivas. O utilizador deixa assim de ser parte do sistema (pois é reduzida a interacção entre este e o utilizador) e passa a ser o supervisor do sistema, actuando apenas quando for estritamente necessário [5].

Da visão de Weiser podem-se deduzir três requisitos fundamentais para a computação ubíqua: (i) os computadores precisam estar ligados em rede, distribuídos e acessíveis de forma transparente; (ii) a interacção humano-computador deve tornar-se invisível; (iii) a computação precisa ser sensível ao contexto, de forma a optimizar o seu funcionamento no meio ambiente. Poslad [20] acrescenta a estes requisitos mais dois: (iv) os computadores podem operar de forma autónoma, sem intervenção humana, ser auto-regulados, em contraste com a interacção tradicional humano-computador; (v) os computadores podem lidar com uma multiplicidade de acções e interacções dinâmicas, governados por decisões e por uma interacção organizacional inteligente.

2.2 Contexto e Sensibilidade ao Contexto

Como referido anteriormente, a sensibilidade ao contexto representa um tema importante de investigação abordado em computação ubíqua, em que as aplicações se adaptam dinamicamente às mudanças nas actividades do utilizador e do ambiente [9]. O contexto é uma situação particular

do utilizador, envolvendo a computação sensível ao contexto a detecção dessas situações e o fornecimento de informação e serviços adequados ao utilizador.

2.2.1 Definições

A importância de uma definição clara do contexto dos utilizadores e dos dispositivos, é determinante para que as aplicações sensíveis ao contexto possam atingir o objectivo de fornecer informação e serviços contextualizados ao utilizador. O próprio conceito de contexto, apesar de muito estudado, continua a não obter um consenso entre os vários autores.

A expressão "sensível ao contexto" (*context-aware*) foi usada pela primeira vez em 1993 por Schilit *et al.* [21] para se referir à localização, identidade de pessoas ou objectos na proximidade e ainda alterações nesses objectos. Estes autores definem três aspectos de contexto considerados essenciais: onde se está, com quem se está e quais os recursos disponíveis na proximidade.

De igual forma, Brown *et al.* [22] definem o contexto como a localização do utilizador, identificação das pessoas ao seu redor, a hora do dia, a estação do ano, a temperatura, entre outros.

Ryan *et al.* [23] definem contexto como sendo a localização do utilizador, o ambiente, a identidade e o tempo.

Dey [24] propõe uma das definições de contexto mais usualmente aceites:

> *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves (…) where relevancy depends on the user's task."*

Reconhecendo o facto de esta definição ser bastante ampla e de oferecer aos programadores pouca orientação sobre o que qualificar como contexto, muitos autores têm tentado definir o contexto relevante e formas de representação do contexto em áreas de aplicação específicas [25–27], sem que contudo, nenhum deles tenha sido amplamente utilizado.

A sensibilidade ao contexto procura explorar as interacções pessoa-máquina, fornecendo dispositivos de interacção com o conhecimento do ambiente do utilizador, podendo ser usada para diminuir a quantidade de entrada explícita que o utilizador é obrigado a fornecer ao sistema.

Uma das definições mais aceites para sistema sensível ao contexto é a proposta por Dey [28]:

> *"A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."*

Neste trabalho, usamos como referencial as definições de Dey para contexto [24] e para sensibilidade ao contexto [28], por serem suficientemente abrangentes, mas igualmente precisas para sustentar os objectivos do trabalho. São usados como base neste trabalho, os contextos de localização, de tempo, de proximidade e de estado. O contexto de estado refere-se ao estado do dispositivo móvel e permite a definição de uma grande variedade de situações ao possibilitar o estabelecimento de regras representadas através de expressões booleanas baseadas num conjunto de variáveis que definem o estado do dispositivo. Além disso, é possível expandir a capacidade de sensoriamento através da ligação de sensores externos ao dispositivo móvel permitindo enriquecer a percepção do ambiente.

2.2.2 Classificações do Contexto

De forma a facilitar a identificação do que se deve considerar contexto, vários autores propõem classificações para o mesmo. As classificações mais relevantes separam a informação de contexto segundo o critério das dimensões dos seus elementos, outras distinguem o contexto quanto ao nível da granularidade da informação ou à periodicidade de actualização, e outras ainda fazem uso da relevância em relação ao foco de atenção do utilizador.

Apesar da localização e a identidade serem um elemento comum nas definições anteriores, uma completa definição de contexto deve incluir também o "quando" que identifica o contexto temporal (por exemplo a data corrente, a estação do ano, entre outros) e o "quê" que identifica o que o utilizador está a realizar [29]. Dey [28] enumera exemplos de contexto acrescentando novas dimensões, como o estado emocional do utilizador e o seu foco de atenção, para além das dimensões habituais da localização e orientação, data e hora, objectos e pessoas no ambiente envolvente.

Na literatura é comum ser referenciado ainda como contexto, o "porquê" que representa a motivação por trás das acções, e o "meio" que identifica como a informação de contexto é recolhida. O contexto formado por estas seis dimensões é conhecido como 5W+1H que correspondem às iniciais dos termos em inglês *who*, *where*, *when*, *what*, *why* e *how* [30, 31]. Estas dimensões são consideradas básicas para contextualizar uma situação e englobam, de forma geral, os atributos mais usuais de contexto. A tabela seguinte, adaptada de Morse *et al.* [30] e de Truong *et al.* [31], resume estas dimensões do contexto.

Dimensão	Descrição
Who (identidade)	Informação sobre as entidades envolvidas numa determinada tarefa, como por exemplo os utilizadores e objectos físicos ou computacionais.
Where (localização)	Um dos elementos de contexto mais utilizados, podendo conter dados como a longitude, a latitude e a altitude ou então informação simbólica como a rua, cidade ou país.
What (actividade)	Corresponde ao que está a ser realizado pelo utilizador. Pode ser uma tarefa complexa adquirir esta informação, principalmente em sistemas com actividades muito diversificadas.
When (tempo)	Corresponde ao contexto temporal como, por exemplo, a data corrente, a estação do ano, ou a duração de uma tarefa desempenhada pelo utilizador.
Why (motivação)	Corresponde à motivação do utilizador ao executar determinada actividade. A obtenção da informação que possa caracterizar o raciocínio de uma pessoa é um dos maiores desafios da computação sensível ao contexto, sendo geralmente obtida a partir da combinação de informação diversa de contexto através de inferências sobre essa mesma informação.
How (meio)	Define a forma como a informação de contexto é obtida, como por exemplo dispositivos de GPS para a aquisição do contexto de localização, o relógio do sistema para a obtenção de contexto temporal, entre muitos outros.

Tabela 2.1. Dimensões de contexto 5W+1H (Adaptado de [30, 31])

Schilit *et al.* [32] dividem o contexto em três categorias, com o objectivo de ajudar os *designers* de aplicações a identificar os componentes mais prováveis de contexto:

1. **Contexto Computacional**: especificações de dispositivo, conectividade à rede ou recursos disponíveis;
2. **Contexto do Utilizador**: perfil do utilizador, localização, tarefa do utilizador, entre outras;
3. **Contexto Físico**: condições visuais e sonoras, condições climatéricas e ambiente envolvente.

Chen e Kotz [33] acrescentam a estas categorias mais duas: o **Contexto Temporal**, como por exemplo a data e hora ou a estação do ano, e o **Contexto Histórico** que representa a evolução histórica de determinada informação ou actividade, como exemplo o registo histórico de pesquisas efectuadas pelo utilizador.

Wang *et al.* [34] propõem uma classificação de contexto baseada na sua granularidade a que chamaram de **Contexto Básico** e **Contexto Complexo**. O contexto básico refere-se a elementos de granularidade mais baixa que podem ser obtidos de forma automática através da leitura de sensores (localização, tempo, identidade, entre outros). O contexto complexo é formado a partir da composição ou inferência sobre o contexto básico e pode representar, por exemplo,

actividades do utilizador (exemplos: a falar, a ler ou a caminhar), situações sociais (com quem o utilizador está), actividades sociais (exemplos: em reunião ou no cinema), entre outras.

O contexto pode ser identificado quanto à sua periodicidade de actualização e classificado como **Estático** e **Dinâmico** [35]. O contexto estático corresponde a informação de contexto, em geral fixa que não muda com frequência, como por exemplo os dados pessoais do utilizador ou a localização de pontos de interesse de um local. O contexto dinâmico refere-se a informação de contexto que pode mudar de instante para instante e que necessita ser constantemente monitorizado e actualizado, como por exemplo a localização física de um indivíduo ou o que este se encontra a fazer.

Brézillon e Pomerol [36] propõem uma classificação de contexto baseada na relevância deste em relação ao foco de atenção do utilizador. A ideia subjacente é a de que o contexto não deve ser visto de forma isolada, mas sempre relativamente ao foco de atenção do utilizador o que determina o que pode ser considerado relevante em determinado momento. Estes autores propuseram uma classificação do contexto composta por três partes distintas: **Conhecimento Contextual**, **Conhecimento Externo** e **Contexto Procedimental**. O contexto é visto como um espaço de conhecimento partilhado que é explorado e aproveitado pelos participantes numa determinada interacção.

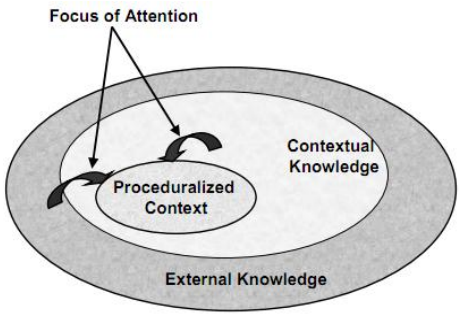


Figura 2.1. Classificação do Contexto [36]

O conhecimento externo (*External Knowledge*) representa a parte do contexto que não tem qualquer relevância com o foco, que embora possa ser considerado informação de contexto, não é relevante para a tarefa que o utilizador está a realizar no momento. O conhecimento contextual (*Contextual Knowledge*) representa o contexto que é relevante e que tem uma forte relação com o foco de atenção do utilizador. Este tipo de contexto pode-se tornar em conhecimento externo quando o foco do utilizador for diferente e portanto sem relevância. Finalmente, o contexto procedimental (*Proceduralized Context*) é o subconjunto do conhecimento contextual que é invocado, organizado, estruturado e situado de acordo com o foco e que, portanto, é a parte do contexto realmente utilizada para apoiar a tarefa em execução naquele foco. Este tipo de contexto

é construído a partir do conhecimento contextual existente e é o que será realmente utilizado para fornecer a informação e/ou serviços sensíveis ao contexto.

2.2.3 Aplicações Sensíveis ao Contexto

Schilit *et al.* [21] fornecem a primeira definição de aplicação sensível ao contexto, como sendo *"um software com capacidade de se adaptar de acordo com a localização, a proximidade de pessoas e de objectos, bem como as alterações a esses objectos ao longo do tempo"*. Desde então, muitas foram as definições que surgiram, relacionadas quase sempre com a questão da adaptação, reactividade, capacidade de resposta e sensibilidade.

Pascoe [37] define computação sensível ao contexto como sendo a capacidade dos dispositivos de computação para detectar, interpretar e responder à localização do utilizador e dos próprios dispositivos de computação.

Dey e Abowd [29] afirmam ser necessária uma definição mais genérica, que não esteja vinculada a uma característica específica como a adaptação, reactividade, capacidade de resposta ou sensibilidade. A definição de Dey e Abowd estabelece que um sistema é sensível ao contexto, se usar o contexto para fornecer informação e/ou serviços relevantes ao utilizador, onde a relevância depende da tarefa que o utilizador se encontra a realizar. De acordo com esta definição, numa aplicação sensível ao contexto, o único requisito é a capacidade de resposta ao contexto não sendo a interpretação, a detecção e a adaptação requisitos obrigatórios. Esta definição faz sentido, por exemplo, em aplicações que simplesmente reflectem o contexto ao utilizador ou as que não detectam ou percebem o contexto sendo a detecção e a interpretação realizada por outras entidades computacionais.

2.2.4 Arquitecturas Típicas

Os sistemas sensíveis ao contexto podem ser implementados de acordo com várias abordagens de arquitectura. A abordagem mais adequada depende sempre dos requisitos e condições especiais do problema em causa, tais como a localização dos sensores (local ou remota), a quantidade de utilizadores (um utilizador ou muitos), os recursos disponíveis nos dispositivos utilizados ou a capacidade de instalar extensões ao sistema tornando-o mais flexível e escalável.

O método de aquisição de dados de contexto é um factor limitativo do tipo de arquitectura do sistema. O rápido avanço na tecnologia dos sensores tem motivado o surgimento de novos projectos na área da sensibilidade ao contexto, tornando possível o sensoriamento de informação de contexto que anteriormente não podia ser adquirida. Chen [38] apresenta três abordagens de arquitectura as quais têm por base a forma de aquisição da informação de contexto:

1. **Acesso directo ao hardware dos sensores**. Esta abordagem é usada em dispositivos com sensores próprios. Muitos dos projectos pioneiros de sistemas sensíveis ao contexto usaram este método de aquisição de contexto [39–42].

   Uma das principais vantagens desta abordagem é que as aplicações têm grande controlo sobre as operações de leitura dos sensores e podem, por conseguinte, ter um melhor conhecimento da forma como os dados são lidos e calculados.

   Esta abordagem apresenta a desvantagem de não ser adequada para ambientes distribuídos de sensoriamento, pois devido à sua natureza de acesso directo, falta-lhe um componente capaz de gerir acessos simultâneos aos sensores o que torna difícil a sua manutenção [24].

2. **Infra-estutura de middleware.** Esta abordagem permite resolver o problema identicado na abordagem anterior, na medida em que introduz uma arquitectura em camadas com a intenção de esconder os detalhes de baixo nível da detecção. A utilização desta abordagem permite que as aplicações se foquem na utilização do contexto em vez de na sua aquisição. Esta técnica facilita a extensibilidade já que o código do cliente não necessita ser modificado, simplificando a reutilização do mesmo.

   O projecto *Odyssey* [43], utiliza agentes sensíveis ao contexto construídos sobre uma infra-estrutura de *middleware* para adquirir o estado da rede de comunicação. O *Context Toolkit* [24] é uma infra-estrutura de *middleware* para suportar a aquisição de contexto. Ao contrário da infra-estrutura de *middleware* do *Odyssey*, o *Context Toolkit* visa a criação de uma solução geral para a construção de módulos de aquisição de contexto escalável e reutilizável. Este projecto usa o conceito de *widget* do *design* de interfaces gráficas, em que cada *widget* é responsável pela aquisição do tipo de contexto para que foi construído.

   Um problema com a abordagem de *middleware* é que ela obriga a um maior poder computacional nos dispositivos onde se encontra o *middleware* a correr, o que poderá ser um problema em dispositivos que possuem menos recursos como por exemplo, telemóveis ou nos chamados *embedded devices*.

3. **Servidor de contexto.** Esta abordagem estende a arquitectura baseada em *middleware* através da introdução de uma gestão centralizada no acesso aos sensores. A aquisição de dados dos sensores é feita através do servidor de contexto para facilitar acessos múltiplos simultâneos. Os projectos *Me-Centric Domain Server* [44] e CoBrA [38] são dois exemplos que usam a abordagem de servidores de contexto. Além da reutilização de sensores, o uso

de um servidor de contexto tem a vantagem de aliviar os clientes de operações de recursos intensivos. Este aspecto é de extrema importânica principalmente em sistemas que usem dispositivos com limitações no poder de computação, espaço de armazenamento, entre outros. Por outro lado, quando se projecta um sistema baseado neste tipo de arquitectura, esta abordagem obriga a ter em consideração os protocolos apropriados, o desempenho da rede, parâmetros de qualidade de serviço, etc.

Winograd [45] estabelece uma classificação dividindo as arquitecturas de aquisição de contexto em (i) *widgets*, (ii) *networked services* e (iii) *blackboard model*. A arquitectura das *widgets* do *Context Toolkit* [24] é uma arquitectura de *middleware* que fornece uma interface pública para os sensores. As arquitecturas *networked services* representam uma abordagem mais flexível e assemelham-se às arquitecturas de servidor de contexto tendo sido utilizadas por Hong e Landay [46]. Em vez de um servidor global, são usadas técnicas de descoberta para encontrar serviços de rede. A arquitectura *blackboard model* representa uma visão centralizada dos dados. Nesta abordagem as aplicações enviam mensagens para uma zona partilhada, o *blackboard*, e registam-se nele de forma a serem notificadas quando algum dos eventos especificados ocorrer. As vantagens deste modelo são a simplicidade de adição de novas fontes de contexto e a fácil configuração.

Indulska e Sutton [47] apresentam uma classificação dos sensores baseada na forma como os dados são capturados, classificando-os em sensores físicos, virtuais e lógicos. Os sensores físicos representam a classe mais utilizada, existindo hoje em dia muitos tipos de sensores capazes de capturar a maior parte de dados físicos. A Tabela 2.2 apresenta alguns exemplos de sensores físicos para vários tipos de contexto.

Tipo de Contexto	Exemplo de Sensores
Luz	Fotodíodos, sensores de cor, sensores de infra-vermelhos e ultra violetas
Contexto Visual	Câmaras
Audio	Microfones
Movimento, Aceleração	Interruptores de mercúrio, sensores angulares, acelerómetros, detectores de movimento, campos magnéticos
Localização	Outdoor: Global Positioning System (GPS), Global System for Mobile Comunications (GSM); Indoor: Active Badge system
Toque	Sensores de toque implementados em dispositivos móveis
Atributos físicos	Biosensores para medir a resistência da pele, pressão arterial

Tabela 2.2. Exemplos de sensores. (Adaptado de [48])

Os sensores virtuais utilizam fontes de dados de contexto de aplicações ou serviços. Por exemplo, é possível determinar a localização de uma pessoa, não só por intermédio de sistemas

de localização (sensores físicos), mas também por um sensor virtual, utilizando dados provenientes do *browser*, de uma agenda electrónica, um sistema de reserva de viagens, *emails*, entre outros. Outros atributos de contexto que podem ser detectados por sensores virtuais incluem, por exemplo, a actividade do utilizador, através da verificação do movimento do rato e da actividade do teclado.

Os sensores lógicos utilizam várias fontes de informação, e combinam sensores físicos e virtuais com informação adicional proveniente de bases de dados ou outras fontes, a fim de determinar contextos mais complexos. Por exemplo, um sensor lógico pode ser construído para detectar a localização actual de uma pessoa através da análise dos seus *logins* nos computadores e de uma base de dados que faça o mapeamento dos dispositivos na sua localização física.

2.2.5 Requisitos no Desenvolvimento de Aplicações Sensíveis ao Contexto

Dey [24] apresenta um conjunto de requisitos bastante comuns e discutidos na literatura [49, 50] que visam orientar o desenvolvimento de aplicações sensíveis ao contexto. Estes requisitos possuem graus de relevância diferentes em função do domínio da aplicação que se está a desenvolver. De seguida são resumidos os requisitos definidos por Dey, os quais foram tidos em conta na análise de requisitos deste trabalho detalhada na Secção 3.5.

2.2.5.1 Especificação do Contexto

Este é considerado o requisito mais importante no desenvolvimento de aplicações sensíveis ao contexto, e consiste num mecanismo que permite aos programadores especificar o contexto que a aplicação necessita. Este mecanismo deverá garantir que, quando as aplicações receberem o contexto que tenham solicitado, ele será útil para a aplicação e, idealmente, não deverá exigir interpretação ou análise adicional.

O mecanismo e a linguagem de especificação do contexto devem permitir aos programadores especificar o contexto de acordo com várias dimensões:

Contexto simples e Contexto composto. Um contexto simples pode ser a localização de um utilizador. Falamos de contexto composto se incluirmos, por exemplo para além da localização o tempo livre que o utilizador tem antes do próximo compromisso. O contexto composto traduz-se na prática, na avaliação de expressões booleanas que podem ser mais ou menos complexas.

Contextos não relacionados e Contextos relacionados. Contextos relacionados são contextos relativos à mesma entidade. Por exemplo, o contexto composto referido no ponto anterior

corresponde a um contexto relacionado relativo a uma mesma entidade – o utilizador. Contextos não relacionados podem ser a localização de um utilizador e a cotação de determinada acção.

Contexto não filtrado e Contexto filtrado. Por exemplo, um pedido de notificação não filtrado sobre a mudança de localização de um utilizador resultaria na notificação sempre que há qualquer alteração na localização deste. Em contrapartida, um pedido filtrado permitiria por exemplo, que a notificação fosse gerada apenas quando o utilizador se mudasse para um prédio diferente.

Contexto não interpretado e Contexto interpretado. Por exemplo, o GPS devolve a localização como um par composto pela latitude e pela longitude. Se a aplicação estiver interessada no nome da rua, deverá existir a possibilidade de converter este par no nome de rua sendo esta a informação devolvida à aplicação, resultando desta forma num contexto interpretado.

2.2.5.2 Separar a Aquisição do Tratamento do Contexto

Não existe uma forma *standard* para a aquisição e tratamento do contexto. Em geral os programadores escolhem a técnica mais fácil de implementar, em detrimento da generalização e reutilização. Alguns dos métodos utilizados e estudados na Secção 2.2.4 resultam em aplicações difíceis de construir e evoluir.

A separação entre a aquisição e a forma como o contexto é utilizado, permite às aplicações utilizar o contexto sem se preocuparem com os detalhes de um sensor e de como adquirir o contexto a partir dele. É assim desejável um fraco acoplamento entre as aplicações e os mecanismos de aquisição, sendo a complexidade do tratamento do contexto abstraída para as aplicações.

2.2.5.3 Interpretação do Contexto

A interpretação do contexto compreende um conjunto de métodos e processos que viabilizam a manipulação, a agregação, o raciocínio, a derivação, e a predição de tendências sobre o contexto adquirido. O objectivo é produzir informação mais refinada e relevante de modo a melhorar o entendimento de um determinado contexto pelas aplicações e auxiliá-las na tomada de decisões. Desta forma, é possível obter contexto de alto nível que será posteriormente explorado pelas aplicações para diversos fins.

A combinação dos mecanismos de interpretação com os de especificação permite que as aplicações possam requisitar determinado contexto e adquiri-lo na sua forma interpretada sempre que necessário.

2.2.5.4 Comunicação Distribuída e Transparente

Por natureza, os sistemas sensíveis ao contexto são modulares e distribuídos, devendo a comunicação entre os seus componentes ser tratada de forma apropriada. Devem ser utilizados protocolos de comunicação e *standards* para integração de componentes, devendo ainda ser tidas em consideração, questões relacionadas com a segurança, privacidade e escalabilidade.

2.2.5.5 Aquisição Contínua do Contexto

É necessário que os componentes que capturam o contexto sejam executados continuamente, pois uma aplicação pode requerer informação a qualquer momento. Num cenário de computação sensível ao contexto, as aplicações devem assumir que o contexto estará disponível a qualquer hora e lugar, independentemente da localização das entidades envolvidas.

2.2.5.6 Armazenamento do Contexto

A necessidade de manter um histórico de contexto é um requisito ligado à aquisição de contexto, bem como à disponibilidade contínua dos componentes de captura do contexto. Este histórico pode ser utilizado para estabelecer tendências e prever valores futuros de contexto. Sem o armazenamento do contexto, esse tipo de análise não é possível ser realizado.

2.2.5.7 Descoberta de Recursos

O mecanismo de descoberta de recursos informa onde encontrar os componentes de captura de contexto, bem como a descrição dos mecanismos de acesso a eles. Forstadius *et al.* [51] apontam ainda o desafio de se localizar o componente mais apropriado para uma dada situação em que os utilizadores se encontrem.

2.3 Representação do Contexto

Uma boa abordagem para representação e recuperação do contexto é um factor chave em sistemas sensíveis ao contexto [52]. Um modelo de representação formal de contexto deve levar em consideração factores como a interoperabilidade, a extensibilidade, a partilha e a sua reutilização, de forma a permitir (i) promover a partilha de informação contextual (ii) facilitar consultas complexas envolvendo informação de múltiplos contextos e ainda (iii) desassociar os detalhes da gestão do contexto da implementação da aplicação em si. Strang e Linnhoff-Popien resumem as abordagens mais relevantes para modelar o contexto de acordo com as estruturas de dados utilizadas para o representar e para o partilhar (Tabela 2.3).

Modelo	
Par Chave-Valor	+ simples
Linguagem de Marcação	
Gráfico (UML/ORM)	
Orientado a Objectos	
Lógica	
Ontologias	+ formal

Tabela 2.3. Abordagens de representação de contexto [52]

2.3.1 Par Chave-Valor

O modelo chave-valor é o mais simples para modelar informação de contexto. Este modelo não considera qualquer tipo de hierarquia entre os atributos. Sendo um modelo bastante simples de utilizar e manipular, ele não permite estruturações mais sofisticadas que possibilitem algoritmos eficientes de recuperação de contexto. Para identificar o valor de um atributo de contexto, usa-se uma comparação de caracteres(*string matching*) com um resultado exacto de comparação (*exact matching*) [52].

Este modelo, apesar de não ser muito formal, é frequentemente utilizado em *frameworks* de serviços distribuídos em que os serviços de contexto são descritos através de uma lista de atributos simples.

Schilit *et al.* [32] utilizaram variáveis de ambiente para armazenar informação contextual sob a forma chave-valor. Outros exemplos que usam este modelo são o *Context Toolkit* [12] e o *Context Management System* [53].

2.3.2 Linguagem de Marcação

Este modelo utiliza a *eXtensible Markup Language*[1] (XML) para modelação do contexto. São utilizadas estruturas de dados hierárquicas que consistem em rótulos de marcação ou *tags* com atributos e conteúdo. Uma vez que o XML é um *standard* largamente utilizado, a partilha de informação entre sistemas heterogéneos fica facilitada.

Esta abordagem é bastante utilizada para a modelação de perfis de utilizador e de dispositivos. Um exemplo de sistema que usa este modelo é o *Comprehensive Structured Context Profiles* (CSCP) [54]. O CSCP é um modelo desenvolvido através da linguagem *Resource Description Framework*[2]

[1]http://www.w3.org/XML/

[2]http://www.w3.org/RDF/

(RDF) que representa o contexto como perfis de sessão. Os nomes dos atributos são interpretados de acordo com a sua posição na estrutura do perfil. Esses atributos incluem, por exemplo, a localização, as características da rede, requisitos da aplicação e informação sobre a sessão. Esta técnica procura definir um vocabulário comum e extensível para representação do contexto.

Em Strang e Linnhoff-Popien [52] podem ser encontrados vários exemplos de utilização deste modelo.

2.3.3 Gráfico

A *Unified Modeling Language*[3] (UML) é uma linguagem de modelação bastante utilizada com uma forte componente gráfica de diagramas, que por ser genérica também se presta para a modelação de contexto. Bauer propõe uma extensão para representação de contextos através da UML [55]. Esta proposta mapeia a utilização dos diagramas UML na representação de contexto como a seguir se descreve.

Diagrama	**Utilização**
Diagramas de Caso de Uso	Modela os actores envolvidos na interacção e as respectivas influências
Diagramas de Componentes	Modela os sistemas envolvidos que contêm informação de contexto
Diagramas de Classe	Representa a informação estrutural do domínio
Diagramas de Sequência	Contém os diferentes cenários de activação de contextos

Tabela 2.4. Extensão da UML para modelação do contexto [55]

Henricksen *et al.* [56] propõem um modelo gráfico que usa uma extensão de contexto ao *Object Role Modeling*[4] (ORM) [57]. No ORM, o conceito básico de modelação é o facto, e a modelação de um domínio utilizando ORM envolve a identificação dos tipos de factos adequados e os papéis que as entidades desempenham nestes. Henricksen estende o ORM para permitir a categorização de vários tipos de factos, de acordo com a sua persistência e origem, quer sejam estáticos (factos que permanecem inalteradas, enquanto as entidades que descrevem persistirem) quer sejam dinâmicos. Outro indicador de qualidade criado por Henricksen é um tipo de facto histórico para cobrir o aspecto temporal do contexto. A última extensão ao ORM feita por Henricksen para fins de modelação de contexto é a dependência de factos, que representam um tipo especial de relação entre os factos, onde uma mudança num facto leva automaticamente a uma mudança de outro facto (relação "depende de"). A Figura 2.2 ilustra um exemplo da notação de Henricksen.

[3]http://www.uml.org/

[4]http://www.orm.net/

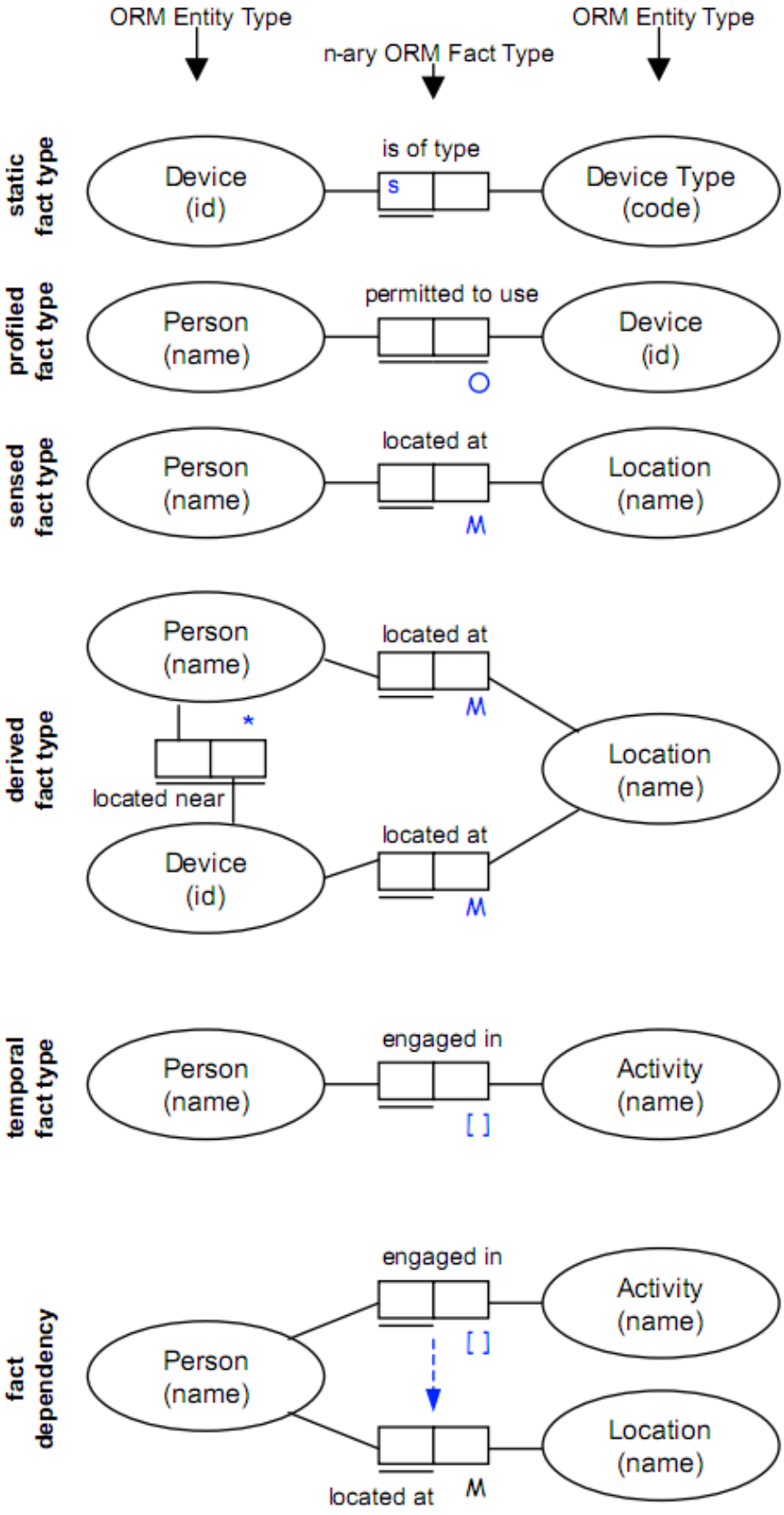


Figura 2.2. Exemplo de extensão de contexto ao ORM [56]

2.3.4 Orientado a Objectos

A abordagem de modelação de contexto usando o paradigma da orientação a objectos visa empregar propriedades como encapsulamento, herança e reutilização, para lidar com os problemas da dinâmica do contexto [52]. O detalhe do processamento do contexto é encapsulado ao nível do objecto de contexto e o acesso à informação de contexto é realizado por meio de interfaces. Neste modelo, a informação contextual é definida através de classes, usando os recursos de herança para identificar informação contextual derivada. Cada classe possui as propriedades, funções e regras associadas à informação contextual que ela representa, as quais são reutilizadas nas subclasses. A abordagem orientada a objectos permite organizar melhor o conhecimento e simplificar a criação de regras de manipulação do contexto. A herança e reutilização simplificam a representação do conhecimento e a definição de propriedades e regras em sistemas e domínios complexos.

Esta abordagem foi utilizada por Bouzy e Cazenane numa versão sensível ao contexto do jogo Go [58]. Neste modelo, eles destacam os contextos temporais (se o jogo está no início, no meio ou no fim), o contexto dos objectivos (para identificar bons movimentos visando atingir algum objectivo), contextos espaciais (ambiente espacial de um grupo de pedras) e contextos globais (por exemplo bloquear o jogo ou a natureza da posição da pedra). O contexto temporal é representado como uma classe mais geral chamada *Stage* e três subclasses denominadas *Beginning*, *Middle* e *EndGame*.

Outro exemplo desta abordagem é o *Active Object Model* do projecto GUIDE [59]. A abordagem escolhida derivou principalmente do requisito do sistema ser capaz de gerir uma grande variedade de informação de contextos pessoais e ambientais, mantendo a escalabilidade. Todos os detalhes da recolha de dados e da fusão são encapsulados dentro dos objectos activos e, portanto, ocultos para os outros componentes do sistema.

2.3.5 Baseado em Lógica

Os modelos baseados em lógica têm um alto grau de formalidade. Normalmente, os factos, as expressões e as regras são usados para definir um modelo de contexto. Um sistema baseado em lógica é então utilizado para gerir os termos acima referidos e permite adicionar, actualizar ou remover novos factos. O processo de inferência (também chamado de raciocínio) pode ser utilizado para obter novos factos com base nas regras existentes nos sistemas. A informação contextual deve ser representada de maneira formal como factos. Uma das primeiras abordagens foi publicada por McCarthy e Buvac [60].

2.3.6 Baseado em Ontologias

Uma ontologia fornece definições de conceitos básicos de um domínio, apropriadas para o processamento automático. As ontologias são utilizadas para realizar inferências sobre os objectos de um domínio, sendo uma técnica bastante promissora para a modelação de informação de contexto, devido à sua elevada formalidade e à possibilidade de aplicação de técnicas de raciocínio. Uma representação explícita de uma ontologia consiste num modelo estruturado de vocabulários (classes e propriedades) e na semântica associada (relações entre as diferentes classes e propriedades), bem como funções, axiomas e instâncias [34].

O surgimento de linguagens para representar ontologias, das quais se destaca a OWL (*Web Ontology Language*) [61], vieram facilitar a reutilização de conhecimento entre os sistemas, e permitir o uso de motores de inferência existentes.

Várias são as propostas de utilização de ontologias de contexto entre as quais a CoBrA-Ont para ambientes ubíquos [62] e a CONON, para casas inteligentes [34]. A linguagem CoOL (*Context Ontology Language*) [63] apoia a criação de ontologias de contexto, baseada no modelo *Aspect-Scale-Context* (ASC), onde aspectos representam classificações (por exemplo distância espacial) e podem possuir várias escalas (por exemplo quilómetro ou metro), que são dimensões individuais dos aspectos, e expressam alguma informação contextual.

2.3.7 Avaliação dos Modelos

Strang e Linnhoff-Popien [52] fizeram uma avaliação dos modelos atrás descritos e concluíram que as ontologias constituem a abordagem mais promissora, devido às suas características de permitir a partilha de conhecimento entre pessoas e entre agentes de *software*, permitir a reutilização de conhecimento entre aplicações, bem como de permitir o uso de motores de inferência existentes para inferir contextos complexos. Contudo, para outros autores, como Henricksen *et al.* [64], a utilização de ontologias na representação de contexto, apresenta actualmente alguns problemas como, por exemplo, o facto das ontologias e os seus motores de inferência serem ainda imaturos e limitados; o OWL ainda não fornecer suporte directo para regras axiomáticas, o que limita os tipos de raciocínio; as ontologias não suportarem adequadamente raciocínio sobre informação de contexto imperfeita (imprecisas, ambíguas, incompletas); e o processo de criar ou estender ontologias de contexto ser complexo e sujeito a falhas, uma vez que as linguagens de escrita das ontologias não são intuitivas.

Santos [65] apresenta um resumo das vantagens e desvantagens da utilização dos modelos anteriores, bem como dos métodos utilizados para processamento e recuperação dos contextos.

Modelo	Vantagens	Desvantagens	Processamento e Recuperação
Chave-valor	Estrutura simples, de fácil implementação e uso.	Não considera hierarquia.Inadequado para aplicações com estruturas complexas.	Busca linear com casamento exacto de nomes.
Linguagem de Marcação	Baseado em XML. Prevê hierarquia. Esquema de marcação implementa o próprio modelo. Utilização típica em perfis.	Incompletude e ambiguidade na informação devem ser tratadas pelo sistema. Inadequado para representar estruturas complexas.	Linguagem de consulta baseada em marcação.
Gráfico	Facilita o desenho, especificação e a definição dos requisitos de contexto.	Não permite o processamento. Requer estruturas de dados adicionais.	Podem ser transformados para XML e utilizar as ferramentas de processamento de XML.
Orientado a Objectos	Encapsulamento, herança, reusabilidade. Permite abstrair o tratamento do contexto.	Invisibilidade no tratamento do contexto, pelo encapsulamento, dificulta formalismo do modelo.	Algoritmos e compiladores.
Baseado em Lógica	Alto grau de formalidade. Informação contextual representada como factos. Podem-se obter novos factos com base nos existentes (raciocínio). Viabiliza a formalização compreensão e partilha por humanos e computadores.	Potencialmente ineficiente. Não é flexível e não se adapta a circunstâncias que mudam frequentemente.	Motor de Inferência.
Ontologias	Contextos modelados como conceitos e factos. Viabiliza a formalização, compreensão e partilha por humanos e computadores.	Tecnologia de manipulação imatura.	Motor de Inferência, Linguagem de consulta baseada em OWL.

Tabela 2.5. Resumo dos Modelos de Representação de Contexto (Adaptado de [65])

Cada modelo possui vantagens e desvantagens, não existindo um modelo que possa ser considerado unanimamente o ideal para todos os tipos de aplicações sensíveis ao contexto. Os dados do contexto poderão ser obtidos de diferentes formas a partir de várias fontes heterogéneas o que dificulta a sua partilha, processamento e gestão do contexto em geral. Power [66] afirma mesmo não ser viável pensar-se na representação de contexto baseada num único modelo sugerindo que uma abordagem híbrida poderá ser a mais apropriada.

2.4 Workflow Sensível ao Contexto

O *workflow* sensível ao contexto tornou-se um tópico de investigação importante como forma de introduzir a sensibilidade ao contexto e a mobilidade nos *workflows*. O trabalho pioneiro de Jing *et al.* [67], Chakraborty e Lei [68], Cho *et al.* [69] e Jang *et al.* [70], beneficiaram da investigação anterior na área da gestão de processos. Os sistemas de gestão de processos não são uma

novidade existindo vários produtos quer comerciais como o *IBM WebSphere Process Server*[5] e o *Microsoft BizTalk Server*[6], quer *open source* como o Apache ODE[7] (*Orchestration Director Engine*), o Orchestra[8] e o Riftsaw[9] (este último baseado no Apache ODE).

A maior parte do trabalho de investigação nesta área foca-se na questão da modelação do *workflow* (*workflow modeling*) e nos motores de execução (*execution engines*).

A WS-BPEL[10] (*Web Services Business Process Execution Language*) tornou-se o *standard* para as linguagens de execução de *workflow*. A WS-BPEL é uma linguagem de orquestração usada para descrever o comportamento dos processos de negócio baseados em *Web Services*. Ela estende o modelo de interacção dos *Web Services* de forma a suportar transacções de negócio. A WS-BPEL é bastante popular no seio da comunidade *open source* existindo vários motores WS-BPEL, como os já referidos Apache ODE, Orchestra e Riftsaw. O *IBM WebSphere Process Server* e o *Microsoft BizTalk Server* são exemplos de motores WS-BPEL comerciais que são capazes de executar os processos descritos através da WS-BPEL. Nativamente, a WS-BPEL não suporta utilizadores móveis nem *workflows* sensíveis ao contexto, o que abre espaço para a investigação nesta área.

Pela sua importância na área dos sistemas de *workflow*, a WS-BPEL irá ser descrita de seguida para depois se efectuar uma análise de trabalhos relacionados que visam dar resposta à questão da mobilidade e sensibilidade ao contexto nos sistemas de *workflow*.

2.4.1 WS-BPEL

A WS-BPEL (*Web Services Business Process Execution Language*) é uma linguagem de programação para processos de negócios baseada em XML, que descreve como esses processos são especificados e o modelo que determina o seu funcionamento. Um processo de negócio WS-BPEL orquestra interacções com parceiros de serviços *web* para atingir um objectivo de negócio.

A WS-BPEL apresenta semelhanças com as linguagens de programação, na medida em que disponibiliza determinados tipos de primitivas como estruturas de repetição, estruturas condicionais, variáveis e atribuições. Este facto possibilita que o processo de negócio seja visto

[5]http://www-01.ibm.com/software/integration/wps

[6]http://www.microsoft.com/biztalk

[7] http://ode.apache.org

[8]http://orchestra.ow2.org

[9]http://www.jboss.org/riftsaw

[10]http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

como um algoritmo, porém mais simples e limitado de forma a facilitar a sua utilização e aprendizagem.

Todos os processos de negócios são escritos em ficheiros de definição de processos. Cada definição de processo é carregada e executada num motor WS-BPEL, uma máquina virtual que abstrai grande parte dos detalhes de comunicação com os *Web Services* e executa processos de negócio de longa duração.

A estrutura geral de um processo de negócio especificado em WS-BPEL é formada por quatro secções: *PartnerLinks*, *Variables*, *FaultHandlers* e *Activities*, as quais são descritas de seguida.

PartnerLinks. Secção que define as partes envolvidas que interagem com o processo de negócio durante a sua execução. Elas são usadas para identificar a funcionalidade que deve ser fornecida por cada serviço parceiro. As ligações de parceiros (*partner link*) devem estar associadas a um tipo de ligação entre parceiros (*partner link type*) definidos na especificação de serviços *web* em WSDL[11].

Variables. Secção que define as variáveis utilizadas pelo processo de negócio. As definições são feitas em termos de tipos de mensagem WSDL, elementos ou tipos simples de esquemas XML. Elas são usadas para manter os dados de estado e o histórico do processo com base nas mensagens trocadas. As variáveis devem estar associadas a tipos de mensagens (*messages*) definidos na especificação WSDL.

FaultHandlers. Secção que contém os tratadores de falhas que definem as actividades a serem executadas em resposta às falhas resultantes da invocação de serviços de avaliação e de aprovação.

Activities. Secção que contém a descrição do comportamento normal para a execução do processo de negócio. Existem dois tipos de actividades:

(i) **basic activities.** Descrevem os passos elementares do comportamento do processo. Algumas dessas actividades básicas envolvem a interacção com algum parceiro, como por exemplo `invoke`, `receive` e `reply`, enquanto outras são executadas sem a interacção com qualquer parceiro, incluindo-se neste grupo as primitivas `wait`, `terminate`, `assign`, `empty`, `throw` e `compensate`;

(ii) **structured activities.** Descrevem a lógica do controlo de fluxo e possuem analogias com as linguagens de programação comuns. São usadas para agrupar actividades básicas

[11]http://www.w3.org/TR/wsdl

dentro de algumas estruturas de fluxo, e incluem primitivas como `if`, `while`, `repeatUntil`, `pick`, `flow`, `sequence`, `switch` e `scope`.

A lógica do processo é realizada através do uso das actividades, que como se viu atrás, podem ser classificadas como básicas ou estruturadas. As actividades estão representadas na definição de processo como elementos XML, com as suas propriedades descritas como atributos ou elementos filho. Algumas actividades podem ter actividades filho, e todas as actividades são filhas do elemento `process`, que é a raiz de um processo WS-BPEL.

A Tabela 2.6 descreve de forma sucinta as actividades mais utilizadas.

Actividade	Significado
`<invoke>`	Invoca um *Web Service*
`<receive>`	Aguarda a resposta de um cliente
`<reply>`	Gera uma resposta síncrona
`<assign>`	Manipula dados
`<throw>`	Indica falhas ou excepções
`<wait>`	Espera um certo tempo
`<terminate>`	Finaliza um processo
`<sequence>`	Sequência de actividades a serem executadas
`<flow>`	Conjunto de actividades que deverão ser executadas em paralelo
`<if>`	Selecção
`<while>`	Repetição
`<pick>`	Aguarda por um evento

Tabela 2.6. Actividades mais utilizadas na WS-BPEL

Cada parceiro tem uma interface WSDL com pelo menos um tipo de porta (*port type*); um processo WS-BPEL também expõe uma interface WSDL com pelo menos um tipo de porta, permitindo-se a ser incluído em composições maiores. A relação entre um serviço de parceiro e um processo de negócio WS-BPEL é codificada através de um *partner link* obrigatório; cada *partner link* possui até dois papéis, e declara que tipo de porta cada papel exige e que serviços de comunicação devem satisfazer, para a interacção poder ocorrer.

Nem o *port type* nem o *partner link* especificam os formatos de mensagem ou informação específica de transporte sobre os parceiros *Web Service*. Esta informação concreta de ligação é fixada em tempo de *design* de uma forma de execução específica, ou em tempo de execução através do uso de referências de terminal.

As variáveis WS-BPEL representam o estado de um processo de negócio. Elas podem ser de três tipos: mensagem WSDL, XML-Schema, e elemento XML-Schema. As variáveis do tipo mensagem WSDL referenciam definições de tipo no próprio processo ou numa interface WSDL de um dos seus parceiros. São usadas para armazenar o resultado de uma chamada a um *Web Service* ou no envio de uma mensagem de invocação. O outro tipo de variável que precisa de explicação é o tipo XML-Schema-element. Imagine-se uma definição XML-Schema como uma estrutura, então as variáveis do tipo elemento XML-Schema referem-se a um componente dessa estrutura.

