Instituto Superior de Engenharia do Porto

Edição Gráfica de Layout Fabril no Apoio ao Planeamento e Controlo da Produção

Grupo de Investigação em Engenharia do Conhecimento e Apoio à Decisão

Paulo Alexandre Ferreira Taveira

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Área de Especialização em

Sistemas Gráficos e Multimédia

Orientadora: Ana Maria Dias Madureira Pereira

Co-Orientador: João Paulo Jorge Pereira

Júri:

Presidente:

Doutor José António Reis Tavares, Instituto Superior de Engenharia do Porto

Vogais:

Doutor Filipe de Faria Pacheco Paulo, Instituto Superior de Engenharia do Porto

Doutora Ana Maria Dias Madureira Pereira, Instituto Superior de Engenharia do Porto

Doutor João Paulo Jorge Pereira, Instituto Superior de Engenharia do Porto

Porto, Novembro 2012

Aos meus pais, à minha irmã, familiares e amigos.

Agradecimentos

Quero agradecer à minha família pelo apoio prestado, em especial aos meus pais pela força e incentivo que me têm dado ao longo de toda a minha graduação, aos quais são fruto do meu trabalho.

Aos meus amigos e colegas de faculdade pela motivação e apoio incondicional. Particularmente às minhas amigas Cinira Alves e Cláudia Pereira pela ajuda prestada na revisão do documento.

Quero agradecer também às pessoas que participaram na sessão de avaliação de usabilidade, pois foi um dos pontos essenciais na realização desta tese de mestrado. Particularmente ao investigador Eduardo Piairo, com o qual trabalhei e pela boa disposição mostrada.

Aos meus orientadores do ISEP Ana Madureira e João Paulo Pereira, pelo esclarecimento de dúvidas, pela revisão e acompanhamento constante do projecto e documento.

Resumo

O contributo da área de investigação Interacção Humano-Computador (HCI) está patente não só na qualidade da interacção, mas também na diversificação das formas de interacção. A HCI define-se como sendo uma disciplina que se dedica ao desenho, desenvolvimento e implementação de sistemas de computação interactivos para uso humano e estudo dos fenómenos relevantes que os rodeiam.

Pretende-se, no âmbito desta tese de mestrado, o desenvolvimento de um Editor Gráfico de *Layout* Fabril a integrar num SAD para suporte ao Planeamento e Controlo da Produção. O sistema deve ser capaz de gerar um *layout* fabril do qual constam, entre outros objectos, as representações gráficas e as respectivas características/atributos do conjunto de recursos (máquinas/processadores) existentes no sistema de produção a modelar.

O módulo desenvolvido será integrado no projecto de I&D ADSyS (*Adaptative Decision Support System for Interactive Scheduling with MetaCognition and User Modeling Experience)*, melhorando aspectos de interacção referentes ao sistema *AutoDynAgents*, um dedicado ao escalonamento, planeamento e controlo de produção.

Foi realizada a análise de usabilidade a este módulo com a qual se pretendeu realizar a respectiva avaliação, através da realização de um teste de eficiência e do preenchimento de um inquérito, da qual se identificaram um conjunto de melhorias e sugestões a serem consideradas no refinamento deste módulo.

Palavras-Chave (Tema): Interfaces, Interacção Humano-Computador

Palavras-Chave (Tecnologias): Java, XML

Abstract

The contribution of the Human-Computer Interaction (HCI) research area is reflected not only on the quality of interaction, but also in diversified forms of interaction. HCI is defined as a discipline that is dedicated to the design, development and implementation of interactive computing systems for human use and study of relevant phenomena that surround them.

It is intended, in the context of this thesis, the development of a Graphical Editor of a Factory Layout, to be integrated in a DSS, supporting the Production Planning and Control. The system must be able to generate a factory layout which contains, among other objects, graphical representations and their characteristics / attributes of the feature set (machines / processors) in the production system to model.

The developed module will be integrated in the R&D ADSyS (*Adaptive Decision Support System for Interactive Scheduling with MetaCognition and User Experience Modeling*), improving interaction aspects for the AutoDynAgents system, one dedicated to the production scheduling, planning and control.

Usability analysis of this module was conducted which it was intended to accomplish their evaluation by conducting a test of efficiency and filling out a survey, which was identified a number of improvements and suggestions for consideration in the refinement of this module.

Key Words (Theme): Interfaces, Human-Computer Interaction

Key Words (Technologies): Java, XML

Índice

Índice de Figuras

Índice de Tabelas

Notação e Glossário

API	Application Programming Interface
AWT	Abstract Window Toolkit
EPL	Eclipse Public License
FCT	Fundação para a Ciência e Tecnologia
IBM	International Business Machines
JFC	Java Foundation Classes
JNI	Java Native Interface
MVC	Model View Controller
LGPL	Lesser General Public License
SWT	Standard Widget Toolkit
WYSIWYG	What You See Is What You Get
XML	Extensive Markup Language

1 Introdução

O presente trabalho de investigação foi desenvolvido no âmbito da dissertação do Mestrado em Engenharia Informática do Instituto Superior de Engenharia do Porto, enquadra-se no projecto de ADSyS - Sistema de Apoio à Decisão Adaptativo e Interactivo para Escalonamento com Metacognição e Modelação da Experiência do Utilizador (PTDC/EME-GIN/109956/2009) e visa o desenvolvimento de um módulo de edição gráfica para um sistema de escalonamento.

1.1 Motivação

O contributo da área de investigação Interacção Humano-Computador (HCI) está patente não só na qualidade da interacção, mas também na diversificação das formas de interacção. A HCI define-se como sendo uma disciplina que se dedica ao desenho, desenvolvimento e implementação de sistemas de computação interactivos para uso humano e estudo dos fenómenos relevantes que os rodeiam.

1.2 Objectivos

Pretende-se, no âmbito desta tese de mestrado, o desenvolvimento de um Editor Gráfico de Layout Fabril a integrar num SAD para suporte ao Planeamento e Controlo da Produção. O sistema deve ser capaz de gerar um *layout* fabril do qual constam, entre outros objectos, as representações gráficas e as respectivas características/atributos do conjunto de recursos (máquinas/processadores) existentes no sistema de produção a modelar.

De forma sucinta, o trabalho desenvolvido consistiu nas seguintes fases:

- Pesquisa bibliográfica referente ao estado da arte sobre: Interacção Homem/Computador e Desenvolvimento de Interfaces com o Utilizador;
- Análise e levantamento de sistemas de escalonamento comerciais e académicos;
- Análise e levantamento de Bibliotecas e API's no desenvolvimento de interfaces gráficas;
- Análise e Estudo de aspectos relacionados com o desenvolvimento de interfaces utilizador/SAD;

- Formulação e modelação do problema - especificação dos requisitos para a interface do editor de layouts: tarefas, subtarefas e metodologias envolvidas;
- Desenvolvimento do Protótipo do editor de *layouts*;
- Teste e validação do protótipo.

1.3 Organização do relatório

O presente relatório está dividido em sete capítulos.

No segundo capítulo é elaborado o estado da arte, onde se descrevem aspectos gerais de interfaces e usabilidades, o conceito de usabilidade e alguns aspectos relacionados. Também se aborda o conceito dos factores humanos, de algumas normas e linhas de orientação relacionados, algumas orientações acerca dos estilos de interacção. Por fim é descrita a metodologia de desenvolvimento de *software* em estrela, e também do conceito de prototipagem rápida.

O terceiro capítulo apresenta os resultados de um estudo efectuado a alguns sistemas de escalonamento existentes (académicos e comerciais), referindo algumas funcionalidades, vantagens e limitações, acompanhados de figuras e identificando alguns aspectos de usabilidade. Por fim é efectuado um estudo ao sistema AutoDynAgents, onde também se identificam alguns aspectos a poderem ser melhorados.

No quarto capítulo é efectuado um estudo a algumas API's de desenvolvimento de gráficos e de interfaces gráficas, referindo algumas funcionalidades e elaborando um exemplo ilustrativo comum de modo a obter uma percepção do potencial de cada API. Cada um destes estudos termina com uma tabela comparativa, onde se sistematizam vantagens e limitações de cada API, e a sugestão da API de desenvolvimento de interfaces gráficas e de gráficos, justificando devidamente essa escolha.

No quinto capítulo refere-se ao desenvolvimento do módulo de edição do *layout* fabril, onde se efectua a análise de requisitos, o processo de desenvolvimento do protótipo, as tecnologias utilizadas e as várias funcionalidades desenvolvidas.

No sexto capítulo é descrita a análise de usabilidade ao módulo desenvolvido, onde se indicam as várias fases de elaboração do inquérito efectuado. Em seguida são analisados os resultados.

No sétimo capítulo, conclusão, faz um resumo dos vários capítulos abordados anteriormente. São indicados os objectivos realizados, limitações do protótipo desenvolvido e identificadas perspectivas de trabalho.

2 Interacção Pessoa/Computador

Neste capítulo aborda-se alguns aspectos relativos aos sistemas interactivos, tais como a composição dos mesmos, a importância da interface com o utilizador e os domínios a ter em conta na criação de um sistema interactivo. Também se irá abordar o conceito de usabilidade, também indicando várias definições, linhas de orientação, estilos de interacção, alguns princípios orientadores do *design* de interacção e, por fim, uma metodologia de desenvolvimento para a criação de um sistema interactivo. Uma parte deste capítulo foi concretizado tendo em conta a estrutura do livro *Developing User Interfaces* de Deborah Hix e H. Rex Hartson (Hix & Hartson, 1993).

2.1 Composição de um sistema interactivo

Um sistema interactivo pode ser dividido em duas componentes distintas: funcional ou computacional e comunicacional. A componente computacional implementa a funcionalidade do sistema. A componente comunicacional implementa a interacção com o utilizador (Hix & Hartson, 1993, p. 5).

2.2 Importância da interface com o utilizador

Um sistema considera-se bom sistema quando não só efectua as actividades exigidas, como também permite uma boa comunicação com o utilizador. Assim, a comunicação do utilizador com o sistema consegue ser tão importante como a computação que o sistema efectua (Galitz, 2007).

Um sistema bom em computação mas menos bom em comunicação provavelmente não terá muito sucesso, pois o utilizador não terá tanta segurança na sua utilização. Provavelmente irá acabar por ocupar mais tempo a tentar perceber melhor o sistema, que propriamente a utilizar.

A importância da interface com o utilizador, é de tal ordem que, muitas das vezes tem como consequência a aceitação ou a rejeição do sistema por parte do utilizador. Uma interface boa poderá implicar uma boa aceitação, enquanto uma interface má poderá implicar numa rejeição.

Se um utilizador não sentir facilidade na sua utilização, ou porque os seus procedimentos são muito complicados ou por outro motivo que não agrade ao utilizador, o sistema interactivo poderá não ter sucesso.

2.3 Usabilidade: conceito e definições

A usabilidade de um sistema permite ao utilizador saber quando um sistema consegue satisfazer as necessidades dos seus potenciais utilizadores, ao que designamos de aceitabilidade do sistema. Este conceito é dividido em dois subconceitos, sendo um deles a aceitabilidade social e a aceitabilidade prática (Nielsen, 1993).

A aceitabilidade social consiste, exemplificando, em efectuar uma investigação aos utilizadores, com vista a verificar se existe alguma inconsistência nos dados, efectuando perguntas aos utilizadores. Existem utilizadores que podem achar que este sistema é bom para prevenir certas fraudulências, tendo, para estes utilizadores, uma aceitabilidade social elevada, mas outros utilizadores podem-se sentir ofendidos e, desta forma, tendo uma aceitabilidade social, para estes utilizadores, reduzida (Nielsen, 1993).

A aceitabilidade prática pode ser definida pela capacidade de satisfazer as necessidades do utilizador, mantendo uma boa comunicação entre ambos, sem que o utilizador se "canse" de o utilizar. Isto implica que vários factores sejam tidos em conta, tais como os custos, compatibilidade, viabilidade, e também outros relacionados com a usabilidade do sistema, tais como a aprendizagem, eficiência, memorabilidade, satisfação, e com pouca tolerância a falhas (Nielsen, 1993).

De uma forma geral, usabilidade pode ser definida pela facilidade de aprendizagem de um determinado sistema, a eficiência na utilização do mesmo, algo com que o utilizador se sinta à vontade na sua utilização, ao mesmo tempo satisfazendo as necessidades do utilizador em tempo óptimo. Esse sistema pode ser uma aplicação, um sítio *web*, livro, ou algo com que o utilizador possa interagir.

Um sistema interactivo que possua estas características pode ser considerado como um bom sistema.

Usabilidade normalmente é medida tendo vários utilizadores a testar o sistema. Estes utilizadores devem ser peritos na área em que o sistema se situa, efectuando tarefas pré-definidas ou deixando ao critério do utilizador a definição dessas tarefas.

Para testar a usabilidade de um sistema, são tidos em conta várias medidas de usabilidade em que uma toma o valor médio e verifica quando este valor é melhor que o mínimo especificado. Mas como a opinião dos utilizadores pode diferir bastante então são tidos em conta todos os valores de todas as medidas de usabilidade (Nielsen, 1993).

A usabilidade também pode ser definida tendo em conta várias definições (Nielsen, 1993):

- Aprendizagem: O quão fácil de aprender é, na utilização de um sistema interactivo?
- Eficiência: Quanto tempo demora o utilizador na realização de uma tarefa?
- Memorabilidade: Após algum tempo sem a utilização do recurso, o quão fácil será à pessoa se lembrar dos processos que o sistema engloba?
- Erro: Quantos erros o utilizador comete até atingir o que pretende do sistema?
- Satisfação: O quão agradável é o sistema?

2.3.1 **Aprendizagem**

Aprendizagem é um conceito bastante significativo na definição de usabilidade, pois convém que os sistemas sejam de fácil aprendizagem, e normalmente a primeira experiência do utilizador é a mais crucial na decisão do uso de um determinado sistema. Inicialmente, o sistema destina-se mais a um utilizador menos experiente e, com o decorrer do tempo, destinam-se cada vez mais aos utilizadores mais experientes.

Alta aprendizagem pode resultar num melhor aproveitamento do sistema, sem haver grandes custos a nível de tempo ou erro, aumentado a competência e a eficiência no uso do sistema. Este conceito pode ser medido recorrendo a utilizadores que pretendam usar este sistema, que nunca o tenham utilizado, e calculam o tempo que demoraram até atingir um certo nível de experiência nesse sistema. Não convém que os testes sejam calculados mediante se efectuou ou não, pois o que interessa é a quantidade de tempo que demorou a efectuar as tarefas e não o facto de se o utilizador realmente as fez ou não.

2.3.2 **Eficiência**

Eficiência é um conceito usado para medir o número de recursos utilizados para a realização de uma tarefa ou procedimento. Neste caso, a eficiência resume-se no tempo usado por utilizadores experientes na realização da tarefa. Quanto maior o tempo, menor a sua eficiência.

2.3.3 **Memorabilidade**

Memorabilidade é outro conceito de elevada importância para um sistema. Um sistema com elevada aprendizagem pode fazer com que o sistema seja de fácil memorabilidade. Este conceito não se aplica quando um utilizador está a experimentar o sistema pela primeira vez, mas, por exemplo, quando um utilizador que já usufruiu do sistema, tenha estado algum tempo sem o utilizar, voltar a usar o programa e ainda saber como efectuar certas operações.

Estes testes normalmente não são feitos por utilizadores novatos ou experientes, mas sim com utilizadores casuais, que são utilizadores que não têm nem tão pouca experiência como os novatos, nem tanta como os experientes, pois os mais experientes tendem a lembrar-se do modo de funcionamento das tarefas mesmo após elevado tempo fora do uso do sistema. Daí os utilizadores casuais serem o foco principal para testar esta medida.

2.3.4 **Erro**

Um erro é qualquer acção não esperada que leva a resultados inesperados. Normalmente mede-se tendo em conta o tempo que o utilizador demora e o número de tentativas na realização de um procedimento. Erros de maior gravidade devem ser tratados separadamente e dando-lhes mais atenção.

2.3.5 **Satisfação Subjectiva**

Satisfação subjectiva resume-se ao quão agradável é a utilização do sistema. O que é medido é a quantidade de divertimento que o utilizador tem na utilização do sistema. Este teste pode ser medido efectuando certos testes psicofisiológicos, tais como o batimento cardíaco, pressão do sangue, adrenalina no sangue, mas estes sistemas costumam ser bastante intimidantes no sentido que é necessário o utilizador estar preso a uma máquina dedicada para esse fim, e este tipo de testes poderá não ser o melhor.

Outra alternativa é perguntando ao utilizador a sua opinião relativamente ao sistema e ao nível de agrado do mesmo, e para assegurar a consistência da opinião do utilizador, é feito um questionário com várias perguntas acerca do sistema.

2.4 Domínio estrutural e comportamental

Na construção de um sistema interactivo há que ter em consideração que um sistema não só deverá satisfazer as necessidades do utilizador a nível de execução de tarefas, como também o utilizador deve sentir-se confortável na sua utilização. Podemos dividir o sistema em dois domínios distintos: o domínio estrutural e o domínio comportamental (Hix & Hartson, 1993, p. 6).

No domínio comportamental, não são tidos em conta os problemas a nível de interfaces e usabilidade mas sim o comportamento do utilizador e a interface com que ele interage. Neste domínio estão envolvidos os factores humanos, as limitações cognitivas do utilizador, estilos de interacção, cenários, especificações de usabilidade e prototipagem rápida. Focando-se mais neste domínio resulta numa maior usabilidade que no domínio estrutural (Hix & Hartson, 1993, pp. 6, 7).

No domínio estrutural é desenvolvido o código que implementa a componente comportamental do *design*, desde a programação, bibliotecas, tratamento de evento, circulação de dados, entre outros (Hix & Hartson, 1993, p. 7).

Cada *design* comportamental deve ser convertido num *design* estrutural que é o ponto de vista do computador e de como o comportamento tem de ser suportado.

Estes dois domínios são obrigatórios no desenvolvimento de uma aplicação. Elaborar uma interface perfeita poderá ser bastante complicado para o programador, ou seja "o que é melhor para o utilizador, raramente é o mais fácil para o *designer* de interacção ou para o programador implementar", e estes dois domínios são devidamente separados para prevenir conflito de interesses entre os dois domínios. A Tabela 1 ilustra uma comparação entre os domínios tendo em conta estes aspectos.

Tabela 1: Domínio Comportamental vs. Estrutural (adaptado de (Hix & Hartson, 1993, p. 7))

	Comportamental	**Estrutural**
O que está a ser desenvolvido	Componente interactiva da interface	*Software* da interface (para suportar a interacção)
Ponto de vista que é adoptado	Do utilizador.	Do sistema.
O que é descrito	Acções do utilizador, percepções e tarefas.	Acções do sistema em resposta ao que o utilizador efectua.
O que é envolvido	Factores humanos, cenários, representações detalhadas, especificações de usabilidade, avaliação.	Algoritmos, *callbacks*, estrutura de dados, *widgets*, programação.
Local	Onde *designers* de interacção e avaliadores aplicam o seu trabalho.	Onde implementadores de *software* aplicam o seu trabalho.
Teste	Procedimentos efectuados pelo utilizador.	Procedimentos efectuados pelo sistema.

2.5 Factores humanos

Aquando da elaboração de um sistema interactivo, há que ter em conta o comportamento do utilizador, nomeadamente os factores humanos. Aplicando estes factores pode resultar num sistema interactivo que seja de fácil aprendizagem e de uso eficiente.

Estes factores envolvem um estudo dos aspectos humanos com vista a melhorar o desempenho, a segurança, redução de custos, e melhorar a experiência do utilizador.

O conceito de factores humanos surgiu na segunda guerra mundial (Hix & Hartson, 1993, p. 17), em que era necessário considerar a interface humana para a disponibilização dos controlos aéreos e do seu ambiente visual. Tendo em conta estes factores, estes podem facilitar o utilizador na resolução das suas tarefas ou desafios, afectando a sua rentabilidade. Por exemplo, um utilizador que se encontre a pilotar um avião, e seja necessário a sua ejecção, convém que essa opção se encontre num sítio de acesso fácil e rápido para a resolução de situações mais desastrosas.

