Improving the Use of Interactive Systems

PROJETO DE MESTRADO

Jorge Diogo Almada Ornelas
MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

novembro | 2015

Improving the Use of Interactive Systems

PROJETO DE MESTRADO

Jorge Diogo Almada Ornelas
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTADOR
José Luís Cardoso da Silva

CO-ORIENTADOR
João Carlos Cardoso da Silva

UNIVERSIDADE da MADEIRA

Improving the Use of Interactive Systems

Jorge Diogo Almada Ornelas

Constituição do júri:

Prof.ª Doutora Karolina Baras,

Prof.º Doutor Pedro Campos,

Prof.º Doutor José L. Silva.

Fevereiro 2016

Funchal – Portugal

Agradecimentos

Não poderia começar de outra forma, que não agradecendo ao meu orientador, o professor Doutor José Luís Cardoso da Silva, da Faculdade de Ciências Exatas e da Engenharia da Universidade da Madeira, pela disponibilidade demonstrada e por todo o apoio e orientação continuamente prestados, no sentido de tornar possível a realização deste projeto. Aproveito ainda para referir que foi um privilégio trabalhar sob a sua supervisão.

Agradeço também ao professor João Carlos Silva, do departamento de Tecnologias do Instituto Politécnico do Cávado e do Ave, por me ter co-orientado ao longo de todo o trabalho desenvolvido.

Pretendo agradecer ao Sr. Raimund Hocke, o atual desenvolvedor e *maintainer* do projeto SikuliX, pelo suporte prestado relativamente à ferramenta, e ainda, às equipas responsáveis pela ferramenta HAMSTERS (nomeadamente à Srª. Célia Martinie) e pelo CTTE (ao Sr. Fabio Paternò, em especial), pela disponibilidade e esclarecimentos dados sempre que solicitados.

Também quero deixar um agradecimento especial aos meus colegas do Mestrado em Engenharia Informática da Universidade da Madeira, pela motivação e ajuda que desde o início me prestaram.

Não menos importante, expresso a minha gratidão aos meus pais, José Ornelas e Maria Odélia Almada, e irmã, Vânia, por todo o alento, suporte, paciência e atenção para comigo.

Para terminar, um grande obrigado a todos os que contribuíram para que este desafio tivesse sido superado com sucesso.

Resumo

Os avanços que se têm verificado, quer ao nível das técnicas de *design* quer de usabilidade, não são ainda suficientes para ultrapassar alguns dos desafios enfrentados no uso de sistemas interativos. Sendo os utilizadores finais, em especial os principiantes, os mais prejudicados. Estes têm que se adaptar aos diferentes sistemas que utilizam para realizar as suas tarefas diárias. Ou por tentativa e erro ou através da obtenção de ajuda, têm de aprender qual a sequência de passos e como realizá-los para executar uma tarefa. Apesar de alguns sistemas interativos já oferecerem estilos de interação que atendem às necessidades dos diferentes tipos de utilizadores, isso pode não ser suficiente e muitos ainda não o fazem.

É nesse sentido que aqui se apresenta uma abordagem de suporte à utilização de sistemas interativos. Esta faz uso de modelos de tarefas enriquecidos e da computação orientada por imagem, para conseguir a automatização de tarefas. A ferramenta USS (*User Support System*) serve de base à adaptação de sistemas interativos às necessidades dos utilizadores. A abordagem abrange ainda outras duas aplicações: 1) Sistema de Ajuda, que permite a adição de ajuda baseada na demonstração, facilitando a aprendizagem e o uso de um sistema interativo; e 2) Sistema de Simplificação, que permite a integração, numa GUI simplificada, de tarefas relativas a diferentes sistemas interativos.

A abordagem pretende melhorar a eficiência dos utilizadores e reduzir os desafios que estes enfrentam ao tentar executar tarefas, utilizando um ou vários sistemas interativos. O trabalho é ilustrado por meio de alguns casos de estudo, juntamente com a discussão dos resultados de viabilidade e utilidade do mesmo.

Palavras-Chave: Automação, Modelação de Tarefas, Sistemas Interativos, Computação Orientada a Imagem, Script, Sistemas de Suporte ao Utilizador, Ajuda baseada na Demonstração

Abstract

The advances achieved at the design or usability techniques level, are still not enough to overcome some of the challenges faced in the use of interactive systems. End-users, and particularly beginners, are the most affected. They have to adapt to different systems they use to perform their daily tasks. They have to learn, or by trial and error or by getting help, what sequence of steps should follow and how to accomplish them to perform a task. Some interactive systems offer different interaction styles to attend different user needs, but the great majority still do not. This is sometimes not enough and several systems do not even do it yet.

We present here an approach to support the use of interactive systems by task automation that relies on enriched task models and image-driven computing. USS (*User Support System*) tool is the basis for the adaptation of interactive systems to users' needs. The approach also covers two other applications: 1) Help System, that enables the addition of demonstration-based help to existing interactive systems, making their learning and use easier, and 2) Simplifying System, for the integration of tasks related to different interactive systems, in a new simplified GUI.

The approach aims to improve the users' efficiency and reduce the challenges they face when trying to perform tasks using one or multiple interactive systems. The work is illustrated by means of some case studies together with the discussion of the viability and utility of it.

Keywords: Automation, Interactive Systems, Picture-Driven Computing, Script, Task Modelling, User Support Systems, Demonstration-based Help

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

Lista de abreviaturas (ordenadas por ordem alfabética)

API	Interface de Programação de Aplicações, do inglês *Application Programming Interface*
BSS	*Blender Support System*
CPU	Unidade Central de Processamento, do inglês *Central Processing Unit*
CTTE	*Concurrent Task Trees Environment*
FSM	Máquina de Estados Finita, o inglês *Finite-State Machine*
GUI	Interface Gráfica de Utilizador, do inglês *Graphical User Interface*
HAMSTERS	*Human-centered Assessment and Modeling to Support Task Engineering for Resilient Systems*
IDE	Ambiente de Desenvolvimento Integrado, do inglês *Integrated Development Environment*
IHC	Interação Humano-Computador
JAR	*Java ARchive*
LOA	Níveis de Automatização, do inglês *Level of Automation*
MARIAE	*Model-based lAnguage foR Interactive Applications Environment*
MPIH	Modelo do Processamento de Informação Humano
PDF	*Portable Document Format*
PTS	*Presentation Task Set*
SC	Sistema Cognitivo
SM	Sistema Motor
SO	Sistema Operativo
SP	Sistema Percetivo
TI	Tecnologias de Informação, do inglês *Information Technology*
TIC	Tecnologias de Informação e Comunicação
USS	*User Support System*
WSS	*Word Support System*
XML	eXtensible Markup Language

Índice

1 - Introdução

Os sistemas interativos têm sofrido evoluções significativas, muito devido aos constantes avanços de que a tecnologia tem sido alvo. Os progressos têm ocorrido no sentido de explorar novas formas de interação que tornem mais simples e fácil a utilização dos mesmos por parte dos utilizadores em geral [80]. Os sistemas interativos aqui tratados podem ser entendidos como sistemas que permitem uma contínua comunicação (transferência de informação) bidirecional, entre um computador e um utilizador [79], onde se destaca a imprevisibilidade das suas respostas.

Sabe-se que, desde o seu aparecimento até à realidade atual, muitas são as formas e meios de interação entre o homem e a máquina que têm sido aplicadas, passando pela tradicional interface de linha de comandos, as interfaces gráficas de utilizador, o estilo de interação pergunta - resposta, o teclado, o rato, o reconhecimento de voz, o ecrã tátil, etc. Relativamente aos estilos de interação, o mais habitual é o menu, enquanto o método de pergunta-resposta, apesar de simples e autoexplicativo, adequa-se apenas a utilizadores com baixo nível de especialização, e não é comum encontrar nos sistemas atuais. O que estes normalmente oferecem a esse tipo de utilizadores é ajuda ou tutoriais que os auxiliem na realização das tarefas.

Contudo, apesar dos avanços contínuos quer ao nível da usabilidade dos sistemas, quer da sua intuição, os utilizadores (principalmente, com baixo nível de especialização) de sistemas interativos continuam a se deparar com vários desafios, sobretudo quando os estão a usar pela primeira vez. Um deles é o facto de a generalidade dos sistemas não ter em conta as necessidades de cada tipo de utilizador e o contexto em que estão inseridos, de forma a se poder adaptar aos mesmos.

Com o objetivo de facilitar a utilização de interfaces têm surgido diversas soluções, tais como, a ajuda contextual, os tutoriais, o estilo de interação pergunta/resposta, etc. No entanto, e particularmente quando se trata da realização de tarefas inter-sistemas, nenhuma das soluções se revela totalmente adequada. É certo que os utilizadores aprendem mais rapidamente através de tutoriais ilustrados por *screenshots* do que ao ler apenas o texto [81], porém muitas vezes revelam dificuldades em localizar na interface os elementos presentes no tutorial [82]. Para isto contribui o facto de nenhuma das soluções executar a ajuda na interface, ou seja, fornecer ajuda automatizando a execução da tarefa desejada.

Posto isto, os utilizadores veem-se obrigados a se adaptar aos diferentes sistemas de forma a satisfazerem as suas necessidades. Têm de recorrer, normalmente, à procura de informação (manuais de ajuda) ou ao método de tentativa e erro para aprender a interagir com o sistema, isto é, saber que passos devem seguir para completar uma dada tarefa. Assim sendo, acredita-se que qualquer melhoria no suporte dado aos utilizadores no uso de sistemas interativos conduzirá a importantes contribuições nesta área.

Seguindo essa ideia, o trabalho aqui apresentado propõe facultar aos utilizadores um inovador método de interação com qualquer sistema interativo existente, sem qualquer acesso ao código fonte do mesmo e de forma autónoma ao Sistema Operativo (SO). Assim, pretende-se a integração de sistemas interativos independentes e, ao mesmo tempo, a adaptação dos mesmos

às necessidades dos utilizadores, com maior enfoque sobre os principiantes, com o intuito de reduzir a carga cognitiva que lhes é exigida.

A abordagem é baseada em modelos de tarefas enriquecidos e na computação orientada por imagem e pretende resolver problemas de usabilidade e eficiência na interação com interfaces gráficas de utilizador (*GUI*), recorrendo à automatização de tarefas. Neste ponto, devido à necessidade de conhecer o comportamento das *GUIs* dos sistemas interativos, os modelos de tarefas, por identificarem a interação entre o utilizador e o sistema, representam um ponto de partida ideal para alcançar o objetivo deste projeto.

De uma forma explícita, seguem-se os principais objetivos da abordagem aqui tratada:

- Adaptar, tanto quanto possível, os sistemas às necessidades do utilizador, reduzindo as dificuldades na sua utilização;
- Aumentar, consequentemente, a sua produtividade;
- Torná-los capazes de realizar tarefas que antes eram incapazes, garantindo-lhes maior independência;
- Oferecer ajuda de forma a facilitar a aprendizagem de Sistemas Interativos;
- Atrair público com conhecimentos reduzidos na área.

A abordagem aplica-se a dois diferentes contextos de utilização, o contexto do desenvolvedor e o contexto do utilizador final. Aos primeiros é disponibilizada uma ferramenta que, a partir de um modelo de tarefa e um cenário de simulação criados, permite a criação automática de scripts de automatização e o posterior desenvolvimento de novas e simplificadas *GUIs*. O segundo contexto divide-se em duas partes, uma que, via automatização de tarefas, permite que mesmo utilizadores com pouco ou nenhum conhecimento das Tecnologias de Informação (TI), sejam capazes de executar as tarefas pretendidas; e ainda outra que funciona como um sistema de ajuda (ilustrativa) complementar a qualquer sistema interativo, fornecendo ajuda baseada na demonstração (execução automatizada de tarefas).

A presente tese está estruturada da seguinte forma. No Capítulo 2 são contextualizados os conceitos que serviram de base ao desenvolvimento da abordagem, são eles, automatização, modelação de tarefas e computação orientada por imagem, bem como descritos alguns trabalhos relacionados. O Capítulo 3 diz respeito à especificação da implementação da abordagem, ilustrando as várias etapas seguidas e expondo as razões para tal. O Capítulo 4 apresenta diferentes casos de estudo através dos quais se pretende ilustrar os diferentes contextos em que a abordagem se pode aplicar. Os resultados da utilidade e viabilidade obtidos na avaliação realizada quer no uso da abordagem, quer à usabilidade das aplicações desenvolvidas (alusivas aos casos de estudo) são apresentados no Capítulo 5. Por fim, no Capítulo 6 são apresentadas as conclusões obtidas bem como as limitações encontradas e o trabalho perspetivado para o futuro.

2 – Contextualização e Estado da Arte

2.1 - Automatização

Com o passar dos anos, a tecnologia tem evoluído no sentido de simplificar e melhorar a interação entre o homem e a máquina. De forma que os sistemas se apliquem aos mais diversos propósitos e que o maior número de pessoas tenha acesso aos mesmos, novas formas de interação têm sido desenvolvidas e outras já existentes, melhoradas [2]. Tudo isto com o objetivo de aumentar a usabilidade dos sistemas interativos, tornando as interações do utilizador com o sistema mais naturais e intuitivas, e aproximá-los cada vez mais das necessidades dos utilizadores [1][2]. Mas para se conseguir criar sistemas fáceis de utilizar é preciso ter em conta diversos fatores, o que torna mais complexo todo o processo. E há um fato que é possível constatar na maioria dos sistemas interativos atuais, que é a necessidade do utilizador se adaptar ao sistema e não o contrário, o sistema adaptar-se consoante as particularidades ou necessidades de cada utilizador [3][7]. Aqui reside talvez a principal razão que levou ao desenvolvimento desta nova abordagem que visa suportar a utilização de sistemas interativos pelos utilizadores. Abordagem cujo resultado final consiste na automatização da interação humano-computador.

Primeiro que tudo é obrigatório perceber o que se entende por automatização, que é muito mais que modernização ou mesmo inovação tecnológica. No *Oxford English Dictionary*, a palavra pode significar o "controlo automático do fabrico de um produto através de um número de etapas sucessivas" ou "por extensão, o uso de dispositivos eletrónicos ou mecânicos para substituir o trabalho humano" [5]. Ainda neste ponto, mas numa ótica mais próxima da interação humano-computador, segundo Parasuraman *et al.* [5], a automatização é vista como um sistema capaz de realizar uma função (parcial ou totalmente), que foi ou poderia ser, previamente efetuada (parcial ou totalmente) por um operador humano.

Há já várias décadas que esta abordagem vem sendo explorada, nas mais diversas áreas e recorrendo a diferentes mecanismos, mas com um objetivo final comum, que é o de automatizar os processos, onde ao nível mais elevado deixa de ser requerida a ação por parte do ser humano.

2.1.1 - Automatização da Interação Humano-Computador

No contexto do projeto em causa, podemos definir a automatização da interação entre o homem e o computador [4] como a simplificação do processo de utilização de um qualquer sistema interativo por parte dos utilizadores. Ela visa reduzir o número de passos que um utilizador tem de efetuar até atingir um determinado objetivo, simplificando assim a realização de uma dada tarefa.

Antes de aplicá-la, a qualquer que seja o projeto, é essencial conhecer os diferentes níveis em que ela se caracteriza e os vários tipos de automatização existentes [4][5]. A identificação dos vários níveis é conhecida como abordagem *LOA* (*Levels of Automation*) [8], e refere-se ao grau de autonomia que um sistema goza, ao quão o operador humano está envolvido no controlo do sistema, ao nível de auxílio à realização de uma tarefa prestado pelo computador. Billings [9] definiu os *LOAs* em termos da capacidade de resposta de um sistema e da capacidade de este monitorizar a sua própria ação. A *LOA* acaba por definir a atribuição de controlo a um sistema,

entre o humano e o computador, representando o grau em que ambos estão envolvidos no funcionamento do mesmo. O objetivo desta abordagem é encontrar o nível de automatização que melhor se adequa às capacidades humanas [8], não descurando todo o contexto envolvente.

2.1.2 - Níveis de Automatização (*LOA*)

Posto isto, é importante perceber como essa classificação dos *LOAs* é atribuída, esclarecendo quais os critérios em que se baseiam para tal. Ao longo dos tempos podemos encontrar diferentes taxonomias, elaboradas por diferentes autores e em diferentes contextos.

Inicialmente, no decorrer do ano de 1978, Sheridan e Verplank apresentaram uma hierarquia de *LOAs*, aplicada a um contexto específico - o controlo de um teleoperador submarino [8][14]. Aqui, os diferentes *LOAs* foram estabelecidos com o objetivo de serem aplicados a funções de tomada de decisão e de seleção de ação [5] e, além disso, tiveram em conta quais as informações que devem ser dadas pelo sistema ao utilizador (por exemplo, questões sobre o funcionamento de tarefas) [8][15]. Com esta hierarquia, os autores concentraram-se principalmente em definir “quem" (ser humano ou computador) tinha o controlo, deixando para trás a tarefa de explicitar “como” seria feita a partilha de funções de processamento de informação, no controlo de um sistema complexo [14].

Como é possível observar na Tabela 1, a hierarquia divide-se em dez níveis, os quais são classificados entre baixa automatização (nível 1), em que o ser humano está encarregue de realizar a tarefa manualmente, e automatização total (nível 10), onde o computador é completamente autónomo [4][5][15].

Tabela 1. Níveis de Automatização de Tomada de Decisão e Seleção de Ação (Sheridan & Verplanck, 1978)

Low	1	O computador não oferece qualquer assistência, o humano deve tomar todas as decisões e ações.
	2	O computador oferece um conjunto completo das possíveis decisões/ações.
	3	Limita a seleção a apenas algumas das alternativas.
	4	Sugere uma alternativa, que o humano pode ou não seguir.
	5	Executa essa sugestão se o humano aprovar.
	6	O computador dá ao humano um tempo limitado para este poder interrompê-lo, antes de efetuar a tomada de decisão.
	7	O computador executa automaticamente, e informa, obrigatoriamente, o humano.
	8	O computador apenas informa o humano se este o solicitar.
	9	O computador decide sobre o que deve ou não informar o humano.
High	10	O computador decide tudo, de forma autónoma, ignorando o humano.

Embora seja possível, através destes níveis, obter uma melhor compreensão da automatização, eles revelam-se insuficientes para a avaliação da automatização de um sistema, pois esta deve ser feita a um grau mais profundo, isto é, função a função. Para tal ser possível, deve haver uma

descrição detalhada dessas funções, que permita dar suporte ao processo de conversão/transferência de uma função da responsabilidade do operador (humano) para o sistema, ou vice-versa [4].

Anos mais tarde surgiu um novo estudo [15] revelando uma forma mais completa de classificar os *LOAs*, que se baseia na hierarquia de Sheridan e Verplank, mas com a particularidade de associar a esses níveis algumas funções. Funções essas que proveem do modelo de processamento de informação humano, sendo traduzidas para as seguintes funções do sistema equivalentes: (1) aquisição de informação, (2) análise de informação (3), tomada de decisão e seleção de ação e (4) implementação/execução de ação [15]. O paradigma existente é assim estendido de forma a cobrir a automatização de diferentes tipos de funções num sistema homem-máquina. Ao contrário da classificação representada na Tabela 1, que tal como já foi referido, diz respeito apenas à automatização de funções de tomada de decisão e de seleção de ação (funções de saída), com esta expansão é possível alargar ao âmbito das funções que precedem uma qualquer tomada de decisão (funções de entrada) [4][5].

2.1.2.1 - Modelo do Processamento de Informação Humano

Da mesma forma que um sistema de processamento de informações é descrito, na engenharia de computação, em termos de memória, processador, os seus parâmetros e interconexões, Card *et al* [16] desenvolveram o Modelo do Processamento de Informação Humano (MPIH), composto por três subsistemas: Sistema Percetual (SP), Sistema Cognitivo (SC) e Sistema Motor (SM), com a intenção de ajudar a prever, aproximadamente, a interação humano-computador, com relação a comportamentos [17]. Ele pode ser visto como uma visão simplificada do processamento (humano) envolvido numa interação [19].

O primeiro (SP) é responsável por transportar sensações do mundo físico (estímulos externos), detetadas por sistemas/recetores sensoriais do corpo, transformando-as em representações internas. Como intermediário existe o Sistema Cognitivo, que, basicamente, conecta as entradas do Sistema Percetual com as saídas corretas do Sistema Motor. E este, após se dar o processamento percetual e cognitivo, viabiliza uma resposta do Sistema Cognitivo, traduzindo o pensamento em ação através da ativação de padrões de músculos [18][19].

Objetivamente, este permite a compreensão das capacidades e limites do ser humano, de forma que possam ser aproveitados da melhor maneira possível na conceção de interfaces, adequando-as aos seres humanos. Esse processo torna-se mais fácil se formos capazes de compreender como estes percebem o mundo à sua volta e como armazenam e processam as informações. Daí que seja de real importância que o utilizador perceba a informação apresentada numa interface, ou seja, ele deve saber como se processa a interação com a mesma, através de sinais que a complementam, sendo eles visuais, auditivos, táteis ou outros.

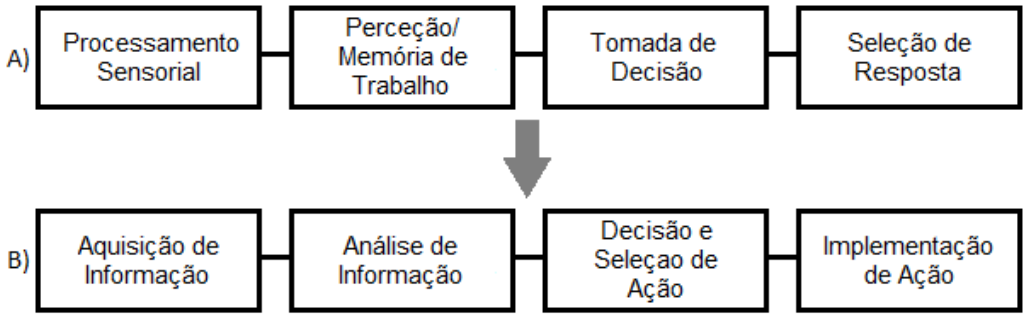


Figura 1. (A) Modelo de Processamento de Informação Humano, e (B) Funções de Sistema Equivalentes.

O MPIH (Figura 1-A) divide-se em quatro etapas - Processamento Sensorial, Perceção/Memória de Trabalho, Tomada de Decisão e Seleção de Resposta.

A primeira diz respeito à aquisição e registo, por parte de recetores sensoriais, de estímulos externos [20], apresentados através de diversas fontes de informação (cinco sentidos – visão, audição, olfato, tato e paladar). Fazem parte desta etapa o posicionamento e orientação dos ditos recetores, o próprio processamento sensorial, o pré-processamento de dados anterior à perceção total, e ainda a atenção seletiva [5]. A segunda etapa engloba a perceção consciente e a manipulação de informações processadas e recuperadas na memória de trabalho. Reúne ainda as operações cognitivas, que ocorrem anteriormente ao ponto da tomada de decisão, entre as quais se incluem a integração e a inferência. É na terceira etapa o ponto em que as decisões são tomadas, tendo por base processos cognitivos. A última etapa é responsável pela execução/implementação de uma resposta ou ação que vá de encontro com a decisão tomada na etapa anterior [4][5].

As quatro etapas contempladas no modelo não têm de seguir uma rigorosa sequência desde o estímulo até à resposta, podendo também ser coordenadas em ciclos "perceção-ação". Na prática, o desempenho da maioria das tarefas envolve uma interdependência de etapas, que se sobrepõem temporalmente ao longo do seu processamento.

Em suma, este modelo lida com a forma como as pessoas recebem, armazenam, integram, recuperam e utilizam as informações. Percebe-se que ele é apenas uma simplificação dos muitos componentes do processamento de informação humano, descobertos e aprofundados por psicólogos especialistas na área [21].

Neste ponto torna-se pertinente relembrar que o MPIH tem uma correspondência semelhante para com as funções de sistema que se seguem: Aquisição de Informação, Análise de Informação, Decisão e Seleção de Ação e Implementação da Ação, como se encontra representado na Figura 1-B.

Voltando ao assunto supracitado, referente à classificação dos *LOAs*, importa perceber que cada uma destas funções pode ser automatizada a um grau diferente (e em diferentes aspetos de uma mesma tarefa, criando vários níveis de autonomia de uma tarefa) [4][8], querendo isto dizer que os diferentes níveis de automatização podem ser implementados em qualquer uma das quatro etapas do MPIH [5].

Por exemplo, através do desenvolvimento de sensores (hardware), a primeira etapa, Processamento Sensorial, poderia ser convertida para a função de Aquisição de Informação. Da mesma forma que a segunda etapa poderia ser automatizada recorrendo à construção de algoritmos de inferência (utilizados em sistemas de recomendação, por exemplo), também seria possível, recorrendo a algoritmos capazes de efetuar uma seleção a partir de várias alternativas, automatizar a terceira etapa [4].

Concretamente, o sistema "Theater High Altitude Area Defense" (Exército dos EUA, 2007), que visa a interceção de mísseis, tem altos níveis de automatização da aquisição e análise de informação, e da tomada de decisão, mas por outro lado, o ser humano é que controla o disparo/lançamento, não requerendo automatização na execução da ação. Também os sistemas de apoio à decisão médica podem ter diferentes classificações, de acordo com o grau de automatização aplicado nas diversas fases. Existem, por exemplo, aqueles que ajudam diretamente na realização do diagnóstico, mas não no tratamento, e os que são criados diretamente para dar apoio no tratamento [22].

Contudo, o nosso objetivo central aqui não é debater a estrutura teórica do sistema cognitivo humano, servindo este apenas para dar a conhecer um pouco do que pode estar envolvido no desenvolvimento de projetos de *software*, mais concretamente, de Sistemas Interativos automatizados.

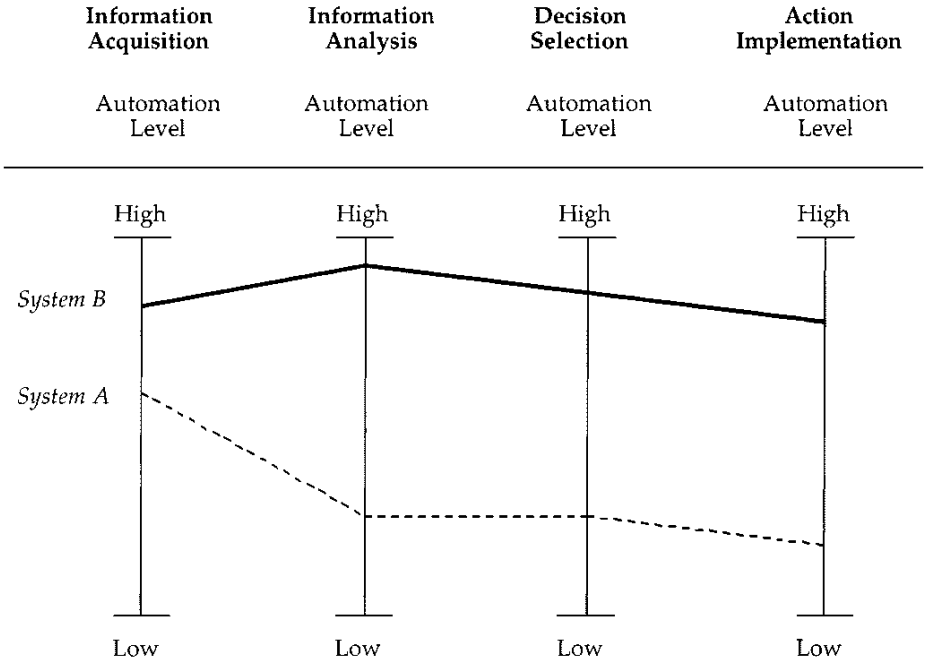


Figura 2. Níveis de Automatização para as funções independentes de aquisição de informação, análise de informação, tomada de decisão e implementação da ação [5].

Na imagem acima (Figura 2) podemos observar que num mesmo sistema podem ser aplicados diferentes níveis de automatização às várias dimensões funcionais [5][23]. Por exemplo, o sistema A foi desenhado de forma a possuir um nível moderado-alto de automatização de aquisição, e por outro lado ter um baixo nível das restantes três dimensões de automatização (análise, decisão e ação).

Uma outra hierarquia de *LOAs* foi apresentada em 1987, por Endsley [8], no contexto de utilização de sistemas dedicados a complementar a tomada de decisão humana. Nesta ficaram estipuladas as diferentes formas utilizadas para dar apoio à realização de uma tarefa. Sendo assim, uma tarefa pode ser realizada recorrendo a:

- Controlo manual, isto é, sem qualquer auxílio do sistema;
- Apoio à Decisão do operador, sob a forma de recomendações fornecidas pelo sistema;
- Inteligência artificial consensual (do sistema), com o consentimento obrigatório do operador para realizar as ações;
- Inteligência artificial monitorizada (do sistema), a ser implementada automaticamente, a menos que o operador o proíba;
- Automatização completa, sem qualquer interação por parte do operador.

A classificação aqui apresentada tem aplicação maioritária em tarefas cognitivas onde, a capacidade do operador responder e tomar decisões baseadas em informações do sistema, é essencial para o desempenho geral. Do ponto de vista conceptual, é muito semelhante à hierarquia de Sheridan e Verplank, só que esta revela-se mais simplista e resumida.

Mais tarde, Endsley e Kaber (1997, 1999) desenvolveram, com base neste trabalho, uma outra taxonomia de *LOAs* (10 níveis), que veio oferecer maior aplicabilidade a uma grande variedade de tarefas (cognitivas e psicomotoras, que exigem controlo em tempo real), e de domínios, onde se incluem o controlo de tráfego aéreo, pilotagem de aeronaves, produção avançada e controlo remoto (*teleoperations*) [8][15]. Todos estes domínios partilham, entre outras, as seguintes características:

- Vários objetivos concorrentes;
- Várias tarefas a concorrer pela atenção do operador, com diferente importância para os objetivos do sistema;
- Tarefas de elevada exigência, com recursos de tempo limitados.

Analogamente às funções do sistema de Parasuraman, Sheridan e Wicken (ver [5]), também esta taxonomia tem como base as seguintes quatro funções genéricas [8]:

- Monitorização - considerar todas as informações relevantes para a compreensão do estado do sistema;
- Geração - formular opções ou estratégias (de tarefas) para atingir os objetivos;
- Seleção - decidir sobre uma opção ou estratégia em especial;
- Implementação - concretizar a opção escolhida, através de ações de controlo sobre uma interface.

A partir daqui, cada nível é encarregue de atribuir, tanto ao operador humano como ao sistema, uma função ou uma combinação destas [15]. Na tabela que se segue (Tabela 2) estão representados os *LOAs* segundo Endsley e Kaber, juntamente com o respetivo papel desempenhado, quer pelo humano quer pela máquina, em cada uma das quatro funções. Como complemento à tabela, estão explicitados os vários *LOAs* da presente classificação:

1) Controlo Manual (*Manual Control*) – o operador humano executa todas as tarefas;

2) Suporte à Ação (*Action Support*) – o humano recebe ajuda/apoio do sistema no desempenho da ação selecionada;
3) Processamento em *Batch* (*Batch Processing*) – o ser humano é responsável por gerar e selecionar as opções a serem realizadas, mas é o sistema quem as executa, automaticamente (o sistema de *cruise control* de alguns carros é um bom exemplo);
4) Controlo Partilhado (*Shared Control*) – o ser humano é quem toma a decisão sobre que opção seguir (opções estas geradas tanto por pelo humano como pelo computador), ficando a execução das ações partilhada entre ambos;
5) Apoio à Decisão (*Decision Support*) – apesar de o ser humano poder gerar as suas próprias opções, também é disponibilizada pelo computador uma lista de opções, a partir da qual o ser humano pode tomar a sua decisão. Posteriormente, cabe ao computador executá-la, sendo este o ponto diferenciador em relação ao nível anterior;
6) Tomada de Decisão Combinada (*Blended Decision Making*) – o computador tem a responsabilidade de criar a lista de opções, selecionar uma delas e por fim executá-la, se o ser humano o aprovar. Caso contrário, este (humano) pode escolher outra das opções geradas ou gerar as suas próprias;
7) Sistema Rígido (*Rigid System*) – neste nível o sistema disponibiliza ao operador um conjunto limitado de ações, a partir do qual o operador apenas tem de escolher;
8) Tomada de Decisão Automatizada (*Automated Decision Making*) – o sistema seleciona a melhor opção, de entre uma lista de alternativas (gerada por si e complementada com sugestões do operador), e implementa-a;
9) Controlo de Supervisão (*Supervisory Control*) – o sistema é encarregue das funções de geração, seleção e implementação, enquanto o humano trata da monitorização do sistema e, quando e se necessário, intervém. Esta intervenção diz respeito a tomar uma opção diferente (das opções já geradas); e
10) Automatização Completa (*Full Automation*) – o ser humano não tem a possibilidade de intervir, sendo o sistema responsável por executar todas as ações [8].

Tabela 2. Taxonomia de LOAs de Endsley e Kaber (1999) [15].

	Roles			
Level of automation	Monitoring	Generating	Selecting	Implementing
(1) Manual control	Human	Human	Human	Human
(2) Action support	Human/computer	Human	Human	Human/computer
(3) Batch processing	Human/computer	Human	Human	Computer
(4) Shared control	Human/computer	Human/computer	Human	Human/computer
(5) Decision support	Human/computer	Human/computer	Human	Computer
(6) Blended decision making	Human/computer	Human/computer	Human/computer	Computer
(7) Rigid system	Human/computer	Computer	Human	Computer
(8) Automated decision making	Human/computer	Human/computer	Computer	Computer
(9) Supervisory control	Human/computer	Computer	Computer	Computer
(10) Full automation	Computer	Computer	Computer	Computer

O processo de automatização tem-se mostrado importante quer ao nível da realização de testes com interfaces, simulando as interações do utilizador, como também na verificação de resultados, tarefa que para o humano pode ser bastante morosa e desmotivante [6]. Além disso, é sabido que grande parte das linhas de produção das fábricas mais desenvolvidas recorrem a processos automatizados, pois tratam-se de tarefas que têm de ser, inevitavelmente, repetidas de forma exaustiva, e daí, sendo o ambiente conhecido, a automatização oferece mais celeridade e maior precisão ao processo [14]. Pode-se afirmar também que, como em tudo na vida, o que é de mais acaba por se revelar prejudicial ou mesmo infrutífero. Assim, ao aumentar

o nível de automatização de um sistema, sabemos que estamos a garantir mais benefícios, no ponto em que esta está a ser aplicada, mas do lado oposto também aumentam os riscos concomitantes. Por isso é que existem os modelos de classificação dos *LOAs*, servindo de base para uma escolha mais correta do grau de autonomia a atribuir a ambos os intervenientes, o ser humano e o computador [14].

2.2 - Computação Orientada por Imagem

Antes da própria aplicação da automatização a um qualquer sistema é necessário identificar sobre quais tarefas esta será aplicada. Este é um procedimento que tanto pode ser feito nas diferentes fases do processo de desenvolvimento do *software* alvo, como também em sistemas legados, por exemplo. E é em casos como este último, onde não é possível ter acesso ao seu código fonte, que há a necessidade de recorrer a uma forma alternativa de proceder à automatização de tarefas. É aqui que entra a Computação Orientada por Imagem (*Picture-Driven Computing*), que cada vez mais é vista como uma alternativa viável [6].