Na Figura 2.3 podemos ver um exemplo de um processo WS-BPEL para aprovação de empréstimos.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
BPEL Process Definition
-->
<process name="loanApprovalProcess" suppressJoinFailure="yes"
      targetNamespace="urn:active-endpoints.samples.bpel1_1.2005-
10.loan_approval"
      xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
      xmlns:bpws="”http://schemas.xmlsoap.org/ws/2003/03/business-process/”
      xmlns:lns="urn:active-endpoints.resources.wsdl.2005-10.loan_approval"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<partnerLinks>
<partnerLink myRole="loanService" name="customer"
            partnerLinkType="lns:loanPartnerLinkType"/>
<partnerLink name="approver"
            partnerLinkType="lns:loanApprovalLinkType"
partnerRole="approver"/>
<partnerLink name="assessor"
            partnerLinkType="lns:riskAssessmentLinkType"
partnerRole="assessor"/>
</partnerLinks>

<variables>
<variable messageType="lns:creditInformationMessage" name="request"/>
<variable messageType="lns:riskAssessmentMessage" name="risk"/>
<variable messageType="lns:approvalMessage" name="approval"/>
<variable messageType="lns:errorMessage" name="error"/>
</variables>

<faultHandlers>
<catch faultName="lns:loanProcessFault" faultVariable="error">
<reply faultName="lns:unableToHandleRequest" operation="request"
      partnerLink="customer" portType="lns:loanServicePT" variable="error"/>
</catch>
</faultHandlers>
<flow>
<links>
<link name="receive-to-assess"/>
<link name="receive-to-approval"/>
<link name="assess-to-setMessage"/>
<link name="assess-to-approval"/>
<link name="setMessage-to-reply"/>
<link name="approval-to-reply"/>
</links>
<receive createInstance="yes" operation="request"
            partnerLink="customer" portType="lns:loanServicePT"
variable="request">
<source linkName="receive-to-assess"
```

```
50        transitionCondition="bpws:getVariableData('request','amount') &lt;
51  10000"/>
52  <source linkName="receive-to-approval"
53        transitionCondition="bpws:getVariableData('request','amount') &gt;=
54  10000"/>
55  </receive>
56  <invoke inputVariable="request" operation="check"
57              outputVariable="risk" partnerLink="assessor"
58             portType="lns:riskAssessmentPT">
59  <target linkName="receive-to-assess"/>
60  <source linkName="assess-to-setMessage"
61              transitionCondition="bpws:getVariableData('risk','level')='low'"/>
62  <source linkName="assess-to-approval"
63
64        transitionCondition="bpws:getVariableData('risk','level')!='low'"/>
65  </invoke>
66  <assign>
67  <target linkName="assess-to-setMessage"/>
68  <source linkName="setMessage-to-reply"/>
69  <copy>
70  <from expression="'yes'"/>
71  <to part="accept" variable="approval"/>
72  </copy>
73  </assign>
74  <invoke inputVariable="request" operation="approve"
75               outputVariable="approval" partnerLink="approver"
76                portType="lns:loanApprovalPT">
77  <target linkName="receive-to-approval"/>
78  <target linkName="assess-to-approval"/>
79  <source linkName="approval-to-reply"/>
80  </invoke>
81  <reply operation="request" partnerLink="customer"
82               portType="lns:loanServicePT" variable="approval">
83  <target linkName="setMessage-to-reply"/>
84  <target linkName="approval-to-reply"/>
85  </reply>
86  </flow>
87  </process>
```

Figura 2.3. Exemplo de um processo de negócio WS-BPEL para aprovação de empréstimos

A Figura 2.4 mostra o processo anterior desenhado com o *ActiveVOS Designer* que é uma ferramenta gráfica para desenho de processos[12].

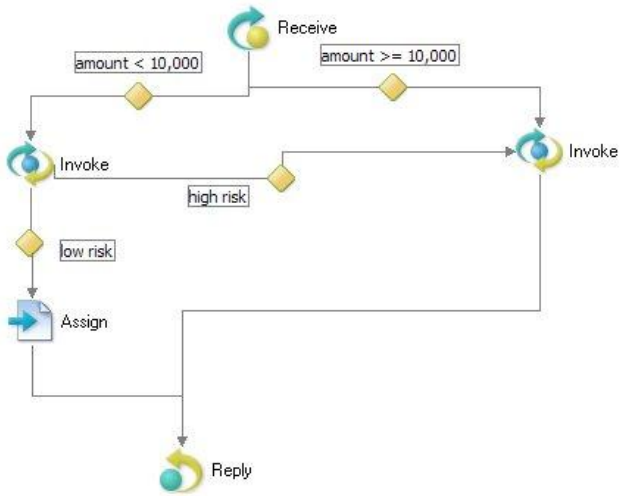


Figura 2.4. O mesmo processo editado no ActiveWebflow

[12] http://www.activevos.com/products/activevos/features/designer

2.4.2 Trabalho Relacionado

O Context4BPEL [71] e o CAWE [72] são projectos focados no uso de *frameworks* e *execution engines* para domínios muito específicos sendo incapazes de dar uma resposta mais genérica. O Context4BPEL é uma variante do WS-BPEL para a modelação de *workflows* sensíveis ao contexto e foi testado em processos de produção na indústria de transformação. O CAWE é uma *framework* para o desenvolvimento de aplicações sensíveis ao contexto baseada em *Web Services*, que se adaptam às características do utilizador e do contexto de execução, tendo sido testado num cenário de gestão hospitalar.

Projectos como o Sliver [73], o xBPEL [68], o WHAM [67] e o proposto por Pajunen e Chande [74], são *execution engines* que suportam utilizadores móveis. O xBPEL e o WHAM usam apenas a localização como único tipo de contexto, não sendo por conseguinte, genéricos o suficiente para permitir uma maior expressividade na modelação de *workflows* sensíveis ao contexto.

O CAWD [75] é um trabalho recente que fornece uma ferramenta de desenho para desenvolver modelos de *workflow* para aplicações sensíveis ao contexto. Ele é construído sobre o *Microsoft Windows Workflow Foundation 3.5* e funciona como uma aplicação *standalone* para o desenho e verificação de *workflows* sensíveis ao contexto.

Tang *el al.* [76] propõem um modelo de contexto e um algoritmo para a gestão de *workflows* sensíveis ao contexto, que foi utilizado para a navegação ubíqua num *campus* e que foi modelado através de redes de Petri.

O modelo de *workflow* orientado a eventos (*event-driven workflow*) usado neste trabalho foi em parte inspirado no trabalho pioneiro de Dayal *el al.* [77] onde a notação ECA (*Event-Condition-Action*) foi proposta. Jang *et al.* [70] propuseram uma abordagem similar chamada GAT (*Guard-Action-Trigger*) para o desenho de um motor de *workflow* que suporta a definição e execução de processos de negócio de longa duração.

No essencial, os projectos vistos nesta secção visam permitir a mobilidade e a sensibilidade ao contexto aos *workflows*, actuando o dispositivo móvel como uma extensão do sistema de *workflow*, segundo uma lógica de orquestração de serviços.

A principal diferença entre estes projectos e o proposto neste trabalho está na forma como um *workflow* é visto. Neste trabalho um *workflow* é visto não como um processo de negócio, mas como parte de uma aplicação sensível ao contexto que é criada e utilizada por utilizadores finais. São usados os mesmos conceitos de *workflow* sensível ao contexto, mas numa lógica de execução

local das actividades disponíveis. As actividades são usadas como blocos de construção dos *workflows* as quais são executadas quando o contexto de início do *workflow* for verificado.

2.5 Desenvolvimento pelo Utilizador Final

Hoje em dia, a maior parte das pessoas está familiarizada com as funcionalidades e interfaces básicas dos computadores, fazendo uso dos programas principalmente como utilizador final. Os programadores desses programas, muitas vezes não conhecem os detalhes do trabalho das pessoas que os vão utilizar, acabando por adicionar muitas funcionalidades que a maioria dos utilizadores não vai utilizar, numa tentativa de dar resposta à maior parte das necessidades desses utilizadores. Por outro lado, o utilizador pode necessitar de outras funcionalidades que não foram previstas ou que precisam ser implementadas devido a novos requisitos. Neste caso, o utilizador pode caso a aplicação que usa o permita, estender as funcionalidades de um programa ou configurá-lo segundo as suas necessidades.

Lieberman *et al.* [78] definem programação como a arte de ensinar novos comportamentos a um computador. Esse ensinamento pode ocorrer de duas formas: a forma tradicional de programar em que o utilizador (programador profissional ou não) usa uma linguagem de programação adequada; e ainda através de metáforas de programação que permitem a utilização de funcionalidades ou ferramentas que auxiliam o utilizador a alterar ou definir novos comportamentos para os programas.

A expressão programação pelo utilizador final (*End-User Programming* ou apenas EUP) foi introduzida por Nardi [79] no trabalho de pesquisa acerca da utilização das folhas de cálculo no local de trabalho. Um utilizador final é qualquer utilizador de computador, enquanto a EUP é a programação efectuada pelo próprio utilizador para alcançar determinado objectivo para uso pessoal em vez de uso público.

Na prática, o que distingue os utilizadores finais programadores dos programadores profissionais é o objectivo com que realizam as tarefas de programação. Os profissionais são pagos para desenvolver e manter *software* ao longo do tempo, enquanto os utilizadores finais, pelo contrário, escrevem programas para auxiliar nalgum objectivo nos seus domínios de especialização [80]. Os utilizadores finais programadores podem ser secretárias, contabilistas, cientistas, ou qualquer pessoa que escreve programas para a auxiliar no seu trabalho ou num *hobby*. Apesar das competências de programação de alguns utilizadores finais poderem ser grandes, essas são usadas de forma diferente dos programadores profissionais. Por exemplo, os administradores de sistemas usam a programação apenas como um meio para manter uma rede e outros serviços

online, o mesmo se passando com muitos cientistas que escrevem programas para auxiliar nos seus trabalhos de pesquisa [80].

Lieberman *et al.* [78] definem desenvolvimento pelo utilizador final (*End-User Development* ou apenas EUD) como:

> *"End-User Development can be defined as a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artifact."*

Esta definição introduz a noção do ciclo de vida do *software* na medida em que refere a utilização e adaptação do *software* ao longo do tempo. Os artefactos referidos na definição de Lieberman *et al.* [78] podem ser objectos que descrevem um comportamento automático ou uma sequência de controlo. Fischer [81] acrescenta que estes artefactos também podem referir-se à criação de conteúdo gerado pelo utilizador que podem ser ou não interpretados por computadores. Estes objectos são descritos através de paradigmas de programação, não existindo um único método ou paradigma. As metáforas de programação mais adequadas, as capacidades e limitações, são muito diferentes dependendo dos utilizadores e dos seus objectivos com a programação.

Actualmente, algumas das formas de EUD têm um uso bastante generalizado e com sucesso em *software* comercial, como seja a gravação de macros dos processadores de texto, a customização de folhas de cálculos e a definição de filtros de *email* [78]. Estes autores identificam duas formas de EUD: a Parametrização ou Customização e a Criação e Modificação de programas. Estas formas de EUD são descritas de seguida.

2.5.1 Parametrização ou Customização

Esta forma de EUD envolve actividades que permitem aos utilizadores escolher entre comportamentos alternativos já disponíveis na aplicação. Apesar de estar amplamente implementada em muito *software* comercial, esta forma de EUD está limitada às opções pré-definidas.

Os sistemas adaptativos são sistemas que permitem que esta personalização aconteça de forma automática em resposta à observação do comportamento do utilizador.

2.5.2 Criação e Modificação de Aplicações

Envolve actividades que implicam algumas modificações, visando a criação de programas a partir do nada ou modificação de um artefacto de *software* já existente. Exemplos desta abordagem

incluem a gravação de macros, linguagens de *script*, programação por exemplos (também chamada de programação por demonstração) e a programação visual.

2.5.2.1 Gravação de Macros

A gravação de macros, disponível em muitas aplicações, permite que o utilizador grave uma sequência de acções. Essa macro pode ser associada a uma opção de menu, botão ou atalho do teclado para ser reproduzida sempre que o utilizador a invocar. A vantagem desta abordagem é a familiaridade do utilizador com a aplicação, uma vez que a gravação da sequência de acções se processa com o utilizador a usar normalmente a aplicação. A grande desvantagem diz respeito ao facto deste processo efectuar uma gravação literal, replicando os eventos gerados pelo utilizador e mantendo constantes os valores atribuídos às variáveis do contexto da macro.

2.5.2.2 Linguagens de Script

As linguagens de *script* exigem ao utilizador um esforço de aprendizagem dessas linguagens, que por mais simples que sejam obrigam a conhecer a sua sintaxe e convenções para as usar. Nesta perspectiva as linguagens de *script* assemelham-se muito com as linguagens de programação. A grande diferença entre elas é que as linguagens de *script* são interpretadas podendo ser escritas num simples editor de texto.

2.5.2.3 Programação por Exemplos

A programação por exemplos é uma técnica para ensinar um comportamento novo ao computador, demonstrando as acções a partir de exemplos concretos. O sistema regista as acções do utilizador e generaliza um programa que pode ser usado em novos exemplos. Na área das aplicações sensíveis ao contexto, existem alguns projectos que usam esta técnica para a criação de aplicações. *a Cappella* [9] disponibiliza um ambiente de programação por demonstração para utilizadores finais de forma a permitir a prototipagem de aplicações sensíveis ao contexto, utilizando quatro componentes principais: um sistema de gravação, uma detecção de eventos, uma interface de utilizador e um sistema de aprendizagem. O Topiary [13] permite a "demonstração" de cenários que representam contextos locais, para depois usar esses cenários na criação de *storyboards* que descrevem sequências de interacção que podem ser executados em dispositivos móveis

2.5.2.4 Programação Visual

A programação visual refere-se a um conjunto de técnicas de interacção e notações visuais para expressar programas [80]. Esta técnica procura fazer uso da facilidade que os seres humanos apresentam no processamento de informação visual, utilizando representações visuais bem como diagramas para descrever o conjunto de dados ou operações e as suas relações.

A **programação visual de fluxos de controlo**, visa a criação de representações visuais para as principais estruturas de programação como iterações, condições e chamadas a funções. Os objectos gráficos usados seguem o padrão dos ícones utilizados nas aplicações, sendo portanto mais amigável para o utilizador que as linguagens textuais.

Na **programação visual baseada em regras** o utilizador define condições e, para cada uma destas condições, as acções associadas. São um subconjunto do mecanismo anterior, uma vez que só existe representação para a estrutura condicional.

2.5.3 Custo de Aprendizagem e Abrangência

O objectivo das ferramentas EUD é o de reduzir a carga de aprendizagem, proporcionando funcionalidades capazes de dar resposta a um vasto leque de problemas. Dado que o peso de aprendizagem existirá sempre, estas ferramentas deverão motivar os utilizadores para a sua utilização. Fisher *et al.* [82] propõem uma abordagem de *meta-design*, em que os utilizadores são motivados para a aprendizagem através de exemplos e demonstrações de sistemas em funcionamento de forma a mostrar-lhes o que é possível conseguir. Estes autores apresentam uma classificação de linguagens e ferramentas de programação, de acordo com o custo de aprendizagem e a sua abrangência.

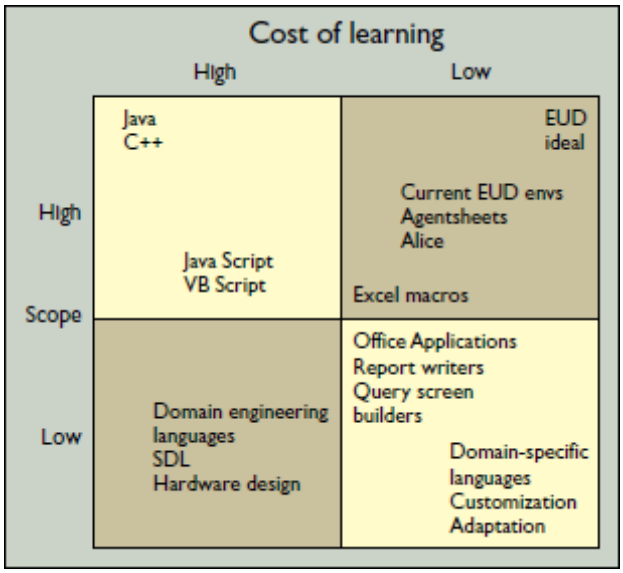


Figura 2.5. Custo de aprendizagem vs. abrangência das ferramentas EUD [82]

No quadrante superior esquerdo correspondente ao maior custo de aprendizagem e maior abrangência, temos as linguagens de programação tradicionais como a Java e a C++, utilizadas por programadores profissionais ou não. Na fronteira com o quadrante inferior temos as linguagens de *script* as quais surgem como versões simplificadas de linguagens de programação completas. O quadrante inferior esquerdo referente a menor abrangência e alto custo é ocupado

por um pequeno número de linguagens de programação específicas de um determinado domínio e criadas para dar resposta a requisitos em domínios complexos da engenharia, tais como controladores industriais PLC (*Programmable Logic Controller*). No quadrante referente a baixo custo e pouca abrangência (quadrante inferior direito), temos as linguagens EUD específicas de domínio, as quais reduzem a carga de aprendizagem mas que apenas respondem a uma área de aplicação específica. Neste quadrante temos a personalização de pacotes de *software*, em que a tarefa de programação é reduzida à configuração de parâmetros em janelas de diálogo. Na fronteira com o quadrante superior, temos as linguagens macro que estendem as aplicações de escritório, como por exemplo, as fórmulas das folhas de cálculo e as linguagens de consulta de base de dados. Finalmente, o quadrante superior direito, correspondente a baixo custo de aprendizagem e grande nível de abrangência, temos as linguagens e ferramentas ideais de EUD, as quais hoje em dia não existam como tal. O estado da arte actual em EUD fornece ambientes gráficos para criar agentes programáveis, mas que ainda obrigam a um grande nível de aprendizagem na concepção de regras do tipo *condition/action*, e no desenho de modelos dos agentes.

Capítulo 3

Framework IVO

Conteúdo

Resumo

Este capítulo apresenta a *framework* de suporte à plataforma que permite que os utilizadores finais possam criar e utilizar aplicações sensíveis ao contexto usando como dispositivo ubíquo *smartphones* e *tablets* actualmente disponíveis no mercado. Começa-se por definir um conjunto de princípios orientadores que serviram de referência para a definição da arquitectura. São depois apresentados um conjunto de cenários de casos de uso que permitiram definir o conjunto de actores do sistema já na secção seguinte. Com base nos cenários de casos de uso, é de seguida efectuado o desenho da arquitectura onde são identificados os principais componentes e o levantamento de requisitos.

3.1 Princípios Orientadores

A *framework* proposta neste trabalho baseia-se no conjunto de princípios orientadores descritos de seguida. Esta *framework* visa dar resposta ao problema enunciado nos capítulos anteriores de

forma a facilitar e generalizar o desenvolvimento e utilização de aplicações sensíveis ao contexto por parte de utilizadores finais.

3.1.1 Abordagem de Desenho Centrada no Utilizador

O objectivo do desenho centrado no utilizador (*User Centered Design* ou UCD) é o de criar produtos utilizáveis que respondam às necessidades do utilizador sendo considerada uma filosofia de desenho que coloca o utilizador no centro do processo [83]. Numa abordagem de desenvolvimento pelo utilizador final como é o caso do presente trabalho, o utilizador é um actor sempre presente nas diversas actividades que compõem o processo de desenvolvimento e utilização das aplicações criadas através da plataforma. Assim, no modelo aqui proposto, os utilizadores são envolvidos num processo iterativo de desenho e testes de forma a aperfeiçoar as aplicações criadas e a garantir que cumprem os requisitos.

As questões de usabilidade assumem um papel preponderante no desenho das ferramentas que permitem aos utilizadores finais criarem as aplicações. As interfaces deverão ser simples e fáceis de utilizar com ajudas e mensagens claras que permitam a utilizadores com diferentes formações base e sem formação prévia nas ferramentas, utilizá-las sem qualquer dificuldade.

3.1.2 Desenho Baseado em Cenários

É proposto neste trabalho um modelo de desenho baseado em cenários o qual é suportado pela metodologia de desenvolvimento detalhada no Capítulo 5. Os cenários são apropriados para descrever o comportamento de um sistema a estímulos do ambiente externo [84–86], sendo por isso um mecanismo interessante para descrever aplicações sensíveis ao contexto. Potts [87] define um cenário como sendo um caso particular de como o sistema deve ser usado, ou em sentido mais amplo, um cenário é simplesmente uma proposta de uso específico do sistema o qual se pode traduzir sob a forma de pequenas histórias [85].

Potts [87] considera que um cenário deverá ser caracterizado pelos elementos: (i) *setting* que define o estado de partida para o episódio que é descrito; (ii) *agents* ou *actors* intervenientes no cenário que tipicamente têm determinado objectivo na realidade do cenário; e (iii) *actions* que definem o enredo do cenário, onde se incluem sequências de acções e eventos, coisas que os *agents* ou *actors* fazem, coisas que lhes acontecem e mudanças no ambiente.

No contexto deste trabalho considera-se que um cenário deverá ser caracterizado pelos seguintes elementos:

- **Informação contextual** associada ao cenário, podendo ser estática ou dinâmica e com capacidade de adaptação ao contexto. Adopta-se como formato de representação da

informação o HTML[13] por ser um formato popular com capacidade de mostrar imagens, vídeos ou *Flash*[14]. Esta informação deverá ser capaz de ser apresentada em dispositivos móveis com ecrãs de várias dimensões.

- **Situação** que define com exactidão o contexto do cenário de acordo com as dimensões de localização, de tempo, de proximidade e ainda de estado. A dimensão de estado refere-se a variáveis de estado do dispositivo móvel tais como o nível da bateria ou luminosidade. A situação corresponde a uma condição que é avaliada sempre que há uma mudança do ambiente em que o utilizador se encontra. Esta condição pode ser simples se usada em apenas uma dimensão de contexto ou composta se incluirmos várias, traduzindo-se na prática na avaliação de expressões booleanas que podem ser mais ou menos complexas. Os contextos podem ser partilhados entre vários cenários.

- **Workflow** que deverá ser executado quando a situação definida anteriormente se verificar. O motor de *workflow* que irá correr os processos deverá ser suficientemente expressivo para permitir a criação de fluxos.

- **Formulários e quizzes** que são executados como uma actividade de *workflow***.** Os formulários e os *quizzes* podem ser partilhados entre vários cenários.

3.1.3 Modelo Evento-Condição-Workflow

O modelo de funcionamento proposto neste trabalho foi chamado de Evento-Condição-Workflow (*Event-Condition-Workflow*) ou simplesmente ECW, o qual foi em grande parte inspirado no trabalho pioneiro de Dayal *el al.* [77] onde a notação ECA (*Event-Condition-Action*) foi utilizada pela primeira vez. Jang *et al.* [70] propuseram uma abordagem similar chamada GAT (*Guard-Action-Trigger*) para a concepção de um motor de *workflow* que suporta a definição e execução de processos de negócio de longo prazo. Os projectos de Dayal *et al.* e de Jang *et al.*, visam adicionar mobilidade e sensibilidade ao contexto a *workflows*, com o dispositivo móvel a actuar como uma extensão de um sistema gestão de processos, seguindo uma lógica de orquestração de serviços. A principal diferença entre esses projectos e o aqui proposto está na forma como um *workflow* é utilizado. Neste trabalho, os *workflows* são utilizados não como representação de um processo de negócio, mas como parte de uma aplicação sensível ao contexto que é criada e usada por utilizadores finais.

---

[13] HTML ou *Hyper Text Markup Language*, é uma linguagem de marcação utilizada para produzir páginas na *web*.

[14] Adobe Flash ou simplesmente Flash, é um *software* utilizado geralmente para a criação de animações interactivas que funcionam embebidas num *browser web* em ambientes desktops, telemóveis, *smartphones* e *tablets*.

O modelo ECW aqui proposto permite que os *workflows* sejam iniciados por eventos que resultam da avaliação de um mecanismo de regras do tipo *if-then* significando coisas como "*se eu* ..." ou "*quando eu* ..." estou numa determinada situação, executar determinado conjunto de acções. A maior parte das aplicações sensíveis ao contexto pode ser descrita desta forma. O estudo conduzido por Dey *et al.* [14], sobre o modelo mental dos utilizadores acerca de aplicações sensíveis ao contexto, concluiu que a maioria deles se expressa em termos de regras *if-then*. As condições das regras *if-then* deverão ser suficientemente abrangentes para permitir a avaliação de condições que façam uso dos operadores matemáticos e booleanos mais comuns. Por outro lado, a utilização de *workflows* permite definir o fluxo do programa, expresso como uma sequência de actividades, que podem incluir caminhos alternativos, ciclos e paralelismo.

3.1.4 Programação Visual e de Controlo de Fluxo

Devem ser disponibilizadas ferramentas de desenvolvimento baseadas em ambientes de programação visuais, uma vez que estes tipos de ambientes têm provado ser eficazes em tirar partido das capacidades de raciocínio espacial do utilizador [88]. Os ambientes gráficos de controlo de fluxo para a construção dos *workflows*, tornam mais fácil ao utilizador entender como as coisas se ligam, ao contrário da programação imperativa, que se concentra em como as coisas acontecem.

3.1.5 Linguagem de Marcação para Descrever as Aplicações

Propõe-se uma abordagem de linguagem de marcação para descrever as aplicações. Esta linguagem deve permitir descrever todos os elementos que constituem os cenários nomeadamente a informação contextual, a situação a que o cenário se aplica, os *workflows*, os formulários utilizados e os *quizzes*.

A opção por esta abordagem justifica-se pelo facto da utilização da XML tornar mais simples a comunicação distribuída e a interoperabilidade, não havendo necessidade de utilizar modelos mais formais como as ontologias. Este facto permite que uma aplicação criada possa ser distribuída e correr em dispositivos móveis baseados em várias plataformas como *Android*, *iPhone* ou *Windows Phone* sem necessidade de manter versões distintas para cada uma delas.

As ferramentas de construção disponibilizadas devem permitir criar e alterar ficheiros XML segundo o esquema descrito no Capítulo 4, o qual foi chamado de *IVO Markup Language* ou IVOML. A IVOML é uma linguagem de marcação simples mas suficientemente completa para permitir descrever completamente uma aplicação construída com a plataforma, podendo ainda

ser embebida em formatos *standards* de representação de informação geográfica, como seja o KML [89] e o GPX [90].

3.1.6 Comunicação Distribuída e Transparente

Por natureza os sistemas sensíveis ao contexto são modulares e distribuídos, devendo a comunicação entre os seus componentes ser tratada de forma adequada, através da utilização de protocolos de comunicação e *standards* para integração de componentes. Os *Web Services* [91] permitem dar resposta a estas necessidades ao propiciarem uma comunicação padronizada entre aplicações diferentes, promovendo a integração entre elas. Propõe-se neste trabalho a utilização de uma API baseada em *Web Services* REST [92] para a comunicação entre os vários componentes da plataforma, por estes serem cada vez mais uma tendência no desenvolvimento de serviços e aplicações na *Web* 2.0.

3.2 Cenários de Uso

Nesta secção são apresentados vários cenários de uso, os quais juntamente com os requisitos genéricos para o desenvolvimento de aplicações sensíveis ao contexto apresentados na Secção 2.2.5, foram tidos em conta na elaboração dos requisitos genéricos do IVO e apresentados na secção seguinte.

Cenário 1 – Criação de um guia turístico

"José é um professor de História do secundário que adora "viver a História ao vivo", fazendo frequentemente pequenas escapadelas pelos mais diversos locais históricos do mundo. Estas escapadelas são sempre preparadas ao pormenor de forma a aproveitar todo o tempo disponível. A família acompanha-o sempre, mas nem sempre estas visitas agradam aos filhos que preferiam outros tipos de actividades, ocorrendo por vezes pequenas birras...

José está a planear para o fim-de-semana a próxima escapadela Histórica desta vez por Évora. Em vez de levar consigo as notas e impressões que tira durante a preparação das visitas, decide desta vez tentar colocar essa informação no seu novo smartphone Android oferecido pelos filhos no Natal. Um colega havia-lhe falado de um site que disponibiliza guias turísticos, e que permite também construir os próprios guias e partilhá-los de seguida. Decide experimentar e começa por criar uma conta no site. Verifica que não existe ainda nenhum guia disponível para Évora, e como o colega lhe tinha dito que a criação de novos guias era um processo fácil e que nem era preciso perceber muito de informática, decide ele próprio meter mãos à obra.

Ele já tem uma ideia do que quer visitar, e começa por fazer uma pesquisa na internet à procura de mais informação sobre Évora. Descobre que o site do turismo da região lhe dá muita informação que ele próprio complementa aqui e ali com o seu próprio conhecimento sobre História.

Depois de reunida a informação sobre os pontos a visitar, é tempo de voltar ao tal site e criar o guia. Começa por marcar todos os pontos a visitar no mapa disponível no site e adiciona a cada um deles a informação entretanto recolhida usando as ferramentas disponibilizadas incluindo os contactos, os horários de abertura, informação sobre as facilidades existentes nos locais.

Repara que pode enriquecer com imagens esta informação, bem como executar automaticamente um conjunto variado de acções quando entra e quando sai de cada um dos pontos de interesse, como por exemplo mostrar um vídeo, tocar uma música de fundo, colocar o telefone em modo silencioso ao entrar e voltar a colocá-lo em modo normal ao sair, adicionar áudio explicativo do local sem ter que gravar nada bastando escrever o que quer que seja falado. Tudo isto pode tornar a visita bem mais apelativa e só tem de definir o que quer que seja feito automaticamente! Descobre ainda que existem ferramentas para criar quizzes o que pode tornar a visita mais lúdica e educativa, bem como a possibilidade de ao sair de um ponto, o visitante ser guiado para o próximo ponto da visita usando um navegador como o do GPS do carro.

Depois de estar satisfeito com o guia criado, decide disponibilizá-lo, mas como quer testar primeiro tudo quando for realmente fazer a visita, decide não o tornar ainda público colocando-o apenas disponível para si e para o filho mais velho que tem um iPhone.

Acede à página do site dedicada à instalação do software necessário para que o guia que acabou de criar possa correr no seu smartphone. É redireccionado para a loja de aplicações do Android (Google Play), de onde descarrega e instala esse pequeno programa. Depois de instalado, o programa informa-o que tem o guia que acabou de criar pronto para ser descarregado para o seu telefone o que ele faz de imediato começando logo a explorar o guia que criou. Vê como a informação lhe aparece no telefone e percorre as funcionalidades que o programa lhe oferece. O resto fica para o fim-de-semana…"

Cenário 2 – Utilização do guia turístico

"Bob e Alice são dois estudantes Ingleses que este ano vão fazer um semestre à Universidade de Évora ao abrigo do programa Erasmus. Através do Facebook de um dos seus professores, descobrem os posts que José tinha colocado quando foi fazer a visita a Évora e verificam que também eles podem usar a aplicação do José para conhecer melhor Évora assim que lá chegarem. Ainda em Inglaterra, começaram por instalar a aplicação, mas rapidamente descobrem que esta só está disponível com os conteúdos em Português. Apercebem-se que eles próprios podem efectuar a tradução dos conteúdos usando a mesma ferramenta que o José usou para a criar, e decidem meter mãos à obra – afinal sempre dava para treinar o português. No final do dia já tinham todo o conteúdo traduzido para inglês e disponibilizam esta versão aos utilizadores do mundo inteiro.

Chegados a Évora e depois de se instalarem na residência, Bob e Alice aproveitam o que resta do fim-de-semana para conhecer a cidade usando a versão do guia traduzida por eles. Começam por correr a aplicação, surgindo-lhes

um ecrã onde podem seleccionar quais as acções que pretendem que sejam automaticamente executadas de entre aquelas que o José havia definido quando estava a criar o guia. Bob escolhe não fazer actualizações de estado no Facebook, enquanto a Alice decide que os amigos vão ficar a saber por onde ela anda através das actualizações automáticas no Facebook que a aplicação faz. Ao entrar em cada um dos pontos, o estado do seu Facebook é actualizado informando os amigos que ela se encontra a visitar esse ponto, sendo ainda colocado no post a informação associada ao ponto que eles entretanto tinham traduzido para inglês – assim os amigos podem não só ficar a saber onde ela está, como também ficarem a conhecer melhor, eles próprios, Évora.

Depois de definidas estas preferências, surge-lhes um mapa com a localização dos pontos de interesse que José tinha criado. Começam por explorar a funcionalidade de realidade aumentada, o que lhes permite ver o espaço em redor através da câmara do smartphone com informação sobreposta acerca dos pontos de interesse nas proximidades. Depois de familiarizados com o espaço ao seu redor, dirigem-se para o primeiro ponto usando as indicações de direcção fornecidas. Ao aproximarem-se desse primeiro ponto de interesse, a informação sobre esse ponto é automaticamente mostrada e é falado o texto que eles haviam traduzido para inglês. Como se trata de uma igreja, José definiu que os smartphones deveriam ser automaticamente colocados em modo silencioso para não incomodar. Ao sair, os smartphones são colocados de novo em modo normal e é mostrado o quiz associado ao ponto de interesse. Respondem em conjunto e partilham o resultado no Facebook permitindo que também os amigos possam aprender algo mais sobre a igreja que acabaram de visitar. De seguida, é automaticamente lançado o navegador que lhes fornece indicações de como se dirigirem até ao próximo ponto através de um percurso pedestre. Enquanto se dirigem para esse ponto vão ouvindo a música de fundo que José definiu para ser tocada a seguir ao lançamento do navegador. Trata-se de um canto Alentejano, ao qual foi sobreposto um texto que é falado pelo smartphone explicando as origens, a tradição e o futuro deste género musical que vai ser candidato a património imaterial da Humanidade.

Chegados ao novo ponto de interesse é mostrada a descrição deste e mais uma vez falado o texto acerca do mesmo. Terminada a visita, usam a aplicação para fazer uma avaliação deste ponto de interesse e deixam um comentário juntamente com uma fotografia que eles próprios tiraram para que os outros visitantes possam ter feedback acerca deste local.

Como a hora de almoço se aproximava e o próximo ponto de interesse estava prestes a fechar, o sistema sugere-lhes um restaurante de comida tradicional perto do local onde se encontravam – a visita iria continuar depois…"

Cenário 3 – Aplicação de suporte urbanístico

"Maria, a mulher do José, trabalha na Câmara Municipal no Departamento de Ordenamento do Território e é responsável pela manutenção de toda a informação geográfica existente nos municípios: a cartografia, o cadastro rústico, as ortofotografias, os planos municipais de ordenamento do território (Planos Directores Municipais, Planos de Urbanização e Planos de Pormenor), os equipamentos, informação estatística e turística diversa,

recorrendo para o efeito ao Sistema de Informação Geográfica (SIG) adquirido pelo Município. Os técnicos que estão no terreno têm de levar consigo um conjunto de mapas, plantas e outros documentos, acontecendo por vezes não terem toda a informação que necessitam ou então esta não estar actualizada.

Depois da visita a Évora, a Maria lembrou-se que seria útil disponibilizar alguma desta informação aos técnicos que estão no terreno bem como à Protecção Civil. Eles podiam usar smartphones ou, melhor ainda, tablets que permitissem ver com mais clareza a informação. Claro que seria impensável voltar a reintroduzir todos os dados no IVO, mas se houvesse uma possibilidade de importar os dados no IVO a partir de um dos formatos exportados pelo SIG da Câmara, então tudo seria mais simples.

Depois de criar uma conta no IVO verifica que este importa ficheiros KML, um dos formatos mais utilizados para representação geográfica, e em poucos minutos tem toda a informação geográfica do Município disponível em smartphones ou tablets através do IVO e devidamente categorizada: infra-estruturas e equipamentos, planos directores, turismo, entre outras. Verifica como essa informação foi importada e ajusta aqui e ali de forma a lhe parecer melhor.

Fazendo uso da funcionalidade de desenho de formulários do sistema, decide aventurar-se na criação de um para permitir o reporte ao Departamento de Obras da necessidade de intervenção em determinada área.

Depois de satisfeita, disponibiliza esta aplicação aos funcionários da câmara seleccionados fornecendo ao sistema o seu email e uma mensagem personalizada a explicar a aplicação e as instruções de como a instalar.

O Joaquim que trabalha no Gabinete de Apoio Técnico da Câmara começou a usar a aplicação que a Maria criou desde que ela a disponibilizou. Numa das visitas que tem de fazer para mostrar lotes de terreno a um possível interessado, depara-se com um cano de abastecimento de água roto. Usa o IVO preenchendo o formulário de necessidade de intervenção no local, anexando uma foto do incidente. Estes dados juntamente com as coordenadas GPS do local são enviados automaticamente para o Chefe do Departamento de Obras que logo envia uma equipa ao local para solucionar o problema. No local, mostra ao potencial investidor os diversos terrenos disponíveis usando a informação que o sistema automaticamente lhe mostra sempre que se aproxima dos diversos lotes de terreno – a área, as infra-estruturas existentes e claro o preço."

Cenário 4 – Aplicação para vinicultura

"O Mário é Técnico Agrícola numa empresa de produção de vinhos. Uma das suas funções é a de efectuar visitas regulares aos fornecedores de uva de forma a realizar o acompanhamento necessário para garantir a qualidade do produto final. Os fornecedores encontram-se dispersos por vários locais do Alentejo produzindo cada um deles vários tipos de uva usada na produção do vinho. De acordo com o resultado das observações realizadas, assim têm de ser tomadas determinadas acções de prevenção dos eventuais problemas resultantes do estado fenológico das

vinhas. Em cada visita, o Mário tem de preencher a ficha fitossanitária que é posteriormente introduzida no sistema informático de gestão da produção da empresa o que já tem dado origem a erros de introdução de dados.

Com a ajuda do responsável informático decide automatizar este processo, de forma a que, ele e os restantes Técnicos Agrícolas possam utilizar os smartphones novos da empresa na realização das visitas.

Começa por definir através do mapa disponível no IVO as áreas referentes a cada vinha adicionando informação acerca das uvas aí plantadas, com todas as características das mesmas e em que vinhos são utilizadas. Gostaria de ter também o histórico das visitas para que quando se encontrar no local ter toda a informação que o possa auxiliar nas diversas tomadas de decisão. Como não sabe como fazer isso pede a ajuda do responsável da informática que o ajuda criando uma interface web que lhe fornece esse histórico já formatado em HTML.

Cria um formulário para a ficha fitossanitária através da ferramenta de desenho de formulários e define a acção associada ao botão de Save como correspondendo à execução do Web Service que o responsável informático lhe criou para inserir os dados da ficha no sistema de gestão da produção.

Com a ferramenta de desenho de workflows define o fluxo com as actividades e as regras a serem executadas e verificadas quando estiver no terreno. Quando se encontrar em cada uma das vinhas a aplicação deve lançar automaticamente o preenchimento do formulário com os dados de cadastro da mesma já preenchidos, e depois, de acordo com o resultado das observações realizadas, deve definir as acções apropriadas a serem tomadas para a prevenção dos eventuais problemas resultantes do estado fenológico das vinhas – realizar determinado tratamento ou manutenção e agendar uma nova visita. Em função destas acções assim a informação deve seguir para determinados departamentos da empresa para ser dado seguimento."

Cenário 5 – Agenda

"O Manuel tem a discussão da sua tese de doutoramento marcada para a próxima segunda-feira às 14:00. Marca na sua agenda electrónica este importante evento da sua vida e para garantir não ser incomodado com o telefone a tocar, define no software de agenda que no início do compromisso o telefone deve ser automaticamente colocado em modo silencioso e deve ser colocado também automaticamente nas redes sociais do Facebook e Twitter uma mensagem a informar os amigos que finalmente este dia chegou. Quando o compromisso terminar o perfil do telefone deverá voltar ao modo em que estava anteriormente."

3.3 Actores do Sistema

Dos cenários acima descritos pode-se identificar duas categorias de actores: os criadores de aplicações e os utilizadores de aplicações.

Criadores de Aplicações. São utilizadores com formação base diversificada que utilizam as ferramentas disponibilizadas pela plataforma de modo a criar novas aplicações sensíveis ao contexto. O foco da sua actividade enquanto criadores das aplicações são as funcionalidades específicas destas, enquanto as funcionalidades genéricas como a gestão do contexto, a avaliação de regras e a coordenação das actividades que compõem os *workflows* deverão ser fornecidas pelo *runtime* que permite correr as aplicações criadas.

Utilizadores das Aplicações. São os utilizadores que usam as aplicações criadas beneficiando de alguma forma desta utilização. Usam as aplicações da forma descrita nos cenários de caso de uso: movimento e orientação em torno de uma determinada área, visualização de informação associada a pontos de interesse, obtenção de pontos de interesse nas proximidades, automação de determinadas tarefas sempre que determinada condição ocorrer, entre outras.

3.4 Arquitectura do Sistema

Tendo por base os princípios orientadores e os cenários de casos de uso apresentados anteriormente definiu-se a arquitectura da plataforma a qual se encontra ilustrada na Figura 3.1 e que é composta pelos componentes descritos de seguida.

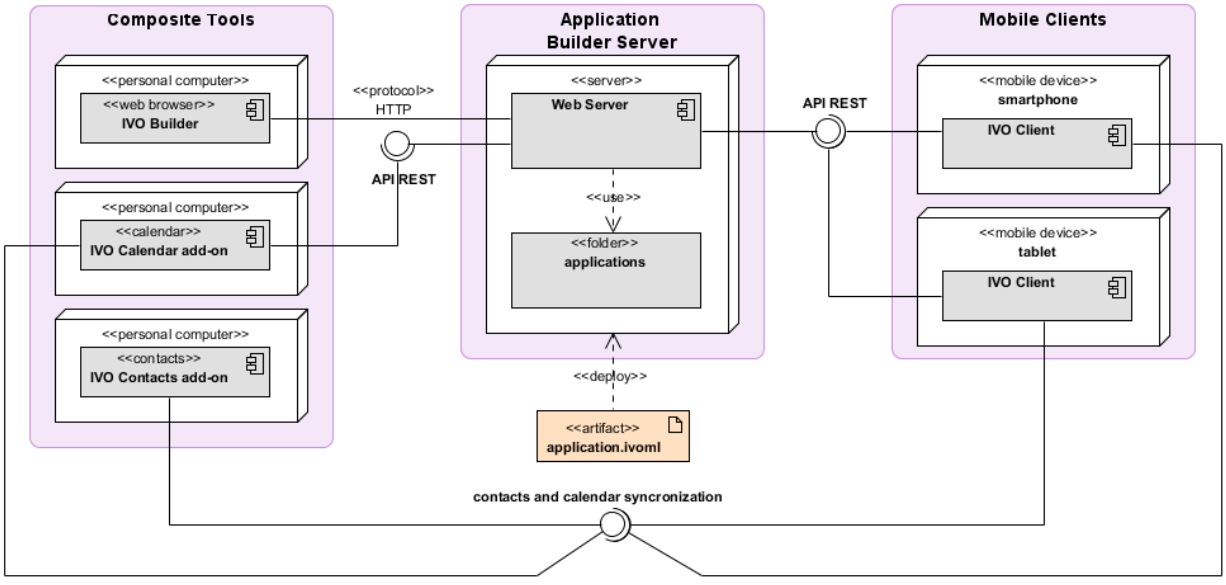


Figura 3.1. A arquitectura do IVO

3.4.1 Composite Tools

Esta arquitectura fornece três ferramentas de composição: o *IVO Builder*, o *IVO Calendar* e *o IVO Contacts*. Estas três ferramentas possibilitam a criação e disponibilização de aplicações sensíveis ao contexto por utilizadores finais sem necessidade de escrever qualquer código.

3.4.1.1 IVO Builder

O *IVO Builder* constitui a ferramenta principal de construção das aplicações e inclui funcionalidades que permitem aos utilizadores transpor facilmente os cenários idealizados por eles para os correspondentes elementos, nomeadamente as situações que indicam as condições que fazem iniciar os *workflows* representados através de um fluxo de actividades que serão executadas sempre que o dispositivo móvel se encontrar na situação especificada. Por exemplo, pode-se definir uma aplicação que mostra automaticamente informação sobre um conjunto de pontos de interesse sempre que o utilizador se aproximar das respectivas áreas circundantes e executar de igual forma e automaticamente determinado conjunto de acções tal como indicado nos cenários de uso anteriores.

3.4.1.2 IVO Calendar

O *IVO Calendar* facilita o desenvolvimento de aplicações baseadas em contextos temporais tirando partido da larga utilização das ferramentas de agenda. Ele permite que os utilizadores possam definir facilmente condições temporais, através da incorporação de *workflows* nos compromissos da agenda do utilizador, sendo estes executados "*quando o compromisso começa*" ou "*quando o compromisso termina*" tal como ilustrado anteriormente no cenário 5. O *IVO Calendar* é um *add-on* que adiciona uma região na parte inferior da janela dos compromissos que contém a interface que permite aos utilizadores definir os *workflows* com as mesmas características e actividades que estão disponíveis no *IVO Builder*. O *IVO Calendar* também pode associar contextos de localização aos compromissos para permitir a criação de regras do tipo "*neste instante no tempo e quando estiver neste local*". Estes contextos de localização podem ser definidos através de um mapa fornecido pelo *IVO Calendar* ou podem ser importados a partir das aplicações criadas com o *IVO Builder* através de uma API REST.

3.4.1.3 IVO Contacts

O *IVO Contacts* permite a utilização de contextos de proximidade associando um dispositivo *Bluetooth* (o telefone) a um dos contactos do utilizador. Tal como o *IVO Calendar*, também o *IVO Contacts* é um *add-on* que adiciona uma região na parte inferior da janela dos contactos que contém a interface que possibilita a associação do dispositivo *Bluetooth* ao contacto. Esta informação pode ser utilizada não só no *IVO Builder* como também no *IVO Calendar*, possibilitando definir contextos como "*neste instante no tempo, quando estiver neste local e quando estiver na presença desta(s) pessoa(s)".* As ferramentas de sincronização fornecidas pelo fabricante do dispositivo móvel garantem a sincronização quer da agenda quer dos contactos com este.

3.4.2 Application Builder Server

O *Application Builder Server* fornece as páginas *web* usadas pelo *IVO Builder* e armazena as aplicações criadas podendo estas ser disponibilizadas aos utilizadores através de uma API REST de acordo com as permissões de acesso.

3.4.3 Mobile Clients

Os clientes móveis (*Mobile Clients*) funcionam como dispositivos de interacção ubíqua e correm em *smartphones* ou *tablets* as aplicações criadas com as ferramentas de composição disponibilizando um motor de *workflow* para permitir a coordenação do fluxo de actividades como um processo de *workflow*.

3.5 Requisitos

Os cenários de uso atrás descritos sugerem uma utilização da plataforma, na criação de aplicações para diversos fins desde uma área mais de lazer e entretenimento (cenários 1 e 2) até a um uso mais profissional ou empresarial (cenários 3 e 4).

Do ponto de vista de restrições de acesso, os cenários indicam um uso público no caso dos cenários 1 e 2, corporativo com apenas alguns utilizadores a terem acesso no caso dos cenários 3 e 4, e pessoal no caso do cenário 5. O cenário 5 ilustra um outro tipo de utilização em que o utilizador usa a agenda electrónica para marcar compromissos acrescentando-lhes uma semântica que permite a execução automática de acções quando o compromisso começar e quando terminar.

Para além das ferramentas de construção que permitem aos utilizadores finais criar aplicações sensíveis ao contexto, a plataforma deve disponibilizar um *runtime* que permita correr estas aplicações em dispositivos móveis *smartphones* e *tablets* suportando os sistemas operativos actualmente mais usados, *Android*, *iOS* e *Windows Phone*.

As ferramentas de construção devem fornecer um conjunto de ambientes gráficos que permitam a criação dos elementos definidos na abordagem de desenho baseado em cenários apresentados na Secção 3.1.2. Os *workflows* devem permitir a execução de um conjunto variado de actividades desde as mais habituais num telemóvel, como mudar o perfil do telefone, enviar SMS, enviar *email*, tirar fotografias ou integração com as redes sociais, até actividades de interface com o utilizador mais avançadas, como sejam a utilização de texto falado, reconhecimento de voz, formulários e *quizzes*.

O conjunto completo de requisitos de cada um dos componentes é descrito de seguida.

3.5.1 Composite Tools e Application Builder Server

Nesta secção são apresentados os requisitos das ferramentas de composição (*Composite Tools*) e do *Application Builder Server* uma vez que os dois componentes estão intimamente ligados.

3.5.1.1 Requisitos Funcionais

Definição de Cenários. O sistema deve fornecer uma interface que permita facilmente transpor os cenários definidos pelo utilizador para o sistema. Um cenário representa uma determinada situação (o contexto) e o comportamento do sistema quando essa situação ocorrer (o *workflow*).

Especificação do Contexto. O sistema deve fornecer interfaces que permitam definir facilmente os contextos podendo estes ser simples ou compostos de acordo com as dimensões de localização (*where*), de tempo (*when*), de proximidade (*proximity* ou *who*) e ainda de estado (*state*):

- Os contextos de localização devem ser visualizados e criados através de um mapa e podem corresponder a duas situações distintas: entrada ou aproximação do local e a saída ou afastamento do local. O mapa deve permitir os modos de visualização habituais de rua, satélite e híbrido. Deve ser possível deslocar e fazer *zoom* ao mapa e ser possível localizar ruas através de uma funcionalidade de pesquisa sobre o mapa. Deve ser possível criar contextos de localização como um único ponto, como um polígono que limita a área envolvente do ponto e ainda importar áreas a partir de ficheiros KML [89];

- Os contextos temporais devem ser criados através de uma interface apropriada que permita uma grande expressividade das diversas situações temporais como, por exemplo, horários de verão e de inverno, dias úteis e fins-de-semana, feriados ou excepções aos horários normais;

- Os contextos de proximidade de pessoas ou equipamentos devem funcionar através da detecção de dispositivos *Bluetooth* quer por indicação do respectivo *mac address*, quer por indicação do nome do utilizador. Neste último caso, o registo do contacto do telefone deve ter a informação do *mac address* do telefone do utilizador o qual é introduzido através do *IVO Contacts*;

- Os contextos de estado devem ser criados através de um editor de regras que permita a criação de condições usando variáveis de estado que definem o estado do telefone, tal como a luminosidade ou o estado da bateria.

Informação Contextual. O sistema deve fornecer um editor HTML que permita a criação da informação contextual associada ao cenário podendo esta ser estática ou dinâmica e ter a

capacidade de adaptação ao contexto. Esta informação deve ser categorizada e estruturada de forma a facilitar a sua visualização e utilização nos dispositivos móveis. Deve, por exemplo, ser possível definir contactos, horários ou as facilidades existentes nas proximidades de um local.

Desenho de Formulários. Deve ser possível criar formulários através de uma interface visual WYSIWYG. Os formulários devem ser compatíveis com o XForms [93] devendo suportar a introdução dos tipos de dados mais comuns e poder incluir ainda campos do tipo imagem e de vídeo. Os campos dos formulários poderão ser inicializados quer com valores pré-definidos quer com os valores actuais das variáveis de contexto (adaptação ao contexto). Os dados preenchidos nos formulários poderão ser guardados quer no *Application Builder Server* quer noutro repositório qualquer através da execução de um *Web Service* REST que terá de ser desenvolvido caso a caso tal como ilustrado no cenário 4. Para o caso dos dados guardados no *Application Builder Server,* dever existir uma funcionalidade que permita visualizar e exportar os dados do formulário para posterior tratamento.

Desenho de Quizzes. Deve ser possível criar *quizzes* através de uma interface visual WYSIWYG. Os *quizzes* devem ser totalmente configuráveis quer em termos de número de questões, quer em termos de possibilidades de respostas, quer ainda em termos de comportamento como, por exemplo, permitir ao utilizador desistir de responder, possibilidade de dar *feedback* em cada questão ou definir se queremos os *quizzes* falados através de tecnologia *Text-To-Speech*.

Desenho de Workflows. O sistema deve fornecer um editor visual que permita desenhar os *workflows* que devem ser executados quando a situação ou contexto se verificar. A ferramenta deve permitir a criação de diversos tipos de controlo de fluxos, tais como condições, ciclos e paralelismo com sincronismos *join-and* e *join-or* (ver Secção 4.7.1).

Disponibilização de Aplicações. O sistema deve fornecer uma interface que permita disponibilizar aos utilizadores as aplicações criadas de acordo com as permissões de acesso pretendidas. Deve existir um mecanismo de versões que permita, nos dispositivos móveis, saber se existem novas aplicações ou actualizações das já descarregadas.

Registo e Acesso de Utilizadores. O sistema deve permitir o registo de novos utilizadores e só permitir o acesso aos utilizadores desde que devidamente autenticados através de um nome de utilizador e de uma senha. Deve ainda permitir a recuperação da senha de acesso através do fornecimento do *email* do utilizador e possibilitar a alteração dos dados da conta.

3.5.1.2 Requisitos não Funcionais

Desempenho. A aplicação deve responder rapidamente às solicitações dos utilizadores.

Capacidade de Manutenção. A disponibilização de uma actualização da aplicação deverá ser possível enquanto os utilizadores móveis se encontrem a executar as mesmas. Após a disponibilização da nova versão da aplicação os utilizadores poderão actualizar a aplicação através de mecanismos de sincronização adequados.

Segurança. Os dados pessoais das contas dos utilizadores devem ser mantidos em segurança.

Interoperabilidade. As representações de informação utilizadas e os protocolos de comunicação devem basear-se nos *standards* suportados nas várias plataformas móveis.

Escalabilidade. A plataforma deverá suportar um grande número de utilizadores em simultâneo a aceder ao sistema quer a desenvolver quer a descarregar ou sincronizar aplicações.

Usabilidade. Deve ser fácil aos utilizadores usar a ferramenta. Os menus e funções devem ser claros, fáceis de aceder, auto explicativos e bem documentados.

3.5.2 Mobile Clients

Como visto nos cenários de uso, as aplicações desenvolvidas com as ferramentas de composição poderão correr em dispositivos móveis do tipo *smartphone* e *tablet* nas plataformas mais usadas como *Android, iOS* e *Windows Phone*. Para isso, deve ser disponibilizado um *runtime* cujos requisitos funcionais e não funcionais são descritos de seguida.

3.5.2.1 Requisitos Funcionais

Preferências do utilizador. O sistema deve permitir ao utilizador seleccionar as acções (correspondentes às actividades de *workflow*) que pretende serem executadas automaticamente. Por exemplo, um utilizador pode pretender que não seja actualizado o estado do seu *Facebook* enquanto outro pode desejar que isso aconteça (veja-se o cenário de uso 2).

Informação de contexto. Deve ser possível visualizar a informação associada aos contextos. Se existir informação de número de telefone, deve ser possível a marcação directa através de um botão. Se existir informação de *email*, deve ser possível o envio directo de uma mensagem de *email* através de um botão.

Contextos de localização. No caso da existência de contextos de localização na aplicação, o sistema deve:

- Permitir a visualização destes contextos num mapa. O mapa deve permitir os modos de visualização habituais de rua, de satélite e híbrido. Deve ser possível deslocar e fazer *zoom* ao mapa. Deve ser possível visualizar apenas os contextos de localização pertencentes às categorias seleccionadas. Deve ser possível visualizar no mapa a localização actual do utilizador.

- Possibilitar a visualização de fotografias do *Panoramio* geo-referenciadas nas imediações (o que se vê aqui perto).

- Possibilitar o acesso rápido, através de botões, ao navegador com indicações de direcção a pé e de carro desde o local onde o utilizador se encontra até ao local definido no contexto de localização.

- Permitir a visualização de realidade aumentada que sobreponha à imagem capturada pela câmara do dispositivo móvel os contextos de localização. Deve ser possível definir um raio que indique o alcance máximo para a visualização desses contextos.

Pesquisa de informação. Deve ser possível a pesquisa rápida de qualquer tipo de contextos, como por exemplo que locais posso visitar na proximidade do local onde me encontro ou que museus posso visitar neste momento.

Execução de formulários. O sistema deve fornecer um processador de XForms que permita a execução dos formulários criados com a ferramenta de criação de formulários do *IVO Builder*.

Execução de quizzes. O sistema deve peritir a execução dos *quizzes* criados com a ferramenta de criação de *quizzes* do *IVO Builder*.

Partilha de informação. O sistema deve permitir a partilha de informação através dos métodos habituais disponíveis nos dispositivos móveis como redes sociais, *email*, *Bluetooth*, *Dropbox*, entre outros. Por exemplo, o sistema deve possibilitar a partilha da informação de contexto ou a imagem aumentada visualizada no écran através da funcionalidade de realidade aumentada.

Leitura de códigos IVO e NFC. Deve ser possível a leitura de códigos de barra 2D (QR-Codes) e *tags* NFC que fornecerão informação contextualizada. A leitura de NFC só estará disponível nos dispositivos móveis que suportarem esta tecnologia.

Carregar aplicação. O sistema deve permitir ao utilizador escolher a aplicação que deseja executar.

Sincronizar aplicações. Deve ser providenciado um mecanismo que permita descarregar e actualizar as aplicações criadas e disponibilizadas para o utilizador.

3.5.2.2 Requisitos não Funcionais

Monitorização de Sensores. O sistema deve ser capaz de gerar eventos a partir de alterações de contexto resultantes de sensores internos ao dispositivo móvel, bem como de sensores externos com os quais o dispositivo móvel possa comunicar. Deve fornecer uma abstração dos sensores de forma a garantir uma fácil extensabilidade dos sensores suportados. Considera-se que devem ser suportados os seguintes sensores base:

1. Localização por GPS e por triangulação de redes Wi-Fi e GSM;
2. *Bluetooth* para detecção de equipamentos na proximidade do utilizador;
3. Leitura de *tags* NFC (*Near Field Communication*) para os dispositivos que suportarem;
4. Leitura de códigos 2D (QR Code ou *Quick Response Code*);
5. Calendário;
6. Sensores externos.

Gestão de Contexto. A informação de contexto proveniente dos eventos gerados pela monitorização dos sensores deve ser utilizada na avaliação das condições que determinam inícios e transições do *workflow*. O sistema deve fornecer um mecanismo *register/unregister* de condições de forma a possibilitar a avaliação das condições activas e a dar resposta às alterações do contexto. As validações das condições poderão dar origem a:

1. Iniciar um novo *workflow*, porque a sua condição de início foi verificada;
2. Uma transição num *workflow*, porque a sua condição de guarda foi verificada;
3. Tratamento adequado de excepções relativas a estados indefinidos do contexto.

Motor de Workflow. O sistema deve fornecer um motor de *workflow* baseado em eventos com suporte ao conceito de condições de guarda. As actividades de *workflow* disponíveis devem possibilitar a execução das tarefas mais habituais num telemóvel, como por exemplo mudar perfil, enviar um SMS, enviar um *email*, marcar um número ou tirar uma foto. Deve fornecer uma abstração das actividades de *workflow* suportadas de forma a garantir uma fácil extensabilidade das actividades suportadas.

Desempenho. A aplicação deve responder rapidamente às alterações de contexto e às interacções dos utilizadores.

Capacidade de Manutenção. A alteração de uma aplicação deve ser possível enquanto os utilizadores se encontrem a executar as mesmas. Após a disponibilização da nova versão da

aplicação os utilizadores poderão actualizar a aplicação através de mecanismos de sincronização adequados.

Segurança. Os dados pessoais do utilizador, dados registados e outra informação sensível devem ser mantidos em segurança. Ao armazenar dados num sistema diferente do dispositivo móvel, estes devem ser tornados anónimos, ou seja, a remoção de todas as informações pessoais a partir dos dados.

Interoperabilidade. As representações de informação utilizadas e os protocolos de comunicação devem basear-se nos *standards* suportados nas várias plataformas móveis.

Usabilidade. Deve ser fácil aos utilizadores instalar e usar as aplicações. Os menus e funções devem ser claros, fáceis de aceder, auto explicativos e bem documentados.

Acessibilidade. Devem ser tidas em consideração as recomendações e boas práticas no que diz respeito à acessibilidade por parte de cidadãos com necessidades especiais.

Requisitos de Mobilidade. O sistema deve suportar comunicações confiáveis, em *offline* e assíncronas. O utilizador poderá seleccionar os elementos de dados a serem transferidos para o dispositivo móvel e deve ser garantida a independência do dispositivo.

3.6 Conclusão

Neste capítulo apresentou-se a *framework* do IVO a qual permite aos utilizadores finais (sem conhecimentos de programação) criar e utilizar aplicações sensíveis ao contexto usando como dispositivo ubíquo *smartphones* e *tablets*.

Com base nos princípios de desenho e no conjunto de cenários de uso apresentados foi definida a arquitectura de referência, a qual é constituída por três componentes fundamentais: (i) *Composite Tools*; (ii) *Application Builder Server*; e (iii) *Mobile Clients*. Estes três componentes fornecem ao utilizador as ferramentas e mecanismos que permitem criar, partilhar e utilizar as aplicações criadas com a plataforma.

Os requisitos identificados para cada um dos componentes, bem como os princípios de desenho, serviram como ponto de partida, nos capítulos seguintes, para: (i) definição da linguagem de marcação que possibilita a representação de todos os elementos que constituem uma aplicação IVO; (ii) apresentação de uma metodologia de desenvolvimento de uma aplicação através da plataforma; e (iii) implementação da plataforma a qual permitirá efectuar uma avaliação da mesma.

Capítulo 4

IVO Markup Language

Conteúdo

Resumo

Neste capítulo é feito um resumo da IVOML (*IVO Markup Language*), que é a linguagem de programação baseada em XML utilizada pelo IVO. Uma aplicação IVO deve ser um documento IVOML válido e codificado em UTF-8[15]. As ferramentas de composição descritas no capítulo anterior geram ficheiros IVOML de acordo com o formato descrito de seguida. O formato completo pode ser consultado através do respectivo XSD[16] (*XML Schema Definition*) em http://www.integratedvirtualoperator.com/ivoml/1.0/ivoml10.xsd e a documentação relativa a este esquema em http://www.integratedvirtualoperator.com/ivoml/1.0/doc.

[15] UTF-8 (8-bit Unicode Transformation Format). É um tipo de codificação Unicode de comprimento variável.

[16] XSD (XML Schema Definition). O XSD descreve a estrutura de um documento XML.

4.1 Esquema Geral de Uma Aplicação

A Figura 4.1 representa o esquema do elemento `ivoml` que é o elemento raiz da IVOML.

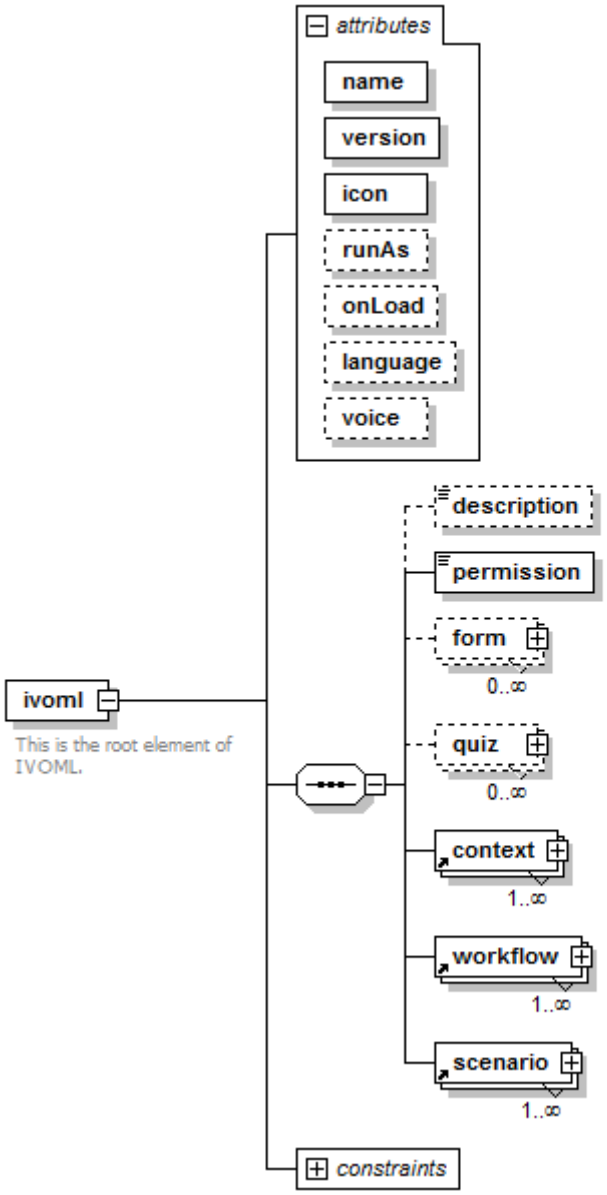


Figura 4.1. Esquema geral de uma aplicação

Os atributos deste elemento permitem caracterizar globalmente a aplicação, sendo descritos de seguida.

`name`	Nome da aplicação.
`version`	Versão da aplicação a qual é utilizada para determinar alterações nas aplicações e fazer a sincronização dos clientes.
`icon`	Permite definir o icon da aplicação.
`runas`	Permite definir o modo de funcionamento da aplicação e pode tomar o valor (i) `service` que permite colocar a aplicação a correr em *background* como um serviço; ou (ii) `application` que permite correr a aplicação como uma aplicação normal. Este atributo é opcional sendo o seu valor por omissão `application`.

onload Atributo opcional que permite definir o *workflow* que é iniciado quando a aplicação é carregada.

language Atributo opcional que indica a linguagem utilizada nos diversos conteúdos da aplicação. Corresponde ao código internacional de país e no caso de nada estar definido é utilizado o valor definido nas definições do dispositivo móvel.

voice Atributo opcional que indica a linguagem utilizada nos conteúdos falados da aplicação (*text-to-speech*) e nas actividades de reconhecimento de voz. Corresponde ao código internacional de país e no caso de nada estar definido é utilizado o valor definido nas definições do dispositivo móvel.

As restrições definidas através das *constraints*, que no diagrama da figura anterior se encontram escondidas, estabelecem as seguintes regras de validação entre os diversos objectos.

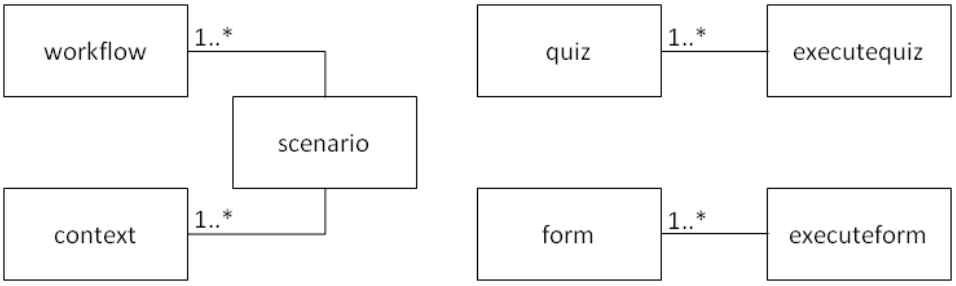


Figura 4.2. Relações estabelecidas pelas constraints

No esquema da Figura 4.2, podem-se observar os principais objectos que constituem uma aplicação IVO. Os contextos são o elemento principal e podem existir num ou mais cenários de utilização. Os cenários permitem definir qual o comportamento da aplicação (ou seja qual o *workflow*), quando determinado contexto se verifica. Desta forma, é possível utilizar o mesmo contexto ou o mesmo *workflow*, em vários cenários de utilização. Os formulários e os *quizzes* são utilizados nas actividades de *workflow* que permitem executar quer um, quer outro. O mesmo formulário ou *quiz*, pode ser utilizado em um ou mais *workflows*.

De seguida irão ser descritos cada um dos elementos que constituem uma aplicação IVO.

4.2 Descrição da Aplicação

Elemento opcional que contém uma descrição da aplicação. Pode conter HTML, devendo estar numa secção `CDATA`[17] de modo a preservar o seu conteúdo que de outra forma seria interpretado pelos *parsers* de XML. Esta informação é usada pelos clientes IVO após o carregamento da aplicação para fornecer ao utilizador informação acerca da mesma. A Figura 4.3 ilustra um exemplo do ecrã de uma aplicação de Guia Turístico para a zona de Belém em Lisboa,

[17] `XML CDATA` - http://www.w3schools.com/xml/xml_cdata.asp

o qual contém informação relevante para o turista que vai iniciar a utilização do sistema como guia da zona.

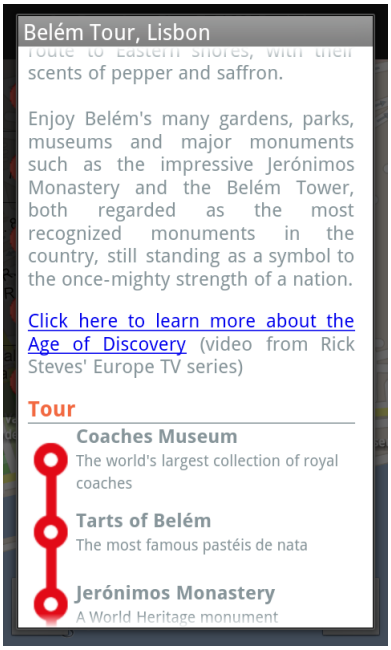


Figura 4.3. Visualização da descrição de uma aplicação num cliente IVO

4.3 Definição de Permissões

Elemento que define as permissões da aplicação, as quais correspondem a uma lista dos seguintes valores:

`Internet`	A aplicação necessita de uma ligação à internet.
`GPS`	A aplicação usa localização por GPS.
`Network`	A aplicação usa localização de rede através da triangulação das redes GSM/3G e Wi-Fi.
`Bluetooth`	A aplicação usa serviços de *Bluetooth*.
`AR`	A aplicação usa *browser* de Realidade Aumentada.
`Calendar`	Aplicação usa o calendário para a definição de contextos de tempo e/ou de proximidade.

O exemplo seguinte ilustra uma aplicação que necessita usar uma ligação à internet, localizações por GPS e Realidade Aumentada.