Factores humanos é um conceito que analisa as acções do utilizador, efectuando testes empíricos de modo a optimizar o desempenho, diminuindo o risco de erro e

aumentando a satisfação e conforto do utilizador. Este conceito, aplicado correctamente, pode influenciar significativamente a usabilidade de um sistema interactivo.

Para uma correcta aplicação destes factores, é necessário efectuar testes, sendo estes efectuados aos utilizadores. Estes testes são concretizados em várias fases, incluindo a formação de uma hipótese, efectuando um *design* de um estudo com utilizadores apropriados para a área do sistema, observando o utilizador a realizar certas tarefas, tais como análise de dados e, por fim, a confirmação ou negação dessa hipótese. Estes testes são efectuados por engenheiros da área de factores humanos, cientistas cognitivos, ou pessoas que tenham uma experiência vasta no estudo do desempenho do utilizador.

Tendo em conta estes factores, foi então decidido que o utilizador não deve adaptar-se ao sistema mas sim o contrário. No primeiro caso, o utilizador poderá despender mais energia e tempo para tentar perceber o sistema, enquanto no segundo caso, sendo um sistema intuitivo e de fácil aprendizagem, a taxa de erros e os custos deles serão menores.

Existem vários tipos de informação relativamente aos factores humanos, dos quais quatro influenciam significativamente o sistema (Hix & Hartson, 1993, p. 18).

- Normas de interacção com o utilizador;
- Linhas de orientação de interacção com o utilizador;
- Guias de estilo comerciais;
- Guias de estilo personalizadas.

2.5.1 **Normas**

Normas de interacção com o utilizador são documentos oficiais e públicos que fornecem requisitos para o *design* de uma interface com o utilizador. Visto serem oficiais, estes requerem que sejam seguidos adequadamente. A desvantagem do uso destas normas é o facto de serem tão superficiais que muitas das vezes torna-se complicado saber quando uma norma está a ser bem seguida ou não (Hix & Hartson, 1993, pp. 18, 19).

2.5.2 Linhas de orientação de interacção com o utilizador

Linhas de orientação de interacção com o utilizador são documentos, artigos, relatórios, livros, entre outros publicamente disponíveis, de um ponto de vista mais geral, elaborado por pessoas com experiência da área de interfaces. Estas linhas de orientação dão exemplos de senso comum, mas que requer interpretação e adaptação ao contexto específico da sua aplicação.

Uma diferença entre as linhas de orientação e as normas é o facto de as normas serem documentos onde se é "obrigado" a seguir o que é proposto, enquanto as linhas de orientação são mais sugestões para o desenvolvimento de uma boa interface.

Visto estas linhas de orientação serem sugestões, provavelmente irão surgir algumas contradições entre si. Na secção 2.7 descrevem-se algumas linhas de orientação de modo a tentar colmatar estas divergências.

2.5.3 Guias de estilo comerciais

Guias de estilo comerciais são documentos, geralmente elaborados por um utilizador ou organização, com vista a disponibilizá-lo comercialmente. Um guia de estilo normalmente contém exemplos mais concretos e úteis para o *design* de uma interface do que as normas (Hix & Hartson, 1993, p. 21).

Um guia de estilo normalmente contém uma descrição do estilo (ou objecto) de interacção específico, incluindo a aparência (*look*) e o comportamento (*feel*), e uma orientação de quando e como se deve utilizar um certo estilo de interacção (Hix & Hartson, 1993, p. 21).

A maior parte dos estilos de orientação contêm secções com as componentes que são usadas, tais como janelas, menus, controlos, e também alguns exemplos do uso das mesmas. Os guias de estilo comerciais normalmente vêm com uma ferramenta que suporta os estilos do respectivo guia. Por exemplo, os estilos podem utilizar certas componentes que se encontram numa determinada ferramenta. Estas ferramentas depois podem permitir uma personalização, mais ou menos abrangente, das suas componentes (Hix & Hartson, 1993, p. 21).

Existem alguns aspectos a ter em conta relativamente aos guias de estilo. Um deles é o facto de estes não terem um foco profissional a nível de factores humanos ou de avaliação, mas como existe algum foco a nível de usabilidade, poderá ser um bom

começo no desenvolvimento de interfaces. Outro aspecto é o facto de um guia de estilo, por si só, não garantir uma boa usabilidade numa interface. Pode disponibilizar componentes interessantes a esse nível, mas cabe ao utilizador definir a sua aplicação. Por fim, apesar de disponibilizar essas componentes, nem sempre indicam quais devem ser usadas em situações mais específicas, o que constitui um grande problema para os *designers*.

2.5.4 Guias de estilo personalizados

Guias de estilo personalizados diferem dos guias de estilo comerciais não só pelo facto de não serem comerciais, mas também por se destinarem a um certo projecto ou grupo de projectos. Estes guias contêm ajuda bastante específica para projectos específicos, são importantes no sentido de melhorar o desenvolvimento em equipa (Hix & Hartson, 1993, p. 24).

Um estilo de guia personalizado deve conter (Hix & Hartson, 1993, pp. 25, 26):

- Introdução, onde explica o paradigma básico da interface com o utilizador, o motivo da sua escolha e outros conceitos relacionados com outros projectos que influenciaram esta decisão, tais como a usabilidade, limitações a nível de *hardware* ou software, entre outros;
- Uma secção onde são indicados os dispositivos de entrada e saída utilizados;
- Um esboço da aparência do ecrã, nomeadamente o ecrã principal bem detalhado, mantendo uma consistência durante a interface;
- Uma secção separada a indicar todas as componentes que são utilizadas, o que são, como aparentam ser e como utilizá-los;
- Uma secção de mensagens onde mostra mensagens de erro, informação, aviso e outro tipo de *feedback*.

A criação destes tipos de guias requer muita experiência e trabalho. Inicialmente este documento poderá não conter muita informação acerca de certas regras ou esboços de ecrãs mas, à medida que o projecto se desenvolve, este pode e deve ser melhorado para não haver ambiguidade.

Existem vantagens na utilização destes guias de estilo personalizados. Para além de reduzir custos a nível de decisão de certas componentes a utilizar, também permite consistência, informação explícita e não ambígua para um *design*.

2.5.5 Conclusão

Um guia de estilo normalmente tende a ser mais específico a nível de aparência e comportamento do que as normas ou as linhas de orientação e, se forem bem descritas, poderá não requerer tanta interpretação, aumentado a sua consistência no *design* de interacção com o utilizador.

2.6 Princípios orientadores do *design* de interacção

Na criação de um sistema interactivo não basta haver um aspecto apelativo, ou que as funcionalidades estejam a funcionar correctamente. É necessário que o utilizador se sinta à vontade na sua utilização. Para tal foram definidos vários princípios que convém serem cumpridos para ter mais sucesso. Alguns desses princípios são o princípio da estrutura, da simplicidade, da visibilidade, do *feedback*, da tolerância e da reutilização (URL33).

O princípio da estrutura consiste em estruturar o *design* de forma consistente, fazendo com que tarefas relacionadas estejam aproximadas umas das outras, e as tarefas pouco relacionadas estejam mais afastadas.

O princípio da simplicidade consiste numa definição de tarefas que sejam de rápida utilização para o utilizador, providenciando ao mesmo, caminhos mais rápidos para tarefas mais longas.

O princípio da visibilidade consiste em disponibilizar, de forma clara, todas as funcionalidades do sistema sem que o utilizador se distraia com informação redundante ou sem importância. Bons *designs* não sobrecarregam os utilizadores com informação desnecessária ou com alternativas para a mesma funcionalidade.

O princípio do *feedback* consiste numa definição clara, concisa e não ambígua no alerta de determinadas tarefas que o utilizador possa ter efectuado, tais como, procedimentos, erros ou excepções.

O princípio da tolerância consiste numa criação de um *design* que, aquando de erros efectuados pelo utilizador, reduza o seu custo, permitindo o *undo* e *redo* das tarefas em causa.

O princípio da reutilização permite a reutilização de componentes, mantendo a sua consistência e, desta forma, facilita a memorabilidade do sistema por parte do utilizador.

2.7 Linhas de orientação

Linhas de orientação são documentos que ilustram, de um ponto de vista mais geral, sugestões no desenvolvimento de *design*s de interacção com o utilizador. Tal como indicado anteriormente, estes documentos são de senso comum, mas que são importantes no desenvolvimento.

2.7.1 *Design* centrado no utilizador

Nesta secção descrevem-se algumas linhas de orientação no *design* centrado no utilizador.

2.7.1.1 Praticar o *design* centrado no utilizador

Este conceito é de elevada importância, pois, no desenvolvimento de interfaces, há que ter em conta o ponto de vista do utilizador, mais do que o ponto de vista do sistema. Face à grande divergência de opiniões acerca dos vários tipos de utilizadores, elaborar uma interface perfeita poderá ser bastante complicado para o programador, ou seja "o que é melhor para o utilizador, raramente é o mais fácil para o *designer* de interacção ou para o programador implementar". As linhas de orientação servem para ajudar neste tipo de problemas. Este tipo de *design* requer bastante prática, experiência e tempo para uma produção mais eficaz de interfaces (Hix & Hartson, 1993, p. 29).

2.7.1.2 Conhecer o utilizador

Em seguida, há que saber o que os utilizadores pretendem, não só a nível de tarefas, mas também a nível comportamental, isto é, há que saber o comportamento dos utilizadores, efectuando entrevistas, observando a sua interacção com os sistemas. Também é necessário estudar os utilizadores, efectuando uma análise às tarefas que efectuam, e também um estudo "*time and motion*", isto é, repartir uma tarefa complexa em várias mais simples, com vista a detectar informação redundante ou dispensável. Outra maneira de perceber melhor os utilizadores é efectuando um estudo da maneira como os utilizadores estão a efectuar as suas tarefas, recorrendo ou

não a um sistema interactivo dedicado a esse estudo (Hix & Hartson, 1993, pp. 29, 30).

2.7.1.3 *Design* participativo

Outro conceito de elevada importância é o de *design* participativo por parte do utilizador. Se no desenvolvimento de sistemas interactivos, o utilizador não fizer parte, no sentido de ajudar a produzir uma boa interface, emitindo a sua opinião, a probabilidade de esse sistema interactivo ser bem-sucedido será bastante menor. Convém também que o utilizador tenha algum conhecimento acerca dos dispositivos que o sistema utiliza, pois o utilizador poderá não se sentir à vontade na sua utilização pela falta de prática no uso desses dispositivos (Hix & Hartson, 1993, pp. 30, 31).

2.7.1.4 Prevenir erros do utilizador

Prevenção de erros por parte do utilizador significa antecipar a possibilidade da sua ocorrência e, dentro do possível, evitar que os mesmos sejam cometidos. Se tal não for possível, tratá-los de maneira a que o utilizador não se sinta intimidado com certas ocorrências inesperadas.

De um ponto de vista prático, o sistema deve saber quando um utilizador se sente menos à vontade com o sistema, na medida em que o utilizador desconhece certas funcionalidades do sistema. Exemplificando, numa linha de comandos, o sistema deve disponibilizar logo à partida todos os comandos disponíveis, e para cada um deles, os seus parâmetros, seguidos de uma breve descrição sobre eles (Hix & Hartson, 1993, pp. 31, 32).

Num sistema que disponibiliza uma interface gráfica, se existirem certas operações que o utilizador esteja impossibilitado de as efectuar, se o sistema disponibilizar essa operação e o utilizador decidir efectuar, ocorrerá um erro e o utilizador poderá sentir-se intimidado acerca do erro. Se o sistema disponibilizar essa operação mas impossibilitando o utilizador de poder efectuá-la, o utilizador poderá sentir-se frustrado ao tentar perceber o motivo da opção em causa estar desactivado. A solução mais adequada é simplesmente desactivar opções inválidas. Assim o utilizador não tem que se preocupar com certos erros ou certas limitações do sistema (Hix & Hartson, 1993, pp. 31, 32).

2.7.1.5 Optimizar operações do utilizador

Optimizar significa tornar o sistema mais eficiente, no sentido de o utilizador necessitar de menos esforço para uma determinada tarefa, tais como a utilização de teclas aceleradoras, especialmente implementado para utilizadores mais experientes. Outra forma é o uso de *macros* ou abreviações para reduzir o tempo e esforço por parte do utilizador, no caso de tarefas mais morosas ou comandos mais extensos (Hix & Hartson, 1993, p. 32).

2.7.1.6 O utilizador deve ter controlo sobre o sistema

Este aspecto também é importante no sentido em que o utilizador deve sentir que controla o sistema e não o contrário. Por exemplo, o sistema em vez de requerer ao utilizador uma determinada operação de um modo imperativo deve antes esperar que o utilizador introduza essa operação. Caso contrário, o utilizador poderá sentir pressão e não terá uma boa impressão da interface (Hix & Hartson, 1993, pp. 32, 33).

2.7.1.7 Ajudar o utilizador a integrar-se no sistema

O sistema deve ajudar o utilizador a integrar-se no sistema, no sentido de fornecer informação minimalista acerca do sistema, tais como as funcionalidades principais do sistema, botões de ajuda e a sua localização, com vista a o utilizador sentir-se confortável com ele e a tornar o sistema mais útil e mais agradável para o utilizador (Hix & Hartson, 1993, p. 33).

2.7.2 **Modelo do sistema**

Modelo do sistema consiste em dar uma perspectiva do sistema ao utilizador, relativamente aos dispositivos que são utilizados, dados, operações e que informação é enviada nessas operações, produzindo assim um modelo conceptual do sistema. Após o utilizador adquirir informação sobre o modelo, este se tornará num modelo mental do utilizador, que exprime a interacção do utilizador com o sistema. Um bom modelo mental resultará num melhor aproveitamento do tempo na compreensão e na interacção com o sistema e, um modelo mental consistente facilitará o utilizador no cumprimento de tarefas (Hix & Hartson, 1993, pp. 33, 34).

Exemplificando, numa interface gráfica, o utilizador concretiza as suas tarefas primeiramente seleccionando o objecto e em seguida a acção, designado de paradigma objecto-acção, e aplica-o em toda a interface. Numa linha de comandos, o paradigma

é oposto, designado de acção-objecto: primeiramente define o comando a utilizar e em seguida o objecto alvo (Hix & Hartson, 1993, pp. 33, 34).

2.7.3 **Consistência e Simplicidade**

Manter a consistência significa que, para uma dada tarefa, se espera que tarefas similares sejam efectuadas de maneiras similares. Um utilizador espera que o sistema seja consistente em toda a sua interface, mantendo a mesma aparência e comportamento (*look and feel*). Visto a consistência ter um significado bastante extenso, outras linhas de orientação poderão entrar em conflito. Exemplificando, num processador de texto o comando para copiar texto não deve ser nomeado "duplicar" para um menu e "copiar" num menu diferente (Hix & Hartson, 1993, pp. 34, 35).

Um sistema deve ser simples, no sentido de facilitar certas tarefas mais complexas, dividindo-as em subtarefas mais simples, recorrendo a recursos normalmente utilizados pelo utilizador, Por exemplo, usar uma disquete como ícone para a opção de copiar/gravar ou uma tesoura para a opção de cortar.

2.7.4 **Questões cognitivas**

Um sistema que tem em consideração certas questões cognitivas, poderá resultar numa melhor memorabilidade, isto é, minimizar transformações mentais, no sentido de ajudar o utilizador a memorizar certas tarefas, tais como a utilização de teclas aceleradoras mais "óbvias" para determinados comandos, como por exemplo (*Ctrl-c*) para copiar e não *Shift+T*, pois torna-se mais complicada a sua memorização. Comandos mais frequentes tendem a usar teclas aceleradoras mas, usando o caso do copiar e colar, tendo em conta que começam pela mesma letra, a tecla aceleradora mais habitual é usada pela operação que é mais frequente (Hix & Hartson, 1993, pp. 38, 39).

2.7.4.1 Analogias com o mundo real

Ter em conta analogias e aplicá-las no seu sistema, tais como a utilização de ícones, imagens, metáforas, contribuirá para uma melhor memorização por parte do utilizador. Mas nem sempre esta é a melhor solução, pois nem todos os ícones podem ser compreendidos de tal forma que um utilizador consiga associar um determinado comando. Caso isto aconteça, o melhor é adicionar texto ao ícone. Há que ter atenção, na utilização dos ícones associados, pois não convém que um ícone seja agressivo de modo a que o utilizador se sinta ameaçado na utilização dessa tarefa. Por exemplo, um

botão de ajuda consegue ser representado de várias maneiras. Se for representado um humano a afogar, é possível determinar que aquele botão é de ajuda, mas o contexto da imagem pode trazer bastantes transtornos visto não se tratar de uma imagem muito agradável (Hix & Hartson, 1993, p. 39).

2.7.5 **Feedback**

Nesta secção descrevem-se algumas orientações na ilustração de *feedback* com o utilizador.

2.7.5.1 Usar feedback informativo

A transmissão de feedback por parte do sistema ao utilizador é uma componente essencial para uma melhor percepção do que o utilizador efectuou. O utilizador gosta de saber o que aconteceu quando efectuou uma determinada tarefa, para prevenir certas admirações por parte deste (Hix & Hartson, 1993, pp. 39, 40).

2.7.5.2 Usar indicadores de estado apropriados

Na realização de tarefas um pouco mais morosas por parte do sistema, o utilizador pretende que o sistema lhe indique que se encontra a efectuar essa tarefa, caso contrário o utilizador questionará se o sistema realmente se encontra a efectuar ou se entrou num estado suspenso. Para tarefas realmente morosas (com duração superior a vários segundos), essas mensagens do sistema convém que transmita informação durante o processamento das tarefas, ou o utilizador, mesmo que o sistema indique que está a efectuar essa tarefa, não sabe se essa tarefa bloqueou ou não (Hix & Hartson, 1993, pp. 40, 41).

2.7.6 **Mensagens do sistema**

Nesta secção abordam-se algumas orientações na ilustração de mensagens do sistema.

2.7.6.1 Usar mensagens centradas no utilizador e não no sistema

O modo como as mensagens são representadas também influencia o estado de espírito do utilizador, isto é, convém que as mensagens sejam perceptíveis para o utilizador perceber o que aconteceu numa determinada operação. Utilizar mensagens centradas no sistema resultará numa admiração ou pânico por parte do utilizador, e este ficará confuso se a mensagem é de alguma gravidade ou não (Hix & Hartson, 1993, p. 41).

2.7.6.2 Usar texto não ameaçador e positivo nas mensagens de erro

Mensagens de erro têm um elevado impacto psicológico no utilizador. Uma mensagem de erro tende a ser ameaçadora para o utilizador. Usar mensagens agressivas tais como, "erro catastrófico", "desastre ocorrido" não é aconselhável. Para o utilizador não revela grande utilidade, muito pelo contrário, apenas o assustará. Para isso, as mensagens devem ser positivas e não ameaçadoras, mas não com sentido de humor, pois o utilizador pode não estar de um bom humor e pode reagir negativamente a esse tipo de mensagem (Hix & Hartson, 1993, pp. 41, 42).

2.7.7 **Usar termos construtivos e específicos nas mensagens de erro**

Utilizando as mensagens "erro catastrófico" e "desastre ocorrido" como exemplos, estes certamente não representam grande utilidade para o utilizador. Em vez disso, pode levar o utilizador a pensar no motivo do erro ou levar à frustração. Estas mensagens de erro devem ser o mais construtivas e específicas possíveis para dar uma melhor percepção ao utilizador do que ocorreu de mal e para facilitar a sua correcção (Hix & Hartson, 1993, p. 42).

2.7.7.1 Fazer com que o utilizador não se sinta culpado com os erros

Quando um sistema informa de um erro, para além da construtividade que esse erro deve ter, o utilizador não pode sentir-se culpado por esse erro ter ocorrido. O sistema deve "assumir" a culpa do erro. Exemplificando, indicar um erro do tipo "comando inválido" está a culpar o utilizador pelo erro ter sido causado por ele, enquanto indicar um erro do tipo "comando não reconhecido", indica ao utilizador que o sistema não consegue reconhecer o comando e, de certa forma, está a culpar-se por esse erro ter ocorrido (Hix & Hartson, 1993, p. 43).