Este novo paradigma, orientado a imagens, que foi recentemente introduzido no contexto geral da programação e, mais em particular, na realização de testes de interfaces de utilizador, oferece aos desenvolvedores de *software* uma lógica totalmente alternativa. É uma abordagem que recorre a algoritmos de visão computacional para analisar, instantaneamente, o conteúdo e a evolução de uma interface gráfica (GUI) [7].

Em [12], este conceito é definido como sendo um paradigma de computação que se centra na abstração de dados da informação visual, num domínio espacial bidimensional. Essa informação representa o que é visível num dispositivo de saída gráfica, normalmente o ecrã de um computador. Por sua vez, a informação visual é, por norma, uma imagem que pode ser a totalidade ou parte do que nele é mostrado.

Nos “paradigmas-padrão” o que se verifica é que os sistemas tendem a trocar blocos de informação assentes em abstrações de dados de informação numérica e textual. Contrariamente ao que acontece no paradigma acima referenciado. Um bom exemplo é o ato de efetuar uma simples pesquisa na web, de um modelo específico de um carro. A abordagem comum consiste em especificar o pedido no campo de pesquisa através de informação textual (*keyword*), como por exemplo “Audi A4”. Por outro lado, graças aos estudos recentes, é permitido, recorrendo a serviços orientados a imagens, efetuar o pedido através da introdução de uma foto do objeto pretendido (neste caso, o carro em questão) no campo de pesquisa. Basicamente o que diferencia as duas abordagens é o mecanismo de pesquisa. Enquanto um utiliza palavras-chave, o outro recorre a imagens representativas daquilo que está a ser procurado.

Como é referido por G. Kourousias e S. Bonfiglio em [12], o paradigma da computação orientada por imagem pode ser um bom aliado das tecnologias de apoio (*assistive technologies*) no desenho de acessibilidade (*Accessibility Design*) em sistemas informáticos. O Desenho de Acessibilidade refere-se à conceção de produtos ou serviços destinados a pessoas com deficiência. Este assegura tanto o acesso direto – sem ajuda –, como o acesso indireto – compatível com tecnologias de apoio às pessoas (os leitores de ecrã de computador são um

exemplo). Em suma, as tecnologias de apoio, como o próprio nome indica, pretendem fornecer ajuda a pessoas com deficiência, garantindo-lhes maior independência [28], ao torná-las capazes de realizar tarefas que antes eram incapazes ou sendo capazes, revelavam grandes dificuldades para o conseguir. Isto pode ser conseguido através de aperfeiçoamentos ao nível da tecnologia utilizada para realizar essas tarefas ou alterando a forma de interagir com a mesma [27].

É do conhecimento geral que as Tecnologias de Informação e Comunicação (TIC) representam um grande impacto no estilo de vida da sociedade atual, nomeadamente ao nível da comunicação, interação, entretenimento, socialização e do trabalho. É também percetível que quantos mais acessíveis forem essas TICs, mais aceitação irão ter e acabarão por beneficiar, principalmente, os utilizadores com mais dificuldades e mesmo os portadores de deficiências. Entre os avanços que mais se destacam ao nível da acessibilidade encontram-se o surgimento de melhores ergonomias, novos dispositivos de entrada/saída e interfaces de utilizador aprimoradas. E este novo paradigma de computação é também um grande aliado dos avanços que se têm verificado.

2.2.1 - Visão Computacional

A área da Visão Computacional, apesar de complexa e exigente, é de elevado interesse em muitos domínios, particularmente na Medicina e na Engenharia [35]. Em termos comparativos, relativamente à tecnologia de reconhecimento de voz, podemos dizer que a visão computacional aplicada à IHC ainda se encontra atrasada [24]. Importa realçar também que, esta primeira viu, por volta de 1930, surgirem os primeiros estudos relacionados, por intermédio de Homer Dudley [31]. A partir daí, o seu crescimento e evolução têm sido progressivos, assistindo-se já a vários anos de viabilidade comercial.

Apesar disso, a visão computacional encontra-se, atualmente, num ponto singular do seu desenvolvimento, apesar dos estudos preliminares acerca do tema terem surgido por volta de 1960. No entanto, as áreas desenvolvidas estão muito dispersas, além de que, muitas ideias úteis não têm fundamentação teórica, o que acaba por dificultar o estudo do tema. Só recentemente começou a ser possível construir sistemas computacionais úteis, empregando ideias da visão computacional. O facto dos computadores e dos *imaging systems* tornarem-se mais acessíveis, em termos de custos, não é de todo a única razão para o impulso que se tem verificado no seu progresso, mas é, sem dúvida, um dos principais fatores para tal [13]. O termo *imaging* consiste basicamente na representação (visual) da forma de um objeto, ou seja, é a formação de uma imagem. Neste processo de *imaging*, importa dar ênfase à compreensão da física e geometria básicas envolvidas, que juntamente com a estatística aplicada, têm mostrado uma influência enorme nesta área.

Como é normal, derivado à juventude do paradigma orientado a imagem, ainda existem obstáculos na tentativa de se conseguir sistemas de visão computacional de qualidade, nomeadamente, a falta de velocidade e de robustez reveladas no seu funcionamento [13]. Nesta ordem de ideias, os sistemas cujo desempenho se dá a uma velocidade considerável normalmente são os que demonstram mais fragilidades nos seus resultados, enquanto os mais completos e robustos tendem a ser excessivamente lentos [24]. Também outros obstáculos mais frequentemente enfrentados são as condições de aquisição difíceis (iluminação variável, por

exemplo), oclusão, ruído, distorção geométrica, formas e topologias complexas e movimentos não rígidos [35].

Contudo, nas últimas décadas, tem-se registado um progresso significativo nas tecnologias de visão ao nível da interação humano-computador. Esse progresso tem ocorrido na direção de procurar técnicas de visão cada vez mais robustas e que operem em tempo-real, muito devido aos avanços verificados ao nível do desempenho do *hardware*, nos quais a "Lei de Moore" [34] tem dado um grande contributo [24]. Um exemplo disso é o crescente interesse na instalação de câmaras à nossa volta, e recorrer à visão computacional para observar as pessoas ("*look at people*") [13][24]. Tudo isto com o intuito de possibilitar a deteção e o reconhecimento do rosto humano, acompanhamento dos movimentos da cabeça, rosto, mãos e até do corpo, a análise de expressões faciais e movimentos corporais e ainda o reconhecimento de gestos. Esta procura de informação visual sobre as pessoas levou a diversas áreas de investigação no âmbito da visão computacional, focadas principalmente na modelação, reconhecimento e interpretação do comportamento humano [24].

A um nível mais científico, a visão computacional é a disciplina da computação cuja função é dar aos computadores a capacidade de "ver", através do processamento de imagens e/ou vídeos [24]. Para D. Forsyth e Jean Ponce [13], é vista como um serviço que utiliza métodos estatísticos para desagregar dados, recorrendo a modelos construídos com base em teorias da geometria, da física e da aprendizagem. Em suma, esta tecnologia tem como objetivo criar um modelo do mundo real, a partir de imagens. Para tal, recupera as informações úteis sobre uma cena a partir das suas projeções bidimensionais.

Já ao nível da interação humano-computador, esta tecnologia foca-se na utilização da visão como uma modalidade de entrada [24]. Aqui existe a vantagem da deteção se realizar de forma passiva e não-intrusiva, visto que não é requerido contato com o utilizador (uma câmara é o dispositivo usado).

Esta visão depende de uma consistente compreensão das câmaras (sensor) e do processo (físico) de formação da imagem, para conseguir inferir informações úteis sobre o mundo, a partir de imagens (do valor de cada *pixel* individual) [24]. Posteriormente envolve também a combinação da informação disponível em várias imagens num todo (lógico), a imposição de uma certa ordem a conjuntos de pixéis, com vista a separá-los (uns dos outros) ou a deduzir informações sobre a sua forma (*shape*), e ainda o reconhecimento de objetos, quer através de dados geométricos quer de técnicas probabilísticas [13]. Essas informações podem aludir a diferentes aspetos, tais como, a cor de uma amostra de tecido, o tamanho de um obstáculo situado à frente de um robô, a identificação do rosto de uma pessoa num sistema de vigilância, ou a localização de um tumor numa ressonância magnética [24]. Podemos dizer que os algoritmos aos quais ela se recorre são desenvolvidos com o intuito de permitir a realização, de forma automática ou semiautomática, das operações e tarefas normalmente desenvolvidas pelos (complexos) sistemas de visão dos seres vivos [35].

O ponto forte da visão computacional é que consegue extrair descrições do mundo a partir de imagens ou sequências destas, o que se revela muito útil. Por exemplo, a técnica *structure from motion* [26], utilizada na área fotográfica, permite extrair uma representação tridimensional de um qualquer objeto, e a forma como a câmara é movida, a partir de um conjunto sequencial de

imagens [13]. Esta técnica é utilizada na indústria de entretenimento, para construir modelos computacionais de edifícios (em três dimensões), onde é mantida a estrutura e ignorado o movimento. Por outro lado, para o controlo de robôs, preserva-se o movimento e a estrutura é descartada, pois, a partir de informações sobre como a câmara (inserida no robô) se está a mover, consegue-se saber o local onde este está a atuar.

Também há a noção que o trajeto percorrido pelas diferentes soluções existentes é distinto e independente, pois estão sujeitas a diversos fatores externos e claro, à aceitação dos utilizadores. Dois casos distintos são, por exemplo, os sistemas de deteção e reconhecimento faciais, que demonstraram ser boas apostas, revelando um grande sucesso comercial, e do lado oposto o reconhecimento gestual, que ainda não se conseguiu impor totalmente no mercado [13][24].

Mas, se foi possível chegar até aqui, onde algumas variantes já começam a ser bem recebidas e aceites por uma fatia considerável de mercado, também será a partir das debilidades reveladas pelas mesmas que se fomentará o despoletar de novos e mais avançados estudos de investigação. Só assim, procurando soluções que contornem esses pontos fracos é que se torna possível a evolução. Por isso acredita-se que o futuro da visão computacional será bem-sucedido. Para isso teve-se por base alguns fatos indiciadores. Começando pelas tecnologias de componentes individuais, que têm sido alvo de um considerável progresso, passando pelas áreas que se têm tornado comercialmente viáveis, nomeadamente ao nível do reconhecimento facial, a investigação neste campo, que cada vez mais vê aumentar os seus recursos, contribuindo para a origem de novas ideias, suscetíveis de serem aplicadas às tecnologias de interação baseadas na visão [24]. Todos estes fatores são indiciadores de que, mais tarde ou mais cedo, a visão computacional se tornará em mais uma tecnologia amplamente inserida no nosso dia-a-dia.

Um mero exemplo onde esta tecnologia pode ser aplicada é no caso de um utilizador que quer usar uma funcionalidade desconhecida de um programa. Assim, ele poderia pesquisar na web, por informações relacionadas, usando um *screenshot* relativo a esse elemento da GUI. Investigadores já comprovaram que através desta abordagem visual reduz-se consideravelmente o tempo despendido, em média, para encontrar informação útil [29].

Entre as operações mais comumente albergadas pelos sistemas de visão, situam-se a identificação, o *tracking* e reconhecimento de movimentos, a correspondência, a interpolação de formas (simulação) e ainda a obtenção de informação tridimensional (visão 3D). Algumas destas tarefas aqui mencionadas serão mais aprofundadas imediatamente abaixo [35].

Como já se sabe que cada uma das áreas de aplicação emprega várias e diferentes tarefas da visão computacional, de seguida expomos as mais comuns, que são:

- Reconhecimento (*Recognition*) – O principal problema da visão computacional está em determinar se uma imagem contém algum objeto, recurso ou atividade específica. Isso é feito através do reconhecimento.

- Análise de Movimento (*Motion Analysis*) – Engloba as várias tarefas relacionadas com a estimativa de movimento, onde uma sequência de imagens é processada para produzir uma estimativa da velocidade, em cada um dos pontos na imagem, ou da câmara que a gera.

- Reconstrução de Cena (*Scene Reconstruction*) – A partir de uma ou mais imagens de uma cena, esta é responsável por computar um modelo tridimensional da cena. Nos casos mais simples, esse modelo passa apenas por um conjunto de pontos 3D [39].

- Restauração de Imagem (*Image Restoration*) – Basicamente, consiste na remoção do ruído/distorção (*motion blur*, etc.) das imagens, com o intuito de aproximá-la do "original". Os métodos mais simples para remoção de ruído residem no uso de vários tipos de filtros [38]. Enquanto nos mais complexos, em primeiro lugar é feita a análise aos dados de imagem em termos das estruturas locais (como linhas, cantos ou extremidades). De seguida, tem lugar o controlo da filtração, com base em informações locais retiradas da análise [40]. Um exemplo disto é o *inpainting*, o processo de reconstruir partes perdidas ou deterioradas de imagens e/ou vídeos.

Agora, e porque é de maior interesse para o trabalho, serão aprofundadas mais detalhadamente as duas primeiras tarefas. Essas são também as mais relevantes na área e mais abrangentes ao nível das mais diversas áreas de aplicação. Relativamente à tarefa de Reconhecimento, existem três diferentes tipos - Reconhecimento de Objetos, Identificação e Deteção. O **Reconhecimento de Objetos** é um tópico central de investigação no âmbito da visão computacional. Reside no determinar que objetos estão e onde, numa imagem digital [36]. Isto é, os objetos (um ou vários), já conhecidos pelo sistema, poderem ser reconhecidos, juntamente com as suas posições bidimensionais na imagem [38]. Enquanto os seres humanos desempenham esta tarefa de forma fácil e instantânea, ainda é um desafio para os sistemas de visão computacional, havendo muitas abordagens implementadas ao longo de várias décadas [37]. A **Identificação** consiste em reconhecer uma instância individual de um objeto. Neste caso, temos como exemplos a identificação do rosto ou impressão digital de uma pessoa exata, de carateres manuscritos ou de um veículo específico [38]. O terceiro tipo, a **Deteção**, refere-se à digitalização de imagens (*image data*) sob uma condição específica. Esta baseia-se em computações relativamente simples e rápidas. Possíveis utilizações desta vertente são a deteção de possíveis células anormais, em imagens médicas ou de um veículo, num sistema de portagens automático [35][38].

Por seu lado, a Análise de Movimento pode se referir às seguintes tarefas. ***Egomotion***, que diz respeito à estimativa do movimento (3D) de uma câmara (rotação e translação) em relação a uma cena rígida, a partir de uma sequência de imagens, por ela produzidas [37][38]. A estimativa da posição de um carro em movimento, em relação às linhas de sinalização observáveis pelo mesmo, é um bom exemplo da sua aplicação. O ***Tracking*** consiste em seguir (acompanhar) os movimentos de um conjunto de pontos de interesse ou objetos (por exemplo, um ser humano) na sequência de imagens analisada [38]. E por último, o **Fluxo Ótico**, que alude ao padrão de movimento aparente de objetos, superfícies e arestas, numa cena visual, causado pelo movimento relativo entre o observador (uma câmara, por exemplo) e a cena [37]. Esse movimento aparente, que corresponde a como é que cada ponto da imagem se move em relação ao plano de imagem, resulta de como o respetivo ponto tridimensional se move na cena e de como a câmara se move em relação à cena [38].

Com isto, passa-se agora a alguns exemplos concretos da ampla variedade de aplicações da visão computacional. Algumas desde a fase mais inicial do seu desenvolvimento, como por exemplo, navegação móvel de robôs, inspeção industrial e inteligência militar, e outras mais atuais, tais

como, interação humano-computador, recuperação de imagens em bibliotecas digitais e análise de imagens médicas [13]. A realização de inspeções, por exemplo, na área industrial, pode ser feita tirando fotos ao objeto de modo a determinar se está de acordo com as especificações. No contexto militar, pode, por exemplo, ser utilizada na interpretação de imagens de satélite para cobrir a necessidade de determinar os danos causados por um bombardeamento.

2.2.2 - Ferramenta *Sikuli*

Note-se que, atualmente, a designação da ferramenta sofreu uma ligeira alteração - chama-se *SikuliX* – contudo, no âmbito deste trabalho será usada a nomenclatura comum, i.e. *Sikuli*.

Nos tempos primórdios dos computadores, os programas funcionavam à base de comandos (textuais), os quais o utilizador tinha de memorizar. Para interagir com o programa, ele tinha de os digitar, um a um, linha a linha. E a resposta não seria mais que simples linhas de texto. Só mais tarde, a partir dos anos 80, é que isso mudou, em razão do surgimento de uma abordagem nova e totalmente diferente, a incorporação de interfaces gráficas de utilizador (*GUIs*) nas aplicações informáticas [3][7]. Com isto, todos os componentes, tais como funções e dados do programa, passaram a ser representados por imagens (bidimensionais) – ícones, botões, *widgets*, janelas, etc. – o que veio facilitar toda a interação entre o utilizador e o computador, tornando-a muito mais intuitiva [29].

No entanto, esta mudança também trouxe complicações associadas, principalmente para os programadores. Isto porque, todos esses componentes constituintes das *GUIs* consistiam em blocos de código e a tarefa de desenvolver ou personalizar um programa continuava a passar pela manipulação desse mesmo código. Mas como é a partir dos obstáculos que surgem as novas ideias, isto é, na tentativa de os ultrapassar, este novo sistema – *Sikuli* – surgiu com intuito de alterar essa forma de desenvolver programas, facilitando a tarefa dos desenvolvedores. O *Sikuli* foi desenvolvido por investigadores do Instituto de Tecnologia de Massachusetts (MIT), nomeadamente, Rob Miller, Tsung-Hsiang Chang e Tom Yeh. O conteúdo deste projeto, que foi alvo de um artigo (*paper*) [33], garantiu-lhes um prémio na conferência "*User Interface Software and Technology 2009*", da ACM (*Association for Computing Machinery*) [30].

É o mais conhecido ambiente de programação que faz uso do paradigma orientado a imagens [12]. O *software* permite aos utilizadores escrever programas, que funcionarão sobre a interface gráfica de utilizador (do inglês, *GUI*) de um programa, recorrendo à utilização de capturas de ecrã (*screenshots*) dessas mesmas *GUIs* [7][12]. Assim, permite o *scripting* visual e a automatização de ações sobre qualquer interface gráfica.

Em suma, o objetivo centra-se no desenvolvimento de *scripts* - pequenos programas capazes de dar suporte à utilização de programas mais complexos, quer combinando ou estendendo as suas funcionalidades [7][29].

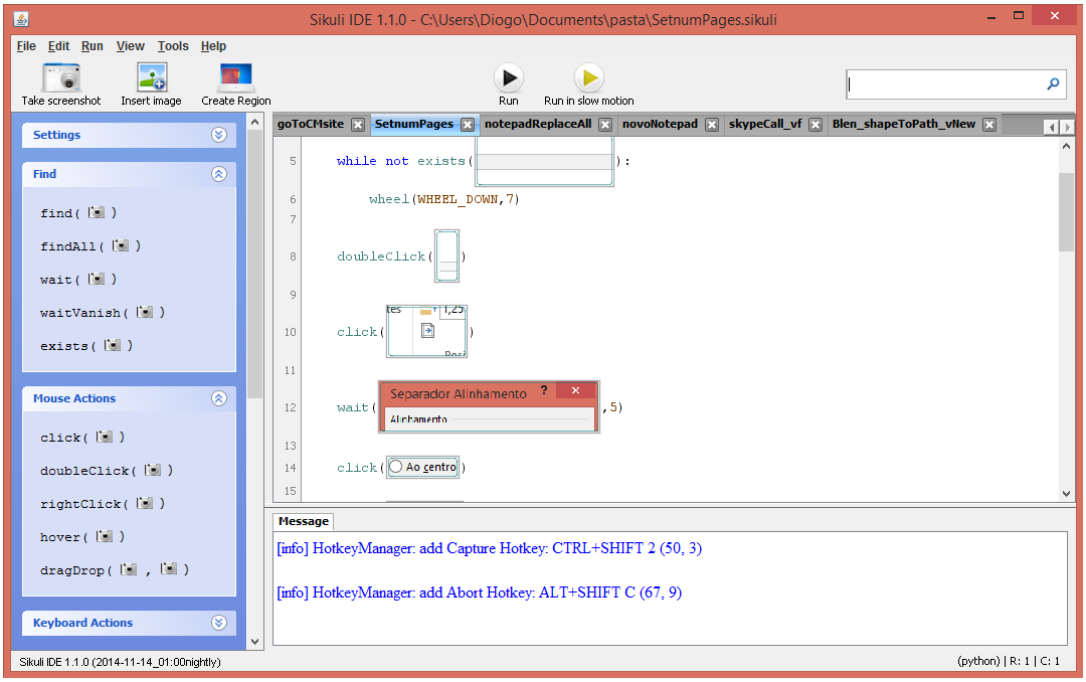


Figura 3. IDE da ferramenta *Sikuli.*

Um fator interessante é que não é requerido, ao utilizador, qualquer conhecimento em relação ao código subjacente aos programas alvo da automatização de funcionalidades [3]. Pois, quando pretender executar uma funcionalidade do programa alvo, através da *script*, o utilizador apenas tem fazer o seguinte: utilizando o *IDE* (*Integrated Development Environment*) do *Sikuli* (Figura 3) para escrever a script, há a opção de selecionar (com o rato) o respetivo fragmento da *GUI* associada, obtendo a captura da imagem e por fim, inseri-la, juntamente com a função própria, diretamente numa linha de código *Python* [3][29]. O utilizador não é o único que não precisa de obter esses conhecimentos, pois também o *Sikuli* não tem qualquer tipo de acesso ao código fonte relativo aos programas alvo. Ele apenas atenta aos pixéis no ecrã, da mesma forma que um mero utilizador quando interage com a *GUI* [29]. Tecnicamente, o processo recorre a algoritmos de visão computacional para analisar, de forma quase instantânea, o que está a acontecer no ecrã, isto é, a evolução do conteúdo da *GUI* [7]. Posto isto, é possível utilizar o *Sikuli*, sem qualquer modificação adicional, para operacionalizar com todo e qualquer programa possuidor de uma interface gráfica [3][29].

Acrescenta ainda o facto de, neste ambiente, mesmo os utilizadores principiantes serem capazes de criar os seus próprios programas, pois não há necessidade de possuir um grande domínio de programação ou *scripting* [29][30]. No entanto, requer alguma familiaridade com a linguagem de *scripting* a utilizar (neste caso foi o *Python*, apesar do *IDE* suportar também *Ruby* e *JavaScript*) [3], nomeadamente sobre o uso dos comandos básicos, tais como, "*click*", "*wait*" ou "*dragDrop*", que descrevem como serão tratados os componentes da *GUI* [29].

Uma das mais adotadas utilizações do *Sikuli* é nos grandes projetos de desenvolvimento de *software*, para realizar os testes de *software*. Devido à elevada complexidade do código, uma pequena alteração poderá originar comportamentos inadequados da *GUI* já desenvolvida, e de forma a poder detetá-los é imprescindível a realização, dia após dia, desses testes [3]. Assim, criam-se *scripts* que ficam responsáveis por testar, automaticamente, os diversos componentes

da respetiva *GUI* [30]. Desta forma garante-se uma diminuição dos custos, quer financeiros quer de tempo, decorrentes dos testes de garantia de qualidade, que normalmente são elevados [29].

Um caso de uso convencional, ilustrativo da aplicação desta tecnologia, pode ser a criação de uma *script* capaz de alertar o utilizador (enviando uma mensagem para o *smartphone* ou reproduzindo um som no próprio computador) quando a temperatura interna da *CPU* atingir um determinado valor. Necessitando para isso, apenas de um *software* que registe instantaneamente essa temperatura. A partir daí, pode captar *screenshots* dos fragmentos da interface cuja utilização será requerida pela *script* [3].

2.2.3 - Comparação com Outras Ferramentas

A computação orientada por imagem será o meio utilizado para a automatização da interação com a *GUI* das aplicações. Como já referido, esta é uma abordagem que permite a criação de *scripts* que tornam a interação com a *GUI* de qualquer sistema uma tarefa, parcial ou totalmente, automática, ou seja, podendo deixar de haver necessidade de qualquer ação por parte do utilizador.

Para tal, e como é exposto no tópico anterior, a ferramenta *Sikuli* é talvez a mais utilizada para esse efeito. Foi através de [3] que tive conhecimento da existência da mesma, e a partir daí, foi-lhe dedicada maior atenção. Após alguma investigação, em simultâneo com a realização de testes/experiências no próprio *Sikuli*, comecei a perceber melhor em que consistia e a base do seu funcionamento, optando mesmo por utilizá-la no projeto em questão. Esta é uma ferramenta, que como já foi explicado anteriormente, utiliza o reconhecimento de imagem para identificar e controlar os componentes da *GUI* das aplicações. Assim, permite aos utilizadores tirar *screenshots* de elementos da *GUI* e com eles construir *scripts* que serão executados automaticamente, auxiliando os utilizadores a alcançar o objetivo pretendido.

Contudo, além do *Sikuli*, existem também outras ferramentas alternativas com capacidades semelhantes a esta. Entre elas, foi dada maior atenção à *Automa*, à *RIATest* e à *EggPlant Functional*.

O *Automa* é uma ferramenta para o Windows que permite automatizar tarefas repetitivas na *GUI* do computador. Permite controlar o PC com o uso de comandos simples, tais como "*start*", "*click*" e "*write*", aumentando assim a produtividade do utilizador. A sequência de comandos introduzidos pode ser guardada num ficheiro, o qual pode ser reproduzido de diferentes maneiras, como por exemplo, ao clicar num botão, dentro de um intervalo de tempo especificado ou a partir de uma ferramenta de gestão de testes [66].

RIATest é uma ferramenta de automação de testes de *GUIs*, apenas para aplicações Windows. É capaz de automatizar qualquer item no ecrã, que seja acessível através da API (*Application Programming Interface*) *Windows UI Automation*. De seguida apresentam-se algumas das suas principais características [65]:

- Identifica os elementos da *GUI* da aplicação através do *Object Inspector*, utilizando recursos de localização simples mas poderosos;

- Não é necessário executar a script para verificar se existem erros de sintaxe, pois possui um *debugger* que os verifica, à medida que as *scripts* são desenvolvidas;
- Permite pausar a execução de uma script, editá-la, e depois continuar, sem reiniciar a execução; e
- Permite configurar a velocidade de execução das *scripts* [65].

Por último, a *EggPlant Functional* permite automatizar a execução de testes funcionais, através de uma abordagem baseada em padrões de imagens. Permite escrever testes de forma muito intuitiva. Utiliza algoritmos sofisticados de pesquisa, de texto e imagem, para localizar os objetos no ecrã e, assim, controlar a aplicação. Efetua a interação da mesma maneira que um utilizador. É de realçar a diferença no modelo usado por esta ferramenta para gravar e executar as *scripts*. Enquanto o *Sikuli* trata-se de uma ferramenta *stand-alone*, que grava e executa as *scripts* na mesma máquina, o *EggPlant* requer dois sistemas, um que funcione como cliente, onde se mantém a script gravada e o servidor, onde a script será executada [64]. Esta permite testar qualquer tecnologia e em qualquer plataforma, sendo feito a partir da perspetiva do utilizador, e não do código. Exemplificando, para clicar no botão "*OK*" o programa analisa o ecrã, através de algoritmos de reconhecimento de imagem, e localiza esse botão, de seguida, despoleta um evento ao nível do sistema para clicar sobre o botão [63].

2.3 - Modelação de Tarefas

Outro dos focos centrais do projeto, que importa explorar, é a modelação de tarefas. Numa visão global, este tipo de abordagens, baseadas em modelos, devem ser uma aposta firme, pois normalmente dão destaque apenas às informações relevantes e também ajudam a gerir e a lidar com a complexidade de todo um sistema. Por seu turno, os modelos, no geral, partilham as seguintes propriedades: focam-se no aspeto específico do mundo real (neste caso, a interface do utilizador ou a aplicação interativa) a ser representado, aumentam o nível de abstração e ainda tendem a ser declarativos, ao invés de procedimentais [48]. Existem diversas possibilidades onde estes podem ser aplicados. Tanto podem ser usados para descrever um sistema existente, como para definir um novo sistema, que está a ser pensado [48].

Continuando para o conceito propriamente dito, um modelo de tarefas, segundo Paternò, é uma descrição lógica das atividades a serem executadas para alcançar os objetivos do utilizador [41]. Ele é o resultado do processo de análise de tarefas, processo esse cujo objetivo é capturar a gama de possíveis comportamentos que um operador humano pode ou deve desempenhar perante o sistema, de forma a atingir os seus objetivos [10]. Assim sendo, a modelação de tarefas permite identificar as interações entre o utilizador e o sistema, com foco nos principais aspetos a considerar aquando do processo de conceção de aplicações interativas [3][43]. Ela oferece uma representação abstrata da interação do utilizador com o sistema [50]. Podemos afirmar que um modelo de tarefas envolve os seguintes elementos: objetivo, ações e objetos de domínio. Estes últimos representam os elementos que um utilizador pode ver, aceder e manipular numa interface de utilizador [47]. Os objetos, juntamente com as relações entre eles, definem um modelo de domínio [48]. Neste contexto, vamos considerar que cada tarefa, no modelo, corresponde a uma atividade lógica [49] que deve ser realizada com a intenção de alcançar um objetivo definido pelo utilizador [43][50]. Por seu turno, definimos objetivo como

sendo uma alteração desejada do estado do sistema ou a tentativa de obter informações sobre o mesmo [47]. Enquanto, por um lado, cada tarefa é associada apenas com um objetivo, por outro, cada objetivo pode estar associado a múltiplas tarefas [48]. Importa ainda adiantar que, dentro dessas tarefas, há as que se podem chamar independentes, e existem outras cuja execução está dependente da conclusão de uma outra. Todos esses elementos contribuem para ajudar a identificar o comportamento das interfaces de utilizador [48].

Os modelos de tarefas devem, sobretudo, manter uma descrição explícita das tarefas, onde é de interesse que os objetivos e as atividades a executar para alcançá-los estejam também explicitamente definidos [41]. Só assim será possível uma melhor compreensão e um entendimento mais detalhado dos passos a seguir (durante uma utilização do sistema), havendo uma maior reflexão sobre os procedimentos que levarão o utilizador a alcançar, mais eficientemente, o objetivo. E devido à análise, crítica e reelaboração, que são feitas, destes procedimentos, é possível torná-los mais simples e eficazes, aumentando assim a usabilidade do sistema [41].

Prosseguindo, como estes modelos pretendem representar o modo de utilização das aplicações, descrevem princípios acerca da maneira como os utilizadores devem interagir com as mesmas, de modo a atingirem o seu objetivo [3][7], ou seja, devem refletir a visão dos utilizadores relativamente às atividades a executar [46]. Além disso, eles apresentam as restrições (conhecimento estratégico exposto por meio de condições lógicas), temporais e/ou semânticas, que visam especificar quando é que as atividades/ações podem ser executadas ou o que elas devem cumprir/satisfazer [10]. Assim, estas devem ser respeitadas, pelo utilizador, aquando da interação com o sistema. Esta característica revela-se fundamental para se conseguir descrever, tanto as tarefas paralelas – caminhos alternativos para o mesmo objetivo, - como a permissão dinâmica e a desativação de tarefas [46]. Em suma, os modelos de tarefas além de descreverem as principais atividades que os operadores humanos podem executar, ainda nos apresentam as relações entre elas [48]. Desta forma, percebe-se que a criação dos modelos de tarefas vai de encontro a um dos princípios fundamentais de usabilidade, que é o facto do foco central recair sobre os utilizadores e as suas tarefas [47][49].

No geral, estes modelos contêm certos conceitos relacionados, representando os aspetos relevantes das tarefas e dos utilizadores. Alguns desses são comuns à maioria dos tipos de modelos de tarefas existentes, enquanto outros são específicos de um determinado tipo. Portanto, de seguida apresentam-se os dois conceitos mais comuns:

- Decomposição de tarefa - descrição hierárquica da tarefa, onde no nível mais elevado está a tarefa principal, que é sucessiva e recursivamente dividida em subtarefas menores, até aos mais baixos níveis, que descrevem ações físicas (não decomponíveis), que estão especialmente associadas com a interação sobre um dispositivo;
- Relações causais/temporais - descrevem o fluxo da tarefa, indicando a ordem em que as subtarefas são executadas, geralmente através de operadores característicos (estes serão abordados um pouco mais abaixo);

Para além destes, existem ainda outros elementos, os quais pretendem descrever os objetos (manipulados pelas tarefas), os papéis e atores (responsáveis pelas tarefas), as restrições (por

ex. pré/pós condições) e as propriedades (frequência de realização, prioridade, etc.), relativas às tarefas [41][58].

O facto de beneficiarem de notações flexíveis e expressivas com semânticas exatas, de métodos sistemáticos que ajudam a indicar como é que a informação deve ser aplicada, e ferramentas automáticas disponíveis para usar tal informação de forma eficiente [48], também tem servido de alavanca para a sua crescente adoção. A existência das ferramentas de suporte, que visam apoiar a edição dos modelos de tarefas e a análise das relações entre as tarefas e outras informações relacionadas, ajuda a superar o facto da modelação de tarefas ser, por vezes, um processo desencorajador e que consome muito tempo [58]. Como no caso específico das aplicações interativas, as possíveis tarefas a serem executadas dependem do estado atual da aplicação, torna-se bastante difícil compreender o comportamento dinâmico resultante das relações temporais especificados no modelo de tarefas. Por isso, a existência de simuladores interativos em algumas das ferramentas revela-se muito útil. Estes permitem que a qualquer momento o *designer* selecione, interativamente, uma tarefa e o simulador mostra as tarefas que serão ativadas após a realização da tarefa selecionada [58]. Outra grande vantagem destes modelos está na sua "linguagem", que é bastante simples e interativa, a qual pode ser facilmente compreendida por qualquer interveniente do projeto para o qual está a ser concebido. Pode-se mesmo afirmar que um modelo de tarefas deve ser complementado com os resultados da discussão entre os diferentes atores envolvidos no projeto em causa [46], considerando todos os pontos de vista (*designers*, desenvolvedores, utilizadores finais, cliente, etc.) [11]. A utilização destes modelos pode ter diferentes finalidades. Garante uma melhor compreensão do domínio da aplicação, obrigando os *designers* a esclarecer que tarefas devem ser suportadas, as suas relações mútuas e restrições/limitações; suporta o processo de *effective design*, desde que haja uma correspondência direta entre as tarefas e a interface de utilizador; suporta os testes de avaliação de usabilidade, e ainda é capaz de ajudar o utilizador durante uma sessão [48], por exemplo, sendo usados, em tempo de execução, para gerar automaticamente ajuda orientada à tarefa, explicando-o como a deve realizar [11]. Sendo assim, a sua utilização pode ser benéfica tanto para os *designers* como para os utilizadores finais. Aos primeiros, oferecem abordagens estruturadas e de alto nível, que permitem uma integração tanto dos aspetos funcionais como dos aspetos da interação. A vantagem oferecida aos utilizadores finais é que, dão um grande contributo para que cheguem até si sistemas mais compreensíveis [48].