```
1  <permission>Internet GPS AR</permission>
```

Figura 4.4. Exemplo de definição permissões de uma aplicação

4.4 Definição de Formulários

Uma aplicação IVO pode conter vários formulários, sendo cada um deles definido no elemento `form` do elemento raiz. Os formulários do IVO são compatíveis com o XForms que utiliza uma

abordagem MVC (*Model-View-Controller*), a qual se traduz numa separação clara entre a descrição dos dados utilizados pelo formulário (*Model*), a interface com o utilizador (*View*) e as regras de negócio que definem as validações dos dados fornecidos (*Controller*). A interface é descrita de forma abstracta, cabendo ao interpretador definir a forma como cada elemento será apresentado ao utilizador, dependendo da plataforma utilizada. Como se pode ver na Figura 4.5, a definição dos formulários é feita embebendo na IVOML os elementos `model` e `group` do XForms, os quais permitem definir o modelo de dados, as regras de negócio e a interface.

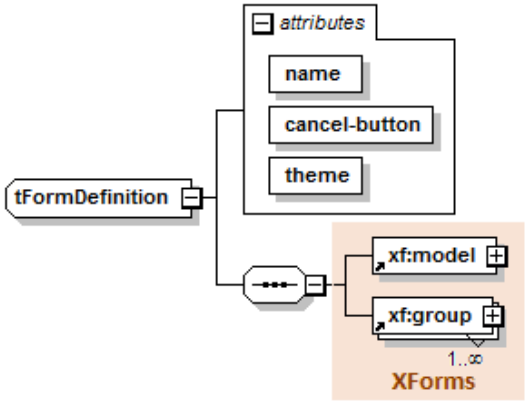


Figura 4.5. Esquema dos formulários

Na Figura 4.6 está ilustrado um exemplo de um formulário simples relativo à recolha de informação de um pagamento. O modelo de dados está descrito entre as linhas 3 e 10 através do elemento `xf:instance`, enquanto a lógica de negócio que define as validações está representada entre as linhas 11 e 14. Os elementos `xf:bind` definem o tipo de dados, bem como a obrigatoriedade de preencher todos os campos. A apresentação, na forma de controlos e as suas respectivas etiquetas, encontra-se entre as linhas 16 e 39.

```
1  <form name="Pagamento" cancel-button="true" theme="light">
2     <xf:model xmlns:xf="http://www.w3.org/2002/xforms">
3        <xf:instance id="pagamento-instance">
4           <pagamento>
5              <valor/>
6              <formaPagamento/>
7              <cartao/>
8              <dataExpiracao/>
9           </pagamento>
10       </xf:instance>
11       <xf:bind nodeset="valor" type="xs:decimal" required="true()"/>
12       <xf:bind nodeset="formaPagamento" type="xs:select1" required="true()"/>
13       <xf:bind nodeset ="cartao" type="xs:string" required="true()"/>
14       <xf:bind nodeset ="dataExpiracao" type="xs:date" required="true()"/>
15    </xf:model>
16    <xf:group xmlns:xf="http://www.w3.org/2002/xforms">
17       <xf:label>Pagamento</xf:label>
18       <xf:input ref="valor">
19          <xf:label>Valor</xf:label>
20       </xf:input>
21       <xf:select1 ref="formaPagamento">
22          <xf:label>Forma de Pagamento</xf:label>
23          <xf:item>
24             <xf:label>Visa</xf:label><xf:value>VISA</xf:value>
25          </xf:item>
```

```
26          <xf:item>
27             <xf:label>Master Card</xf:label><xf:value>MC</xf:value>
28          </xf:item>
29          <xf:item>
30             <xf:label>American Express</xf:label><xf:value>AE</xf:value>
31          </xf:item>
32       </xf:select1>
33       <xf:input ref="cartao">
34          <xf:label>Cartão</xf:label>
35       </xf:input>
36       <xf:input ref="dataExpiracao">
37          <xf:label>Data Expiração</xf:label>
38       </xf:input>
39    </xf:group>
40 </form>
```

Figura 4.6. Exemplo de um formulário

A Figura 4.7 ilustra este mesmo formulário a correr num cliente IVO Android. Neste exemplo foi definido que o utilizador pode cancelar o preenchimento do formulário (`cancel-button="true"`) e foi usado um tema claro (`theme="light"`). Os controlos para introdução de dados permitem apenas a introdução do tipo de dados definido, efectuando o sistema todas as validações que forem definidas e que, neste caso, são apenas a obrigatoriedade de preenchimento dos campos (`required="true()"`).

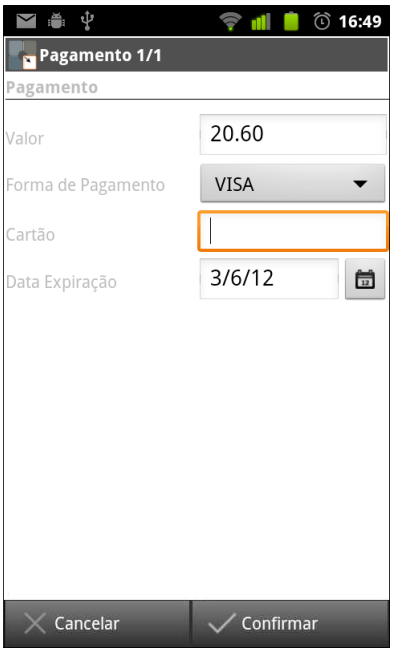


Figura 4.7. Apresentação do formulário num cliente IVO

4.5 Definição de Quizzes

À semelhança dos formulários, uma aplicação IVO pode conter vários quizzes definidos através do elemento `quiz` cujo esquema se encontra na Figura 4.8. Um `quiz` pode ter várias perguntas cada uma com várias respostas possíveis das quais apenas uma é a correcta. Os atributos de um `quiz` permitem configurar o comportamento do mesmo e podem ser:

name	Nome do `quiz`.
cancel-button	Atributo opcional que indica se o utilizador pode desistir de fazer o `quiz` sem o terminar. O valor por omissão é `true`.
min-score	Atributo opcional que indica a percentagem mínima de respostas correctas para que o `quiz` seja considerado ultrapassado. O valor por omissão é 0%.
show-correct-answer	Atributo opcional que permite definir se deve ser fornecido *feedback* após cada resposta. O valor por omissão é `true`.
onPass-text	Atributo opcional que indica a mensagem a mostrar ao utilizador após este conseguir acertar a pelo menos o valor definido em `min-score`.
onFail-text	Atributo opcional que indica a mensagem a mostrar ao utilizador caso este não consiga acertar a pelo menos o valor definido em `min-score`.
use-tts	Atributo opcional que indica se o `quiz` deve ser falado. O valor por omissão é `true`.
theme	Atributo opcional que indica o tema a usar no `quiz` o qual pode ser `dark` ou `light`. O valor por omissão é `light`.
share-result	Atributo booleano opcional que indica se o resultado alcançado no `quiz` pode ser partilhado nas redes sociais. O valor por omissão é `true`. Em caso afirmativo e no final do `quiz`, surge disponível ao utilizador, um botão que lhe permite partilhar o resultado.

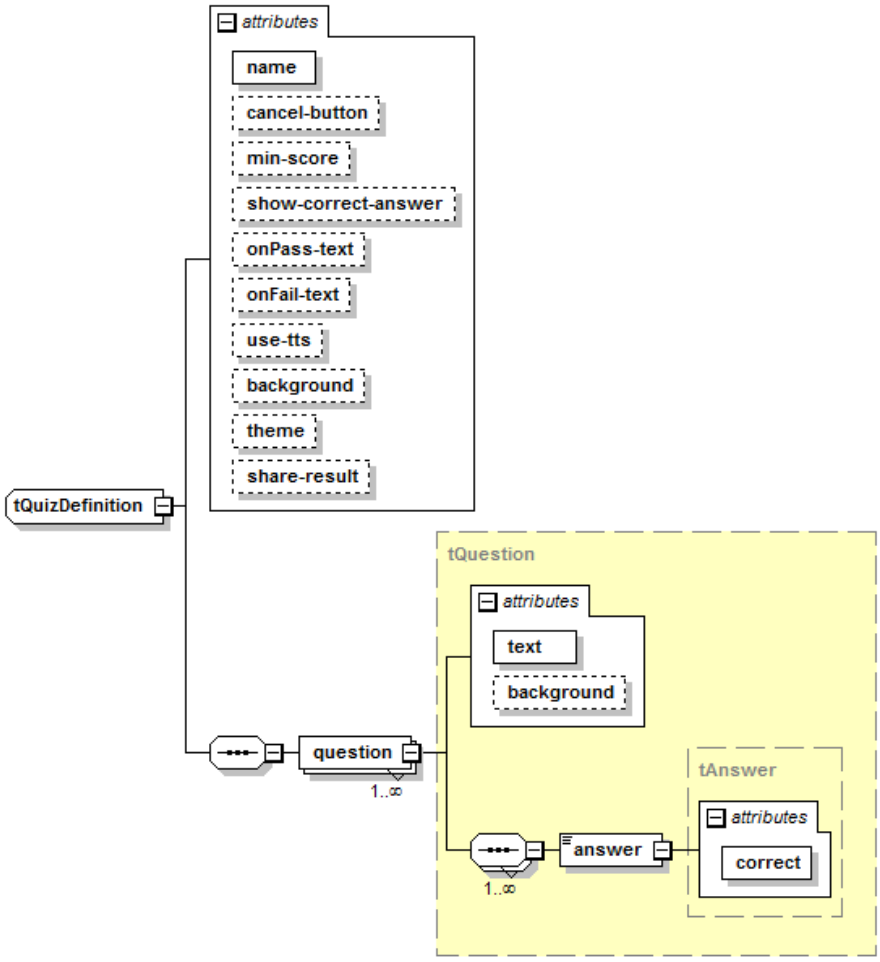


Figura 4.8. Esquema de um quiz

Na Figura 4.9 está ilustrado um exemplo de um `quiz` com duas perguntas relativas ao Mosteiro dos Jerónimos, enquanto a Figura 4.10 ilustra este mesmo `quiz` a correr num cliente IVO Android.

```
<quiz name="Mosteiro dos Jerónimos" min-score="0.5" share-result="true"
      background="Jeronimos.png" onPass-text="Dirija-se agora para o próximo
ponto do roteiro - o Padrão dos Descobrimentos." >
   <question text="Qual o estilo representativo do Mosteiro dos Jerónimos?"
             background="jeronimos.png" >
      <answer correct="false">Gótico</answer>
      <answer correct="false">Barroco</answer>
      <answer correct="true">Manuelino</answer>
      <answer correct="false">Renascentista</answer>
   </question>
   <question text="Quem mandou construir o Mosteiro dos Jerónimos?"
             background="jeronimos.png" >
      <answer correct="false">Vasco da Gama</answer>
      <answer correct="true">D. Manuel I</answer>
      <answer correct="false">D. João III</answer>
      <answer correct="false">D. Sebastião</answer>
   </question>
</quiz>
```

Figura 4.9. Exemplo de um quiz

Figura 4.10. O quiz anterior a correr num cliente IVO Android

4.6 Definição de Contextos

A definição de contextos é feita através do elemento `context`, cujo esquema está ilustrado na Figura 4.11.

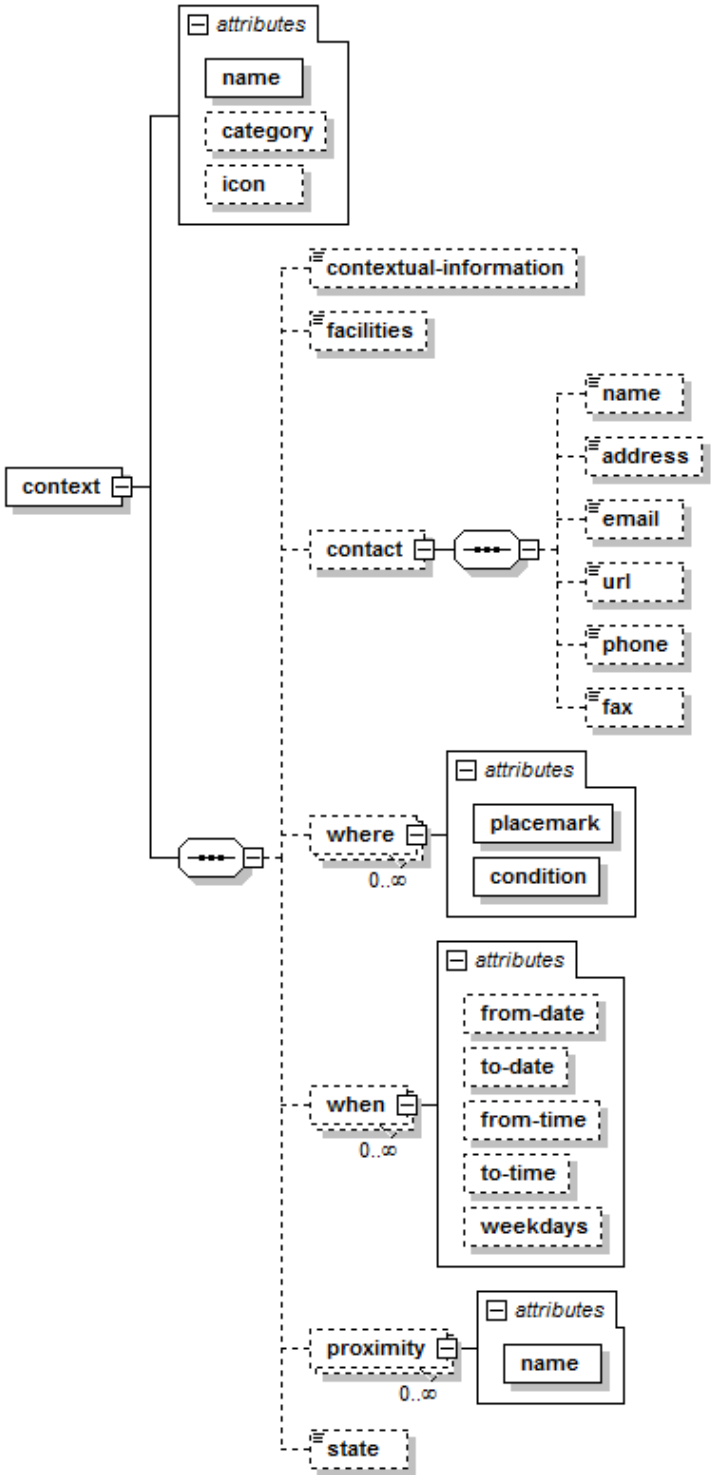


Figura 4.11. Representação de contextos

Cada contexto é identificado por um nome definido no atributo `name` e por uma categoria opcional definida no atributo `category`, podendo conter vários contextos de localização representados por elementos `where`, vários contextos temporais representados por elementos `when`, vários contextos de proximidade representados por elementos `proximity` e um contexto de estado representado pelo elemento `state`. Na definição de um contexto deverá existir pelo menos um destes elementos.

Na Figura 4.12 está ilustrado um exemplo de contexto que utiliza dois contextos temporais. O primeiro refere-se ao horário de verão do Mosteiro dos Jerónimos e o segundo ao horário de inverno.

```
1  <context name="Entrada no Mosteiro dos Jerónimos" category="Monumentos">
2     <where placemark="Mosteiro dos Jerónimos" condition="onEnter"/>
3
4     <when  from-date="05-01" to-date="09-30"
5           from-time="10:00:00" to-time="18:00:00"
6           weekdays="Sun Tue Wed Thu Fri Sat"/>
7     <when from-date="10-01" to-date="04-30"
8           from-time="10:00:00" to-time="17:00:00"
9           weekdays="Sun Tue Wed Thu Fri Sat"/>
10
11    <state>
12       <![CDATA[
13          NOT (([STATE.MONTH] == 1 AND [STATE.DAY] == 1)        // Excepto 1-Jan
14          OR ([STATE.MONTH] == 5 AND [STATE.DAY] == 1)          // e 1-Mai
15          OR ([STATE.MONTH] == 12 AND [STATE.DAY] == 25))       // e 25-Dez
16       ]]>
17    </state>
</context>
```

Figura 4.12. Exemplo de contexto

São utilizados os atributos `from-date` e `to-date` (linhas 4 e 7) para especificar os intervalos (em meses) e os atributos `from-time` e `to-time` (linhas 5 e 8) para especificar as horas do dia durante as quais o Mosteiro dos Jerónimos está aberto. Os dias da semana são indicados através do atributo `weekdays` (linhas 6 e 9; todos os dias excepto à segunda-feira). É ainda usada uma expressão (`state`), para definir as excepções à regra normal de horário (linhas 13 a 15; excepto dia 1 de Janeiro, 1 de Maio e 25 de Dezembro). Estas expressões podem fazer uso dos operadores lógicos mais comuns e do conjunto de variáveis referidas na Tabela 6.3. A gramática utilizada nas expressões, é a suportada pela *Java Expression Language* (JEXL). Note-se nos exemplos anteriores, que as expressões utilizadas nos contextos estão numa secção `CDATA` de forma a preservar o seu conteúdo. A utilização do contexto de localização permite limitar a verificação dos contextos apenas quando o utilizador entrar na área definida através do atributo `placemark`. A referência para o `placemark` (linhas 2) será explicada mais à frente na Secção 4.9.

Na Figura 4.13 está representado outro exemplo de contexto que usa a variável de estado referente ao nível da bateria, de forma a ser activado quando esse nível for inferior a 10%.

```
1  <context name="Bateria Fraca">
2     <state>
3        <![CDATA[
4           [STATE.BATTERYLEVEL]<0.1
5        ]]>
6     </state>
7  </context>
```

Figura 4.13. Exemplo de contexto de estado

4.7 Definição de Workflows

Uma aplicação pode conter vários *workflows* correspondendo cada um deles ao comportamento desejado do sistema sempre que o respectivo contexto se verifique. A figura seguinte representa o esquema de um *workflow*, o qual é caracterizado pelos atributos `name` que indica o nome do *workflow*, o atributo `show-contextual-information` para indicar se deve ser mostrada a informação contextual associada ao contexto que deu origem ao *workflow* e um conjunto de actividades que se encontram tipificadas na Tabela 4.1.

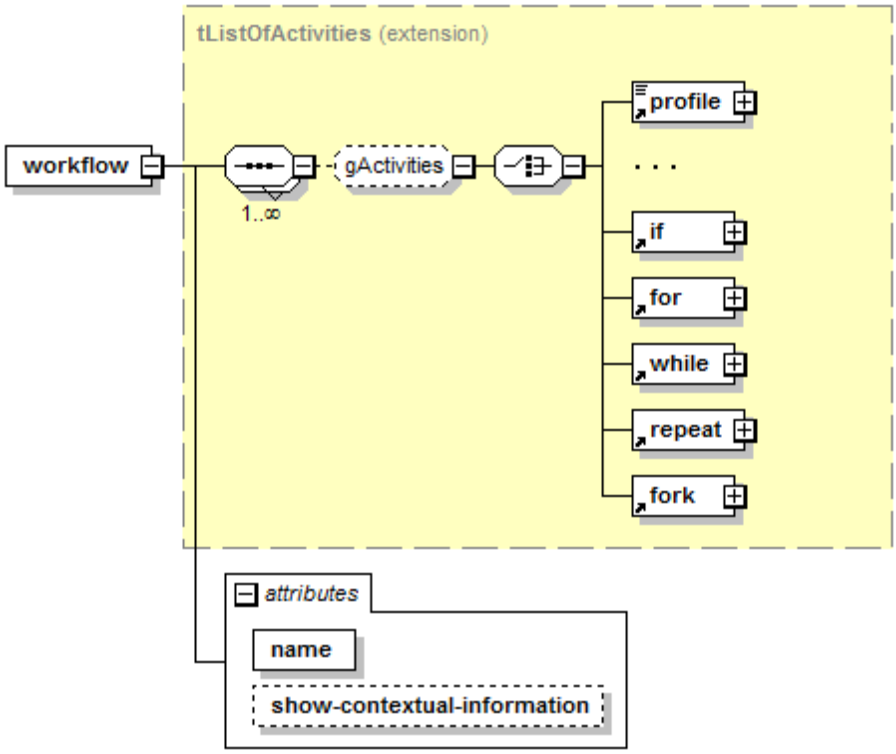


Figura 4.14. Esquema de um workflow

Na figura anterior estão omissas muitas das actividades disponíveis uma vez que o formato destas segue sempre um modelo semelhante. Na realidade a maior parte das actividades é composta por elementos vazios apenas com o conjunto de atributos que caracterizam a actividade, tal como exemplificado na figura seguinte.

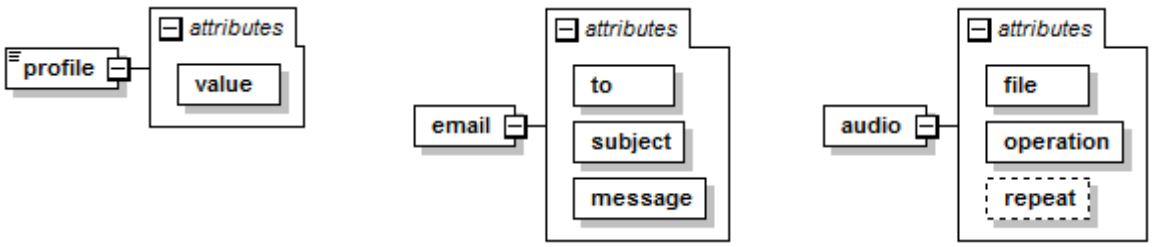


Figura 4.15. Esquema das actividades de (a) profile; (b) email; e (c) audio

Actividade/Elemento	Descrição
profile	Altera o perfil do telefone, incluindo ligar e desligar som, *Bluetooth*, wifi e gps
dial	Efectua a marcação de um número de telefone
sms	Envia um SMS para determinado número
email	Envia um *email*
alert	Mostra uma janela de alerta ao utilizador
message	Mostra uma mensagem ao utilizador
input	Permite a introdução de dados
audio	Permite o controlo de um ficheiro de áudio (play, pause, resume e stop)
video	Permite o controlo de um vídeo (play, pause, resume e stop)
photo	Lança a câmara para tirar uma fotografia
url	Abre uma página Web
webservice	Executa um *webservice*
run	Corre uma aplicação
script	Corre um script
navigateto	Lança o navegador para guiar o utilizador para determinado local
facebook	Actualiza o estado do utilizador no *Facebook*
twitter	Actualiza o estado do utilizador no *Twitter*
assign	Atribuição de valor a uma variável
executeform	Executa um formulário
executequiz	Executa um quiz
tts	Permite falar o texto fornecido
voicerecognition	Permite reconhecer voz
waitfor	Espera por determinada condição
sleep	Espera determinado tempo (em segundos)
if	Fluxo condicional
for	Ciclo `for`
while	Ciclo `while`
repeat	Ciclo `repeat`
fork	Executa blocos de actividades em paralelo

Tabela 4.1. Actividades actualmente suportadas

4.7.1 Actividades de Controlo de Fluxo

Este grupo de actividades permite controlar o fluxo do *workflow*, fornecendo mecanismos equivalentes às linguagens de programação para as instruções do tipo `if-then-else`, ciclos `for`, `while`, `repeat-until` e `fork`. Estas actividades apresentam um esquema recursivo uma vez que podem conter blocos de actividades dentro dos elementos. O tipo

`tListOfActivities`, que surge nos esquemas seguintes escondido por questões de espaço, corresponde ao mesmo tipo de elemento ilustrado no esquema do *workflow* da Figura 4.14.

4.7.1.1 Fluxo condicional - a actividade if

A Figura 4.16 ilustra o esquema da actividade `if` e a Figura 4.17 exemplifica um *workflow* que faz uso desta actividade para definir fluxos condicionais.

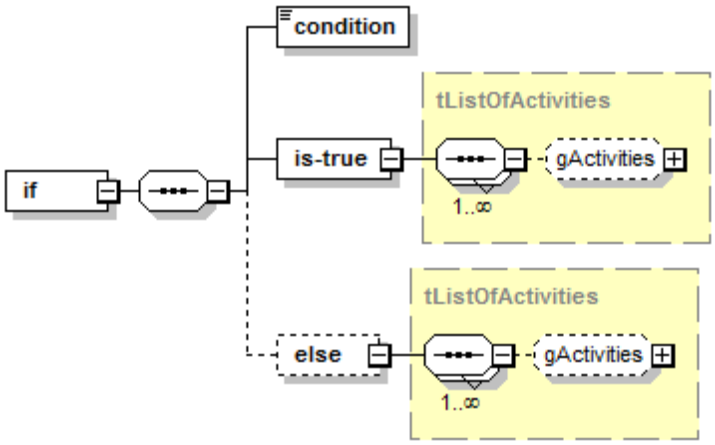


Figura 4.16. Esquema da actividade if

```
1  <workflow name="Encomenda">
2    <executeform name="Encomenda"/>
3    <if>
4      <condition>
5        <![CDATA[[Encomenda.Desconto] > 10]]>
6      </condition>
7      <is-true>
8        <email subject="Aprovar Encomenda"
9            message="Solicita-se a sua aprovação para a encomenda em anexo."
10           to="gestor@mycompany.com"/>
11     </is-true>
12     <else>
13       <webservice url="http://www.mycompany.com/encomenda.php"
14                user="xxx" password="xxx"/>
15     </else>
16   </if>
17 </workflow>
```

Figura 4.17. Exemplo de um workflow com uma condição

No exemplo anterior, é utilizado na condição do `if` a variável `Desconto` do formulário `Encomenda` (linhas 4 a 6), para determinar o fluxo que o *workflow* deve seguir. Repare-se, mais uma vez, na necessidade da condição estar numa secção `CDATA`. No caso do valor desta variável ser superior a 10, é executada a actividade constante no elemento `is-true` (linhas 7 a 11; enviar um *email*), e `else` caso contrário (linhas 12 a 15; executar um *webservice*). No exemplo fornecido, cada ramo do `if`, apenas contém uma actividade, mas pode na realidade conter um número indeterminado de qualquer uma das actividades disponíveis.

4.7.1.2 Ciclos – as actividades for, while e repeat

À semelhança das linguagens de programação, as actividades `for`, `while` e `repeat`, permitem executar um bloco de actividades um determinado número de vezes (ciclo `for`), enquanto uma dada condição não se verificar (ciclo `while`), ou ainda, até que determinada condição se verifique (ciclo `repeat`). A Figura 4.18 ilustra os esquemas destas actividades.

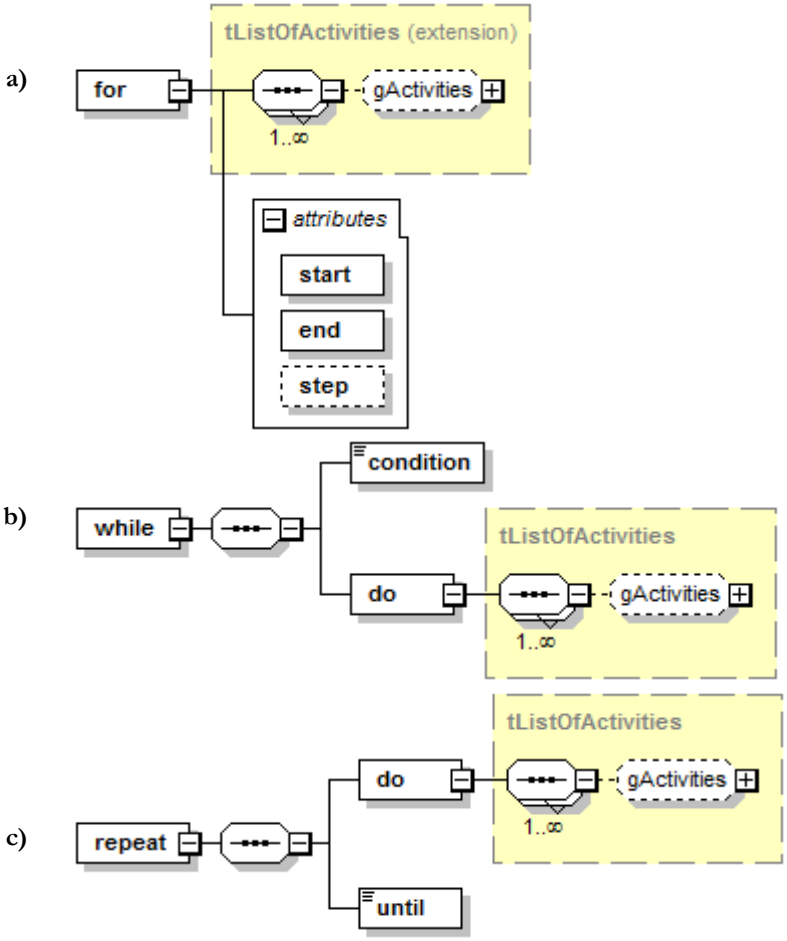


Figura 4.18. Esquema dos ciclos (a) for, (b) while e (c) repeat

A Figura 4.19 ilustra um ciclo `for` que fala três vezes o texto "Olá mundo!" (linha 2) com um intervalo de dez segundos entre elas (linha 3).

```
1 <for start="1" end="3">
2   <tts text="Olá mundo!"/>
3   <sleep time="10"/>
4 </for>
```

Figura 4.19. Exemplo de um ciclo for

O ciclo `while` testa a condição no início do ciclo enquanto o `repeat` só efectua o teste no final do ciclo executando desta forma pelo menos uma vez o ciclo. A Figura 4.20 ilustra a utilização do ciclo `repeat` de forma a obrigar o utilizador a introduzir um valor do mês entre 1 e 12. Neste exemplo, é utilizada a actividade de entrada de dados (actividade `input` na linha 3)

para criar uma variável de processo com o nome `Mes`. Esta variável é testada no final do ciclo (linhas 5 a 7) repetindo-se este, caso a variável não esteja entre 1 e 12.

```
1 <repeat>
2   <do>
3     <input prompt="Qual o mês (1-12)?" type="integer"variable="Mes"/>
4   </do>
5   <until>
6     <![CDATA[[PROCESS.Mes] >=1 AND [PROCESS.Mes] <= 12]]>
7   </until>
8 </repeat>
```

Figura 4.20. Exemplo de um ciclo repeat

4.7.1.3 Paralelismo - a actividade fork

A actividade `fork` permite executar blocos de actividades em paralelo, podendo a sincronização desses blocos ser feita através de um `join-and` em que o *workflow* continua apenas quando todos os blocos paralelos estiverem concluídos, ou, através de um `join-or` em que o *workflow* continua logo que um dos blocos tenha terminado. A Figura 4.21 ilustra o esquema do `fork`. O atributo `join` permite especificar o tipo de sincronização (`and` ou `or`) dos blocos. Cada bloco de actividades é composto por um elemento `do`.

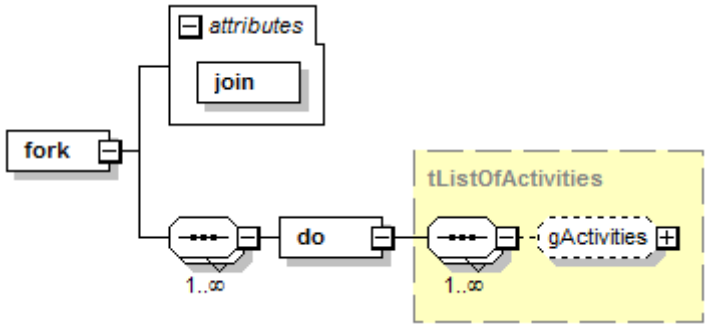


Figura 4.21. Paralelismo – a actividade fork

No exemplo seguinte, a execução do *webservice* e o envio do SMS correm em paralelo só continuando o *workflow* depois de ambas terem terminado, porque foi definido que a sincronização dos blocos deverá ser feita através de um `join-and` (linha 1).

```
1 <fork join="and">
2   <do>
3     <webservice url="http://www.mycompany.com/encomenda.php"
4           user="xxx" password="xxx"/>
5   </do>
6   <do>
7     <sms phone="xxxxxx" message="Encomenda registada."/>
8   </do>
9 </fork>
```

Figura 4.22. Exemplo de um fork

4.8 Definição de Cenários

Um cenário representa uma determinada situação e o comportamento do sistema quando essa situação ocorrer. A Figura 4.23 ilustra o esquema de um cenário e a Figura 4.24 mostra dois cenários referentes à entrada do utilizador no Mosteiro dos Jerónimos no horário de verão (linhas 1 a 3) e no horário de inverno (linhas 5 a 7).

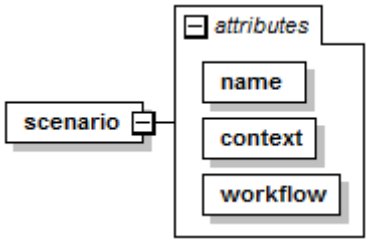


Figura 4.23. Esquema de um cenário

```
1  <scenario name="Visita ao Mosteiro dos Jerónimos - Verão"
2        context="Entrada no Mosteiro dos Jerónimos - Horário de Verão"
3        workflow="Entrada no Mosteiro dos Jerónimos"/>
4
5  <scenario name="Visita ao Mosteiro dos Jerónimos - Inverno"
6        context="Entrada no Mosteiro dos Jerónimos - Horário de Inverno"
7        workflow="Entrada no Mosteiro dos Jerónimos"/>
```

Figura 4.24. Exemplos de dois cenários com o mesmo comportamento

Neste exemplo, o comportamento da aplicação deverá ser o mesmo nas duas situações traduzindo-se na execução do *workflow* identificado como "Entrada no Mosteiro dos Jerónimos" (linhas 3 e 7).

4.9 IVOML embebida em KML

A IVOML pode ser embebida em KML [94] permitindo usar este formato de anotação de dados geográficos e de visualização. Desde 2008 que a KML se tornou um *standard* da Open Geospatial Consortium sendo suportado pela maior parte dos Sistemas de Informação Geográfica. A KML permite ser extendida, incluindo outros formatos através do elemento `ExtendedData` (ver Figura 4.25). A utilização da IVOML embebida na KML, torna possível usar contextos de localização mais elaborados, tirando partido das facilidades de definição de pontos e áreas geográficas da KML. O facto da KML ser um *standard* suportado pela maior parte dos Sistemas de Informação Geográfica, facilita a importação de contextos de localização para o IVO apartir das inúmeras bases de dados geográficas existentes, como Planos de Ordenamento dos Municípios, Plantas e Roteiros Municipais, entre muitas outras.

O *IVO Builder*, referido no capítulo anterior, disponibiliza uma funcionalidade de importação de ficheiros KML, o que permite adicionar a uma aplicação IVO os indicadores de locais (elementos `Placemark`) existentes nesse KML, os quais definem pontos e áreas de interesse.

Desta forma, uma aplicação IVO pode ser definida à custa de um ficheiro IVOML puro (não podendo neste caso possuir contextos de localização) ou de um ficheiro KML com IVOML embebido. Os clientes IVO devem ser capazes de interpretar ambos os formatos.

A Figura 4.25 ilustra um exemplo de KML com código IVOML embebido entre as linhas 14 e 24. Neste exemplo, o *workflow* "Sair do Escritório" (linhas 19 a 21) é executado sempre que o utilizador sai da área definida pelo*Placemark* "ID_Escritorio" (linhas 26 a 28). Este *workflow* apenas executa uma actividade (linha 20) correspondendo ao envio de uma SMS para o telefone especificado e com a mensagem indicada. Nas linhas 2 a 11 são definidos os `namespaces` utilizados, bem como os respectivos XSD que permitem validar a formatação de todo o ficheiro. A ligação da IVOML com a KML é feita através do atributo `placemark` dos contextos de localização `where` (linha 17).

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <kml xmlns=http://www.opengis.net/kml/2.2
3       xmlns:ivo=http://www.integratedvirtualoperator.com/ivoml/1.0
4       xmlns:xf=http://www.w3.org/2002/xforms
5       xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
6       xsi:schemaLocation="http://www.opengis.net/kml/2.2
7          http://schemas.opengis.net/kml/2.2.0/ogckml22.xsd
8          http://www.integratedvirtualoperator.com/ivoml/1.0
9          http://www.integratedvirtualoperator.com/ivoml/1.0/ivoml10.xsd
10         http://www.w3.org/2002/xforms
11         http://www.w3.org/MarkUp/Forms/2002/XForms-Schema.xsd">
12   <Document>
13     <ExtendedData>
14       <ivo:ivoml name="Sair do Escritório" version="1.0">
15         <ivo:permission>Internet GPS</ivo:permission>
16         <ivo:context name="Sair do Escritório">
17           <ivo:where condition="onExit" placemark="ID_Escritorio"/>
18         </ivo:context>
19         <ivo:workflow name="Sair do Escritório">
20           <ivo:sms message="A sair do escritório." to="..."/>
21         </ivo:workflow>
22         <ivo:scenario name="Sair do Escritório"
23              context="Sair do Escritório" workflow="Sair do Escritório"/>
24       </ivo:ivoml>
25     </ExtendedData>
26     <Placemark id="ID_Escritorio">
27       placemark-definition
28     </Placemark>
29   </Document>
30 </kml>
```

Figura 4.25. IVOML embebida em KML

Capítulo 5

Metodologia de Desenvolvimento IVO

Conteúdo

Resumo

Neste capítulo descreve-se uma proposta de metodologia para o desenvolvimento de aplicações sensíveis ao contexto através do IVO, detalhando as principais actividades de forma a proporcionar uma abordagem sistémica para desenvolver as aplicações. Esta metodologia, propositadamente flexível, pretende dar uma orientação aos desenvolvedores, de forma a guiá-los na construção das aplicações.