2.7.8 **Antropomorfização**

Antropomorfizar significa atribuir características humanas a objectos não humanos. Um dos exemplos mais populares era o agente do Microsoft Word. Esta componente foi interessante no sentido em que o utilizador podia, de certa forma, comunicar com ele, mas o problema é que o utilizador podia abstrair-se demasiado da tarefa que pretendia efectuar e prestava mais atenção à componente em si. E muitas das vezes essa componente elabora questões do tipo "Como está?", quando na realidade o computador não se interessa com o estado de espírito do utilizador, e também o objectivo dessas componentes é acompanhar o utilizador na realização de

tarefas, e não da criação de uma relação com ele. Este conceito certamente não deve ter sido em conta no *design* de interfaces (Hix & Hartson, 1993, pp. 43, 44).

2.7.9 **Modalidade e acções reversíveis**

Nesta secção descrevem-se algumas orientações a nível de modalidade das janelas e reversão de acções do utilizador.

2.7.9.1 Usar modos cuidadosamente

Um modo é um estado em que o utilizador, realizando a mesma tarefa, pode obter resultados diferentes, dependendo do estado que se encontra activo. Por exemplo, tendo a tecla *CAPS LOCK* activada produz resultados diferentes que tendo-a desactivada. Outro exemplo mais prático é a utilização do *Microsoft Paint*, em que o utilizador, clicando no ícone do rectângulo e desenhando, produz um resultado diferente que tendo o ícone do círculo activado. Existem dois tipos de modos, sendo um deles modo preemptivo, em que o utilizador precisa de completar uma determinada tarefa de uma janela para ter acesso às tarefas de outra janela; *modeless* em que o utilizador encontra-se livre de escolher as tarefas de qualquer janela que deseja efectuar, impossibilitando ao utilizador de cometer erros (Hix & Hartson, 1993, p. 44).

2.7.9.2 Tornar as acções do utilizador facilmente reversíveis

Esta funcionalidade é, provavelmente, a mais usada e das que tem mais impacto para o utilizador, permitindo ao mesmo a reversão de erros que possa ter efectuado. Esta funcionalidade é bastante útil para o utilizador, mas para o implementador é bastante complexa a sua implementação. Com esta funcionalidade o utilizador ganha mais confiança e pode explorar mais funcionalidades do sistema sem que tenha medo de efectuar erros graves (Hix & Hartson, 1993, pp. 44, 45).

2.7.10 **Chamar a atenção do utilizador**

Existem várias formas de chamar a atenção de um utilizador. Nesta secção abordam-se algumas sugestões na criação de interfaces.

2.7.10.1 Texto

Para texto é recomendado o uso de apenas dois níveis de intensidade para o mesmo ecrã. Recomenda-se a utilização de sublinhado, negrito, de forma a atrair a atenção para esse objecto. Não se deve utilizar mais de três tipos de letra, quatro tipos de

tamanhos para o mesmo ecrã, e recomenda-se a utilização de um tipo *serif* pois ajuda o utilizador a ler mais eficazmente o texto. Usar texto capitalizado apenas na primeira letra, excepto no caso dos títulos, que pode ser utilizado com todas as letras capitalizadas. Usando todo o texto capitalizado pode reduzir a velocidade de leitura em 10% (Hix & Hartson, 1993, pp. 45, 46).

2.7.10.2 Ícones a piscar

O piscar de determinadas componentes também deve ser cuidadosamente utilizado, pois é recomendado apenas às componentes mais importantes o usufruo desta funcionalidade. Tendo bastantes componentes a piscar pode causar irritação ao utilizador (Hix & Hartson, 1993, p. 46).

2.7.10.3 Áudio

O áudio é uma componente também importante no desenvolvimento de interfaces com o utilizador. Um sistema que contém sons agressivos em situações não críticas pode causar admiração e um certo receio no utilizador, ou até mesmo causar irritação. Neste caso, deve utilizar-se sons suaves para mensagens de informação ou *feedback*, e sons mais graves para mensagens de alerta ou situações críticas (Hix & Hartson, 1993, p. 46).

2.7.10.4 Cor

Cor é mais uma componente de elevada importância para o utilizador, mas também das mais sobre usadas pelos sistemas. Estes casos podem ser utilizados por aplicações para crianças, tais como bandas desenhadas, mas, apesar de existirem bastantes cores, não significa que todas sejam utilizadas. É necessário haver alguma moderação na utilização de cor. Mas existem também utilizadores com deficiência visual (cerca de 8% dos homens Caucasianos) e o conteúdo da interacção com o utilizador deve fazer sentido independentemente da cor.

Geralmente, para o mesmo ecrã, não se deve utilizar mais do que quatro cores e, caso exista texto, não mais de sete cores. Como cor de fundo recomenda-se o azul ou preto, combinado com texto de cor amarela e branca respectivamente. Não se recomenda a utilização de azul para texto, pois é das cores que torna mais complicado a sua leitura. Para o *feedback* do sistema para o utilizador, deve-se utilizar códigos de cores, ou seja, utilizar a cor verde quando tudo se encontra bem, amarelo para pequenas alertas e vermelho para avisos críticos ou mensagens de emergência.

Confundindo estas cores pode causar grande confusão ao utilizador (Hix & Hartson, 1993, pp. 46, 47, 48).

2.7.11 Questões de exibição de ecrãs

Consistência é uma das características que se deve ter em conta na criação de interfaces em ecrãs, pois um utilizador que visualize as mesmas componentes no mesmo local, para diferentes ecrãs, não sente admiração, e assim contribui para uma melhor compreensão do sistema. Na utilização de várias janelas para o mesmo sistema, as aparências das componentes do sistema, tais como botões, ícones, devem encontrar-se o mais consistentemente que for possível. Caso isto não aconteça, o utilizador terá mais problemas a acomodar-se à interface do sistema, aumentando a fadiga visual e, consequentemente, a sua produtividade.

Outra recomendação é a não utilização de informação não relevante ou desnecessária, tornando o sistema mais simplificado. Neste caso, recomenda-se que o topo e a parte inferior do ecrã contenham menos informação, e também deve existir mais de 25% de espaço vazio para prevenir a redução de desempenho por parte do utilizador. Informação relacionada também deve encontrar-se agrupada de modo a aumentar a eficiência e a eficácia do utilizador (Hix & Hartson, 1993, pp. 48, 49, 50).

2.7.12 Diferenças individuais dos utilizadores

Nem todos os utilizadores são iguais. Há uns com mais competências técnicas do que outros, e um sistema deve reagir consoante essas capacidades. O sistema deve permitir ao utilizador a alteração de preferências do sistema de modo a conhecer melhor o sistema, aumentando também o conforto e liberdade do utilizador, e estas alterações devem ser persistentes durante várias utilizações do sistema.

Existem pelo menos três tipos de utilizadores baseados na sua experiência (Hix & Hartson, 1993, p. 52):

- Utilizador novato;
- Utilizador intermitente;
- Utilizador frequente.

Um utilizador novato é aquele que não tem qualquer conhecimento do sistema, independentemente das suas capacidades técnicas. Para este tipo de utilizadores o

sistema deve ser bastante simples e conciso, retornando *feedback* informativo e mensagens construtivas, e deve disponibilizar manuais, tutoriais e demonstrações.

Um utilizador intermitente contém algum conhecimento semântico, mas não sintáctico do sistema. Para este tipo, o sistema deve fornecer tarefas simples e consistentes, funções de fácil memorização, assistência *online*, ajuda e manuais.

Um utilizador frequente contém conhecimento semântico e sintáctico, pretende uma interacção rápida, com recurso a comandos poderosos, mensagens de erro personalizáveis a nível de detalhe, *feedback* e personalização da interface.

Com prática, um utilizador novato pode tornar-se frequente, mas sem prática, um utilizador frequente pode regredir num utilizador novato.

2.7.13 **Conclusão**

Estas linhas de orientações servem apenas como consulta ou guia. Podem existir algumas mais superficiais e outras demasiado específicas em certas situações. Elas podem indicar que componentes utilizar mas nem sempre indicam o momento da sua utilização, não sendo uma solução óptima, mas sim como sugestão no desenvolvimento de interfaces.

2.8 Estilos de interacção

Nesta secção abordam-se alguns aspectos a ter em conta ao nível do *design* de interfaces, tais como a utilização de janelas, menus, formulários, caixas e interfaces gráficas.

2.8.1 **Janelas**

Uma janela é um ecrã onde são representados os objectos de interacção, tais como botões, caixas de texto, entre outros, e são os objectos a que o utilizador recorre para a interacção com o sistema. Existem várias recomendações no *design* de uma interface com o utilizador.

Em termos de utilização de janelas, estas não devem ser sobre usadas, pois pode causar irritação ao utilizador. No aparecimento de uma nova janela, esta deve ser exibida sempre na mesma localização e um utilizador deve poder reposicioná-las e redimensioná-las de acordo com as suas necessidades, aumentado o seu desempenho.

"A aparência e o comportamento da janela principal devem ser consistentes" (Hix & Hartson, 1993, p. 61). Isto significa que, quando um utilizador pretende regressar à janela principal, esta deve encontrar-se na mesma localização em que foi ultimamente utilizada.

Na realização de tarefas independentes, estas devem ser mostradas em diferentes janelas, permitindo ao utilizador a realização de múltiplas tarefas simultaneamente.

Se existirem várias maneiras de realização de tarefas, estas devem ser mostradas em diferentes janelas.

2.8.2 **Menus**

Menu é um estilo de interacção em que o utilizador pode interagir seleccionando um ou mais itens desse objecto, o que reduz a necessidade de memorização de certos termos que o sistema utiliza, pois esses termos já se encontram visíveis nesses items. Existem bastantes tipos de menus, de entre os quais se destacam (Hix & Hartson, 1993, p. 62):

- *Push-button menus*;
- Botões de rádio;
- *Checkboxes*;
- Menus;
- *Pop-up menus*;
- Botões de selecção;
- *Spinners*;
- Menus sequenciais;
- *Pie menus*;
- *Palette menus*;
- Menus embebidos;
- Menus dinâmicos.

2.8.2.1 Push-Button menus

Push-button menus são botões normalmente ilustrados numa secção exclusiva para a exibição deste tipo de menus. São comuns nos sistemas, normalmente representando texto tal como *cancel*, *ok*, *help*. Este texto deve ser suficientemente claro para não deixar ao utilizador dúvidas em relação ao que o botão realmente efectua. Na altura de

escolha de um dos botões para interacção, a mais frequente deve conter uma tecla aceleradora, tal como a tecla *Enter*, e deve mostrar uma diferente cor para indicar ao utilizador o botão que está a ter o foco. A Figura 1 ilustra um exemplo de um *Push-button* (Hix & Hartson, 1993, pp. 62, 63).

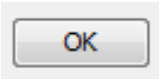

Figura 1: Push-button menu.

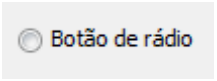

Figura 2: Botão de rádio.

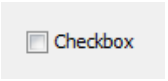

Figura 3: Checkbox.

2.8.2.2 Botões de rádio

Este tipo de botões permite ao utilizador apenas uma escolha entre várias possíveis, tipicamente utilizando o rato, e a opção seleccionada normalmente é representada com um círculo. Este tipo de menus deve ser utilizado quando o utilizador apenas deve efectuar uma escolha. A Figura 2 ilustra um exemplo de um botão de rádio (Hix & Hartson, 1993, p. 64).

2.8.2.3 Checkboxes

Checkboxes, ao contrário dos botões de rádio, permitem ao utilizador a escolha de várias opções, mas, da mesma maneira que que os botões de rádio, estes são activados pelo clique do rato. Tipicamente estes tipos de botões, quando seleccionados, representam um visto no interior do quadrado (Hix & Hartson, 1993, p. 64).

2.8.2.4 Menus

Este tipo de botões tipicamente aparece na parte superior de uma janela, em que o utilizador, clicando numa das opções disponíveis pelo sistema, apresenta em forma de lista, todas as funcionalidades relacionadas com o título da opção principal. A Figura 4 representa um exemplo de utilização de um menu (Hix & Hartson, 1993, p. 64).

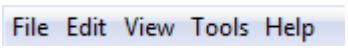

Figura 4:Menu.

2.8.2.5 Pop-up Menu

Um *pop-up* menu, ilustrado na Figura 5, é uma componente gráfica de interacção tipicamente representada aquando da utilização do botão secundário do rato por parte do utilizador. Este menu normalmente contém funcionalidades limitadas e relacionadas com o objecto seleccionado.

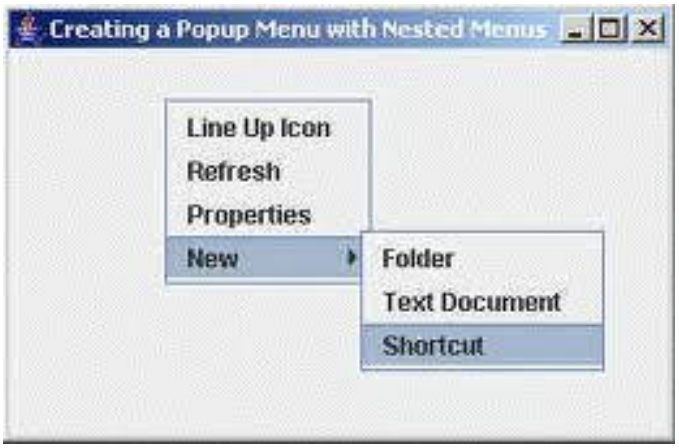

Figura 5: Pop-up Menu.

2.8.2.6 Botões de selecção

Este tipo de botões representa-se como um campo em que são disponibilizadas múltiplas opções em forma de lista, contendo botões de navegação entre as várias opções quando a lista é extensa o suficiente para não representar, todas a opções dessa lista. A opção seleccionada é representada nesse campo.

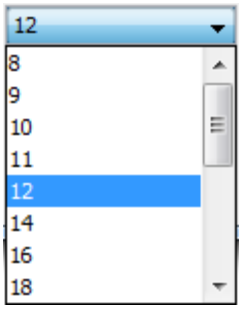

Figura 6: Botão de selecção.

2.8.2.7 Spinners

Um *spinner*, à semelhança do *option menu*, representa-se por um campo com o seu valor escolhido visível. A diferença entre estes botões reside no facto de os itens da lista de valores serem acedidos rotativamente. A desvantagem na utilização deste tipo de menus é o facto de os valores não serem todos representados simultaneamente. A Figura 7 ilustra um *spinner*.

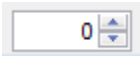

Figura 7: Spinner.

2.8.2.8 Menus hierárquicos

Menus hierárquicos comportam-se de maneira semelhante aos menus. Quando um utilizador selecciona algumas das opções desses menus, é apresentado um outro menu de forma sequencial com a lista de opções referentes à opção seleccionada anteriormente. Estes tipos de menus tipicamente são representados com uma seta no lado direito, de forma a indicar a localização do submenu. A Figura 8 ilustra um exemplo de um menu hierárquico.

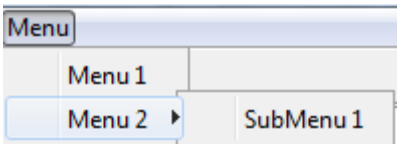

Figura 8: Menu hierárquico.

2.8.2.9 Pie menus

Pie menus são componentes disponibilizadas em forma circular, particularmente útil quando existe um número pequeno de opções, pois se o número de opções for elevado, não haverá espaço para representar a descrição de cada selecção. Cada opção desta componente pode ser disponibilizada usando cores diferentes. A Figura 9 representa um exemplo de um *Pie menu*.

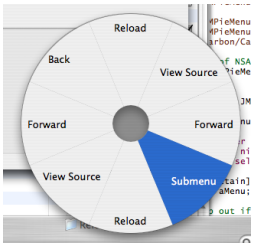

Figura 9: Pie Menu (URL34).

2.8.2.10 Palette menus

Palette menus são componentes representadas por ícones, ao invés de uma descrição textual que têm a mesma representação dos *push-button menus* mas estes encontram-se agrupados usando exclusividade mútua. O botão seleccionado normalmente contém uma cor diferente dos restantes. A Figura 10 ilustra um exemplo do uso de *Palette menus*.

Figura 10: Palette Menu.

2.8.2.11 Menus embebidos

Menus embebidos são mais conhecidos como hiperligações. Estas são referências, em hipertexto. Utilizando menus embebidos é possível produzir interfaces não lineares, recorrendo a texto, imagem ou outros objectos.

2.8.2.12 Menus dinâmicos

Menus dinâmicos são componentes cujas opções podem variar durante a execução de um programa. Um dos exemplos é o uso de componentes inactivas que o utilizador não pode seleccionar (Hix & Hartson, 1993, p. 69).

2.8.3 **Linhas de orientação no *design* de menus**

Esta secção apresenta algumas recomendações no *design* de menus.

2.8.3.1 Usar agrupamentos coerentes nos menus

Agrupar funcionalidades relacionadas pode aumentar a eficiência do sistema. Exemplificando, todas as funcionalidades relativamente à alteração do tipo de letra devem estar agrupadas (Hix & Hartson, 1993, p. 70).

2.8.3.2 Ordenar logicamente menus.

Existem três tipos de ordenação: numérica, alfabética ou temporal. A ordenação alfabética é mais útil para um grupo de opções alargado. A ordenação temporal consiste em ordenar um grupo de opções pela sua ordem de utilização. Exemplificando, a opção "Abrir" deve aparecer primeiro que a de "Fechar" pois o utilizador tende primeiro a utilizar a opção "Abrir" (Hix & Hartson, 1993, p. 70).

2.8.3.3 Utilizar descrições breves nos menus

A utilização de descrições nos menus afecta a decisão do utilizador. Recomenda-se a utilização de palavras únicas quando uma dada funcionalidade do menu for óbvia (Hix & Hartson, 1993, pp. 70, 71).

2.8.3.4 Usar um *layout* consistente em todos os menus

A definição de um *layout* deve-se encontrar claramente definida num guia de estilo comercial, para suportar as expectativas do utilizador. Recomenda-se a utilização de espaço em branco para a separação entre grupos de opções, mas a sua utilização em demasia pode levar a uma distracção por parte do utilizador. A descrição desse grupo de opções deve ser consistente, e deve ser visível no mesmo local para cada menu. O mesmo acontece também para mensagens de erro, contendo uma descrição concisa para diferentes erros (Hix & Hartson, 1993, p. 71).

2.8.3.5 Permitir atalhos

A utilização de atalhos permite ao utilizador a escolha de opções sem necessitar de fazer um esforço maior o acesso a essas opções. Esta funcionalidade é bastante útil em *pull-down* menus (Hix & Hartson, 1993, p. 71).

2.8.4 **Formulários**

Um formulário é um ecrã que contém múltiplos campos que o utilizador deve preencher. Estes campos podem ser preenchidos de forma aleatória e podem existir campos obrigatórios, em que o utilizador necessite de os preencher, e campos opcionais, em que não é necessário o seu preenchimento.

Também existem outros tipos de campos, tais como os campos em que já existe um valor por omissão, mas que o utilizador pode alterar, se assim o entender; campos em que é necessário a escrita de texto; e campos dependentes, que são representados quando certos valores de certos campos são atingidos. A Figura 10 ilustra um exemplo de um formulário.

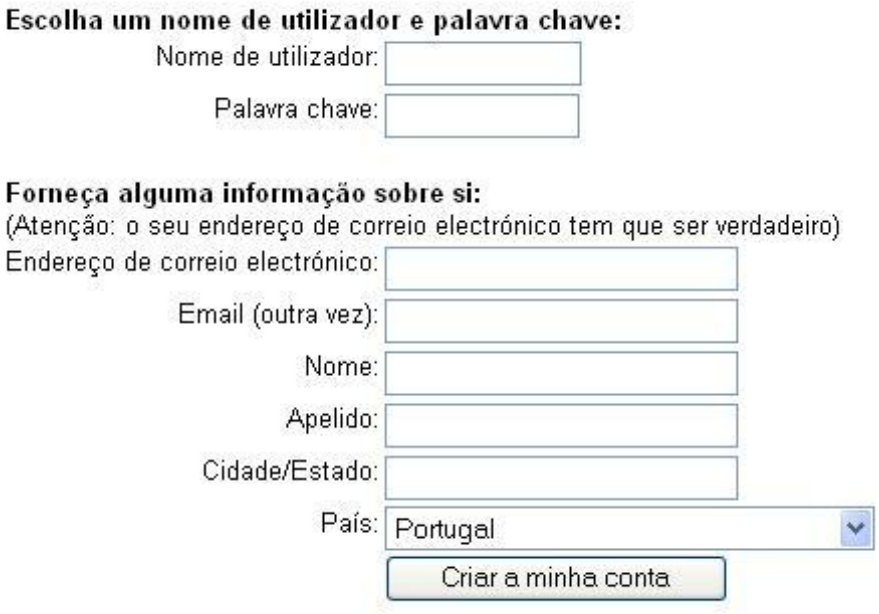

Figura 11: Exemplo de um formulário (retirado de URL31).