Um exemplo concreto onde os modelos de tarefas revelam a sua importância é quando é requerido o suporte de novas tarefas ou a possibilidade de executar tarefas já existentes de diferentes formas. Aí, os modelos são capazes de indicar a parte da aplicação que tem de ser modificada e como esta se relaciona com as outras [46], impulsionando, desta forma, a geração de soluções alternativas de *design*, para suportar uma mesma tarefa interativa [47].

Atualmente, estes encontram-se entre os modelos que são mais amplamente utilizados no desenvolvimento de sistemas interativos, com mais enfoque na fase do desenho [50]. Importa referir que além da sua utilização se mostrar útil no suporte às fases de desenvolvimento, também serve de apoio às fases de teste [43].

2.3.1 - Notações relativas aos Modelos de Tarefas

Em suma, destacam-se agora algumas das principais vantagens oferecidas pela utilização de modelos de tarefas. Podem ser usados para avaliar a complexidade das tarefas a serem executadas pelo utilizador, para estimar o tempo de execução das tarefas, para simular o comportamento do sistema resultante, para construir, automaticamente, interfaces com base no modelo, para gerar ajuda/instruções orientadas para o utilizador e, por outra perspetiva, também facilita a comunicação entre as várias partes interessadas no desenvolvimento do sistema.

Posto isto, é sabido que existem diversas formas de representar um modelo de tarefas, mas neste trabalho apenas será dada atenção à representação hierárquica [50], que foi a considerada mais adequada à aplicação no contexto do trabalho. Mais precisamente, recorrendo às notações CTT (*Concurrent Task Trees*) [11] e HAMSTERS (*Human-centered Assessment and Modeling to Support Task Engineering for Resilient Systems*) [53], e respetivas ferramentas de suporte. Outras possibilidades de representação seriam através da família GOMS (*Goals*, *Operators*, *Methods*, *Selection Rules*), da UAN (*User Action Notation*), de diferentes sintaxes (textual *versus* gráfica) ou de diferentes níveis de formalidade [48].

2.3.1.1 - Notação CTT (Concurrent Task Trees)

Esta é, provavelmente, a linguagem/notação mais popular de modelação e análise de tarefas [50], e tem o seu foco nas atividades, que são os aspetos mais relevantes na conceção de um sistema interativo [45]. Baseia-se numa decomposição hierárquica das (sub)tarefas [59] que devem ser realizadas para atingir um determinado objetivo [43][51]. Isso deve-se, em parte, ao facto de ser uma notação fácil de utilizar e de suportar o desenho de aplicações, quer de média ou de grande dimensão [50]. A notação mostra-se bastante intuitiva e de fácil interpretação, pois possui uma sintaxe gráfica [11] e é representada através de uma estrutura hierárquica de tarefas, em árvore [45] (ver exemplo da Figura 4). Desta forma, permite a decomposição de tarefas em outras mais pequenas, a definição de tarefas concorrentes e a atribuição de relações temporais entre as tarefas [50]. Pode também ser utilizada para descrever as tarefas em diferentes níveis de abstração [46]. Aqui o objetivo principal que o utilizador pretende alcançar situa-se na raiz da árvore (topo da hierarquia) [43]. Tal como é apanágio nos modelos de tarefas, a notação CTT foca-se nas atividades, que são o aspeto mais importante no desenvolvimento de sistemas interativos [50]. Com esta notação são possíveis representar quatro diferentes tipos de tarefas, dependendo da alocação do seu desempenho [45][46]. São eles:

-  Tarefas de Interação - representam a interação (*inputs*) do utilizador com a aplicação;
-  Tarefas da Aplicação - tarefas realizadas exclusivamente pelo sistema (*outputs* transmitidos pela aplicação ao utilizador);
-  Tarefas do Utilizador - tarefas realizadas exclusivamente pelo utilizador, que representam pontos de decisão por parte deste;

- Tarefas Abstratas - não são tarefas em si, são uma representação de uma composição de tarefas que auxilia a decomposição [11], aparecendo como nós internos da árvore, enquanto as anteriores devem aparecer como folhas [43][51][50][61].

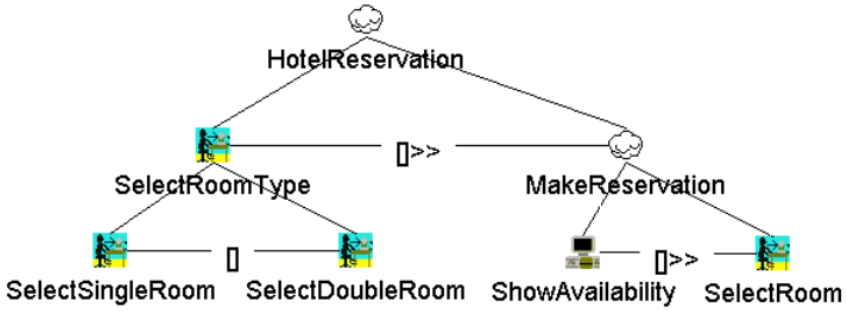


Figura 4. Exemplo de um modelo de tarefas na notação CTT, referente a uma reserva de hotel [61].

Relativamente à semântica do modelo, ela é definida pelos percursos que são possíveis realizar através da árvore [43]. A interpretação é realizada mediante a travessia da mesma, de cima para baixo e partindo da esquerda para a direita, sempre regida pelos operadores aplicados, quer a tarefas quer a pares de tarefas [43][51]. Um ponto a destacar é o facto de a notação possuir uma ferramenta de suporte, através da qual é possível desenvolver os próprios modelos ou modificar os já existentes [51]. A ferramenta é a *ConcurrentTaskTree Environment* (CTTE), a qual será abordada mais abaixo. Como já foi mencionado, a notação CTT também permite definir relações temporais entre as tarefas [11], recorrendo a um forte conjunto de operadores (unários e binários) representativos [46][50]. Estes devem ser aplicados a tarefas que se situem num mesmo nível hierárquico [51]. Na Tabela 3 estão representados os principais operadores existentes. Em seguida será apresentada uma descrição mais detalhada de cada um dos operadores.

Tabela 3. Principais operadores na notação CTT.

	Operador	**Significado**
Binários	>>	*Enabling*
	[]>>	*Enabling with Information Passing*
	[]	*Choice*
	|=|	*Order Independence*
	|||	*(Independent) Concurrent*
	|[ ]|	*Concurrent with Information Exchange*
	[>	*Disabling*
	|>	*Suspend/Resume*
Unários	T*	*Iterative*
	[T]	*Optional task*

Apresentam-se agora em detalhe os operadores já acima mencionados [11][51].

- T_1 >> T_2 : a tarefa T_2 só pode ser realizada assim que terminar a execução da tarefa T_1.
- T_1 []>> T_2 : a tarefa T_2 é ativada quando a tarefa T_1 terminar a sua execução, visto que, esta última produz informação necessária à execução da primeira.

- T_1 [] T_2 : indica que as duas tarefas, T_1 e T_2, estão ativas, e permite escolher qual delas deve ser executada. Após ter feito a escolha, a outra tarefa fica indisponível.
- T_1 |=| T_2 : ambas as tarefas, T_1 e T_2, têm de ser executadas, independentemente da ordem (quando uma começar tem de ser terminada para poder iniciar a outra);
- T_1 ||| T_2 : as tarefas podem ser realizadas por qualquer ordem, ou ao mesmo tempo, sendo possível iniciar uma tarefa sem que a outra já esteja terminada.
- T_1 |[ ]| T_2 : as duas tarefas podem ser executadas concorrentemente, mas precisam de estar sincronizadas de forma a poderem trocar informação;
- T_1 [> T_2 : ambas as tarefas estão ativas ao mesmo tempo, mas quando se dá início à tarefa T_2, a tarefa T_1 é desativada;
- T_1 |> T_2 : a tarefa T_2 pode interromper a execução da tarefa T_1, no entanto, quando T_2 terminar regressa-se ao ponto da execução de T_1 onde houve a interrupção;
- T* : significa que a tarefa T é iterativa, i.e. no final da sua execução esta pode voltar a ser repetida as vezes que forem pretendidas;
- [T] : indica que a tarefa T é opcional, não tem de ser obrigatoriamente realizada.

Cada tarefa do modelo pode ter, a si associados, objetos, os quais têm de ser manipulados de forma a permitir que a mesma seja executada [11]. São lhes reconhecidos dois grandes tipos, os objetos da interface de utilizador e os do domínio da aplicação, que por vezes encontram-se associados [11] (por exemplo, a temperatura pode ser apresentada na interface através de um gráfico de barras). Além disso, as tarefas devem ter especificados os seus próprios atributos, dos quais se destacam o identificador da tarefa, o tipo e categoria a que pertence, se é ou não iterativa, se possui alguma pré-condição que deve ser satisfeita e o tempo de desempenho previsto [59].

Agora relativamente à ferramenta *CTTE*, é de interesse revelar que tem sido utilizada numa série de cursos universitários [59], tanto para fins de ensino como para projetos de investigação, em diversas áreas de aplicação, tais como, controlo de tráfego aéreo e planeamento de recursos empresariais [58]. Nesta ferramenta destacam-se as seguintes funcionalidades [62]:

- Permite inserir as informações requeridas pela notação - tarefas e seus atributos (nome, categoria, tipo, objetos relacionados, etc.) -, as relações temporais entre tarefas e outras informações;
- Ajuda a melhorar o *layout* da especificação através de: suportar o alinhamento automático e o movimento de níveis ou sub-árvores do modelo de tarefas, cortar e colar partes do modelo, mostrar e esconder sub-árvores, fornecer estatísticas relativas ao modelo (número de tarefas por categoria, número de relações temporais por operador, o número de níveis, etc.);
- Permite verificar a exatidão da especificação (por exemplo, é possível detetar se estão a faltar operadores temporais entre tarefas no mesmo nível).
- Inclui um simulador da execução de tarefas (*Task Simulator*).
- Suporta a inserção de outros modelos de tarefas no que está a ser criado.
- Guardar o modelo ou parte dele na forma de imagem.

Ainda nesse ambiente, aquando da especificação de uma tarefa existem ícones exclusivos que identificam o tipo de tarefa [46]. Além do CTTE permitir a criação direta de modelos de tarefas,

ainda permite ao utilizador carregar uma descrição informal (textual) de um cenário ou caso de utilização e a partir daí selecionar a informação relevante para a modelação [11] (por exemplo, indicações de que tarefas devem ser suportadas ou os objetos que elas têm de manipular [62]). Esse processo é explicado em detalhe por Paternò, em [11] e [67].

Como nesta área já foram desenvolvidos vários projetos de investigação, principalmente no âmbito da criação de abordagens baseadas na utilização de modelos de tarefas, para o desenvolvimento de aplicações interativas, destacam-se de seguida dois deles.

O primeiro é a utilização da ferramenta CTTE como suporte ao desenho de uma aplicação web adaptável para um museu, aplicação essa que dispõe de uma interface e forma de navegação diferentes, consoante o tipo de utilizador (turista, *expert* ou estudante). Para isso, foi elaborada uma lista de tarefas a incorporar, a partir da qual, foi desenvolvido, no CTTE, um modelo de tarefas diferente para cada tipo de utilizador [45].

Outro caso real é o projeto MEFISTO (*Modelling, Evaluating and Formalising Interactive Systems using Tasks and interaction Objects*), onde diferentes equipas utilizaram o CTTE para modelar várias aplicações de controlo de tráfego aéreo, e dar suporte aos processos de desenho e avaliação das mesmas. Aqui, encontraram alguma dificuldade na compreensão exata da semântica relacionada com os operadores temporais (usados nos modelos de tarefas). Os *designers* notaram que seria útil ter alguma bagagem para seguir a abordagem proposta, mas a estrutura hierárquica e a sintaxe gráfica da notação facilitou a compreensão dos modelos [45].

2.3.1.2 - Notação HAMSTERS

A notação HAMSTERS é uma linguagem de modelação de tarefas que foi desenvolvida tendo por base as notações que já existiam [53], especialmente a notação CTT, não sendo por acaso o facto de ser em tudo muito similar a esta. Acrescenta ainda o ter sido criada de forma a ser compatível, do ponto de vista de quem constrói os modelos, com a notação CTT [52]. É baseada também numa sintaxe gráfica e representada através de uma estrutura hierárquica em árvore, apesar desta revelar algumas discrepâncias relativamente à arvore de um modelo de tarefas em CTT. Tal como acontece com a notação anterior, aqui também existem quatro tipos principais de tarefas [4]:

-  Abstratas - que envolvem subtarefas de qualquer tipo;
-  De sistema - realizadas exclusivamente pelo sistema;
-  De utilizador - que se referem a atividades do utilizador (as quais podem ser motoras, cognitivas ou percetivas);
-  Interativas - que dizem respeito a interações entre o utilizador e o sistema (estas dividem-se em input, output e input-output).

No entanto, é de realçar que a notação HAMSTERS apresenta alguns tipos de tarefas mais específicos [52], nomeadamente no que se refere às tarefas de utilizador e às interativas. Há também a possibilidade das tarefas serem opcionais, iterativas ou ambas [4]. É por meio de operadores que são representadas as relações temporais entre as tarefas [4], tal como acontece na notação CTT, só com a diferença na forma de como são apresentados na árvore, pois, como

é possível observar na Figura 5, na notação HAMSTERS eles encontram-se ligados ao nó "pai" [53].

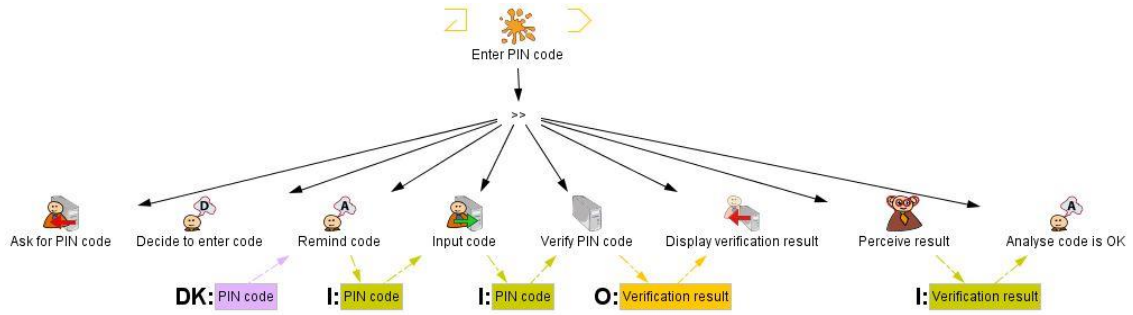


Figura 5. Exemplo de um modelo de tarefas segundo a notação HAMSTERS [53].

Para além de suportar condições associadas à execução das tarefas, esta notação considera também o conceito de objeto, que se refere aos elementos do mundo (informação ou conhecimento) que podem ser manipulados pelas tarefas. Apesar de aqui serem considerados menos operadores, estes são equivalentes aos apresentados em cima, relativamente à notação CTT. Assim sendo, de seguida encontram-se detalhados os principais tipos de operadores da notação HAMSTERS [4][54]:

- *Enable* (>>) - a tarefa à direita só é executada após a tarefa à esquerda estar concluída.
- *Concurrent* (|||) - as tarefas podem ser realizadas simultaneamente.
- *Choice* ([]) - permite ao utilizador optar pela tarefa que pretende.
- *Disable* ([>) - quando a tarefa à direita é executada, interrompe e desativa a execução da tarefa à esquerda.
- *Suspend/Resume* (|>) - quando a tarefa à direita é executada, suspende a execução da tarefa à esquerda
- *Order Independent* (|=|) - as tarefas podem ser executadas, uma depois da outra, em qualquer ordem.

Para além das relações temporais entre as tarefas, a HAMSTERS permite ainda definir outro tipo de relações, que são o fluxo de informações entre elas [52]. Na Figura 6, onde o PIN, que é introduzido na tarefa "Enter PIN number", é transferido para a tarefa seguinte, pode-se ver como isso é representado no modelo. Por outro lado, também podem ser descritos alguns aspetos quantitativos relativos às tarefas [54]. Destacam-se aqui a indicação da duração da tarefa (tal como na CTT), podendo atribuir um tempo mínimo e máximo estimados para a sua execução, e do *delay* verificado até à mesma estar disponível [53].

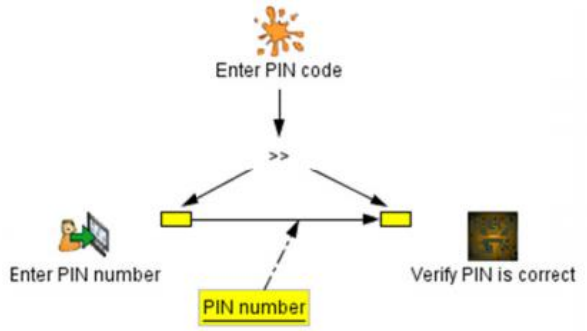


Figura 6. Exemplo da representação do fluxo de informação entre duas tarefas (HAMSTERS) [52].

A implementação desta notação foi pensada com o intuito de a tornar facilmente extensível, possuindo uma ferramenta que facilita o processo de conceção de modelos de tarefas. É esta ferramenta de suporte que principalmente a diferencia da notação CTT, em comparação com a ferramenta oferecida por essa. A ferramenta HAMSTERS oferece uma API dedicada à simulação de modelos de tarefas [53], que permite a observação, edição e simulação da execução de sequências de tarefas [4]. Pode-se afirmar que esta é uma notação que, juntamente com a ferramenta que oferece, não está totalmente definida [52], pois os autores estão abertos a sugestões para possíveis expansões, de modo a satisfazer novas necessidades por parte dos utilizadores.

2.3.2 - Comparação e Seleção

Por estes modelos serem bastante úteis ao desenvolvimento de sistemas interativos, devido ao suporte oferecido, quer ao processo de *design* quer ao de avaliação dos mesmos, é cada vez maior o interesse em ferramentas que tornem mais fácil a sua conceção e permitam aos *designers* construírem-nos da forma mais explícita possível [52]. Só desta forma é que poderão ser retirados todos os benefícios da sua (re)utilização. Essas ferramentas são capazes de produzir representações dos vários aspetos do modelo de tarefas que podem depois ser manipuladas por outros sistemas (por exemplo, através de representações baseadas em XML) [57]. Principalmente quando se trata de modelos de tarefas grandes e complexos, a sua gestão será mais fácil quanto mais recursos a ferramenta de suporte for capaz de oferecer [52].

Como já foi referido anteriormente, os modelos de tarefas servirão de base para alcançar o objetivo deste trabalho, definição de uma metodologia de suporte à utilização de sistemas interativos. Daí ter surgido a necessidade de optar por uma notação para ser utilizada no decorrer do projeto. E o primeiro requisito procurado foi a existência de uma ferramenta que desse suporte à notação a utilizar. Desde então centrei-me principalmente nas duas notações abordadas anteriormente: a HAMSTERS e a CTT, isto porque além de oferecerem ferramentas de apoio à criação e edição de modelos de tarefas, ambas fazem uso de representações hierárquicas dos modelos [43] e estão entre as notações mais reconhecidas e consequentemente mais utilizadas pela comunidade de IHC [41], havendo mais material disponível para quem se está a introduzir na área.

Aí foi uma questão de perceber os prós e contras de cada notação e respetiva ferramenta, os quais se apresentam resumidos imediatamente abaixo. Relativamente à notação/ferramenta HAMSTERS destaco os seguintes pontos [4][53][55]:

- É *open source*;
- É uma notação mais precisa e mais forte do que a CTT;
- Lida tanto com casos de estudo pequenos (laboratório) como reais (empresas);
- Requer muito pouco tempo de aprendizagem;
- É inspirada em notações existentes, aproveitando as vantagens de todas elas;
- Os ícones são facilmente percetíveis;
- Os operadores temporais são anexados ao nó pai;
- É possível adicionar condições associadas à execução de tarefas;
- Permite representar o fluxo de informações entre as tarefas;
- A ferramenta permite simular a execução de cenários.

De seguida, no que à notação CTT (e ferramenta CTTE) diz respeito, apresentam-se as principais características [43][11][50][56]:

- É a abordagem mais comum para a análise/modelação de tarefas;
- É amplamente utilizada, quer a nível universitário quer industrial;
- É flexível e expressiva;
- Oferece uma representação compacta e de fácil compreensão;
- É possível especificar vários atributos das tarefas, nomeadamente, nome, plataforma, objetos e o tempo de execução estimado;
- Permite construir um modelo a partir da seleção de informação relevante da descrição (textual) informal de um *use case*/cenário [67];
- A ferramenta suporta tarefas simultâneas e cooperativas (tarefas que devem ser executadas por dois ou mais utilizadores [58]), cálculo de métricas, avaliação do desempenho de tarefas e simulação interativa;
- Suporta a decomposição de tarefas em vários diagramas que comunicam, através de tarefas colaborativas [52];
- Os testes seguem a utilização prevista do sistema.

Tendo em conta toda a informação recolhida e os pontos realçados anteriormente, aliados a uma pequena fase de experiência com as duas ferramentas, foi necessário tomar uma decisão. Esta recaiu sobre a notação *ConcurTaskTrees*. Nesta destaca-se a sua capacidade de expressão na modelação de sistemas interativos [41]. Depois, para além da principal ferramenta de suporte à notação, a CTTE, existe também uma outra complementar, a MARIAE (Model-based lAnguage foR Interactive Applications Environment [60]). Esta, apesar de estar mais focada na descrição de interfaces de utilizador (a nível abstrato e concreto) e na geração destas a partir de modelos de tarefas, suporta também a análise de modelos CTT [60], e uma série de funcionalidades relacionadas com a animação de modelos de tarefas, que serão úteis numa fase mais adiantada deste trabalho. Entre essas destacam-se a existência de um simulador de tarefas e a possibilidade de extração das simulações sob o formato XML. Também é de realçar o facto de terem sido detetadas falhas de coerência no funcionamento do simulador da ferramenta HAMSTERS. Assim sendo, todos estes fatores contribuíram para a tomada de decisão.

3 - Especificação da Abordagem

Como já foi referido anteriormente, os modelos de tarefas servirão de base para alcançar o objetivo deste trabalho – a definição de uma abordagem de suporte à utilização de sistemas interativos. Isto porque eles representam as tarefas que o utilizador deve realizar para atingir o seu objetivo, descrevendo a lógica da camada interativa das aplicações, ou seja, permitem identificar as interações entre o utilizador e o dispositivo e descrever como essas devem ser realizadas.

Assim, devido à existência de várias notações a este nível, que oferecem diferentes formas de representar os modelos de tarefas, foi necessário optar por aquela que melhor pudesse contribuir para o objetivo do trabalho. Tudo isso está explicitado acima, no ponto 2.3.2 - Comparação e Seleção. A partir da tomada de decisão o foco passou a estar apenas na notação CTT (e respetiva ferramenta de suporte, CTTE). Um dos fatores apresentados para a escolha desta notação foi o facto dos modelos de tarefas serem representados numa estrutura hierárquica. Isso facilita o processo de conceção do modelo e, acima de tudo, torna mais intuitiva a sua análise e compreensão.

Prosseguindo para a apresentação da abordagem seguida, esta pode ser dividida em quatro etapas fundamentais, as quais consistem em:

1. Extração da informação relevante do Modelo de Tarefas (nome das tarefas, ficheiros de imagem associados, etc.);
2. Extração, a partir do Cenário de Simulação, da indicação da ordem de execução das tarefas;
3. Geração da Script de Automatização a partir das informações obtidas em 1 e 2; e
4. Criação da Interface (de Simplificação/de Ajuda), através da qual o utilizador dá a ordem de execução das *scripts*.

A principal questão para garantir uma adaptação dos sistemas interativos aos utilizadores passa pela criação automática de *scripts* de automatização, a partir de modelos de tarefas e de cenários. Nesta secção é apresentada uma descrição detalhada de cada etapa da abordagem. Primeiramente apresenta-se o processo de enriquecimento dos modelos de tarefas, bem como as regras a seguir para este ser corretamente aplicado. Em segundo lugar é dado ênfase ao modo de obtenção dos cenários que complementam o modelo anterior. Segue-se a explicação do processo de transformação dos modelos de tarefas e cenários em *scripts* de automatização. Por último encontram-se descritos os passos necessários para a criação de interfaces simplificadas que permitem ao utilizador final interagir com os sistemas interativos, mediante a ação das *scripts*.

3.1 - Modelos de Tarefas Enriquecidos

O primeiro passo, que servirá de base para a aplicação da abordagem, inicia-se com a obtenção dos modelos de tarefas referentes às atividades do sistema considerado, que o desenvolvedor pretende que sejam automatizadas. Para tal, podem ser seguidos dois diferentes caminhos, dependendo do estado em que se encontra a aplicação alvo e claro, do que tiver sido disponibilizado pelos seus desenvolvedores. O primeiro aplica-se nos casos em que não é dado acesso aos modelos de tarefas, e consiste na construção, a partir do zero, dos respetivos modelos. Por outro lado, para aplicações em fase de desenvolvimento, onde os modelos de tarefas são facultados, a necessidade passa apenas por enriquecê-los, aplicando-lhes as modificações adequadas.

Esta necessidade de enriquecer os modelos tornava-se óbvia pois os modelos de tarefas "originais" não eram possuidores de toda a informação necessária para a transformação ser bem-sucedida. Aqui destaca-se claramente a ausência dos *screenshots* requeridos pelas *scripts* do *Sikuli*. Para esse efeito, foi desenvolvida uma notação que permite introduzir numa das propriedades da tarefa – *Description* – a localização desses *screenshots*, seguindo o exemplo da Figura 7. Nem só informação acerca dos *screenshots* pode ser adicionada, definiu-se também que seriam úteis outros dados para complementar algumas funções do *Sikuli* a utilizar nas *scripts* obtidas. Esses dados estão explicados na seção 3.1.1 - Regras de Enriquecimento dos Modelos de Tarefas - podendo ser, por exemplo, a indicação do tempo de espera (função *wait*) ou das mensagens nas caixas de diálogo (por exemplo, na função *popup*). A descrição foi o campo escolhido por estas serem informações adicionais e externas ao modelo em si, pois não têm qualquer influência sobre a interpretação do mesmo nem sobre a execução dos cenários de simulação. Para além do campo *Description*, a notação também requer que os nomes das tarefas (campo *Name*, ver Figura 7) sejam modificados, de acordo com as regras a seguir apresentadas, para corresponderem corretamente às respetivas funções do *Sikuli*. Exemplificando, para executar a função *click* (*Sikuli*) o nome da tarefa deve incluir no início a palavra reservada *Press*, seguida de um nome indicativo da tarefa.

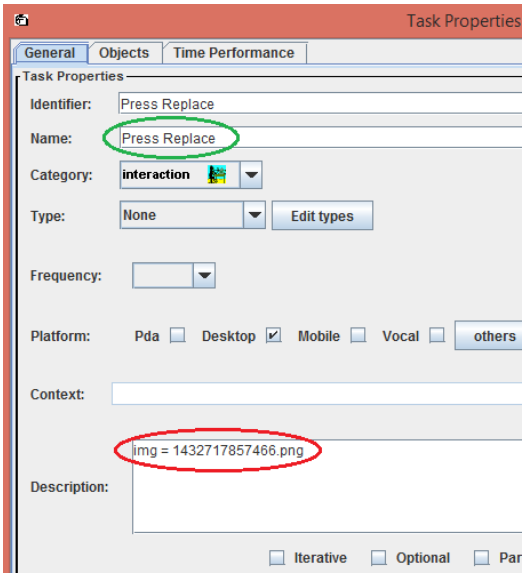


Figura 7. Principais propriedades (*Name* e *Description*) de uma tarefa CTT.

3.1.1 - Regras de Enriquecimento dos Modelos de Tarefas

De maneira a haver uma correspondência entre os modelos de tarefas e as *scripts* obtidas, foi necessário definir um conjunto de regras para cada caso específico, as quais devem ser estritamente respeitadas para o correto funcionamento da abordagem aqui apresentada. Complementando o que foi descrito anteriormente, mostra-se agora uma descrição detalhada da lista completa de regras para a atribuição de nomes (campo *Name*) às tarefas e para as informações a adicionar no campo *Description* das mesmas, durante o enriquecimento dos modelos de tarefas. Apresentam-se igualmente as funções do *Sikuli* correspondentes, e ainda uma explicação de como/quando estas devem ser usadas, sustentando com alguns exemplos da sua aplicação.

Funções de clique:

- **Press** <tarefa> -> ***click****(img)* - utilizar para clicar (botão esquerdo) sobre um dado elemento (img)
    - Introduzir no campo "Descrição" o nome da imagem (img) no formato "img = nome.png", em que nome é a única variável
- **PressR** <tarefa> -> ***rightClick****(img)* - utilizar para clicar com o botão direito sobre um elemento (img)
    - Introduzir no campo "Descrição" o nome da imagem: "img = nome.png"
- **PressD** <tarefa> -> ***doubleClick****(img)* - utilizar para fazer duplo clique sobre um elemento (img)
    - Introduzir no campo "Descrição" o nome da imagem: "img = nome.png"

Funções de introdução/edição de dados:

- **Enter** <tarefa> -> *text =* ***input*** *();* ***paste****(img, text)* - utilizar para pedir ao utilizador a introdução manual de dados (text) a inserir num determinado campo (img)
    - Introduzir no campo "Descrição" o nome da imagem: "img = nome.png"
- **EnterPassw** <tarefa> -> *passw =* ***input*** *("Digite a sua password",hidden=True);* ***paste****(img, passw)* - utilizar para pedir ao utilizador a introdução manual de uma password (passw) a inserir num determinado campo (img)
    - Introduzir no campo "Descrição" o nome da imagem: "img = nome.png"
- **EnterSemiAuto** <tarefa> -> ***paste(input****(text))* - utilizar para pedir ao utilizador a introdução manual de dados, indicando o que é pretendido (text)
    - Introduzir no campo "Descrição" o texto indicativo da caixa de diálogo (text), por ex. "Digite o seu nome"
- **EnterAuto** <tarefa> -> ***paste****(text + Key.ENTER)* - utilizar para o computador introduzir automaticamente dados predefinidos (text)
    - Introduzir no campo "Descrição" o texto a ser introduzido (text), por ex. "www.google.pt"
- **EnterKey** <tarefa> -> *type(key)* – utilizar para premir automaticamente uma tecla especial (key) (ou para introdução normal de texto)
    - Introduzir no campo "Descrição" a tecla a introduzir (*key*), por ex. "Key.F11"
- **EnterCopy** <tarefa> -> ***type****("c", KeyModifier.CTRL)* - utilizar quando for pretendido copiar alguma informação já selecionada

- **TextPaste** <tarefa> -> ***paste****(img,* ***Env.getClipboard()*** *)* - utilizar para colar o texto já copiado num determinado campo (img)
    - Introduzir no campo "Descrição" o nome da imagem: "img = nome.png"

De seguida (Figura 8 e Figura 9) é visível um exemplo da aplicação de uma regra de introdução de dados.

```
<Task Identifier="EnterPassw YourKeyWord" Category="inte
    <Name>EnterPassw YourKeyWord</Name>
    <Description>img = 1434574443948.png</Description>
    <Parent name="Paste Option"/>
    <SiblingLeft name="WaitT final"/>
</Task>
```

Figura 8. Excerto 1 do ficheiro (XML) de um Modelo de Tarefas.

Figura 9. Script do *Sikuli* (ficheiro *Python*) correspondente ao excerto 1 da Figura 8.

Funções de espera:

- **WaitAppear** <tarefa> -> ***wait****(img,* ***10****)* - utilizar quando pretender esperar (10s) até que algo seja visível no ecrã (img)
    - Introduzir no campo "Descrição" o nome da imagem: "img = nome.png"
- **WaitT** <tarefa> -> ***wait****(time)* - utilizar para colocar o sistema em espera durante um determinado tempo (time)
    - Introduzir no campo "Descrição" o tempo de espera (time), por ex. "10"
- **WaitDisappear** <tarefa> -> ***waitVanish****(img)* - utilizar quando pretender esperar até que algo (img) desapareça do ecrã
    - Introduzir no campo "Descrição" o nome da imagem: "img = nome.png"

Funções de janelas modais:

- **Popup** <tarefa> -> ***popup****(msg)* - utilizar para mostrar uma janela modal (de aviso ou erro), podendo especificar o conteúdo da mesma (msg)
    - Introduzir no campo "Descrição" a informação a mostrar na janela modal (msg), por ex. "Operação Inválida"
- **PopAsk** <tarefa> -> ***popAsk****(msg)* - utilizar para mostrar uma janela modal com uma questão (msg) de resposta "Sim/Não"
    - Introduzir no campo "Descrição" a questão a mostrar na janela modal (msg)

Aqui (Figura 10 e Figura 11) são exemplificadas três diferentes tipos de regras, uma de janelas modais, outra de espera e ainda uma de introdução/edição de dados.

```
<Task Identifier="Paste Option" Category="abstraction" Iterative="false" Optio
    <Name>Paste Option</Name>
    <SubTask>
        <Task Identifier="Popup Info" Category="application" Iterative="false"
            <Name>Popup Info</Name>
            <Description>Selecione e copie o texto que pretende</Description>
            <TemporalOperator name="SequentialEnabling"/>
            <Parent name="Paste Option"/>
            <SiblingRight name="WaitT CopyText"/>
        </Task>
        <Task Identifier="WaitT CopyText" Category="application" Iterative="fa
            <Name>WaitT CopyText</Name>
            <Description>5</Description>
            <TemporalOperator name="SequentialEnabling"/>
            <Parent name="Paste Option"/>
            <SiblingLeft name="Popup Info"/>
            <SiblingRight name="TextPaste clipboard"/>
        </Task>
        <Task Identifier="TextPaste clipboard" Category="interaction" Iterative
            <Name>TextPaste clipboard</Name>
            <Type>None</Type>
            <Description>img = 1434478888045.png</Description>
            <TemporalOperator name="SequentialEnabling"/>
            <Parent name="Paste Option"/>
            <SiblingLeft name="WaitT CopyText"/>
            <SiblingRight name="WaitT again"/>
```

Figura 10. Excerto 2 de um Modelo de Tarefas.

```
1 popup("Selecione e copie o texto que pretende")
2 wait(5)
3 paste(                                          ,Env.getClipboard())
```

Figura 11. Script (Sikuli) correspondente ao excerto 2.

Ciclo While:

- **FindW** <tarefa> -> ***while not exists****(img):* - utilizar para percorrer o ecrã (verticalmente) até encontrar no ecrã um dado elemento (img)
    - Introduzir no campo "Descrição" o nome da imagem: "img = nome.png"

Expressão condicional – If/Else:

- **YesNo_IF** <tarefa> -> ***if*** *(text)****:*** - utilizar para verificar se a resposta dada pelo utilizador (através da função **popAsk**) foi positiva/negativa
    - Introduzir no campo "Descrição" o nome da variável booleana (text)
- **ShowM** <tarefa> -> ***if exists****(img):* - utilizar para verificar se um dado elemento (img) é visível no ecrã
    - Introduzir no campo "Descrição" o nome da imagem: "img = nome.png"
- **_IF** <tarefa> - utilizar para indicar uma tarefa a ser executada se a condição se verificar
- **_LastIF** <tarefa> - utilizar para indicar a última tarefa a ser executada dentro da condição

- **_firstELSE** <tarefa> -> ***else:*** - utilizar para indicar a primeira tarefa a realizar caso a condição não se verifique

- **_ELSE** <tarefa> - utilizar para indicar uma tarefa a ser executada caso a condição não se verifique
- **_LastELSE** <tarefa> - utilizar para indicar a última tarefa a realizar caso a condição não se verifique

Nota: Nestes últimos 5 casos (começados por "_"), antes do "_" terá de indicar o nome da função pretendida, por exemplo: "**Press_IF**", "**Wait_firstELSE**", "**Enter_ELSE**", etc.