5.1 Visão Global do Processo

A engenharia de *software* é descrita na literatura como um conjunto de actividades que incluem, segundo Fuggetta [95] e Sommerville [96], a análise de requisitos e especificação, o desenho, o desenvolvimento, a verificação e validação, a implementação, a operação e a manutenção. A inclusão da sensibilidade ao contexto nas aplicações implica uma forma diferente de pensar a

engenharia de *software*. Deve ser dada uma ênfase maior à análise de interacções e ao que os utilizadores esperam da reacção da aplicação a determinada variação do contexto. Desta forma, o desenvolvimento destas aplicações deve ter em consideração um conjunto de actividades especificamente relacionadas com a manipulação do contexto. Vieira *et al.* [97] classificam estas actividades em três categorias principais: *context specification*, *context management* e *context usage*. Vieira [98] apresenta ainda um processo que direcciona a execução de actividades relacionadas com a especificação do contexto e com o projecto de sistemas sensíveis ao contexto. De igual forma, Henricksen e Indulska [99] propõem uma metodologia de engenharia de *software*, que define o fluxo de actividades que deve ser tido em conta para se desenvolver uma aplicação sensível ao contexto, a qual envolve fases de análise, projecto, implementação, customização da infra-estrutura e teste. Contudo, estas metodologias não têm em conta o desenvolvimento por parte de não-programadores.

Uma vez que os utilizadores que irão utilizar o IVO para criar as aplicações possuem potencialmente diferentes formações base, cultura, competências e métodos de organização e de trabalho, a metodologia de desenvolvimento de uma aplicação construída com a plataforma deverá ser propositadamente flexível.

A metodologia típica pode ser traduzida através da figura seguinte, a qual encontra forte inspiração nas metodologias ágeis de desenvolvimento de *software* em particular na Programação Extrema [100] (*Extreme Programming* ou XP), na SCRUM [101] e no Desenvolvimento Orientado a Funcionalidades [102] (*Feature Driven Development* ou FDD).

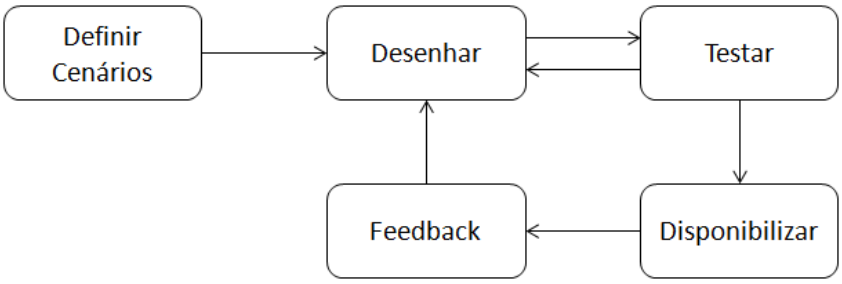


Figura 5.1. Metodologia de Desenvolvimento de uma aplicação IVO

Esta metodologia partilha o objectivo dos métodos iterativos e incrementais de criar partes do *software* que possam ser lançadas em curtos períodos tempo. Ela permite desenvolver uma aplicação de forma rápida e introduzir facilmente novas funcionalidades, as quais podem corresponder a novos cenários ou novas acções associadas aos cenários existentes. Uma nova funcionalidade pode demorar entre poucos minutos, se por exemplo estivermos apenas a acrescentar uma actividade num *workflow*, até duas semanas se tivermos a falar da criação de um

novo cenário com recolha e tratamento de informação, incluindo a produção de textos, imagens, vídeos, formulários e eventuais interfaces com sistemas externos através de *Web Services*.

5.2 Definir Cenários

A primeira actividade no desenvolvimento de uma aplicação deverá ser a definição de cenários, os quais descrevem situações de utilização que os utilizadores gostariam que o sistema venha a disponibilizar. Assemelham-se aos casos de uso da UML[18] e às histórias da Programação Extrema, funcionando ainda como os documentos de requisitos nos métodos tradicionais de desenvolvimento de *software*.

Cada cenário deverá descrever com exactidão a situação a que se aplica (o contexto) e as acções que deverão ser executadas quando essa situação ou contexto se verificar (o *workflow*).

Na definição do cenário deve também, sempre que possível, ser fornecida informação contextual e/ou as possíveis fontes de informação associadas ao cenário.

Os cenários poderão ser agrupados por conjuntos funcionais, isto é, conjuntos de cenários que fazem sentido disponibilizar apenas juntos. Isto permite uma abordagem iterativa e incremental.

Os cenários são ainda a base para a elaboração de testes que permitem verificar se o programa implementa os cenários correctamente.

5.3 Desenhar

Esta actividade consiste na utilização das ferramentas de construção para implementar os cenários desejados para a aplicação. O conjunto de tarefas a realizar nesta actividade pode ser resumido no esquema da figura seguinte.

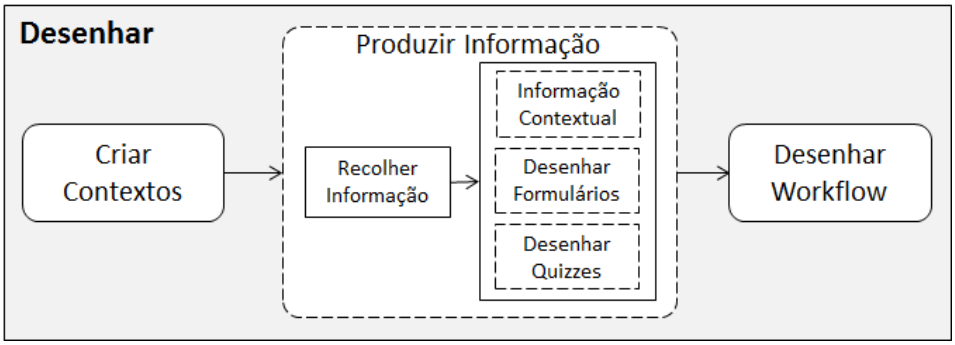


Figura 5.2. Actividade de Desenho no desenvolvimento de uma aplicação IVO

[18] Object Management Group - UML (http://www.uml.org)

5.3.1 Criar Contextos

Nesta tarefa está incluída a criação dos contextos dos cenários definidos. As ferramentas de construção disponibilizam modelos de interacção que facilitam a criação destes contextos, como por exemplo, a utilização de mapas para a criação de contextos de localização e interfaces adequadas para os outros tipos de contexto, tais como editores de expressões para contextos compostos.

5.3.2 Produzir Informação

Esta tarefa é opcional já que podem existir situações em que não é necessário produzir qualquer tipo de informação. Como exemplo disto, considere-se o caso em que apenas se pretende mostrar um alerta sempre que o estado da bateria é inferior a determinado valor, ou o caso em que se pretenda avisar o utilizador acerca do início de um evento.

Nesta tarefa está incluída a recolha de informação, a produção de informação contextual, o desenho de formulários e o desenho de *quizzes*.

Recolher Informação. Este passo corresponde à recolha de informação associada ao cenário. Poderão ser utilizadas as fontes de informação disponibilizadas na fase anterior (Definir Cenários), ou pode exigir a realização de pesquisas mais elaboradas dependente da natureza e disponibilidade de informação que se pretender disponibilizar durante a utilização da aplicação quando o contexto se verificar.

Informação Contextual. Este passo corresponde à elaboração da informação contextual associada ao cenário tendo por base as pesquisas e recolhas realizadas no passo anterior. Serão produzidos conteúdos usando editores HTML num ambiente WYSIWYG[19] de forma a facilitar a sua produção. Estes conteúdos podem incluir textos, imagens, animações Flash[20] e vídeos.

Desenhar Formulários. Este passo corresponde à criação dos formulários que possam ser utilizados no *workflow* associado ao cenário. O mesmo formulário pode ser utilizado em vários *workflows*, pelo que, a sua criação só é necessária uma vez. As ferramentas de construção disponibilizam um ambiente visual que permite a construção dos formulários de uma forma intuitiva, devendo o utilizador puder observar o resultado final à medida que vai definindo os diversos elementos do formulário.

[19]What You See Is What You Get

[20]Software utilizado geralmente na criação de animações interactivas que funcionam embebidas num *browser web* e também em computadores *desktops*, *smartphones*, *tablets*.

Desenhar Quizzes. Tal como nos formulários, a construção de *quizzes* é feita com base em ambientes visuais que facilitam a sua construção. Também um *quiz* pode ser utilizado em vários *workflows*, pelo que, a sua criação só é necessária uma vez.

5.3.3 Desenhar Workflow

Esta tarefa corresponde ao desenho do *workflow* associado ao cenário. Mais uma vez, são usados ambientes visuais que facilitam o desenho dos *workflows*. O *workflow* deve obedecer ao fluxo identificado na fase de definição do cenário, o qual pode fazer uso do conjunto de actividades disponibilizado pelo sistema.

5.4 Testar Aplicação

De forma muito genérica, esta actividade consiste em executar a aplicação desenvolvida com os casos de teste derivados dos cenários definidos.

A tarefa de testar aplicações sensíveis ao contexto envolve desafios adicionais face às aplicações tradicionais. Bylund e Espinoza [103] referem as dificuldades de testar aplicações que dependem de dados de sensores, e defendem ser essencial realizar testes não só com dados reais mas também com dados simulados. Infelizmente, não existem soluções satisfatórias para este problema, de modo que se descreve um processo de testes típico, em vez de um processo de testes ideal.

O facto do comportamento de uma aplicação sensível ao contexto ser dependente de muitas variáveis (incluindo os vários tipos de contexto, a configuração específica de cada utilizador e a existência de eventos gerados pelas variações de contexto), torna difícil antecipar com exactidão o comportamento da aplicação através do conhecimento exclusivo do modelo de contexto, das preferências do utilizador e dos possíveis eventos gerados. Consequentemente, grande parte dos testes do sistema consiste na realização de experiências que permitam ajustar as preferências, os contextos e a geração dos eventos resultantes das alterações destes, de forma a se obter o comportamento pretendido.

O estágio final de testes deve avaliar a aplicação no terreno, utilizando um ambiente real com utilizadores e dados reais obtidos dos sensores. Nesta fase poderão ser levantadas questões relativas à qualidade da informação, à privacidade e outras fontes de informação de contexto, levando a possíveis reiterações através da actividade anterior de desenho.

5.5 Disponibilizar

Após a aplicação ou uma sua actualização ser considerada pronta para ser disponibilizada, o seu criador deverá disponibilizá-la de acordo com os acessos que pretender para a mesma: privada, partilhada ou pública. A partir do momento em que a aplicação é disponibilizada, ela passa a estar disponível para *download* pelos utilizadores a quem tenha sido concedido acesso.

5.6 Feedback

A existência de ciclos de *feedback* no desenvolvimento das aplicações sensíveis ao contexto é particularmente importante em situações em que haja grande volatilidade nas informações contextuais a disponibilizar e nos contextos que dão origem aos *workflows*. Quanto mais se repetir este ciclo, mais *feedback* será produzido e mais a aplicação irá convergir para um produto que responda às necessidades reais e actuais dos utilizadores.

Capítulo 6

Implementação da Plataforma

Conteúdo

Resumo

Neste capítulo descreve-se a implementação de cada um dos componentes da plataforma apresentados no Capítulo 3. Estas descrições far-se-ão com grande recurso às interfaces desenhadas, por estas assumirem um papel fundamental no objectivo principal deste trabalho de facilitar a criação de aplicações sensíveis ao contexto por utilizadores finais sem necessidade de escrever qualquer linha de código. Começa-se por descrever a implementação do *Application Builder Server*, para depois se descreverem as ferramentas de composição *IVO Builder*, *IVO Calendar* e *IVO Contacts* e finalmente o *Mobile Client*.

6.1 Application Builder Server

O *Application Builder Server* fornece as páginas *web* usadas pelo *IVO Builder* e armazena as aplicações criadas. Disponibiliza também uma API que fornece um conjunto de serviços que permitem listar e descarregar as aplicações e ainda guardar o resultado do preenchimento dos formulários.

O *Application Builder Server* foi implementado em C# com a *Framework .NET 3.5* numa arquitectura *Microsoft* baseada no *Internet Information Server*.

6.1.1 IVO API

A IVO API foi desenhada tendo por base os princípios de desenho dos *web services* REST (*Representational State Transfer*). O REST é um estilo de arquitectura para sistemas distribuídos, tal como a própria *web*. Para implementar uma arquitectura REST existe um conjunto de directrizes que definem regras a serem seguidas, chamado ROA (*Resource-Oriented Architecture*).

A abstracção principal do REST são os recursos. Todos os recursos devem ter um URI (Uniform Resource Identifier) associado e a utilização destes URIs torna possível interligar recursos, tal como num ambiente hipermédia. É possível ter diferentes representações para o mesmo recurso. Por exemplo, um servidor pode devolver a representação de um recurso em HTML para ser lido por pessoas, ou então em XML e JSON se for para ser interpretado por computadores.

Os clientes interagem através dos métodos *standards* HTTP de GET, POST, PUT e DELETE. Estes métodos definem o que pode ser feito a um recurso. Por funcionar sobre HTTP, o REST tem uma comunicação sem estado, o que significa que os servidores não mantêm o estado das aplicações clientes, devendo estas enviar toda a informação de estado necessária nos pedidos.

No caso do IVO, todos os serviços necessitam receber o utilizador e a senha que garantem o acesso aos mesmos. Por este motivo, optou-se por fazer todos os pedidos dos serviços em POST, passando sempre o parâmetro `username` para o nome do utilizador e o parâmetro `password` para a senha. A senha deve ser encriptada com MD5 de forma a não circular na rede em texto puro. Caso o utilizador e/ou a senha não seja válida os pedidos devolvem sempre o *status* HTTP 401 (*Unauthorized*). O conjunto dos *status* que podem ser devolvidos na execução destes serviços são os ilustrados na Tabela 6.1.

Sucesso	Erro no Cliente	Erro no Servidor
200 – OK	400 – Bad Request	500 – Internal Server Error
201 – Created	401 – Unauthorized	501 – Not Implemented
	404 – Not Found	503 – Service Unavailable
	405 – Method Not Allowed	507 – Insufficient Storage

Tabela 6.1. Códigos de status

A IVO API disponibiliza quatro serviços que permitem: (i) a listagem de aplicações a que o utilizador tem acesso; (ii) o *download* de uma aplicação; (iii) o *upload* de ficheiros utilizados nos campos dos formulários do tipo fotografia e vídeo; e (iv) o armazenamento dos dados resultantes

do preenchimento de formulários. Na Figura 6.1 está ilustrado o funcionamento dos serviços disponibilizados pelo IVO.

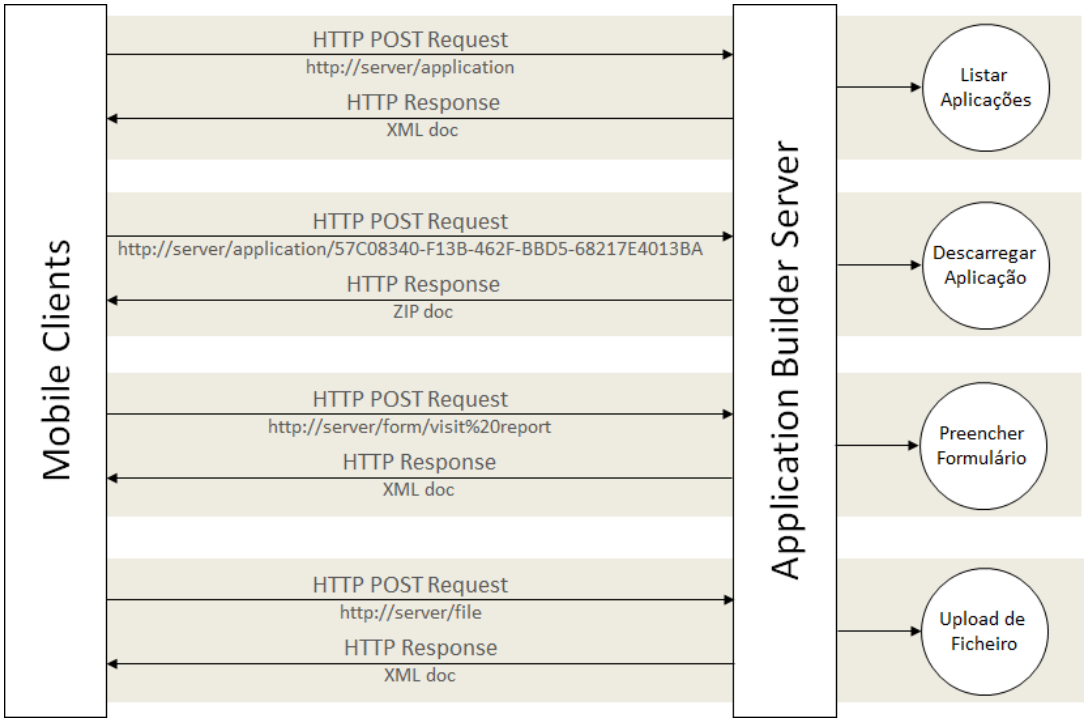


Figura 6.1. Funcionamento da API do IVO

Listagem de Aplicações

Este serviço devolve uma listagem com todas as aplicações a que o utilizador tem acesso. O URI associado a este serviço é:

```
http://server/application
```

Figura 6.2. Pedido de listagem de aplicações

Como resposta *o Application Builder Server* devolve uma listagem no formato exemplificado na Figura 6.3.

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <applications>
3     <application name="Belem Tour, Lisbon" version="1.1"
4         uri="http://server/application/57C08340-F13B-462F-BBD5-68217E4013BA" />
5     <application name="Obras de Gaudí" version="1.0"
6         uri="http://server/application/656A221E-46F1-430D-9518-9272692C4A10" />
7     <application name="Visita a Fornecedores" version="1.0"
8         uri="http://server/application/6298B649-D7A3-406E-93A8-EBBAE829627A" />
9 </applications>
```

Figura 6.3. Resposta a um pedido de listagem de aplicações

O atributo `version` deve ser usado pelos *Mobile Clients* para detectar a existência de novas versões da aplicação devendo sugerir ao utilizador a sua actualização. O atributo `uri` indica o recurso que deve ser utilizado para descarregar a aplicação. O identificador de cada uma das aplicações (como no exemplo anterior o `57C08340-F13B-462F-BBD5-68217E4013BA`) é um GUID[21] (*Globally Unique Identifier*) gerado automaticamente quando o utilizador disponibiliza a aplicação ou uma nova versão de uma já existente (ver Secção 6.2.6).

Download de Aplicação

O *download* de uma aplicação deve ser feito através do URI da aplicação como exemplificado na Figura 6.4, o qual descarrega a aplicação com o nome "`Belem Tour, Lisbon`" devolvida da listagem anterior.

```
http://server/application/57C08340-F13B-462F-BBD5-68217E4013BA
```

Figura 6.4. Descarregamento de uma aplicação

Como resposta, o *Application Builder Server* devolve um ficheiro ZIP contendo o ficheiro IVOML com a aplicação e quatro pastas com os ficheiros que a aplicação pode utilizar: ficheiros de imagem, de áudio, vídeo e de *scripts* (ver Secção 6.2.5).

Preenchimento de Formulário

Este serviço permite guardar no *Application Builder Server* os valores introduzidos no preenchimento de um formulário. O formato do URI associado ao serviço encontra-se exemplificado na Figura 6.5 para o caso de um formulário com o nome "Relatório de Visita". O nome do formulário encontra-se codificado (URL Encoding[22]) de forma a permitir a utilização de caracteres não válidos no URI (`Relat%F3rio%20de%20Visita`).

```
http://server/form/Relat%F3rio%20de%20Visita
```

Figura 6.5. Exemplo de pedido de preenchimento de formulário

Para além dos parâmetros referidos anteriormente (o `username` e o `password`), este serviço requer também os seguintes parâmetros:

[21] http://en.wikipedia.org/wiki/Globally_unique_identifier

[22] A codificação URL de um caracter consiste na sua substituição pelo símbolo % seguido da representação hexadecimal com dois dígitos do código ISO-Latin do caracter

Parâmetro	Descrição
`application`	Nome da aplicação IVO que contém o formulário.
`timestamp`	Data e hora em que o formulário foi preenchido.
`values`	Valores preenchidos pelo utilizador devendo cada campo estar separado do seguinte por um ponto e vírgula.

Tabela 6.2. Parâmetros do pedido de preenchimento de formulário

Em caso de sucesso o *status* de pedido deve ser 201 (*Created*).

O *IVO Builder* disponibiliza uma interface que possibilita o descarregamento de um ficheiro CSV[23] com o resultado do preenchimento dos formulários por parte dos utilizadores.

Upload de Ficheiro

O serviço para *upload* de ficheiros é utilizado pelo processador de formulários para enviar os campos do tipo fotografia e vídeo para o *Application Builder Server*. Sempre que num formulário for encontrado um campo de um destes dois tipos, o ficheiro que lhe está associado é enviado pelo cliente através de um pedido POST em formato `MIME multipart`. O URI associado ao *upload* de ficheiros é:

```
http://server/file
```

Figura 6.6. Upload de ficheiro

Como resposta o servidor responde com o URL para o ficheiro que acabou de ser carregado no formato XML exemplificado na Figura 6.7. O nome do ficheiro é gerado através de um GUID, sendo a extensão do ficheiro igual à do ficheiro que foi carregado. No caso do exemplo da Figura 6.7 o nome do ficheiro criado foi `7E9F3848-184E-4C30-94E9-BCCCE4009FAB.png`.

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <url>http://server/file/7E9F3848-184E-4C30-94E9-BCCCE4009FAB.png</url>
```

Figura 6.7. Resposta a um pedido de upload de ficheiro

O processador de formulários coloca no valor do respectivo campo (do tipo fotografia ou vídeo) o valor do URL devolvido para que este possa ser referenciado.

[23] *Comma Separated Values* - ficheiro que consiste num determinado número de registos, separados por quebras de linha, e em que cada registo é composto por campos separados por um delimitador geralmente a vírgula.

6.2 IVO Builder

O *IVO Builder* constitui a ferramenta principal de construção das aplicações. Esta ferramenta permite a edição dos ficheiros IVOML referidos no Capítulo 4, sendo uma aplicação *web* cujo diagrama de componentes (ilustrado na Figura 6.8) é composto por oito componentes.

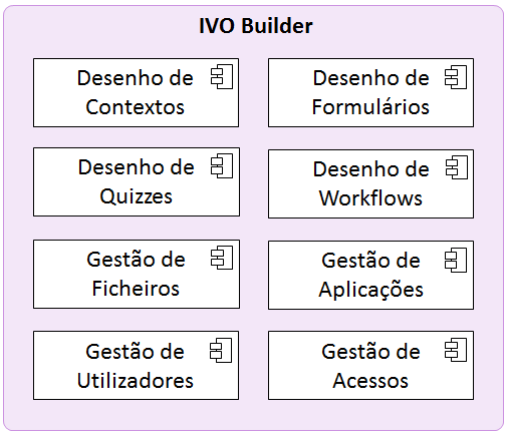


Figura 6.8. Diagrama de componentes do IVO Builder

Estes componentes disponibilizam um conjunto de interfaces que foram desenhadas de acordo com os princípios orientadores e os requisitos funcionais e não funcionais referidos no Capítulo 3. Estas interfaces visam facilitar a criação de aplicações sensíveis ao contexto, estando disponíveis apenas aos utilizadores registados. Só estes utilizadores podem criar as aplicações, podendo contudo disponibilizá-las a outros utilizadores que as poderão executar nos seus dispositivos móveis, tal como ilustrado nos vários cenários de uso da Secção 3.2.

A Figura 6.9 representa o écran principal do *IVO Builder* carregado com uma aplicação de Guia Turístico para a cidade de Portalegre. A zona 3 da figura disponibiliza o acesso às funções básicas que permitem a gestão das aplicações e da conta do utilizador; a zona 2 disponibiliza um mapa para a gestão de contextos de localização, enquanto a zona 1 dá acesso aos elementos fundamentais do desenho de uma aplicação: os cenários, os formulários, os *quizzes* e os ficheiros que a aplicação utiliza.

De seguida será explicado cada um dos elementos do diagrama de componentes referidos na Figura 6.8 ilustrando-os com as respectivas interfaces.

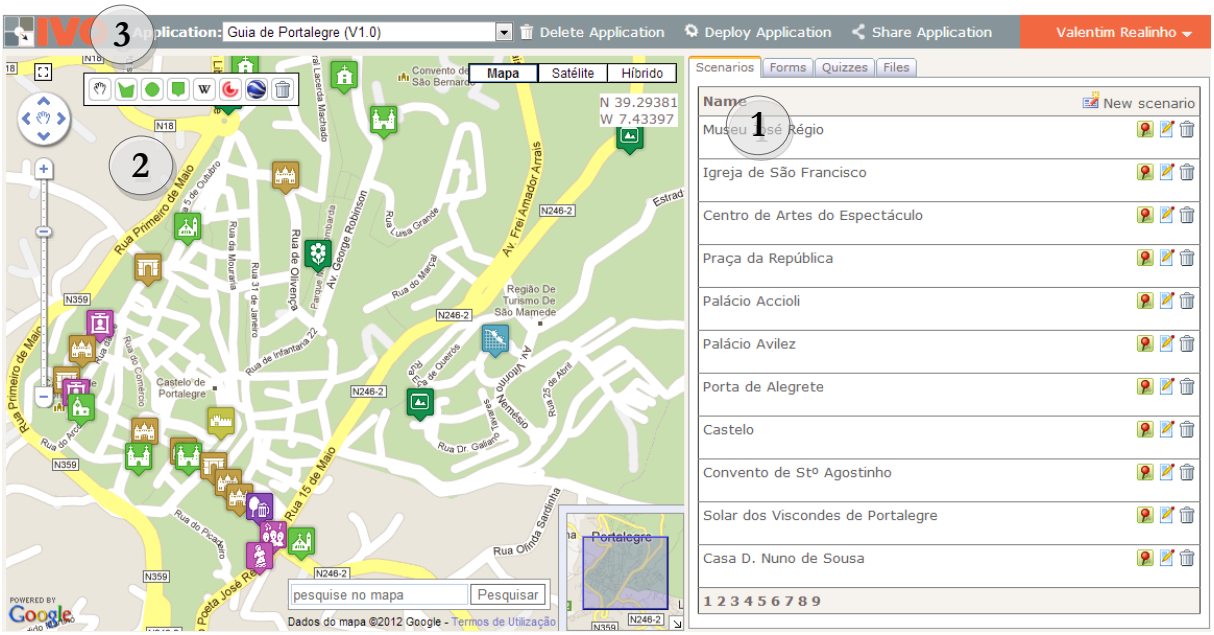
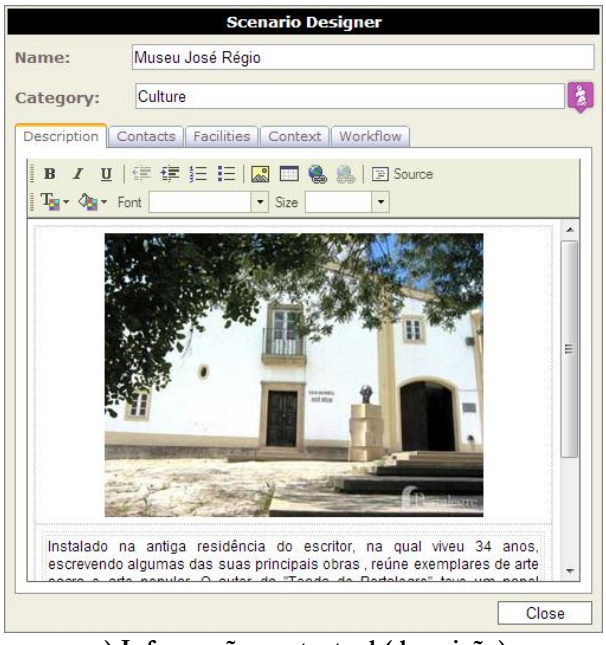


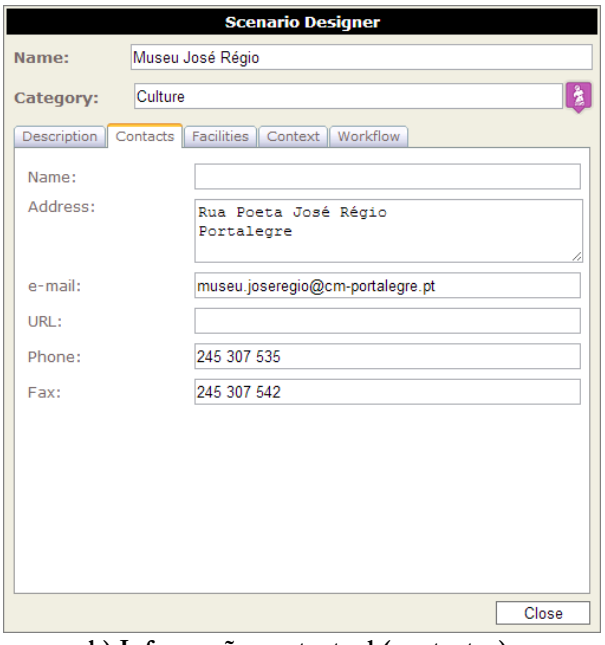
Figura 6.9. Écran principal do IVO Builder

6.2.1 Desenho de Cenários

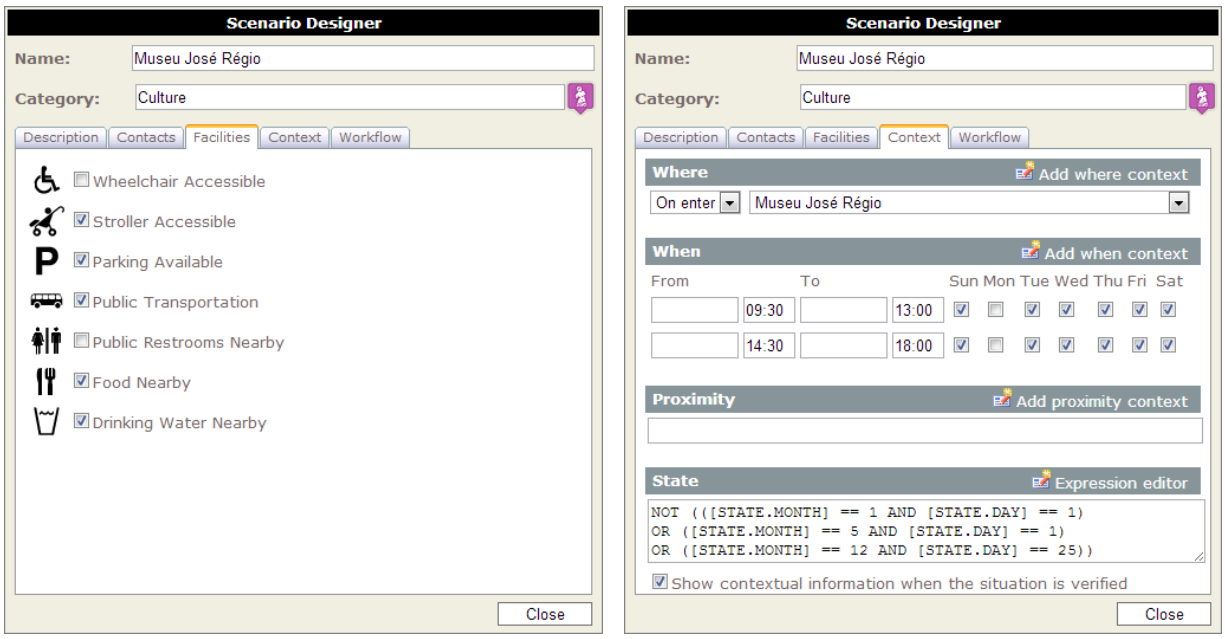
Na zona 1 da Figura 6.9, o separador "*Scenarios*" mostra todos os cenários existentes na aplicação. O desenho de cenários faz-se a partir deste separador através das janelas da Figura 6.10 onde estão ilustradas as interfaces que permitem ao utilizador definir um cenário.



a) Informação contextual (descrição)



b) Informação contextual (contactos)



c) Informação contextual (facilidades) **d) Contexto associado ao cenário**

Figura 6.10. Desenho de cenários

Estas janelas disponibilizam interfaces que permitem definir os atributos que constituem um cenário, nomeadamente a informação contextual constituída pela descrição, contactos e facilidades e ainda o contexto a que o cenário se aplica.

Definição da informação contextual

A informação contextual não é obrigatória estar preenchida e pode ser constituída por:

- **Descrição.** Corresponde à descrição do contexto a qual é criada através de um editor visual HTML (Figura 6.10a). Esta descrição pode conter os elementos habituais de HTML como imagens ou *links*;
- **Contactos.** Informação relativa a contactos associados ao contexto (Figura 6.10b) como morada, *email*, telefone, fax ou o *url* para o site com informação sobre o contexto;
- **Facilidades.** No caso de se tratar de um contexto de localização, este elemento contém informação relativa às facilidades na proximidade do local (Figura 6.10c), como por exemplo as acessibilidades para utilizadores em cadeira de rodas e com carros de bebé, a existência de casas de banho, parques de estacionamento ou transportes públicos entre outros.

Definição do contexto

O contexto a que o cenário se aplica é definido através da janela ilustrada na Figura 6.10d e permite associar várias dimensões de contexto como a localização (*Where*), tempo (*When*), proximidade de pessoas ou equipamentos (*Proximity* ou *Who*) e ainda estado (*State*). Todos os cenários devem ter um contexto a que se aplicam devendo estar definido pelo menos uma destas

dimensões de contexto. No exemplo da Figura 6.10d o contexto corresponde à entrada do utilizador na área definida para o Museu José Régio todos os dias entre as 9:30h e as 13:00h e entre as 14:30h e as 18:00h com excepção das segundas-feiras e dos dias 1 de Janeiro, 1 de Maio e 25 de Dezembro. Para definir esta situação foram definidas as seguintes dimensões de contexto:

Where Uma localização para quando o utilizador entrar no Museu José Régio. Esta localização foi criada previamente através das ferramentas de criação de contextos de localização (ver Secção seguinte);

When Dois períodos de tempo correspondentes ao horário da manhã e ao horário da tarde;

State Uma expressão que utiliza as variáveis de estado referentes ao dia e mês actual para definir as excepções ao horário de abertura normal: encerrado dia 1 de Janeiro, 1 de Maio e 25 de Dezembro.

Cada uma das dimensões de contexto pode ser constituída por várias instâncias desse mesmo tipo. Genericamente, a avaliação do contexto associado a um cenário é dada pela seguinte equação:

$$Context = \prod \left(\sum Where_n , \sum When_n , \sum Proximity_n , \sum State_n \right) \qquad (6.1)$$

Esta equação faz o `E` lógico do resultado da avaliação de cada dimensão de contexto. Por sua vez, o resultado da avaliação de cada dimensão é feito com um `OU` lógico sobre cada uma das instâncias que a constituem. Esta abordagem possibilita que a mesma informação contextual se aplique a: (i) vários locais bastando adicionar as diversas dimensões de localização (e se estas se aplicam à entrada ou à saída dos locais); (ii) vários períodos de tempo como, por exemplo, o período da manhã e da tarde (exemplo da Figura 6.10d) ou o horário de inverno e de verão; e (iii) proximidade de várias pessoas ou equipamentos.

6.2.1.1 Contextos de localização

Os contextos de localização são definidos na secção *Where* da janela da Figura 6.10d e devem ser criados antes de serem aqui utilizados através das ferramentas de criação de locais.

O lado esquerdo do écran principal (zona 2 na Figura 6.9) está reservado à representação num mapa das localizações da aplicação. No exemplo desta figura, as localizações estão representadas através de um ponto com o icon correspondente à categoria do ponto de interesse.

A barra de ferramentas nesta área dá acesso às funcionalidades que permitem a criação de novas localizações, nomeadamente:


Criação manual de uma localização através da definição de um polígono que limita a área.


Criação manual de uma localização representada através de um círculo.


Criação manual de uma localização representada através de um ponto.


Importação de localizações a partir da *Wikimapia* para a área visível do mapa.


Importação de localizações a partir de artigos geo-referenciados da *Wikipedia* para a área visível do mapa.


Importação de localizações a partir de um ficheiro KML com os locais (etiquetas *placemarks* do ficheiro KML).

Para além desta barra de ferramentas, esta zona disponibiliza botões para fazer *zoom* ao mapa e alterar o modo de visualização entre vista de mapa, de satélite e híbrido. Está ainda disponível uma caixa de texto para pesquisa rápida de lugares através da API de pesquisa do *Google*.

Criação manual de localizações. Estão disponíveis três botões na barra de ferramentas que permitem a criação manual de localizações. Estes botões permitem a definição de uma localização como um polígono que limita a área envolvente, como um círculo e ainda como um ponto.

Criação de localizações a partir da Wikimapia. A *Wikimapia* é um serviço de pesquisa e mapeamento de locais na internet. Qualquer pessoa tem permissão de inserir um local no mapa e até mesmo de editar locais já criados. Em Fevereiro de 2012 a *Wikimapia* contava com cerca de 18 milhões de locais marcados em todo o mundo. A marcação destes locais envolve basicamente a definição da área ocupada, título, descrição, endereço e categorias do local. O *IVO Builder* permite a importação destes locais através do respectivo botão da barra de ferramentas. Ao clicar nesse botão é efectuada uma consulta na *Wikimapia* aos locais que estejam dentro da área visível do mapa. A Figura 6.11 ilustra o resultado dessa pesquisa para a área de Évora mostrada no mapa, salientando-se a quantidade bastante razoável de locais marcados por utilizadores voluntários da *Wikimapia*. Estes locais estão marcados com a área de cor amarela e com o icon da *Wikimapia*. O *click* neste icon permite mostrar o balão com o nome da área e um *link* que permite a importação para o IVO. Na Figura 6.11 mostra-se este balão para o caso da área referente à Universidade de Évora. Assinalado no mapa desta figura com a cor verde está representada uma localização previamente criada para o jardim de Diana.

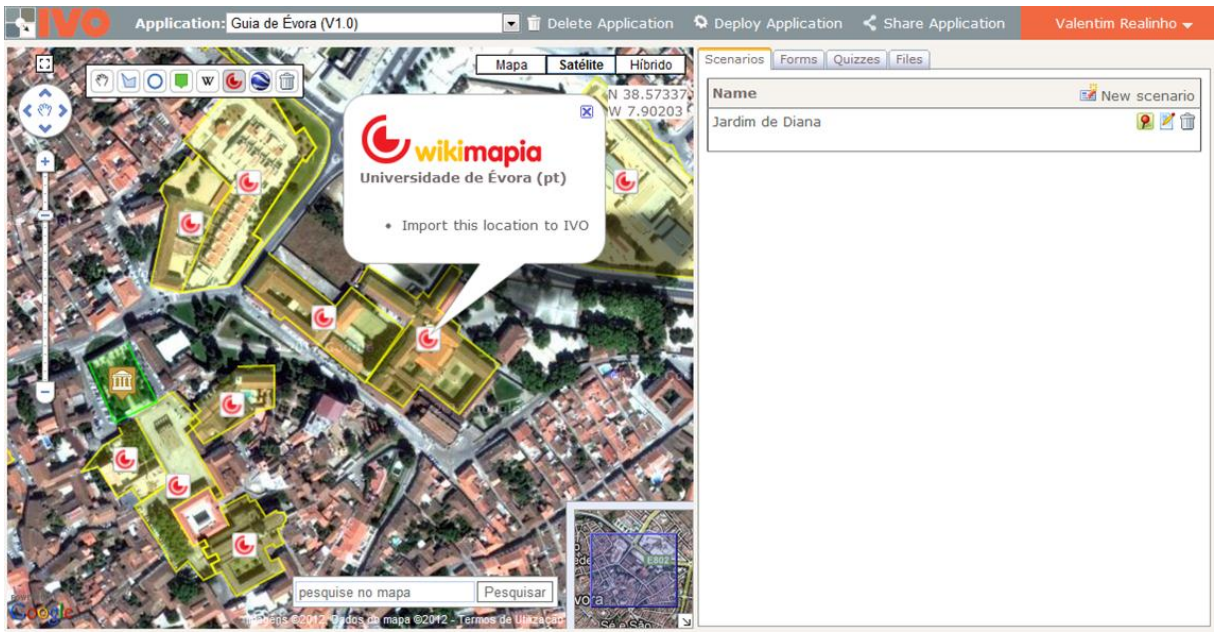


Figura 6.11. Importação de áreas da Wikimapia

Criação de localizações a partir da Wikipedia. O *IVO Builder* permite também a importação de artigos geo-referenciados da *Wikipedia* através do respectivo botão da barra de ferramentas. Tal como no caso anterior, ao clicar nesse botão é efectuada uma consulta aos artigos da *Wikipedia* que estejam geo-referenciados na área visível do mapa, usando para o efeito o serviço de pesquisa de dados geográficos do GeoNames[24].

A Figura 6.12 ilustra o resultado dessa pesquisa para a área de Évora mostrada no mapa. Neste caso, foram encontrados dois artigos os quais se encontram assinalados com o icon da *Wikipedia*: um referente à Igreja de São Mamede e outro referente à Sé Catedral de Évora. Mais uma vez, o *click* no respectivo icon mostra o balão com o nome do ponto e um *link* que permite a importação para o IVO. Nesta figura pode-se observar também o local associado à Universidade de Évora criado através da importação ilustrada anteriormente.

[24] http://www.geonames.org

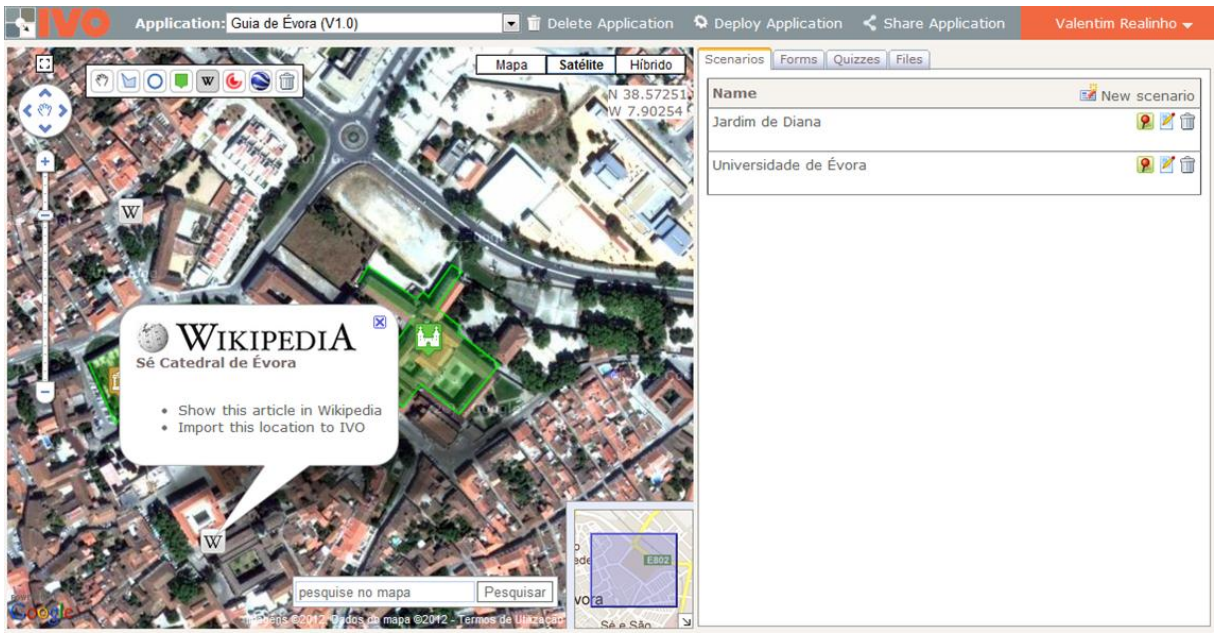


Figura 6.12. Importação de artigos geo-referenciados da Wikipedia

Criação de localizações a partir de ficheiros KML. O cenário de uso 3 (Secção 3.2) ilustra uma situação em que o utilizador importou localizações a partir de ficheiros KML tirando partido do facto da maior parte dos Sistemas de Informação Geográfica permitirem a exportação para este formato bastante popular. O IVO possibilita a importação de localizações definidas em ficheiros KML, bastando o utilizador indicar qual o ficheiro. Todos os locais marcados nos elementos *placemark* do ficheiro KML são importados para a aplicação IVO.

6.2.1.2 Contextos de Estado

Os contextos de estado são definidos na secção *State* da janela da Figura 6.10d e podem ser escritos com o auxílio do editor de expressões. A Figura 6.13 ilustra a interface do editor de expressões com a expressão da Figura 6.10d referente à excepção dos dias 1 de Janeiro, 1 de Maio e 25 de Dezembro.

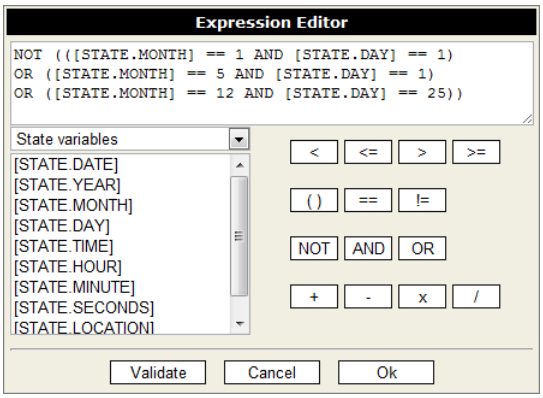


Figura 6.13. Editor de expressões na criação de dimensões de estado

O editor de expressões disponibiliza uma interface que facilita a criação das regras ou condições com possibilidade de validação da respectiva expressão. As expressões são compatíveis com a

Java Expression Language (JEXP)[25] da Apache permitindo a utilização dos operadores booleanos e matemáticos mais comuns. Na definição das expressões podem ser utilizadas as variáveis de contexto, podendo o utilizador escolher entre mostrar as variáveis de estado (como ilustrado na figura anterior) ou as variáveis de processo. Estas variáveis são explicadas de seguida.

Variáveis de Contexto. As variáveis de contexto, ou simplesmente variáveis, são usadas não só nas dimensões de estado para especificar condições, mas também nos formulários e nas condições de guarda das actividades (ver Secção 6.2.4). Os identificadores que representam as variáveis podem ser qualquer nome entre parênteses rectos e são *case sensitive*. Existem dois tipos de variáveis as quais são descritas a seguir: variáveis de estado e variáveis de processo.

Variáveis de Estado. Correspondem a variáveis que representam o estado actual do ambiente, incluindo os valores dos sensores, tais como a localização, a luminosidade, o nível da bateria, data/hora, a memória livre ou o ID do utilizador (Tabela 6.3). Estas variáveis são globais a todas as aplicações.

Variável	Descrição
`[STATE.DATE]`	Data actual
`[STATE.DAY]`	Dia actual (1 a 31)
`[STATE.MONTH]`	Mês actual (1 a 12)
`[STATE.YEAR]`	Ano actual no formato YYYY
`[STATE.TIME]`	Hora actual no formato HH:MM.SS
`[STATE.HOUR]`	Hora actual (0 a 23)
`[STATE.MINUTE]`	Minutos actuais (0 a 59)
`[STATE.SECONDS]`	Segundos (0 a 59)
`[STATE.LOCATION]`	Localização actual no formato decimal
`[STATE.LOCATIONACCURACY]`	Precisão da localização em metros
`[STATE.QRCODE]`	Código de barras 2D lido (QR-Code)
`[STATE.NFC]`	Código NFC lido
`[STATE.BATTLEVEL]`	Nível da bateria em percentagem
`[STATE.LIGHTNESS]`	Luminosidade
`[STATE.AREANAME]`	Nome do contexto de localização onde o utilizador se encontra
`[STATE.AREAID]`	Identificador do contexto de localização onde o utilizador se encontra
`[STATE.USER]`	Nome do utilizador
`[STATE.FREEMEM]`	Memória livre no dispositivo em MB

Tabela 6.3. Variáveis de estado

[25] http://commons.apache.org/jexl

Variáveis de processo. São variáveis visíveis apenas na instância de um processo de *workflow* e incluem as variáveis de formulário e as variáveis de atribuição:

1. **Variáveis de formulário** são variáveis utilizadas nos formulários. O tipo de dados é definido pela variável do formulário associada e o seu valor pode ser alterado através da actividade de execução de formulários (actividade *Form*) e por meio da actividade de atribuição (actividade *Assign).* A Figura 6.14 ilustra um formulário que utiliza as variáveis `[Product]`, `[Qty]`, `[Unit Price]`, `[Comments]`, `[Cash]` e `[Discount]`.

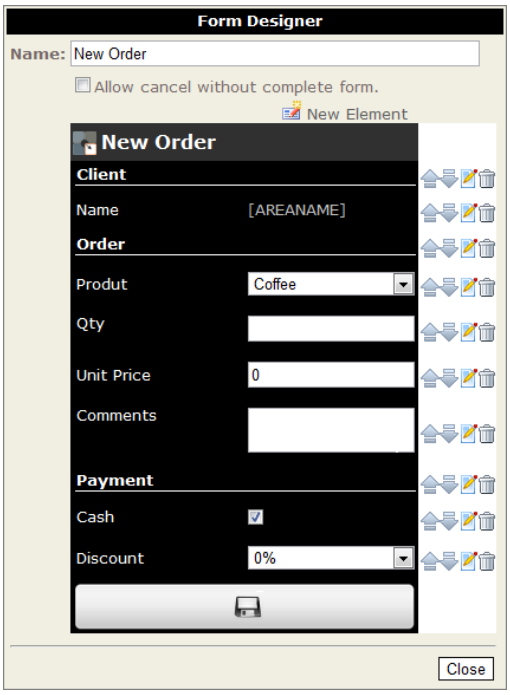
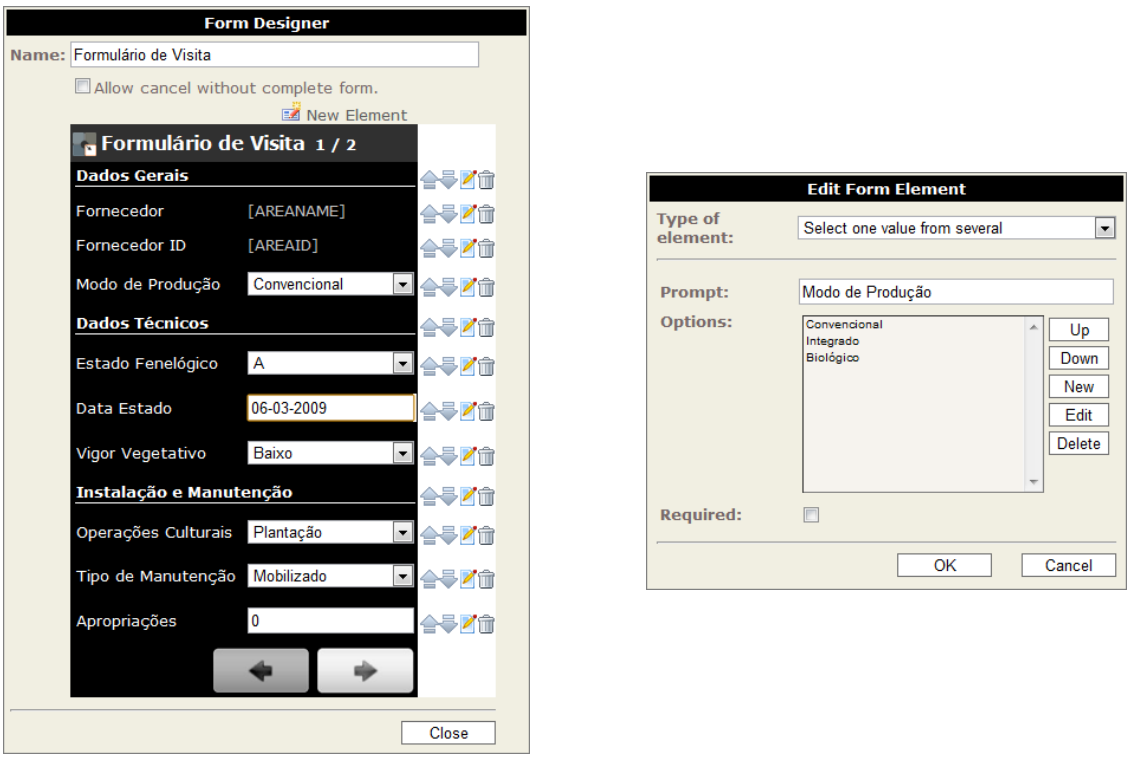


Figura 6.14. Exemplo de criação de formulário

2. **Variáveis de atribuição** são criadas quando usadas pela primeira vez numa actividade de atribuição (actividade "*Assign*"). Podem ser usadas em formulários como valor default de uma variável de formulário.