2.8.5 **Guias de *design* de formulários**

Nesta secção apresentam-se algumas recomendações no *design* de formulários.

2.8.5.1 Usar um *layout* e conteúdo consistente e visualmente apelativo

Todos os formulários devem conter um título breve mas conciso, contendo instruções compreensíveis. Os campos devem se encontrar devidamente agrupados e separados por determinadas aparências visuais, tal como espaço em branco (Hix & Hartson, 1993, pp. 73, 74).

2.8.5.2 Usar aparências visuais apropriadas para campos nos formulários

A aparência de determinados campos ao utilizador influencia na compreensão desses campos. Exemplificando, o código postal num formulário pode ser representado por uma caixa de texto, onde o utilizador insere 4 dígitos, seguido de um traço "-", seguido de outra caixa de texto onde o utilizador insere 3 dígitos, para facilitar a inserção do utilizador. Campos obrigatórios e opcionais podem ser diferenciados pela presença de um asterisco "*".

2.8.5.3 Suportar a edição e correcção de campos

O utilizador deve poder editar ou corrigir campos aos quais não se encontrem devidamente preenchidos de maneira a não seja necessário de voltar a escrever tudo. O sistema deve permitir a inserção de novos conteúdos de um campo (Hix & Hartson, 1993, p. 76).

2.8.5.4 Utilizar mensagens de ajuda em campos de inserção de dados

Para campos de maior dificuldade de compreensão, o sistema deve disponibilizar uma ajuda ao utilizador, providenciando uma breve descrição do campo em causa. Estes campos devem ser concisos de maneira a não obstruir a atenção do utilizador, e devem ser consistentes na sua localização (Hix & Hartson, 1993, p. 76).

2.8.5.5 Usar componentes de finalização nos formulários

Aquando da conclusão do preenchimento de um formulário por parte do utilizador, deve existir uma componente de finalização da inserção de dados, normalmente recorrendo a um botão *OK*, permitindo ao utilizador a revisão e/ou alteração dos campos do formulário (Hix & Hartson, 1993, p. 77).

2.8.6 **Caixas**

Uma caixa é um ecrã tipicamente rectangular que serve para solicitar texto ao utilizador, mostrar *feedback* por parte do sistema ou representar determinada

informação. Existem pelo menos três tipos de caixas: caixas de entrada, caixas de mensagem e caixas de diálogo.

Uma caixa de entrada contém as funcionalidades básicas de edição de texto e serve para requisitar *input* ao utilizador. A Figura 11 ilustra um exemplo da aplicação de uma caixa de entrada.

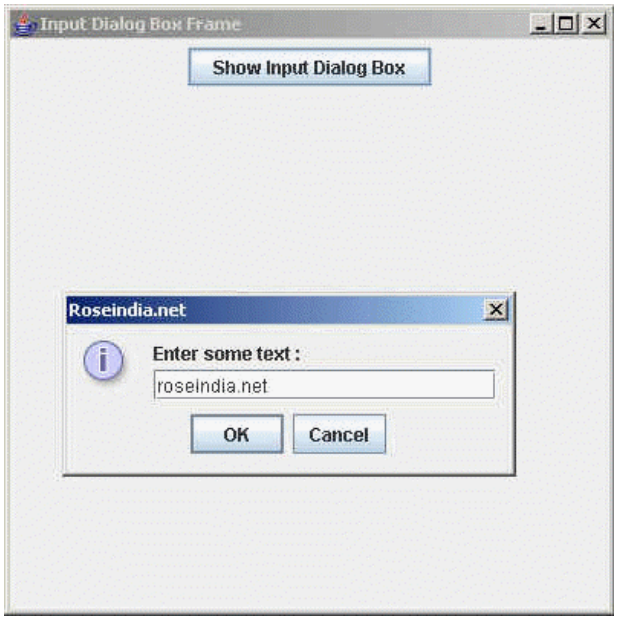

Figura 12: Caixa de entrada (URL35).

Uma caixa de mensagem é uma caixa onde requer *input* ao utilizador. Este *input* é limitado e quando o utilizador selecciona uma opção a caixa é fechada. Nesta caixa é possível exibir mensagens informativas, avisos, ou determinadas acções a serem confirmadas pelo utilizador. Este tipo de caixas usualmente é preemptivo. A Figura 12 ilustra um exemplo de uma caixa de mensagem.

Figura 13: Caixa de mensagem (URL8).

Uma caixa de diálogo é um objecto de interacção onde pode conter outros objectos de interacção tais como botões, listas, entre outros. Esta normalmente aparece aquando da selecção de uma opção de um *pull-down* menu ou do uso de uma tecla aceleradora. Estas caixas podem ser modais ou *modeless*. Nas caixas modais, o utilizador deve tomar determinadas acções para aceder a outras futuras funcionalidades. Nas caixas *modeless* o utilizador não necessita do preenchimento de determinados campos para aceder a outra caixa.

2.9 Metodologia de desenvolvimento de interacção com o utilizador

Para um *designer* de interfaces, é vantajoso seguir metodologias de desenvolvimento, pois aumenta a eficiência e organização no desenvolvimento de software. Nesta secção aborda-se a metodologia de desenvolvimento em estrela.

2.9.1 Desenvolvimento em estrela

Esta metodologia introduziu o conceito de iteração. Este tipo de metodologia tanto suporta os conceitos de *top-down* e *bottom-up*, como o conceito de *inside-out* e *outside-in*. Desenvolvimento *inside-out* implica um desenvolvimento mais fechado, no sentido que existe pouca interacção com o utilizador e assume-se o pressuposto de o sistema vai ser bem-sucedido, enquanto o desenvolvimento *outside-in* inicialmente tende a haver uma boa comunicação com o utilizador de modo a saber os requisitos exactos do mesmo. A Figura 14 ilustra o ciclo de vida em estrela para o desenvolvimento de interacção.

Com este tipo de desenvolvimento, o *designer* pode começar o desenvolvimento de interfaces em qualquer fase, e passar para outra fase à escolha. Visto estas fases estarem fortemente interligadas, convém que haja testes de usabilidade na passagem de uma fase para outra, estando este tipo de desenvolvimento centrado na avaliação. Estas fases podem ser:

- Análise funcional, do sistema e do utilizador
- Especificação de requisitos
- *Design* e a sua representação
- Prototipagem Rápida

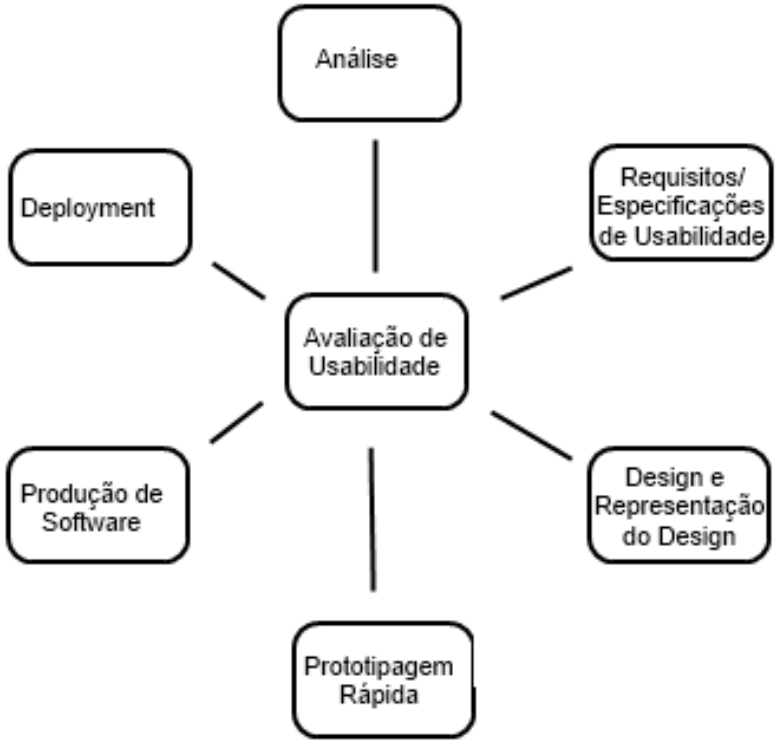

Figura 14: Ciclo de vida em estrela (adaptado de (Hix & Hartson, 1993)).

2.9.1.1 Prototipagem Rápida

A necessidade de interacção com o cliente para efeitos de *feedback* a nível de *design* é um dos factores mais importantes para a prevenção de tempo e dinheiro desnecessário. Para este *feedback* tende-se a representar ao cliente um sistema bastante detalhado, o que pode causar ao cliente bastantes insatisfações, e para o *designer* torna-se um grande desperdício de recursos.

O conceito de prototipagem rápida veio colmatar este problema, na medida em que se apresenta ao cliente, apenas o *design* essencial, consumindo menos recursos e havendo mais tempo para mudanças mais significativas, com vista a melhorar a usabilidade do sistema.

Num *design* iterativo, recorrendo às linhas de orientação pode não ser suficiente para o sucesso de um sistema interactivo, nem sempre o utilizador concorda com essas linhas. Sendo assim, os *designers* devem adoptar o método de artilharia, que consiste num ciclo de vida no contexto de prototipagem, sendo as fases *Ready*, *Fire* e *Aim*. *Ready* refere-se à fase de *design* da interface, *Fire* significa a implementação do

protótipo e *Aim* a avaliação e análise. A Figura 15 ilustra o diagrama do método de artilharia.

Deve-se ter em conta esta técnica na realização de um sistema interactivo, pois melhora a eficiência a nível de recursos, aumenta a usabilidade do sistema, sendo ideal esta técnica ser utilizada nas fases iniciais do sistema interactivo.

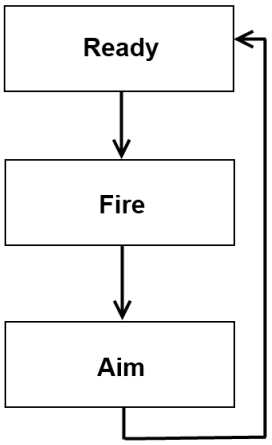

Figura 15: Método de artilharia.

2.10 Conclusão

Neste capítulo efectuou-se um estado da arte acerca da interacção Humano-Computador. Inicialmente descreveu-se alguns aspectos gerais acerca de interfaces, usabilidade e algumas definições do mesmo. Em seguida abordou-se o conceito de factores humanos, normas de interacção com o utilizador, linhas de orientação com o utilizador e guias de estilo comerciais. Descreveram-se também os vários estilos de interacção utilizados, tais como janelas, menus, formulários e caixas, procedeu-se a uma análise das metodologias de desenvolvimento de *software* e por fim abordou-se o conceito de prototipagem rápida.

3 Sistemas de Escalonamento

Neste capítulo, pretende-se realizar o levantamento e sistematização das características genéricas dos interfaces de sistemas de planeamento e controlo da produção e escalonamento. Neste sentido foram analisados os sistemas *SCADA* (URL1), o sistema *Lekin* (URL2), e o sistema *Izaro* (URL4), quer ao nível das ferramentas de escalonamento quer de simulação de *layouts* fabris. Foi ainda realizada a análise do sistema desenvolvido no âmbito do projecto de investigação *AutoDynAgents*, considerando que este servirá de base para a especificação e desenvolvimento para o sistema ADSYS (URL3).

3.1 Sistrade® SCADA & *Shop Floor Control*

A SISTRADE propôs um conjunto de soluções tecnológicas, no ramo dos Sistemas de Informação Empresariais, mais precisamente um *Manufacturing Execution System (MES)* para suporte à supervisão industrial e planeamento e controlo da produção, *design*ado comercialmente por Sistrade SCADA & *Shop Floor Control* (URL1).

O sistema *Shop Floor Control* é mais dedicado apenas ao controlo em tempo real todo o processo produtivo, tais como os materiais e, desta forma, haver um controlo mais eficiente da produção (URL1).

Este sistema contém vários projectos, sendo os mais relevantes:

- SCADA – Supervisão Industrial;
- Planeamento de Produção.

O sistema "Scada – Supervisão Industrial" é uma ferramenta do *software* Sistrade® ERP e tem como principal foco a supervisão do estado da produção e o estado de funcionamento de cada recurso, recorrendo a ilustrações gráficas (URL1).

A imagem da Figura 16 ilustra as várias componentes que esta ferramenta dispõe. Destacando a componente de Supervisão, onde o sistema desenha o *layout* fabril, esta componente é desenhada com base em parâmetros recebidos anteriormente, e o sistema desenha conforme esses parâmetros. O protótipo desenvolvido no âmbito desta tese de mestrado apresenta uma vantagem em relação à funcionalidade do sistema Sistrade na medida em que, no protótipo, o nível de personalização do *layout*

fabril é superior, pois utilizam-se e personalizam-se elementos de desenho para a definição do respectivo *layout*.

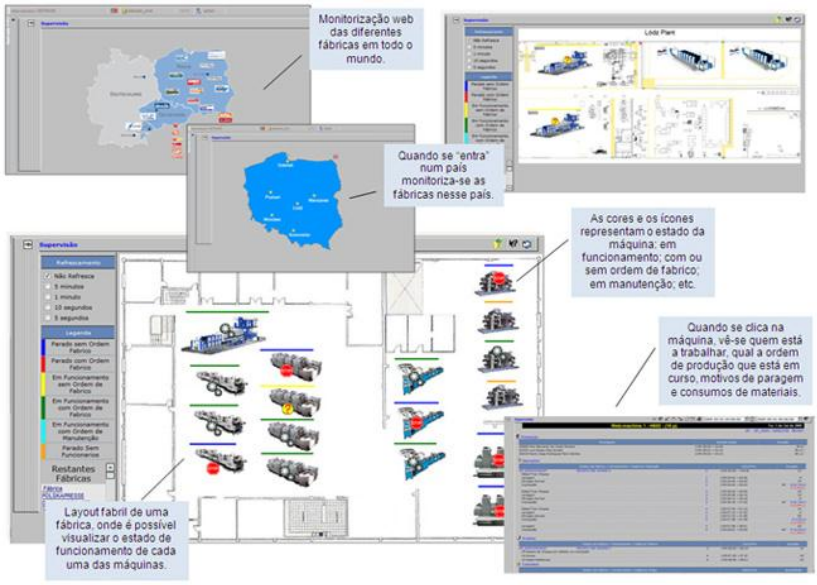

Figura 16: Supervisão (SCADA) global e local – multi-fábrica (URL1).

O sistema de planeamento de produção, tal como o nome indica, permite o planeamento do processo de produção, tais como a calendarização de tarefas ou o escalonamento das ordens de fabrico ou encomendas (URL1).

A imagem da Figura 17 ilustra várias das componentes deste sistema.

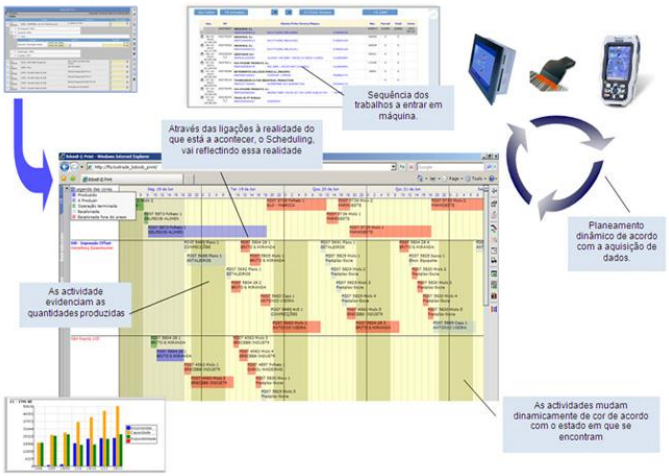

Figura 17: Consulta de ordens de fabrico planeadas de forma analítica/gantt (URL1).

3.2 Sistema Lekin

O sistema LEKIN (URL2), é uma ferramenta académica para escalonamento da produção desenvolvido no âmbito de um projecto de investigação liderado pelo professor Michael L. Pinedo na *Stern School of Business* da Universidade de Nova Iorque. Este sistema contém várias funcionalidades, tais como (URL2):

- Seis ambientes de *workspace*;
- Exemplos de problemas;
- Heurísticas e outras regras;
- Suporte a diagramas de Gantt com recurso a arrastamento;
- Análise comparativa de diferentes técnicas de escalonamento;
- Impressão completa de gráficos;
- Importação, exportação e anexação de algoritmos externos.

A imagem da Figura 18 ilustra um exemplo de inserção de dados. Nestas caixas de diálogo o sistema permite a inserção de um identificador a uma tarefa, comentários, e também a data de lançamento e de entrega, juntamente com o peso. Também é possível definir a rota de uma tarefa.

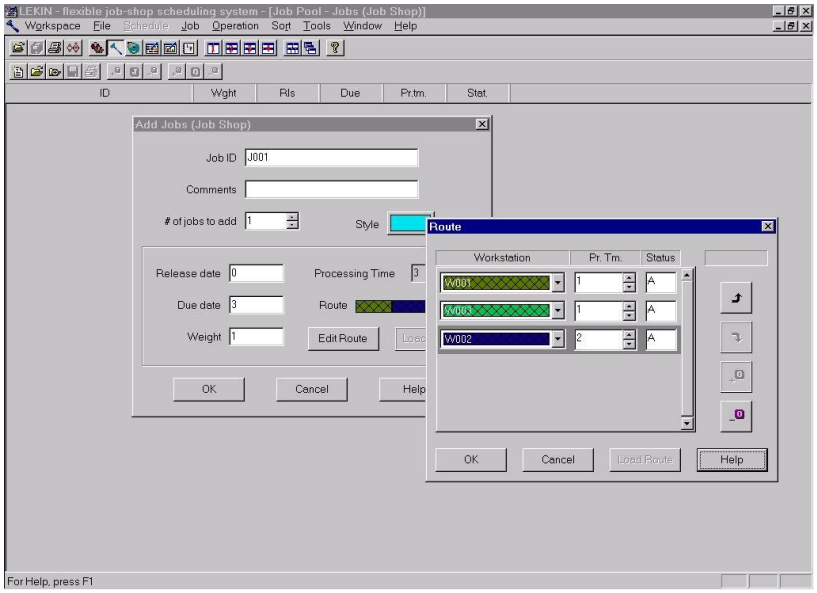

Figura 18: Inserção de dados do sistema LEKIN (URL2).

Em termos de interface e usabilidade, apresenta uma interface simples, de fácil compreensão, mas as cores usadas poderão não trazer muito conforto ao utilizador.

A imagem da Figura 19 ilustra um gráfico de Gantt referente a um processo de escalonamento do sistema LEKIN, onde se visualizam as operações a serem efectuadas, a ordem de processamento das mesmas e em que máquinas são processadas. Também é possível a alteração das datas de lançamento, data de entrega, peso e estilo.

Em termos de interface e usabilidade, este sistema de escalonamento não está muito enriquecido, as datas das respectivas tarefas encontram-se em modo numérico ao invés em data, o que não trata problemas reais, mas suporta arrastamento das tarefas. Para a definição de máquinas, tarefas, operações, entre outras, cada uma destas funcionalidades requer uma subjanela, o que poderá causar alguma confusão, em alguns casos pelo elevado número de janelas abertas.