Em baixo (Figura 12 e Figura 13) são mostradas diferentes regras relativas à expressão condicional *if-else*.

```
    <Task Identifier="ShowM SelectionWindow" Category="appli
        <Name>ShowM SelectionWindow</Name>
        <Description>img = 1423061288657.png</Description>
        <Platform>Desktop</Platform>
        <TemporalOperator name="Disabling"/>
        <Parent name="SelectionWindow"/>
        <SiblingLeft name="Wait SelectionWindow"/>
        <SiblingRight name="Popup"/>
    </Task>
    <Task Identifier="Popup" Category="application" Iterativ
        <Name>Popup_IF</Name>
        <Description> </Description>
        <Platform>Desktop</Platform>
        <TemporalOperator name="Disabling"/>
        <Parent name="SelectionWindow"/>
        <SiblingLeft name="ShowM SelectionWindow"/>
        <SiblingRight name="Wait WindowHide"/>
    </Task>
    <Task Identifier="Wait WindowHide" Category="application
        <Name>Wait_LastIF WindowHide</Name>
        <Description>3</Description>
        <Parent name="SelectionWindow"/>
        <SiblingLeft name="Popup"/>
    </Task>
</SubTask>
                    </Task>
                    <Task Identifier="Press Preview"
<Name>Press_firstELSE Preview</Name>
<Description>img = 1423061333332.png</Description>
<Platform>Desktop</Platform>
```

Figura 12. Excerto 3 de um Modelo de Tarefas.

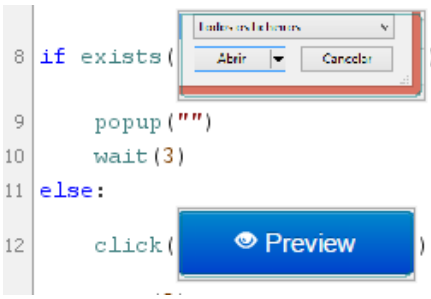


Figura 13. Script (Sikuli) correspondente ao excerto 3.

Funções de abrir/fechar aplicação:

- **OpenApp** <tarefa> -> ***App.open(*** *path* ***)*** – abrir aplicação, indicando a sua localização (path)
    - Introduzir no campo "Descrição" a localização da aplicação, por ex. "C:\Windows\system32\notepad.exe"
- **CloseApp** <tarefa> -> ***App.close(*** *path* ***)*** - fechar aplicação, indicando a sua localização (path)

- Introduzir no campo “Descrição” a localização da aplicação, por ex. “C:\Windows\system32\notepad.exe”

- **FocusApp** <tarefa> -> ***App.focus(*** *title* ***)*** - focar uma aplicação, indicando uma expressão presente no título da janela da mesma (*title*)
  - Introduzir no campo “Descrição” uma parte do título da janela da aplicação, por ex. “Word”

Exemplifica-se, nas Figura 14 e Figura 15, o uso da regra relativa à inicialização de uma aplicação.

```
<Task Identifier="OpenApp Browser" Category="application" Iterative="false" Optional="false"
    <Name>OpenApp Browser</Name>
    <Description>C:\Program Files (x86)\Google\Chrome\Application\chrome.exe</Description>
    <Platform>Desktop</Platform>
```

Figura 14. Excerto 4 de um Modelo de Tarefas.

```
1 App.open("C:\Program Files (x86)\Google\Chrome\Application\chrome.exe")
```

Figura 15. Script (Sikuli) correspondente ao excerto 4.

Estando as regras clarificadas, torna-se exequível desenvolver um Modelo de Tarefas Enriquecido. Só após a conclusão dessa etapa é que deve ser realizado o cenário de simulação pretendido, caso contrário poderão surgir futuras incongruências na geração da *script,* nomeadamente na comparação entre os nomes das tarefas dos dois “modelos” (modelo de tarefas e cenário), o que irá desvirtuar o resultado final.

3.1.2 - Cenários de Simulação

Os modelos de tarefas, por vezes, dão a possibilidade de se seguir diferentes percursos até completar a execução de uma tarefa. Isto quer dizer que para atingir o mesmo objetivo, o número de passos ou mesmo a sequência seguida não tem de ser obrigatoriamente a mesma. Isto provocou a necessidade de se obter, de alguma forma, a identificação da sequência de tarefas a realizar para chegar ao objetivo definido, pois só assim seria possível simplificar a automatização do processo. Seguindo esse raciocínio, foram identificadas duas alternativas capazes de solucionar esse problema.

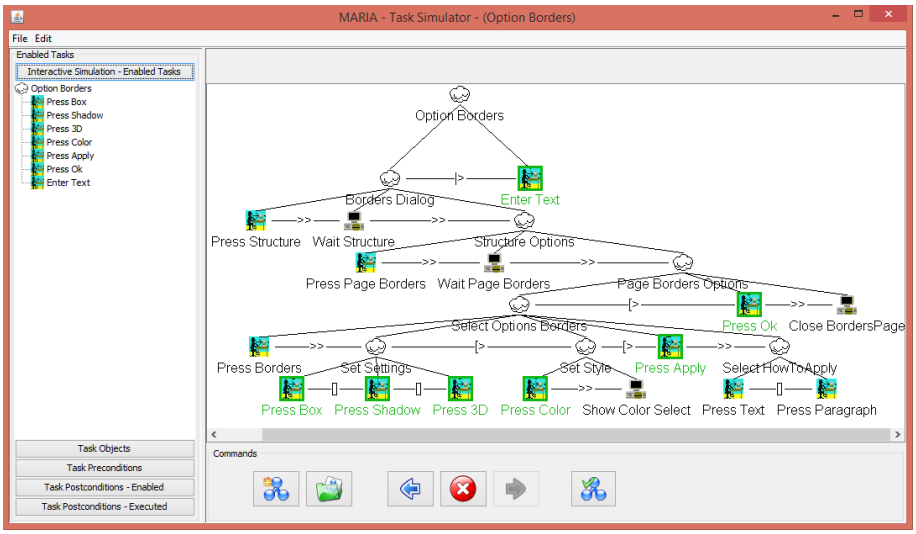


Figura 16. Interface do Simulador de Tarefas da ferramenta *MARIAE*.

A primeira baseia-se numa funcionalidade – simulador de tarefas (ver Figura 16) - que tanto é disponibilizada pelo *CTTE* como pelo *MARIAE*. Esta permite a simulação da execução de tarefas, como se estivesse a interagir com a aplicação, possibilitando guardar o cenário realizado. Aqui os dois ambientes revelam uma diferença, pois enquanto o *MARIAE* permite guardar o cenário num ficheiro XML (e.g. Figura 17) o *CTTE* apenas guarda no formato SCN. Daí que, para este efeito, tenha sido utilizada a ferramenta *MARIAE*.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<scenarioType xmlns="http://giove.isti.cnr.it/ctt/scenario" modelId="Skype makeCall">
    <step executed_task="Start Skype">
        <completed_task name="Start Skype" level="1"/>
    </step>
    <step executed_task="Wait SkypeWindow">
        <completed_task name="Wait SkypeWindow" level="1"/>
    </step>
    <step executed_task="Wait Logged in">
        <completed_task name="Wait Logged in" level="1"/>
    </step>
    <step executed_task="Press Find">
        <completed_task name="Press Find" level="2"/>
    </step>
    <step executed_task="Wait Loading">
        <completed_task name="Wait Loading" level="2"/>
    </step>
    <step executed_task="Enter ContactName">
        <completed_task name="Enter ContactName" level="2"/>
    </step>
    <step executed_task="Press Contact">
        <completed_task name="Option Find" level="1"/>
        <completed_task name="Press Contact" level="2"/>
    </step>
```

Figura 17. Excerto de um cenário relativo à tarefa de efetuar chamada pelo *Skype*.

A segunda opção recorre a uma funcionalidade disponibilizada apenas pelo *MARIAE*, que é a geração de um "Presentation Task Set (PTS) List" a partir de um modelo de tarefas, armazenando-o também num ficheiro XML. Apresenta-se um exemplo na figura seguinte (Figura 18).

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<presentationTaskSetList cttFilePath="/C:/Users/Diogo/Documents/skypeCall_files/ctt/TM_skypeCall_orig.xml">
    <presentationTaskSet id="1" idSequence="33">
        <task id="Start Skype"/>
        <task id="Wait SkypeWindow"/>
        <task id="Wait Logged in"/>
        <connection targetPtsId="4" expressionPath="/root/right/right"/>
    </presentationTaskSet>
    <presentationTaskSet id="4" idSequence="33">
        <task id="Press Find"/>
        <task id="Wait Loading"/>
        <connection targetPtsId="6" triggeredBy="Press Find" expressionPath="/root/right/right/right/left/right"/>
    </presentationTaskSet>
    <presentationTaskSet id="6" idSequence="33">
        <task id="Enter ContactName"/>
        <task id="Press Contact"/>
        <connection targetPtsId="7" expressionPath="/root/right/right/right"/>
    </presentationTaskSet>
```

Figura 18. Excerto de um ficheiro de *PTSs* (tarefa - efetuar chamada pelo *Skype*).

O objetivo nesta altura passava por perceber qual dos dois métodos seria mais adequado para obter a informação necessária à criação da *script* de automatização (nomeadamente a sequência ordenada de tarefas a executar para atingir o objetivo predefinido).

Em termos conceptuais, a lista de *PTSs* gerada no segundo ficheiro, pode ser vista como uma máquina de estados finita (FSM) do sistema. Nele são apresentados, separadamente, os vários conjuntos de tarefas (isto é, os *PTSs*, análogos aos estados da *FSM*), identificados por um id único, e indicando também todas as transições possíveis a partir de cada *PTS*, ou seja, para que *PTSs* é possível transitar a partir do atual. Posto isto, ficava a faltar um detalhe que era considerado imprescindível, não havia nenhuma indicação de quais tarefas é que despoletavam essas transições. Após alguma troca de *emails* com os responsáveis pela ferramenta em utilização, foi lançada uma versão atualizada da *MARIAE*, onde a geração deste ficheiro passou a incluir a indicação da tarefa que tinha de ser executada para despoletar a respetiva transição.

No entanto, um dos problemas detetados neste tipo de ficheiro foi a inconsistência dos estados gerados, pois ao representar manualmente uma máquina de estados, esses estados não tinham total correspondência com os *PTSs* gerados pela ferramenta, ao contrário do que seria de esperar. Haviam tarefas que deviam pertencer a um determinado *PTS* (estado) e no entanto, no ficheiro gerado encontravam-se noutro, e assim não foi possível tirarmos qualquer conclusão destas inconsistências. De resto estava presente toda a informação necessária, desde os vários *PTSs*, compostos pelas diferentes tarefas, até às transições existentes entre os *PTSs*, com a indicação da tarefa a ser executada para se proceder à respetiva transição. Aqui destaca-se ainda o facto do ficheiro ser gerado automaticamente, enquanto que, para obter o outro ficheiro, o desenvolvedor tem de simular um cenário, executando a sequência de passos até completar a tarefa que pretende automatizar.

Porém, no ficheiro exportado a partir da simulação do cenário, é apresentada a sequência de tarefas executadas, pela ordem em que foram realizadas, o que desde logo representou uma grande vantagem. Enquanto no anterior havia referência a todas as tarefas que componham o modelo de tarefas, no ficheiro de um cenário, apenas as tarefas executadas durante o mesmo é que estavam representadas. Para além disso, surgiam outras informações adicionais, tais como a indicação do nível em que as tarefas se encontram na árvore. No entanto, essas não se revelavam de interesse para o desenvolvimento da abordagem, por isso não foram consideradas. Contudo, também aqui detetámos uma inconsistência, mas como essa incidia sobre informações que não iam ser utilizadas, foi possível ignorá-las sem qualquer prejuízo para o trabalho.

Por fim, e comparando as duas alternativas, achou-se que a melhor solução seria utilizar o ficheiro resultante da simulação (cenário), pois, apesar de ser requerida uma simulação manual para selecionar o cenário do modelo de tarefas a ser automatizado, a informação contida no ficheiro XML é mais objetiva e consequentemente menos suscetível a erros de interpretação. Acrescenta ainda o facto de este ficheiro revelar, de forma exata e ordenada, os passos a seguir, desde a primeira até à última tarefa, até atingir o objetivo predefinido. Por outro lado, pelo ficheiro dos PTSs não temos a noção de qual é o objetivo definido, visto que, nada indica qual é a primeira tarefa a realizar nem em qual tarefa se vai terminar.

Desta forma, os dados necessários para dar início à automatização de tarefas, garantida através da geração de *scripts*, são este ficheiro (cenário) juntamente com o modelo de tarefas enriquecido.

3.2 - Processo de Obtenção das *Scripts*

Agora que foram clarificados todos os passos necessários para a obtenção, quer do modelo de tarefa enriquecido, quer do cenário relativo à simulação da tarefa a ser automatizada, é altura de dar forma a essa automatização. Como mencionado, para a concretização deste processo é exigida a exportação de ambos, modelo de tarefa e cenário, para dois distintos ficheiros, no formato XML. A partir daqui o processo divide-se nas seguintes etapas:

- Extração da informação relevante, a partir do modelo de tarefa (nome das tarefas e outras informações adicionais associadas, tais como a localização dos *screenshots* requeridos pela script);
- Extração da indicação da ordem de execução dos passos da tarefa, a partir do cenário;
- Criação automática da *script* de automatização, com base nas informações obtidas através das duas etapas anteriores;

Inicialmente era fundamental prestar atenção a ambos os ficheiros XML gerados pelas ferramentas para perceber que informação é que estava presente em cada um e dentro dessa, qual a que poderia ser útil para o trabalho em causa. Assim, no ficheiro relativo ao modelo de tarefa, optou-se por considerar apenas os elementos <*Name*> e <*Description*> (Figura 19). Tal como já foi explicado, o conteúdo do campo <*Name*> é que dará origem à respetiva função do *Sikuli*. Por outro lado, o campo <*Description*> pode conter diferentes tipos de dados, dependendo da regra em utilização (ver secção 3.1.1 - Regras de Enriquecimento dos Modelos de Tarefas).

```
<Task Identifier="OpenApp Skype" Category="application" Iterative="false"
    <Name>OpenApp Skype</Name>
    <Description>C:/Program Files (x86)/Skype/Phone/Skype</Description>
    <TemporalOperator name="SequentialEnabling"/>
    <Parent name="Skype makeCall"/>
    <SiblingRight name="WaitT SkypeWindow"/>
</Task>
```

Figura 19. Fragmento do ficheiro de um modelo de tarefa enriquecido.

No outro ficheiro, Figura 20, onde surge a sequência de passos executados para completar a tarefa pretendida, apenas foi tido em conta o elemento <*step*>, mais precisamente, o seu atributo "*executed_task*". Este contém o nome de cada um desses passos, e deve ser comparado com os nomes presentes no ficheiro do modelo de tarefa, de forma a verificar a compatibilidade de ambos.

```
<step executed_task="Enter ContactName">
    <completed_task name="Enter ContactName" level="2"/>
</step>
<step executed_task="Press Contact">
    <completed_task name="Option Find" level="1"/>
    <completed_task name="Press Contact" level="2"/>
</step>
```

Figura 20. Fragmento do ficheiro de um cenário.

Para a concretização dessa análise e posterior extração de informação presente nos ficheiros XML, foi seguida uma abordagem baseada no *XPath Parser*, por via da linguagem *Java*. O *XPath*

(*XML Path Language*) é uma linguagem que permite a procura de informação num ficheiro XML, navegando através dos elementos e atributos aí existentes. Para tal, é requerida a definição de expressões *XPath*, que usam uma notação semelhante à utilizada pelos *URLs*, para tratarem de partes específicas do ficheiro. Essas podem ser avaliadas com diferentes fins, por exemplo, para selecionar nodos para processar, para testar se uma condição é verdadeira ou não, ou apenas para devolver um valor, dando assim origem a diferentes tipos de objeto (*node-set*, *boolean*, *number* e *string*). Exemplificando, a expressão "**livro/autor**" retorna um conjunto de nós dos elementos <autor> contidos nos elementos <livro>. Ainda é possível adicionar predicados às expressões, por exemplo, "**livro[@tipo="Ficção"]**" refere-se aos elementos <livro> cujo atributo "tipo" tenha o valor "Ficção".

Prosseguindo para a abordagem em estudo, foi criado um objeto *Document* para cada um dos ficheiros, a partir de uma *stream* (*FileInputStream*). A partir daqui é sobre esses *Documents* que serão avaliadas as expressões *XPath* criadas para obter dos ficheiros *XML* a informação pretendida. Para selecionar/obter, a partir dos ficheiros dos modelos de tarefas, o nome (<Name>) e o conteúdo presente na descrição (<Description>) de todas as tarefas que possuam este último elemento, foram usadas, respetivamente, as seguintes expressões: "***//Task[Description]/Name/text()***" e "***//Task[Description]/Description/text()***". Relativamente aos cenários, a expressão aplicada para retirar os nomes das tarefas foi ***"//step/@executed_task***". Todas as diferentes informações são armazenadas separadamente para posteriormente serem tratadas.

Após a obtenção dos dados requeridos para a criação das *scripts*, passou-se à implementação de um algoritmo capaz de manipular a informação extraída e assim gerar a *script* desejada. Este passa inicialmente por comparar os nomes das tarefas do modelo com os retirados do ficheiro relativo ao cenário, de forma a guardá-los de acordo com a sequência em que as tarefas foram executadas. De seguida é feita uma leitura iterativa a todos esses nomes, para verificar qual das regras de nomenclatura está a ser utilizada em cada um, e assim obter as funções do *Sikuli* correspondentes, mantendo-as numa lista. Ao mesmo tempo que são listadas, seguindo a mesma ordem, as informações úteis presentes na descrição de cada uma das tarefas. É aí que então é criado o ficheiro PYTHON. Neste são escritas algumas linhas de código comuns a todas as *scripts*, nomeadamente no que se refere ao tratamento de possíveis erros que podem surgir durante a execução das *scripts*. Adicionalmente, ao iterar sobre as listas criadas anteriormente, serão escritas, linha a linha, as funções *Sikuli* apropriadas. Tudo isto dará origem à script de automatização da tarefa que foi executada *à priori* no cenário de simulação.

Na figura que se segue (Figura 21) é apresentada uma vista geral de todo este processo, no contexto de um caso de estudo (função de substituição do *Notepad*). É visível que os nomes das tarefas estão de acordo com as regras definidas anteriormente para o enriquecimento dos modelos, sendo ilustrado o processo de transição desses para as respetivas funções *Sikuli*.

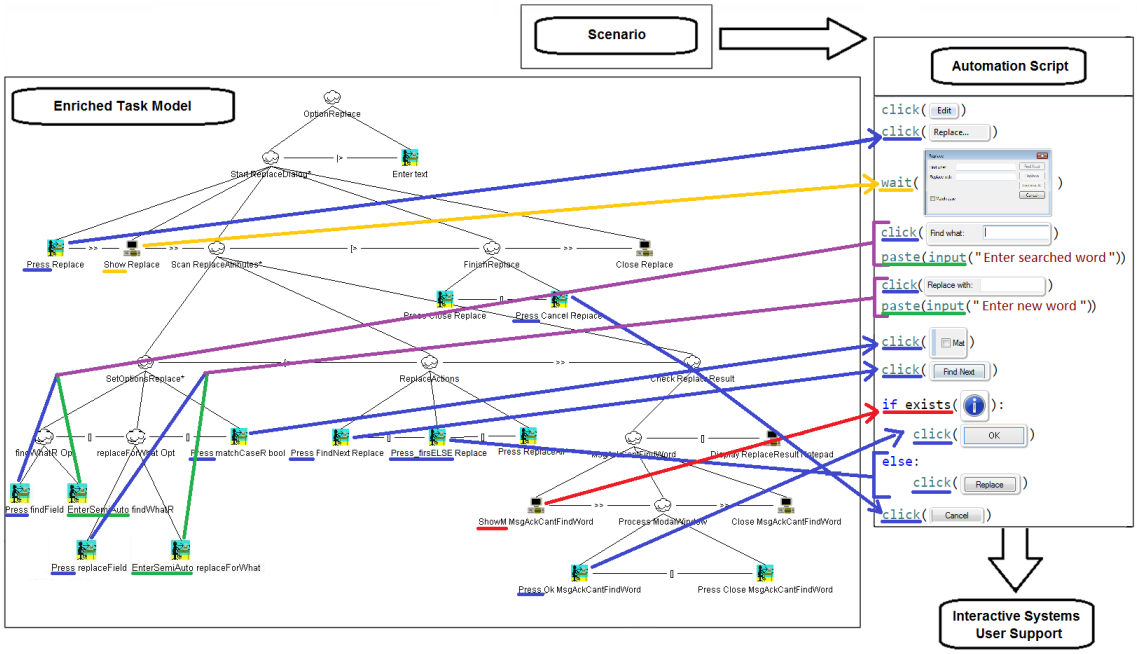


Figura 21. Visão geral do processo de transformação (Modelo de Tarefas Enriquecido –> Cenário –> Script de Automatização -> Aplicação Simplificada), no contexto de um caso de estudo (função de substituição do *Notepad*).

3.2.1 - User Support System (USS)

O USS é uma ferramenta que pretende servir apenas os desenvolvedores/programadores. Esta disponibiliza o mecanismo através do qual eles poderão obter, automaticamente, as *scripts* de automatização pretendidas, de acordo com os modelos desenvolvidos (o modelo de tarefas enriquecido e a descrição da simulação do cenário). Sendo por isso um dos pilares para se conseguir aplicar a abordagem, pois para o desenvolvimento de qualquer uma das interfaces expostas no capítulo 4, é imprescindível o uso do *USS*.

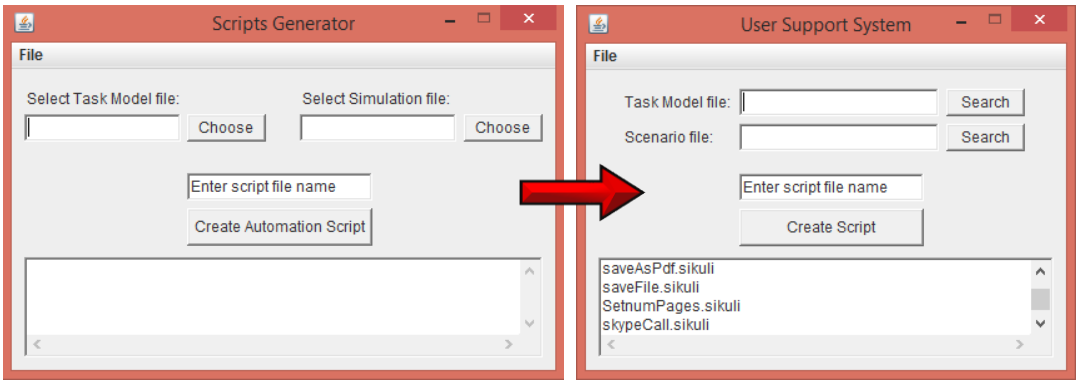


Figura 22. *Evolução da GUI do* User Support System.

O uso desta interface só é requerido após o processo inicial ter sido concretizado, ou seja, a obtenção do modelo (enriquecido) da tarefa a automatizar e a simulação da sequência de passos a seguir para completar essa mesma tarefa. Tal como é possível observar na Figura 22, ela só necessita que o desenvolvedor indique o caminho dos respetivos ficheiros (do modelo de tarefas e da simulação), mostrando uma caixa de diálogo onde o utilizador os pode procurar e

selecionar. Feito isto, existe ainda um campo de texto "*Enter script file name*", onde deve dar o nome ao ficheiro da script que será criado. Por fim, resta apenas clicar sobre o botão "*Create Script*", para que, como o próprio nome indica, a aplicação possa gerar, com base em informações recolhidas a partir dos ficheiros já carregados, a script de automatização da tarefa pretendida.

3.2.2 - IDE Sikuli vs RunServer

Como aqui se pretende gerar automaticamente as *scripts* de automatização de tarefas, de forma a não ser requerido nenhum conhecimento de programação, nomeadamente acerca da linguagem de *scripting* utilizada para o efeito (*Python*), a opção de criá-las, diretamente, através do *IDE Sikuli* não se coloca.

Um dos focos centrais deste projeto passa por oferecer aos utilizadores uma experiência mais agradável, quer a nível de facilidade quer de poupança de tempo, na utilização de Sistemas Interativos. Daí que um dos pontos que exigiu maiores esforços baseou-se na tentativa de reduzir ao máximo, sem quaisquer contrapartidas assinaláveis, o tempo de execução das *scripts*. Inicialmente, percebeu-se que poderiam ser aplicadas à abordagem diferentes formas de execução das *scripts*. No entanto, por motivos alheios, numa primeira fase sentiu-se a obrigação de enveredar pelo método mais comum, a execução das *scripts* através da linha de comandos (usando, indiretamente, o *IDE* do *Sikuli*), recorrendo a alguns comandos específicos. Com a instalação do *IDE* (através de um ficheiro *JAR*) é criado o ficheiro *runsikulix.cmd*, o qual é responsável por oferecer essa possibilidade. Este tanto permite inicializar o *IDE* como executar as *scripts*, sem ter de recorrer diretamente ao mesmo.

No Windows, a estrutura dos comandos exigidos é a seguinte: "**<*Sikuli Path*>/runsikulix(.cmd) [opções]**". "***Sikuli Path***" é a pasta onde o *Sikuli* foi instalado, contendo todo o material necessário ao seu funcionamento. Executando esse comando sem qualquer opção adicional inicia-se simplesmente o *IDE Sikuli*. Se a intenção recair sobre a execução de uma script, a expressão a adicionar ao comando anterior consiste em "**-r <sikuli-folder>/<file>**". Onde "**sikuli-folder**" corresponde ao local onde estão armazenados os ficheiros *SIKULI*, enquanto "**file**" é o nome do ficheiro a ser executado (por exemplo, C:/SikuliX/notepadReplace.sikuli). Como a implementação da abordagem baseou-se na linguagem de programação *Java*, era necessário dar a ordem de execução das *scripts* a partir da aplicação aí desenvolvida. Desse modo, o comando necessário, que é disponibilizado pela biblioteca "sikulixapi.jar", é o que se segue: ***Sikulix.run("java -Dsikuli.FromCommandLine -jar*** <*Sikuli JAR Path*> ***-r*** <*SIKULI file Path*>**")**.

Em contraste, encontra-se a opção de atribuir a responsabilidade da execução das *scripts* a um servidor, de seu nome *RunServer* [76]. Só devido ao facto desta se encontrar em desenvolvimento à data de início da implementação da abordagem, é que teve de ser deixada para último plano. Com a utilização do *RunServer*, o processo de execução das *scripts* torna-se mais eficiente, pois ao invés do que acontece quando essa execução é feita através da linha de comandos – existe um *delay* de 10-15 segundos entre o lançamento da script e o início da sua execução, cada vez que o processo é realizado – com esta solução apenas existe um tempo de espera inicial [76] (aquando da inicialização do Sistema de Simplificação), para ligar e configurar o servidor, que ronda os 2 segundos. Contudo, ao lançar depois as *scripts*, estas iniciam-se

instantaneamente. Optou-se por seguir este caminho pois o tempo que era necessário esperar em cada ação, realizada na alternativa anterior, acabava por tornar o processo mais maçador para o utilizador, e com esta nova solução, conseguiu-se uma melhoria bastante significativa.

A realização dos vários pedidos ao servidor é feita através do carregamento de endereços específicos (num *browser*), os quais serão posteriormente apresentados. Assim sendo, e como a utilização direta de um *browser* não seria de todo agradável para a usabilidade da aplicação, a alternativa encontrada foi recorrer a um *batch file*. Um *batch file* é um tipo de ficheiro de script que consiste numa sequência de comandos a ser executados pelo interpretador da linha de comandos (cmd.exe), armazenados num ficheiro de texto simples. Este em particular permite abrir um endereço sem a necessidade de recorrer diretamente ao *browser*. A sua utilização é bastante simples, bastando apenas introduzir - ***callurl*** "<endereço>" **-** na linha de comandos, a partir da localização do ficheiro (exemplificando, ***callurl "http://www.google.pt"***). De seguida, descreve-se os passos a percorrer de acordo com neste novo método.

- Inicialmente é necessário iniciar o servidor na máquina onde as *scripts* serão executadas, para isso é executado o comando "<Sikuli Path>***/runsikulix -s***" na linha de comandos.
- Após este estar ativo, deve-se criar uma pasta – *sikulixrunserver* - na "home" (do sistema em questão), onde as *scripts* ficarão armazenadas e, através do método explicado acima, carregar o endereço "***http://localhost:50001/scripts/home/sikulixrunserver***" para a sua configuração.
- Seguidamente, é requerida a ativação do "*script runner*" (neste caso para *Python*), que é feita através do endereço "***http://localhost:50001/startp***".
- Assim, para executar uma script deve ser introduzido "***http://localhost:50001/run/***<*script name*>".
- No final de todo o processo, para desativar o servidor é usado o endereço "***http://localhost:50001/stop***".

É de notar que no caso em que é pretendido que o servidor (*RunServer*) corra sobre outra máquina, que não aquela onde estão a ser efetuados os pedidos (por exemplo, numa máquina virtual, como será adotado neste trabalho), nos comandos anteriores no lugar de "*localhost*" deverá ser introduzido o *IP* dessa primeira máquina.

Outra das possibilidades a seguir como forma de execução das *scripts* consiste na utilização da *API* do *Sikuli* (sikulixapi.jar) como uma biblioteca (padrão) *Java*, no programa em desenvolvimento [75]. Isto porque o núcleo do *Sikuli* é escrito nessa linguagem. Para tal ser possível é requerido que o *Sikuli* esteja totalmente configurado no sistema. A partir daí, falta apenas importar as classes necessárias à escrita do código para criação das *scripts*. Esta não foi adotada sobretudo por acarretar maiores obstáculos, nomeadamente no processo de geração automática das *scripts* (ficheiros Java, neste caso), e daí não se tirar vantagens significativas.

3.3 - Interface Simplificada

Numa fase em que todo o mecanismo já foi devidamente implementado, fica a faltar o meio mais importante, do ponto de vista dos utilizadores finais, i.e., a interface gráfica. Esta pretende ser o ponto de comunicação entre o utilizador e as aplicações sobre as quais será aplicada a automatização de tarefas, ou seja, é através dela que o utilizador pode dar ordem ao sistema interativo para executar uma dada tarefa. Daí que, é importante que garanta uma maior simplificação, eficiência e satisfação na realização das tarefas pretendidas pelos seus utilizadores. Será por diante designada de Interface Simplificada.

A interface simplificada a ser criada dependerá do uso que se lhe quer dar. Isto quer dizer que tanto a especificação do objetivo da aplicação intermédia (aplicação de suporte) a desenvolver, como o seu público-alvo, entre outros fatores, devem ser tidos em conta durante o processo de elaboração da interface, de forma a adaptá-las o máximo possível aos seus utilizadores. É possível comprovar este facto na presente descrição dos diferentes casos de estudo, nomeadamente nas secções 4.2 - Sistema de Ajuda e 4.3 - Sistema de Simplificação - Home Care Application.

Para esta fase, é utilizado o editor de *GUIs* de aplicações *Java*, presente no *IDE Eclipse*. Recorrendo a este editor, os desenvolvedores são livres de criar novas GUIs, através das funções de "*drag and drop*" sobre os *widgets* disponíveis. Nas Figura 24 e Figura 28 são visíveis duas das interfaces criadas no âmbito deste projeto, recorrendo ao dito editor. Posteriormente à disposição final dos vários elementos ao nível da interface, é necessário associar a cada *widget*, a ação desencadeada por um evento de utilizador que habilitará a execução da *script Sikuli* correspondente. Será através destes *widgets*, presentes na nova GUI, que o utilizador poderá dar a ordem, ao sistema interativo, para realizar a tarefa pretendida. Isso é facilmente concretizado, bastando adicionar um dos seguintes fragmentos de código, dependendo do método que será utilizado para executar as *scripts*. Assim sendo, para a execução das *scripts* através da linha de comandos, o comando é:

- ***Sikulix.run ("java -Dsikuli.FromCommandLine -jar*** "<*Sikuli Path* (*JAR*)>" ***-r*** "<*SIKULI file Path*>"**")**.

Se for usado este método, mas sobre a máquina virtual, o excerto de código consiste em:

- ***vmrun -T ws -gu*** <*Username*> ***-gp*** <*password*> ***runProgramInGuest*** "<*VMX file Path*>" ***-activeWindow -interactive*** "<*Path* - ficheiro *EXE* do *Java*>" ***-Dsikuli.FromCommandLine -jar*** "<*Sikuli JAR file Path*>" ***-r*** "<*script file Path*>".

Para a execução através do servidor *RunServer*, o fragmento de código a incluir é:

- ***Runtime.getRuntime().exec("cmd /c*** "<*callurl batch file Path*>" ***http://localhost:50001/run/***<*script file name*>***")***.

Da mesma forma, que se for aplicado a uma máquina virtual, o código altera-se para:

- ***Runtime.getRuntime().exec("cmd /c*** "<*callurl batch file Path*>" ***"http://***<IP_Máquina Virtual>***:50001/run/***< *script file name*>***" ")***.

Note-se que as explicações dos diferentes métodos aqui mencionados encontram-se nos respetivos capítulos deste trabalho. Posto isto, no final de todo este processo, a nova *GUI* simplificada, qualquer que ela seja, estará preparada para começar a ser utilizada.

3.4 - Utilização da Máquina Virtual

Ao longo do desenvolvimento do nosso projeto e, mais precisamente, da respetiva investigação realizada, nomeadamente a partir de [33], decidiu-se explorar a ideia de colocar a execução das *scripts* sobre uma máquina virtual. Isto deveu-se à razão de o *Sikuli* retirar ao utilizador o controlo da utilização do computador (rato e teclado). E contrastando esse facto, tentava-se uma forma de conseguir obter um melhor aproveitamento dessa utilização. No entanto, tendo em conta que uma máquina virtual, apesar de ser instalada sobre um sistema nativo, é independente deste, ou seja, por si só não partilham qualquer memória ou informação, isso poderia levantar alguns entraves à aplicação prática desta ideia. Por exemplo, suponha-se o seguinte caso: um utilizador está a editar um documento no Word, no sistema nativo, e a dado momento necessita de ajuda para realizar uma dada tarefa (inserir numeração de páginas personalizada, por exemplo), a qual é disponibilizada pelo Sistema de Ajuda (ver secção 4.2 - Sistema de Ajuda). Assim sendo, ele opta por utilizá-la. Mas, sendo essa ajuda fornecida através da execução da respetiva *script* na máquina virtual, não é possível ela operar sobre esse mesmo documento que está a ser atualmente editado no sistema nativo.