6.2.2 Desenho de Formulários

Os formulários construídos com o *IVO Builder* são compatíveis com o XForms 1.1 [93]. Os clientes devem incluir um processador de XForms de forma a serem capazes de executar os formulários. A Figura 6.15a mostra a janela de desenho de um formulário utilizado numa aplicação do sector agrícola.



a) Desenho de um formulário

b) Edição do elemento "Modo de Produção" do formulário

Figura 6.15. Desenho de um formulário no IVO Builder

Um formulário é constituído por vários elementos que podem ser de um dos tipos representados na Tabela 6.4.

Tipo	Descrição
Separador	Coloca uma linha com um título de forma a separar secções do formulário.
Etiqueta	Coloca um texto no formulário.
Caixa de texto	Permite a introdução de valores de texto.
Inteiro	Permite a introdução de valores inteiros.
Decimal	Permite a introdução de valores decimais.
Data	Permite a introdução de valores do tipo data.
Booleano	Permite a introdução de valores booleanos do tipo sim/não ou verdadeiro/falso, sendo representado através de uma *checkbox*.
Selecção de um valor entre vários	Permite a selecção de um valor pré-definido sendo representado através de uma *combobox*.
Selecção múltipla de valores	Permite a selecção de várias opções sendo representadas através de várias *checkboxs*.
Foto	Permite a selecção de uma foto. O utilizador poderá seleccionar uma foto existente ou então tirar uma no momento.
Vídeo	Permite a selecção de um vídeo. O utilizador poderá seleccionar um vídeo existente ou então gravar um no momento.

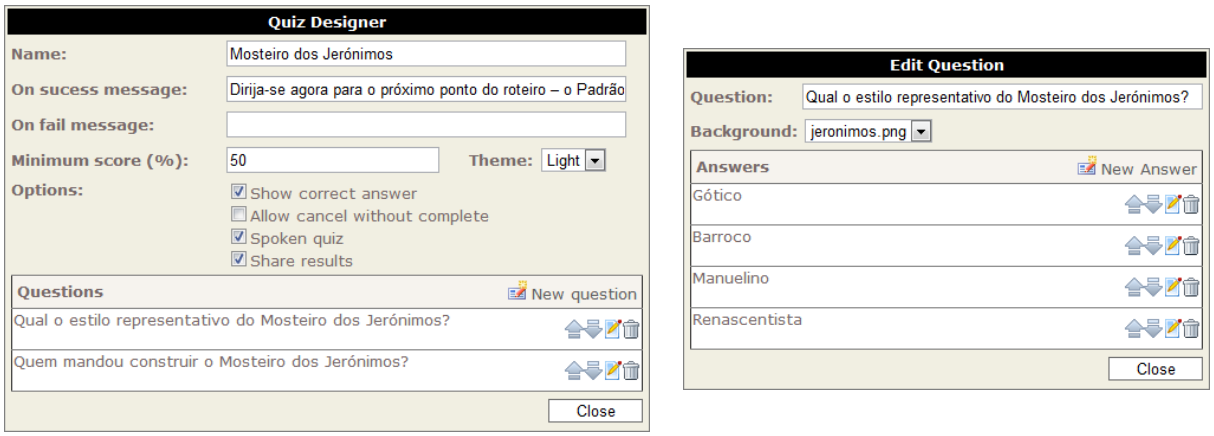
Tabela 6.4. Tipo de elementos dos formulários

A Figura 6.15b mostra a edição do elemento "Modo de Produção" do formulário da Figura 6.15a. Neste caso, trata-se de um campo com três valores pré-definidos dos quais o utilizador deverá escolher um, sendo representado como uma *combobox*.

Todos os campos de introdução de dados têm uma variável associada que é dada pela *Prompt* (ver Figura 6.15b). Como visto na Secção 6.2.1.2, estas variáveis são designadas por **variáveis de formulário** podendo ter associado um valor por omissão e possuir regras de validação (como valores mínimos e máximos) e podendo também ser configuradas como de preenchimento obrigatório ou não.

6.2.3 Desenho de Quizzes

Um *quiz* pode conter várias perguntas com um máximo de quatro respostas possíveis cada, sendo possível configurar como estes elementos são mostrados no cliente IVO. Por exemplo, pode ser definida uma pontuação mínima (em percentagem) que o utilizador deverá atingir para que ocorra a transição para a actividade seguinte do *workflow* ou ainda definir o tipo de *feedback* a dar ao utilizador entre cada pergunta. O *quiz* também pode ser falado utilizando mecanismos *text-to-speech*, caso esta tecnologia esteja disponível no lado do cliente.



a) Um quiz com duas perguntas **b) Criação de uma pergunta**

Figura 6.16. Desenho de quizzes

Na Figura 6.16a está ilustrado o desenho de um *quiz* sobre o Mosteiro dos Jerónimos, que inclui duas perguntas. O *quiz* está configurado para mostrar a mensagem indicada (*On sucess message*) sempre que o utilizador atingir a percentagem mínima de acertos (neste caso 50%); dar *feedback* ao utilizador mostrando a resposta correcta (*Show correct answer*); não permitir ao utilizador cancelar o *quiz* sem o terminar (opção *Allow cancel without complete* não seleccionada); *quiz* definido como falado (*Spoken quiz*); e permitir a partilha dos resultados nas redes sociais (*Share results*). A Figura 6.16b ilustra a edição da primeira questão deste *quiz* com as suas quatro respostas possíveis. O

resultado deste mesmo *quiz* a ser executado num dispositivo móvel Android pode ser observado na Figura 4.10.

6.2.4 Desenho de Workflows

Os *workflows* são fluxos de actividades controlados pelos dados contextuais lidos pelos sensores do dispositivo móvel ou pela interacção directa do utilizador. Os *workflows* podem ser sequenciais, incluir caminhos alternativos e ciclos, sendo implementados no lado do cliente como uma máquina de estados finita baseada em eventos (*event-driven finite state machine*).

Uma actividade de *workflow* é uma acção (por exemplo enviar um *email*) que deve ser executada automaticamente pela aplicação quando é atingido o estado correspondente da máquina de estados. Corresponde a uma das primitivas disponíveis como blocos de construção que podem ser usados para modelar *workflows* sequenciais, paralelos, condicionais ou iterativos.

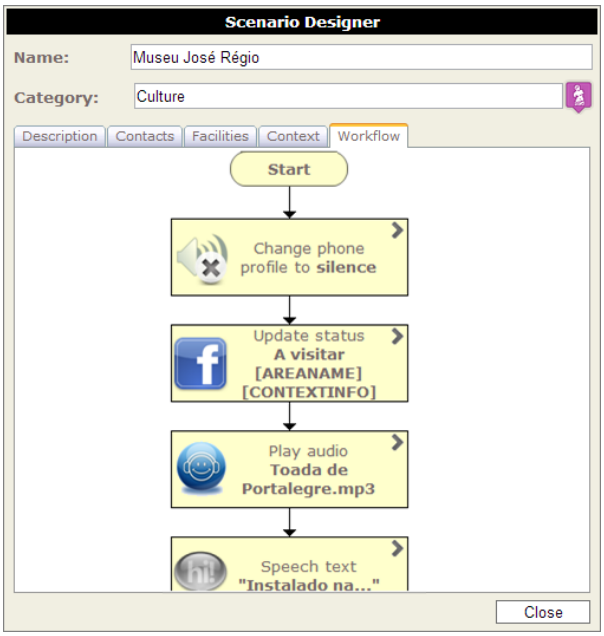


Figura 6.17. Desenho de workflow representado através do gráfico actividade/transição

Cada rectângulo da Figura 6.17 corresponde a uma actividade. As setas ilustram transições e podem ter uma condição de guarda associada que se refere a condições definidas pelas variáveis de contexto. Uma transição só ocorre se a condição de guarda se verificar, caso contrário o *workflow* fica bloqueado nessa actividade até que a condição se verifique. O *workflow* inicia-se quando a situação do contexto que lhe está associada se verifica, e que no caso anterior corresponde à situação ilustrada na Figura 6.10d. Neste caso, trata-se de um *workflow* sequencial que coloca o dispositivo em modo silencioso; coloca uma mensagem no mural do *Facebook* do

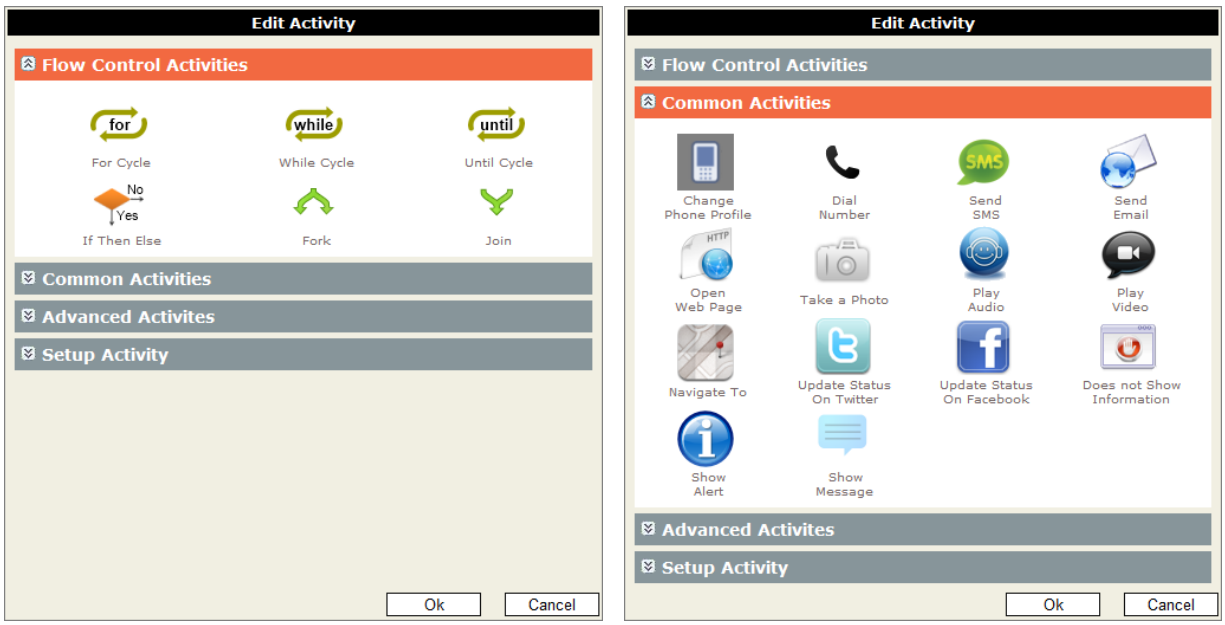
utilizador; toca o ficheiro de áudio com a Toada de Portalegre de José Régio e sobrepõe-lhe a mensagem referida na actividade seguinte usando *text-to-speech*.

A escolha do conjunto de blocos de construção (as actividades) disponíveis, foi feita de forma a permitir uma grande expressividade das aplicações que podem ser criadas. As actividades actualmente disponíveis foram escolhidas com base nos seguintes requisitos os quais levam em conta as primitivas básicas do WS-BPEL (vistas na Secção 2.4.1) e os blocos de construção básicos definidos pela *Workflow Management Coalition* (WfMC)[26]. Estes requisitos podem ser classificados em três categorias: controlo de fluxo, actividades comuns e actividades avançadas.

1. **Controlo de Fluxo**. Permitem a modelação de fluxos sequenciais, paralelos, condicionais e iterativos; permitem a execução de actividades síncronas e assíncronas; e, fornecem mecanismos para lidar com erros durante a execução das actividades.
2. **Actividades Comuns**. Fornecem informação contextual aos utilizadores; fornecem operações comuns nos telemóveis, como a mudança de perfil, incluindo ligar/desligar som, Wi-Fi, GPS ou *Bluetooth*, marcar um número, tirar uma foto ou fazer um vídeo, reproduzir áudio ou vídeo; fornecem protocolos comuns disponíveis em dispositivos móveis, tais como SMS, MMS, *Bluetooth*, email e HTTP; navegação por GPS; integração com redes sociais.
3. **Actividades Avançadas**. Fornecem mecanismos de interacção com o utilizador mais avançados como formulários, quizzes e interfaces de voz; execução de aplicações externas e suporte a linguagens de *script* para o desenvolvimento de funcionalidades mais avançadas; permitem também a integração com *web-services* externos.

A partir destes requisitos, foi desenvolvido o primeiro conjunto de actividades mostrado na Figura 6.18, que representa a janela de edição de actividades do *IVO Builder* a qual permite aos utilizadores seleccionarem uma nova actividade a ser inserida e configurada num *workflow*.

[26] http://www.wfmc.org/



a) Actividades de controlo de fluxo

b) Actividades de uso comum

c) Actividades avançadas

d) Parâmetros da actividade seleccionada

Figura 6.18. Janela de edição de actividades do IVO Builder

6.2.5 Gestão de Ficheiros

O componente de gestão de ficheiros permite gerir os ficheiros utilizados pela aplicação, podendo estes ser de um dos seguintes quatro tipos:

- **Imagens** utilizadas nas descrições dos contextos como exemplificado na Figura 6.10a;
- **Audio** utilizado nas actividades de *workflow* "*Play Audio*";
- **Video** utilizado nas actividades de *workflow* "*Play Video*";
- **Scripts** utilizados na actividades de *workflow* "*Script*".

O serviço de sincronização (*Syncronization Service*), referido na Secção 6.5.4, transfere a aplicação (o ficheiro IVOML) juntamente com estes ficheiros num único ficheiro ZIP[27] comprimido. O lado direito da Figura 6.19 ilustra a interface referente a este componente.

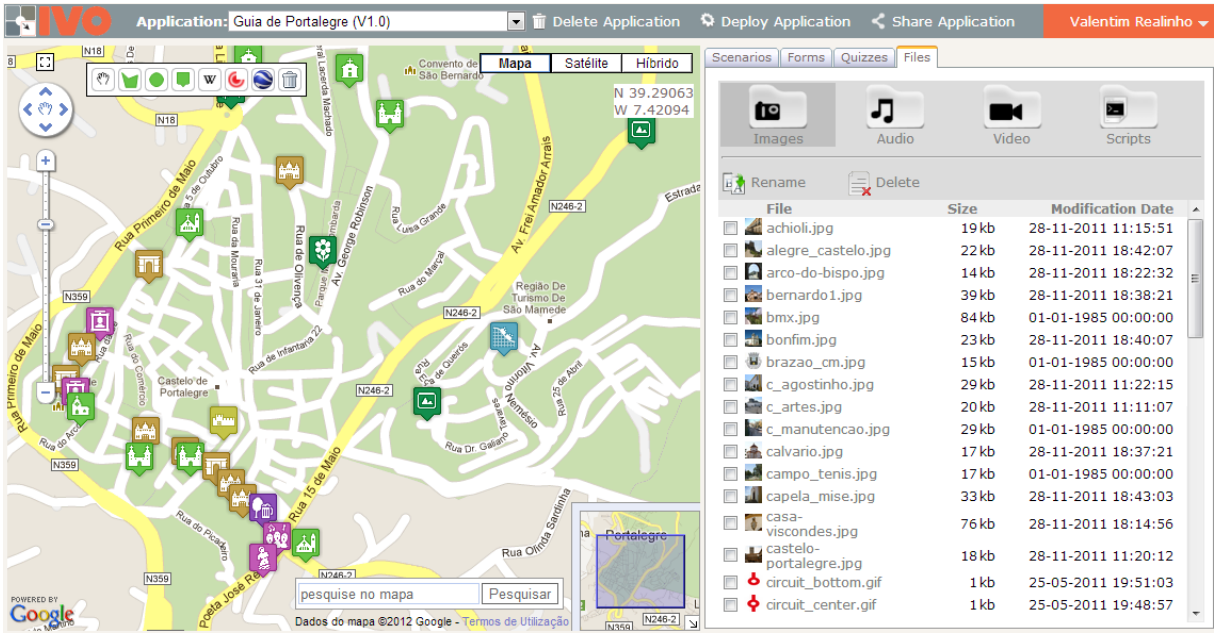
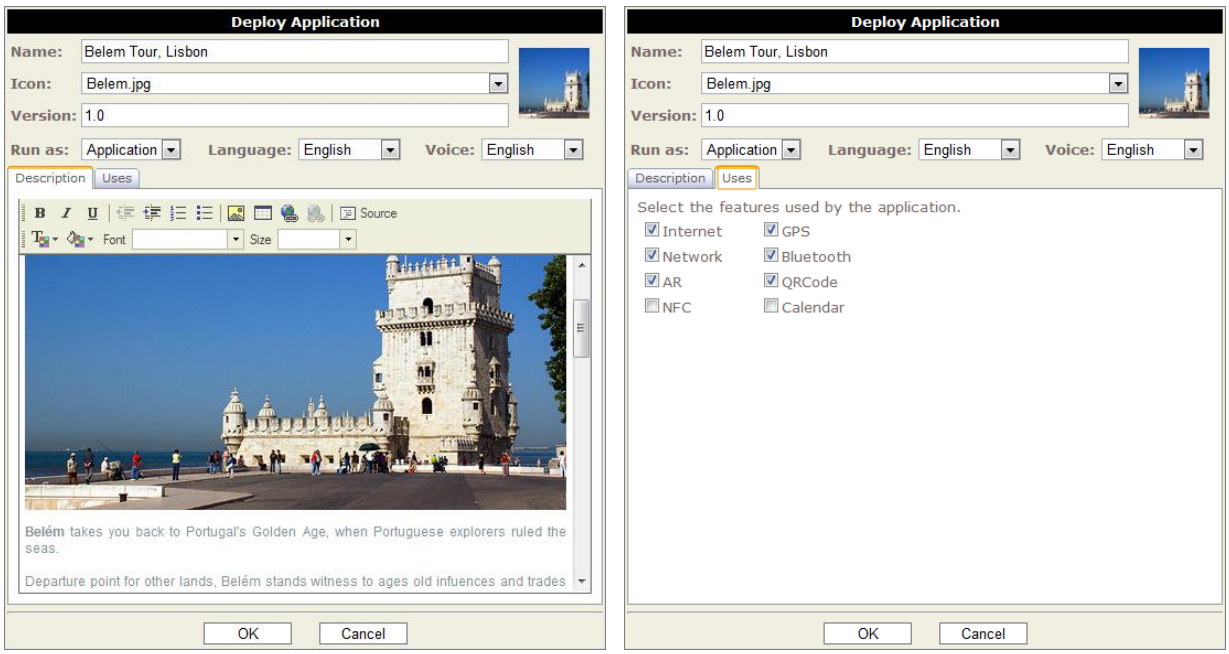


Figura 6.19. Gestão de ficheiros

6.2.6 Gestão de Aplicações

Na parte superior do écran principal do *IVO Builder* (zona 3 da Figura 6.9) temos as opções que possibilitam a gestão das aplicações, incluindo o carregamento de aplicações existentes, a criação, alteração, eliminação, disponibilização e partilha de aplicações. A disponibilização de uma nova aplicação ou de uma nova versão de uma já existente permite que os utilizadores com acesso a possam descarregar e utilizar através dos mecanismos de sincronização disponíveis.

[27] http://en.wikipedia.org/wiki/Zip_(file_format)



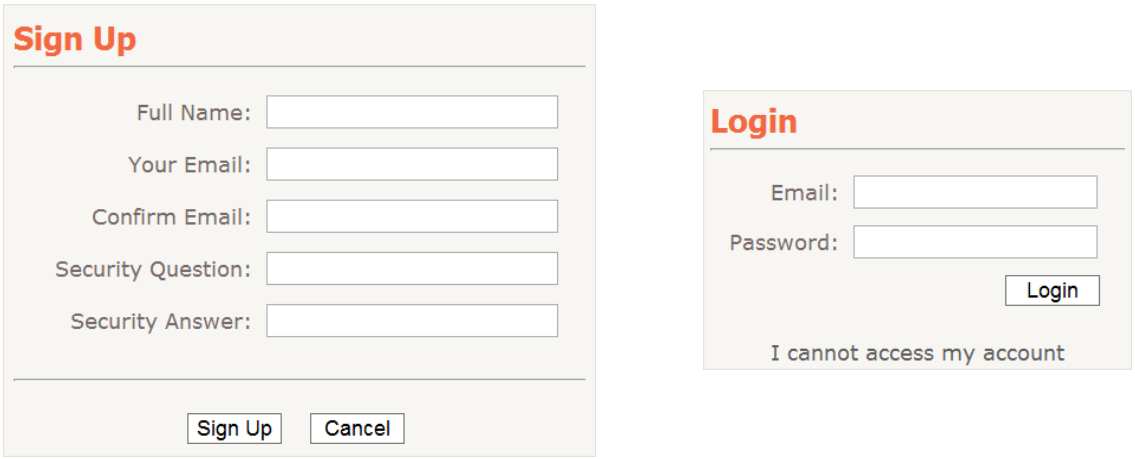
a) Opções e descrição da aplicação **b) Permissões da aplicação**

Figura 6.20. Disponibilização de uma aplicação

A Figura 6.20 ilustra os écrans que permitem a disponibilização de uma aplicação. Na Figura 6.20a ilustra-se as opções que configuram a aplicação, tais como o icon, a versão, o tipo de aplicação, a linguagem e a voz a ser utilizada nas tarefas de texto falado. Nesta mesma janela pode-se definir a descrição da aplicação a qual é mostrada no dispositivo móvel sempre que o utilizador carregar a aplicação ou na opção "Acerca de…" disponível no dispositivo móvel. A Figura 6.20b ilustra as permissões que a aplicação necessita. A correcta definição destas permissões é um factor importante, na medida em que possibilita a activação e desactivação de recursos no dispositivo móvel o que é particularmente relevante para uma utilização mais racional da bateria. Neste sentido deverão ser desactivados todos os serviços não utilizados pela aplicação.

6.2.7 Gestão de Utilizadores

Este componente é responsável por toda a gestão de utilizadores do IVO, disponibilizando as funções básicas para: (i) registo de novo utilizador (Figura 6.21a); (ii) entrada no sistema (*Login*) (Figura 6.21b); (iii) recuperação de senha(Figura 6.21b); e (iv) alteração dos dados da conta.



a) Registo de novo utilizador **b) Entrada no sistema**

Figura 6.21. Registo de novo utilizador e entrada no sistema

6.2.8 Gestão de Acessos

O componente de gestão de acessos permite gerir os acessos à aplicação. Uma aplicação pode ser privada (só o seu criador tem acesso), pública (todos os utilizadores têm acesso) e partilhada (apenas os utilizadores seleccionados têm acesso). Os acessos a uma aplicação partilhada são definidos através da indicação do *email* do(s) utilizador(es). Os utilizadores indicados receberão um *email* com uma mensagem personalizada com *links* que permitem a instalação do *runtime* no dispositivo móvel e para a aplicação propriamente dita, tal como ilustrado no cenário de uso 2 (Secção 3.2). A Figura 6.22 ilustra a interface que permite conceder acessos à aplicação.

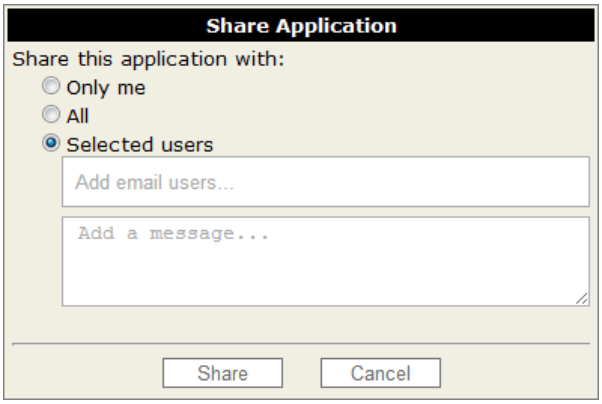


Figura 6.22. Gestão de acessos

Ao mudar o tipo de acesso de público ou de partilhado para privado os utilizadores que tinham a aplicação instalada deixam de ter acesso às actualizações das mesmas.

6.3 IVO Calendar para o Microsoft Outlook

Os contextos temporais podem ser definidos na aplicação de Calendário do *Outlook*, através de um *add-on* que adiciona uma região na parte inferior da janela de compromissos contendo a

interface que permite aos utilizadores definirem *workflows*. A Figura 6.23 ilustra a janela de um compromisso do Outlook onde se pode observar na parte inferior a interface do IVO. Este exemplo representa o cenário de uso 5 (Secção 3.2), com um *workflow* que começa "quando o compromisso tem início" e um *workflow* que começa "quando o compromisso termina". Neste caso, as actividades definidas para o início do compromisso são colocar o telefone em modo silencioso e actualizar o estado do utilizador no *Facebook* e no *Twitter*. Ao terminar o compromisso o perfil do telefone é colocado em modo normal.

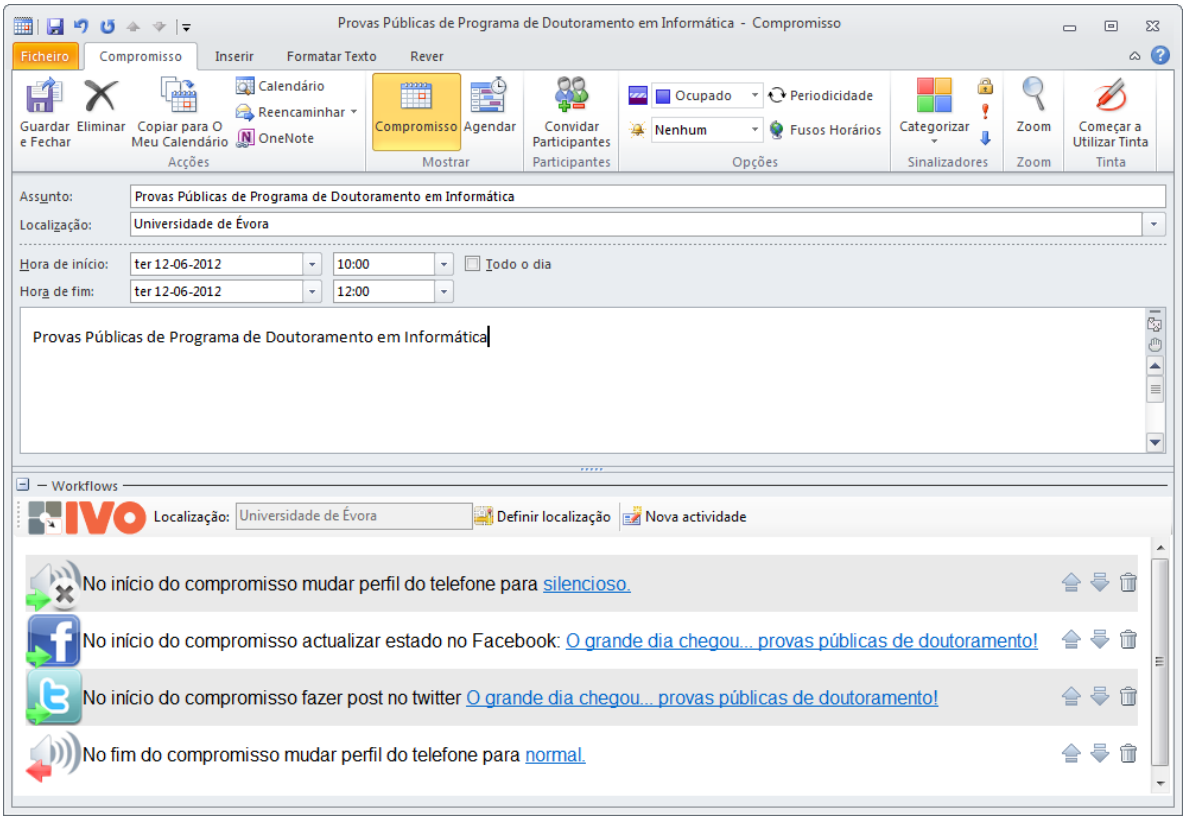


Figura 6.23. IVO Calendar para o Microsoft Outlook

As actividades disponíveis no IVO Calendar apenas permitem o desenho de *workflows* sequenciais, não estando disponível o conjunto de actividades de controlo de fluxo, nomeadamente as actividades `if-then-else`, `for`, `while`, `repeat-until` e `fork` (ver Secções 4.7.1 e 6.2.4). A Figura 6.24 ilustra a janela que permite adicionar uma nova actividade de *workflow*. O utilizador deve seleccionar a condição (no início ou no fim do compromisso) e a actividade a ser executada.

É ainda possível associar ao compromisso um contexto de localização, o qual pode ser definido através de um mapa fornecido ou ser importado a partir das aplicações criadas com o *IVO Builder* através da API REST. Desta forma, o sistema pode lidar com condições mais complexas como "*neste instante no tempo e quando estou neste local*" executar estas acções.

Figura 6.24. Adicionar uma nova actividade de workflow a um compromisso

6.4 IVO Contacts para o Microsoft Outlook

O *IVO Contacts* permite a utilização de contextos de proximidade, associando um dispositivo *Bluetooth* (o telefone) a um dos contactos do utilizador. Tal como o *IVO Calendar*, também o *IVO Contacts* é um *add-on* que adiciona uma região na parte inferior da janela dos contactos do Outlook, contendo a interface que possibilita a associação do dispositivo *Bluetooth* ao contacto. A Figura 6.25 ilustra a janela de um contacto do Outlook onde se pode observar na parte inferior a interface do IVO.

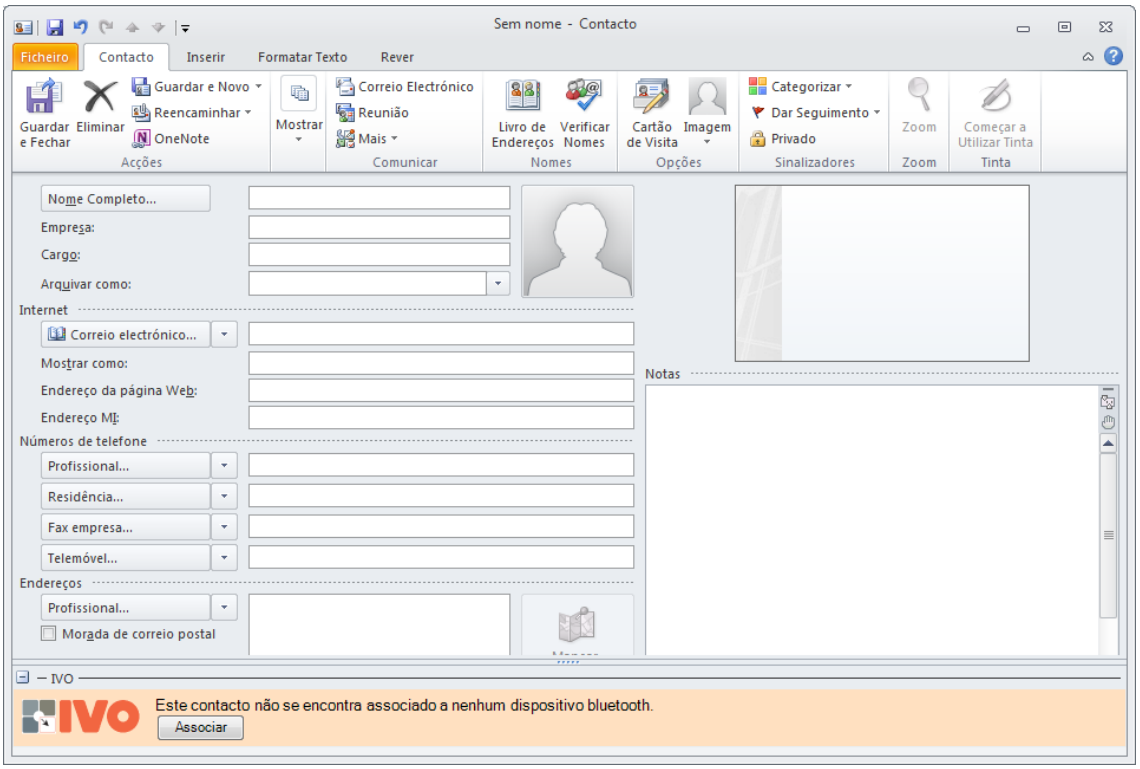
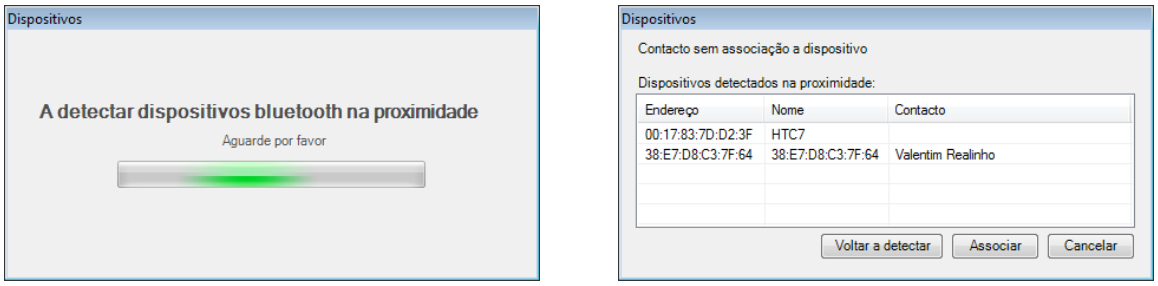


Figura 6.25. IVO Contacts para o Microsoft Outlook

Ao clicar no botão "Associar" é lançada a janela que permite a detecção dos dipositivos *Bluetooth* na proximidade permitindo a sua associação (Figura 6.26a). A Figura 6.26b mostra os dispositivos detectados, estando o segundo da listagem já associado a um contacto.



a) Detectar dispositivos Bluetooth **b) Dispositivos Bluetooth detectados**

Figura 6.26. Associar dispositivo Bluetooth a contacto

6.5 Mobile Client

Quando uma aplicação é carregada são activadas apenas as funcionalidades utilizadas pela mesma, de acordo com as definições introduzidas (ver Figura 6.20b). Se a aplicação tiver sido definida para correr como serviço, esta é executada em *background* sem possibilidade de nenhuma interacção com o utilizador, caso contrário é mostrado o écran principal.

A Figura 6.27 ilustra a *framework* implementada dos *Mobile Clients* a qual é composta por quatro componentes principais: *Sensing Monitor*, *Event-Condition-Workflow Engine*, *Workflow Engine* e *User Interface*. Estes componentes são descritos de seguida.

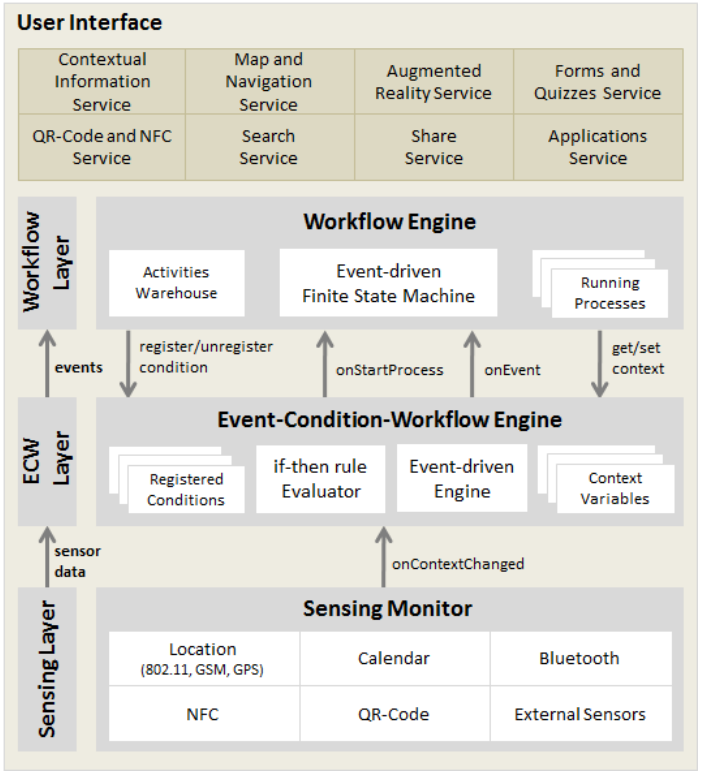


Figura 6.27. Framework do cliente IVO

6.5.1 Sensing Monitor

A *Sensing Monitor* faz a monitorização do ambiente, e pode ser ampliada através do desenvolvimento de novos *handlers* de sensores para permitir uma melhor capacidade de leitura do ambiente. A criação desses *handlers* é uma tarefa que envolve a programação de uma extensão a uma classe abstracta. Actualmente estão disponíveis *handlers* de sensores para fornecer contextos de localização através de GPS, GSM e redes Wi-Fi, *Bluetooth* para contextos de identidade ou proximidade de pessoas e equipamentos, leitura de códigos de barras 2D (QR-codes); contextos temporais definidos no IVO Builder ou na aplicação de Calendário do dispositivo móvel; e leitura de *tags* NFC.

Periocidade de Actualização dos Sensores. A frequência com que os sensores produzem dados depende da sua natureza. Por exemplo, o GPS produz dados quase de forma contínua, enquanto os códigos de barra (QR-Code) ou o NFC apenas quando detectam um código ou uma *tag* válida. A periocidade de actualização dos dados dos sensores pode ser configurada de duas formas distintas: (i) quando for detectada uma alteração nos dados do sensor; ou (ii) se tiver decorrido um determinado período de tempo desde a última actualização. Sempre que é actualizado um dado do sensor é gerado um evento `onContextChanged`, o qual é tratado pelo nível superior da *framework* responsável pelo mecanismo ECW.

Sensores Externos. Além dos sensores internos ao dispositivo móvel, é possível ligar o IVO a uma infra-estrutura externa de sensores que comunica com o IVO através de UDP (User Datagram Protocol). Para isso, foi desenvolvido um *handler* que está à espera de datagramas UDP num porto pré-definido. Este *handler* permite a ligação a qualquer infra-estrutura externa de sensores desde que esta seja capaz de enviar datagramas UDP no seguinte formato:

```
data/hora;variável;valor
```

Figura 6.28. Formato dos datagramas enviados por sensores externos

O campo `data/hora` contém a data e hora relativa ao momento em que os dados foram obtidos pelo sensor. O campo `variável` refere-se à variável associada ao sensor, de acordo com o formato genérico para a definição de nomes de variáveis visto anteriormente. Finalmente o campo `valor` representa o valor lido pelo sensor.

Este *handler* foi testado com um sensor de movimento desenvolvido para o efeito e que se baseia numa *webcam* ligada a um PC. Sempre que é detectado movimento é enviado um datagrama como o que se segue:

```
2011-12-29 09:23:04;[MOTION];true
```

Figura 6.29. Exemplo de datagrama enviado por um sensor de movimento

A aplicação que foi criada com o *IVO Builder* para testar este sensor, envia um SMS para um número pré-definido sempre que é detectado movimento. Esta aplicação tem um único contexto associado a uma situação que usa a variável do sensor na expressão `[MOTION] == true`. Finalmente, a aplicação possui um cenário que associa este contexto a um *workflow* que apenas envia um SMS para o número pré-definido.

6.5.2 Event-Condition-Workflow Engine

Esta camada fornece um mecanismo do tipo *register/unregister* de condições que é utilizado pelo *Workflow Engine* a fim de ser notificado de forma assíncrona pelo *event-driven engine* sempre que uma determinada condição é verificada. Sempre que um evento `onContextChanged` é gerado pela camada de detecção (*Sensing Layer*), o *event-driven engine* avalia todas as condições registadas. Um evento `onStartProcess` é gerado quando uma condição de início de *workflow* for detectada, enquanto um `onEvent` é gerado sempre que a condição associada a uma condição de guarda é verificada. As condições de início dos *workflows* são registadas quando a aplicação é carregada, enquanto que as condições de guarda são registadas quando termina uma actividade cuja transição para a actividade(s) seguinte(s) tem uma condição de guarda associada. As condições são avaliadas pelo *if-then rule evaluator* que usa a *Java Expression Language* (JEXL)[28].

Esta camada tem duas colecções que mantêm as condições registadas (*Registered Conditions*) e as variáveis de contexto existentes (*Context Variables*). As variáveis de contexto são actualizadas quer no processamento de um evento `onContextChanged` sempre que este é gerado (variáveis de estado), quer pelas actividades de *workflow* do tipo *Assign* e do tipo *Form* (variáveis de processo). Estes tipos de variáveis foram vistos na secção 6.2.1.2.

6.5.3 Workflow Engine

A lógica interna do *Workflow Engine* foi construída através de uma máquina de estados finita baseada em eventos (*event-driven finite state machine*) com suporte a condições de guarda. Os *workflows* são representados através de uma tabela de transição de actividades. As máquinas de estado finitas são muito expressivas e podem ser programadas através de uma abordagem puramente orientada a eventos e representadas através de um gráfico de actividade/transição.

[28] http://commons.apache.org/jexl

A máquina de estados (FSM) inicia-se quando a condição de início do *workflow* se verifica e é gerado um evento `onStartProcess`. O evento `onEvent` é gerado quando se verifica uma condição de guarda provocando a transição para a actividade seguinte do *workflow*.

O conjunto de actividades de *workflow* foi implementado como uma extensão de uma classe abstracta que fornece o método `execute`, o qual permite executar a actividade, como por exemplo, enviar um *email*, SMS ou actualizar o estado no *Facebook* (Figura 6.30).

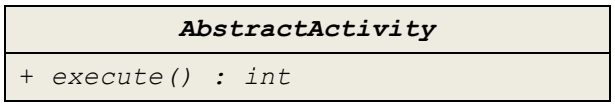


Figura 6.30. Classe abstracta que representa uma actividade

6.5.4 User Interface

Esta camada disponibiliza toda a interacção com o utilizador através dos écrans desenhados que fornecem informação nas mais variadas formas, como por exemplo texto, imagem, vídeo e realidade aumentada. São ainda disponibilizadas tecnologias de áudio, texto falado e reconhecimento de voz, estando esta camada divida de acordo com os tipos de serviço disponibilizados ao utilizador.

No cliente Android, a maioria dos serviços disponibilizados pelo IVO pode ser acedida através do botão de menu destes dispositivos, enquanto outros são acedidos através da navegação ao longo dos diversos écrans. A Figura 6.31 ilustra o menu do IVO aberto sobre o serviço de mapas (ver "*Maps and Navigation Service*") com uma aplicação de guia turístico carregada.

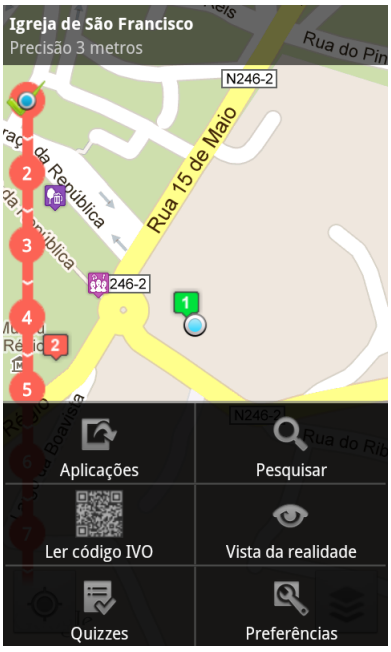


Figura 6.31. Menu do IVO

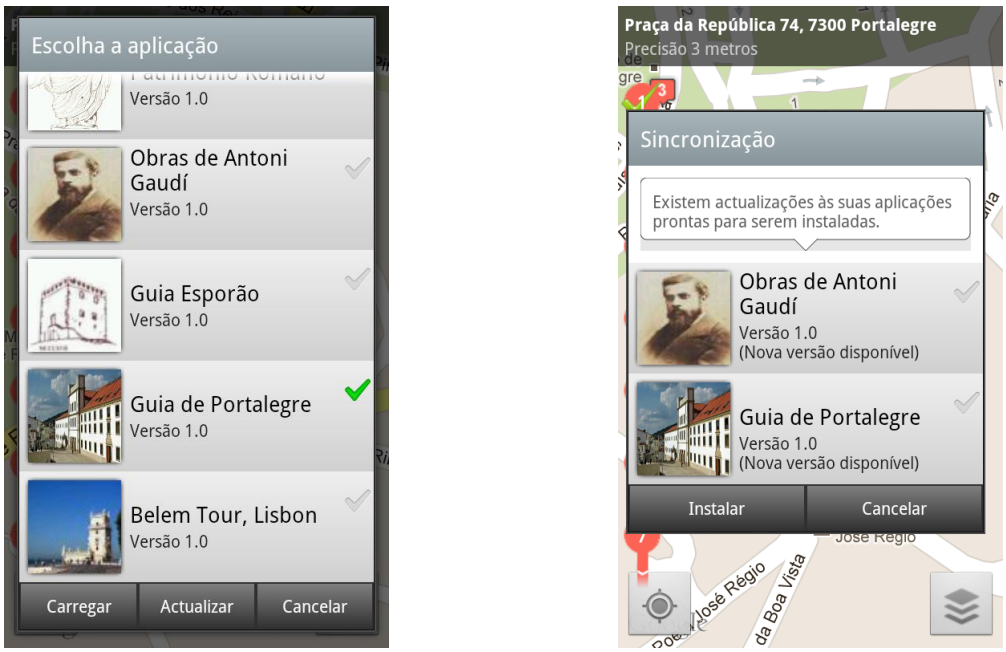
As opções disponíveis neste menu dão acesso a:

- Aplicações (*Applications Service*);
- Pesquisar (*Search Service*);
- Ler código IVO (*QR-Code and NFC Service*) – opção só disponível se na aplicação existir uma condição de estado que faça uso da variável `[STATE.QRCODE]` (ver Secção 6.2.1.2);
- Ler *tag* NFC (*QR-Code and NFC Service*) – opção só disponível se na aplicação existir uma condição de estado que faça uso da variável `[STATE.NFC]` (ver Secção 6.2.1.2);
- Vista da realidade (*Augmented Reality Service*) – opção só disponível se a aplicação tiver sido configurada para usar a realidade aumentada (ver Secção 4.3 referente à definição de permissões das aplicações);
- Formulários e Quizzes (*Forms and Quizzes Service*) – opção só disponível se a aplicação contiver formulários ou quizzes; se só contiver quizzes o texto do menu é "Quizzes" (como no exemplo da Figura 6.31); se só contiver formulários o texto do menu é "Formulários"; se contiver ambos o texto é "Formulários e Quizzes";
- Preferências (*Preferences Service*).

De seguida são detalhados todos os serviços disponibilizados pelo IVO.

Applications Service. Este serviço permite efectuar a gestão das aplicações a que o utilizador tem acesso. A opção do menu "Aplicações", permite seleccionar de entre as que estão instaladas, a que o utilizador quer correr (Figura 6.32a).

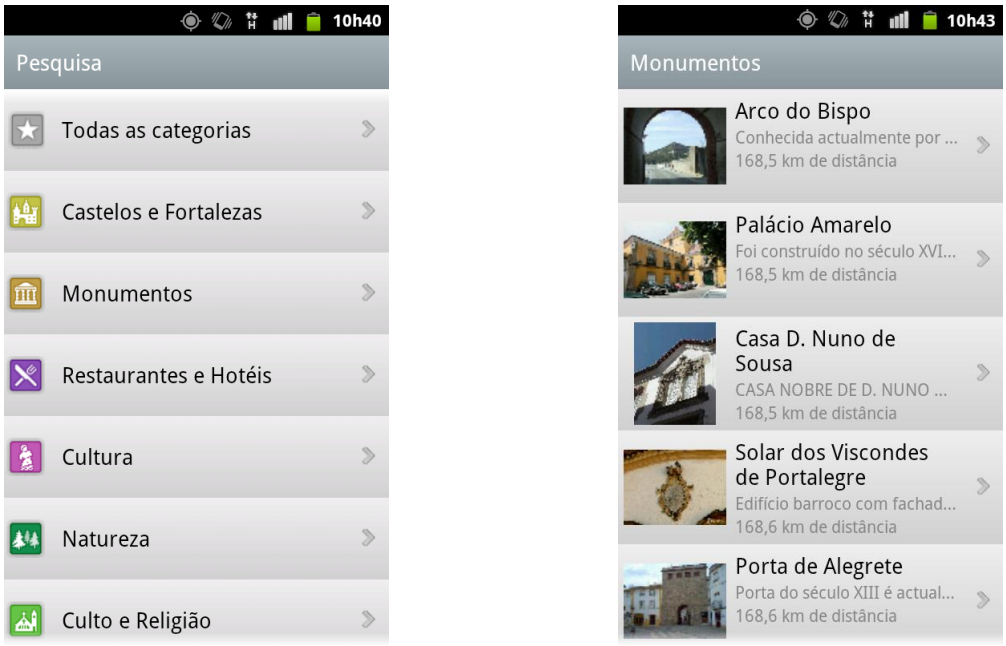
Sempre que existir uma nova aplicação ou uma actualização a uma aplicação que o utilizador já tenha instalado, o utilizador é avisado podendo escolher as aplicações que quer instalar ou actualizar. A Figura 6.32b ilustra o écran com actualizações a duas aplicações instaladas no dispositivo móvel do utilizador.



a) Selecção da aplicação a correr **b) Instalar/actualizar aplicações**

Figura 6.32. Correr, instalar/actualizar aplicações

Search Service. Este serviço permite a pesquisa de cenários de acordo com a sua categoria. A Figura 6.33a ilustra o écran com as categorias existentes numa aplicação de guia turístico e a Figura 6.33b o écran com o resultado dos contextos encontrados na categoria "Monumentos". No caso de contextos que possuam a dimensão de localização, os resultados são ordenados por ordem crescente da distância a esse ponto.



a) Selecção da categoria **b) Contextos encontradoss**

Figura 6.33. Pesquisa de contextos

QR-Code and NFC Service. As opções do menu do IVO que permitem a leitura de QR-Codes e *tags* de texto NFC, ficam disponíveis quando existir na aplicação uma condição de estado que faça uso da variável `[STATE.QRCODE]` ou da variável `[STATE.NFC]` respectivamente. Os QR-Codes podem ser gerados por um dos inúmeros sites *web*, como por exemplo o http://zxing.appspot.com/generator. A leitura destes códigos é feita através da aplicação gratuita "*Barcode Scanner*", a qual permite a integração com aplicações externas. Esta é lançada a partir da opção do menu IVO, sendo o valor do código lido retornado sempre que for detectado um código válido.

A Figura 6.34 ilustra a definição de um contexto de estado para a leitura de um QR-Code ou de uma *tag* NFC com o texto "Toada de Portalegre". Ao ser lido o código correspondente à expressão, o *workflow* que estiver associado a esta situação é iniciado.