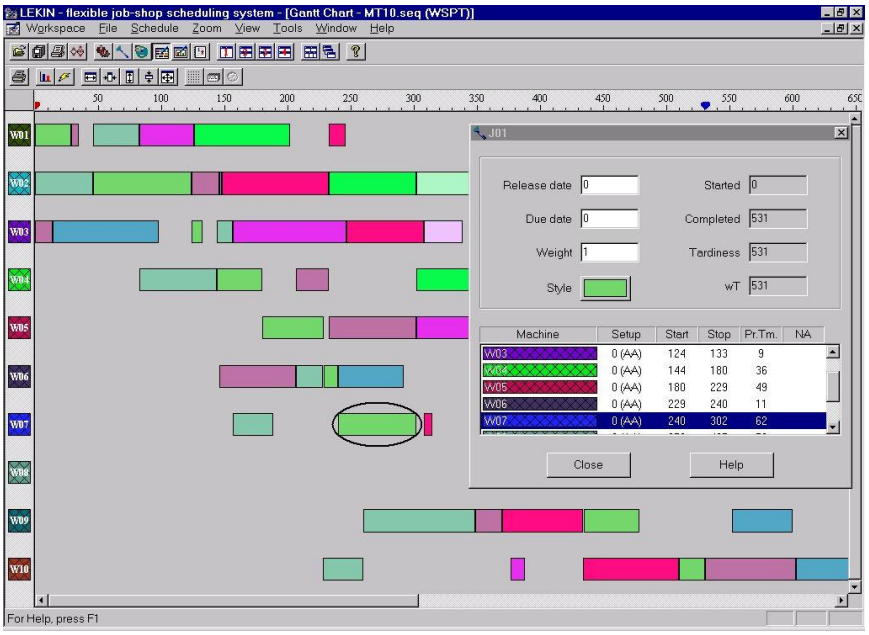

Figura 19:Sistema de escalonamento LEKIN (URL2).

3.3 Ferramenta Izaro

A Softi9 disponibiliza um conjunto de soluções modulares na área dos Sistemas de Informação Empresarias nomeadamente o IZARO ERP (Softi9, n.d.), um Sistema Integrado de Gestão (ERP II) indicado para a generalidade dos sector de actividades,

desde a Indústria, ao Comércio ou Serviços e o sistema IZARO APS (Softi9-2, n.d.) para o Escalonamento e Optimização da Produção. A imagem da Figura 20 ilustra o menu principal da ferramenta IZARO ERP, onde demonstra uma interface intuitiva e amigável.

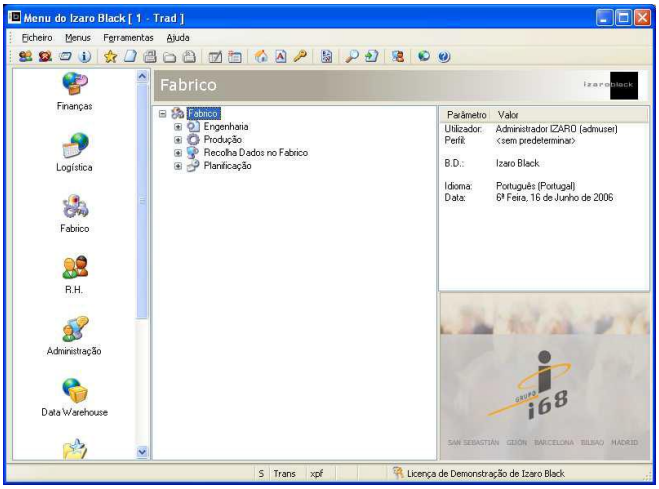

Figura 20: Menu principal do sistema IZARO ERP (Softi9, n.d.).

A Izaro APS é uma ferramenta gráfica para o suporte ao planeamento e optimização da Produção em Empresas de fabrico repetitivo. A Figura 21 ilustra um exemplo de um gráfico de Gantt utilizando a ferramenta Izaro APS. Esta figura apresenta uma interface simples e intuitiva, recorrendo a diferentes cores para diferentes recursos.

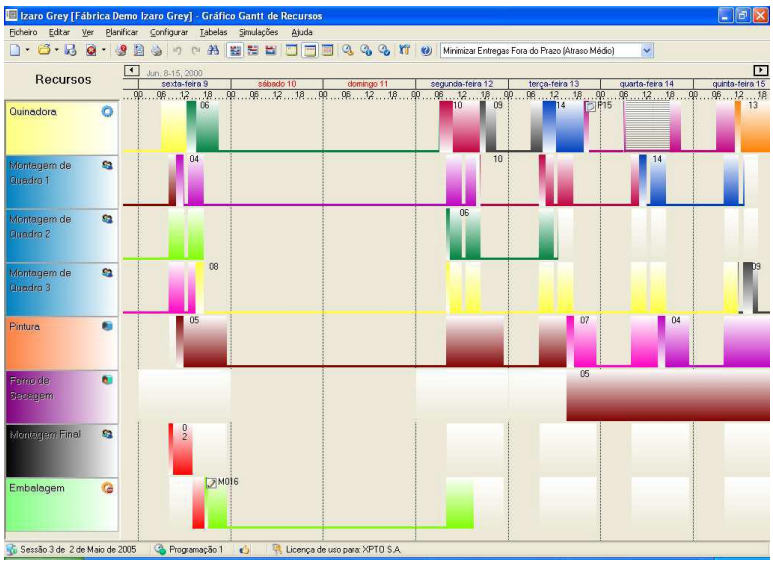

Figura 21: Gráfico de Gantt da ferramenta Izaro APS (Softi9-2,n.d.).

O planeamento de produção de capacidade finita recorre a critérios de optimização e ordenação e também a uma interface gráfica e intuitiva para este efeito, onde são ilustrados gráficos de Gantt, podendo estes serem personalizáveis e configuráveis o nível de movimentos manuais.

A leitura dos dados pode ser feita de forma manual, a partir das máquinas, com recurso a terminais de recolha de dados no fabrico ou com terminais que se encontram ao pé das máquinas (URL4).

Também suporta a funcionalidade de obtenção dos registos, onde são disponibilizadas capacidades de monitorização de fábrica (URL4).

3.4 AutoDynAgents

O *AutoDynAgents* (*Autonomic Agents with Self-Managing Capabilities for Dynamic Scheduling Support in a Cooperative Manufacturing System*) é um sistema de escalonamento que utiliza sistemas multiagentes no suporte ao escalonamento dinâmico e distribuído em sistemas de fabrico (Madureira et al, 2008).

A Figura 22 ilustra a arquitectura do sistema AutoDynAgents.

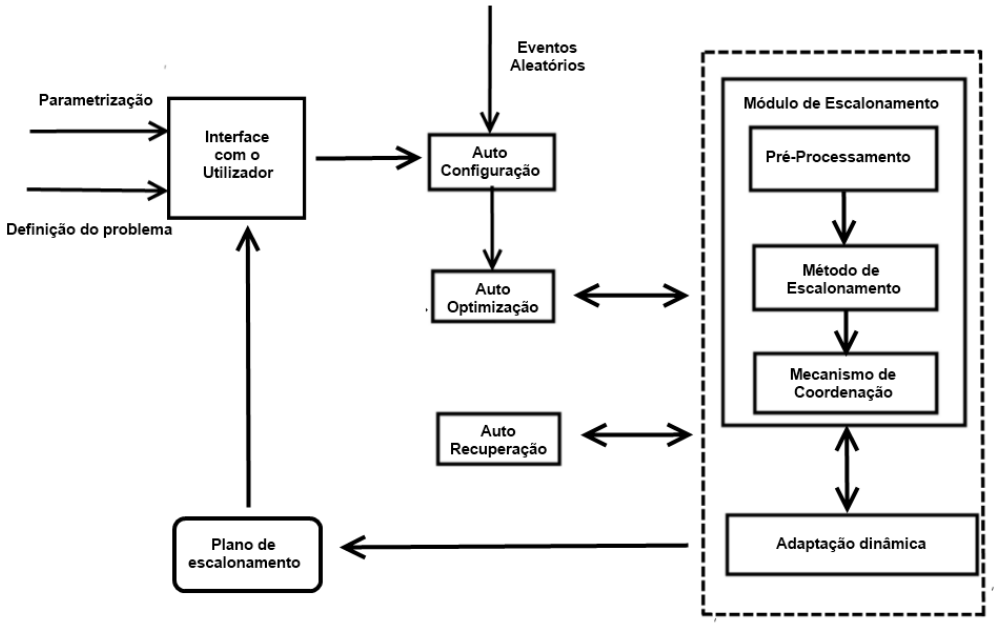

Figura 22: Arquitectura do sistema AutoDynAgents (Adaptado de Madureira et al. 2011)

No âmbito desta tese de mestrado trabalha-se a componente da interface com o utilizador. Contém uma componente de especificação do problema, onde o utilizador especifica os recursos referentes ao processo de planeamento de produção, tais como as máquinas a utilizar que constitui o *layout* fabril, tarefas a serem processadas e respectivas operações. Para as tarefas são especificadas a descrição, a data de lançamento, a data de entrega e o peso (prioridade). Na especificação das operações, o utilizador especifica a máquina onde será processada e as precedências entre as várias operações de uma tarefa. Contém funcionalidades de importação e exportação de dados, recorrendo a ficheiros XML para o seu armazenamento. Para as operações existe uma visão gráfica onde o utilizador pode visualizar mais intuitivamente as tarefas, respectivas operações e precedências entre estas. Esta informação também pode ser visualizada de forma hierárquica. A Figura 23 ilustra a interface desta componente.

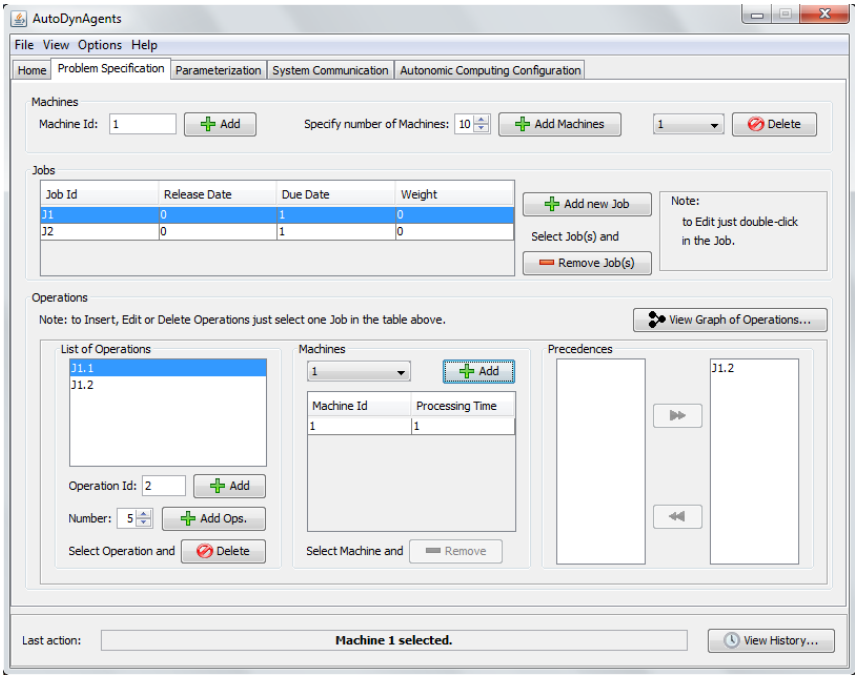

Figura 23: Especificação do problema do sistema AutoDynAgents.

Existe também uma componente onde o utilizador especifica as técnicas que são utilizadas para o escalonamento com base nos dados inseridos na componente de especificação do problema. Cada técnica de escalonamento contém diferentes campos

a preencher e cabe ao utilizador decidir a técnica a utilizar e existem várias técnicas, sendo elas a Pesquisa Tabu, utilizando Algoritmos Genéticos, *Simulated Annealing*, Colónia de Formigas, *Particle Swarm Optimization* e Optimização por Colónia de Abelhas. A Figura 24 ilustra a interface desta componente.

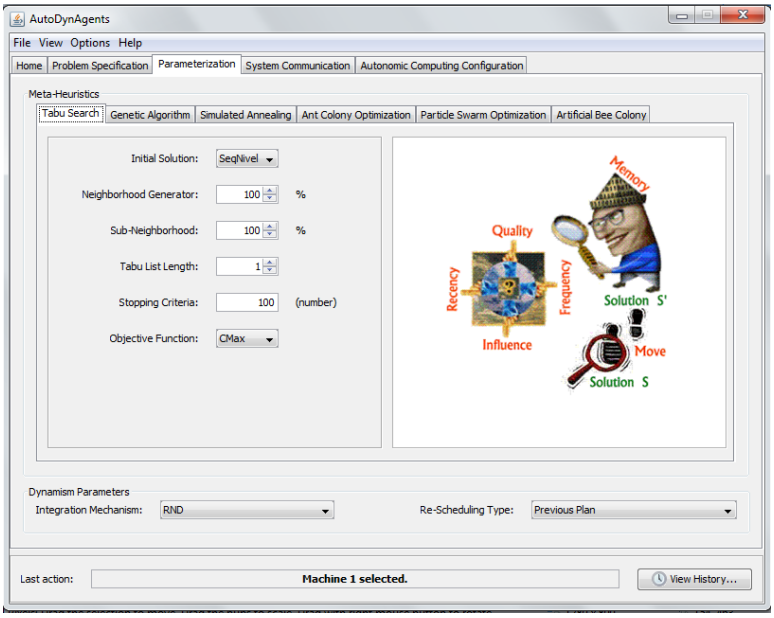

Figura 24: Parametrização do sistema AutoDynAgents.

Na funcionalidade de *System Communication* o utilizador selecciona a técnica de escalonamento que pretende usar para o processo de optimização e o número de simulações a realizar. Após o processamento da técnica de escalonamento seleccionada pelo utilizador, são visualizados os resultados para cada execução. Estes resultados podem ser visualizados em forma de tabela (relatório) ou num diagrama de Gantt. A Figura 25 ilustra a interface desta componente.

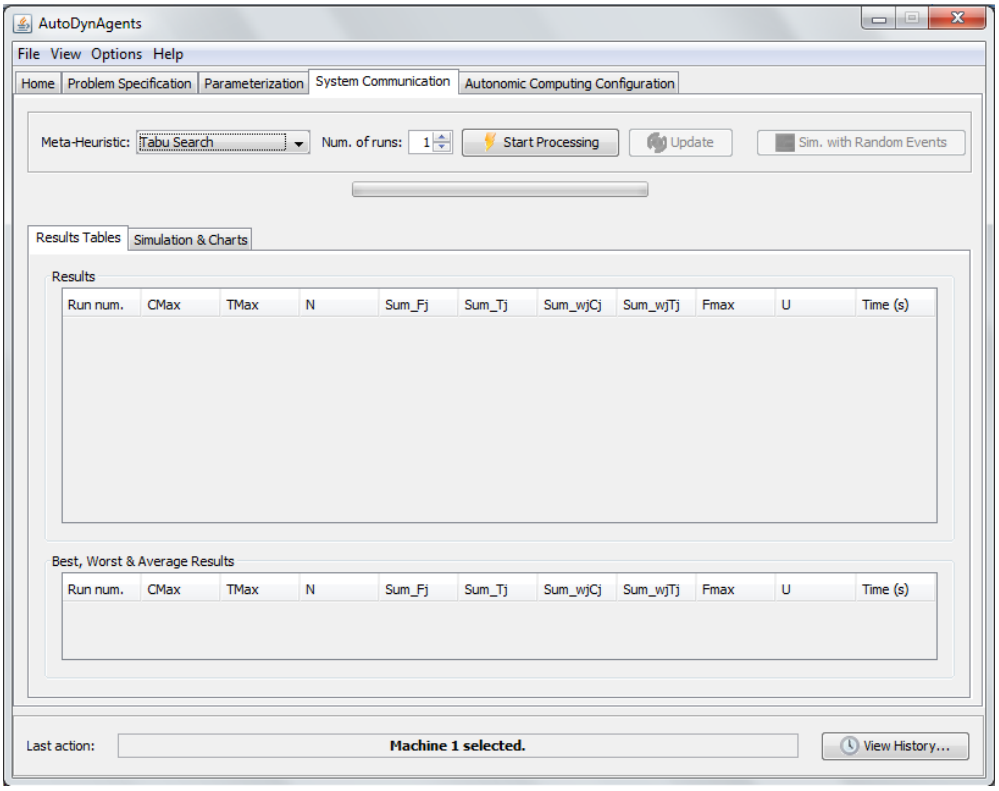

Figura 25: Funcionalidade System Communication do sistema AutoDynAgents.

Existem também operações mais complexas, para utilizadores mais avançados e peritos, que se referem aos mecanismos de cooperação e de negociação. O sistema recorre a agentes, que na altura do processamento da solução, irão cooperar e negociar entre si.

3.4.1 Aspectos que podem ser melhorados

Da análise realizada foram identificados vários aspectos que poderão ser melhorados em termos de usabilidade e interacção com o utilizador.

- Aquando da escolha do método de escalonamento, o utilizador pode por si mesmo efectuar essa escolha, ou, caso se encontre indeciso, solicitar ao sistema a recomendação de um método com base em experiência anterior.
- O sistema actual apenas gera o diagrama das tarefas, lendo o ficheiro de input. Uma possível melhoria seria o processo desse diagrama ser gráfico e interactivo, podendo efectuar toda a estrutura do problema de uma forma gráfica, sendo mais intuitivo e apelativo para o utilizador.

- O sistema deverá permitir a configuração de técnicas de escalonamento usando como base uma configuração pré-definida (valores por omissão escolhidos pelo sistema), dando uma ideia de cooperação do sistema.
- Uma outra funcionalidade seria a de permitir ao utilizador a interacção com os gráficos gerados após o processamento no sentido de sugerir alterações pontuais aos resultados obtidos e, desta forma, gerar feedback para o sistema e permitir a aprendizagem e adaptação deste às recomendações do utilizador.
- Por fim, o sistema pode ter funções de undo e redo para o utilizador, caso se engane, não corrigir certas operações manualmente, o sistema efectua automaticamente sem necessidade de um esforço extra por parte do utilizador.

3.5 Conclusão

Neste capítulo procedeu-se ao levantamento de alguns sistemas de planeamento e controlo de produção existentes descrevendo-se, de uma forma resumida, algumas características, e procedendo a uma breve análise de usabilidade e interacção destes sistemas. Também se efectuou um estudo mais alargado do sistema *AutoDynAgents*, detalhando algumas funcionalidades e também melhorias a poderem ser efectuadas neste sistema, as quais foram tidas em conta na realização deste trabalho de mestrado.

4 Análise de tecnologias

Neste capítulo será sistematizado o levantamento de API's e bibliotecas Java para a criação de interfaces com o utilizador e gráficos. Foi realizada uma análise detalhada de algumas ferramentas de criação de gráficos, identificando as funcionalidades de cada sistematizadas numa tabela comparativa.

O levantamento e análise de ferramentas de criação de interfaces gráficas foram concretizados pela identificação de funcionalidades, vantagens e limitações acompanhadas com a ilustração através de um exemplo, comum a todas as API's, do uso das componentes que cada ferramenta disponibiliza.

O capítulo termina com a recomendação de uma ferramenta para a criação de gráficos e também de uma ferramenta ou API (*Application Programming Interface*) para a criação de interfaces gráficas, a utilizar no âmbito desta tese de mestrado.

4.1 Ferramentas de criação de gráficos

Nesta secção procede-se ao levantamento e análise das ferramentas de criação de gráficos, tais como o VARCHART JGantt (URL5), o JFreeChart (URL7), o JavaGantt (URL10) e o JideChart (URL23).

4.1.1 VARCHART JGantt

VARCHART JGantt (URL5) é uma API de criação e desenvolvimento de aplicações de escalonamento. Foi desenvolvida pela *Necrotic Software GmbH*, uma empresa dedicada ao desenvolvimento de diagramas de Gantt, e contém outras aplicações relacionadas, tais como o XGantt, XTree e XNet.

Possui uma interface bastante intuitiva e é uma ferramenta útil de planeamento para os utilizadores. Esta ferramenta tem sido melhorada ao longo dos anos e também pode ser utilizada em áreas como o planeamento e controlo de produção, a logística, a gestão de projectos e de serviços.

É possível personalizar todas as componentes desta ferramenta, de modo a satisfazer os requisitos do utilizador, tais como a inclusão de gradientes nas componentes, tornado a visualização mais apelativa (URL5).