Posto isto, é importante distinguir os diferentes tipos de ajuda que se pode fornecer através da abordagem desenvolvida, por um lado, recorrendo ao uso de uma máquina virtual, e por outro, executando as *scripts* diretamente no sistema nativo.

Como ponto de partida, realça-se as principais vantagens que se podem retirar da execução das *scripts* sobre uma máquina virtual. Pode-se esconder a realização de passos intermédios que não sejam relevantes, mostrando ao utilizador apenas o resultado final (objetivo pretendido), consegue-se que a tarefa pretendida seja realizada sem haver uma perda de controlo, quer do rato quer do teclado, por parte do utilizador, e ainda, como essa execução se pode processar de forma completamente invisível ao utilizador, permite que este possa executar outras tarefas (no sistema nativo) enquanto aguarda. Por outro lado, a execução de uma *script* no SO nativo implica que o utilizador perca o controlo da interação com o sistema, tendo de esperar até ao final da sua execução.

Obviamente, esta variante da abordagem só faz sentido em casos cuja automatização da tarefa seja total, não sendo requerida a introdução de dados intermédios por parte do utilizador para que a tarefa seja materializada. Ao contrário da abordagem "original", esta não serve o propósito de ensinar/treinar os utilizadores a usar certos sistemas interativos (Sistema de Ajuda), pois uma das suas implicações é tornar a execução dos passos invisível, mostrando apenas o resultado final desejado.

Prosseguindo para a implementação propriamente dita, a ferramenta encontrada capaz de passar a ideia acima referida, da teoria à prática, foi a *vmrun* [78]. Refere-se a uma aplicação que permite controlar máquinas virtuais a partir da linha de comandos da máquina física, e que vem incluída no *VMware Workstation* [77]. Este é um ambiente que permite gerir e configurar

diferentes máquinas virtuais sobre uma mesma máquina física, e é o que foi adotado neste projeto. Para o correto funcionamento da *vmrun*, é necessário ter instalado na máquina virtual em utilização o *VMware Tools*, um pacote com drivers (entre outros *softwares*) que permitem melhorar o seu desempenho. Para além das tarefas básicas de iniciar, pausar e encerrar uma máquina virtual, a *vmrun* permite ainda gerir e controlá-la internamente, nomeadamente ao nível da execução de *software* aí instalado. Tudo isto pode ser realizado através da execução dos respetivos comandos, disponibilizados pela ferramenta, pela linha de comandos da máquina física (*host*). Especificam-se, de seguida, os principais comandos que podem vir a ser úteis na aplicação da abordagem.

- Colocar uma máquina virtual a correr no *VMware Worstation*, através da ferramenta *vmrun*: ***vmrun start*** <"*VMX file Path*">;
- Executar um programa na máquina virtual: ***vmrun -T ws -gu*** <nome de utilizador> ***-gp*** <password> ***runProgramInGues****t* <"*VMX file Path*"> ***-activeWindow -interactive*** <"*application EXE file name*">;
- Execução de um ficheiro/aplicação JAR: ***vmrun -T ws -gu*** <*username*> ***-gp*** <*password*> ***runProgramInGuest*** <"*VMX file Path*"> ***-activeWindow -interactive*** <"*Java EXE file Path*"> ***-jar*** <"*application JAR file Path*">; e por fim
- Execução de uma *script*: ***vmrun -T ws -gu*** <*username*> ***-gp*** <*password*> ***runProgramInGuest*** <"*VMX file Path*"> ***-activeWindow -interactive*** <"*Java EXE file Path*"> ***-Dsikuli.FromCommandLine -jar*** <"*Sikuli JAR file Path*"> ***-r*** <"*script Path*">. Caso seja através do servidor *RunServer*, o comando é o seguinte: "***cmd /c*** <"*callurl batch file Path*"> ***"http://***<IP_máquina virtual>***:50001/run/***<*script file name*>".

Em seguida é apresentado um caso exemplificativo da utilidade de uma máquina virtual na aplicação da abordagem, o qual foi aplicado num dos casos de estudo que serão explicitados na secção 4. Uma pessoa idosa, com um conhecimento ínfimo de sistemas informáticos, pretende fazer uma chamada (via *Skype*) para o seu filho, (sabendo de antemão que as configurações necessárias para tal foram já efetivadas). Assim, é disponibilizado no sistema nativo uma nova interface, mais simplista, através da qual o utilizador vai, indiretamente, interagir com o *Skype*. Ele só tem de clicar sobre o botão "Ligar para…", sendo os restantes passos do processo efetuados através da respetiva script. Esta será então executada sobre uma máquina virtual, e é responsável por iniciar o *Skype* (tendo os dados de utilizador previamente registados), procurar o contacto do "filho" e fazer a chamada para o mesmo. Só após se ter dado início à chamada, é apresentada ao utilizador a janela do *Skype*, mostrando que a mesma está a decorrer, enquanto os passos efetuados até então, foram ocultados.

Neste caso em particular, automatização do *Skype*, apenas é mostrado um exemplo específico de utilização, pois, se o computador utilizado for partilhado por várias pessoas, não seria apropriado haver inicialmente uma sessão já iniciada no *Skype*. Para tal, antes de qualquer utilização da aplicação simplificada, seria pedido ao utilizador que introduzisse a sua identificação (nome e palavra-passe). Assim, a pessoa responsável por criar e disponibilizar os automatismos aos utilizadores, teria de associar previamente a cada identificação o respetivo modelo de tarefas, o qual possuiria todas as informações necessárias para o correto desenrolar do processo.

4 - Casos Práticos de Estudo

Dado que neste ponto a abordagem está já totalmente apresentada e elucidada, é altura de demonstrar a sua utilidade prática. Foram desenvolvidos vários casos de estudo, que fizessem uso das diferentes vertentes da abordagem, cada um com o seu objetivo bem definido. A utilização dos mesmos foi, posteriormente, experienciada por pequenos grupos de utilizadores escolhidos aleatoriamente (de entre os públicos-alvo previamente definidos). Assim sendo, serão apresentadas, neste capítulo, as diferentes aplicações desenvolvidas com base na abordagem em estudo, explicando as diferenças existentes ao nível das *GUIs*, dos seus objetivos e públicos-alvo, dos métodos a que recorrem, exemplificando ainda algumas das funcionalidades oferecidas.

4.1 - Função de Procura/Substituição no *Notepad*

Inicialmente a demonstração do USS passava por construir, através da ferramenta *CTTE*, o modelo de tarefas relativo à opção de substituição de uma dada palavra por outra, do *Notepad*. Assim, de forma a auxiliar o processo, foram feitas várias experiências práticas de utilização dessa mesma funcionalidade, no *Notepad*. Isto para perceber os vários caminhos passíveis de serem percorridos no decorrer da execução, dependendo da evolução do estado da aplicação. Exemplificando, se não for encontrada (no texto) nenhuma ocorrência da palavra a substituir, obviamente que a resposta do sistema será díspar do caso em que a substituição tenha sido realizada com sucesso. Tal como é possível verificar na Figura 21, na construção desse modelo, já se aplicou as regras de enriquecimento do modelo, conforme definidas no capítulo 3.1.1 - Regras de Enriquecimento dos Modelos de Tarefas, de forma a deixá-lo pronto para o processo de transformação. Após obter o modelo de tarefas enriquecido, este é guardado no formato XML para depois ser importado para a ferramenta *MARIAE*. É aí que se procede à simulação da realização dos vários passos até atingir o objetivo definido, utilizando para tal o simulador de tarefas oferecido. Depois, esse cenário é também guardado no formato XML. Com estes dois ficheiros em posse, e recorrendo à ferramenta *USS*, é assim possível gerar automaticamente a respetiva *script*. A Figura 23 ilustra parte da execução dessa mesma *script*.

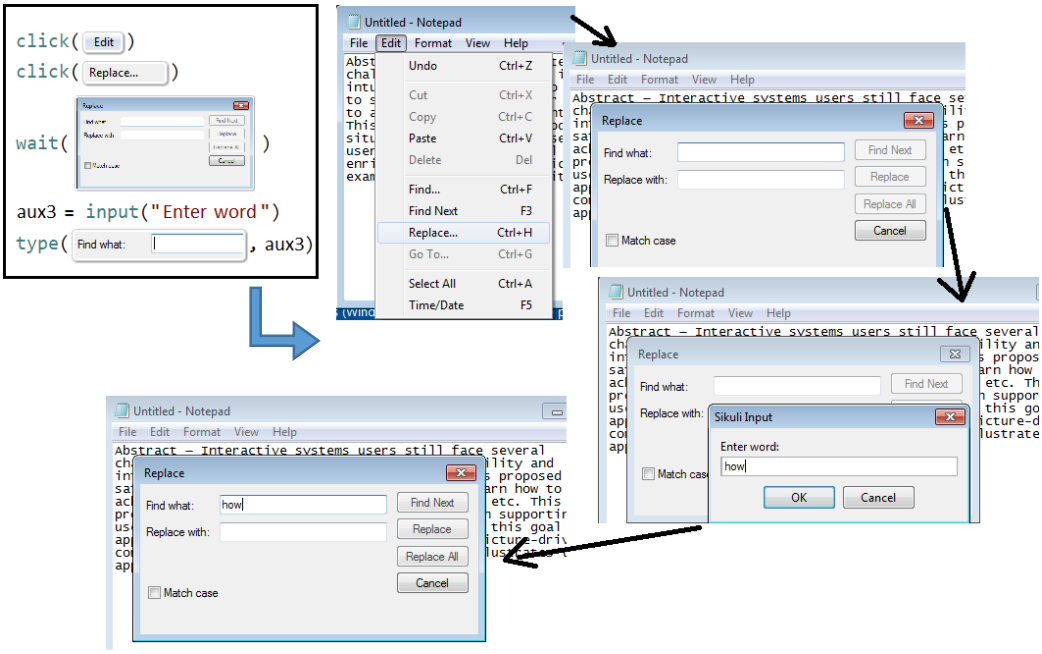


Figura 23. Ilustração da execução de parte da *script* relativa à procura/substituição no *Notepad*.

4.2 - Sistema de Ajuda

Esta variante da abordagem pretende servir ao utilizador como um sistema de ajuda baseada na demonstração, no uso de sistemas interativos. A grande vantagem em relação aos tipos de ajuda normalmente encontrados nos programas atuais, meramente textual, é que ela é obtida através da visualização da execução de tarefas automatizadas. Quer isto dizer que para além de ensinar a realizar determinadas tarefas, também tem a capacidade de as executar pelo utilizador. Acredita-se que desta forma está-se a beneficiar o processo de aprendizagem do utilizador, ao mesmo tempo que se reduz o esforço necessário para realizar as tarefas. Isto comparando com o facto de o utilizador ter de ler o que deve fazer para completar uma tarefa e só depois poder passar à prática. Para além disso, permite substituir o utilizador na realização de tarefas que, manualmente se revelam aborrecidas dada a sua elevada repetição, tornando mais agradável a utilização destes sistemas. Parte do objetivo passa também por dar a conhecer ao utilizador algumas das funcionalidades mais avançadas de uma aplicação, mostrando-lhe como devem ser executadas. Assim sendo, definiu-se que este sistema se destina principalmente a utilizadores com razoáveis conhecimentos informáticos.

É possível observar pelas Figura 24 e Figura 28, que independentemente da aplicação para a qual o sistema de ajuda será aplicado, o *layout* da interface manter-se-á. Assim sendo, o sistema de ajuda tem, em cima, uma caixa de texto/pesquisa onde o utilizador deve introduzir uma expressão relacionada com a tarefa para a qual está a procurar a ajuda, e em baixo, será apresentada a lista de tarefas resultantes da procura efetuada. Por defeito, são listadas todas as tarefas, da aplicação alvo, cuja ajuda é disponibilizada. Para obter ajuda, os utilizadores clicam sobre o item da lista pretendido, que aciona automaticamente a execução da respetiva tarefa na GUI do sistema principal. Por fim, para além disso, ao longo dos passos que compõem a tarefa, por vezes é requerida a intervenção do utilizador para completar algumas ações (tarefas semi-automatizadas), sendo este sempre previamente informado para tal (ver Figura 27).

Após definida esta variante, passou-se aos casos práticos de estudo. Para tal foram selecionadas duas diferentes aplicações, o *Microsoft Word*, conhecida pela generalidade dos utilizadores, e o *Blender* [72], que consiste num *software* de computação gráfica 3D, *open-source*, usado para modelação, animação, texturização, composição, renderização, edição de vídeo e criação de aplicações interativas em três dimensões [71].

4.2.1 - Word Support System (WSS)

A primeira aplicação deste sistema de ajuda recaiu sobre o Word, pois apesar de ser uma das ferramentas de *software* mais utilizadas, é dotada de algumas funcionalidades mais avançadas, desconhecidas de grande parte dos seus utilizadores, e por outro lado, outras cuja frequência de utilização é tão elevada que se tornam aborrecidas devido à sua repetição.

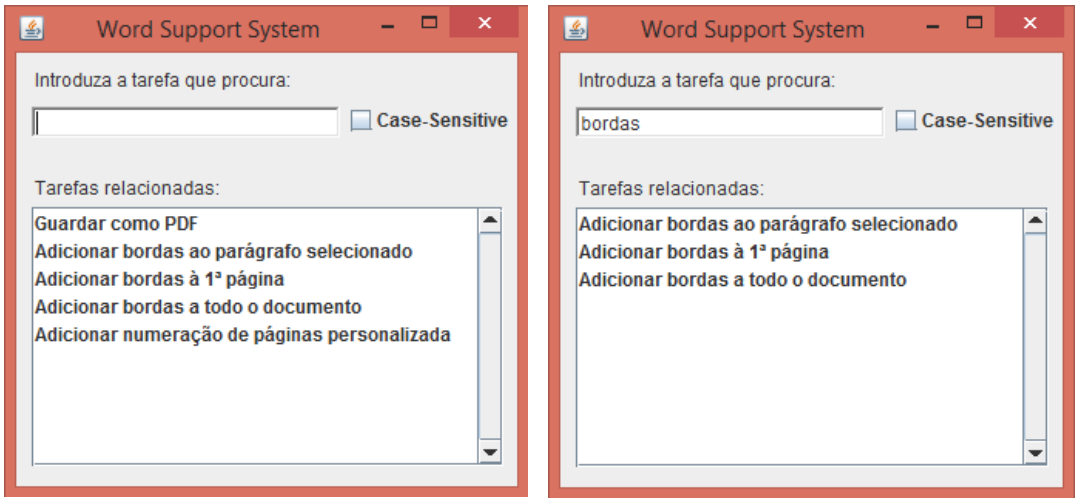


Figura 24. Interface de suporte ao Word (à direita, efetua-se a pesquisa por "bordas").

Decidimos aplicar a abordagem às seguintes funcionalidades: guardar o documento no formato PDF, adicionar bordas personalizadas ao documento (ao parágrafo selecionado, apenas à primeira página ou a todo o documento) e por fim numerar, de forma personalizada, as páginas do documento.

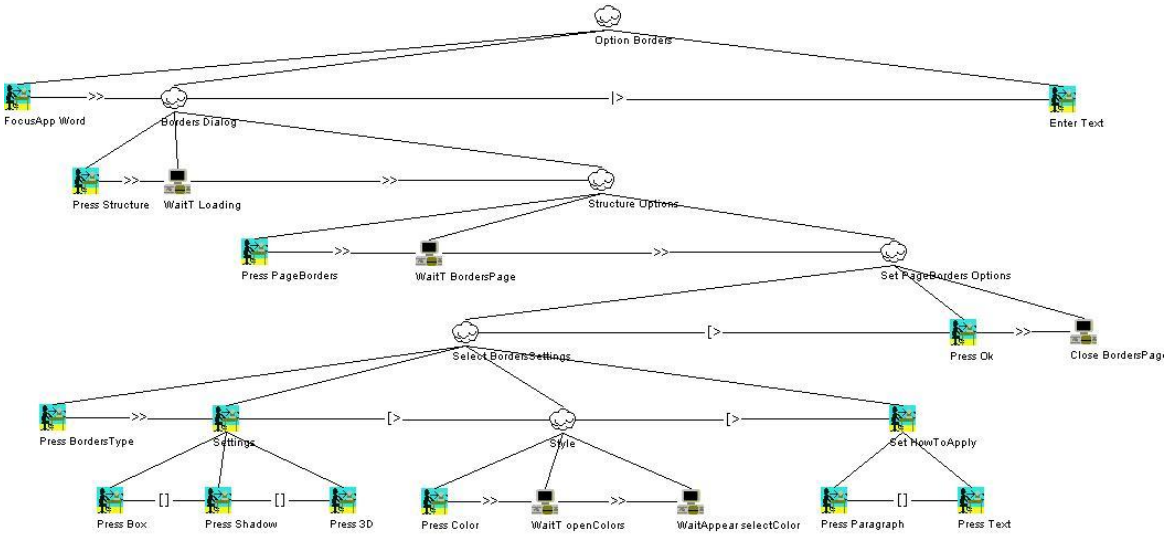


Figura 25. Modelo de tarefas da opção de adicionar bordas ao documento.

Presentemente apresentam-se imagens relativas a parte do processo de transformação aplicado sobre a tarefa de adicionar bordas ao documento, sendo que a Figura 25 representa o modelo de tarefas e a Figura 26 mostra um excerto do respetivo cenário, realizado através do simulador de tarefas, e ainda a *script* (completa) gerada a partir desses dois ficheiros.

```
<step executed_task="Press PageBorders">
    <completed_task name="Press PageBorders" level="3"/>
</step>
<step executed_task="WaitT BordersPage">
    <completed_task name="WaitT BordersPage" level="3"/>
</step>
<step executed_task="Press BordersType">
    <completed_task name="Press BordersType" level="5"/>
</step>
<step executed_task="Press Box">
    <completed_task name="Settings" level="5"/>
    <completed_task name="Press Box" level="6"/>
</step>
<step executed_task="Press Color">
    <completed_task name="Press Color" level="6"/>
</step>
<step executed_task="WaitT openColors">
    <completed_task name="WaitT openColors" level="6"/>
</step>
<step executed_task="WaitAppear selectColor">
    <completed_task name="Style" level="5"/>
    <completed_task name="WaitAppear selectColor" level="6"/>
</step>
<step executed_task="Press Paragraph">
    <completed_task name="Select BordersSettings" level="4"/>
    <completed_task name="Set HowToApply" level="5"/>
    <completed_task name="Press Paragraph" level="6"/>
</step>
```

```
from org.sikuli.natives import Vision
Vision.setParameter("MinTargetSize", 8)
try:
        App.focus("Word")
        click( ESTRUTURA )
        wait(1)
        click( Limites de Página )
        wait(2)
        click( Cε )
        popup("Selecione o estilo da linha")
        wait(10)
        popup("Selecione a cor")
        wait(1)
        wait( Largura: ,10)
        click( OK )
except FindFailed:
        popup("Element not found")
```

Figura 26. Fragmento do cenário da simulação da tarefa de adicionar bordas ao documento (à esquerda); *script* de automatização gerada (à direita).

Adicionalmente, é também ilustrada a sequência de passos a seguir para a execução dessa mesma *script* (Figura 27).

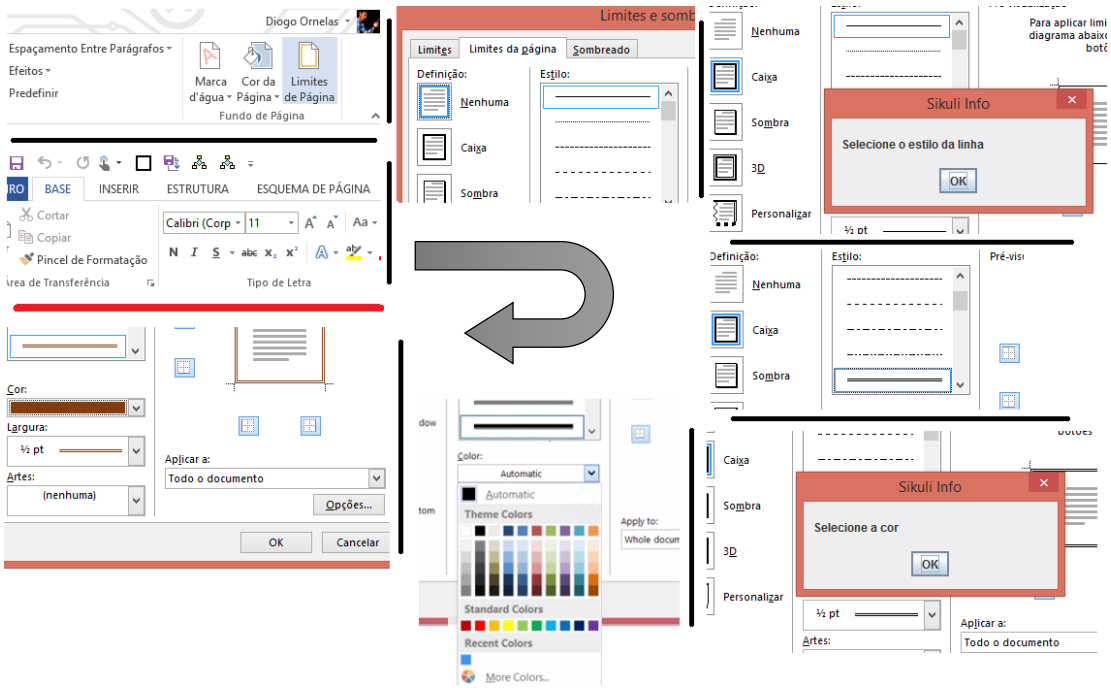


Figura 27. Ilustração da execução da *script* presente na Figura 26.

4.2.1.1 - Comparação com Macros

Atualmente existe no *Word* uma ferramenta, denominada "*Macros*", cujo objetivo vai de encontro a um dos que se definiram para o desenvolvimento desta abordagem, ou seja, automatizar tarefas repetitivas, poupando tempo e esforço ao utilizador. Daí apresentar-se aqui algumas ideias acerca dessa mesma funcionalidade. Basicamente, permite ao utilizador gravar, numa *macro*, os vários passos que são necessários realizar para completar uma determinada tarefa, *macro* essa que fica disponível na Barra de Ferramentas, e que pode ser executada através de um clique ou uma combinação de teclas definida para o efeito. Para proceder à gravação de uma *macro*, o utilizador apenas tem de clicar sobre o ícone "Gravação de Macros", existente na Barra de Estado, que abrirá uma janela de configuração, onde deve dar um nome à mesma, a forma como será despoletada (através do rato ou teclado), e se ela estará disponível apenas para o documento atual ou para todos os documentos. Após concluir esta simples configuração, o sistema está pronto a gravar, pelo que todos os passos que efetuar de seguida, quer sejam ações com o rato ou com o teclado, ficarão armazenados na *macro* atual. Por fim, o utilizador tem de clicar novamente no ícone de "Gravação de Macros", para terminar e guardá-la. É óbvio que esta é uma excelente solução para poupar tempo nas tarefas que se recorrem com mais frequência, e que acabam por se tornar aborrecidas devido às muitas repetições.

Comparativamente à abordagem por nós desenvolvida, esta destaca-se claramente pelo facto de o sistema não tomar o controlo sobre o rato/teclado, e de a tarefa automatizada ser realizada de forma instantânea e completamente invisível para o utilizador, o que permite uma poupança de tempo mais significativa. Por outro lado, no caso de tarefas em que o resultado não seja claramente visível, este pode ser um ponto fraco, pois o utilizador não recebe *feedback* à medida que os passos são executados, não sabendo assim quando é que a tarefa foi concluída, ao contrário do que acontece utilizando a nossa abordagem. Outra vantagem da nossa abordagem é que, como os passos executados pela *script* são visíveis ao utilizador, pode ser também uma forma aprendizagem. Um outro ponto forte da nossa abordagem em relação à *Macros* é que, enquanto esta última é uma ferramenta exclusiva do Office (Word), a abordagem desenvolvida pode ser utilizada para automatizar tarefas em qualquer tipo de aplicação.

Por fim, importa ainda realçar que a *Macros* destina-se a utilizadores que queiram automatizar as suas próprias tarefas, isto é, têm de ter conhecimento de como realizá-la, pois para gravarem a *macro* têm de executar a tarefa uma vez. No caso da nossa abordagem, para a criação das *scripts* existe o USS, que se destina a uma espécie de utilizador intermediário (normalmente, um programador) com conhecimentos ao nível de modelos de tarefas. Enquanto isto, o utilizador final, aquele que irá usufruir da automatização das tarefas, mesmo não sabendo como realizar uma dada tarefa, através da ferramenta disponibilizada - *Word Support System* - será capaz de a efetuar.

4.2.2 - Blender Support System (BSS)

Como já foi referido, o segundo caso de estudo prendeu-se com a aplicação *Blender*. Foi a partir de uma situação que, por experiência própria, achou-se que estava em falta algo deste género. O sucedido foi que, ao experimentar pela primeira vez a ferramenta, deparou-se com dificuldades na realização de algumas tarefas básicas (por exemplo, selecionar e mover um objeto). Aí consultou-se o manual disponibilizado *online*, que apesar de apresentar *screenshots* da *GUI* da aplicação, não nos livrou de alguns obstáculos verificados na passagem do que lá estava escrito para a realização efetiva da tarefa, nomeadamente na localização de alguns elementos da interface presentes nos *screenshots*. Concluindo, esta não era uma maneira eficaz de ultrapassar os nossos problemas, não sendo capaz de mostrar imediatamente ao utilizador como este devia proceder para realizar a tarefa pretendida. Então decidiu-se por aplicar o referido sistema de ajuda. Sendo esta uma ajuda visual, muito mais intuitiva e esclarecedora para o utilizador, é, em simultâneo, capaz de executar a tarefa cuja ajuda era requerida, através da automatização (*script*). Desta forma, não restam dúvidas ao utilizar de como a tarefa deve ser realizada.

Na imagem que se segue (Figura 28) é possível visualizar a simultaneidade com que são apresentadas ao utilizador as duas aplicações em execução, o *Blender* (à direita) e o sistema de suporte (à esquerda). Para que tal aconteça, quando o utilizador tem o *Blender* ativo, ao iniciar o respetivo sistema de ajuda, a disposição dos elementos no ecrã é assim reorganizada.

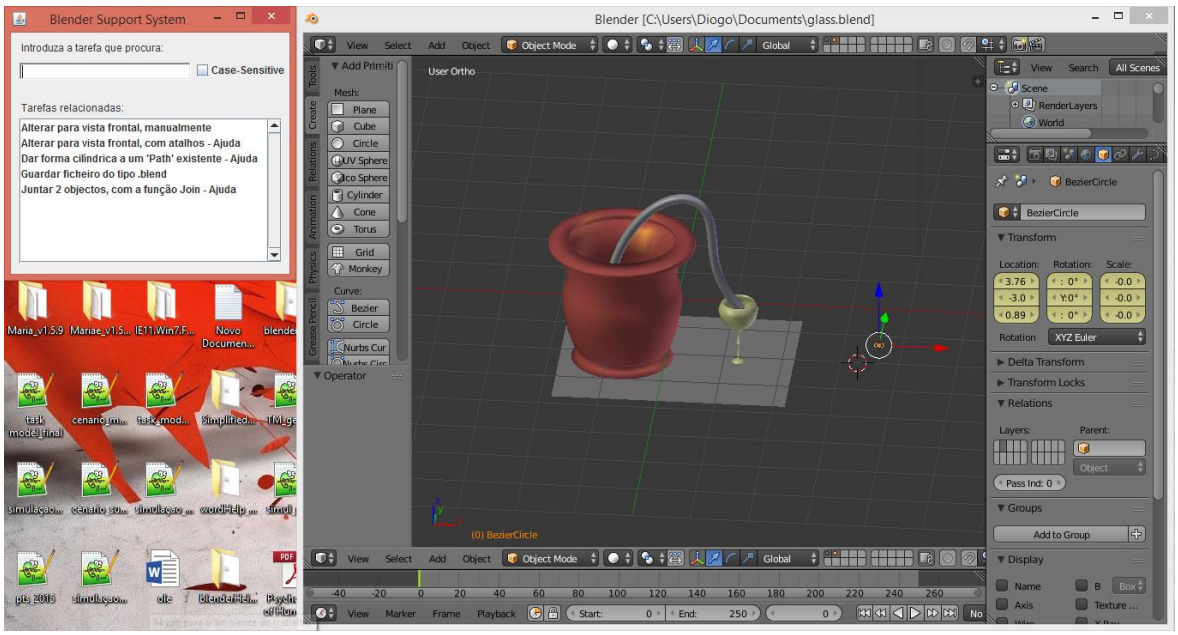


Figura 28. Esquema do ecrã com o *Blender* e o respetivo sistema de ajuda em execução.

Esta disposição das janelas foi pensada e deveu-se a diversos fatores. Primeiramente, era importante que a aplicação de ajuda não se sobrepusesse à aplicação alvo, de forma a não interferir no correto funcionamento do processo de execução das *scripts*. Adicionalmente, também achou-se adequado que a ajuda estivesse sempre disponível no ecrã (desde que o sistema de ajuda tivesse sido iniciado) para quando fosse necessária, poupando o utilizador de ter que a ir "procurar" de cada vez que precisasse de ajuda para realizar uma tarefa.

4.3 - Sistema de Simplificação - Home Care Application

Primeiramente, esta aplicação tem o objetivo de facilitar a utilização de sistemas interativos por parte de pessoas com poucos ou nenhuns conhecimentos na área, contribuindo para a evolução do seu contato com as Tecnologias de Informação. Como é visível na Figura 29, devido às características dos utilizadores alvo, a interface foi pensada e criada da forma mais minimalista possível (relativamente às que o utilizador usaria normalmente para executar as mesmas tarefas), mas cumprindo eficientemente os seus objetivos. Basicamente, consiste num conjunto de (oito) botões, onde cada um será responsável por realizar uma determinada tarefa (previamente definidas pelo desenvolvedor), ou seja, executar a respetiva *script* a ele associada. Considere-se por exemplo a tarefa de aceder às notícias. Normalmente, o utilizador tem de saber que é necessário um *browser*, como executá-lo e, finalmente, introduzir o endereço do *site*. Todas essas etapas e informações necessárias são uma barreira para estes utilizadores, na tentativa de realizar a tarefa. Usando esta abordagem, o utilizador só tem que pressionar um botão para obter o resultado desejado. Não obstante que seja necessário algum conhecimento, ainda que reduzido, para ler as notícias (por exemplo, a manipulação da *scrollbar*).

A nova interface pretende ser um meio através do qual o utilizador vai interagir com diversas aplicações, quer sejam web (e.g., site de previsão do tempo), ou nativas (e.g., *Skype*) existentes no seu SO. Por defeito, é executada no modo *fullscreen*, de forma que a atenção dos utilizadores recaia apenas sobre esta. Neste caso em particular, devido às circunstâncias envolventes, integrou-se apenas tarefas finais, i.e., que não necessitem de qualquer intervenção intermédia por parte do utilizador. Caso contrário estaríamos a desvirtuar o objetivo, que visa facilitar/simplificar ao máximo a utilização de um sistema interativo, por parte de utilizadores novatos.

Com vista a tornar essa utilização do computador uma tarefa mais leve e eficiente, era importante "esconder" a execução da *script* até que estivesse concluída. Isto para não trazer confusão para os utilizadores, pois devido à sua inexperiência, seria algo "estranho" observarem o sistema a funcionar de forma (semi)autónoma, sem que estivessem a interagir com ele. Para tal, e devido ao *Sikuli* apenas operar sobre elementos das *GUIs* que estejam visíveis no ecrã, optou-se por recorrer à ajuda de uma máquina virtual. Esse procedimento está devidamente explicitado em cima (capítulo 3.4 - Utilização da Máquina Virtual).

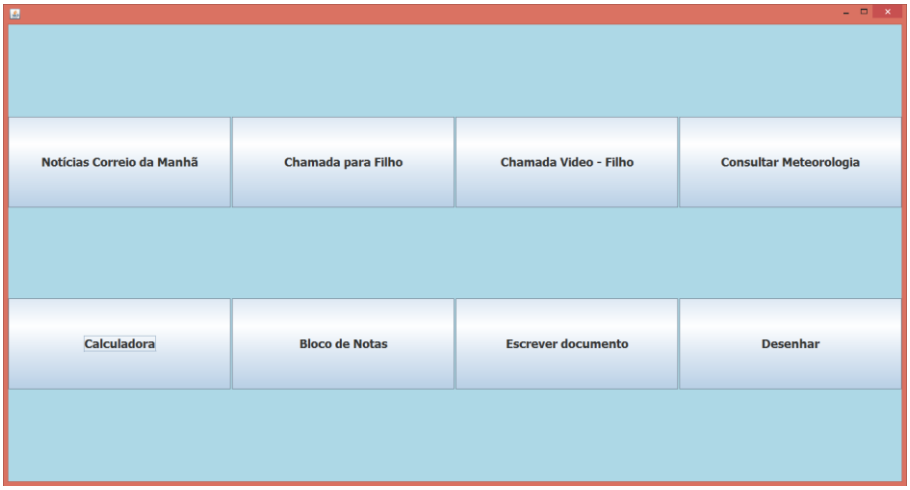


Figura 29. Home Care Application.

Assim sendo, de entre as funcionalidades aqui disponibilizadas, destacam-se as seguintes. Em primeiro lugar deu-se destaque à tarefa de consultar as mais diversas notícias, recorrendo à consulta de um *site* para o efeito (Correio da Manhã) através do *browser*. Outra tarefa que considerou-se pertinente manter nesta aplicação foi a realização de chamadas para uma determinada pessoa (predefinida), quer de voz quer videochamadas, através do *Skype*, isto porque é uma das aplicações mais adotadas pela maioria dos utilizadores, e para estes em particular, pode se revelar muito útil. Outra das possíveis utilizações dadas ao computador que decidimos aqui dar suporte é a consulta do estado do tempo e previsões associadas. Nesta tarefa, optou-se por dar foco à consulta sobre uma localidade específica, esta que deve ser definida no respetivo modelo de tarefas. Além destas, decidiu-se adicionar ainda a Calculadora, por ser simples de utilizar e porque pode ser sempre necessária, o Bloco de Notas, que pode servir para o utilizador treinar as suas capacidades de escrita, e ainda para o caso de este apenas querer um pouco de lazer, pode desenhar através do *Paint*.

Sendo esta interface apenas um caso de estudo, cujo propósito global é dar suporte à utilização de sistemas interativos, é livre de ser modificada e complementada por qualquer desenvolvedor.