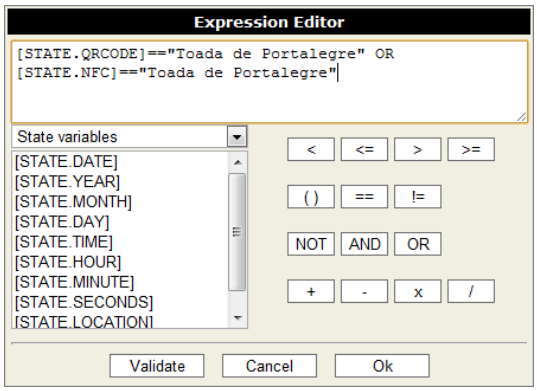


Figura 6.34. Contexto de estado para leitura de um QR-Code ou uma tag NFC

Augmented Reality Service. O serviço de realidade aumentada permite aos utilizadores visualizar informação aumentada do ambiente ao seu redor, sobrepondo às imagens capturadas pela câmara do dispositivo móvel, informação referente aos contextos de localização existentes na aplicação IVO. Este serviço pode usar ainda como fontes de informação o *Panoramio*, a *Wikipedia* e a *Wikimapia*.

Em Graça *et al.* [104] apresentam-se conceitos e questões-chave que surgem no desenvolvimento de um sistema de realidade aumentada para *smartphones*, sendo ainda apresentada a arquitectura de referência que foi utilizada para implementar este serviço no IVO, a qual se organiza nos seguintes componentes:

1. **Percepção.** Responsável por toda a percepção do ambiente em que o utilizador se encontra, incluindo a imagem do mundo real capturada através da câmara do dispositivo móvel, a localização dada pelo GPS, a orientação obtida pela bússola e ainda a inclinação do dispositivo móvel obtida a partir do acelerómetro;

2. **Renderização.** Responsável pela apresentação no écran do dispositivo móvel da imagem capturada pela câmara sobreposta com diversa informação de contexto;

3. **Interacção.** Recolhe e processa qualquer entrada que o utilizador faça deliberadamente, incluindo a percepção de gestos sobre o écran.

A Figura 6.35a ilustra a vista de realidade aumentada com dois contextos de localização do IVO sobrepostos e ainda informação sobre o estado do tempo no local. É possível a partir deste écran partilhar o que o utilizador está a ver no seu dispositivo móvel (Figura 6.35b).

a) Realidade aumentada **b) Imagem partilhada através da realidade aumentada**

Figura 6.35. Serviço de realidade aumentada

Forms and Quizzes Services. Os *quizzes* e os formulários são executados a partir das respectivas actividades de *workflow* ou então manualmente a partir do menu do IVO.

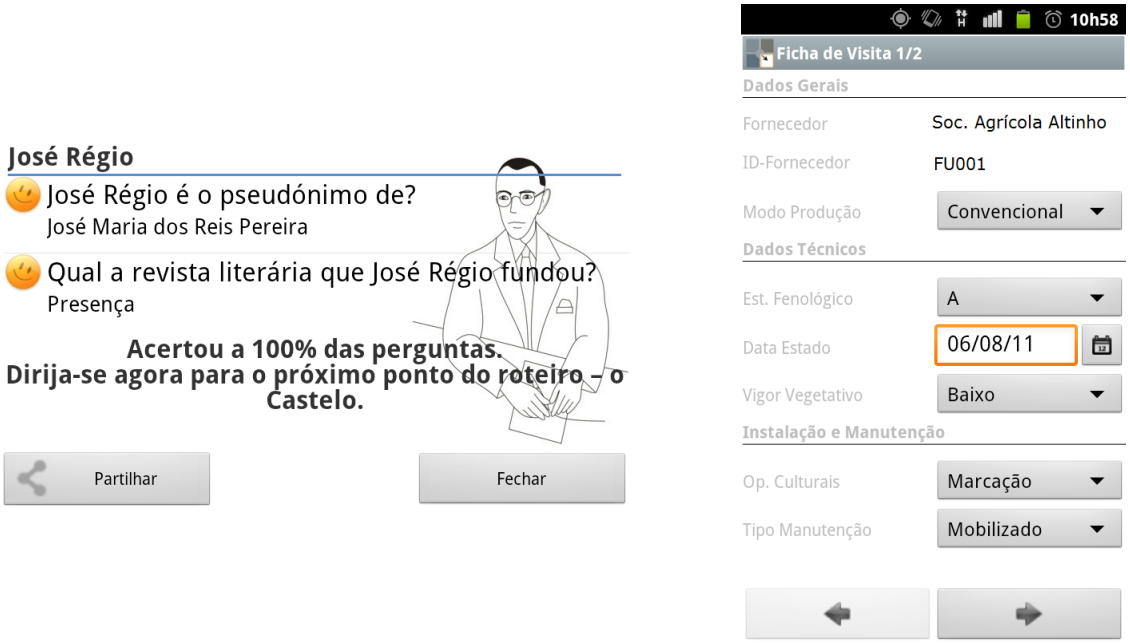
Os *quizzes* representam uma forma interessante e divertida de enriquecer a experiência dos utilizadores em aplicações na área de lazer, como os guias turísticos ou os *peddy-papers*. Tipicamente o *quiz* é executado depois do utilizador visitar determinado local e está relacionado com a visita real, visando essencialmente estimular uma participação mais activa do utilizador. Os *quizzes* podem ser falados através de TTS (*text-to-speech*). Na implementação efectuada neste trabalho, foi usado o motor TTS da SVOX Mobile Voices[29] para as vozes portuguesas e o motor TTS livre disponível com o Android para as vozes inglesas e espanholas.

A Figura 6.36a ilustra o *feedback* final no preenchimento de um *quiz*, podendo o utilizador partilhar este écran através do serviço de partilha.

Ao contrário dos *quizzes*, os formulários destinam-se a uma utilização em aplicações mais na área de negócio ou empresarial e permitem a introdução de dados que podem ser guardados quer no *IVO Builder*, quer num servidor através da execução de um *web-service*. A Figura 6.36b ilustra um

[29] http://svoxmobilevoices.wordpress.com

formulário utilizado numa aplicação do sector vitivinícola para preenchimento, por parte dos técnicos agrícolas, de uma ficha fitossanitária.



a) Feedback final do preenchimento de um quiz

b) Formulário utilizado numa aplicação do sector agrícola

Figura 6.36. Exemplo de (a) um quiz e (b) de um formulário

User Preferences. As preferências do utilizador permitem definir quais as actividades usadas nos *workflows* que podem ser executadas automaticamente. Quando a aplicação é carregada é solicitado ao utilizador que defina as actividades que quer ver automatizadas, conforme mostrado na Figura 6.37. Este écran lista todas as actividades de *workflow* utilizadas pela aplicação, permitindo ao utilizador activar ou desactivar cada uma delas. Por exemplo, um utilizador pode não querer actualizações no *Facebook* ou tocar áudio, pois pode distrair-se ao atravessar as ruas ou simplesmente porque não tem um auscultador.

Figura 6.37. Preferências do utilizador

Maps and Navigation Service. Este serviço possibilita a orientação do utilizador através de um mapa e fornece ainda navegação com indicações de direcção entre o ponto actual e um outro de destino. O serviço de mapas (Figura 6.38a) consiste num mapa ao qual foram adicionadas camadas com os contextos de localização definidos na aplicação (marcados no mapa com o icon associado ao tipo de contexto), artigos geo-referenciados da *Wikipedia* e as áreas definidas de *Wikimapia*.

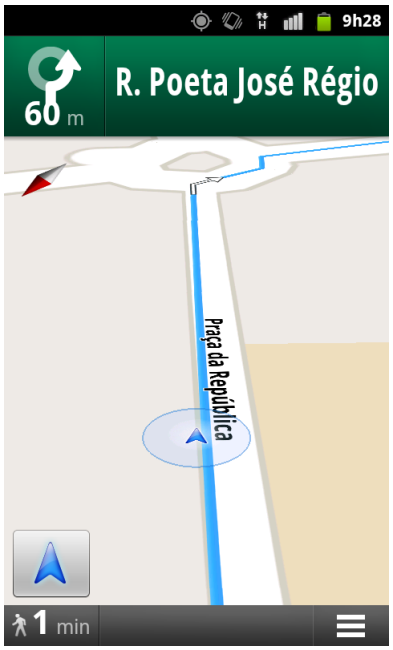
Do lado esquerdo do mapa aparece um roteiro. O IVO não possui a noção de roteiro, contudo este pode ser inferido a partir das actividades do tipo "*Navigate to*" quando estas estão associadas a *workflows* de saída de contextos de localização (ver Secção 6.2.4). Os contextos que fazem parte do roteiro ilustrado na Figura 6.38a surgem no mapa com um icon com um número dentro que se refere ao ponto do roteiro. Todos os pontos do roteiro já visitados pelo utilizador aparecem com um marcador e da cor verde (no exemplo o ponto um já foi visitado). Na Figura 6.38b é ilustrada a informação de contexto referente ao ponto dois do roteiro. A informação que aqui surge é a mesma que surge na Figura 6.40a.

a) Vista de mapa

b) Informação de contexto

Figura 6.38. Serviço de mapas

O serviço de navegação com as indicações de direcção é fornecido pela aplicação *Google Maps Navigation* (Figura 6.39a e Figura 6.39b). Esta aplicação, pode ser lançada automaticamente pelos *workflows* que usam actividades do tipo "*Navigate to*", ou pode ser iniciada manualmente pelo utilizador quando este pressiona os botões disponíveis nos écrans com informação de contexto como nos exemplos da Figura 6.40a e Figura 6.38b. Em qualquer dos casos, o ponto de destino é automaticamente definido pelo IVO.

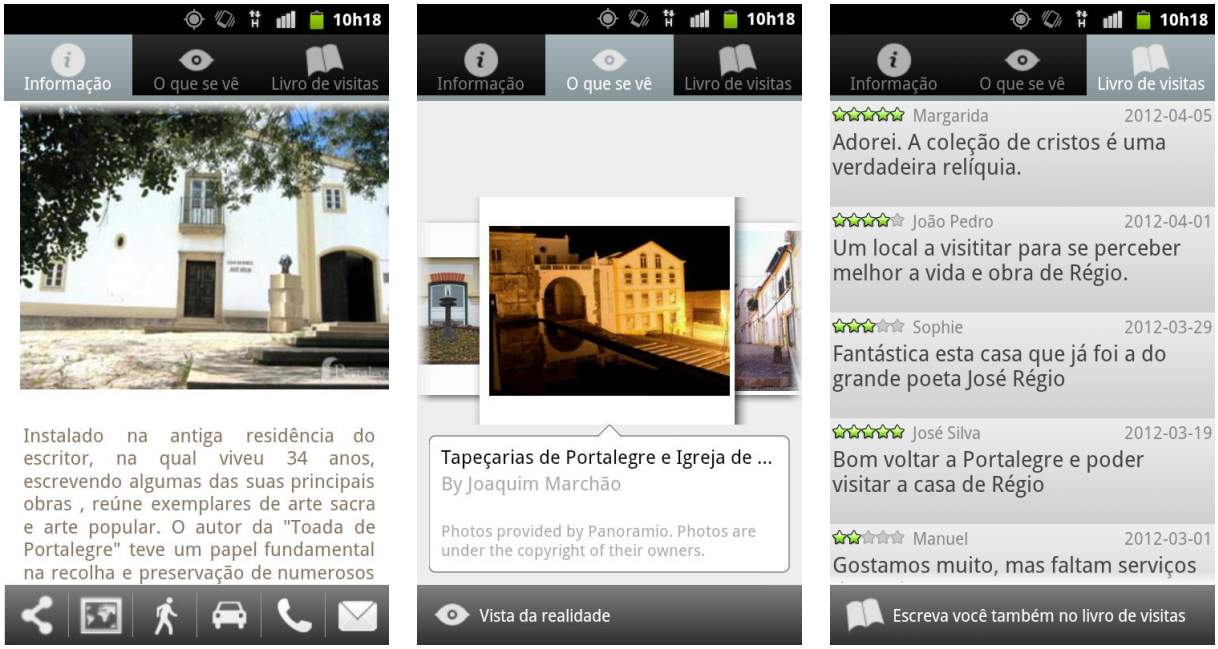


a) Indicações de direcção pedestre

b) Lista de indicações de direcção pedestre

Figura 6.39. Serviço de navegação

Contextual Information Service. Este serviço fornece informação contextual ao utilizador quando é verificada a condição relativa à situação a que o contexto se aplica ou então através do serviço de pesquisa. Esta informação inclui a que foi definida no *IVO Builder* (descrição, contactos e facilidades), e, se for uma aplicação que faça uso do GPS, informação geo-referenciada obtida através do Panoramio (o que se vê nas proximidades) e da previsão do tempo para o local. A Figura 6.40 ilustra a informação contextual tal como é mostrada no dispositivo móvel da aplicação de guia turístico desenhada na Figura 6.10.



a) Descrição do contexto **b) O que se pode ver nas proximidades** **c) Livro de visitas**

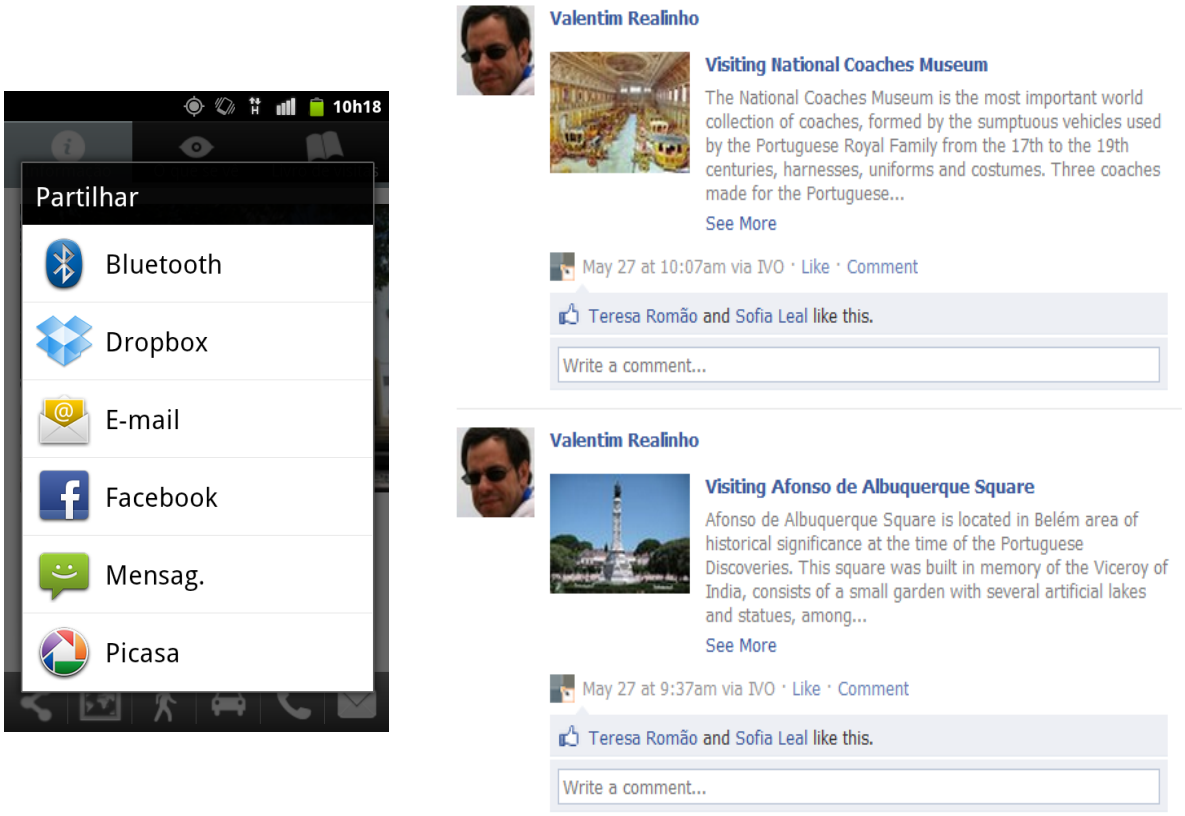
Figura 6.40. Exemplos de informações contextuais associadas a contextos

A informação relativa à descrição, contactos e facilidades é mostrada na Figura 6.40a (parte da informação mostrada na imagem está escondida sendo necessário deslocar o écran). Os botões existentes na parte inferior deste écran possibilitam: (i) a partilha desta informação através dos meios habituais de partilha nos dispositivos móveis como seja o *email*, *Facebook*, *dropbox*, entre outros; (ii) a localização no mapa do ponto de interesse; (iii) (iv) obter indicações de direcção a pé e de carro; (v) telefonar para o número de contacto do ponto de interesse; e (vi) enviar um *email* para o contacto. A Figura 6.40b (O que se vê) mostra a informação geo-referenciada obtida a partir do Panoramio permitindo ainda aceder à vista da realidade através da funcionalidade de realidade aumentada descrita no ponto seguinte. Finalmente, a Figura 6.40c mostra o livro de visitas com a avaliação e os comentários dos utilizadores relativos a este ponto de interesse.

Share Service. Este serviço possibilita a partilha de informação através dos meios disponíveis nos dispositivos móveis. A partilha pode ser feita de forma automática nas redes sociais

(actualmente apenas no *Facebook* e *Twitter*) através da inclusão de actividades de *Facebook* e *Twitter* no *workflow*, tal como no caso ilustrado na Figura 6.17. A Figura 6.41b ilustra o mural do *Facebook* de um utilizador com estas mensagens automáticas.

A partilha pode ainda ser feita de forma manual através do botão de partilha existente em vários écrans tal como o ilustrado na Figura 6.40a. Este botão dá acesso ao écran de partilha manual de informação, onde são listadas todas as formas de partilha disponibilizadas pelo dispositivo móvel (Figura 6.41a).



a) Partilha manual **b) Partilha automática no mural do Facebook do utilizador**

Figura 6.41. Partilha de informação de contexto

Capítulo 7

Avaliação da Plataforma

Conteúdo

Resumo

Este capítulo apresenta a avaliação da plataforma, sendo descritos os testes realizados aos seus diversos componentes, começando pelas ferramentas de composição (*IVO Builder*, *IVO Calendar* e *IVO Contacts*) para depois nos centrarmos na avaliação do cliente (*IVO Client*). Para cada teste realizado, são apresentados todos os procedimentos elaborados e executados, sendo de seguida apresentados os resultados.

7.1 Introdução

A avaliação da plataforma realizou-se na fase final do desenvolvimento com protótipos perfeitamente funcionais dos vários componentes do sistema. Pretendeu-se com estes testes identificar constrangimentos, testando todo o sistema perante as necessidades e práticas dos utilizadores. Os objectivos principais a atingir com esta avaliação foram:

1. **Avaliar as funcionalidades do sistema.** Avaliar até que ponto o sistema responde aos requisitos identificados. Por outras palavras, o desenho do sistema deve permitir aos utilizadores efectuar as diversas tarefas mais facilmente. Isto inclui não só disponibilizar as

funcionalidades identificadas, mas também torná-las facilmente acessíveis ao utilizador em termos dos passos que necessita executar para concluir cada tarefa.

2. **Avaliar a experiência do utilizador na interacção com o sistema.** Além de avaliar o desenho do sistema em termos das suas capacidades funcionais, é importante também avaliar a usabilidade e o impacto do sistema sobre o utilizador. Isto inclui aspectos como a facilidade de aprendizagem, a facilidade de utilização e a satisfação do utilizador ao usar o sistema. Pode também incluir o prazer e a resposta emocional, o que é particularmente relevante no caso de sistemas destinados ao lazer e ao entretenimento. É importante ainda identificar características do sistema que de alguma forma sobrecarregam o utilizador, como por exemplo a necessidade de lembrar uma quantidade excessiva de informação ou passos para executar uma determinada tarefa.

3. **Identificar eventuais problemas com o sistema.** O objectivo final da avaliação é identificar problemas específicos de desenho do sistema, que quando usados no contexto pretendido, causam resultados inesperados ou confusão ao utilizador. Isto está relacionado não só com as funcionalidades, mas também com questões de usabilidade.

7.2 Composite Tools

7.2.1 IVO Builder

O *IVO Builder* foi testado através de testes de usabilidade, os quais ajudaram a observar a interacção dos utilizadores com as ferramentas disponibilizadas. Estes testes permitiram determinar e compreender os problemas de usabilidade desta ferramenta, tendo os resultados sido apresentados e publicados como *full paper* nas actas da conferência *Human-Computer Interaction - INTERACT 2011* [105].

7.2.1.1 Participantes

O *IVO Builder* foi testado por dois grupos de utilizadores com formação base distinta. Um grupo de onze estudantes de mestrado em informática (dois deles do sexo feminino) e um grupo de seis estudantes de licenciatura em *design* (um deles do sexo feminino). Pretendia-se com estes dois grupos avaliar em que medida a área de formação dos utilizadores poderia influenciar os resultados. A idade dos participantes neste teste variou entre os 20 e os 36 anos no grupo 1 (média de 23.8) e entre os 21 e os 43 anos no grupo 2 (média 27.8).

		Nº Utilizadores			Idade		
		M	F	Total	Min	Máx.	Méd.
G1	Formação base de informática	9	2	11	20	36	23.8
G2	Formação base de *design*	5	1	6	21	43	27.8

Tabela 7.1. Participantes nos testes do IVO Builder

7.2.1.2 Metodologia

Todos os participantes tiveram o primeiro contacto com o *IVO Builder* durante o teste e usaram-no em condições semelhantes.

As sessões de testes foram realizadas individualmente para cada utilizador com a colaboração de dois investigadores, que desempenharam o papel de facilitador e de observador.

Antes de começar a usar o sistema, os utilizadores foram informados sobre os objectivos do teste, e foi fornecida uma lista descrevendo as tarefas que deveriam executar, não tendo sido feita nenhuma referência acerca de como estas poderiam ser realizadas.

Os utilizadores foram incentivados a "pensar em voz alta" e a verbalizarem o que estavam a pensar durante a realização dos testes. Desta forma, os utilizadores puderam explorar livremente as interfaces e foi possível identificar mais claramente os problemas de usabilidade.

Durante os testes, o observador tomou notas sobre o comportamento e os comentários dos utilizadores durante a realização das tarefas.

Após a realização dos testes os utilizadores responderam a um questionário que capturou dados pessoais, *feedback* experimental, bem como sugestões e comentários.

7.2.1.3 Questionário

Os dados pessoais do questionário incluíam a idade, sexo, nível de escolaridade, a familiaridade com novas tecnologias e a frequência de uso de Internet, computador, telemóvel e consola de jogos. O *feedback* experimental foi avaliado através das cinco secções de seguida detalhadas.

1. **Facilidade de aprendizagem.** A avaliação da facilidade de aprendizagem do *IVO Builder* foi feita através de três questões (Tabela 7.2). Os utilizadores manifestaram o seu nível de concordância com cada questão numa escala de *Likert* de 5 pontos, com uma resposta de 1 (um) significando "discordo totalmente" e uma resposta de 5 (cinco) significando "concordo totalmente".

Facilidade de aprendizagem	
Q1	É fácil aprender a usar a ferramenta
Q2	Fiquei rapidamente habilitado a usar a ferramenta
Q3	Lembro-me facilmente de como usar a ferramenta

Tabela 7.2. Facilidade de aprendizagem (IVO Builder)

2. **Facilidade de utilização.** O questionário incluiu sete questões relativas à facilidade de utilização do *IVO Builder* (Tabela 7.3). Para avaliar estas questões os utilizadores responderam usando o mesmo procedimento usado para a primeira secção tal como descrito no ponto um.

Facilidade de utilização	
Q4	A ferramenta é fácil de utilizar
Q5	A ferramenta requer poucos passos para realizar o que se pretende
Q6	A ferramenta pode ser utilizada facilmente para a construção de outras aplicações, cenários ou situações
Q7	Consigo usar a ferramenta sem necessitar de instruções escritas
Q8	Não detectei inconsistências de utilização da interface
Q9	Foi fácil recuperar de uma situação inesperada
Q10	A ferramenta respondeu rapidamente às diversas acções do utilizador

Tabela 7.3. Facilidade de utilização (IVO Builder)

3. **Facilidade de execução das tarefas propostas.** A terceira secção do questionário incluiu perguntas sobre a facilidade de executar as tarefas propostas (Tabela 7.4). Para responder a esta pergunta, foi usada uma escala de 1 (um), significando "muito difícil", a 5 (cinco) significando "muito fácil".

Facilidade de execução das tarefas propostas
Q11.a Criar aplicação
Q11.b Criar um contexto de localização através da importação da *Wikipedia*
Q11.c Criar um contexto de localização através da importação da *Wikimapia*
Q11.d Criar um contexto de localização manualmente (polígono)
Q11.e Alterar o polígono que define um contexto de localização
Q11.f Construir um *quiz*
Q11.g Construir um formulário
Q11.h Desenhar *workflow*

Tabela 7.4. Facilidade de execução das tarefas propostas (IVO Builder)

4. **Avaliação das actividades de workflow.** Na quarta secção do questionário, os utilizadores foram solicitados a indicar as três actividades de *workflow* que consideraram mais úteis e as três que consideravam menos úteis.

5. **Envolvimento emocional.** A quinta secção inclui uma pergunta baseada na *Microsoft Product Reaction Cards*. Esta secção visou captar o envolvimento emocional dos utilizadores ao utilizarem o sistema, uma vez que facilitam a mensuração de aspectos intangíveis da experiência do utilizador [106]. Foi solicitado aos utilizadores que escolhessem as palavras que melhor descreveram a sua experiência ao usar o sistema a partir de uma lista de 24 palavras, sendo que 60% dessas palavras exprimem sentimentos positivos e 40% exprimem sentimentos negativos. Os utilizadores podiam escolher quantas palavras quisessem. A Tabela 7.5 ilustra as palavras disponíveis pela ordem com que estas foram apresentadas no questionário.

Envolvimento emocional		
Agradável	Simples	Cansativa
Frustrante	Divertida	Imersiva
Estimulante	Aborrecida	Irritante
Complexa	Inovadora	Entusiasmante
Confusa	Impressionante	Motivante
Péssima	Óptima	Aspecto profissional
Satisfatória	Muito técnica	Sem valor
Útil	Viciante	Stressante

Tabela 7.5. Avaliação do envolvimento emocional do utilizador (IVO Builder)

Finalmente, o questionário incluiu ainda uma questão aberta, a fim de recolher comentários e recomendações quanto a desenvolvimentos e evolução futura de funcionalidades e para obter uma avaliação mais geral do sistema.

7.2.1.4 Cenário de Teste

O cenário de teste utilizado nesta avaliação foi elaborado de forma a permitir o teste de grande parte das funcionalidades disponíveis no sistema. Neste cenário, foi solicitado aos utilizadores que criassem uma aplicação para guiar turistas através de diversos pontos de interesse da cidade de Lisboa usando o *IVO Builder*. Pretendia-se que quando os turistas chegassem aos pontos de interesse, lhes fosse fornecida informação sobre esses pontos usando a *Wikipedia* como fonte de informação. O telefone também deveria ser colocado no modo silencioso e o estado do utilizador no *Facebook* devia ser actualizado informando os amigos que ele se encontrava a visitar o local. No final da visita a um ponto de interesse, o perfil do telefone devia ser alterado para o modo

normal e o Navegador deveria guiar o turista para o próximo ponto de interesse. Em certos pontos, o turista deveria ser desafiado a responder a perguntas sobre o ponto de interesse que acabara de visitar. No final, o turista deveria responder a um questionário electrónico para avaliar a experiência do roteiro.

7.2.1.5 Resultados

As respostas dos utilizadores às questões apresentadas no questionário foram analisadas sendo e os resultados mostrados de seguida.

Foram examinadas as médias das respostas dos utilizadores para ver se houve tendências nas opiniões (opiniões fortes num sentido ou noutro surgindo com valores médios mais próximos de 1 ou de 5). Foi ainda utilizado o desvio padrão para determinar o consenso das respostas.

Para analisar em que medida a formação base dos utilizadores influencia os resultados, foram feitos testes *t* de *Student* sobre as duas amostras. Os testes *F* foram utilizados para verificar a igualdade das variâncias dos dois grupos de forma a determinar o tipo de teste *t* apropriado.

Todos os utilizadores foram capazes de realizar as tarefas propostas, embora tenham ocorrido algumas dificuldades ocasionais, especialmente quando solicitados a alterar o polígono que define uma área de interesse. O tempo para completar o teste variou entre 28 minutos e 1 hora e 5 minutos.

Facilidade de aprendizagem. A Tabela 7.6 apresenta a estatística descritiva dos resultados obtidos a estas questões bem como os resultados dos testes *F* e dos testes *t* de *Student* para cada questão.

		Média	Variância	Desvio Padrão	Teste *F*		Teste *t* de Student	
					$F_{(10,5)}$	*p*	$t_{(2,15)}$	*p*
Q1	G1	3.63636	0.65455	0.80904	3.27273	0.26444	-1.44270	0.17110
	G2	4.20000	0.20000	0.44721				
Q2	G1	3.90909	0.89091	0.94388	4.45455	0.16289	-0.64767	0.52768
	G2	4.20000	0.20000	0.44721				
Q3	G1	4.00000	0.80000	0.89443	1.00000	0.90311	-0.82916	0.42093
	G2	4.40000	0.80000	0.89443				

Tabela 7.6. Resultados da facilidade de aprendizagem (IVO Builder)

A análise dos resultados dos testes *F* permitiu concluir pela não rejeição da hipótese nula H_0 das variâncias serem iguais ($p>0.05$), tendo por isso sido utilizados testes *t* de *Student* para a diferença de médias de duas amostras com variâncias desconhecidas mas iguais.

Também a análise dos testes *t* permitiu concluir pela não rejeição da hipótese nula H_0 das médias das duas amostras serem iguais (p>0.05), querendo dizer não haver diferenças estatisticamente significativas entre os dois grupos no que diz respeito à facilidade de aprendizagem, apesar de o grupo de *design* (G2) avaliar melhor todas as questões desta secção.

A maioria dos utilizadores considerou ser fácil lembrar de como usar o *IVO Builder* (Q3) e que rapidamente se tornaram hábeis a utilizá-lo (Q2). A questão Q1: "É fácil aprender a usar a ferramenta", recebeu a avaliação mais baixa ainda que bastante positiva (3.6 ± 0.80904).

Globalmente, e considerando a média das respostas de todos os utilizadores às questões desta secção, estes avaliaram a facilidade de aprendizagem com 4.0 ± 0.81187.

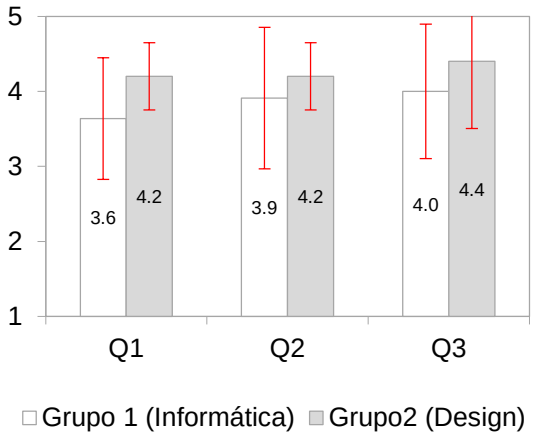


Figura 7.1. Facilidade de aprendizagem (IVO Builder)

Facilidade de utilização. A Tabela 7.7 apresenta a estatística descritiva dos resultados obtidos a estas questões bem como os resultados dos testes *F* e dos testes *t* de *Student* para cada questão.

		Média	Variância	Desvio Padrão	Teste *F*		Teste *t* de Student	
					$F_{(10,5)}$	p	$t_{(2,15)}$	p
Q4	G1	3.90909	0.89091	0.94388	4.45455	0.16289	0.24288	0.81162
	G2	3.80000	0.20000	0.44721				
Q5	G1	3.45455	0.87273	0.93420	0.67133	0.55398	-0.27038	0.79081
	G2	3.60000	1.30000	1.14018				
Q6	G1	3.90909	0.49091	0.70065	0.70130	0.58974	0.27257	0.78917
	G2	3.80000	0.70000	0.83666				
Q7	G1	3.36364	1.25455	1.12006	2.50909	0.38910	0.66144	0.51907
	G2	3.00000	0.50000	0.70711				
Q8	G1	3.63636	1.05455	1.02691	0.45850	0.28935	0.05677	0.95553
	G2	3.60000	2.30000	1.51658				
Q9	G1	3.90909	1.29091	1.13618	4.30303	0.17229	-0.90664	0.37994
	G2	4.40000	0.30000	0.54772				

Q10	G1	4.09091	0.49091	0.70065	0.37762	0.19292	1.07110	0.30224
	G2	3.60000	1.30000	1.14018				

Tabela 7.7. Resultados da facilidade de utilização (IVO Builder)

Também nesta secção se verificou não haver diferenças estatisticamente significativas entre os dois grupos (p>0.05). Esta evidência é particularmente forte na questão relativa à detecção de inconsistências de utilização da interface (Q8) ($t_{(2,15)}$=0.05677, p=0.95553). Também as questões Q4: "A ferramenta é fácil de utilizar" ($t_{(2,15)}$=0.24288, p=0.81162), Q5: "A ferramenta requer poucos passos para realizar o que se pretende" ($t_{(2,15)}$=0.24288, p=0.79081) e Q6: "A ferramenta pode ser utilizada facilmente para a construção de outras aplicações, cenários ou situações" ($t_{(2,15)}$=0.27257, p=0.78917) apresentam uma evidência estatística bastante significativa.

A maioria dos utilizadores considerou ser fácil recuperar de uma situação inesperada (Q9) (3.9 ± 1.13618 e 4.4 ± 0.54772), ser fácil usar o *IVO Builder* (Q4) (3.9 ± 0.94388 e 3.8 ± 0.44721), que pode ser utilizado facilmente para a construção de outras aplicações, cenários ou situações (Q6) (3.9 ± 0.70065 e 3.8 ± 0.83666) e ainda que o *IVO Builder* respondeu rapidamente às diversas acções do utilizador (Q10) (4.0 ± 0.70065 e 3.6 ± 1.14018). Ainda que positivas, as opiniões dos utilizadores acerca do uso da ferramenta sem instruções escritas (Q7) (3.4 ± 1.12006 e 3.0 ± 0.70711), do número de passos necessários para realizar o que se pretende (Q5) (3.5 ± 0.93420 e 3.6 ± 1.14018), e das inconsistências encontradas quando se utiliza a interface (Q8) (3.6 ± 1.02691 e 3.6 ± 1.51658) foi mais baixa e revelaram menor consenso.

Globalmente, os resultados referentes à facilidade de utilização do *IVO Builder* (Q4 a Q10) é bastante positivo (3.7 ± 0.98116), mas houve menor consenso nas opiniões dos utilizadores traduzido por desvios padrões maiores que um a várias perguntas.

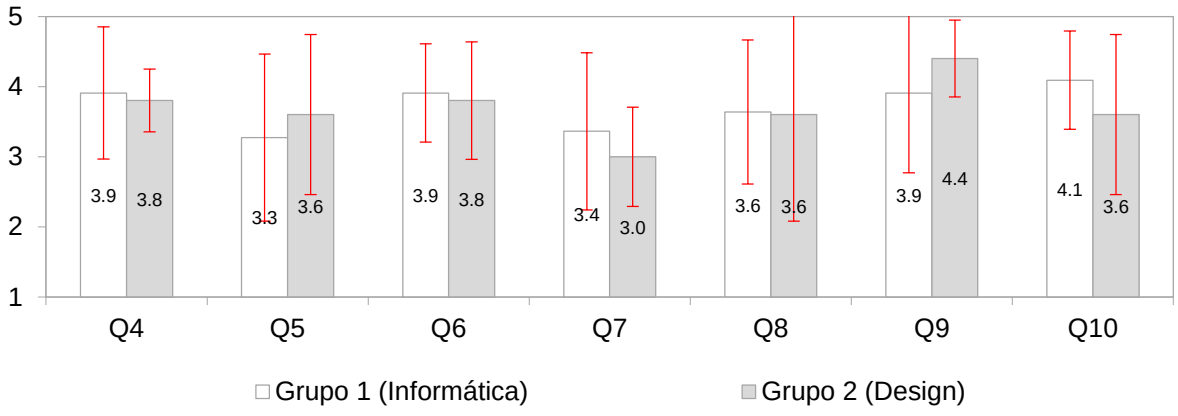


Figura 7.2. Facilidade de utilização (IVO Builder)

Facilidade de execução das tarefas propostas. A Tabela 7.8 apresenta a estatística descritiva dos resultados obtidos a estas questões, bem como os resultados dos testes *F* e dos testes *t* de *Student* para cada questão.

		Média	Variância	Desvio Padrão	Teste *F*		Teste *t* de Student	
					$F_{(10,5)}$	*p*	$t_{(2,15)}$	*p*
Q11.a	G1	4.81818	0.16363	0.40452	0.81818	0.72213	0.08081	0.93674
	G2	4.80000	0.20000	0.44721				
Q11.b	G1	4.00000	0.80000	0.89442	2.66667	0.35716	0.91485	0.37576
	G2	3.60000	0.30000	0.54772				
Q11.c	G1	4.00000	0.80000	0.89442	2.66667	0.35716	1.37228	0.19156
	G2	3.40000	0.30000	0.54772				
Q11.d	G1	4.45454	0.67272	0.82020	0.84091	0.74645	0.12010	0.90611
	G2	4.40000	0.80000	0.89442				
Q11.e	G1	2.63636	0.85454	0.92441	1.22078	0.91689	-3.22040	0.00617
	G2	4.20000	0.70000	0.83666				
Q11.f	G1	4.36363	0.45454	0.67420	1.51515	0.73281	-0.68408	0.50509
	G2	4.60000	0.30000	0.54772				
Q11.g	G1	4.36363	0.65454	0.80904	2.18182	0.47030	-0.58917	0.56513
	G2	4.60000	0.30000	0.54772				
Q11.h	G1	4.00000	0.60000	0.77459	2.00000	0.52645	-1.03414	0.31861
	G2	4.40000	0.30000	0.54772				

Tabela 7.8. Resultados da facilidade de execução das tarefas propostas (IVO Builder)

A maior dificuldade observada em ambos os grupos na execução destas tarefas verificou-se na tarefa de mudar o polígono que define uma área de interesse (Q11.e). Como já se antecipava esta dificuldade, foram preparadas instruções escritas que foram fornecidas aos utilizadores no caso de se verificarem dificuldades na conclusão desta tarefa. Depois de fornecidas estas instruções, os utilizadores foram capazes de realizar a tarefa sem qualquer dificuldade. Esta tarefa revelou uma diferença significativa entre os dois grupos ($t_{(2,15)}$=-3.22040, p<0.00617). O grupo de informática (G1) considerou esta tarefa difícil de executar (2.6 ± 0.92441) enquanto o grupo de *design* (G2) considerou fácil (4.2 ± 0.83666). Uma possível explicação para a diferença entre os dois grupos, é cada um ter interpretado a resposta de forma diferente: o grupo 1 (G1) respondeu considerando as dificuldades que sentiram antes de serem fornecidas as instruções escritas, enquanto o grupo 2 (G2) respondeu considerando a facilidade de execução da tarefa após o fornecimento das instruções.

Não houve diferenças estatisticamente significativas entre os dois grupos no que diz respeito às restantes tarefas (p>0.05). Esta evidência é particularmente forte na tarefa para criar uma nova

aplicação (Q11.a) ($t_{(2,15)}$=0.08081, p=0.93674) e para criar uma área de interesse manualmente (Q11.d) ($t_{(2,15)}$=0.12010, p=0.90611).

De acordo com os resultados apresentados na Tabela 7.8 e na Figura 7.3, as interacções foram geralmente consideradas fáceis de executar (média global 4.1 ± 0.90815). Como já referido, a única excepção foi a tarefa de alterar o polígono que define uma área de interesse (Q11.e), a qual os utilizadores consideraram mais difícil de executar. As opiniões dos utilizadores foram muito heterogéneas e estão em conformidade com o comportamento observado pelo investigador.

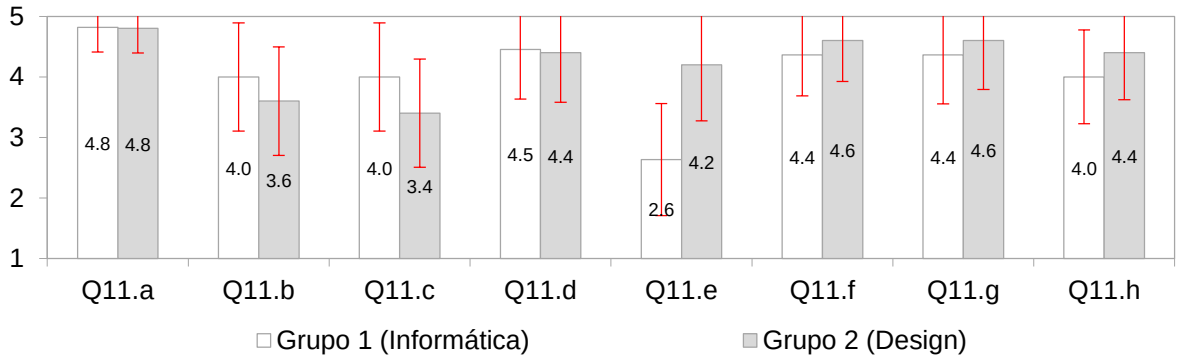


Figura 7.3. Facilidade de execução das tarefas propostas (IVO Builder)

Avaliação das actividades de workflow. Não se verificaram diferenças nas respostas dos dois grupos, parecendo haver um padrão nas respostas quanto às actividades mais úteis. Os utilizadores tendem a referir as actividades utilizadas no teste: "mudar estado do telefone" (14 utilizadores), "ir para local" (12 utilizadores) e "actualizar o estado no *Facebook*" (7 utilizadores) como as actividades mais úteis. As actividades consideradas menos úteis foram "reconhecimento de voz" (7 utilizadores) e "executar um *script*" (4 utilizadores). É de crer que uma vez que os utilizadores não usaram essas actividades durante o teste, poderão não ter percebido a sua utilidade. A Figura 7.4 ilustra as respostas agregadas dos dois grupos de utilizadores.

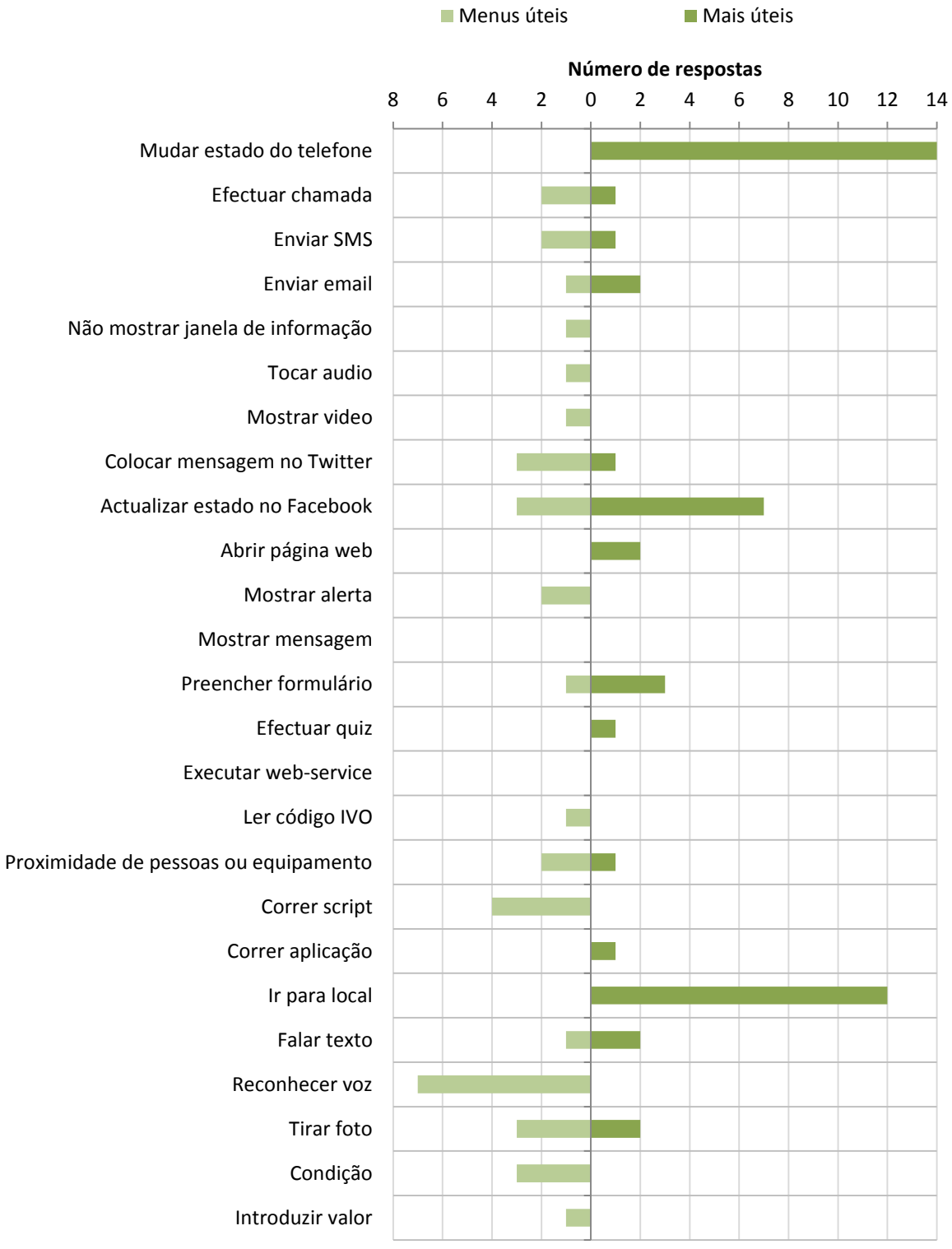


Figura 7.4. Avaliação das actividades de workflow (IVO Builder)

Envolvimento emocional. A partir da análise dos resultados apresentados na Figura 7.5, conclui-se que 90% dos utilizadores tiveram sentimentos positivos ao classificar a sua experiência ao utilizar o sistema. Os sentimentos dominantes dos utilizadores referem-se a considerar o *IVO Builder* uma ferramenta útil (81% dos utilizadores), simples (70% dos utilizadores) e agradável (50% dos utilizadores). O sistema foi considerado profissional e satisfatório por 44% dos

utilizadores e inovador por 38%. Apenas dois utilizadores relataram sentimentos negativos e as palavras usadas foram "confuso", "complexo" e "aborrecido".

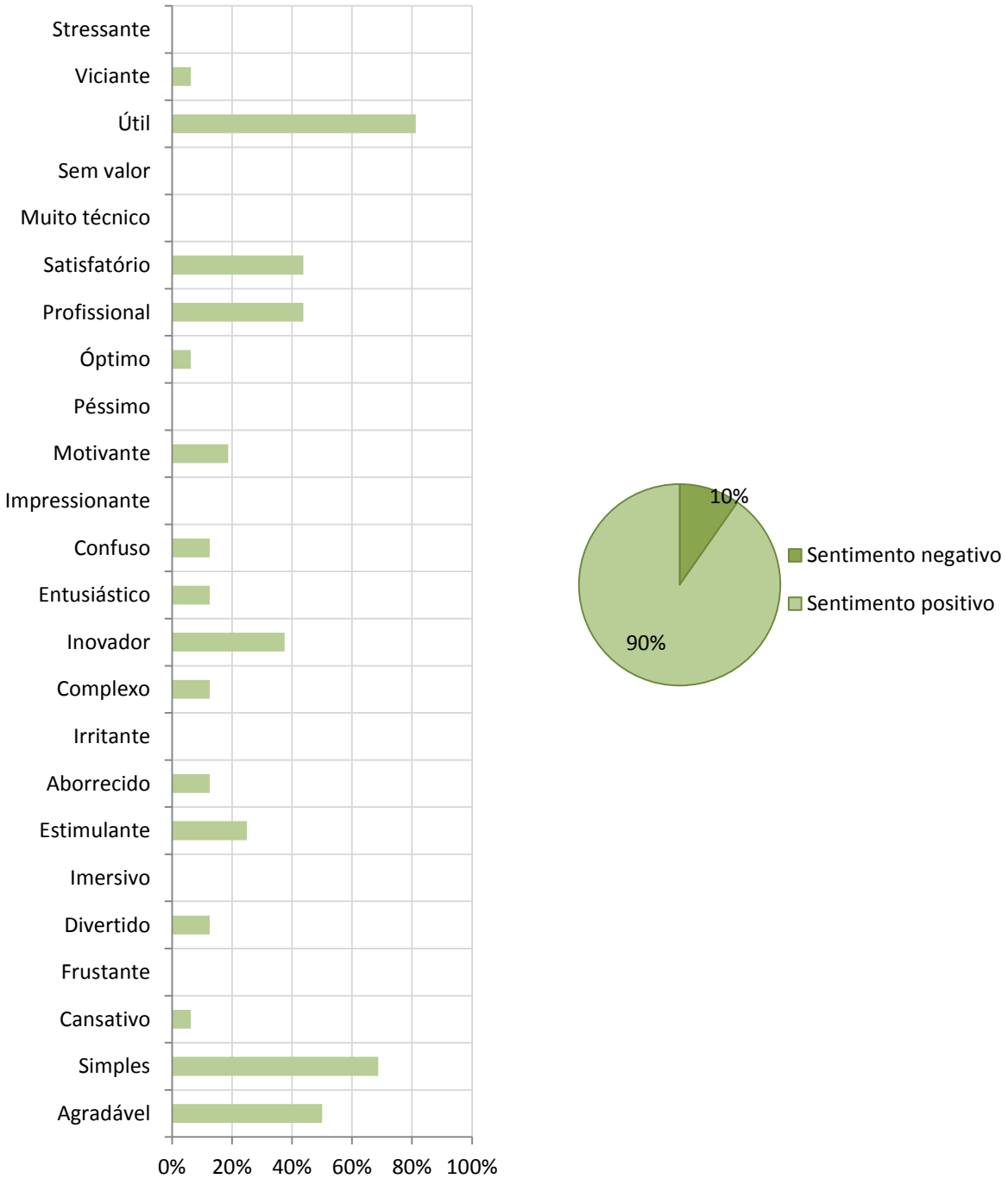


Figura 7.5. Envolvimento emocional dos utilizadores ao usarem o sistema (IVO Builder)

7.2.2 IVO Outlook

Os testes do *IVO Outlook* foram realizados num computador com o *Microsoft Outlook* instalado com os *add-on* do *IVO Calendar* e do *IVO Contacts*.

Para a realização destes testes, foi utilizado o mesmo método utilizado nos testes do *IVO Builder*, tendo o questionário final diferido apenas nas questões referentes à facilidade de execução das tarefas propostas (questões Q11) que neste caso foram:

Facilidade de execução das tarefas propostas
Q11.a Criar um contacto IVO
Q11.b Criar um compromisso IVO
Q11.c Ver contactos e compromissos IVO

Tabela 7.9. Facilidade de execução das tarefas propostas (IVO Outlook)

7.2.2.1 Participantes

Estes testes foram realizados por um grupo de seis participantes (um deles do sexo feminino). A idade dos participantes neste teste variou entre os 21 e os 33 anos com uma média de 24.5.

7.2.2.2 Cenário de Teste

No cenário de teste foi solicitado aos utilizadores para usarem o *IVO Outlook* de forma a criarem contactos e associá-los a um dispositivo *Bluetooth* (o *smartphone*), a fim de permitir actividades do tipo "*pessoas na proximidade*", e ainda a criar compromissos aos quais o utilizador adiciona actividades. O utilizador deve criar um novo compromisso, e criar um *workflow* que será executado no início do compromisso e outro que será no final do compromisso. No início do compromisso, o telefone deve ser colocado no modo silencioso e o estado do utilizador no *Facebook* deve ser actualizado. O telefone também deve falar uma mensagem notificando o utilizador para o início do compromisso. No fim do compromisso o perfil do telefone deve ser alterado para o modo normal.