É uma ferramenta de fácil integração com as aplicações Java. Também é possível apresentar em diferentes idiomas, pois utiliza o sistema *unicode* para representar caracteres especiais (URL6). A Figura 26 e Figura 27 ilustram exemplos de aplicação desta ferramenta. Nestas figuras é possível visualizar, do lado esquerdo os dados referentes às tarefas, onde contém uma descrição e datas de início e de fim.

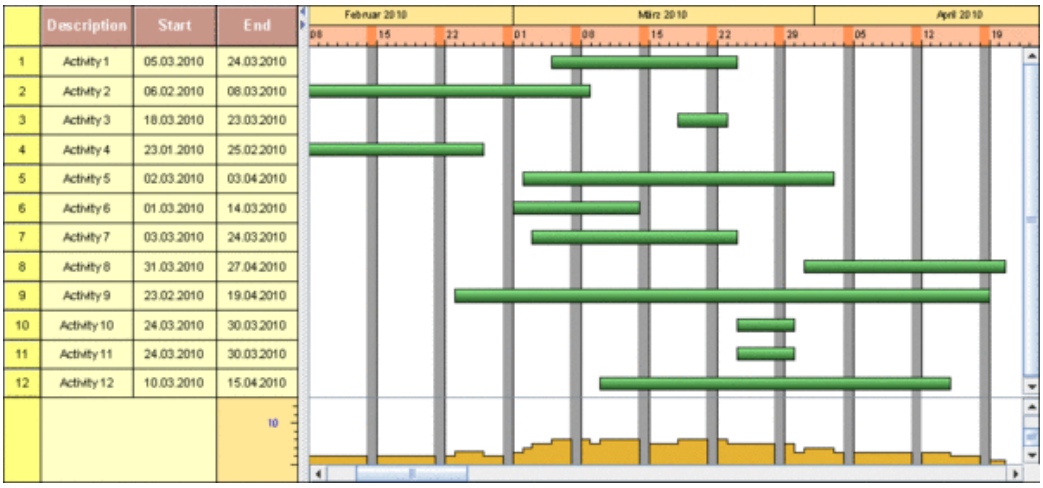

Figura 26: Exemplo de gráfico VarChart Gantt (URL6).

A Figura 27 ilustra um exemplo de aplicação da API Varchart para o escalonamento de produção, com uma interface mais enriquecida. Destaca-se também o facto da possibilidade de agrupamento de várias tarefas.

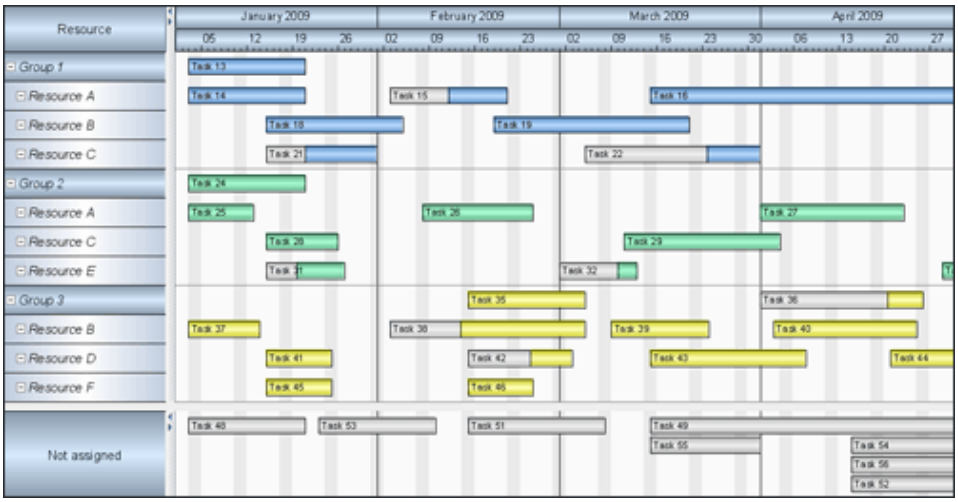

Figura 27: Exemplo de um gráfico VarChart Gantt (URL6).

É uma API comercial, contém licenças de desenvolvimento, manutenção e de actualização e também licenças *run-time*. É constituída por três módulos: Básico (apenas visualização), edição de dados e histograma. A Tabela 2 ilustra os custos para estas licenças:

Tabela 2: Lista de preços para o VarChartt JGantt (adaptado de URL6)

VARCHART JGantt **Módulo**	**Compra**	**Manutenção e suporte** Pagamento annual em €	**Actualizações de** **Apenas um pagamento em €**		
	Apenas um pagamento	*Upgrades* grátis	Versão 2.4 a 2.5	Versão 2.3 a 2.5	Versão 2.2 ou abaixo a 2.5
Básico (apenas visualização)	2,250.0	675.00	450.00	787.50	1,125.00
Edição de dados	750.00	225.0	150.0	262.5	375.0
Histograma	960.0	288.0	192.0	336.0	480.00

Existe também a possibilidade de experimentação da aplicação, através de uma versão demo (a 30 dias). O modelo Básico contém os recursos necessários para a visualização destes diagramas. Suporta também impressão e pré visualização, calendários, entre outros. No modelo Edição de dados é possível, utilizando as funcionalidades disponibilizadas, a edição destes diagramas, bem como o deslizamento vertical e horizontal de barras, criação, alteração e remoção de tarefas e o arrastamento de objectos de uma instância de um objecto JGantt para outro. O modelo Histograma suporta cálculos dos recursos e da sua utilização, mostrando os períodos em que se verifica sobrecarga e subcarga (URL6).

Actualmente encontra-se na versão 2.5, suportando diagramas Gantt sincronizados, informação extra, melhor facilidade de selecção de *links*, entre outros (URL6).

Na Figura 28 encontra-se representado o gráfico de Gantt utilizando o VarChart JGantt.

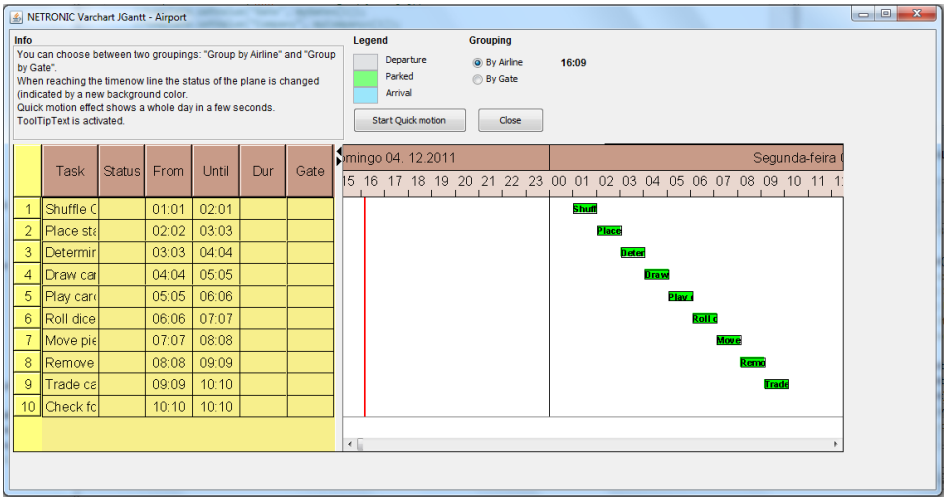

Figura 28: Gráfico Gantt utilizando JGantt.

Esta ferramenta apresenta como vantagem um sistema de *motion*, que permite a simulação de um processo de escalonamento. Quando uma tarefa se encontra em processamento, a cor de fundo dessa tarefa é alterada.

A alteração simples coluna foi demorada e morosa, pois foi necessário alterar a *design*ação em vários campos do ficheiro Java. As opções de arrastamento e o redimensionamento das barras das tarefas não se encontram disponível.

4.1.2 JfreeChart

É uma API *open-source*, desenvolvida pela JFree, implementada em Java, que permite representar, de uma forma profissional, diagramas e gráficos referentes ao planeamento industrial. Suporta funcionalidades interactivas, tais como o *Zooming* e o *Tooltip*.

Esta API oferece uma documentação consistente e organizada, suportando uma vasta gama de tipos de gráficos, fornece suporte a aplicações do lado do cliente e do servidor, e também apresenta compatibilidade com componentes Swing, ficheiros de imagem e outros tipos de ficheiros. É uma API *open-source*, ou seja, é possível obter o código fonte desta ferramenta, sendo também uma aplicação livre (JFreeChart, n.d.). Esta ferramenta suporta a criação de diferentes tipos de gráficos, incluindo gráficos combinados. De referir:

- Gráficos X-Y (linha, *spline* e dispersão), com a possibilidade de incorporar um eixo temporal;
- Gráficos circulares;

- Gráficos Gantt.

Nestes gráficos é possível introduzir marcadores e anotações sobre o traçado, chamados *tooltips* (URL7). A Figura 29 ilustra um exemplo de um gráfico circular que o JFreeChart disponibiliza (URL9).

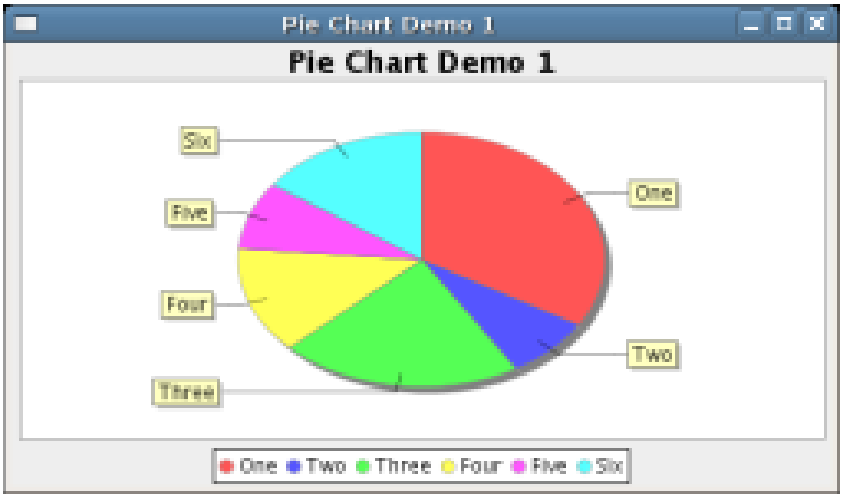

Figura 29: JFreeChart gráfico circular (URL9).

A Figura 30 ilustra um gráfico Dial em que é possível analisar a carga das máquinas.

Figura 30: JFreeChart Gráfico de Dial (URL9).

Actualmente encontra-se na versão 1.0.13 (estável), suporta múltiplas plataformas e possui uma licença LGPL (*Lesser General Public License*). A Figura 31 ilustra o gráfico de Gantt gerado utilizando o JFreeChart.

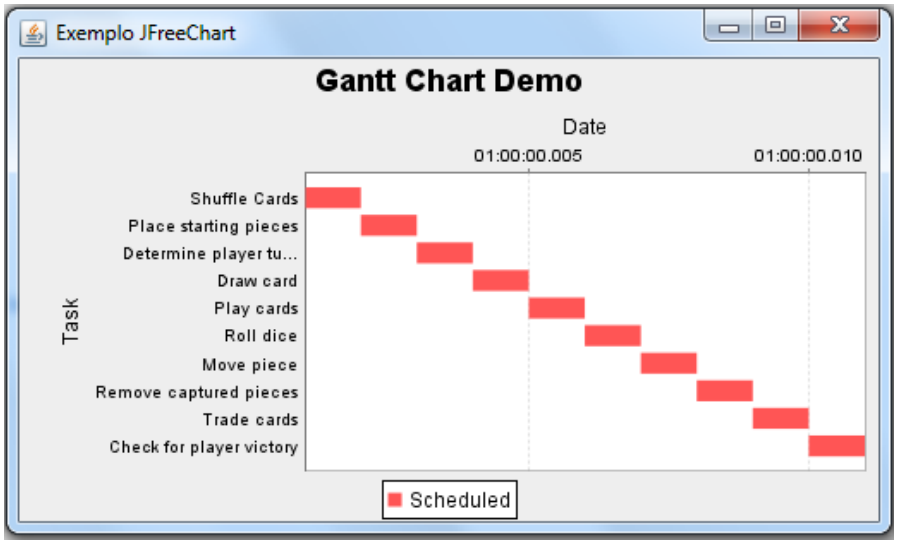

Figura 31: Exemplo JFreeChart.

A criação deste gráfico Gantt foi bastante simples com base no exemplo disponibilizado e na importação das bibliotecas necessárias (*JCommon* e *JFreeChart*) e importar estas bibliotecas para o projecto. Em seguida alterou-se no código as tarefas, de forma a ter como resultado a imagem visualizada na Figura 31. Esta ferramenta é também de fácil integração com as restantes API's de criação de interfaces gráficas. A única limitação encontrada refere-se à impossibilidade de interacção com o gráfico de Gantt (arrastamento de barras, criação de tarefas).

4.1.3 JavaGantt

JavaGantt é uma componente Swing comercial desenvolvida pela BeeSoft para desenho e edição de gráficos Gantt. É uma componente poderosa, de fácil utilização, apesar de complexa. A versão mais recente desta API permite interacção. A definição dos tempos de início e de fim pode ser realizada com o arrastamento dos objectos que representam. Apresenta vantagens, como a possibilidade de personalização de interface, a possibilidade de pintar subcomponentes que são programáveis. A versão de demonstração inclui código fonte.

Esta ferramenta apresenta as seguintes funcionalidades:

- ***Standardview***: contém uma tabela hierárquica no lado esquerdo e a componente *chart* no lado direito, e é possível modificar o aspecto visual de acordo com as necessidades;
- ***Advanced TreeTable***: Combina as funcionalidades da Jtree e Jtable do Swing;

- ***Data Binding***: O modelo do JavaGantt facilita a sua programação e uso, pois é possível introduzir campos sem necessidade de haver mais programação;
- ***Lazy Loading***: Se os dados forem muito grandes, a componente só apresenta a parte visível, havendo melhor desempenho do sistema;
- **Layers**: É possível decidir que *layers* são representados ou escondidos, havendo uma personalização;
- ***Relationships***: As tarefas podem estar relacionadas com outras tarefas (ex. Uma tarefa só começa depois de outra ter sido concluída);
- ***Actions***: A componente permite *zoom-in* e *zoom-out*, criação e remoção de *nodes* e contém acções de *undo* e *redo*;
- ***Localization***: Suporta multilinguagem.

A imagem da figura 32 permite visualizar a interface desta componente (URL10).

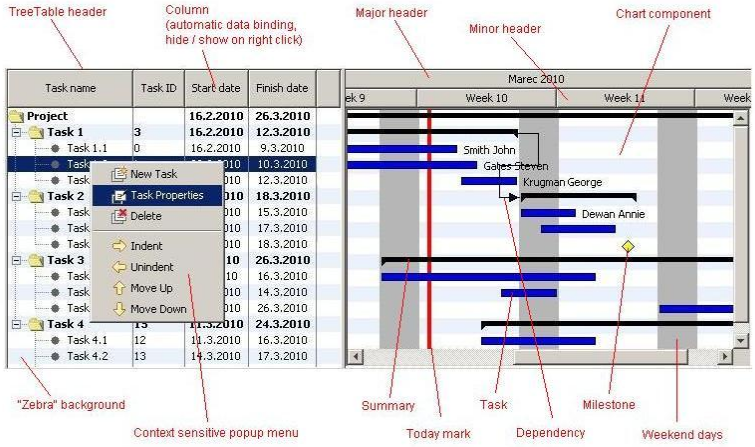

Figura 32: Interface de JGantt (URL10).

É disponibilizada uma versão de demonstração (10 minutos). A aquisição da ferramenta integral implica um custo de 1650 euros (URL11). A Figura 33 ilustra o gráfico de Gantt gerado utilizando o JavaGantt.

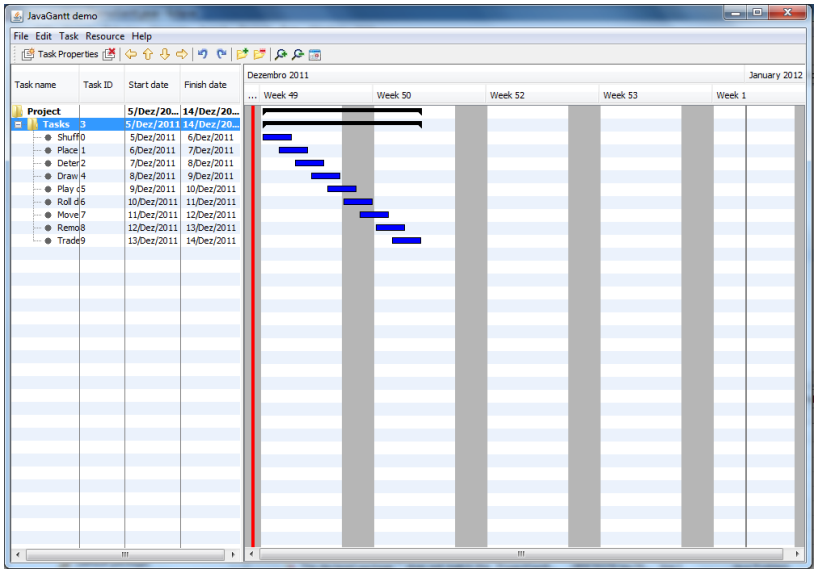

Figura 33: Gráfico Gantt utilizando JavaGantt.

Em termos de importação das classes externas, foi necessário, para além das bibliotecas base, incorporar outras classes directamente no projecto, tais como as classes *Task*, *Resource*, *Project*, entre outras). É possível atribuir *workers* às tarefas e criar dependências entre as tarefas. O código principal é relativamente simples (apenas é necessário a definição de tarefas e, opcionalmente, subtarefas e adicionar ao modelo). Também é possível o arrastamento e o redimensionamento das barras das tarefas para alterar as datas e durações das mesmas.

4.1.4 JIDE GanttChart

JIDE é uma biblioteca Swing comercial para Java que permite a criação de diagramas de Gantt. Tal como as outras componentes JIDE, também permite várias funcionalidades básicas e muitas API's para a respectiva personalização (URL12).

Tem como funcionalidades a vista hierárquica das tarefas, a data de início, a data de fim, e a percentagem de processamento completa de cada tarefa. Esta ferramenta também permite o agrupamento de tarefas, e é possível a definição de uma relação de precedências entre elas (se uma tarefa começa depois de outra, etc.). Também é possível utilizar o rato para mudar as propriedades de uma tarefa, podendo arrastar e alargar/encolher as barras para a alteração dos tempos de início e de fim de uma tarefa. É possível também a navegação com o teclado, suporta o sistema de *zoom-in* e *zoom-out* no diagrama de Gantt, entre outros. A Figura 34 ilustra um exemplo de uma interface usando Jide Gantt Chart.

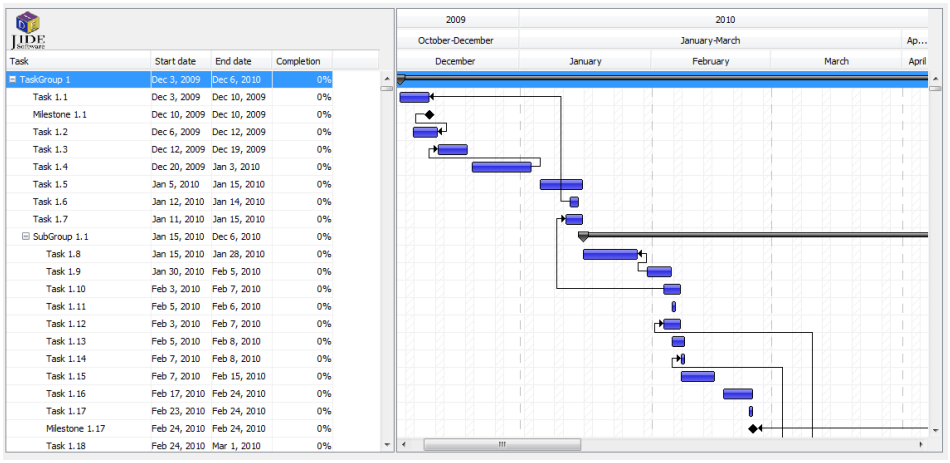

Figura 34: JIDE Gantt Chart Interface (URL12).

Esta componente depende de uma outra *design*ada JIDE Grids, ou seja, para obter o JIDE Gantt Chart também é necessário obter o JIDE Grids. A Tabela 3 sumariza custos associados à aquisição e utilização das componentes (URL13).