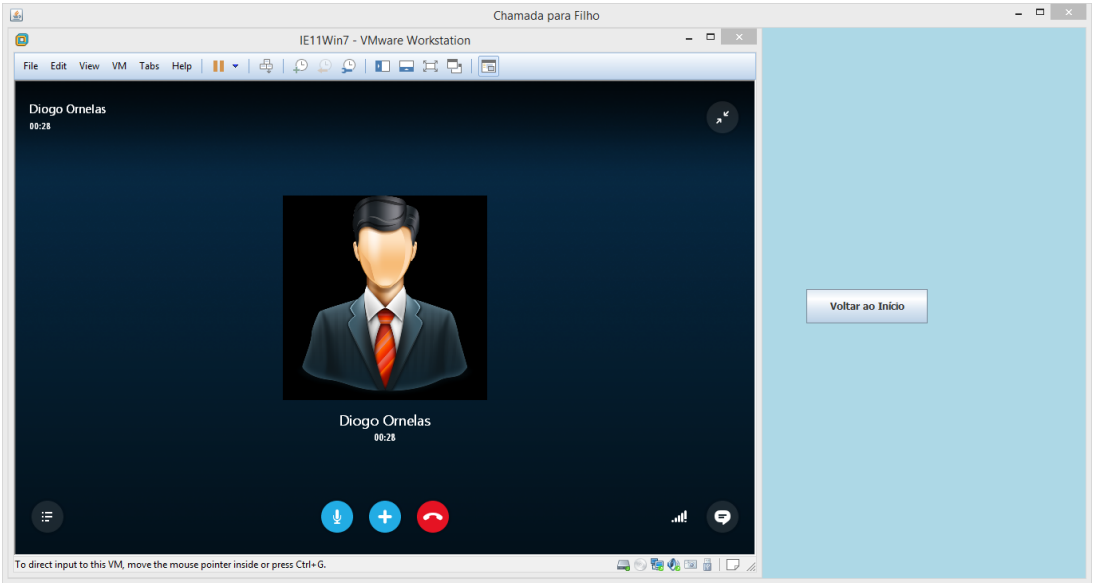


Figura 30. Chamada *Skype* em execução "dentro" da nova *GUI* simplificada.

Na Figura 30 é mostrada a realização de uma chamada *Skype* para ilustrar os benefícios da abordagem. Ao respetivo botão, na nova *GUI*, é associada uma *script* que apenas faz uso de uma aplicação (*Skype*) para executar a tarefa. Assim, o utilizador apenas tem de premir esse botão e todos os passos são automaticamente realizados via execução da *script*, na máquina virtual. O *Skype* é depois mostrado "dentro" da nova *GUI*, apenas quando a chamada é estabelecida, sendo ocultados todos os passos intermediários.

Na imagem seguinte (Figura 31) é apresentado um outro exemplo, mostrando o estado da *Home Care Application* após o utilizador clicar para executar a tarefa de consultar a meteorologia local (neste caso, o Funchal).

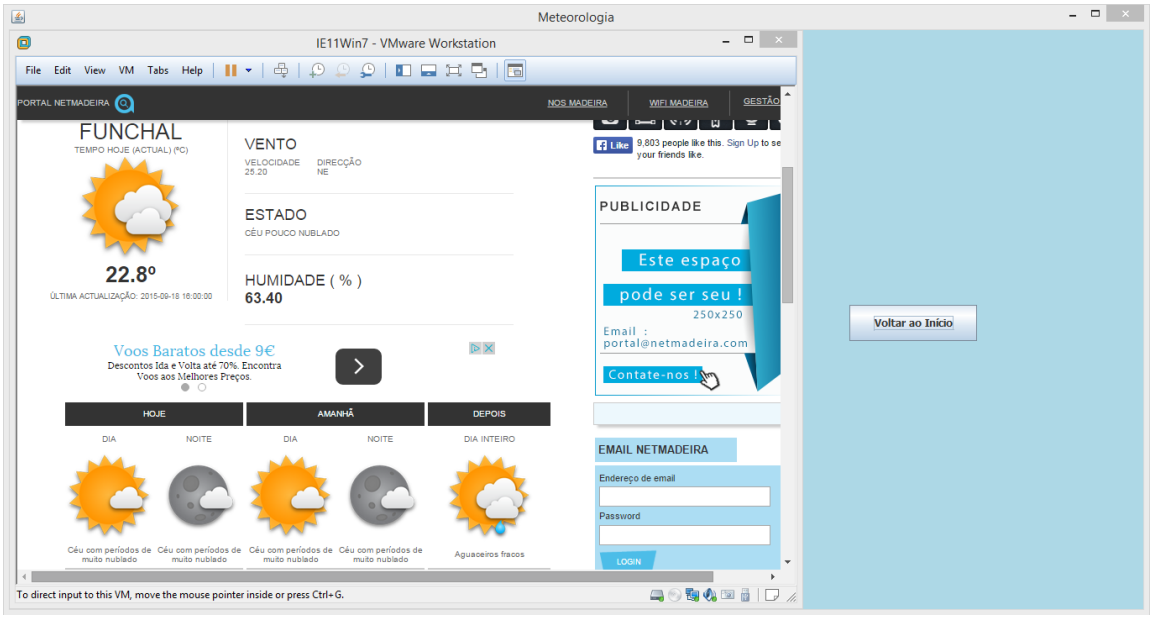


Figura 31. Consulta da meteorologia (Funchal) através da *Home Care Application.*

5 - Avaliação

Dada a natureza da abordagem aqui desenvolvida, e os respetivos casos de estudo levados a cabo com o intuito de comprovar a sua mais-valia e utilidade no suporte à utilização de sistemas interativos, foi realizado um estudo preliminar. Este recaiu especialmente sobre a avaliação das diferentes aplicações apresentadas no capítulo anterior. Para tal, e como essas incidiam sobre diferentes vertentes, foi especificado o público-alvo para cada caso bem como a sequência de procedimentos a seguir nos testes efetuados com cada uma das aplicações.

De forma a retirar algumas ilações relativamente ao pequeno estudo efetuado, são apresentados os principais resultados obtidos através da realização dos testes de avaliação de usabilidade. Entre esses destacam-se os gráficos representativos das respostas dadas pelos utilizadores a algumas das questões mais pertinentes, no preenchimento dos questionários, nomeadamente relacionadas com a eficácia, poupança de tempo, facilidade, entre outras garantias oferecidas pelo uso da abordagem. Questionários esses, que abordam, no essencial, quatro aspetos (tal como é assente no questionário padrão USE [74]): caracterização pessoal, utilidade, facilidade de utilização e satisfação do utilizador. A maioria das questões são apresentadas sob a forma de uma escala de *Likert* de cinco pontos, desde o zero (discordo totalmente) até ao cinco (concordo plenamente). Adicionalmente, há três questões (opcionais) de resposta livre/aberta, relacionadas com os pontos fortes e fracos da aplicação e com a possibilidade de fazerem quaisquer outras observações que pretendam.

Assim sendo, neste capítulo é apresentada, em primeiro lugar, a especificação dos testes de usabilidade, seguida de algumas informações úteis a eles associadas, tais como os questionários entregues aos utilizadores após a realização dos testes. Após cada especificação, são demonstrados e analisados os principais resultados alcançados nos respetivos testes. Para uma análise mais objetiva dos dados recolhidos, são apresentados os valores das médias e modas relativamente aos principais fatores em consideração.

5.1 - Testes de Avaliação de Usabilidade

A realização dos testes de usabilidade foi a derradeira etapa de todo o processo levado a cabo. Apesar de o foco estar principalmente no desenvolvimento da abordagem, a qual permite automatizar o uso de sistemas interativas, tendo por base informações presentes nos modelos de tarefas enriquecidos e em cenários a partir daí construídos, não deixa de ser relevante perceber a sua aplicação prática. Neste âmbito, a partir da abordagem implementada, foram criadas aplicações com objetivos distintos, sobre as quais foram então realizados os testes de usabilidade. Desta forma, apresenta-se de seguida o processo de avaliação levado a cabo.

5.1.1 - Sistema de Ajuda

Começando pelo Sistema de Ajuda, este foi exemplificado através de dois casos de estudo – *Word Support System* e *Blender Support System*. Definiu-se que este tipo de aplicação é destinado a utilizadores com um conhecimento informático satisfatório, nomeadamente no âmbito das aplicações em questão (*Word* e *Blender*, neste caso em particular), podendo também vir a ser útil para aqueles que pretendam ampliar os seus conhecimentos numa determinada aplicação. Tal como já foi referido, este sistema além de disponibilizar ajuda na realização de um

conjunto de tarefas, as quais poderão não ser tão óbvias ou tão simples para alguns utilizadores, tem a vantagem de também as executar pelo utilizador.

O objetivo aqui é verificar como lidam os utilizadores com esta forma de ajuda distinta da maioria que existe hoje em dia, sendo, do nosso ponto de vista, mais intuitiva, apelativa e também mais objetiva. E consequentemente, como já foi referido, perceber até que ponto é que estamos a facilitar a utilização de um Sistema Interativo.

Para a realização dos testes de usabilidade sobre estas aplicações, foram escolhidas as seguintes tarefas: no *Word*, guardar o documento no formato PDF, adicionar numeração de páginas personalizada e adicionar bordas a várias partes do documento; no *Blender*, alterar o tipo de vista (para frontal), agrupar dois objetos usando a função "Join", guardar o trabalho num ficheiro BLEND e aplicar a um *path* criado, a forma circular.

Explicada a lógica de funcionamento da aplicação, bem como identificados os objetivos e o público-alvo, passemos agora à especificação relativa aos testes de usabilidade. Importa dizer que estes não se diferenciaram muito uns dos outros. Consistiu no seguinte: estando já o utilizador em contacto com um computador disponibilizado, pediu-se que iniciasse o *Blender* (ou o *Word*), e de seguida, o respetivo Sistema de Ajuda. Imediatamente o conteúdo visível no ecrã é automaticamente reajustado, de forma a mostrar apenas as aplicações relevantes, ficando estas lado a lado (Figura 28). Aí foi pedida a realização, consecutiva, das várias tarefas.

No que à avaliação diz respeito, apesar de não muito exaustiva, permitiu chegar a algumas conclusões, que por sinal, estão parcialmente de acordo com as pretensões definidas inicialmente. Assim sendo, após a especificação dos testes retratam-se os principais dados obtidos a partir dos mesmos, tanto no *Word Support System* como no *Blender Support System*.

5.1.1.1 - Word Support System

Exemplificando com uma das tarefas realizadas durante os testes, distingue-se os passos necessários à conclusão da mesma, quer utilizando o Sistema de Ajuda, quer usando exclusivamente a aplicação original, o Word. A tarefa consiste em guardar o documento no formato PDF. Numa utilização normal, o utilizador deve proceder da seguinte forma: Clicar em "Ficheiro" -> Clicar em "Guardar Como" -> Clicar em "Procurar" -> Selecionar a pasta onde quer guardar -> No campo "Guardar com o tipo", selecionar "PDF" -> Dar um nome ao documento -> Clicar "Guardar". Agora, recorrendo à abordagem, o utilizador apenas faz o seguinte: Clicar no item "Guardar como PDF" da lista apresentada na aplicação *Word Support System* -> Selecionar a pasta onde quer guardar -> Dar um nome ao documento. Os restantes passos são automaticamente executados pelo sistema, o que por si só permite verificar a redução de carga cognitiva requerida ao utilizador. Este só tem de dar início ao processo, e depois à medida que a tarefa se desenrolar, apenas deve intervir quando o sistema assim o solicitar, não tendo de se preocupar com as opções que deve seguir nem onde as mesmas se encontram.

Na imagem que se segue (Figura 32) é mostrada parte da sequência de passos realizados para a conclusão de uma das tarefas incluídas no teste efetuado - guardar um documento, do *Word*, no formato PDF (*Portable Document File*), com a ajuda do *Word Support System*. A verde estão representadas as ações realizadas automaticamente e a laranja as que estão a cargo do utilizador. Neste caso é pedido ao utilizador para selecionar o destino onde o documento será guardado e indicar o nome do mesmo, enquanto os restantes passos são executados pela *script*.

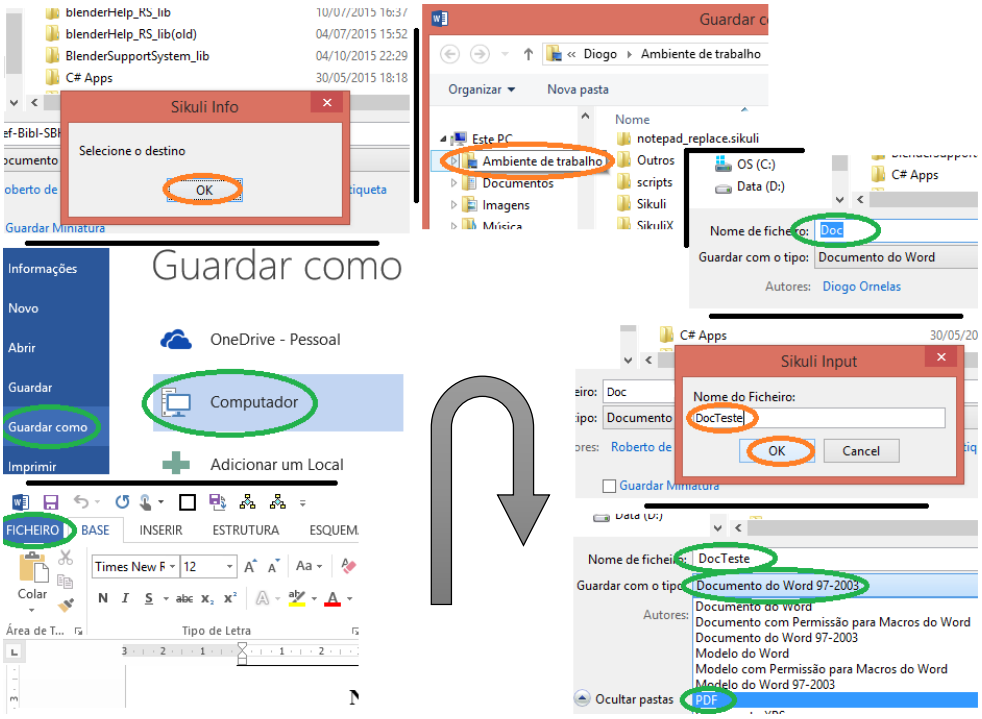


Figura 32. Representação parcial da execução de uma *script* (semiautomática).

5.1.1.1.1 - Resultados

Os participantes dos testes efetuados ao WSS são maioritariamente jovens-adultos, com idades entre os 19 e os 29 anos, possuidores de conhecimentos informáticos satisfatórios. A dimensão da amostra é de 10 elementos, sendo metade do sexo masculino e metade do sexo feminino. Relativamente aos resultados do teste, o gráfico seguinte (Figura 33) é claro em mostrar, do ponto de vista dos utilizadores, o quão útil a ferramenta se pode revelar. A média obtida neste fator foi de 4,9 valores.

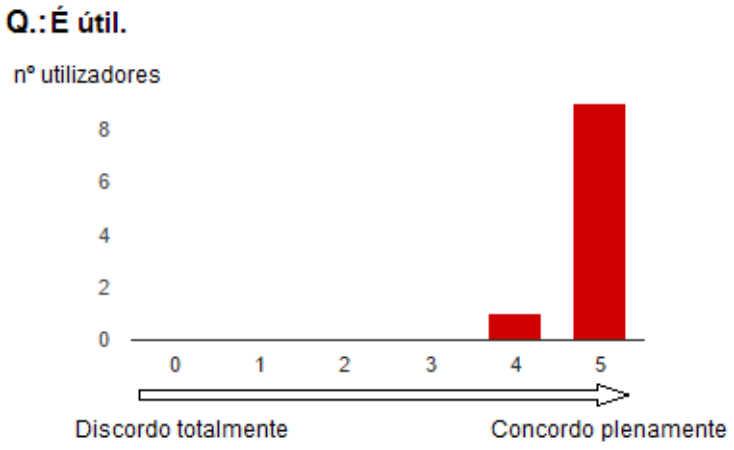


Figura 33. Respostas à questão sobre a utilidade oferecida pela aplicação.

Apesar de menos conclusivos, também não deixam de ser relevantes os dados obtidos acerca do aumento da produtividade e eficácia do utilizador no desempenho das tarefas sobre o Word. Os gráficos da Figura 34 apresentam esses mesmos resultados, notando-se uma maior inclinação para o lado "positivo" da escala, sendo que no primeiro critério obteve-se a média de 3,8 e no segundo de 4,2.

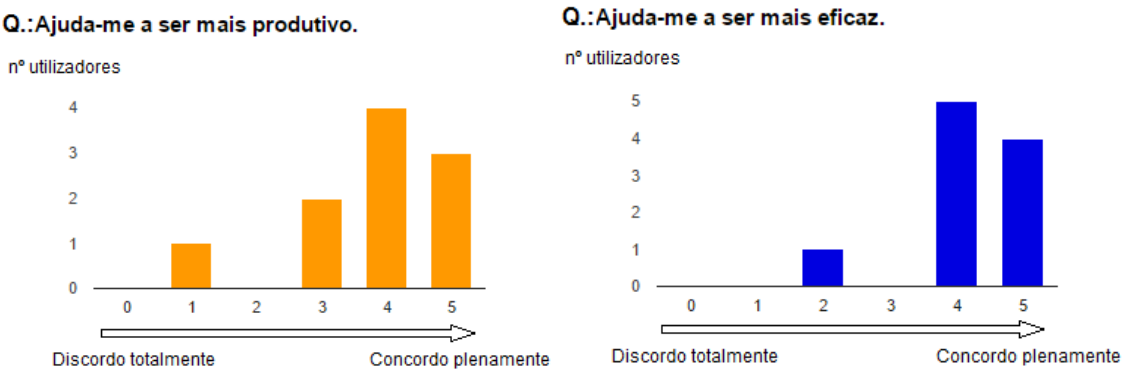


Figura 34. Respostas às questões acerca da produtividade (à esquerda) e eficácia (à direita).

Prosseguindo a análise, observa-se pelos gráficos seguintes (Figura 35), que este sistema de ajuda garante a simplificação da utilização do Word (a média obtida neste critério foi de 4,8), de forma fácil para qualquer utilizador minimamente familiarizado com o uso do computador (sendo 4,6 o valor da média).

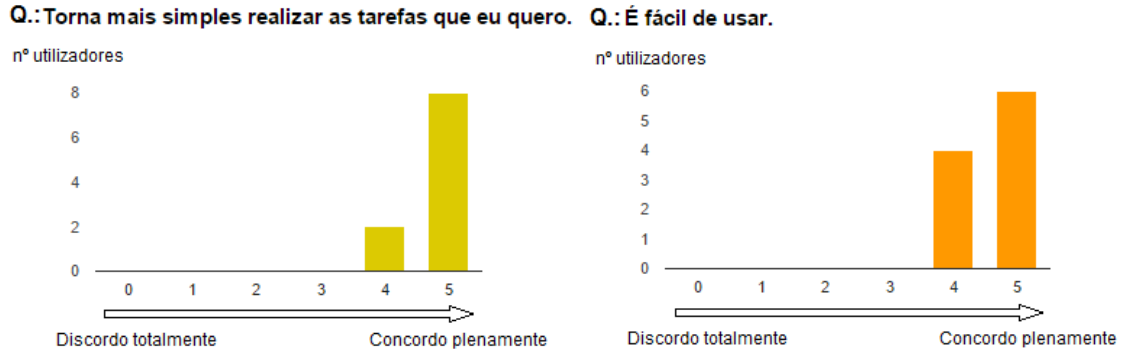


Figura 35. Respostas obtidas nas questões relativas à simplificação da realização de tarefas (à esquerda) e à facilidade de utilização do WSS (à direita).

Contudo, segundo algumas opiniões recolhidas, fica a ideia de que seria positivo tornar a ferramenta mais acessível e intuitiva, independentemente do tipo de utilizador em causa, i.e., tornar a sua utilização mais clara tanto para amadores como para os "*experts*" na área das tecnologias. Importa ainda destacar que, tal como previsto, este sistema de ajuda foi bem aceite como uma nova forma de aprendizagem, mais fácil e apelativa do que a ajuda normalmente encontrada.

Por outro lado, também é de dar valor ao facto de se ter reduzido, em cerca de 8-13 segundos, o tempo de espera entre o momento em que o utilizador clica para dar início à execução e o momento em que esta é efetivamente iniciada. Como já foi mencionado no capítulo 3.2.2 - IDE Sikuli vs RunServer, isto deveu-se à utilização do servidor *RunServer* para a execução das *scripts*.

Neste tipo de ferramenta, todos os passos executados para a concretização de uma tarefa são totalmente visíveis para o utilizador (ver exemplo da Figura 32). Isso implica que ele seja capaz de assistir a essa execução, sem cair na tentação de tentar interagir por si só, a não ser que para tal seja solicitado. O que se está a querer dizer é que, esta abordagem representa uma nova forma de interação com os sistemas interativos, completamente distinta das que o utilizador está familiarizado. Daí que, seja necessário haver um tempo inicial de adaptação, o qual não se teve oportunidade de respeitar, visto que se trata de um projeto académico (em diminuta

escala) cujos limites temporais são reduzidos. Isso fez com que se deparasse com alguns inconvenientes ao longo da realização dos testes, nomeadamente algumas reações de surpresa/desconfiança perante a visualização da execução automática das tarefas, pois alguns utilizadores não se sentiam preparados para esta "perda" do controlo do computador. Todavia, foi também por essa razão (execução visível ao utilizador) que talvez não aconteceu o mesmo do que nos testes realizados à *Home Care Application*, onde, por vezes, os utilizadores acabavam por se aborrecer, considerando que o processo levava demasiado tempo a ser executado.

5.1.1.2 - Blender Support Sytem

Ainda comparando a execução de tarefas, seguindo as duas abordagens, é exemplificado um caso no âmbito do *Blender System Support*. Este refere-se a um tipo específico de ajuda, na qual são dadas ao utilizador indicações do que este deve fazer durante a realização da tarefa, que diz respeito à opção de aplicar a um "*path*" criado, a forma cilíndrica (ver Figura 36).

Desta forma, após a inicialização de ambas as aplicações, *Blender* e *Blender System Support*, o utilizador clica na opção relativa à tarefa em causa, presente na interface da última aplicação. Aí, o processo é iniciado, questionando se o utilizador já desenhou os devidos objetos. Se sim, é mostrado um "*popup*" a indicar que o utilizador selecione o objeto "*path*". Os restantes passos são automaticamente executados. Caso contrário, surge uma janela de "*popup*" indicando que o "*path*" será criado, fazendo-o de seguida. Surge depois outro "*popup*" revelando que a "*shape*" será criada, passando à sua conceção. Por fim é igualmente pedido que utilizador selecione o objeto "*path*", de forma ao sistema poder completar a devida tarefa, aplicando a "*shape*" ao "*path*" criado.

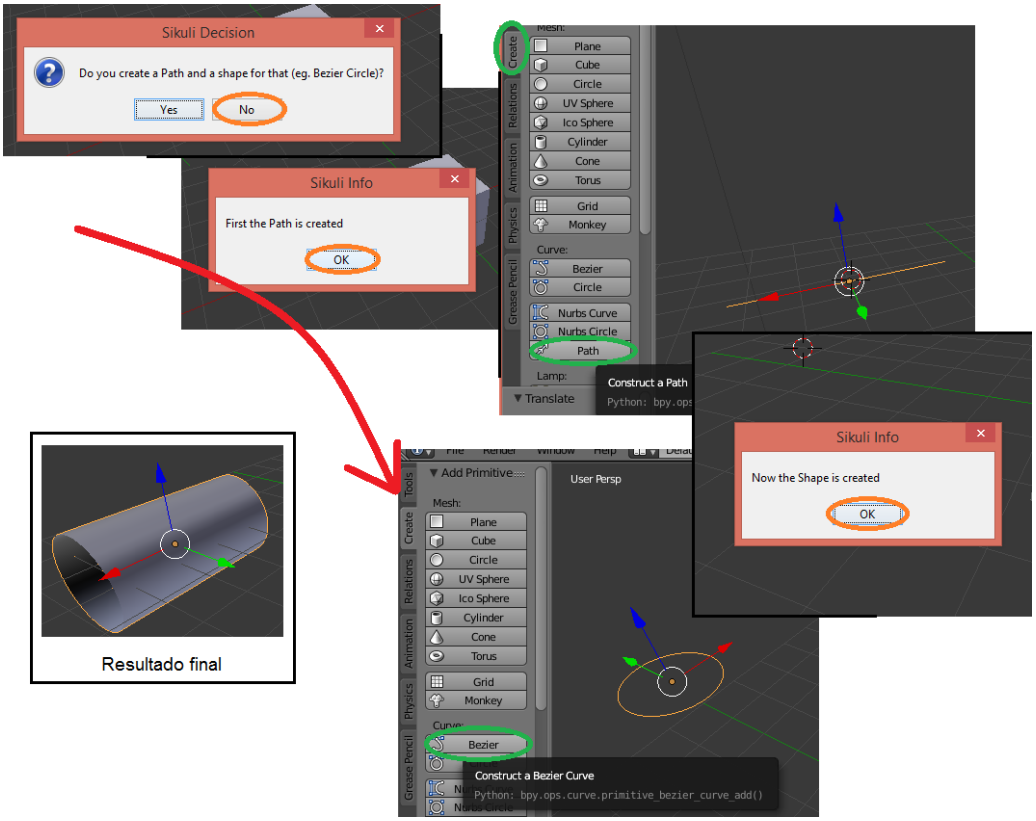


Figura 36. Parte de um cenário de execução de uma *script* (*Blender*).

No decorrer dos testes, ficou-se incumbidos de retirar o máximo de apontamentos que pudessem contribuir para a apresentação de conclusões. Entre eles, destacam-se as respostas às seguintes questões: “Conseguiram realizar a tarefa?”, “Com ou sem ajuda?”, “Como procuraram obter essa ajuda?” e "A ajuda foi clara?”. Após o término, os utilizadores responderam a um simples questionário (conforme o ANEXO 3.2 - Word/Blender Support System), capaz de fornecer algumas indicações acerca da experiência realizada, como por exemplo, a sua utilidade, simplicidade, pontos a melhorar e sugestões.

5.1.1.2.1 - Resultados

No que respeita aos testes sobre esta realizados, os inquiridos têm idades entre os 19 e os 26 anos, onde quatro pertencem ao sexo feminino e três ao sexo masculino. Por se tratarem de testes preliminares, a dimensão da amostra é reduzida (7 elementos), de entre os quais dois não conheciam a ferramenta *Blender*. Os resultados aqui conseguidos não se diferenciaram em muito do caso de estudo anterior. Assim, pode-se comprovar pelos gráficos que se seguem (Figura 37), o grau de utilidade (média de 5) e eficácia (média de 4,4) atribuído pelos participantes à aplicação em causa.

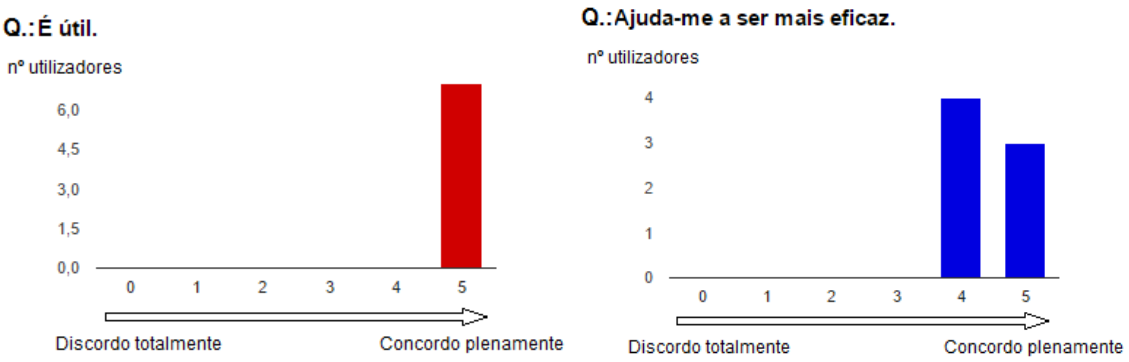


Figura 37. Respostas às questões sobre a utilidade e a eficácia do BSS.

Continuando o estudo, observa-se que em relação ao tempo que a utilização do BSS permite poupar (Figura 38), as opiniões não são totalmente unânimes, como era previsível. No entanto, os resultados não são de todo desanimadores, pois a média obtida (3,7 valores) leva a crer que se está no bom caminho.

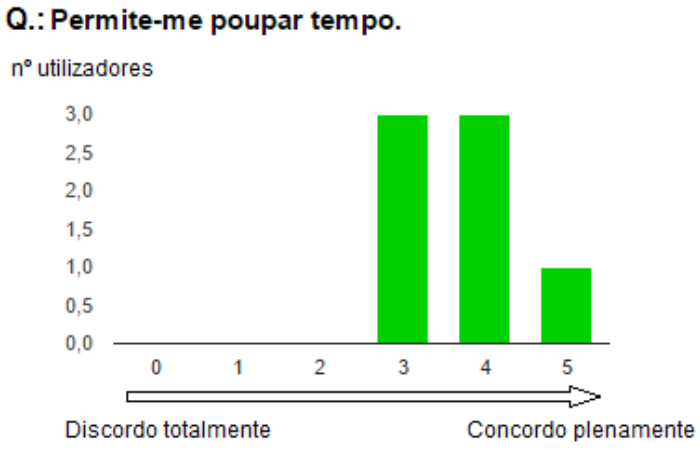


Figura 38. Respostas à questão do tempo que a aplicação permite poupar.

Pela observação do gráfico seguinte (Figura 39), atenta-se que a aplicação mostrou ser de fácil aprendizagem e posterior utilização, tal como se pretendia. Além disso, os inquiridos revelaram o sentimento de que necessitam de possuir uma aplicação deste tipo (obtendo-se uma média de 3,6), o que nos deixa com boas perspetivas futuras.

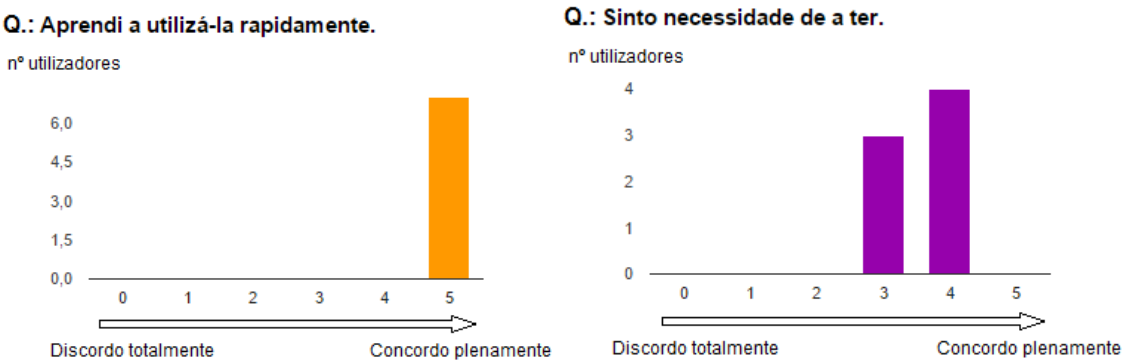


Figura 39. Respostas às questões relacionadas com a aprendizagem da utilização do BSS (à esquerda) e com a necessidade de a ter (à direita).

Por outro lado, ficou a ideia de que seria benéfico alterar a forma como as indicações (por exemplo, sobre que ação deve fazer a seguir ou o que vai ser feito pelo sistema) são apresentadas, dependendo da situação em causa. Uma alternativa seria mostrá-las através de *popups* temporários ou de *tooltips* sobre os respetivos elementos, não exigindo assim ao utilizador mais um clique adicional, como aliás acontece.

É de notar que em certos casos é permitido ao utilizador escolher o caminho a seguir, dependendo do que este for capaz de realizar por si só. É o que acontece na Figura 36, onde a tarefa que está a ser executada pode seguir dois caminhos distintos. No exemplo demonstrado o utilizador pretende que a aplicação o substitua na execução de todo o processo.

Para finalizar, pode-se afirmar que este tipo de aplicação pode ser útil aos utilizadores mais novatos, como um método de aprendizagem rápida, dando-lhes a capacidade de executarem tarefas que requeiram uma série de passos mais complexos, as quais antes eram impensáveis. Em relação aos mais experientes, pode substituí-los nos passos mais repetitivos e monótonos.

5.1.2 - Home Care Application

Passando à *Home Care Application*, destaca-se o facto de o seu público-alvo ser maioritariamente utilizadores principiantes no uso de sistemas de Tecnologias da Informação, que por não terem experienciado algo semelhante anteriormente, podem achar difícil a realização de tarefas básicas. Optou-se aqui por pessoas com mais de 50 anos, por ser a fatia da população que possui, no geral, menos conhecimentos informáticos. As tarefas escolhidas para demonstrar a aplicação e utilidade deste caso de estudo foram a consulta *online* de notícias e da meteorologia local e ainda a realização de chamadas através do *Skype*.

Continuando, esta interface foi pensada com o intuito de facilitar, aos utilizadores principiantes, a realização de qualquer tarefa, que pode envolver um ou vários sistemas independentes,

deixando-as à distância de um clique. Estas podem parecer simples para grande parte dos utilizadores, mas para o público a que se destinam revelam-se difíceis e inacessíveis.

O objetivo aqui assenta em perceber até que ponto é que, com a introdução desta interface intermediária entre o utilizador e a aplicação final, se está a facilitar a utilização de Sistemas Interativos, por parte de utilizadores inexperientes, e ao mesmo tempo incentivá-los para tal.

Passando aos testes propriamente ditos, o utilizador foi colocado frente a um computador, onde o processo estava pronto a ser iniciado. Isto quer dizer que a máquina virtual estava operacional e a *Home Care Application* devidamente inicializada. Assim, foi pedido que este realizasse uma chamada (de voz) para uma dada pessoa ("FilhoA" neste caso) (ou outra das tarefas já mencionadas acima, que tenham sido automatizadas). Para tal atendeu-se a chamada num outro dispositivo, para que o utilizador se pudesse sentir numa utilização real. Depois desligou-se a chamada e foi dito ao utilizador para voltar ao "menu" inicial da interface.

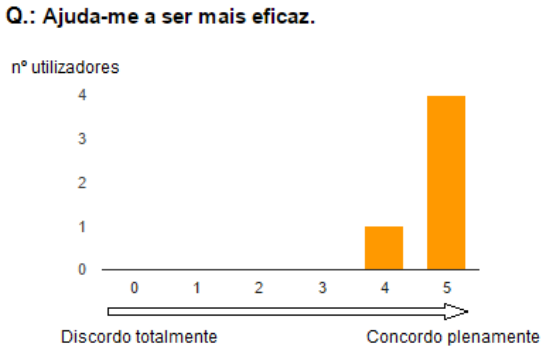
À medida que os testes se desenrolavam, foram tirados alguns apontamentos de forma a recolher informações relevantes (problemas encontrados, questões colocadas, etc.).

Após o término de cada teste, foi pedido aos utilizadores que respondessem a um pequeno questionário (ver ANEXO 3.3 - Sistema de Simplificação – Home Care Application), que permitisse elucidar do seu grau de satisfação relativamente ao uso da interface, dando-lhes também a possibilidade de oferecerem quaisquer sugestões que pudessem contribuir para um aperfeiçoamento da abordagem em estudo.