7.2.2.3 Resultados

Após a realização do teste, os utilizadores responderam ao mesmo questionário utilizado no teste do *IVO Builder* com excepção das questões referentes à facilidade de execução das tarefas propostas (questões Q11) as quais foram as que constam na Tabela 7.9.

As respostas dos utilizadores às questões apresentadas no questionário foram analisadas sendo os resultados mostrados e discutidos de seguida.

Feedback experimental. Todos os utilizadores foram capazes de realizar as tarefas, sem qualquer dificuldade, num período de tempo entre 9 e 16 minutos. O *feedback* dos utilizadores foi bastante positivo e consensual. A pontuação mais baixa, ainda que bastante positiva, verificou-se na questão sobre o uso da aplicação sem instruções escritas (Q7) (3.8 ± 0.75277). Esta é

considerada uma questão muito subjectiva uma vez que os utilizadores não usaram quaisquer instruções escritas e não foi observada qualquer dificuldade quando estavam a realizar as tarefas. A questão referente à execução da tarefa para ver contactos e compromissos IVO (Q11.c) (5.0 ± 0.00000) foi absolutamente consensual tendo recebido a pontuação máxima por parte de todos os utilizadores.

		Média	Variância	Desvio Padrão
Q1	É fácil aprender a usar a ferramenta	4.33333	0.26667	0.51640
Q2	Fiquei rapidamente habilitado a usar a ferramenta	4.66667	0.26667	0.51640
Q3	Lembro-me facilmente de como usar a ferramenta	4.00000	1.60000	1.26491

Tabela 7.10. Facilidade de aprendizagem (IVO Outlook)

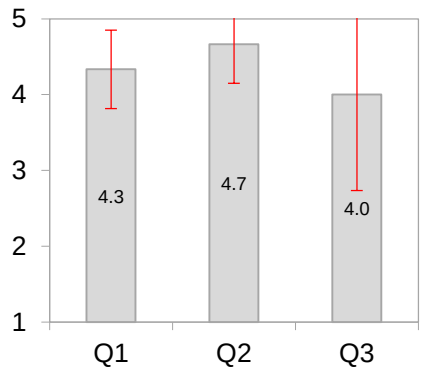


Figura 7.6. Facilidade de aprendizagem (IVO Outlook)

		Média	Variância	Desvio Padrão
Q4	A ferramenta é fácil de utilizar	4.33333	0.26667	0.51640
Q5	A ferramenta requer poucos passos para realizar o que se pretende	4.50000	0.70000	0.83666
Q6	A ferramenta pode ser utilizada facilmente para a construção de outras aplicações, cenários ou situações	4.50000	0.70000	0.83666
Q7	Consigo usar a ferramenta sem necessitar de instruções escritas	3.83333	0.56667	0.75277
Q8	Não detectei inconsistências de utilização da interface	4.16667	1.36667	1.16905
Q9	Foi fácil recuperar de uma situação inesperada	4.83333	0.16667	0.40825
Q10	A ferramenta respondeu rapidamente às diversas acções do utilizador	4.66667	0.26667	0.51640

Tabela 7.11. Facilidade de utilização (IVO Outlook)

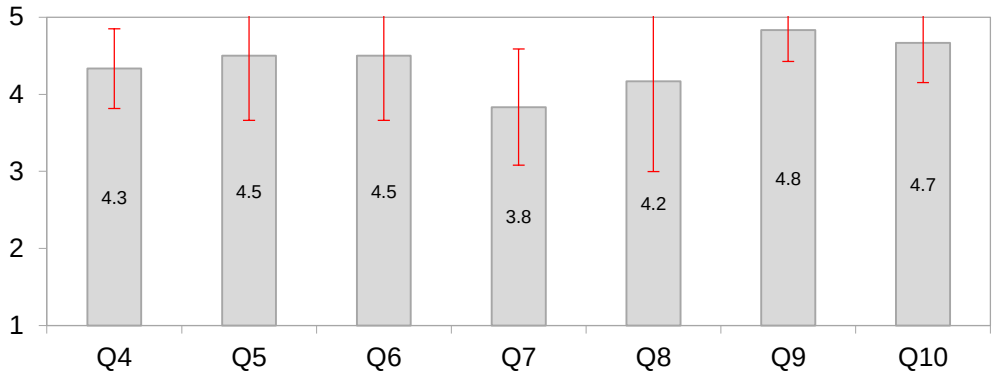


Figura 7.7. Facilidade de utilização (IVO Outlook)

		Média	Variância	Desvio Padrão
Q11.a	Criar um contacto IVO	4.83333	0.16667	0.40825
Q11.b	Criar um compromisso IVO	4.16667	0.56667	0.75277
Q11.c	Ver contactos e compromissos IVO	5.00000	0.00000	0.00000

Tabela 7.12. Facilidade de execução das tarefas propostas (IVO Outlook)

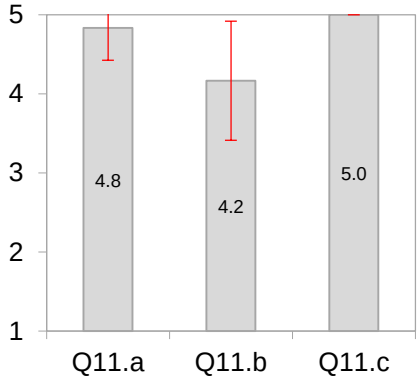


Figura 7.8. Facilidade de execução das tarefas propostas (IVO Outlook)

Avaliação das actividades de workflow. A Figura 7.9 ilustra as respostas dos utilizadores à avaliação das actividades de *workflow*. As actividades consideradas mais úteis foram "mudar estado do telefone" (6 utilizadores), "mostrar alerta" e "executar um *script*" (2 utilizadores). As actividades consideradas menos úteis foram "não mostrar janela de informação" (4 utilizadores), "reconhecimento de voz" (3 utilizadores) e "ler código IVO" (2 utilizadores).

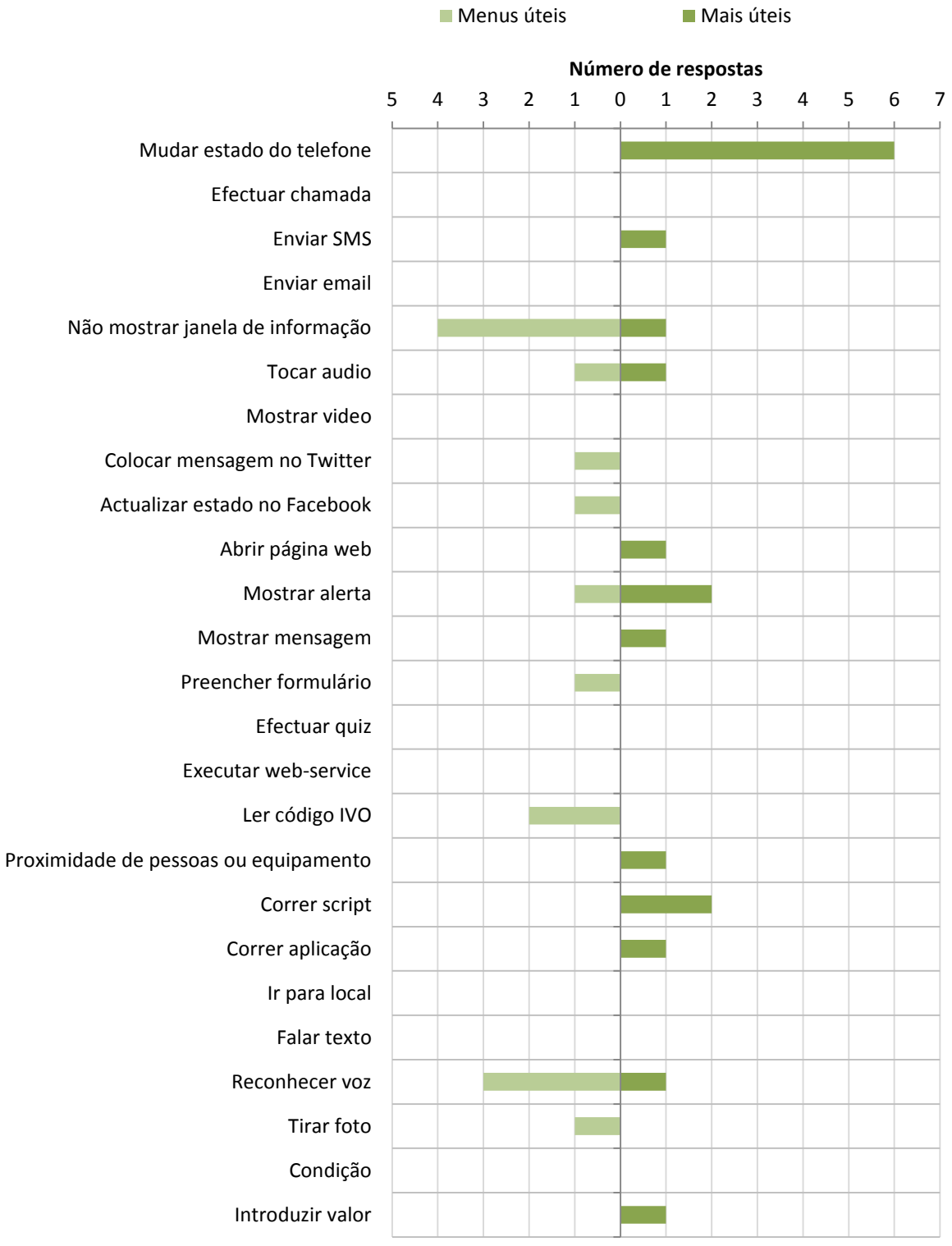


Figura 7.9. Avaliação das actividades de workflow (IVO Outlook)

Envolvimento emocional. A partir da análise dos resultados apresentados na Figura 7.10, conclui-se que 89% dos utilizadores manifestaram sentimentos positivos ao classificar a sua experiência ao utilizar o sistema. Os sentimentos dominantes dos utilizadores referem-se a considerar o *IVO Outlook* uma ferramenta útil e profissional (83% dos utilizadores), simples e agradável (50% dos utilizadores), entusiástica e inovadora (33% dos utilizadores). Apenas dois

utilizadores exprimiram sentimentos negativos e as palavras usadas foram "sem valor" e "confuso".

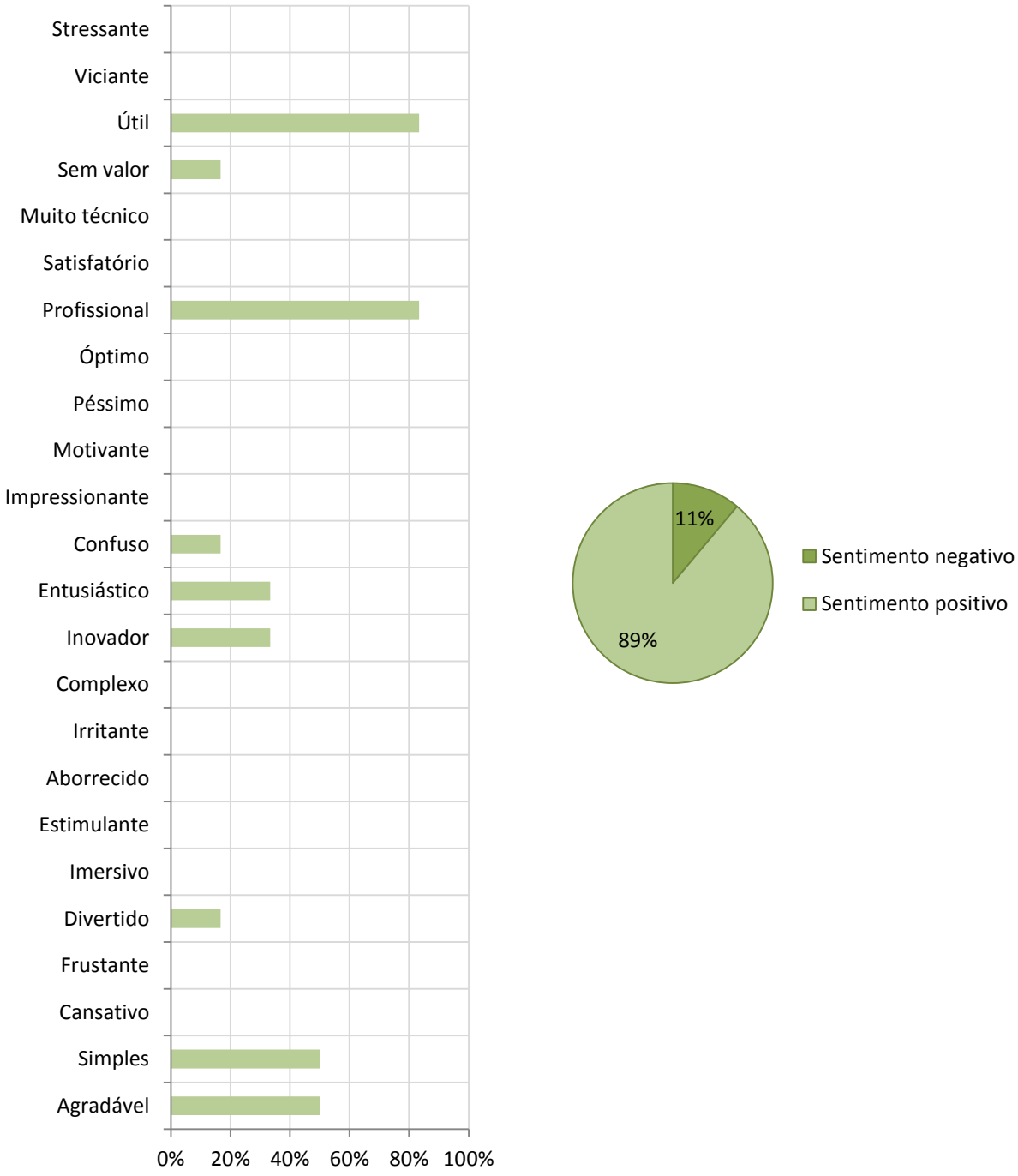


Figura 7.10. Envolvimento emocional dos utilizadores ao usarem o sistema (IVO Outlook)

7.2.3 Discussão

A avaliação das ferramentas de composição (*Composite Tools*) permitiu concluir que até utilizadores que nunca tiveram contacto com as ferramentas de composição conseguem facilmente construir aplicações sensíveis ao contexto utilizando o IVO. Esta avaliação mostrou ainda que o ambiente de programação visual facilita o rápido desenvolvimento deste tipo de aplicações.

Os dois grupos que realizaram os testes do *IVO Builder*, não mostraram uma diferença significativa, o que indica que neste caso, a área de conhecimento dos utilizadores não influenciou os resultados. No entanto, foram observados alguns comportamentos curiosos no grupo 2 (grupo de *design*), típica dos utilizadores de aplicações de *desktop*, como o uso de CTRL-Z para desfazer uma operação, BACK para apagar e *drag* & *drop* em algumas operações.

Os contextos de localização são sem dúvida os mais difíceis de criar e, simultaneamente, os mais utilizados em aplicações sensíveis ao contexto, tendo por isso merecido especial atenção. A sua criação é bastante facilitada com a importação quer da *Wikipedia*, quer da *Wikimapia,* pois muitos são os locais que já se encontram marcados nestes serviços, o que não dispensa contudo uma verificação dos mesmos. A dificuldade verificada na alteração do polígono que define um contexto de localização resultou da utilização das ferramentas sem que tenham sido fornecidas nenhumas ajudas ou sido feita alguma formação. Estas dificuldades foram posteriormente solucionadas através da disponibilização de pequenas janelas de ajuda com indicações precisas de como o utilizador deve proceder nas operações menos imediatas. Depois de informados, os utilizadores executaram estas operações com bastante facilidade.

A utilização do *IVO Calendar* é mais limitada, sendo de uso mais pessoal baseado na agenda do utilizador e permitindo automatizar pequenas tarefas baseadas em contextos temporais. A avaliação dos utilizadores a esta ferramenta foi bastante positiva com algumas questões a alcançarem a avaliação máxima possível.

O facto da avaliação das actividades de *workflow* revelar que os utilizadores tendem a referir as actividades utilizadas no teste como sendo as mais úteis, sugere a necessidade de efectuar mais testes, em que os utilizadores possam dispor de mais tempo e liberdade para experimentar o sistemas e assim poderem perceber melhor todas as suas potencialidades.

7.3 Mobile Client

Para testar o cliente (*Mobile Client*), foram criadas duas aplicações através do *IVO Builder*, as quais foram testadas por dois grupos de utilizadores. Pretendeu-se com estes testes, estudar a usabilidade das aplicações desenvolvidas através do *IVO Builder* e avaliar a capacidade da plataforma em construir aplicações utilizáveis e úteis na área do entretenimento e lazer.

As aplicações criadas permitiram testar uma grande parte das funcionalidades disponíveis. O primeiro grupo de utilizadores testou uma aplicação do tipo *peddy-paper* em que os utilizadores tinham que passar por quatro locais diferentes do *campus* da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa. O segundo grupo testou uma aplicação de guia turístico para a

zona de Belém em Lisboa. Estes últimos testes foram apresentados e publicados nas actas da conferência *Advances in Computer Entertainment Technology - ACE 2011* [107]. A aplicação criada foi também apresentada nesta conferência na categoria de *Creative Showcase and Interactive Art*, podendo os participantes na conferência instalá-la nos seus *smartphones* e experimentá-la [108].

7.3.1 Aplicação Peddy-paper

A primeira aplicação que serviu de teste ao cliente foi um jogo do tipo *peddy-paper*, em que os utilizadores tinham que passar por quatro locais distintos do *campus* da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa.

7.3.1.1 Participantes

A aplicação foi testada por um grupo de dez utilizadores voluntários três dos quais do sexo feminino, com as suas idades a variarem entre os 24 e 44 anos e uma idade média de 33.7.

7.3.1.2 Metodologia

As sessões de teste foram realizadas pelo investigador que representou o papel de facilitador e de observador. Antes de começar a usar a aplicação, os utilizadores foram informados sobre os objectivos do teste e foi-lhes fornecido um *smartphone* Android (HTC Desire) com a aplicação já instalada e equipado com auscultadores. Após o *briefing*, o facilitador levou cada participante para o ponto de partida, e a partir daí estes tinham que seguir as instruções fornecidas pela aplicação. O facilitador acompanhou os utilizadores ao longo de todo o percurso, observando as suas acções e tomando notas.

7.3.1.3 Questionário

Após a realização dos testes, os utilizadores responderam a um questionário que capturou dados pessoais, *feedback* experimental, bem como sugestões e comentários dos utilizadores. As respostas foram posteriormente analisadas e os resultados mostrados e discutidos de seguida.

1. **Facilidade de aprendizagem.** A avaliação da facilidade de aprendizagem foi feita através das três questões apresentadas na Tabela 7.13, tendo sido usada uma escala de *Likert* de 5 pontos, com uma resposta de 1 (um) significando "discordo totalmente" e uma resposta de 5 (cinco) significando "concordo totalmente".

Facilidade de aprendizagem	
Q1	É fácil aprender a usar a aplicação
Q2	Fiquei rapidamente habilitado a usar a aplicação
Q3	Lembro-me facilmente de como usar a aplicação

Tabela 7.13. Facilidade de aprendizagem (Mobile Client – Peddy-paper)

2. **Facilidade de utilização.** A facilidade de utilização foi avaliada através das questões constantes na Tabela 7.14, tendo os utilizadores usado o mesmo procedimento descrito no parágrafo anterior.

Facilidade de utilização	
Q4	A aplicação é fácil de utilizar
Q5	A aplicação requer poucos passos para fazer o que se pretende
Q6	Consigo imaginar a utilização do sistema na construção de outras aplicações, cenários ou situações
Q7	Consigo usar a aplicação sem necessitar de instruções escritas
Q8	Não detectei inconsistências de utilização da interface
Q9	Foi fácil recuperar de uma situação inesperada
Q10	A aplicação respondeu rapidamente às diversas acções do utilizador

Tabela 7.14. Facilidade de utilização (Mobile Client – Peddy-paper)

3. **Facilidade de execução das tarefas propostas.** A terceira secção do questionário avaliou as a facilidade de execução das tarefas que o utilizador teve de realizar, a qual incluiu as questões constantes na Tabela 7.15. Para responder a estas questões, foi usada uma escala de 1 (um), significando "muito difícil", a 5 (cinco) significando "muito fácil".

Facilidade de execução das tarefas propostas	
Q11.a	Navegação e orientação
Q11.b	Procurar um ponto de interesse
Q11.c	Obter informação sobre um ponto de interesse
Q11.d	Explorar a realidade aumentada
Q11.e	Ler código IVO (QR-Code)
Q11.f	Executar *quiz*
Q11.g	Preencher formulário

Tabela 7.15. Facilidade de execução das tarefas propostas (Mobile Client – Peddy-paper)

4. **Avaliação das actividades de workflow.** A avaliação das actividades de *workflow* foi feita através da indicação das três actividades de *workflow* que os utilizadores consideravam mais úteis e as três que consideravam menos úteis ter disponíveis.

5. **Envolvimento emocional.** O envolvimento emocional dos utilizadores ao utilizarem a aplicação foi avaliado usando o mesmo método usado na avaliação das ferramentas de composição (*IVO Builder* e *IVO Outlook*).

7.3.1.4 Cenário de teste

Quando o utilizador se aproximava do ponto de partida, começava a tocar uma música de fundo (actividade "Tocar áudio"), sendo fornecidas instruções de voz (actividade "Falar texto") aconselhando o utilizador a explorar a área envolvente através da realidade aumentada e alertando-o que de seguida se devia dirigir para ocidente. Ao sair da área correspondente ao ponto de partida, eram fornecidas novas instruções faladas (actividade "Falar texto") para notificar o utilizador que deveria responder ao *quiz* que lhe era apresentado (actividade "Efectuar *quiz*"), o qual se referia a informação mostrada na actividade de realidade aumentada. O *quiz* foi configurado para terminar apenas quando o utilizador respondesse correctamente a todas as questões podendo tentar quantas vezes fosse necessário. Quando o utilizador respondesse correctamente a todas as perguntas, era lançado o navegador (actividade "Ir para local") de forma a guiar o utilizador para o segundo ponto do jogo.

Após chegar ao segundo ponto eram dadas novas instruções de voz (actividade "Falar texto") que indicam uma charada: *"... repare que o próximo destino está no código*" significando que o utilizador deve procurar um código de barras QR-Code. Para efeitos de teste, o facilitador que acompanhou o utilizador forneceu este código de barras. O utilizador teve então de descobrir como ler o código através do IVO. Ao ler este código o sistema fornecia novas instruções faladas (actividade "Falar texto") para guiar o utilizador para o terceiro ponto: a Biblioteca.

Ao aproximar-se da Biblioteca, eram apresentadas oralmente novas instruções (actividade "Falar texto"), dizendo para o utilizador visitar a Biblioteca e "*... aproveitar a oportunidade para conhecer mais sobre o edifício, pois vai precisar dessa informação para descobrir qual o último ponto deste peddy-paper*". Ao sair da Biblioteca era mostrado um novo *quiz* com duas perguntas sobre o edifício que o utilizador acabara de visitar (actividade "Efectuar *quiz*"). Mais uma vez, o sistema só avançava após a resposta correcta do utilizador às perguntas, sendo só então indicado o ponto final do *peddy-paper* através de mais instruções faladas (actividade "Falar texto").

Ao chegar ao último ponto, o sistema agradecia ao utilizador a colaboração e solicitava-lhe que preenchesse um formulário para avaliar a sua experiência ao realizar o *peddy-paper* (actividade "Preencher formulário").

Sempre que o utilizador se aproximava de qualquer um dos quatro pontos, o estado do *Facebook* do utilizador era actualizado, informando os amigos de que ele/ela estava a nesse local (actividade "Actualizar estado no *Facebook*").

Foi ainda utilizado um contexto de proximidade de pessoas que falava o texto "*Olá [nome do contacto]*" sempre que o contacto estava na proximidade do utilizador. Para efeitos de teste o facilitador ligava o *Bluetooth* do seu telemóvel para a realização do teste.

7.3.1.5 Resultados

Após a realização do teste, os utilizadores responderam ao questionário apresentado anteriormente, tendo as respostas sido analisadas e os resultados mostrados e discutidos de seguida.

Facilidade de aprendizagem. A maior parte dos utilizadores afirmou ser fácil lembrar-se de como usar a aplicação (Q3) (4.5 ± 0.70711) e que rapidamente se tornou hábil a usá-la (Q2) (4.1 ± 0.073786). Embora bastante positiva, a opinião dos utilizadores acerca da questão "É fácil aprender a usar a aplicação" (Q1) (3.8 ± 0.91894) não foi tão forte.

		Média	Variância	Desvio Padrão
Q1	É fácil aprender a usar a aplicação	3.8	0.84444	0.91894
Q2	Fiquei rapidamente habilitado a usar a aplicação	4.1	0.54444	0.73786
Q3	Lembro-me facilmente de como usar a aplicação	4.5	0.50000	0.70711

Tabela 7.16. Facilidade de aprendizagem (Mobile Client – Peddy-paper)

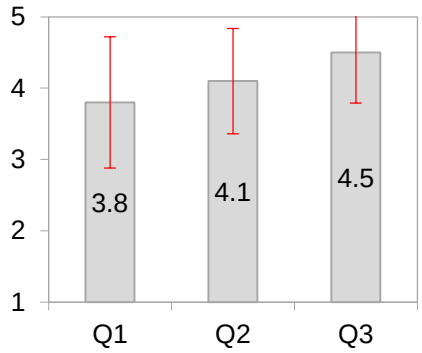


Figura 7.11. Facilidade de aprendizagem (Mobile Client – Peddy-paper)

Facilidade de utilização. Globalmente, os resultados relativos à facilidade de utilização (Q4 a Q10) foram muito positivos. A maior parte dos utilizadores afirmaram conseguir imaginar a utilização do IVO na construção de outras aplicações, cenários ou situações (Q6) (4.6 ± 0.69921), que a aplicação requer poucos passos para fazer o que se pretende (Q5) (4.5 ± 0.52705), e que respondeu rapidamente às acções do utilizador (Q10) (4.4 ± 0.96609). Embora positiva, a opinião dos utilizadores acerca da facilidade de recuperar de uma situação inesperada (Q9) (3.4 ± 0.69921) revela uma pontuação menor. Acredita-se que isto pode estar relacionado com o facto de alguns utilizadores não estarem familiarizados com a interface do *Android*, o que tornou difícil a adaptação imediata ao sistema. Com efeito, a maior parte dos utilizadores não eram utilizadores

Android e não perceberam imediatamente que precisavam usar o botão de menu dos telefones *Android* para ter acesso a algumas das funcionalidades da aplicação. O menor consenso, traduzido num desvio-padrão maior ou igual a 1, verificou-se quando os utilizadores foram questionados acerca da necessidade de instruções escritas para usar a aplicação (Q7) (3.6 ± 1.42984), as inconsistências encontradas quando se utiliza a interface (Q8) (3.9 ± 0.99443) e a resposta rápida do sistema às acções do utilizador (Q10) (4.4 ± 0.96609). Mais uma vez, acredita-se que este menor consenso nestas questões pode estar relacionado com o facto de que alguns utilizadores não estarem familiarizados com a interface do *Android*.

		Média	Variância	Desvio Padrão
Q4	A aplicação é fácil de utilizar	3.9	0.54444	0.73786
Q5	A aplicação requer poucos passos para fazer o que se pretende	4.5	0.27778	0.52705
Q6	Consigo imaginar a utilização do sistema na construção de outras aplicações, cenários ou situações	4.6	0.48889	0.69921
Q7	Consigo usar a aplicação sem necessitar de instruções escritas	3.6	2.04444	1.42984
Q8	Não detectei inconsistências de utilização da interface	3.9	0.98889	0.99443
Q9	Foi fácil recuperar de uma situação inesperada	3.4	0.48889	0.69921
Q10	A aplicação respondeu rapidamente às diversas acções do utilizador	4.4	0.93333	0.96609

Tabela 7.17. Facilidade de utilização (Mobile Client – Peddy-paper)

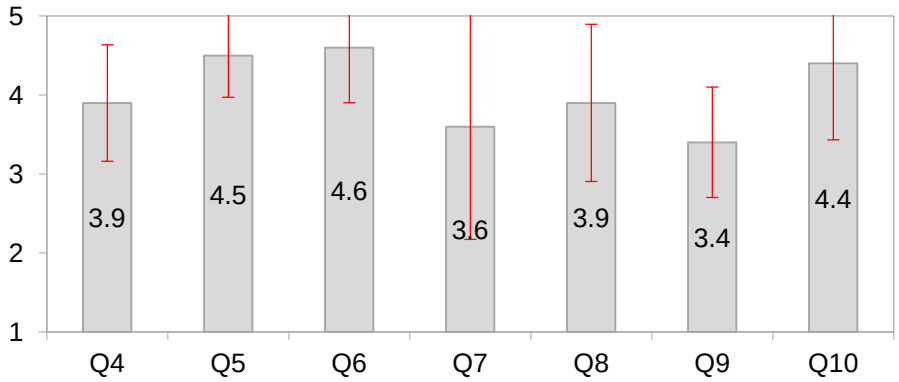


Figura 7.12. Facilidade de utilização (Mobile Client – Peddy-paper)

Facilidade de execução das tarefas propostas. A questão 11 avaliou a facilidade encontrada pelos utilizadores na execução das tarefas mais comuns na aplicação e ilustradas na Tabela 7.18. De acordo com os resultados apresentados nesta mesma tabela, as interacções foram geralmente consideradas fáceis de executar. A utilização da realidade aumentada (Q11d), embora positiva, foi a questão com pontuação mais baixa. Conforme descrito anteriormente, a maioria dos utilizadores não percebeu que tinha que usar o botão de menu, de forma que demoraram algum tempo tentando descobrir como aceder a esta funcionalidade.

		Média	Variância	Desvio Padrão
Q11.a	Navegação e orientação	4.8	0.40000	0.63246
Q11.b	Procurar um ponto de interesse	3.9	1.14286	1.06904
Q11.c	Obter informação sobre um ponto de interesse	4.1	0.41071	0.64087
Q11.d	Realidade aumentada	3.7	0.90000	0.94868
Q11.e	Ler código IVO (QR-Code)	4.5	0.50000	0.70711
Q11.f	Executar *quiz*	4.5	0.50000	0.70711
Q11.g	Preencher formulário	4.6	0.52778	0.72648

Tabela 7.18. Facilidade de execução das tarefas propostas (Mobile Client – Peddy-paper)

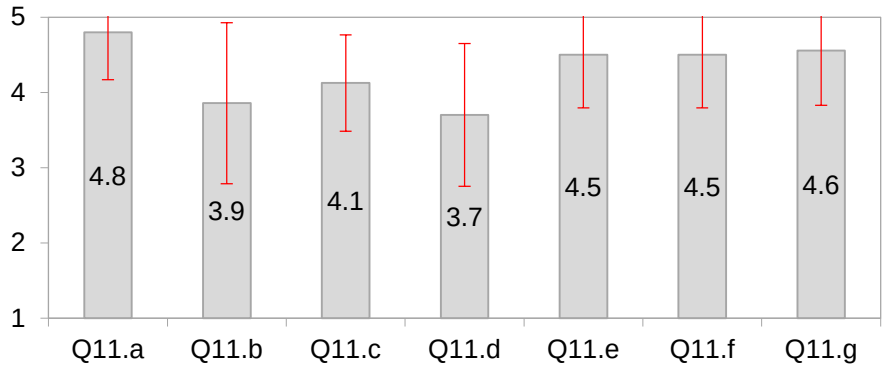


Figura 7.13. Facilidade de execução das tarefas propostas (Mobile Client – Peddy-paper)

Avaliação das actividades de workflow. De acordo com as respostas dos utilizadores (Figura 7.14), as actividades consideradas mais úteis foram "Ir para local" (5 utilizadores), "Ler código IVO" (3 utilizadores) e "Mudar estado do telefone", "Enviar SMS", "Actualizar estado no *Facebook*" e "Tirar foto" (2 utilizadores). As actividades consideradas menos úteis foram "Enviar *email*" (3 utilizadores), "Efectuar chamada", "Correr *script*" e "Condição" (2 utilizadores).

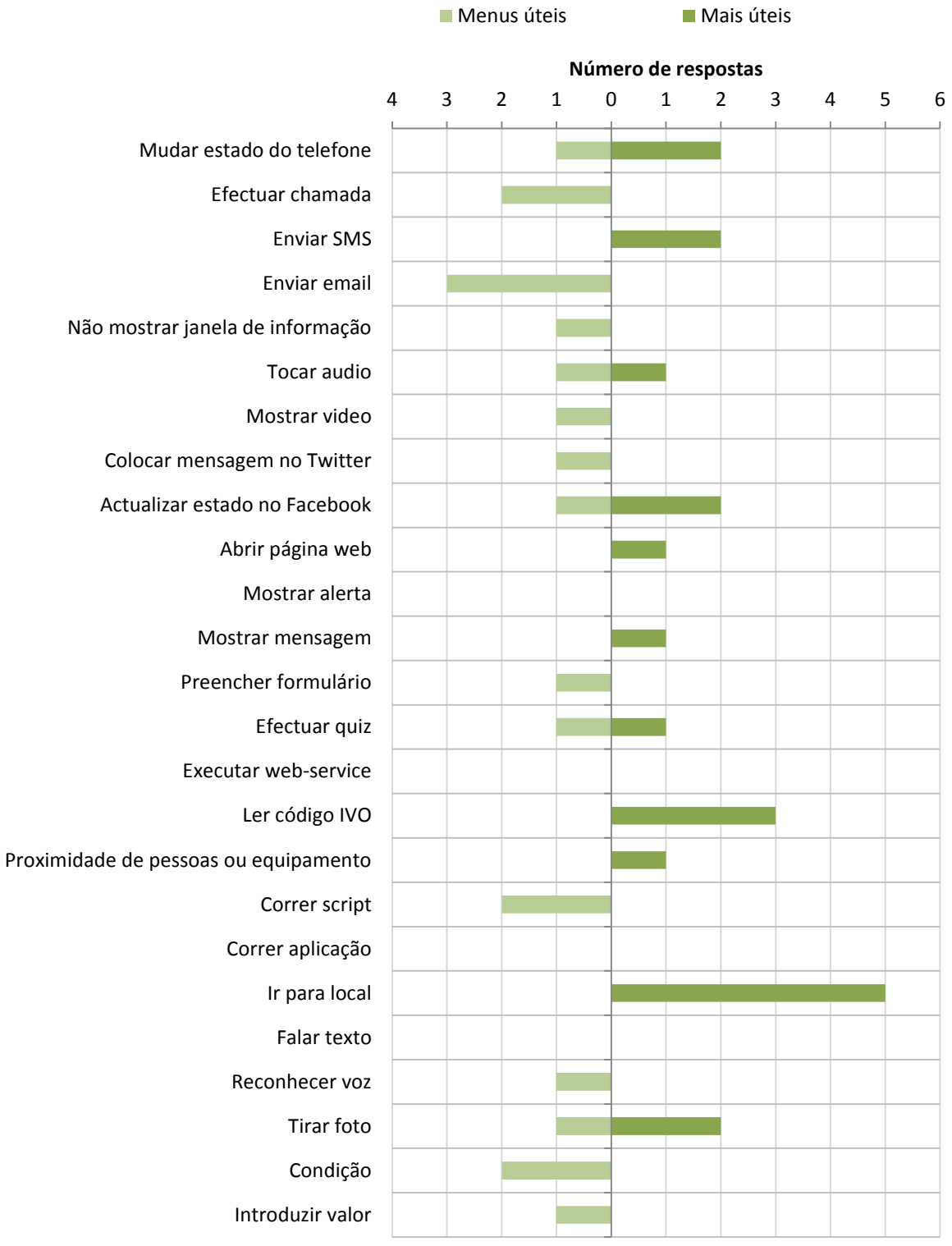


Figura 7.14. Avaliação das actividades de workflow (Mobile Client – Peddy-paper)

Envolvimento emocional. A partir da análise da quarta parte do questionário (Figura 7.15), concluiu-se que os utilizadores exprimiram sentimentos positivos ao classificar a sua experiência com a aplicação (98% da totalidade das palavras seleccionadas). A palavra mais seleccionada foi "Agradável" (70%) seguido da palavra "Simples", "Divertido" e "Útil" (60%). Metade dos

participantes considerou a aplicação "Imersiva" e 40% "Inovadora". Apenas um utilizador exprimiu sentimentos negativos e a palavra usada foi "Confuso".

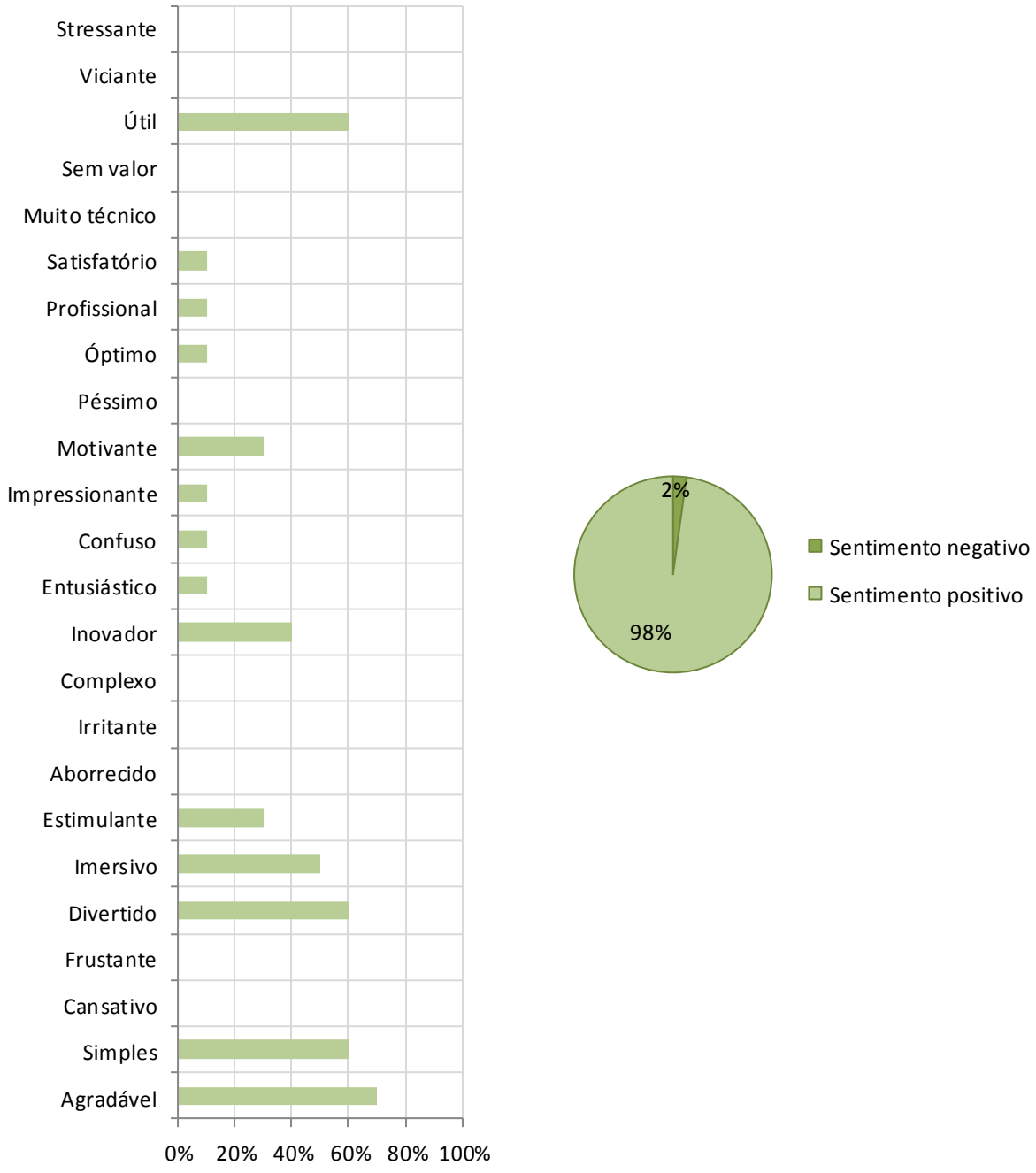


Figura 7.15. Envolvimento emocional (Mobile Client – Peddy-paper)

7.3.2 Aplicação de Guia Turístico

A segunda aplicação que serviu de teste ao *IVO Client*, foi um guia turístico (criado com o *IVO Builder*) para a área de Belém em Lisboa, tendo os testes sido realizados em Junho de 2011 de acordo com os detalhes de seguida referidos.

7.3.2.1 Participantes

A aplicação foi testada por um grupo de sete utilizadores voluntários três dos quais do sexo feminino. Os utilizadores possuíam formação base distintas e as suas idades variaram entre os 24 e 37 anos com uma idade média de 29.6.

7.3.2.2 Metodologia

As sessões de teste foram realizadas pelo investigador, que representou o papel de facilitador e de observador. Antes de começar a usar a aplicação, os utilizadores foram informados sobre os objectivos do teste, e foi-lhes fornecido um dispositivo Android com a aplicação já instalada. Foram usados *smartphones* Android HTC Desire e Samgung Galaxy ACE e ainda um *tablet* Samsung Galaxy Tab com ecrã de 7.0 polegadas. Todos os dispositivos móveis se encontravam equipados com auscultadores.

Após o *briefing* o facilitador levou cada participante para o ponto de partida (Museu Nacional do Coches), e a partir daí, estes tinham que seguir as instruções fornecidas pela aplicação. O facilitador acompanhou os utilizadores ao longo de todo o percurso, observando as suas acções e tomando notas.

7.3.2.3 Questionário

Após a realização dos testes, os utilizadores responderam a um questionário semelhante ao utilizado nos testes da aplicação anterior (*peddy-paper*), mas o qual incluía uma secção para os utilizadores exprimirem a sua opinião relativa à utilidade da aplicação que testaram. As questões que constituem cada uma das secções do questionário encontram-se na Tabela 7.19.

Utilidade	
Q1	A aplicação é útil
Q2	A aplicação dá-me maior controlo sobre as actividades que tenho que realizar
Q3	A aplicação facilita a realização da visita ao local
Q4	A aplicação poupa tempo na realização da visita
Q5	A informação fornecida pela aplicação é útil
Facilidade de Aprendizagem	
Q6	É fácil aprender a usar a aplicação
Q7	Fiquei rapidamente habilitado a usar a aplicação
Q8	Lembro-me facilmente de como usar a aplicação
Facilidade de utilização	
Q9	A aplicação é fácil de utilizar
Q10	Consigo usar a aplicação sem necessitar de instruções escritas

Q11	Não detectei inconsistências de utilização da interface
Q12	Foi fácil recuperar de uma situação inesperada
Q13	A aplicação respondeu rapidamente às diversas acções do utilizador
Q14	A aplicação tem uma interface amigável
Q15 Facilidade de execução das tarefas propostas	
Q15.a	Navegação e orientação
Q15.b	Procurar um ponto de interesse
Q15.c	Obter informação sobre um ponto de interesse
Q15.d	Realidade aumentada
Q15.e	Executar *quiz*
Q15.f	Controlo de áudio
Q15.g	Controlo de vídeo

Tabela 7.19. Questões relativas ao feedback experimental (Mobile Client – Guia Turístico)

Tal como nos testes anteriores, o questionário incluía ainda uma secção que permitia fazer a avaliação das actividades de *workflow*, e outra para a avaliação do envolvimento emocional do utilizar ao utilizar a aplicação.

Todas as secções foram avaliadas usando o mesmo método que nos testes anteriores, tendo as respostas dos utilizadores às questões apresentadas no questionário sido analisadas e os resultados mostrados e discutidos no ponto 7.3.2.5.

7.3.2.4 Cenário de Teste

A aplicação consiste em 16 pontos de interesse da área de Belém em Lisboa, sendo o utilizador guiado para cinco desses locais os quais constituem o roteiro turístico da aplicação:

1. Museu Nacional dos Coches
2. Pastéis de Belém
3. Mosteiro dos Jerónimos
4. Centro Cultural de Belém
5. Padrão dos Descobrimentos

Todos os 16 pontos possuem informação contextualizada que é mostrada e falada (actividade "Falar texto") quando o utilizador entra no espaço em redor de cada um desses pontos.

Os cinco pontos de interesse do roteiro possuem informação mais rica e detalhada com áudio (actividade "Tocar áudio"), vídeos (actividade "Mostrar vídeo") e *quizzes* (actividade "Efectuar *quiz*"), de forma a melhorar a experiência do utilizador.

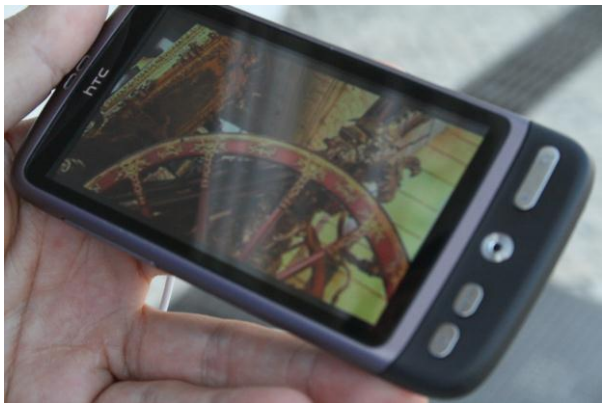
Depois de visitar cada um dos cinco pontos seleccionados, o próximo ponto é indicado por meio de texto falado (actividade "Falar texto") e o navegador GPS é lançado para guiar o utilizador para esse ponto (actividade "Ir para local"). O utilizador pode decidir seguir as indicações de direcção para o próximo ponto, explorar livremente o local ou descansar num dos belos jardins ou cafés da área antes de continuar o passeio.

Sempre que o utilizador se aproxima de qualquer dos pontos de interesse, o estado do *Facebook* do utilizador é actualizado informando os amigos de que ele/ela se encontra a visitar esse local (actividade "Actualizar estado no *Facebook*").

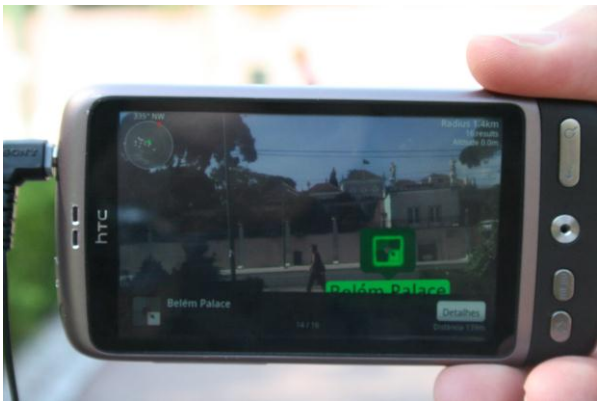
Além disso, em alguns pontos de interesse, o perfil do telefone é colocado automaticamente em modo silencioso sempre que o utilizador entra no espaço envolvente e colocado no modo normal quando se afasta desse espaço.

O utilizador pode optar em qualquer altura por não querer que estas acções automáticas sejam executadas, através das Preferências do Utilizador tal como descrito na Secção 6.5.4.

As fotografias da Figura 7.16 foram tiradas durante a realização dos testes e representam vários cenários de utilização da aplicação.



a) Utilizador a ver um vídeo do Museu dos Coches



b) Explorar a realidade aumentada em frente ao Palácio de Belém

c) Utilizador a receber informação contextualizada relativa à Praça Mouzinho de Albuquerque

d) Utilizadores junto ao Padrão dos Descobrimentos

Figura 7.16. Utilizadores a testar a aplicação (Mobile Client – Guia Turístico)

7.3.2.5 Resultados

Após a realização do teste, os participantes responderam ao questionário anteriormente referido, tendo as respostas apresentadas no questionário sido analisadas e os resultados mostrados e discutidos de seguida (Tabela 7.19).

		Média	Variância	Desvio Padrão
Utilidade				
Q1	A aplicação é útil	4.1	0.14286	0.71270
Q2	A aplicação dá-me maior controlo sobre as actividades que tenho que realizar	4.4	0.66667	0.81650
Q3	A aplicação facilita a realização da visita ao local	4.4	0.23810	0.48795
Q4	A aplicação poupa tempo na realização da visita	4.6	0.66667	0.81650
Q5	A informação fornecida pela aplicação é útil	4.3	0.14286	0.37796
Facilidade de Aprendizagem				
Q6	É fácil aprender a usar a aplicação	4.3	0.57143	0.75593
Q7	Fiquei rapidamente habilitado a usar a aplicação	4.3	0.23810	0.48795
Q8	Lembro-me facilmente de como usar a aplicação	4.4	0.28571	0.53452
Facilidade de Utilização				
Q9	A aplicação é fácil de utilizar	4.4	0.61905	0.78680
Q10	Consigo usar a aplicação sem necessitar de instruções escritas	4.1	0.80952	0.89974
Q11	Não detectei inconsistências de utilização da interface	3.7	1.23810	1.11270
Q12	Foi fácil recuperar de uma situação inesperada	4.1	0.47619	0.69007
Q13	A aplicação respondeu rapidamente às diversas acções do utilizador	4.0	0.66667	0.81650
Q14	A aplicação tem uma interface amigável	4.1	0.47619	0.69007
Facilidade de Execução das Tarefas Propostas				
Q15.a	Navegação e orientação	4.6	0.28571	0.53452
Q15.b	Procurar um ponto de interesse	4.3	0.26667	0.51640

Q15.c	Obter informação sobre um ponto de interesse	4.3	0.26667	0.51640
Q15.d	Realidade aumentada	4.0	1.00000	1.00000
Q15.e	Executar *quiz*	4.9	0.14286	0.37796
Q15.f	Controlo de áudio	5.0	0.00000	0.00000
Q15.g	Controlo de vídeo	4.6	0.61905	0.78680
Q15.h	Navegação e orientação	4.6	0.28571	0.53452

Tabela 7.20. Respostas ao questionário (Mobile Client – Guia Turístico)

Utilidade. A maioria dos participantes classificou as questões relativas à utilidade da aplicação com pontuações altas, tendo-se verificado um grande consenso nas respostas, o que revela uma opinião bastante positiva em relação à utilidade da aplicação (Figura 7.17).

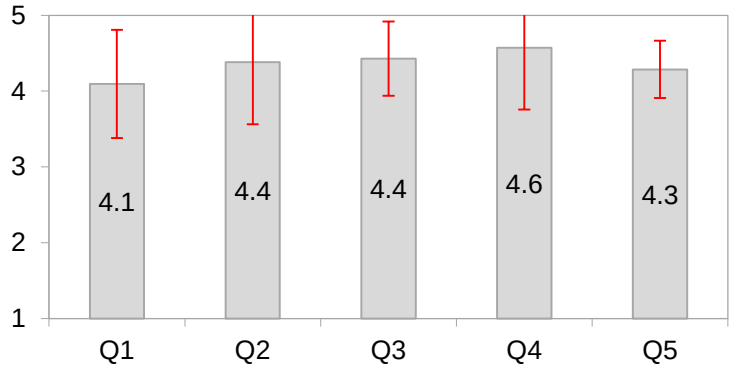


Figura 7.17. Utilidade (Mobile Client – Guia Turístico)

Facilidade de aprendizagem. Tal como na secção anterior, também no que diz respeito às questões relativas à facilidade de aprendizagem, as respostas revelaram pontuações altas e grande consenso na opinião dos participantes no teste (Figura 7.18).