Tabela 3: Preços de JIDE Grids e JIDE Gantt Chart

	JIDE Grids	**JIDE Gantt Chart**
Single Developer License	$399.99	$99.99
Annual Maintenance Renewal	$199.99	$49.99
License and Renewal	$539.99	$134.99

A Figura 35 ilustra o gráfico de Gantt gerado usando o JIDE Gantt Chart. Foi usada uma versão de demonstração disponibilizada pela JIDE. Em termos de vantagens destaca-se o facto da possibilidade de introdução de novas tarefas sem recorrer a código extra. Também se pode arrastar e alargar/encolher as barras de cada tarefa para alterar os tempos de início e de fim, bem como a sua duração.

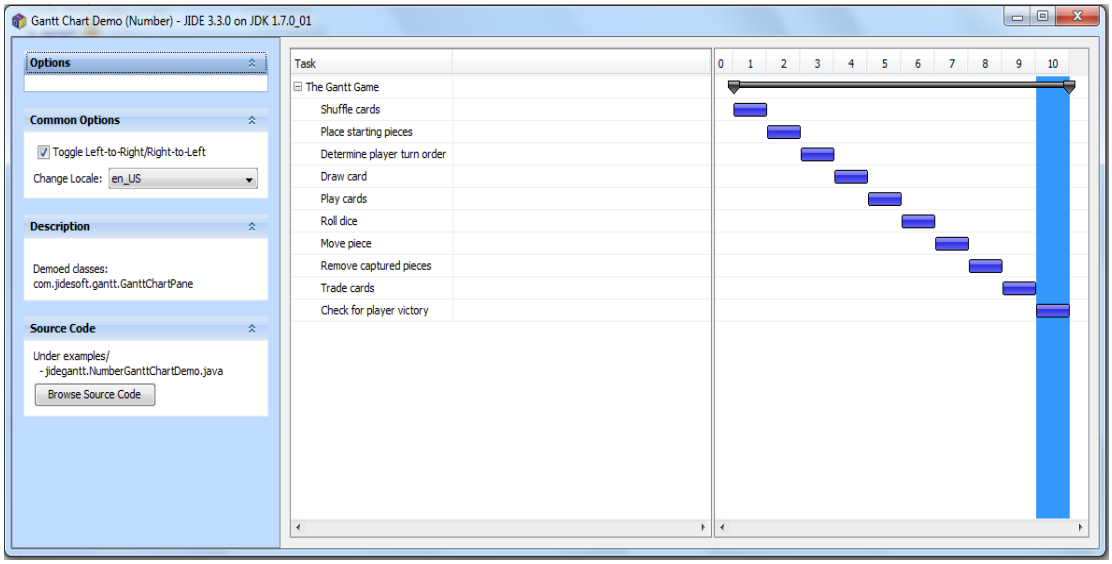

Figura 35: Gráfico de Gantt usando JIDE.

Quanto a limitações destaca-se o facto de o código apresentar elevada complexidade, o que dificulta a sua compreensão. Foi necessário importar classes externas.

4.1.5 **Tabela Comparativa**

Na Tabela 4 foram sistematizadas as características principais das API's em análise, enumerando vantagens e limitações.

Após este estudo e com base nos requisitos necessários para a elaboração deste trabalho de mestrado concluiu-se que o JIDE Gantt, apesar de ter uma boa interface e disponibilizar tutoriais para o utilizador, não será a melhor opção a tomar, pois a sua programação e definição é bastante mais complexa que as restantes.

O JFreeChart é uma ferramenta bastante simples de utilizar, mas não suporta a interacção com os gráficos, o que se torna uma limitação considerando o protótipo a desenvolver, mas visto este ser *open-source*, é possível a implementação dos mecanismos de interacção. O JGantt consegue superar o VarChart JGantt pelo facto de ser possível a sua personalização. Com base nesta análise conclui-se que o JFreeChart é a melhor solução também tendo em conta os custos associados às restantes API's.

Tabela 4: Comparação de API's de criação de gráficos.

API	Vantagens	Limitações	Acesso	*Link*
VARCHART JGantt	Fácil integração nas aplicações Java. É possível prever impressões. Personalização de interface, gradientes.	Arrastamento da versão em modo teste não se encontrava em funcionamento. Alteração de colunas foi necessário uma alteração morosa de código.	Comercial	(URL6)
JFreeChart	Compatível com componentes Swing. Boa variedade de gráficos. Boa documentação. Fácil integração com restantes API's.	Não permite edição de gráficos. Necessário Java 2 instalado.	*Open-Source*	(URL7)
JavaGantt	Fácil personalização da interface. É possível fazer Zoom-in e Zoom-out. Arrastar e encolher. Representação apenas da parte visível. Relação entre tarefas. Criação de tarefas sem programação extra.	Importar classes externas e bibliotecas.	Comercial	(URL10)
JIDE Gantt	Suporta edição das barras das tarefas. Interface apelativa. Contém *user guides*.	Elevada complexidade. Importação de classes externas. Necessário efectuar compra de outra componente.	Comercial	(URL12)

4.1.6 **Conclusão**

Assim sendo, e da análise realizada, conclui-se que a API JFreeChart encontra-se nas melhores condições para ser a mais adequada pelas seguintes razões:

- Fácil personalização da interface e boa documentação;
- *É Open-Source*, e por isso é possível implementar código extra para permitir redimensionamento e posicionamento das tarefas;
- Contém grande variedade de gráficos;
- É compatível com o Swing;

4.2 API's de criação de interfaces gráficas

Nesta secção analisam-se várias API's de criação de interfaces gráficas, tais como o Swing, o AWT e o SWT.

4.2.1 Swing

O Swing é uma ferramenta de desenho de interfaces gráficas que faz parte de uma API da Oracle chamada JFC (Java Foundation Classes) e é usada em aplicações Java. Foi desenvolvida para fornecer componentes de interfaces gráficas mais sofisticadas que o *AWT* (*Abstract Window Toolkit*). É uma ferramenta suportada por todas as plataformas e apresenta o mesmo *look and feel*. Pode ser adaptável e configurável, deixando ao critério do programador a escolha do estilo.

O Swing é uma extensão à biblioteca AWT, que contém novas e melhores funcionalidades para melhorar as funcionalidades das interfaces gráficas. É possível criar aplicações Swing *Standalone* e também *Servlets* e *Applets*.

O Swing usa componentes leves (baixa utilização de recursos) do AWT e, ao contrário do AWT, permite a personalização da interface e do comportamento das componentes. As componentes de ambas as API's podem ser misturadas, permitindo o suporte do Swing em aplicações AWT.

É uma API de mais alto nível que as API's do sistema operativo, o que diminui o desempenho e consome mais memória RAM, mas é mais completa.

O Swing fez inicialmente parte de um conjunto de ferramentas designada AWT (*Abstract Windows Toolkit*). A partir da versão 1.2 do JDK (*Java Development Kit*, um conjunto de utilitários para o desenvolvedor de aplicações Java), o Swing começou a fazer parte integrante do mesmo, tendo como consequência a queda em desuso do AWT (Campos et al, n.d.).

O Swing contém várias características, de entre as quais se destacam:

- A implementação das componentes é feita totalmente em Java;
- *Pluggable Look and Feel* (é possível alterar o *look and feel* em *run-time*);
- Componentes *Lightweight* (componentes de subclasse de Java.Awt.Component);
- Usa um modelo *Model View Controller*.

Em termos de vantagens o Swing suporta uma maior gama de comportamentos, por não usar *"native peers"* (*GUI Widgets*), contém ícones e ferramentas de ajuda, possui um desenvolvimento mais activo e o *look and feel* adapta-se consoante o sistema operativo.

Em termos de desvantagens, a maior parte dos *browsers* necessita de ter o Java instalado, as componentes no geral são mais lentas e com alguns defeitos, e os gráficos podem ter *glitches*, havendo um problema de desempenho. Também apresenta uma desvantagem a nível do *look and feel*, pois as componentes nativas podem funcionar de maneira diferente dos sistemas nativos reais (URL15).

O Swing apresenta várias componentes usuais, tais como botões, menus, janelas e imagens. Também contém funcionalidades mais avançadas (editor de texto).

A Figura 36 ilustra um formulário usando apenas componentes Swing.

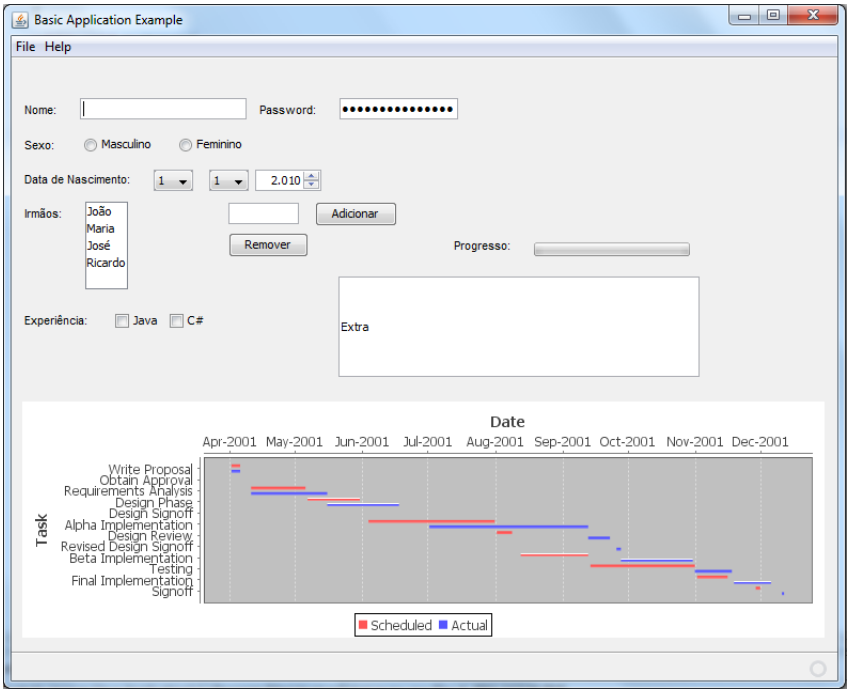

Figura 36: Interface do Swing + Gráfico Gantt

A criação desta interface usando o Swing foi bastante directa e simples, pois existiam todas as componentes, sem necessidade de implementação de código extra. Foi utilizado o ambiente de desenvolvimento NetBeans, que já continha uma componente de criação gráfica de interfaces.

Em termos de desvantagens não foram identificadas, visto que apenas foi necessário o sistema *drag and drop* disponibilizado pelo NetBeans.

4.2.2 **AWT**

AWT, acrónimo para *Abstract Windows Toolkit*, é uma ferramenta original de desenho de interfaces gráficas, também fazendo parte do JFC. A programação é baseada em eventos, que consiste numa técnica onde são definidos métodos a serem executados cada vez que um determinado evento ocorre (URL14).

Esta API é poderosa e flexível mas os utilizadores não conseguem obter essas características por falta de boa documentação. Por exemplo, a descrição dos métodos contém pouca informação, e deixa várias dúvidas por responder.

Como foi referido anteriormente, esta ferramenta começou a cair em desuso desde que o Swing fez parte do JFC.

A ferramenta *AWT* apresenta vantagens ao nível do desempenho, pois usa *"native peers"*, sendo mais rápido e, por consequência, possuindo um melhor desempenho das componentes. Outra vantagem é o facto de a maior parte dos *browsers* suportarem o *AWT*, não sendo necessário instalar o Java. Por fim, as componentes reflectem mais o *look and feel* do sistema operativo que se encontra em uso.

Existem alguns aspectos que influenciam negativamente o uso desta ferramenta, ao nível de portabilidade, pois algumas componentes podem não funcionar em todas as plataformas, e ao nível de funcionalidades, ou seja, não suporta ícones nem ferramentas de ajuda (URL15).

A AWT contém as componentes base de qualquer ferramenta de desenho de interfaces gráficas, incluindo ferramentas de desenho de gráficos e de imagem, gestão de *layouts* e uma plataforma nativa de copiar/colar. A Figura 37 ilustra um formulário usando componentes AWT.

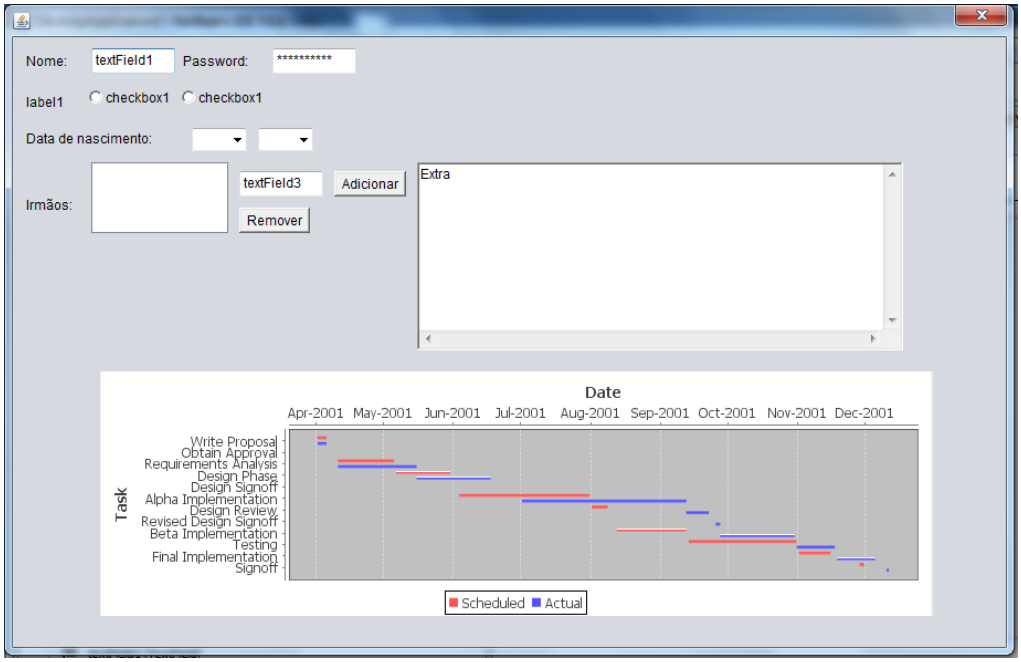

Figura 37: Interface AWT + Gráfico Gantt

Como vantagens, também foi usado o ambiente de desenvolvimento *NetBeans* para a criação desta interface gráfica. Em grande parte das componentes foi apenas necessário o arrastamento das mesmas para a interface.

Em termos de desvantagens, a criação de uma janela em AWT é feita por zonas, dificultando a definição de uma posição exacta para uma determinada componente. Em alternativa foi criada usando o JDialog do Swing. A criação de botões de rádio não foi tão directa como no Swing: foi necessário o uso de duas *checkboxes*, e atribuir às duas uma *checkboxgroup*, o que levou a escrita de código extra. Desta forma, as *checkboxes* foram transformadas em botões de rádio. Para a criação de uma caixa de texto no formato de introdução de uma palavra passe foi necessário introduzir código extra.

4.2.3 **SWT**

SWT (*Standard Widget Toolkit*) é uma ferramenta de desenho de interfaces gráficas *open-source* originalmente desenvolvido pela IBM (*International Business Machines*), actualmente mantida pela *Eclipse*, e escrita para a plataforma Java (URL16). A ferramenta usa a JNI (*Java Native Interface*) para a exibição dos seus *Widgets*, e possui uma licença *open-source* aprovada pelo *Open Source Initiative*.

O SWT e o Swing são portáveis, mas a sua implementação é única para cada sistema. Actualmente o SWT encontra-se na versão 3.7.1, lançada em 10 de Setembro de 2011, encontra-se numa evolução activa, multilinguagem, e contém uma licença EPL (*Eclipse Public License*). É uma ferramenta multiplataforma e suporta as seguintes plataformas (URL16):

- Windows XP, Vista;
- AIX, FreeBSD, Linux, HP-UX, Solaris;
- MAC OS X;
- QNX Photon;
- Pocket PC.

Com esta API não é tão fácil a implementação de herança de classes SWT para *Widgets*, o que para alguns utilizadores pode constituir uma enorme desvantagem. O mesmo não acontece com o Swing. Para o SWT é necessário um esforço extra para esses W*idgets* funcionarem em várias plataformas.

Para a utilização dos objectos SWT é necessário o uso da função *dispose()*, senão poderá haver "fuga" de memória ou comportamentos inesperados. Os únicos objectos que são necessários para o uso desta função são as subclasses de *Resource*, tais como *Image*, *Color* e *Font*.

Existem alguns esforços no sentido de combinar o SWT com o Swing. Duas das abordagens são o SwingWT, permitindo o mesmo *look and feel* e vantagens de desempenho do SWT, com o mesmo modelo de programação do Swing; e o SWTSwing, usando os objectos *Swing* para permitir que SWT funcione nas plataformas compatíveis com o *Swing*.

O SWT apresenta algumas vantagens. Por exemplo, utiliza elementos nativos, tendo assim um comportamento nativo, é suportado pelo Eclipse (aplicação para desenvolvimento de aplicações Java) que contém um editor de interfaces gráficas chamado VEP, contém um largo número de exemplos *online*, e apresenta uma *bridge* entre AWT e SWT para permitir o uso de componentes Swing.

Duas das desvantagens do SWT são o facto de serem necessárias bibliotecas nativas para cada sistema suportado, e também pode não suportar todas as componentes em todos os sistemas devido aos recursos nativos usados.

Uma diferença do Swing para o SWT é o facto de o Swing implementar o Modelo MVC (*Model View Controller*), enquanto o SWT contém uma API chamada JFace que se aproxima desse modelo.

O SWT contém as componentes básicas necessárias para a criação de uma interface gráfica simples. A Figura 38 ilustra um formulário usando algumas das componentes SWT.

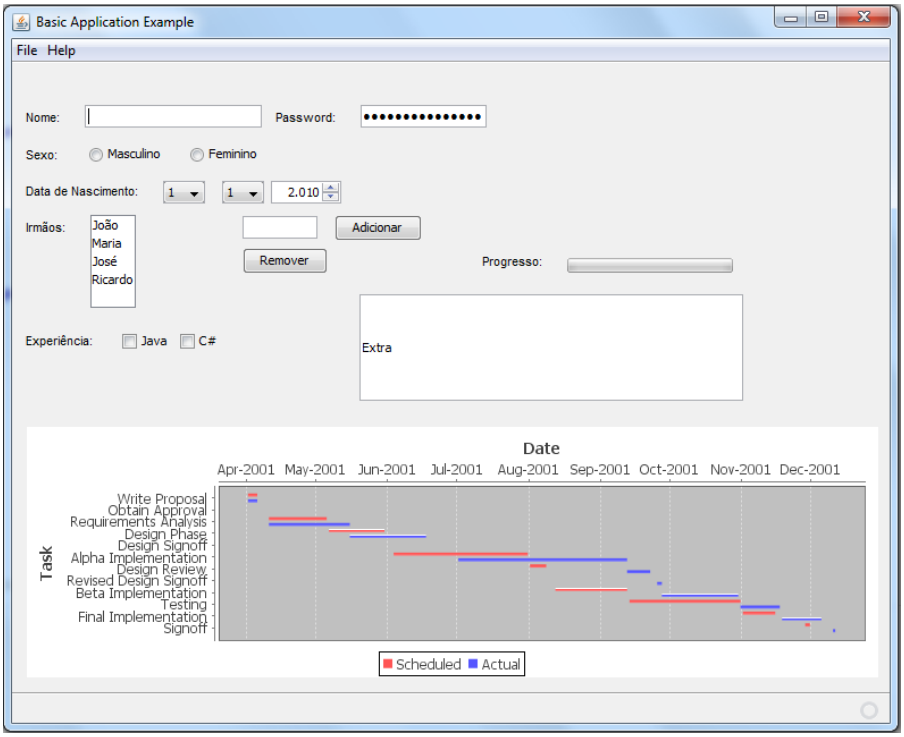

Figura 38: Interface SWT + Gráfico Gantt

A interface foi criada de forma rápida e sem grandes dificuldades. Apenas foi necessário o arrastamento das componentes.