5.1.2.1 - Resultados

A amostra aqui considerada foi relativamente curta, pois não se tratou de uma avaliação exaustiva, mas apenas de um método adicional capaz de dar maior fiabilidade às conclusões a ser retiradas. Os intervenientes no teste encontram-se na casa dos 50-65 anos. Desses, a grande maioria possui poucos conhecimentos acerca das TI, sendo que apenas um deles revelou não ter qualquer conhecimento na área. Assim, a avaliação conduzida pretendeu obter feedback sobre a eficácia da solução no apoio à utilização de sistemas interativos por iniciantes. Prosseguindo para a dita análise, observou-se que os principais objetivos definidos inicialmente estavam a ser cumpridos, i.e. a simplificação da interação entre o utilizador e o computador e a redução da carga cognitiva necessária para o efeito. Através dos questionários, verificou-se, da parte dos participantes, uma reação positiva à aplicação, obtendo na maioria dos critérios apresentados uma moda de cinco valores.

Assim, e como é demonstrado pelos gráficos das figuras seguintes (Figura 40 e Figura 41), os utilizadores corroboraram a eficácia, simplicidade (médias iguais de 4,8) e a utilidade (média de 5), resultantes da aplicação da abordagem neste caso em particular.



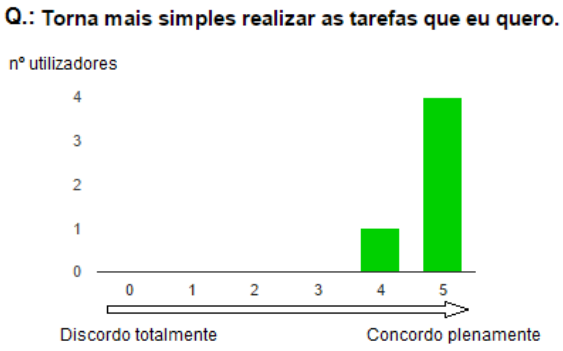


Figura 40. Respostas às questões relativas à eficácia da *Home Care Application* (à esquerda) e à simplicidade da realização das tarefas (à direita).

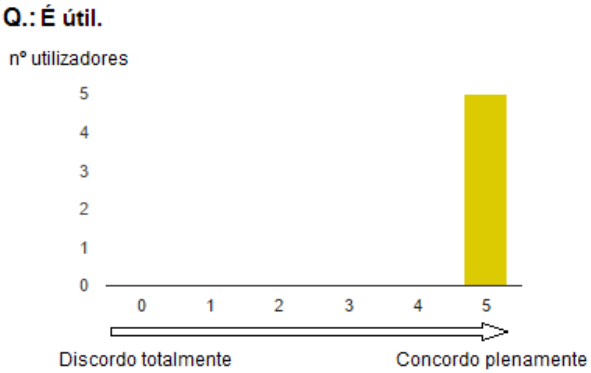


Figura 41. Respostas à questão sobre a utilidade da aplicação.

Por outro lado há um fator que apesar de ter sido aperfeiçoado com o avançar do desenvolvimento da abordagem, nomeadamente com a utilização do servidor *RunServer* para a execução das *scripts*, ainda causa alguma apreensão tanto da nossa parte como da parte dos utilizadores. Segundo a Figura 42 é percetível que se fala do fator "tempo".

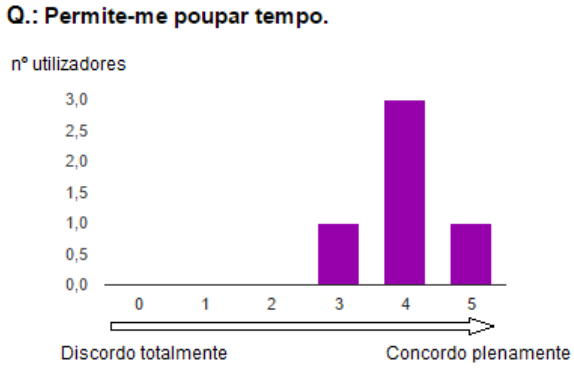


Figura 42. Respostas à questão da poupança de tempo oferecida.

No entanto, nesta interface em particular, a execução intermédia das tarefas está a ser escondida do utilizador (é realizada em segundo plano, na máquina virtual), mostrando apenas o resultado final. Durante esse tempo de processamento, os utilizadores apenas tinham de aguardar, não estando a fazer qualquer outra coisa. Assim, é natural que o tempo de executar uma tarefa possa parecer muito (dependendo da tarefa em questão), mas isso é algo para o qual já se estava ciente. De forma a minimizar, tanto quanto possível, este problema, será adicionado

à interface um indicador de progresso animado com informação visual, para os utilizadores saberem que o sistema está em processamento.

Por fim, importa realçar que a abordagem acaba por ser relevante para todos os tipos de utilizadores. Aos inexperientes, permite executar tarefas complexas sem necessitarem de qualquer conhecimento dos passos a seguir e, tal como foi referido por um dos participantes no questionário, pode ser utilizada como forma de incentivar esta fatia da população a usufruir mais consistentemente das potencialidades de um computador. Relativamente aos especialistas, torna-os mais eficientes, pois permite-lhes que realizem outras atividades enquanto aguardam a execução automática de uma tarefa.

6 - Conclusões e Trabalho Futuro

Para finalizar, é de extrema relevância dar a conhecer o conjunto de ilações a que foi possível se chegar, a partir de todo o trabalho descrito ao longo deste relatório. Assim, importa não só resumir todo o processo, como discutir os pontos de maior saliência, tanto positivos como negativos, e ainda apresentar as conclusões alcançadas.

Os sistemas interativos têm uma presença cada vez mais vincada no nosso quotidiano. E além disso, assiste-se a uma crescente variedade dos mesmos, onde se notam diferenças significativas entre si, quer a nível das *GUIs*, do seu modo de funcionamento, dos estilos de interação possíveis, entre outros aspetos. A utilização desses sistemas revela-se frequentemente difícil, em particular para utilizadores iniciantes. Os avanços que se têm verificado ao nível da usabilidade, *design* e sistemas de ajuda tentam dissipar essa questão, no entanto, permanecem ainda alguns desafios. Um deles é a necessidade dos utilizadores se adaptarem aos sistemas propostos para satisfazer as suas necessidades. Geralmente, são eles que têm de aprender como interagir com o sistema ou saber quais são os passos a seguir para completar uma dada tarefa.

Este trabalho foi desenvolvido nesse sentido, apresentando uma abordagem que serve de base ao propósito da adaptação dos sistemas interativos às necessidades dos utilizadores. Esta centra-se na automatização de tarefas, recorrendo para tal a modelos de tarefas (enriquecidos) e à computação orientada por imagem. O objetivo último é o de fornecer uma *GUI* simplificada de qualquer sistema interativo, sem a necessidade de aceder ao seu código-fonte e independente do sistema operativo. Através desta, o utilizador é auxiliado na execução de uma qualquer tarefa, a qual até pode envolver mais do que uma aplicação nativa. Assim, é-lhe retirada carga cognitiva, pois este não necessita de conhecer detalhadamente a *GUI* da aplicação final. Por exemplo, o utilizador não precisa de saber que botão tem que ser pressionado, que caixa de texto deve ser preenchida nem qual a sequência de passos a realizar para completar a tarefa pretendida.

Posto isto, o presente capítulo é responsável por apresentar, em primeiro lugar, uma discussão dos pontos que se acharam pertinentes, seguida das conclusões finais que foram possíveis obter e por último, o trabalho que está previsto ser desenvolvido no futuro próximo como forma de complementar o que foi até aqui realizado.

6.1 - Discussão

É de realçar que, no âmbito da abordagem apresentada, a escolha das tarefas (e/ou partes delas) a ser automatizadas, e consequentemente integradas na nova *GUI*, bem como a seleção do nível de automatização adequado, fica à responsabilidade do desenvolvedor, tentando sempre elevar ao máximo o desempenho dos possíveis utilizadores finais. Destaca-se o facto de as tarefas ficarem apenas à distância de um clique, tornando a interação mais simples e objetiva e evitando parte dos obstáculos enfrentados no uso direto da aplicação/*GUI* original. A exceção reside nos casos em que as tarefas requeiram certas ações intermédias por parte do utilizador.

No que diz respeito ao *User Support System* – aplicação que disponibiliza o mecanismo de geração automática de *scripts* – há um aspeto que pode influenciar o emprego da abordagem, que é a obtenção do modelo de tarefas. Isto é, se quem pretende aplicá-la são os

desenvolvedores do sistema a automatizar, têm geralmente acesso aos modelos de tarefas (desenvolvidos nas fases iniciais do desenvolvimento). Assim sendo, só têm de enriquecê-los seguindo as regras já apresentadas, que se acredita serem bastante claras e objetivas. Se por outro lado, não houver nenhum modelo desenvolvido disponível, têm de ser criados, de raiz, modelos de tarefas que reflitam o comportamento atual do sistema. Desta forma o processo será obviamente mais demorado.

Prosseguindo para o Sistema de Ajuda, há um ponto onde deve ser dedicada atenção redobrada para um melhor aproveitamento da dita abordagem. Entre os inconvenientes apontados pelos utilizadores, destaca-se a necessidade de esperar que o sistema realize a tarefa pedida, perdendo temporariamente o controlo do dispositivo, e ainda nos casos em que é requerida a sua intervenção nalgum passo intermédio, por vezes o utilizador é "levado" a realizar mais ações do que era suposto (um exemplo concreto verificou-se quando, na opção de adicionar bordas ao documento, era pedido que o utilizador selecionasse o estilo e/ou a cor da linha de contorno, e este, talvez por instinto, além disso acabava por clicar no botão "Ok", aplicando assim as bordas). No entanto, estas são situações que, na nossa opinião, se verificam principalmente por esta ser uma abordagem nova e completamente distinta, comparativamente à utilização normal a que os utilizadores estão habituados. Posto isto, o que é necessário é dar o devido tempo de adaptação e consequente mudança de mentalidades, que só é possível à medida que mais sistemas deste género sejam introduzidos no seu quotidiano.

Na vertente do Sistema de Simplificação, a limitação imposta é que adequa-se apenas à integração de tarefas onde não sejam necessárias interações intermédias por parte do utilizador. Caso contrário, a utilização da máquina virtual deve ser excluída ou as interações intermédias exibidas ao utilizador, o que neste contexto deixava de fazer sentido.

Finalmente, é de louvar o trabalho considerável que já foi desenvolvido, tanto na área da visão computacional como na computação orientada a imagens, no entanto subsistem ainda desafios importantes [35]. Tais como, a capacidade limitada das *scripts Sikuli* na automatização da interação com certos tipos de *widgets* (e.g. *scrollbars*, *sliders* ou *spinners*), a resolução de ecrã e os temas personalizados do Sistema Operativo, que podem afetar o algoritmo de visão computacional do *Sikuli*. Na prática, isto pode significar que as imagens incluídas numa *script* poderiam não funcionar em diferentes personalizações do mesmo Sistema Operativo, o que pode gerar complicações no uso futuro da abordagem.

6.2 - Conclusões

Após todo o processo envolvente, desde a idealização da abordagem até aos resultados dos testes de avaliação preliminar, seguem-se as ideias finais a que foi possível chegar.

Na perspetiva do Sistema de Ajuda, foram implementados dois casos de estudo, um relativo ao *Microsoft Word* e outro ao *Blender*, tendo como base a mesma ideia - servir de auxílio ao utilizador no desempenho de tarefas, que correspondam especialmente a funcionalidades avançadas das aplicações alvo ou por outro lado, a tarefas comuns mas aborrecidas para o utilizador devido à sua elevada repetição.

Perante os resultados obtidos, é clara a utilidade que este tipo de aplicação oferece, essencialmente para os utilizadores com poucos conhecimentos/prática no uso de ferramentas informáticas. É efetivo o facto de que torna mais fácil a conclusão de diversas tarefas sobre as aplicações alvo (*Word* e *Blender*, neste caso). É ainda de destacar o elemento "novidade" presente nesta abordagem, como, aliás, já foi referido, e ainda o facto de servir como um método de aprendizagem, mais apelativo, sendo capaz de, à medida que mostra os passos a seguir, executar a própria tarefa.

No que respeita ao último caso de estudo - *Home Care Application* - tem-se que este visa criar uma camada de abstração intermédia (da *UI*), capaz de tornar a interação com os sistemas interativos um processo mais fácil e atrativo, principalmente para novos utilizadores ou iniciantes. Assim, através da ferramenta é possível a adaptação e integração de sistemas interativos independentes, de acordo com as necessidades específicas do utilizador. Esta permite aos desenvolvedores integrarem várias funcionalidades de diferentes aplicações, independentemente da linguagem de programação ou sistema operativo (OS), numa nova *GUI* de aplicação. Tudo isto por meio da automatização. Aqui, a execução das tarefas (nas aplicações originais) é escondida do utilizador, tendo lugar numa máquina virtual. O utilizador só se preocupa em desencadear a tarefa, interagindo com a nova *GUI* simplificada, o que resulta numa potencial redução da sobrecarga cognitiva, pois os utilizadores não têm necessidade de saber a sequência de passos a completar para executar a tarefa.

Este tipo de aplicação possibilita ainda que os desenvolvedores se foquem apenas na automatização de tarefas que sejam de real interesse para os utilizadores, integrando-as numa *GUI* centralizada. Isto é vantajoso pois muitas das aplicações que são correntemente utilizadas possuem demasiadas funcionalidades, muitas das quais irrelevantes para estes utilizadores, e que acabam por tornar todo o processo de interação mais complexo.

Prosseguindo, é de extrema importância referir que com a utilização do servidor *RunServer*, conseguiu-se minimizar significativamente o problema do atraso/*delay* verificado entre o lançamento e o início efetivo da execução das *scripts Sikuli*. Este inicialmente estava em torno dos 10 a 15 segundos. Agora, além de alguns (1-3) segundos na primeira inicialização do servidor, o atraso em cada execução das *scripts* foi reduzido a valores insignificantes (menos de 1 segundo). No entanto, também devido ao *RunServer* não estar ainda numa fase de grande maturidade, tendo sido disponibilizado há poucos meses, por vezes registam-se falhas de comunicação que impedem, por momentos, o correto funcionamento das aplicações desenvolvidas, nomeadamente do processo de execução das *scripts*. Porém, com o tempo que se conseguiu poupar no momento de utilização das aplicações, ao recorrer a este método, comparativamente à execução através da linha de comandos, achou-se por bem seguir este caminho, mesmo sabendo do risco associado.

Apesar dos pontos menos positivos, acima mencionados, acredita-se na utilidade que a abordagem oferece, permitindo a adição de novas funções/tarefas e/ou a integração de vários sistemas independentes, tornando a utilização de sistemas interativos mais eficiente. Corroborando esta ideia, estão os resultados apresentados acima relativamente aos testes preliminares, efetuados com utilizadores finais, às aplicações desenvolvidas. Assim, afiança-se que esta abordagem de suporte ao uso de sistemas interativos poderá conduzir a importantes

contribuições para este campo, nomeadamente a alterações significativas no modo de utilização dos mesmos.

Como é possível constatar através do presente estudo, a abordagem aqui desenvolvida pode ser aplicada nos mais variados contextos. Aqui foram demonstrados apenas alguns dos possíveis casos de utilização, por sinal, os que se consideraram mais pertinentes.

Para terminar, como se pensa que esta é uma área que necessita de evoluir, e que o projeto aqui desenvolvido poderá ser um excelente ponto de partida para tal, foram submetidos três *papers*, um à *ACM CHI* (rejeitado - borderline), outro à *ACM CHIuXID* (aceite) e em breve será submetido outro à *ACM EICS* 2016. Cada um deles foca diferentes especificidades do trabalho, onde constam os principais resultados alcançados, de forma a dar maior notoriedade ao projeto e também aumentar as possibilidades de continuidade.

6.3 - Trabalho Futuro

As aplicações desenvolvidas para a aplicação da abordagem, nomeadamente, a *Home Care Application*, são apenas exemplos com o intuito de ilustrar a abordagem com tarefas do interesse do seu público-alvo. No futuro, exemplos mais interessantes e envolvendo um maior grau de complexidade serão desenvolvidos. Em particular, pretende-se aperfeiçoar a referida ferramenta, utilizando-a com exemplos mais complexos, e com o desenvolvimento de novas *GUIs* capazes de desempenhar funcionalidades através da execução automática de tarefas em diferentes aplicações (por exemplo, a tarefa de verificar uma agenda antes de reservar um voo). Uma das melhorias que se pretende aplicar durante a execução das *scripts* é a introdução da funcionalidade de *undo*. Esta seria útil em casos onde o utilizador apenas quer aprender a realizar uma dada tarefa, usando a ferramenta de ajuda, não pretendendo alterar o estado do sistema. Apesar da maioria dos sistemas possuir essa funcionalidade, ela poderia se revelar muito útil quando a ajuda incide sobre tarefas envolvendo vários sistemas independentes.

Ações de maior escala com o intuito de avaliar exaustivamente a verdadeira utilidade, facilidade de utilização, eficácia e aceitação da mesma, quer pelos desenvolvedores quer pelos utilizadores finais, também estão planeadas como trabalho futuro. Aqui incluem-se os testes a realizar tanto com a aplicação USS, que permite a geração automática de scripts, como com a aplicação do processo necessário para tal, isto é, o enriquecimento de modelos de tarefas e a criação de cenários de simulação. Tal como nos testes já realizados, será entregue aos participantes um pequeno questionário, o qual se encontra em anexo (ANEXO 3.1 - User Support System / Abordagem).

No que respeita aos resultados dos testes de avaliação de usabilidade, é percetível que os mesmos foram apenas apresentados do ponto de vista do utilizador, ou seja, tendo por base as respostas dadas pelos mesmos nos questionários preenchidos. De forma a complementar esta avaliação, pretende-se, no futuro próximo, colocar os mesmos utilizadores a realizar as mesmas tarefas que realizaram nos testes em que participaram, mas desta vez sem recorrer a qualquer sistema de suporte. Isto é, pretende-se verificar como se comportam perante a execução normal dessas tarefas e comparar com os testes que foram até aqui concretizados. Através deste termo

de comparação poder-se-á obter resultados mais conclusivos, sendo possível perceber melhor o verdadeiro impacto das aplicações de suporte desenvolvidas.

No que diz respeito às *scripts* em si, foi investigada uma funcionalidade que elas poderiam disponibilizar, através de uma extensão do *Sikuli*, denominada *Sikuli Guide*. Esta fornece uma maneira ainda mais inovadora de criar visitas guiadas ou tutoriais para qualquer aplicação interativa. A novidade é que o conteúdo dos tutoriais pode ser exibido diretamente sobre a interface na qual a *script* se processa, ao invés de ser apresentado através de um vídeo ou de um guia "passo-a-passo" complementado com *screenshots* da *GUI*, disponibilizado numa página web, por exemplo [75]. Exemplificando, supondo que se quer criar uma visita guiada da página de documentação do *Sikuli*, e que se pretende começar por chamar a atenção para o logotipo aí apresentado. Assim, recorrendo a algumas funções fornecidas por esta extensão, poderíamos escrever a seguinte script (à esquerda da Figura 43):

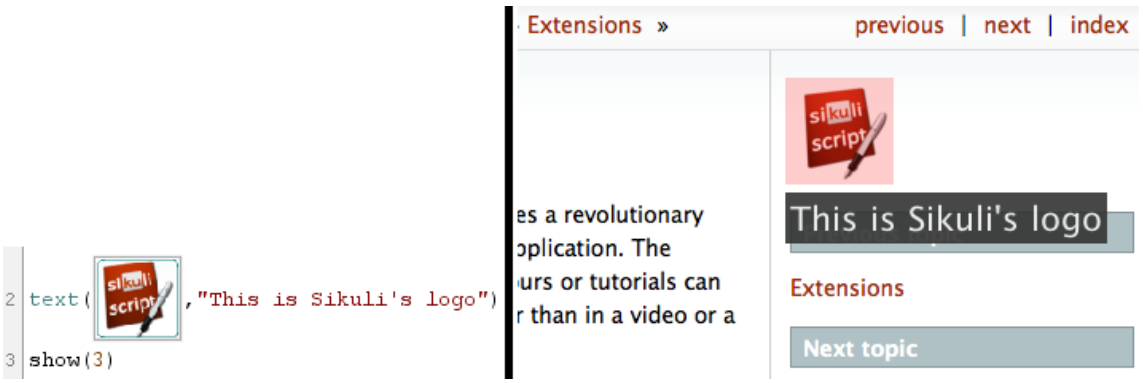


Figura 43. À esquerda, fragmento da *script*. À direita, resultado da execução da *script*.

Na metade direita da Figura 43 é mostrado o resultado aquando da execução desse fragmento da script, onde é possível ver o logotipo realçado (a vermelho), com o respetivo texto exibido por baixo. No entanto como esta opção está a ser alvo de novos desenvolvimentos, nas versões mais atuais da ferramenta *Sikuli* não se encontra disponível, o que é de lamentar.

Referências

[1] Jakob Nielsen and Rolf Molich. “Heuristic evaluation of user interfaces”. In CHI '90 Proceedings, pages 249-256, New York, April 1990. ACM Press.

[2] Paternò, Fabio. “Model-based Design and Evaluation of Interactive Applications.” Springer Science & Business Media, 2012

[3] Silva, JL, and Silva JC. "Methodology to support the use of interactive systems." *Information Systems and Technologies (CISTI), 2014 9th Iberian Conference on* 18 Jun. 2014: 1-6.

[4] Martinie, Célia et al. "Task-Model Based Assessment Of Automation Levels: Application To Space Ground Segments." *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on* 9 Oct. 2011: 3267-3273.

[5] Parasuraman, Raja, Thomas B Sheridan, and Christopher D Wickens. "A Model For Types and Levels of Human Interaction with Automation." *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 30.3 (2000): 286-297.

[6] Chang, Tsung-Hsiang, Tom Yeh, and Robert C Miller. "GUI Testing Using Computer Vision." *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* 10 Apr. 2010: 1535-1544.

[7] Silva, JC, and Silva JL. "A Methodology for GUI Layer Redefinition through Virtualization and Computer Vision." *Computational Science and Its Applications (ICCSA), 2014 14th International Conference on* 30 Jun. 2014: 58-63.

[8] Kaber, David B, and Mica R Endsley. "The effects of level of automation and adaptive automation on human performance, situation awareness and workload in a dynamic control task." *Theoretical Issues in Ergonomics Science* 5.2 (2004): 113-153.

[9] Billings, C. E. “Human-centred aircraft automation: A concept and guidelines” (NASA Tech. Memo. No. 103885) (Moffet Field, CA: NASA-Ames Research Center), 1991

[10] Bolton, Matthew L, and Samaneh Ebrahimi. "An Approach to Generating Human-Computer Interfaces from Task Models." *2014 AAAI Spring Symposium Series* 22 Mar. 2014.

[11] Paternò, Fabio. "ConcurTaskTrees: an engineered approach to model-based design of interactive systems." *The Handbook of Analysis for Human-Computer Interaction, Lawrence Erlbaum Associates* (2002): 483-500.

[12] Kourousias, George, and Silvio Bonfiglio. "Picture-Driven Computing In Assistive Technology And Accessibility Design." *1st International ÆGIS Conference* on 7-8 Oct. 2010: 210-218.

[13] Forsyth, David A. and Ponce J. "Computer Vision: A Modern Approach” Prentice Hall, USA, 2002.

[14] Sheridan, Thomas B, and William L Verplank. "Human and Computer Control of Undersea Teleoperators." (Cambridge, MA: MIT Man–Machine Laboratory) 15 Jul. 1978.

[15] Save, Luca, Beatrice Feuerberg, and E. Avia. "Designing human-automation interaction: a new level of automation taxonomy". *Proc. Human Factors of Systems and Technology* (2012).

[16] Card, S.K., Moran, T.P. & Newell, A. “The Psychology of Human-Computer Interaction”. Lawrence Erlbaum. (1983): 23-44

[17] Sistemas de Informação - Exercícios e Trabalhos, Modelo de Processamento de Informação Humano, Disponível em: <http://bia-roberta.blogspot.pt/2011/04/mpih-modelo-de-processamento-de.html> (Acedido: Jun. 2015)

[18] Alisson Kuhnen, MPIH Modelo de Processamento de Informação, Disponível em: <http://akuhnen.blogspot.pt/2011/06/ihm-mpih-modelo-de-processamento-de.html> (Acedido: Jun. 2015)

[19] Santos, Cristina P., URI – Santo Ângelo, Cognição e Fatores Humanos, Disponível em: <http://www.urisan.tche.br/~paludo/material/IHM/Material%20para%20Aula/2011/CognicaoFatoresHumanos_ParteII.pdf> (Acedido: Jun. 2015)

[20] Dias, Paulo. "Processamento da informação, Hipertexto e Educação." Revista Portuguesa de Educação (1993): 71-83.

[21] Broadbent, Donald Eric. “Perception and communication”. Elsevier, 2013.

[22] Parasuraman, Raja, and Christopher D Wickens. "Humans: Still vital after all these years of automation." *Human Factors: The Journal of the Human Factors and Ergonomics Society* 50.3 (2008): 511-520.

[23] Parasuraman, Raja. "Designing automation for human use: empirical studies and quantitative models." *Ergonomics* 43.7 (2000): 931-951.

[24] Turk, Matthew. "Computer Vision in the Interface." *Communications of the ACM* 47.1 (2004): 60-67.

[25] Memon, Atif M. "GUI Testing: Pitfalls and Process." Computer 8, Aug. 2002: 87-88.

[26] Dellaert, Frank et al. "Structure from motion without correspondence." *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on* 2000: 557-564.

[27] Wikipedia, Assistive Technology, Disponível em: <https://en.wikipedia.org/wiki/Assistive_technology> (Acedido: Ago. 2015)

[28] Phillips, Betsy, and Hongxin Zhao. "Predictors of assistive technology abandonment." *Assistive Technology* 5.1 (1993): 36-45.

[29] Hardesty, L.: Picture-driven computing: New research could enable computer programming based on screen shots, not just code (January 2010), Disponível em: <http://web.mit.edu/newsoffice/2010/screen-shots-0120.html> (Acedido: Jan. 2015)

[30] Communications of the ACM, Picture-Driven Computing (January 2010), Disponível em: <http://cacm.acm.org/news/68276-picture-driven-computing/fulltext> (Acedido: Jan. 2015)

[31] Juang, Biing-Hwang, and Lawrence R Rabiner. "Automatic speech recognition–A brief history of the technology development." *Encyclopedia of Language and Linguistics* (2004): 1-24.

[32] Szeliski, Richard. “Computer vision: algorithms and applications”. Springer Science & Business Media, 2010.

[33] Chang, Tsung-Hsiang, Tom Yeh, and Robert C Miller. "Sikuli: using GUI screenshots for search and automation." *Proceedings of the 22nd annual ACM symposium on User interface software and technology* 4 Oct. 2009: 183-192.

[34] Schaller, Robert R. "Moore's law: past, present and future." *Spectrum, IEEE* 34.6 (1997): 52-59.

[35] R. S. Tavares, João Manuel (Prof. Associado FEUP), “Algoritmos de Visão Computacional em Engenharia e Medicina” (2013), Disponível em: <https://web.fe.up.pt/~tavares/downloads/publications/comunicacoes/ERI2013-Presentation_JT.pdf> (Acedido: Jul.2015)

[36] Hardesty, L. (MIT News Office). Object recognition for free: System designed to label visual scenes according to type turns out to detect particular objects, too. (Maio 2015), Disponível em: <http://newsoffice.mit.edu/2015/visual-scenes-object-recognition-0508> (Acedido: Ago. 2015)

[37] Jain, Ramesh, Rangachar Kasturi, and Brian G Schunck. *Machine vision*. New York: McGraw-Hill, 1995.

[38] Wikipedia, Computer vision, Disponível em: <https://en.wikipedia.org/wiki/Computer_vision> (Acedido: Ago. 2015)

[39] Zhou, Qian-Yi, and Vladlen Koltun. "Dense scene reconstruction with points of interest." *ACM Transactions on Graphics (TOG)* 32.4 (2013): 112.

[40] J. Liu and P. Moulin, “Complexity-Regularized Image Restoration". *Proc. IEEE Int. Conf. on Image Proc. (ICIP'98)*, Vol. 1, pp. 555-559, Chicago, Oct. 1998.

[41] Winckler, Marco AA, and Marcelo Soares Pimenta. "Análise e Modelagem de Tarefas." *VI Simpósio*, IHC’2004.

[42] Sweigart, Albert. "Automate the boring stuff with Python: practical programming for total beginners." Computing and Computers. San Francisco, CA: No Starch Press, 2015.

[43] Silva, JL, Campos, J. Creissac, and Paiva, Ana CR. "Model-based user interface testing with Spec Explorer and ConcurTaskTrees." *Electronic Notes in Theoretical Computer Science* 208 (2008): 77-93.

[44] Paterno, Fabio. “Model-based design and evaluation of interactive applications.” Springer Science & Business Media, 2012.

[45] Mori, Giulio, Fabio Paternò, and Carmen Santoro. "CTTE: support for developing and analyzing task models for interactive system design." *Software Engineering, IEEE Transactions on* 28.8 (2002): 797-813.

[46] Paternò, Fabio, and Cristiano Mancini. "Model-based design of interactive applications." *intelligence* 11.4 (2000): 26-38.

[47] Puerta, Angel R. "A model-based interface development environment." *Software, IEEE* 14.4 (1997): 40-47.

[48] Department of Computer Science - University of Pisa, Course on Formal Methods for Interactive Systems, "Task Modelling: Tools and Methods based on Task Models", Disponível em: <http://www.di.unipi.it/~cerone/courses/fmis/course-slides/course-FMIS-04-TaskModelling.pdf> (Acedido: Fev. 2015)

[49] Department of Computer Science - University of Pisa, Course on Formal Methods for Interactive Systems, "Interface Generation from Task Models: Tools and Methods for the Design of Multi-Device User Interfaces", Disponível em: <http://www.di.unipi.it/~cerone/courses/fmis/course-slides/course-FMIS-05-TaskInterface.pdf> (Acedido: Fev. 2015)

[50] Barbosa, Ana Sofia Barros. "Geração de Casos de Teste a partir de Modelos de Tarefas". Porto, 2010. Dissertação (Mestrado Integrado em Engenharia Informática e Computação) – Faculdade De Engenharia Da Universidade Do Porto.

[51] Silva, José L. Centro de Competência de Ciências Exactas e da Engenharia – Universidade da Madeira, Interação Humano-Computador, "Cenários de Interação e Modelos de tarefas" (Acedido: Mai. 2015)

[52] Martinie, C., Palanque P., and Winckler M. "Structuring and composition mechanisms to address scalability issues in task models". Human-Computer Interaction–INTERACT 2011 conference: 589-609.

[53] Interactive Critical Systems Group, "HAMSTERS", Disponível em: <http://www.irit.fr/recherches/ICS/softwares/hamsters/> (Acedido: Fev. 2015)

[54] Interactive Critical Systems Group, "HAMSTERS - Task types and temporal ordering", Disponível em: <http://www.irit.fr/recherches/ICS/softwares/hamsters/types.html> (Acedido: Fev. 2015)

[55] Boy A. Guy. "The Handbook of Human-Machine Interaction: A Human-Centered Design Approach". England : Ashgate Publishing Limited, 2011. ISBN 978-1-4094-1171-0

[56] Silva, José L. "Geração de Oráculos de Teste a partir de Modelos de Tarefas". Funchal, 2007. Relatório de Estágio (Licenciatura em Engenharia de Sistemas e Informática) – Universidade Do Minho.

[57] Paris, Cécile, Shijian Lu, and Keith Vander Linden. "Environments for the construction and use of task models." *The Handbook of Task Analysis* (2003): 467-482.

[58] Paternò, Fabio, Mori, Giulio, and Galiberti, Riccardo. "CTTE: an environment for analysis and development of task models of cooperative applications." CHI'01 Extended Abstracts on Human Factors in Computing Systems, 31 Mar. 2001: 21-22.

[59] Paternò, Fabio. "ConcurTaskTrees and UML: how to marry them." Proceedings of the TUPIS'2000 Workshop at the UML Oct. 2000.

[60] Paternò, Fabio, Santoro, Carmen, and Spano, L. Davide. "MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments." *ACM Transactions on Computer-Human Interaction (TOCHI)* 16.4 (2009): 19.

[61] Raposo, Alberto, Departamento de Informática, PUC - Rio de Janeiro, "Introdução a Interação Humano-Computador: Análise e Modelagem de Tarefas", 23/05/2011, Disponível em: <http://www.inf.puc-rio.br/~inf1403/docs/alberto2012-1/17_HTA_GOMS.pdf> (Acedido: Ago. 2015)

[62] Paternò, Fabio, G. Ballardin, and C. Mancini. "Modelling Multi-Users Tasks." CNUCE, C.N.R. 2000.

[63] TestPlant: eggPlant Test Automation Tools for software, "eggplant Functional", Disponível em: <http://www.testplant.com/eggplant/testing-tools/eggplant-developer/> (Acedido: Dez. 2014)

[64] Vamsi Krishna, A Tale of Testing, "Comparison Report: Sikuli Vs Eggplant", Disponível em: <http://testwarriors.blogspot.pt/2012/04/comparison-report-sikuli-vs-eggplant.html> (Acedido: Dez. 2014)

[65] Cogitek: Tools for QA Professionals, "Automate Testing of Windows Applications", Disponível em: <http://www.cogitek.com/riatest/features/technologies/windows.html> (Acedido: Dez. 2014)

[66] Automa: Next Generation GUI Automation, "Automa 1.9.0: A Sikuli Alternative", Disponível em: <http://www.getautoma.com/blog/sikuli-alternative> (Acedido: Dez. 2014)

[67] Paternò, Fabio, and Cristiano Mancini. "Developing task models from informal scenarios." CHI'99 *Extended Abstracts on Human Factors in Computing Systems* 15 May. 1999: 228-229.

[68] Paternò, Fabio, Carmen Santoro, and Sophie Tahmassebi. *Formal models for cooperative tasks: concepts and an application for en-route air traffic control*. Springer Vienna, 1998.

[69] Paternò, Fabio. "Task models in interactive software systems." *In Handbook Of Software Engineering And Knowledge* 2001.

[70] Bastide, R. et al. "Designing Interactive Applications for Air Traffic Control with the Support of MEFISTO."

[71] Wikipedia, Blender (software), Disponível em: <https://en.wikipedia.org/wiki/Blender_(software)> (Acedido: Jul. 2015)

[72] Blender, Features, Disponível em: <http://www.blender.org/features/> (Acedido: Jul. 2015)

[73] Microsoft Developer Network, XPath Syntax, Disponível em: <https://msdn.microsoft.com/en-us/library/ms256471(v=vs.110).aspx> (Acedido: Jan. 2015)

[74] Lund, A.M., 2001. Measuring Usability with the USE Questionnaire. STC Usability SIG Newslett. 8(2).

[75] Sikuli Script, “SikuliX Documentation”, Disponível em: <http://sikulix-2014.readthedocs.org/en/latest/index.html> (Acedido: Dez. 2014)

[76] SikuliX powered by RaiMan, Support – “Experimental: Play with the RunServer”, Disponível em: <http://www.sikulix.com/support.html> (Acedido: Jun. 2016)

[77] vmware, “VMware Workstation”, Disponível em: <http://www.vmware.com/products/workstation> (Acedido: Abr. 2015)

[78] VMware, Inc. “Using vmrun to Control Virtual Machines”, Disponível em: <https://www.vmware.com/pdf/vix160_vmrun_command.pdf> (Acedido: Abr. 2015)

[79] Drummond, Jon. "Understanding interactive systems." *Organised Sound* 14.02 (2009): 124-133.