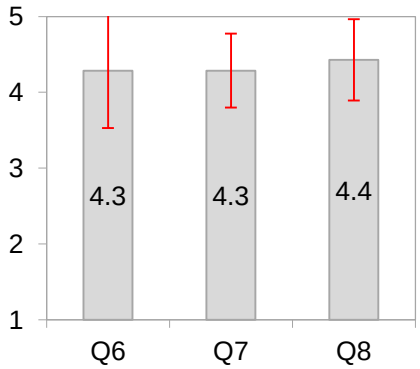


Figura 7.18. Facilidade de aprendizagem (Mobile Client – Guia Turístico)

Facilidade de utilização. No geral, os resultados relativos à facilidade de utilização (questões Q9 a Q15) foram muito positivos. A maioria dos participantes afirmou que a aplicação é fácil de utilizar (Q9); que conseguem usá-la sem instruções escritas (Q10); que foi fácil recuperar de uma situação inesperada (Q12); e, que a aplicação possui uma interface amigável (Q14). Embora

positiva, a opinião dos participantes acerca das inconsistências encontradas na interface (Q11) revela a pontuação mais baixa e também o menor consenso. Acreditamos que isto pode estar relacionado com o facto de alguns utilizadores não estarem familiarizados com a interface do Android, tornando difícil a adaptação imediata ao sistema. Durante as observações, verificou-se que inicialmente os utilizadores não perceberam que deviam usar o botão de menu do Android para aceder a algumas funcionalidades do IVO (tal como ilustrado na Figura 6.31), mas progressivamente essa dificuldade foi sendo ultrapassada pelos utilizadores.

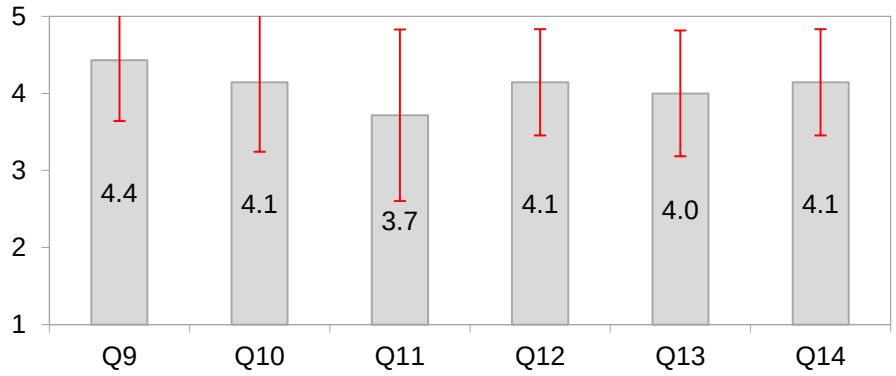


Figura 7.19. Facilidade de utilização (Mobile Client – Guia Turístico)

Facilidade de execução das tarefas propostas. A questão Q15 avaliou a facilidade encontrada pelos participantes na execução das tarefas descritas na Tabela 7.19. De acordo com os resultados apresentados nesta mesma tabela, as interacções foram geralmente consideradas fáceis de executar. O acesso à funcionalidade de realidade aumentada (Q15.d), apesar de positivo, foi a questão com menor pontuação. Conforme descrito anteriormente, a maioria dos participantes não percebeu que tinha de usar o botão de menu do dispositivo Android, e por esse facto demoraram algum tempo e despenderam esforço tentando descobrir como aceder a esta funcionalidade.

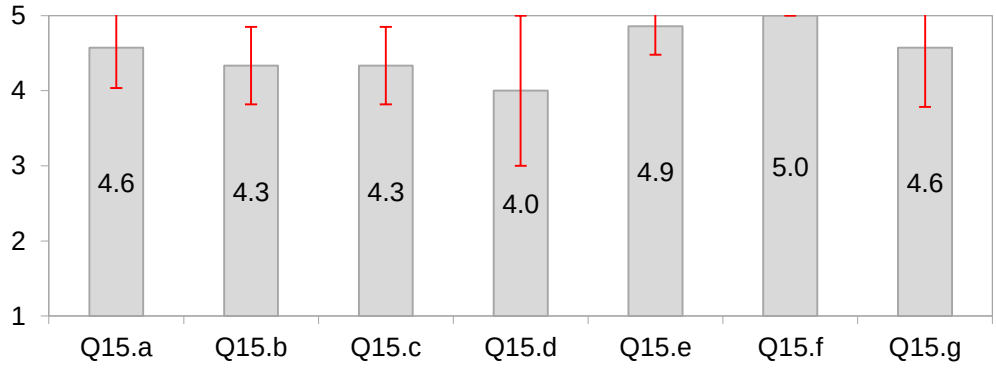


Figura 7.20. Facilidade de execução das tarefas propostas (Mobile Client – Guia Turístico)

Avaliação das actividades de workflow. As actividades consideradas mais úteis foram "Reproduzir áudio", "Colocar mensagem no *Twitter*", "Actualizar estado no *Facebook*", "Efectuar

quiz", "Ir para local" e "Tirar foto" (todas com 2 utilizadores). Acreditamos que os utilizadores tendem a referir as actividades utilizadas durante o teste como as mais úteis.

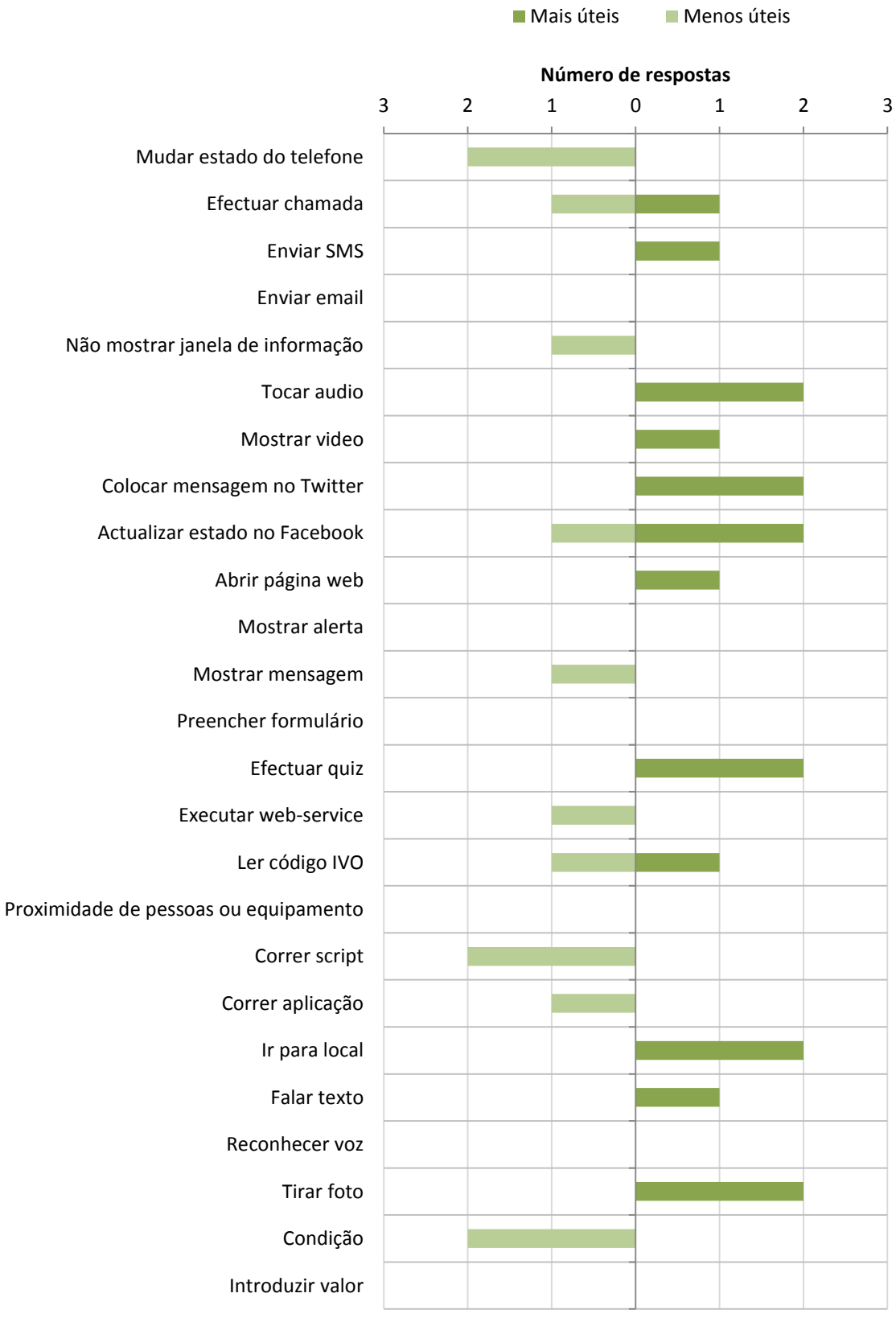


Figura 7.21. Avaliação das actividades de workflow (Mobile Client – Guia Turístico)

Envolvimento emocional. A partir da análise da quinta parte do questionário (Figura 7.22), concluiu-se que os participantes exprimiram sentimentos positivos (98% do total de palavras seleccionadas) para classificar a sua experiência com a aplicação. A palavra mais seleccionada foi "Agradável" (60%), seguida da palavra "Estimulante" e "Útil" (50%). 40% dos participantes consideraram a aplicação "Inovadora" e "Divertida". Apenas um utilizador exprimiu sentimentos negativos e, a palavra usada foi "Confuso".

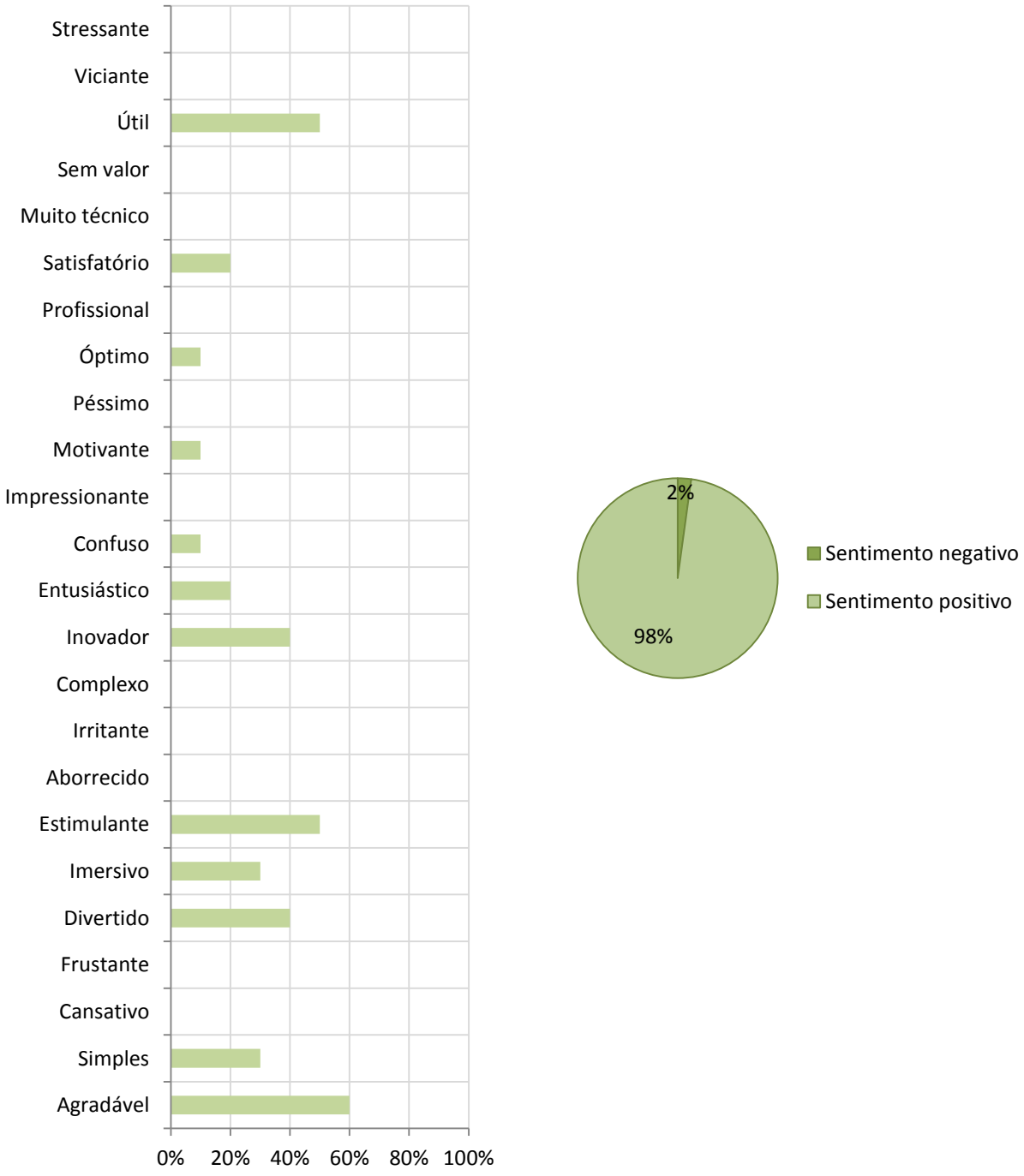


Figura 7.22. Envolvimento emocional (Mobile Client – Guia Turístico)

7.3.3 Discussão

As avaliações efectuadas permitiram concluir que as aplicações criadas possibilitam uma experiência agradável e rica tendo os utilizadores expressado sentimentos muito encorajadores.

A questão aberta dos questionários permitiu obter comentários e sugestões úteis que foram tidos em conta nalguns melhoramentos do *IVO Client*, alguns dos quais já implementados, e outros que estão previstos como trabalho futuro. Por exemplo, uma das sugestões já implementada foi a de que deveria ser possível retomar uma visita que ficou a meio, informando o utilizador dos locais já visitados e sugerir o próximo ponto a visitar. Também foi sugerido que deveria existir um filtro na funcionalidade "O que se diz", a fim de se obter apenas as mensagens mais relevantes do *Twitter*. Foi ainda sugerido, que seria interessante ter *quizzes* e outros jogos num ambiente de cooperação, a fim de proporcionar uma forma mais interessante e divertida de enriquecer a interacção dos utilizadores e promover a sua colaboração.

Os testes realizados revelaram uma vida útil da bateria dos dispositivos móveis entre duas a três horas com uma utilização normal das aplicações (áudio, vídeo, GPS e ligações de dados de rede). Isto representa um constrangimento tendo sido efectuadas alterações ao código no sentido de melhorar a vida útil da bateria. Essas alterações centraram-se na utilização do GPS apenas quando o utilizador estiver em movimento. Para isso foi usado o acelerómetro, sendo o GPS ligado apenas quando for detectado movimento. Se for detectado que o utilizador está parado durante um determinado período de tempo (foi utilizado um minuto) o GPS é desligado. Os testes efectuados com esta abordagem permitiram concluir que a vida útil da bateria aumenta duas a três vezes dependendo do tipo de utilização que se esteja a fazer com mais ou menos paragens do utilizador.

7.4 Avaliações Informais

Para além das avaliações formais apresentadas nas secções anteriores, foram efectuadas outras com a participação de utilizadores finais e que consistiram na construção e utilização de aplicações que permitiram avaliar informalmente a plataforma:

- Uma aplicação na área vitívinicola para automatizar processos usados por técnicos agrícolas na visita a fornecedores de uva;
- Um guia turístico para o distrito de Portalegre;
- Um guia turístico para Barcelona (obras de Gaudí em Barcelona).

7.4.1 Aplicação na área vitivinícola

A aplicação para a área vitivinícola está a ser utilizada desde 2009 no Esporão. Nesta altura, a plataforma IVO ainda não apresentava a totalidade das funcionalidades referida neste trabalho, tendo um suporte mais limitado em termos de *workflow* e do *runtime* do cliente, o qual foi originalmente desenvolvido para *Windows Mobile* e utilizado em *smartphones* HTC Touch Cruise. A empresa vai começar a utilizar o cliente Android apresentado neste trabalho, mas o *IVO Builder* e o *Application Builder Server* continuam a ser os mesmos, com uma instalação nos servidores da própria empresa.

Esta aplicação permite automatizar o processo usado pelos técnicos agrícolas da empresa nas visitas a fornecedores de uva, de forma semelhante ao ilustrado no cenário de uso 4. As áreas referentes às vinhas que se encontravam catalogadas no Sistema de Informação Geográfica (SIG) da empresa, foram importadas para o IVO através de um ficheiro KML gerado no SIG. As restantes foram criadas pelos técnicos agrícolas, através da funcionalidade de criação de polígonos disponibilizada pelo *IVO Builder*, fazendo eles próprios a manutenção destas áreas ao longo do tempo. A informação contextual associada às vinhas inclui a área da mesma em hectares, as castas plantadas e informação dinâmica obtida dos servidores da empresa com o resultado das últimas visitas realizadas à vinha em questão.

Esta aplicação faz um uso do IVO em ambiente empresarial, com recurso a um conjunto de formulários (Figura 7.23) que foram desenvolvidos através do *IVO Builder*. Desde a altura em que entrou em exploração (2009), foram realizadas 580 visitas a fornecedores, a maior parte das quais antes da altura da vindima. Numa visita a um fornecedor, o técnico verifica o estado de cada vinha preenchendo o formulário de "Ficha de Visita" (Figura 7.23a). De acordo com a análise efectuada, assim pode ter necessidade de preencher outros formulários ("Ficha Fitossanitária" – Figura 7.23b, "Tratamento Fitossanitária" – Figura 7.23c ou "Registo de Fertilizante" – Figura 7.23d). No final, ao terminar a visita ao fornecedor, o técnico preenche o "Relatório de Visita" com observações e recomendações ao fornecedor (Figura 7.23e).

a) Ficha de Visita **b) Ficha Fitossanitária** **c) Tratamento Fitossanitário**

d) Registo de Fertilizante **e) Relatório de Visita**

Figura 7.23. Formulários utilizados na aplicação vitivinícola

Estes formulários foram desenvolvidos por um utilizador experiente no IVO, em colaboração com o técnico de informática da empresa, sendo a sua manutenção efectuada internamente pela empresa. A informação recolhida através dos formulários, é guardada no *Application Builder Server*, tendo sido desenvolvidas ferramentas que permitem efectuar uma melhor gestão do negócio e gerar um conjunto de relatórios customizados de acordo com as necessidades da empresa. A Figura 7.24 e a Figura 7.25, ilustram dois relatórios que é possível obter com estas ferramentas.

Esporão S.A.

Ficha de Visita

Nº Visita: 23 **Data:** 23-06-2009 **Hora:** 12:22

Fornecedor: Casa Agrícola Rosado Fernandes, Lda.

Modo de Produção: Convencional

Dados Técnicos

Estado Fenológico: Cacho Fechado **Data:** 23-06-2009 **Vigor Vegetativo:** Médio

Operações de Instalação e Manutenção

Operações Culturais:	**Marcação**	☐	**Plantação**	☐	**Enxertia** ☐
	Retancha	☐	**Aramação**	☐	**Empa** ☐
	Desponta	☐	**Monda de Cachos**	☐	**Poda Inverno** ☐
	Poda em Verde	☐	**Nenhuma Operação em Curso**	☒	

Tipo Manutenção Solo: Enrelvamento

Vinha Irrigada: Sim **Dotações de Rega:** **horas/semana**

Estado Geral da Vinha:

Ficha Fitossanitária: Nº 128; 129; 130; 131; 132

Técnico: José Madeira

Figura 7.24. Ficha de Visita

Esporão S.A.

Ficha Fitossanitária

130

Nº Visita: 23 **Data:** 23-06-2009 **Hora:** 12:57

Fornecedor: Casa Agrícola Rosado Fernandes, Lda.

Talhão: T2

Pragas

Cicadela:	**Nº. Indíviduos/100 Folhas:**	**Sintomas:**	**Fase do Ciclo:**
Ácaros:	**Nº. Indíviduos/100 Folhas:**	**Sintomas:**	**Espécie:**
Traça da Uva:	**Nº. Indíviduos/100 Folhas:**	**Sintomas:**	
Cochonilha Algodão:	**Nº. Indíviduos/100 Folhas:**	**Sintomas:**	

Doenças

Oídio:	**Nível de Infestação:**	**Sintomas:**	
Míldio:	**Nível de Infestação:**	**Sintomas:**	
Doença do Lenho:	**Nível de Infestação:**	**Sintomas:**	**Tipo:**
Podridão Cinzenta:	**Nível de Infestação:**	**Sintomas:**	

Virose: **Tipo:** **Nível de Infestação:**

Carência:

Presença de Auxiliares: **Coleopteros** [X] **Sirfídeos** [ ] **Crisopídeos** [ ] **Fitoseídeos** [ ]

Tratamentos Fitossanitários:

Tipo	Data	Nome Comercial	Produto/hectare	Nº Lote	Débito Pulverizador
Fungicidas	20-05-2009	Prosper	0,6 l/ha	030831-c	400 l/ha
Fungicidas	12-06-2009	Prosper	0,6 l/ha	030831-c	400 l/ha
Fungicidas	20-05-2009	Mevaxil combi	2 kg/ha	141604	400 l/ha
Fungicidas	12-06-2009	Melody	1,3 kg/ha	EQ 28000015	400 l/ha

Registo de Fertilizantes:

Data	Nome Comercial	Produto/hectare

Técnico: José Madeira

Figura 7.25. Ficha Fitossanitária de um talhão (vinha)

7.4.2 Guias Turísticos

O Guia Turístico para o Distrito de Portalegre, está a ser desenvolvido no âmbito do projecto Portalegre Distrito Digital, e visa disponibilizar informação de interesse turístico através de dispositivos móveis do tipo *smartphone* e *tablet*. A aplicação criada no *IVO Builder*, conta actualmente com cerca de 150 locais de interesse criados com a respectiva informação contextual, e possibilita uma experiência ao turista semelhante ao Guia Turístico de Belém apresentado na secção anterior. O *runtime* do cliente disponibilizado, é um *runtime* do IVO modificado e customizado com um *design* próprio (cores, icons, entre outros) e que apenas permite correr esta aplicação, estando disponível nos sistemas operativos Android (já disponível), iPhone (disponível a partir de Dezembro de 2012) e Windows Phone (planeado para Março de 2013). O *IVO Builder* e o *Application Builder Server*, estão instalados nos servidores do Portalegre Distrito Digital e possibilita aos seus técnicos a manutenção da aplicação, continuando a crescer o número de pontos de interesse que a aplicação fornece. A aplicação criada tem um perfil público estando disponível a todos os utilizadores que tenham o *runtime* instalado. Foram realizados testes informais nesta aplicação tendo os utilizadores expressado opiniões bastante positivas.

O outro guia turístico criado, é um guia de Barcelona pelas obras de Gaudí Património Mundial: o Parque Güell; o Palácio Güell; a Casa Milà; a Casa Vicens; a Sagrada Família; a Casa Batlló e a Cripta da Colónia Güell. Esta aplicação foi desenvolvida com conteúdos em Espanhol, encontrando-se pronta para ser testada em ambiente real (no local).

Capítulo 8

Conclusões

Conteúdo

Resumo

Este capítulo apresenta as principais conclusões do trabalho realizado. É feita uma síntese da motivação que levou à realização do trabalho apresentado, sendo apresentadas as principais contribuições seguindo-se uma análise crítica dos resultados obtidos. Por fim são apresentadas algumas considerações relativas ao trabalho futuro.

8.1 Síntese e Contribuições

Esta dissertação apresenta um conjunto de contribuições originais em duas áreas principais: a sensibilidade ao contexto e a computação ubíqua, com especial foco na questão da falta de suporte aos utilizadores finais para o desenvolvimento de aplicações sensíveis ao contexto. Assim, a principal contribuição desta dissertação, refere-se à disponibilização de uma plataforma que dá resposta a esta questão. Esta plataforma permite construir e executar aplicações sensíveis ao contexto que utilizam *smartphones* e *tablets* como dispositivos de interacção ubíqua. Ela foi implementada de forma a permitir que os utilizadores finais possam criar aplicações sem necessidade de escrever qualquer código. As aplicações são desenhadas com as ferramentas de

construção disponibilizadas pela plataforma seguindo a metodologia de desenvolvimento apresentada no Capítulo 5 a qual visa proporcionar uma abordagem sistémica para desenvolver estas aplicações.

Esta plataforma, chamada de *Integrated Virtual Operator* ou simplesmente IVO, segue os requisitos genéricos de desenvolvimento de aplicações sensíveis ao contexto enunciados na Secção 2.2.5 e os seguintes princípios orientadores:

- Abordagem de Desenho Centrada no Utilizador;
- Desenho Baseado em Cenários;
- Modelo *Event-Condition-Workflow* ou ECW;
- Programação Visual e de Controlo de Fluxo;
- Linguagem de Marcação para Descrever as Aplicações;
- Comunicação Distribuída e Transparente.

Com base nestes princípios e no conjunto de cenários de uso apresentados na Secção 3.2, foi definida a arquitectura de referência da plataforma IVO a qual é constituída por três componentes fundamentais:

- *Composite Tools*;
- *Application Builder Server*;
- *Mobile Clients*.

Para cada um destes componentes foi definido um conjunto de requisitos funcionais e não funcionais que resultaram na implementação apresentada no Capítulo 6. As ferramentas de composição (*Composite Tools*) permitem a edição de ficheiros de acordo com a linguagem de marcação usada para descrever as aplicações, chamada de *IVO Markup Language* (IVOML) a qual é descrita no Capítulo 4 e considerada uma contribuição importante desta dissertação. Esta linguagem possibilita a representação de todos os elementos que constituem uma aplicação incluindo a informação contextual, a situação a que os contextos se aplicam, os formulários, *quizzes* e *workflows* que deverão ser executados no dispositivo móvel sempre que a respectiva situação se verificar. A opção pela representação das aplicações numa linguagem de marcação baseada em XML permite que estas possam ser facilmente partilhadas, podendo correr numa variedade de sistemas operativos sem necessidade de compilações, desde que estes forneçam um *runtime* como o implementado nesta dissertação para Android.

O modelo conceptual de funcionamento dos clientes (*Mobile Clients*), chamado de *Event-Condition-Workflow* ou ECW, permite que os *workflows* sejam iniciados por eventos que resultam da

avaliação de um mecanismo de regras do tipo *if-then* significando coisas como "*se eu...*" ou "*quando eu...*" estou numa determinada situação, executar determinado conjunto de acções que correspondem ao *workflow* definido. A maior parte das aplicações sensíveis ao contexto pode ser descrita desta forma [14].

A avaliação da plataforma foi feita ao nível das ferramentas de composição e do cliente, e teve três objectivos principais: avaliar as funcionalidades do sistema, avaliar a experiência do utilizador na interacção com o sistema e ainda identificar eventuais problemas.

Esta dissertação foi objecto das seguintes publicações em conferências internacionais de grande impacto:

1. **Realinho, V.**, Dias, A.E. and Romão, T. (2011) Testing the Usability of a Platform for Rapid Development of Mobile Context-Aware Applications. *In Proceedings of Human-Computer Interaction - INTERACT 2011*, pp. 521-536 (Lisbon, Portugal, September 5-9, 2011), Springer Berlin/Heidelberg. DOI=10.1007/978-3-642-23765-2_36

2. **Realinho, V.**, Romão, T., Birra, F. and Dias, A.E. (2011) Building Mobile Context-aware Applications for Leisure and Entertainment. *In Proceedings of 8th International Conference on Advances in Computer Entertainment Technology - ACE 2011* (Lisbon, Portugal, November 8-11, 2011), ACM Press. DOI=10.1145/2071423.2071459

3. **Realinho, V.**, Romão, T., Birra, F. and Dias, A.E. (2011) Rapid Development of Mobile Context-aware Applications with IVO. *In Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology - ACE 2011* (Lisbon, Portugal, November 8-11, 2011), ACM Press. DOI=10.1145/2071423.2071532

4. Graça, S., Oliveira, J.F. and **Realinho, V.** (2012) WorldPlus: An Augmented Reality Application with Georeferenced content for smartphones - the Android example. *In Proceedings of 20th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision - WSCG 2012* (Pilsen, Czech Republic, June 25-28, 2012), Journal of WSCG 2012.

5. **Realinho, V.**, Romão, T. and Dias, A.E. (2012) An Event-Driven Workflow Framework to Develop Context-Aware Mobile Applications. *In Proceedings of Mobile and Ubiquitous Multimedia - MUM 2012* (Ulm, Germany, December 4-6, 2012), ACM Press. DOI= 10.1145/2406367.2406395.

Está ainda em preparação um artigo a submeter a uma revista.

8.2 Conclusões

O trabalho realizado conduziu ao desenvolvimento e avaliação da plataforma IVO, tendo globalmente os resultados dessa avaliação sido bastante positivos, com os utilizadores a expressarem sentimentos muito encorajadores e afirmando-se dispostos a usá-la, porque a acharam bastante útil.

A avaliação das ferramentas de composição (*Composite Tools*) permitiu concluir que até utilizadores que nunca tiveram contacto com as ferramentas de composição conseguem facilmente construir aplicações sensíveis ao contexto. Esta avaliação mostrou ainda que o ambiente de programação visual facilita o rápido desenvolvimento deste tipo de aplicações.

As aplicações criadas com estas ferramentas permitiram efectuar uma avaliação da utilidade das ferramentas na construção de vários tipos de aplicação. Uma aplicação como o guia turístico para a zona de Belém utilizado nos testes pode demorar cerca de dois dias a criar por um utilizador experiente. Grande parte deste tempo diz respeito à pesquisa e criação da informação contextual incluindo-se neste caso a selecção/edição de vídeos. Uma aplicação com menos riqueza de informação poderá demorar muito menos tempo.

A utilização de *smartphones* e *tablets* como dispositivos de interacção ubíqua possibilita uma ampla utilização da plataforma, uma vez que estes tipos de dispositivos têm tido um crescimento enorme e uma grande procura por parte dos utilizadores. Cada vez mais, eles apresentam maiores capacidades de computação, de ligação em rede e de sensoriamento, sendo por isso ideais para a utilização em aplicações sensíveis ao contexto. A existência de modelos no mercado com écrans de várias dimensões possibilita uma maior adequação às necessidades em termos de visualização da informação e de introdução de dados. Para uma utilização mais pessoal, o dispositivo do tipo *smartphone* com écran mais reduzido parece ser mais adequado por possibilitar maior facilidade de transporte. Já os *tablets* permitem uma utilização onde haja necessidade de visualizar informação com mais detalhe ou a necessidade de introdução de dados mais facilmente, como no caso de uso de formulários. Neste sentido, acredita-se que os *tablets* possam ter uma utilização mais profissional, tal como ilustrado nesta dissertação no cenário de uso 3.

8.3 Trabalho Futuro

De seguida são apresentadas algumas sugestões de trabalho futuro de forma a complementar e melhorar o trabalho já realizado:

- Realização de testes de utilidade em situações do mundo real, em áreas de aplicação diversificadas com alguns projectos já iniciados e outros a iniciar brevemente como na área da logística, na área do ambiente com um projecto de gestão de resíduos biomássicos e na área das acessibilidades com um projecto de orientação em espaços urbanos para cidadãos com necessidades especiais;

- Melhorar as ferramentas de construção dotando-as de módulos que possibilitem testar e depurar as aplicações criadas;

- Testar o IVO em ambientes de interior, usando QR-Codes e *tags* NFC e aproveitando os *smartphones* existentes no mercado que já possuem leitores de NFC;

- Também já em curso está a migração dos clientes IVO para outras plataformas móveis nomeadamente o iPhone e o Windows Phone.

Referências Bibliográficas

[1] M. Weiser, "The Computer for the 21st Century", *Scientific American*, vol. 265, no. 3, pp. 94–104, 1991.

[2] M. Kaku, *A Física do Futuro*, 1ª Edição. Editorial Bizâncio, 2011, p. 463.

[3] A. Greenfield, *Everyware: The Dawning Age of Ubiquitous Computing*. New Riders Publishing, 2006, p. 272.

[4] W. Brett, L. Matt, L. Brian, and S. J. P. Kristofer, "Smart Dust: Communicating with a Cubic-Millimeter Computer", *Computer*, vol. 34, no. 1, pp. 44–51, 2001.

[5] R. Want, G. Borriello, T. Pering, and K. I. Farkas, "Disappearing Hardware", *IEEE Pervasive Computing*, vol. 1, no. 1, pp. 36–47, 2002.

[6] B. Rafael, B. Jan, R. Michael, and G. S. Jennifer, "The Smart Phone: A Ubiquitous Input Device", *IEEE Pervasive Computing*, vol. 5, no. 1, p. 70, 2006.

[7] IDC, "International Data Corporation", 2010. [Online]. Available: http://www.idc.com/about/viewpressrelease.jsp?containerId=prUS22560610. [Accessed: 16-Jan-2011].

[8] K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*, Third Edit. John Wiley & Sons, 2010.

[9] A. K. Dey, R. Hamid, C. Beckmann, I. Li, and D. Hsu, "a CAPpella: programming by demonstration of context-aware applications", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2004, pp. 33–40.

[10] W. N. Schilit, "A System Architecture for Context-Aware Mobile Computing", PhD Thesis, Columbia University, New York, 1995.

[11] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster, "The Anatomy of a Context-Aware Application", *Wireless Networks*, vol. 8, pp. 187–197, 2002.

[12] A. K. Dey, D. Salber, and G. D. Abowd, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", *Human-Computer Interaction*, vol. 16, no. 2, pp. 97–166, 2001.

[13] Y. Li, J. I. Hong, and J. A. Landay, "Topiary: a tool for prototyping location-enhanced applications", in *Proceedings of the 17th annual ACM symposium on User interface software and technology*, 2004, pp. 217–226.

[14] A. K. Dey, T. Sohn, S. Streng, and J. Kodama, "iCAP: Interactive Prototyping of Context-Aware Applications", *Pervasive Computing*, vol. 3968, pp. 254–271, 2006.

[15] G. Ghiani, F. Paternò, C. Santoro, and L. D. Spano, "UbiCicero: A location-aware, multi-device museum guide", *Interacting with Computers*, vol. 21, no. 4, pp. 288–303, 2009.

[16] C. Santoro, F. Paternò, G. Ricci, and B. Leporini, "A Multimodal mobile museum guide for all", in *Proceedings of Mobile Interaction with the Real World*, 2007, vol. 24, pp. 21–25.

[17] G. Ghiani, F. Paternò, C. Santoro, and L. D. Spano, "Enhancing Mobile Museum Guides with Public Displays." 2008.

[18] U. Hansmann, *Pervasive Computing: The Mobile World*. Springer, 2003.

[19] M. Castells, *The Rise of the Network Society, The Information Age: Economy, Society and Culture*. Wiley-Blackwell, 2000, p. 594.

[20] S. Poslad, *Ubiquitous Computing: Smart Devices, Environments and Interactions*, vol. 1st. Wiley & Sons, 2009.

[21] B. N. Schilit, M. M. Theimer, and B. B. Welch, "Customizing Mobile Applications", in *Proceedings of USENIX Mobile LocationIndependent Computing Symposium*, 1993, pp. 129–138.

[22] P. Brown, J. Bovey, and X. Chen, "Context-Aware Applications: From the Laboratory to the Marketplace", *IEEE Personal Communications*, vol. 4, no. 5, pp. 58–64, 1997.

[23] N. Ryan, J. Pascoe, and D. Morse, "Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant", *Computer Applications in Archaeology*, 1998.

[24] A. K. Dey, "Providing Architectural Support for Building Context-Aware Applications." Georgia Institute of Technology, 2000.

[25] M. Keidl and A. Kemper, "Towards Context-Aware Adaptable Web Wervices", in *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters - WWW Alt. '04*, 2004, pp. 55–65.

[26] P. Korpipaa, J. Mantyjarvi, J. Kela, H. Keranen, and E.-J. Malm, "Managing context information in mobile devices", *IEEE Pervasive Computing*, vol. 2, no. 3, pp. 42–51, Jul. 2003.

[27] Z. Maamar, N. Narendra, and S. Sattanathan, "Towards an Ontology-Based Approach for Specifying and Securing Web Services", *Information and Software Technology*, vol. 48, no. 7, pp. 441–455, Jul. 2006.

[28] A. K. Dey, "Understanding and Using Context", *Personal and Ubiquitous Computing*, vol. 5, pp. 4–7, 2001.

[29] A. K. Dey and G. D. Abowd, "Towards a Better Understanding of Context and Context-Awareness", in *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing - HUC '99*, 1999, pp. 304–307.

[30] D. R. Morse, S. Armstrong, and A. K. Dey, "The What, Who, Where, When, and How of Context-Awareness", 2000.

[31] K. N. Truong, G. D. Abowd, and J. A. Brotherton, "Who, What, When, Where, How: Design Issues of Capture & Access Applications", in *UBICOMP 2001: Ubiquitous Computing*, 2001, vol. 220, pp. 209–224.

[32] B. N. Schilit, N. Adams, and R. Want, "Context-aware Computing Applications", in *Workshop on Mobile Computing Systems and Applications*, 1994, pp. 85–90.

[33] G. Chen and D. Kotz, "A Survey of Context-Aware Mobile Computing Research", Department of Computer Science, Dartmouth College, Hanover, NH, USA, 2000.

[34] X. H. Wang, T. Gu, and H. K. Pung, "Ontology Based Context Modeling and Reasoning using OWL", in *IEEE Annual Conference on Pervasive Computing and Communications*, 2004, pp. 18–22.

[35] J. I. Hong and J. A. Landay, "An Architecture for Privacy-Sensitive Ubiquitous Computing", in *Proceedings of the 2nd International Conference on Mobile systems, Applications, and Services - MobiSYS '04*, 2004, pp. 177–189.

[36] P. Brézillon and J. Pomerol, "Contextual Knowledge Sharing and Cooperation intelligent Assistant Systems.", *Le Travail Humain*, vol. 62, no. 3, pp. 223–246, 1999.

[37] J. Pascoe, "Adding Generic Contextual Capabilities to Wearable Computers", in *Proceedings of the 2nd IEEE International Symposium on Wearable Computers (ISWC '98)*, 1998.

[38] H. Chen, "An Intelligent Broker Architecture for Pervasive Context-Aware Systems", PhD Thesis, University of Maryland, 2004.

[39] R. Want, A. Hopper, V. Falcão, and J. Gibbons, "The Active Badge Location System", *ACM Transactions on Information Systems*, vol. 10, no. 1, pp. 91–102, 1992.

[40] F. Bennett, T. Richardson, and A. Harter, "Teleporting-making applications mobile", in *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, 1994, pp. 82–84.

[41] M. Lamming and M. Flynn, "Forget-me-not: Intimate Computing in Support of Human Memory", in *Proceedings of FRIEND21 Symposium on Next Generation Human Interfaces*, 1994, vol. 94, pp. 2–4.

[42] K. Hinckley, J. Pierce, M. Sinclair, and E. Horvitz, "Sensing techniques for mobile interaction", *Proceedings of the 13th annual ACM symposium on User interface software and technology UIST 00*, vol. 2, pp. 91–100, 2000.

[43] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker, "Agile Application-aware Adaptation for Mobility", *ACM SIGOPS Operating Systems Review*, vol. 31, no. 5, pp. 276–287, 1997.

[44] F. Perich, "A Service for Aggregating and Interpreting Contextual Information", Palo Alto, 2002.

[45] T. Winograd, "Architectures for Context", *Human-Computer Interaction*, vol. 16, no. 2, pp. 401–419, Dec. 2001.

[46] J. Hong and J. Landay, "An Infrastructure Approach to Context-Aware Computing", *Human-Computer Interaction*, vol. 16, no. 2, pp. 287–303, Dec. 2001.

[47] J. Indulska and P. Sutton, "Location Management in Pervasive Systems", *In Proceedings of the Australasian Information Security Workshop Conference on ACSW Frontiers 2003*, vol. 34, no. vi, pp. 143–151, 2003.

[48] A. Schmidt and K. Van Laerhoven, "How to Build Smart Appliances", *IEEE Personal Communications*, vol. 8, no. 4, pp. 66–71, 2001.

[49] K. Raatikainen, H. B. Christensen, and T. Nakajima, "Application Requirements for Middleware for Mobile and Pervasive Systems", *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 6, no. 4, pp. 16–24, 2002.

[50] K. Henricksen, J. Indulska, T. McFadden, and S. Balasubramaniam, "Middleware for distributed context-aware systems", *Lecture Notes in Computer Science*, vol. 3760, pp. 846–863, 2005.

[51] J. Forstadius, O. Lassila, and T. Seppanen, "RDF-Based Model for Context-Aware Reasoning in Rich Service Environment", in *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops - PERCOMW 2005*, 2005, pp. 15–19.

[52] T. Strang and C. Linnhoff-Popien, "A Context Modeling Survey", in *Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing (September 2004)*, 2004, pp. 33–40.

[53] A. Zimmermann, A. Lorenz, and M. Specht, "Applications of a Context-Management System", vol. 3554/2005, Springer Berlin/Heidelberg, 2005, pp. 556–569.

[54] A. Held, S. Buchholz, and A. Schill, "Modeling of Context Information for Pervasive Computing Applications", 2002, pp. 1–6.

[55] J. Bauer, "Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic", vol. Diplomarbe. Technische Universit at Berlin, Berlin, 2003.

[56] K. Henricksen, J. Indulska, and A. Rakotonirainy, "Generating Context Management Infrastructure from High-Level Context Models", in *Proceedings of the industry track of the 4th International Conference on Mobile Data Management*, 2003, pp. 1–6.

[57] T. A. Halpin, *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*. San Francisco: Morgan Kaufman Publishers, 2001, p. 976.

[58] B. Bouzy and T. Cazenave, "Using the Object Oriented Paradigm to Model Context in Computer Go", in *Proceedings of the First International and Interdisciplinary Conference on Modeling and Using Context*, 1997.

[59] K. Cheverst, K. Mitchell, and N. Davies, "Design of an Object Model for a Context Sensitive Tourist GUIDE", *Computers and Graphics*, vol. 23, no. 6, pp. 883–891, 1999.

[60] J. McCarthy and S. Buvac, "Formalizing Context (Expanded Notes)", vol. Technical . Stanford University, Stanford, CA, USA, 1994.

[61] S. Bechhofer, F. Van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, and L. Stein, "Web Ontology Language (OWL) Reference, W3C Recommendation." 2004.

[62] H. Chen, T. Finin, and A. Joshi, "Using OWL in a Pervasive Computing Broker", *Workshop on Ontologies in Agent Systems*, 2003.

[63] T. Strang, C. Linnhoff-Popien, and K. Frank, "CoOL: A Context Ontology Language to enable Contextual Interoperability", in *Proceedings of International Federation For Information Processing*, 2003, vol. 2893, pp. 236–247.

[64] K. Henricksen, S. Livingstone, and J. Indulska, "Towards a Hybrid Approach to Context Modeling, Reasoning and Interoperation", in *Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning and Management*, 2004, pp. 54–61.

[65] V. Santos, "CEManTIKA: A Domain-Independent Framework for Designing Context-Sensitive Systems", D.Sc. Thesis, Universidade Federal de Pernambuco, 2008.

[66] R. Power, "Topic Maps for Context Management", in *Workshop on Adaptive Systems for Ubiquitous Computing at the 1st International Symposium on Information and Communication Technologies*, 2003, pp. 199–204.

[67] J. Jing, K. Huff, B. Hurwitz, H. Sinha, B. Robinson, and M. Feblowitz, "WHAM: Supporting Mobile Workforce and Applications in Workflow Environments", in *RIDE '00 Proceedings of the 10th International Workshop on Research Issues in Data Engineering*, 2000, pp. 31–38.

[68] D. Chakraborty and H. Lei, "Pervasive Enablement of Business Processes", in *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications - PERCOM '04*, 2004, pp. 87–97.

[69] Y. Cho, J. Han, J. Choi, and C.-W. Yoo, "A uWDL Handler for Context-Aware Workflow Services in Ubiquitous Computing Environments", in *Embedded and Ubiquitous Computing* –

EUC 2005 Workshops, vol. 3823, T. Enokido, L. Yan, B. Xiao, D. Kim, Y. Dai, and L. T. Yang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 131–140.

[70] J. Jang, A. Fekete, P. Greenfield, and S. Nepal, *An Event-Driven Workflow Engine for Service-based Business Systems*. IEEE, 2006, pp. 233–242.

[71] M. Wieland, O. Kopp, D. Nicklas, and F. Leymann, "Towards Context-aware Workflows", in *CAiSE'07 Proceedings of the Workshops and Doctoral Consortium Vol2*, 2007, pp. 577–591.

[72] L. Ardissono, R. Furnari, A. Goy, G. Petrone, and M. Segnan, "Context-Aware Workflow Management", in *Web Engineering - Lecture Notes in Computer Science*, vol. 4607, L. Baresi, P. Fraternali, and G.-J. Houben, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 47–52.

[73] G. Hackmann, M. Haitjema, C. Gill, and G.-C. Roman, "Sliver: A BPEL Workflow Process Execution Engine for Mobile Devices", in *Proceedings of 4th International Conference on Service Oriented Computing - ICSOC '06*, 2006, pp. 503–508.

[74] L. Pajunen and S. Chande, "Developing Workflow Engine for Mobile Devices", *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*, pp. 279–279, Oct. 2007.

[75] A. Z. Abbasi, M. U. Ahsan, Z. A. Shaikh, and Z. Nasir, "CAWD: A tool for designing context-aware workflows", *In Proceedings of 2nd International Conference on Software Engineering and Data Mining - SEDM 2010*, vol. 95, no. 10, pp. 128–133, 2010.

[76] F. Tang, M. Guo, M. Dong, M. Li, and H. Guan, "Towards Context-Aware Workflow Management for Ubiquitous Computing", in *2008 International Conference on Embedded Software and Systems*, 2008, pp. 221–228.

[77] U. Dayal, M. Hsu, and R. Ladin, "Organizing long-running activities with triggers and transactions", *ACM SIGMOD Record*, vol. 19, no. 2, pp. 204–214, May 1990.

[78] H. Lieberman, F. Paterno, and V. Wulf, *End User Development*. Kluwer/ Springer, 2006.

[79] B. A. Nardi, *A small matter of programming: perspectives on end user computing*. MIT Press, 1993, p. 162.

[80] A. J. Ko, B. Myers, M. B. Rosson, G. Rothermel, M. Shaw, S. Wiedenbeck, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, and H. Lieberman, "The State of the Art in End-User Software Engineering", *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, pp. 1–61, Apr. 2011.

[81] G. Fischer, *End-User Development and Meta-design: Foundations for Cultures of Participation*, vol. 5435. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 3–14.

[82] G. Fischer, E. Giaccardi, Y. Ye, A. G. Sutcliffe, and N. Mehandjiev, "Meta-Design: A Manifesto for End-User Development", *Communications of the ACM*, vol. 47, no. 9, pp. 1–7, 2004.

[83] J. Rubin and D. Chisnell, *Handbook of Usability Testing: How to plan, design and conduct effective tests*, vol. 47, no. 3. Wiley-India, 2008, p. 384.

[84] R. Dssouli and S. Some, "A Service Creation Environment Based on Scenarios", *Information Software Technology*, vol. 41, no. 11/12, p. 697, 1999.

[85] J. Carroll, "Five Reasons for Scenario-based Design", *Interacting with Computers*, vol. 13, no. 1, pp. 43–60, 2000.

[86] J. Kim, M. Kim, and S. Park, "Goal and Scenario Based Domain Requirements Analysis Environment", *Journal of Systems and Software*, vol. 79, no. 7, pp. 926–938, 2006.

[87] C. Potts, "Using Schematic Scenarios to Understand User Needs", in *Proceedings of the conference on Designing interactive systems processes practices methods techniques - DIS 1995*, 1995, vol. 95, pp. 247–256.

[88] N. C. Shu, "Visual programming: perspectives and approaches", *IBM Systems Journal*, vol. 28, no. 4, pp. 525–547, 1989.

[89] Open Geospatial Consortium, "Keyhole Markup Language Schema", 2008. [Online]. Available: http://schemas.opengis.net/kml/. [Accessed: 11-Jun-2011].

[90] TopoGrafix, "GPX 1.1 Schema Documentation", 2009. [Online]. Available: http://www.topografix.com/GPX/1/1/. [Accessed: 11-Jun-2010].

[91] W3C, "Web Services Activity", 2002. [Online]. Available: http://www.w3.org/2002/ws/. [Accessed: 11-Jun-2010].

[92] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures", PhD thesis, University of California, 2000.

[93] T. F. W. Group, "XForms 1.1." [Online]. Available: http://www.w3.org/TR/2009/REC-xforms-20091020/.

[94] I. Open Geospatial Consortium, "OGC KML", 2008. [Online]. Available: http://www.opengeospatial.org/standards/kml. [Accessed: 25-Apr-2010].

[95] A. Fuggetta, "Software Process: A Roadmap", in *Proceedings of the Conference on The Future of Software Engineering - ICSE' 00*, 2000, vol. 97, no. 12, pp. 25–34.

[96] I. Sommerville, *Software Engineering*. Addison Wesley, 8th ed, 2006, p. 864.

[97] V. Vieira, P. Brézillon, A. C. Salgado, and P. Tedesco, "A Context-Oriented Model for Domain-Independent Context Management", *Revue dintelligence artificielle*, vol. 22, no. 5, pp. 609–627, 2008.

[98] V. Vieira, "CEManTIKA: A Domain-Independent Framework for Designing Context-Sensitive System", PhD thesis, Universidade Federal de Pernambuco, 2008.

[99] K. Henricksen and J. Indulska, "Developing Context-aware Pervasive Computing Applications: Models and Approach", *Pervasive and Mobile Computing*, vol. 2, no. 1, pp. 37–64, 2006.

[100] K. Beck, *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 1999, p. 224.

[101] K. Schwaber and J. Sutherland, "The Scrum Guide", *Framework*, vol. 2, no. July, p. 17, 2011.

[102] S. R. Palmer and J. M. Felsing, *A Practical Guide to Feature-Driven Development*. Prentice Hall, 2002, p. 304.

[103] M. Bylund and F. Espinoza, "Testing and demonstrating context-aware services with Quake III Arena", *Communications of the ACM*, vol. 45, no. 1, pp. 46–48, 2002.

[104] S. Graça, J. F. Oliveira, and V. Realinho, "WorldPlus: An Augmented Reality Application with Georeferenced content for smartphones - the Android example", in *Proceedings of 20-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision- WSCG 2012*, 2012.

[105] V. Realinho, A. E. Dias, and T. Romão, "Testing the Usability of a Platform for Rapid Development of Mobile Context-Aware Applications", in *Proceedings of Human-Computer Interaction - INTERACT 2011*, 2011, pp. 521–536.

[106] J. Benedek and T. Miner, "Measuring Desirability: New methods for evaluating desirability in a usability lab setting", in *Proceedings of Usability Professional' Association - UPA 2002*, 2002.

[107] V. Realinho, T. Romão, F. Birra, and A. E. Dias, "Building Mobile Context-aware Applications for Leisure and Entertainment", in *Proceedings of 8th International Conference on Advances in Computer Entertainment Technology - ACE 2011*, 2011.

[108] V. Realinho, T. Romão, F. Birra, and A. E. Dias, "Rapid Development of Mobile Context-aware Applications with IVO", in *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology - ACE 2011*, 2011.