[80] Grudin, Jonathan. "Interactive systems: Bridging the gaps between developers and users." *Computer 4* (1991): 59-69.

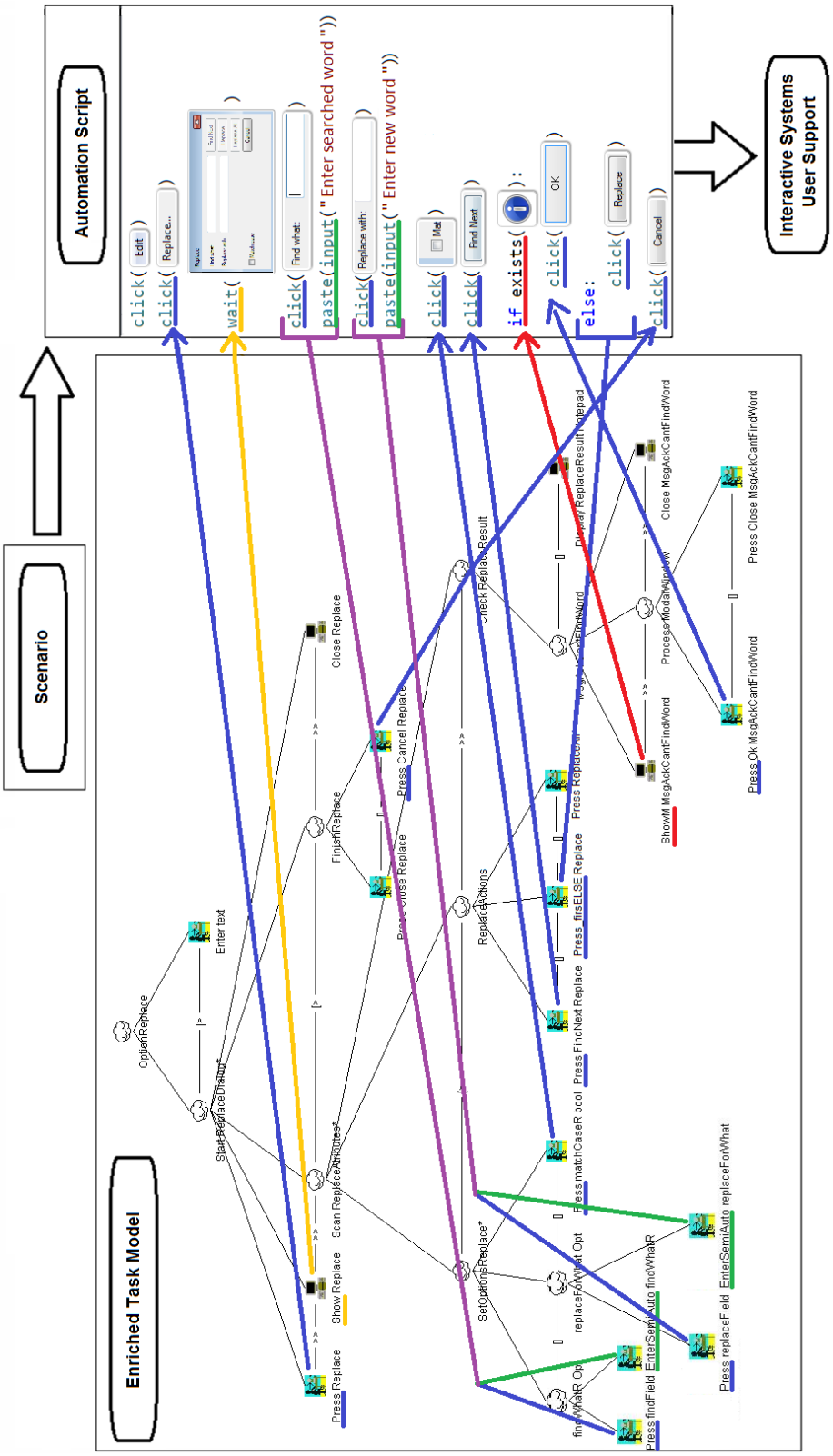
[81] Harrison, S. M. "A comparison of still, animated, or nonillustrated on-line help with written or spoken instructions in a graphical user interface." *Proceedings of the SIGCHI conference on Human factors in computing systems* 1 May. 1995: 82-89.

[82] Knabe, Kevin. "Apple guide: a case study in user-aided design of online help." *Conference companion on Human factors in computing systems* 7 May. 1995: 286-287.

Anexos

1 - Função de Procura/Substituição no Notepad

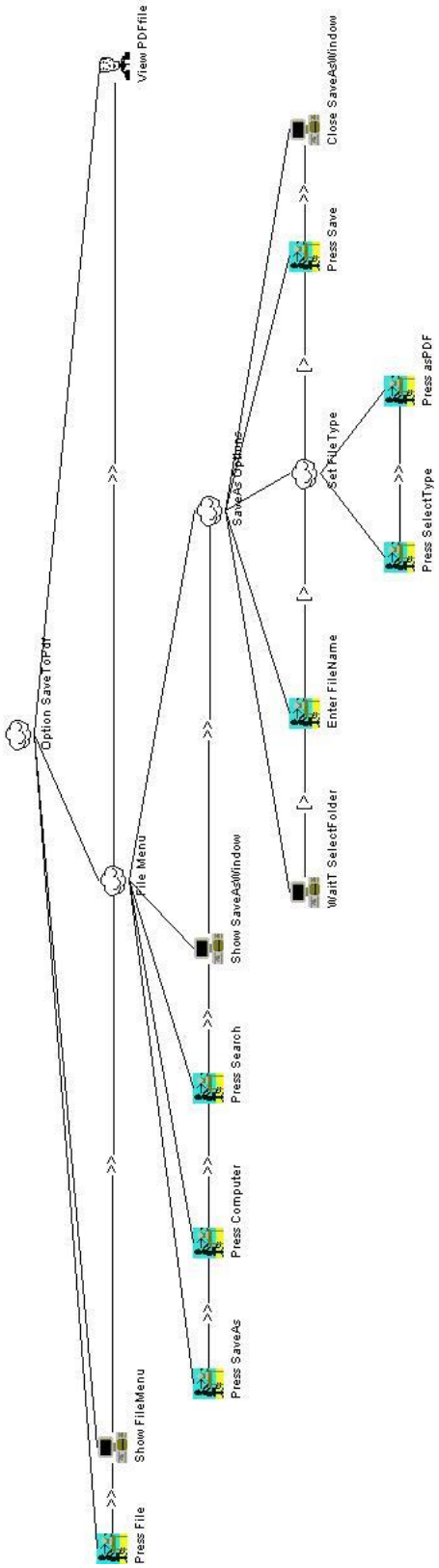
1.1 - Visão geral da abordagem, no contexto deste caso de estudo



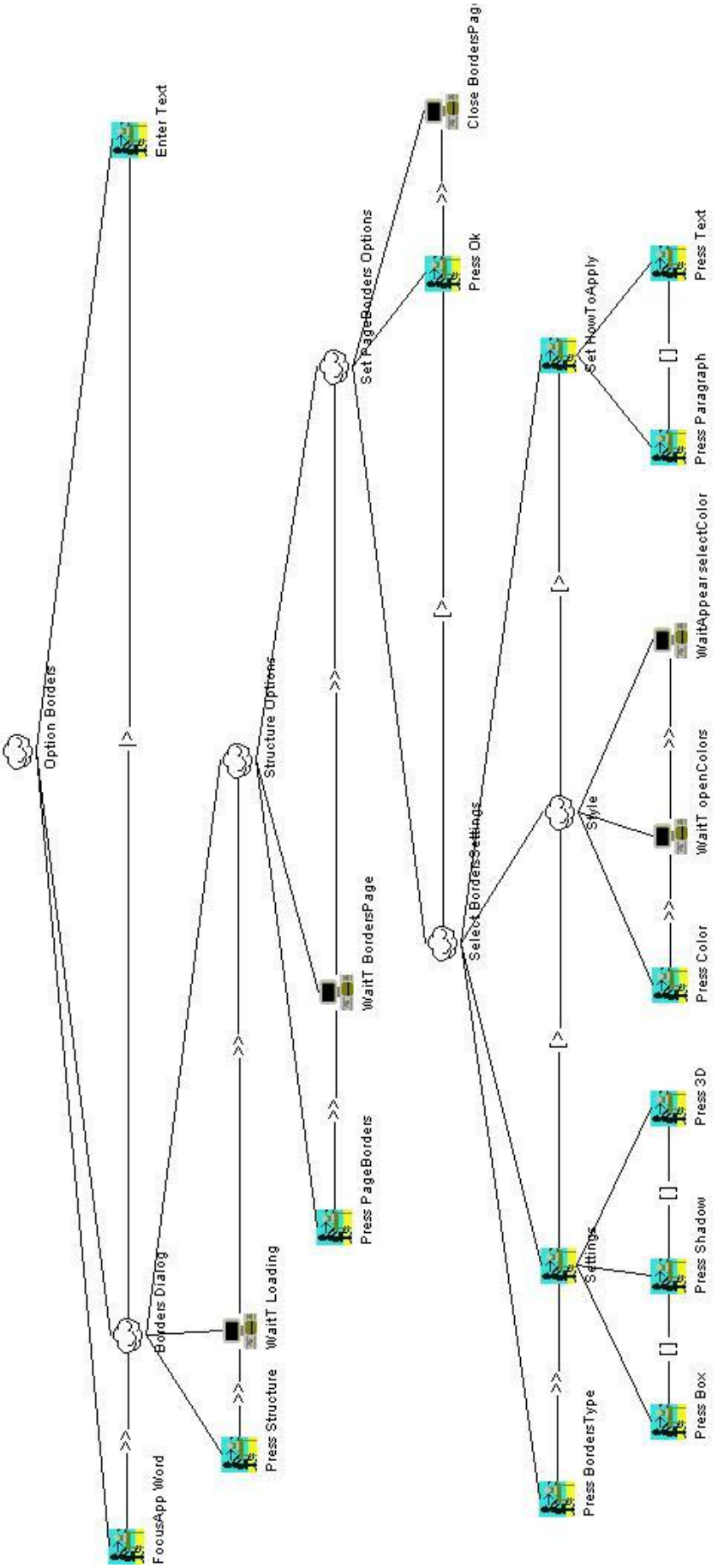
2 - Modelos de Tarefas

2.1 - Word Support System

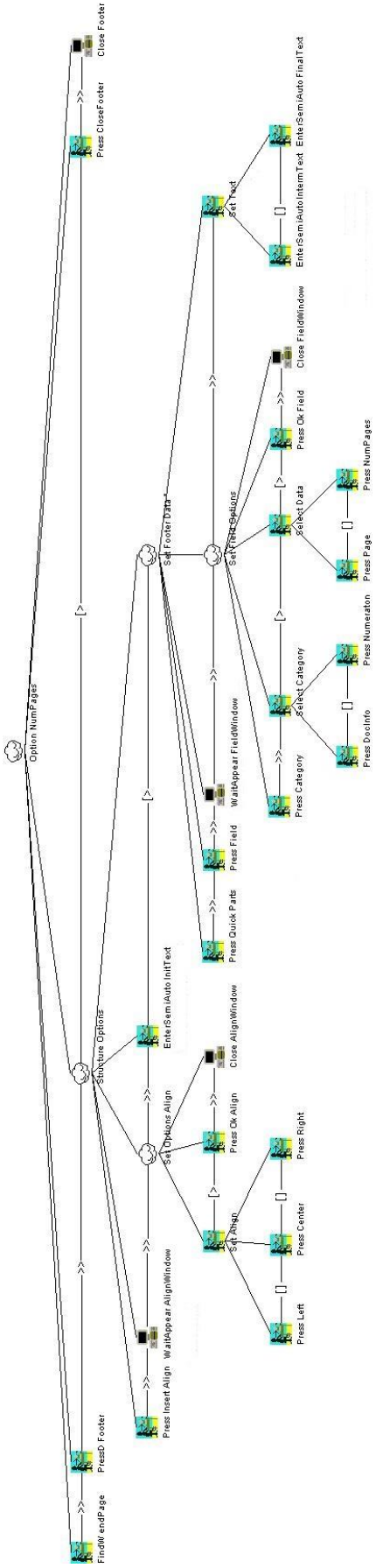
2.1.1 - Guardar documento no formato PDF



2.1.2 - Adicionar bordas ao parágrafo selecionado/1ª página/todo o documento

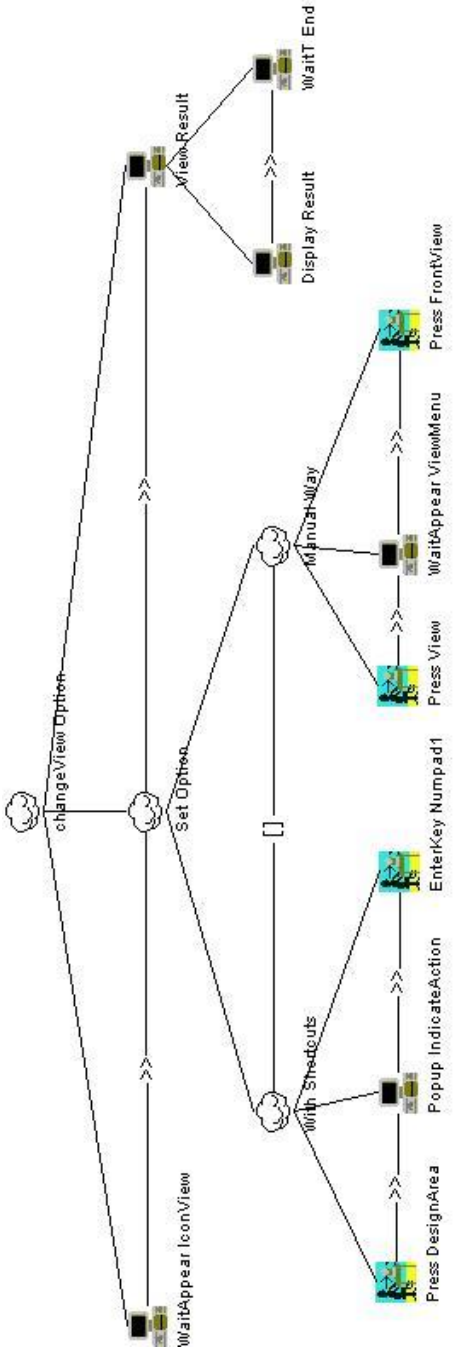


2.1.3 - Adicionar numeração personalizada de páginas

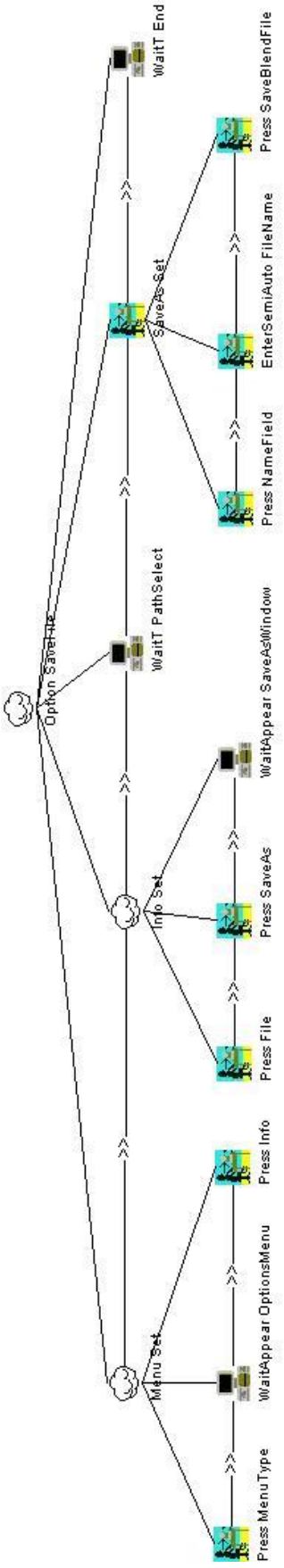


2.2 - Blender Support System

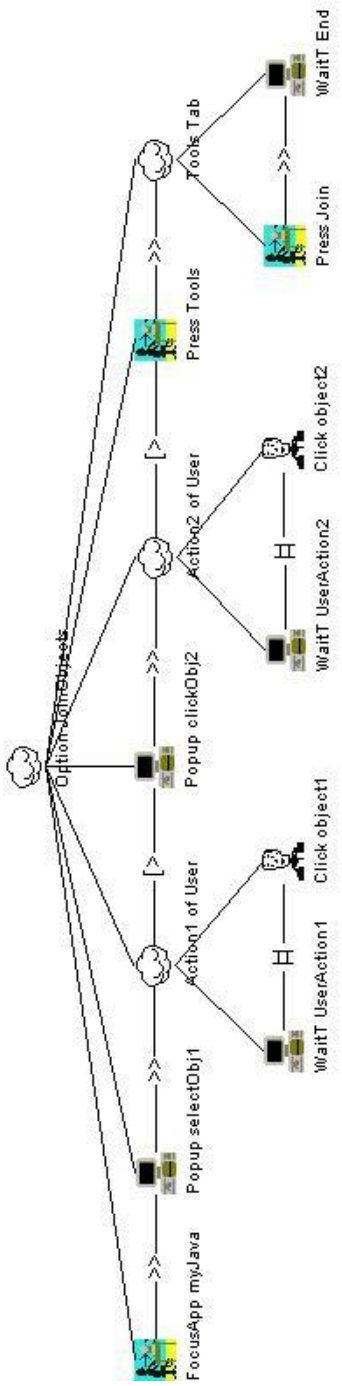
2.2.1 - Alterar tipo de vista para frontal, manualmente/recorrendo a atalhos



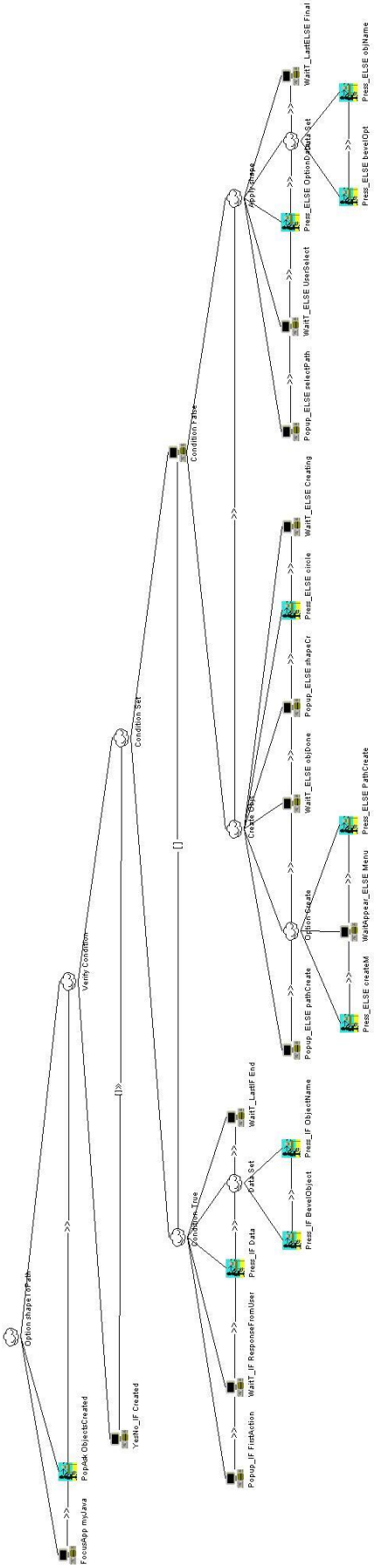
2.2.2 - Guardar ficheiro BLEND



2.2.3 - Juntar dois objetos (Join)

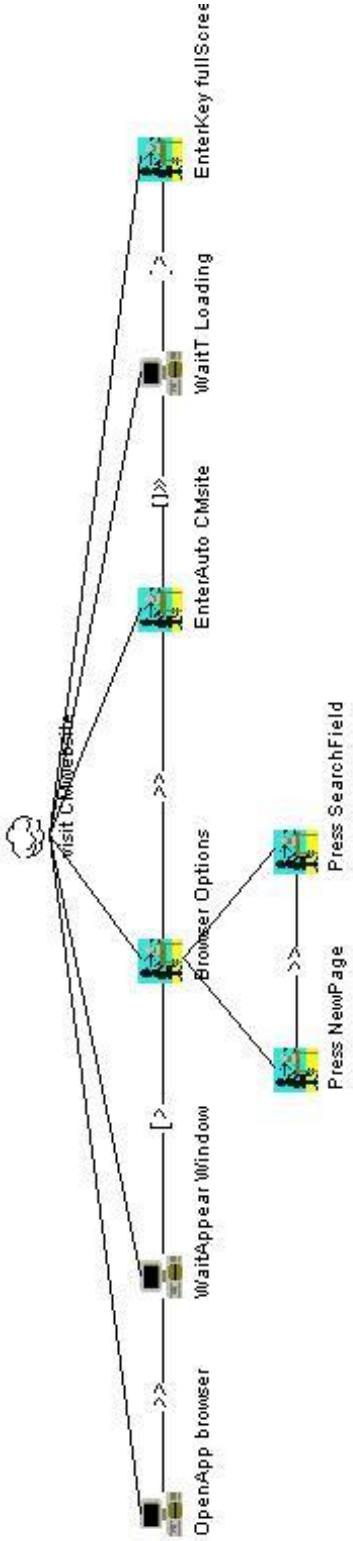


2.2.4 - Aplicar forma cilíndrica a um path

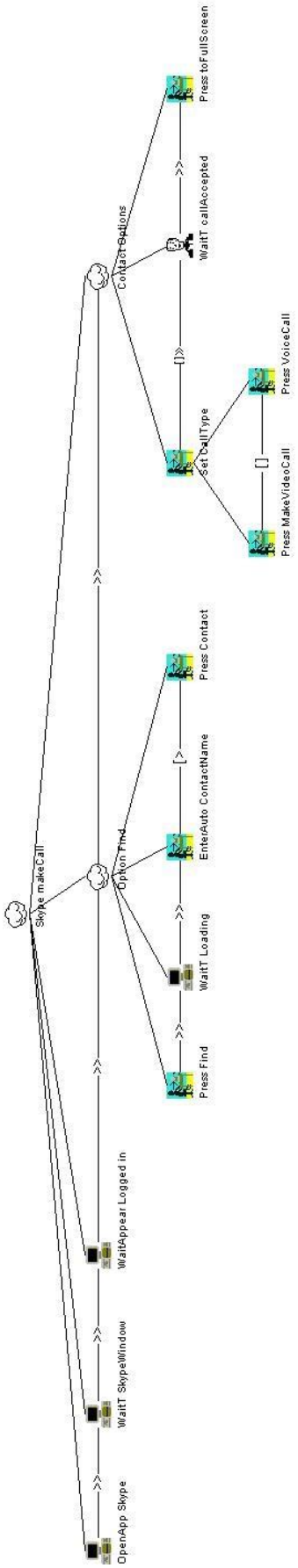


2.3 - Home Care Application

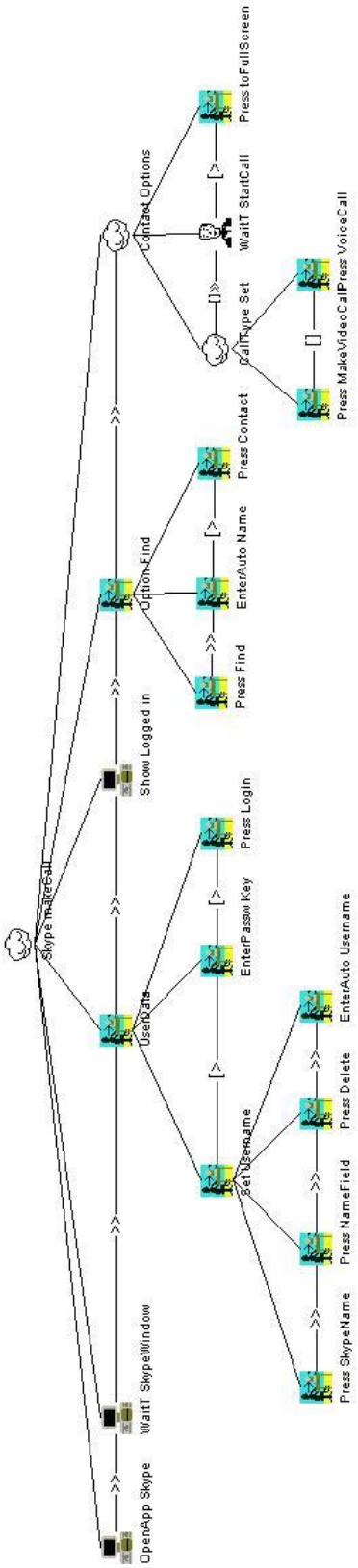
2.3.1 - Consultar notícias no Correio da Manhã



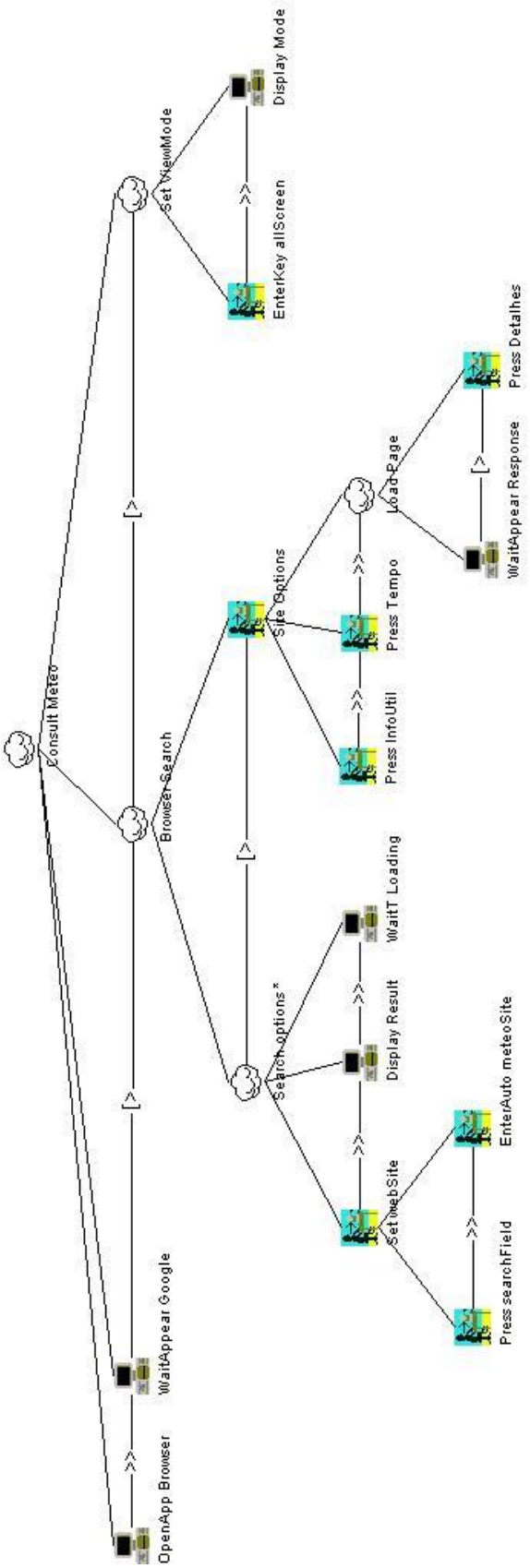
2.3.2 - Efetuar chamada (de voz ou de vídeo) no Skype



2.3.3 - Efetuar login e chamada (voz/vídeo) no Skype



2.3.4 - Consultar meteorologia local online



3 - Questionários

3.1 - User Support System / Abordagem

User Support System

Este questionário é parte do projeto intitulado "Improving the Use of Interactive Systems", inserido no Mestrado de Engenharia Informática da Universidade da Madeira, que consiste, basicamente, no desenvolvimento de uma abordagem de suporte ao uso de Sistemas Interativos. O mesmo visa retirar algumas ilações acerca da utilidade e simplicidade da aplicação da abordagem apresentada, cujo objetivo final passa por simplificar, através da automatização de tarefas, a utilização de Sistemas Interativos.

Agradecemos desde já a sua colaboração!

QUESTÕES PESSOAIS

Idade:

Género:

- Masculino
- Feminino

Já tinha trabalhado com Modelos de Tarefas?

- Sim
- Não

Conhecia a ferramenta CTTE (ConcurTaskTrees Environment)?

- Sim
- Não

Possui conhecimentos ao nível da Programação Orientada a Imagens?

- Sim
- Não

Conhecia a ferramenta SikuliX?

- Sim
- Não

INTERFACE - UTILIDADE

Ajuda-me a ser mais eficaz.

	0	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	○	Concordo plenamente

Ajuda-me a ser mais produtivo.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo plenamente

É útil.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo plenamente

Torna mais simples realizar as tarefas que eu quero.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo plenamente

Atende as minhas necessidades.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo plenamente

INTERFACE - FACILIDADE DE UTILIZAÇÃO

É simples e fácil de usar.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo plenamente

Requer o mínimo de passos possível para realizar o que pretendo.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo plenamente

A sua utilização não requer esforço.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo plenamente

Consigo utilizá-la sem precisar de instruções.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

Não detectei quaisquer incoerências na sua utilização.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

Posso recuperar de erros de forma fácil e rápida.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

INTERFACE - FACILIDADE DE APRENDIZAGEM

Aprendi a utilizá-la rapidamente.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

INTERFACE - SATISFAÇÃO

Estou satisfeito com ela.

1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ Concordo totalmente

Recomendaria a um amigo.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

Sinto que me faz falta.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

Funciona da maneira que eu quero que funcione.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

ABORDAGEM

As ferramentas utilizadas são eficazes.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

É fácil estender o Modelo de Tarefas.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

As regras utilizadas são claras e compreensivas.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

É fácil aplicar todo o processo.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

OPINIÃO

O resultado final correspondeu às suas expectativas?

○ Sim
○ Não

Enumere os principais aspectos negativos.

Enumere os principais aspectos positivos.

Tem alguma sugestão/dica?

Enviar

Nunca envie senhas pelo Formulários Google.

3.2 - Word/Blender Support System

Note-se que os questionários alusivos às duas aplicações de ajuda são idênticos, alterando-se apenas a referência ao nome da aplicação em causa (Word ou Blender).

Interface de Suporte ao Word

Este questionário é parte do projeto intitulado "Improving the Use of Interactive Systems", inserido no Mestrado de Engenharia Informática da Universidade da Madeira, que consiste, basicamente, no desenvolvimento de uma abordagem de suporte ao uso de Sistemas Interativos. Este tem como objetivo perceber e avaliar o grau de satisfação do utilizador relativamente à Interface de suporte à utilização do WORD.

Agradecemos desde já a sua colaboração!

*Obrigatório

QUESTÕES PESSOAIS

Idade: *

Género: *

- Masculino
- Feminino

Nível de conhecimento informático/tecnológico. *

- Nulo
- Reduzido
- Normal
- Elevado

Já conhece/utilizou a ferramenta Office Word? *

- Sim
- Não

UTILIDADE

Ajuda-me a ser mais eficaz.

	0	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	○	Concordo plenamente

Ajuda-me a ser mais produtivo.

	0	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	○	Concordo plenamente

É útil.

	0	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	○	Concordo plenamente

Torna mais simples realizar as tarefas que eu quero.

	0	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	○	Concordo plenamente

Poupa-me tempo quando a uso.

	0	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	○	Concordo plenamente

Atende as minhas necessidades.

	0	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	○	Concordo plenamente

Faz tudo o que eu esperaria que fizesse.

	0	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	○	Concordo plenamente

FACILIDADE DE UTILIZAÇÃO

É fácil de usar.

	0	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	○	Concordo plenamente

Requer o mínimo de passos possível para realizar o que pretendo.

	0	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	○	Concordo plenamente

A sua utilização não requer esforço.

	0	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	○	Concordo plenamente

Consigo utilizá-la sem precisar de instruções.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

Não detectei quaisquer incoerências na sua utilização.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

Posso recuperar de erros de forma fácil e rápida.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

FACILIDADE DE APRENDIZAGEM

Aprendi a utilizá-la rapidamente.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

É fácil de lembrar como utilizá-la.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

SATISFAÇÃO

Estou satisfeito com ela.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

Recomendaria a um amigo.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

Sinto que preciso de a ter.

	0	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	○	Concordo totalmente

É agradável de utilizar.

	0	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	○	Concordo totalmente

Funciona da maneira que eu quero que funcione.

	0	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	○	Concordo totalmente

OPINIÃO (opcional)

Enumere os principais aspectos negativos.

Enumere os principais aspectos positivos.

Tem alguma sugestão/dica?

Enviar

Nunca envie senhas pelo Formulários Google.

3.3 - Sistema de Simplificação – Home Care Application

Interface Sistema Simplificado

Este questionário é parte do projeto intitulado "Improving the Use of Interactive Systems", inserido no Mestrado de Engenharia Informática da Universidade da Madeira, que consiste, basicamente, no desenvolvimento de uma abordagem de suporte ao uso de Sistemas Interativos. E visa recolher informações acerca da utilidade e usabilidade da Interface apresentada, cujo objetivo central é simplificar o uso de Sistemas Interativos a utilizadores principiantes.

Agradecemos desde já a sua colaboração!

*Obrigatório

QUESTÕES PESSOAIS

Idade: *

Género: *

- Masculino
- Feminino

Nível de conhecimento informático e/ou tecnológico. *

- Nulo
- Reduzido
- Normal
- Elevado

Que aplicações mais utiliza no computador?

- Office (Word, Excel, PowerPoint)
- Bloco de Notas
- Browser (navegador de internet)
- Calculadora
- Paint
- Jogos
- Outro:

UTILIDADE

Esta aplicação...

Ajuda-me a ser mais eficaz.

	0	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	○	Concordo plenamente

Ajuda-me a ser mais produtivo.

	0	1	2	3	4	5	
Discordo totalmente	○	○	○	○	○	○	Concordo plenamente

É útil.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo plenamente

Torna mais simples realizar as tarefas que eu quero.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo plenamente

Permite-me poupar tempo.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo plenamente

Atende as minhas necessidades.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo plenamente

Faz tudo o que eu esperava que fizesse.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo plenamente

FACILIDADE DE UTILIZAÇÃO

É simples e fácil de usar.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo plenamente

Requer o mínimo de passos possível para realizar o que pretendo.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo plenamente

Consigo utilizá-la sem precisar de instruções.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

Consigo utilizá-la sem precisar de instruções.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

Não detectei quaisquer incoerências na sua utilização.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

SATISFAÇÃO

Estou satisfeito com ela.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

Recomendaria a um amigo.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

Sinto que me faz falta.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

É agradável de utilizar.

0 1 2 3 4 5

Discordo totalmente ○ ○ ○ ○ ○ ○ Concordo totalmente

OPINIÃO (opcional)

Enumere os principais aspectos negativos.

Enumere os principais aspectos positivos.

Tem alguma sugestão/dica?

Enviar

Nunca envie senhas pelo Formulários Google.