Em termos de desvantagens, foi usado o ambiente de desenvolvimento Eclipse, o que por omissão não continha nenhuma ferramenta de criação gráfica de interfaces. Foi necessária a instalação de uma ferramenta chamada *WindowBuilder*. Na criação da caixa de texto *Password* também foi necessária a modificação do código.

4.2.4 **Tabela Comparativa**

A Tabela 5 ilustra uma comparação entre as API's indicadas anteriormente, indicando as respectivas vantagens e desvantagens, entre outras.

Tabela 5: Tabela Comparativa de API's de criação de interfaces gráficas

API	Vantagens	Desvantagens	Acesso	*Link*
Swing	Faz parte do Java; Funciona de forma similar em todas as plataformas. Contém editor visual no NetBeans; Bons tutoriais. Suportado por extensões Java oficiais (Java OpenGL). Portável.	*Look and Feel* nativo pode funcionar de maneira diferente para sistemas diferentes. É necessário Java2.	Livre	(URL17)
AWT	Elevado desempenho. Portável. Reflecte mais aproximadamente o *look and feel* de cada sistema.	Caiu em desuso. Não é possível representar imagens e texto em botões e labels. Não há tantas componentes como nas outras API's. Nem todas as componentes podem funcionar.	Livre	(URL16)
SWT	Tem comportamento nativo. Suportado por eclipse, VEP editor. Vasta gama de exemplos online. É possível também usar componentes Swing e AWT.	Necessárias bibliotecas nativas. Pode não suportar todos os comportamentos. Recursos do género "tipo de letra" são necessários serem registados num *listener*.	Livre	(URL17)

4.3 Bibliotecas para criação de interfaces gráficas

Nesta secção descrevem-se várias bibliotecas para criação de interfaces gráficas, apresentando algumas funcionalidades, vantagens e limitações das mesmas.

4.3.1 **Qt**

Qt é uma *framework*, desenvolvida em C++, usada para criar aplicações *cross-platform* de alta eficiência. É uma aplicação poderosa, apesar de *open-source* e permite a produção de uma interface gráfica rica (URL18).

A ferramenta apresenta vantagens tais como o facto desta ser uma ferramenta multiplataforma, mantém o mesmo *look and feel* para as diferentes plataformas e contém um *designer* gráfico poderoso (*Qt Designer*).

4.3.1.1 Qt SDK

O Qt também dispõe de um SDK em que é possível criar, usando ferramentas de *design*, aplicações para Symbian e Maemo, Meego (Nokia N9). Essas ferramentas são:

- ***Qt framework***: dispõe de API para rápida criação de UI, usando Qt Quick;
- ***Qt Creator IDE***: ambiente de desenvolvimento multiplataforma, incluindo ferramentas de criação de UI e sistema de *debugging*;
- ***Tools and Toolchains***: Simuladores e compiladores locais e remotos.

4.3.1.2 Qt Jambi

Foi criada uma outra *framework* denominada *Qt Jambi*, que utiliza a tecnologia de *Binding* e permite a criação de aplicações Java, para versões JEE (*Java Enterprise Edition*) e JSE (*Java Standard Edition*). Esta tecnologia é implementada parcialmente em Java e C++ e usa o *Qt/Java Generator* para a geração de grande parte do código (URL19). Contém um comportamento nativo, conferindo um *look and feel* nativo nas plataformas que suporta.

4.3.1.3 Qt/Java Generator

O *Qt/Java Generator* (URL19) é um software utilizado para facilitar a transição entre um ambiente C++ e o Java, e também aumenta a colaboração de desenvolvedores que usem estas tecnologias separadamente. Esta aplicação lê as definições de classes criadas em C++ e gera o código que mapeia a API da biblioteca original para o equivalente em Java. Também gera o código para cada função que mapeia.

A componente implementada em C++ usa o JNI para a comunicação entre o Java e o código nativo compilado.

4.3.1.4 Conversão de tipos

A conversão de tipos de dados é feita usando a marcação XML (*Extensible Markup Language**)*** que indica os atributos de cada classe que foi mapeada. Este documento XML é escrito à mão e contém informação de cada classe (URL19). A aplicação converte certos atributos das classes, tais como objectos de classes, valores, interfaces, primitivas e enumerações.

4.3.1.5 Herança Múltipla

A aplicação não suporta herança múltipla. Desta forma, gera um código que implementa uma interface quando existe essa herança, e quando existe herança múltipla, todas as subclasses são filhas de apenas uma superclasse que foi declarada no documento XML (URL19). A Figura 39 ilustra o formulário elaborado usando a *framework* Qt.

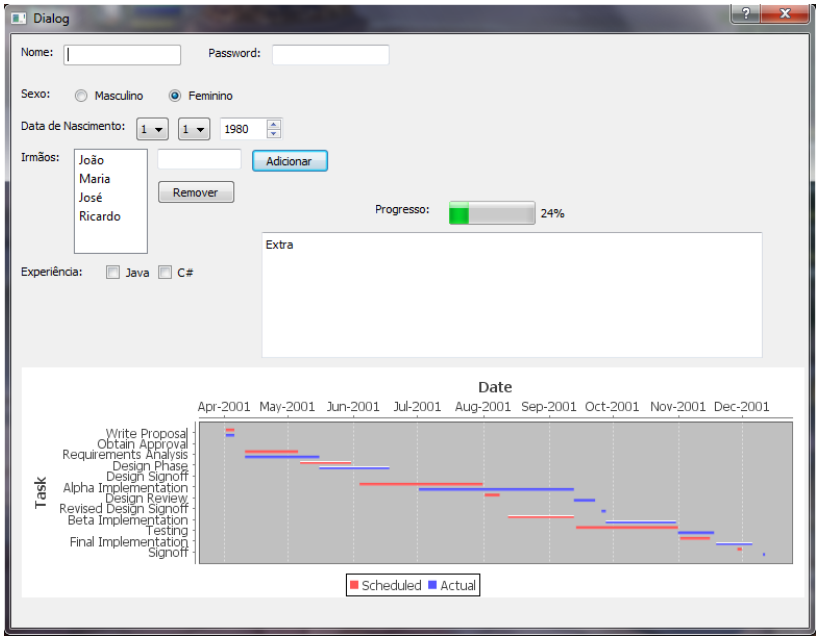

Figura 39: Interface usando Qt + Gráfico Gantt

Em termos de vantagens, a criação da interface com o utilizador foi rápida e decorreu sem quaisquer problemas.

Quanto a desvantagens, para a criação de eventos é necessário efectuar manualmente a introdução manual do código. Foi necessário a instalação do Qt Eclipse Integration e da procura de uma versão do Qt Jambi compatível com esta aplicação.

4.3.2 **JwxWidgets**

É uma ferramenta de criação de interfaces gráficas desenvolvida para Java, sendo esta *framework* originalmente de *wxWidgets*. JwxWidgets encontra-se numa versão pouco desenvolvida (alpha) e suporta as plataformas GTK, MSW e MAC (URL20). Em termos de componentes suporta botões, imagens, listas, eventos de rato, janelas,

menus, tipos de letras, entre outros (URL21). Para a instalação desta ferramenta, é necessário cumprir os seguintes requisitos (URL22):

- JDK 1.4 ou superior;
- ANT 1.6 ou superior;
- MinGW (GCC>= 3.4);
- wxWidgets 2.6.3;
- Para a versão MAC OS também é necessário ter JarBundler.

Esta ferramenta não apresenta vantagens visto não ter sido possível a realização de uma interface, recorrendo a esta ferramenta.

4.3.3 **JIDE**

O JIDE é um provedor profissional de componentes para Java e Swing. Foi desenvolvido pela JideSoft que disponibiliza mais de cem componentes Swing para serem utilizados (URL23).

Os produtos JIDE são focalizados em componentes ricas para melhor facilidade no desenvolvimento de aplicações. Existem quinze produtos de componentes, sendo vários o JIDE *Components*, JIDE *Dialog*, JIDE *Grids*, JIDE *Gantt*, entre outros (URL24).

É uma biblioteca comercial, havendo uma versão *trial* chamada *Evaluation Package* e uma demonstração do mesmo. O JIDE disponibiliza três pacotes, sendo eles o JIDE *Professional Suite*, JIDE *Enterprise Suite* e JIDE *Ultimate Suite*, tendo o JIDE *Professional Suite* menos produtos do que o JIDE *Ultimate Suite* (URL25).

JIDE disponibiliza uma componente chamada JIDE *Common Layer* que contém vantagens tais como facilidade de utilização e compreensão, suporta qualquer *look and feel* sem necessidade de código extra, entre outros (Jide Common Layer, n.d.). JIDE também dispõe de guias para o utilizador para cada produto.

4.3.3.1 Preços e Licenças

Para usufruir destes produtos é necessário uma *Single Developer License*, que inclui também três meses de manutenção e a actualização de produtos e suporte técnico por correio electrónico (URL26). Esta licença permite que um indivíduo usufrua das API's disponibilizadas pelo JIDE em qualquer número de projectos. Não pode ser usada ou partilhada por mais que um desenvolvedor.

4.3.3.2 Source Code License

Ao contrário da licença do desenvolvedor, esta licença é usada apenas num projecto. Se existirem mais do que um desenvolvedor no projecto, apenas será necessário a obtenção de uma licença *Source Code*.

4.3.3.3 Annual Maintenance Renewal

Caso o utilizador queira continuar a receber suporte técnico e actualizações de produtos, terá que adquirir esta licença.

4.3.3.4 Deployment License

Existem casos em que é necessário a aplicação de uma taxa para os produtos do JIDE. Se um utilizador necessitar de expor API dos produtos para os seus clientes, será aplicada uma taxa, que poderá ser negociável. Se o produto do utilizador tem muitas colocações, poderá ser aplicada uma taxa única e de duração ilimitada para cobrir o suporte.

4.3.3.5 Licença livre para projectos *open-source*

É possível obter uma licença *Single Developer License*, caso o projecto a desenvolver seja *open-source.* Para tal, o utilizador deve indicar o nome do projecto *open-source*, o sítio *Web* do projecto e os produtos JIDE que pretende utilizar.

Esta licença é similar à *Single Developer License* com a excepção de duas condições. A primeira condição é a existência de um acordo em existir um *link* para o sítio *Web* do JIDE na aplicação do utilizador, e a segunda condição consiste em aplicar a licença apenas no projecto em causa.

Em termos de preços, a Tabela 6 enumera as componentes de cada *Suite* e os preços relativamente ao *Single Developer License* e Annual *Maintenance Renewal* (URL26).

É uma ferramenta que permite a representação das tarefas no tempo, útil para planeamento e escalonamento de projectos. Quanto ao JIDE não é possível criar uma interface gráfica usando os elementos das interfaces anteriores, pois apenas disponibiliza componentes mais complexas.

Tabela 6: Lista de preços (URL25).

Product Bundle Item #	**JIDE Professional Suite #4280**	**JIDE Enterprise Suite #4680**	**JIDE Ultimate Suite #4880**
JIDE Docking Framework	✓	✓	✓
JIDE Action Framework	✓	✓	✓
JIDE Components		✓	✓
JIDE Grids		✓	✓
JIDE Dialogs		✓	✓
JIDE Pivot Grid			✓
JIDE Shortcut Editor			✓
JIDE Code Editor			✓
JIDE Feed Reader			✓
JIDE Dashboard			✓
JIDE Data Grids			✓
JIDE Charts			✓
JIDE Gantt Chart			✓
JIDE Diff			✓
JIDE TreeMap			✓
Single Developer License	$449.99	$699.99	$1399.99
Annual Maintenance Renewal	$224.99	$349.99	$699.99
License and Renewal	$607.49	$944.99	$1889.99
Notes			Free upgrade to future component products. Please note, JDAF is not included
Multiple-Product Discount is included in the unit price above			
Volume Discount	10% de desconto se adquirir 5 a 9 cópias 20% de desconto se adquirir 10 copias		

4.3.4 **Buoy**

O Buoy é uma ferramenta para criação de interfaces gráficas para plataformas Java criada sob o *Swing*, mas que disponibiliza uma nova gama de classes para representar componentes de interfaces (URL27).

Existem algumas vantagens em relação ao Swing, pois o Buoy é uma API mais simples e consistente, apresenta um melhor mecanismo de apresentação de componentes, tem um mecanismo mais poderoso em relação ao tratamento de eventos e é possível criar a interface em XML, havendo reutilização de código (URL27).

Em termos de funcionalidades o Buoy (URL27):

- Esconde a complexidade do Swing, não necessitando de a tratar em certos casos;
- Ocupa menos espaço e é mais eficiente;
- É escrito em Java;
- Código aberto.

Existem três ferramentas de criação de interfaces gráficas, sendo elas o BuoyBuilder, o Balise e o Minuet. Neste documento apenas se irá analisar o Balise, pois não foram encontradas versões das restantes ferramentas.

Balise é uma ferramenta de criação de GUI em Java, usando a ferramenta Buoy. O objectivo do Balise consiste na criação de uma interface gráfica, recorrendo a ficheiros XML para a abertura e gravação da mesma. O objectivo não é só a criação de um ficheiro XML que fica separado do código, mas também na criação de interfaces que não depende inteiramente do princípio WYSIWYG (*What you see is what you get*) (URL28).

Uma aplicação que contém elementos gráficos pode ser editada usando o princípio WYSIWYG (a nível gráfico) ou, num nível mais lógico, usando ficheiros de texto. A edição lógica é mais poderosa que a edição gráfica, mas tende a ser mais lenta e mais aborrecida, e a curva de aprendizagem não é tão linear, o que leva a concluir que as ferramentas baseadas em WYSIWYG não são necessariamente as mais poderosas.

A edição de GUI não constitui excepção à regra. As GUI modernas usam regras lógicas tais como *anchors, springs* ou *layout* hierárquico para representar

componentes ou *widgets*. E grande parte do tempo é necessário à edição desses ficheiros nessas janelas, como um processador de texto.

Isto pode ser resolvido, atribuindo a estrutura lógica a uma janela de edição. Assim utilizam-se duas janelas para editar uma componente GUI: uma onde a componente em causa está a ser criada, e outra para a edição da sua estrutura lógica. Na janela de edição, é possível correr em *run-time* para prever o seu comportamento. Também é possível aplicar uma janela de edição *Balise* para qualquer *Widget* em *run time*, não só quando esse *Widget* foi criado usando o ficheiro de definição XML, como em código.

4.3.4.1 Funcionalidades técnicas do Buoy

Pode ser considerado como um substituto do Swing, mas a criação dessas novas classes usa componentes *Swing*. No *Swing* usam-se componentes, com Buoy cria-se *Widgets*. Para cada *Widget*, é criada a respectiva componente para implementar esse *Widget*. Por outras palavras, *Widgets* têm uma representação mais simples que as componentes em si. Por exemplo, a versão J2S3 1.4.2 da componente Jcomponente tem trezentos e seis métodos públicos, enquanto a versão actual do *Widget* do Buoy contém apenas trinta e sete.

Alguns desses métodos da componente JComponent deixaram de ser utilizados. Os *listeners* da componente contêm inúmeros métodos, ao contrário do Buoy, que não necessita de métodos diferentes para cada tipo de evento. Existem também métodos da componente que são redundantes. O Buoy usa apenas um.

Tanto o Swing como o Buoy permitem a personalização de *Widgets*, mas a maneira de a efectuar é diferente. No Swing para criar um *Widget* personalizado, define-se uma classe que estenda JComponent. A subclasse tem de fazer *overrride* de paintComponent() para ficar com a aparência da componente. Se for necessário responder a determinados eventos, é necessário implementá-los. No Buoy é diferente. é possível usar a classe *CustomWidgets* e instanciá-la directamente, e o controlo da aparência necessita apenas de um *event link* para *RepaintingEvents*.

Em termos de criação de interfaces, existem duas abordagens importantes. Primeiro, todas as componentes que são instanciadas e os seus atributos estão localizados num determinado espaço de código. A segunda abordagem é o facto de usar "*user interface definition records*" que se encontram separados do programa em que está a desenvolver. Quanto à primeira abordagem, o Buoy funciona de maneira

diferente. Utiliza um mecanismo de serialização, em XML, para a representação dessas componentes (URL27).

Uma outra vantagem do Buoy em relação ao Swing é o facto de, no Swing, se se pretender implementar a classe *MouseListener* e, por exemplo, se for necessário a implementação do método *mouseClicked()*, não só é necessário implementar esse como também o resto dos métodos que essa classe fornece. No Buoy isto não acontece, pode-se criar *handlers* apenas para os eventos que serão necessários implementar, sem necessidade de criar métodos vazios.

Em termos de privacidade, no Swing, se se pretender implementar *event listeners*, todos os métodos deverão ser públicos. É possível contornar este problema, criando uma classe interna e passando para métodos privados, mas fazendo isto aumenta o esforço que o desenvolvedor necessita, e também reduz a manutenção e pode aumentar os defeitos do programa. Como o Buoy utiliza métodos *handlers*, podemos alterar a definição desses métodos como privados ou protegidos.

A Figura 40 ilustra um formulário usando a *framework* Buoy.

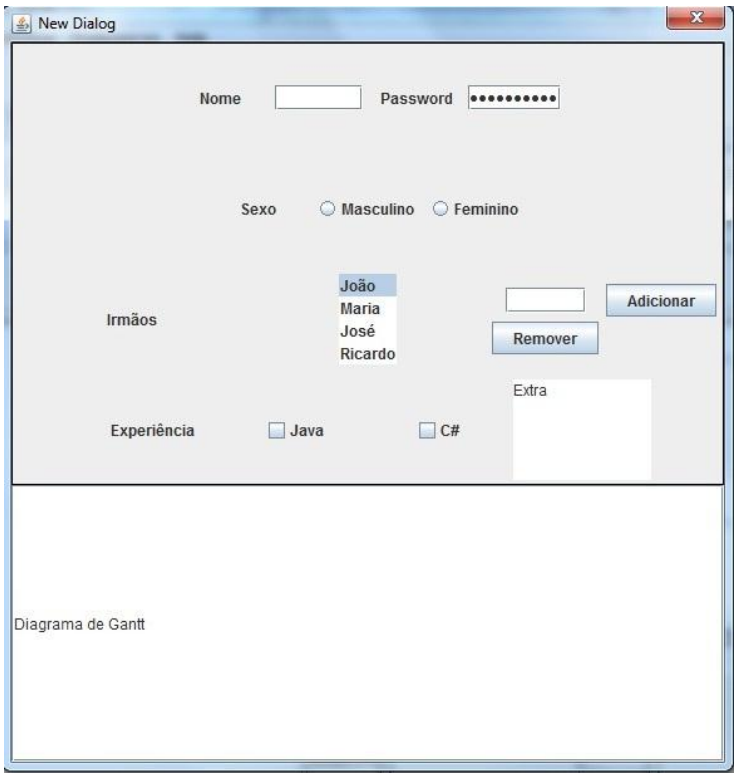

Figura 40: Interface Buoy

Neste caso usou-se o *software* Balise. Para executar este *software* foi necessário criar um projecto em Java, importar as bibliotecas disponibilizadas e usar o código disponibilizado no site do Balise para criar uma instância do software Balise.

Não foram encontradas vantagens na utilização da biblioteca Buoy.

Em termos de desvantagens, a elaboração desta interface é feita por zonas, o que complica a definição de uma posição exacta para as componentes. Também não foi possível a introdução do diagrama de Gantt, pois ocorria um problema de compilação.

4.3.5 **Tabela Comparativa**

A Tabela 7 ilustra as ferramentas de criação de gráficos, indicando algumas vantagens e desvantagens de cada uma delas.

Tabela 7:Tabela comparativa de ferramentas de criação de gráficos.

API	**Vantagens**	**Desvantagens**	**Acesso**	***Link***
Qt	*Widgets* poderosas. Fácil utilização e aprendizagem. Boa documentação. Bom suporte. Desenvolvimento activo.	Necessário introduzir ficheiros extra. Criar componentes complexas pode ser complicado.	*Open-source*	(URL18)
Jwx Widgets		Fase de desenvolvimento *alpha*. Não houve alteração desde 2006. Tutoriais pouco desenvolvidos.	*Open-source*	(URL21)
JIDE	Suportam bastantes componentes Swing. Componentes mais apelativas. Suporta diagramas Gantt.	Não é possível criar uma interface usando componentes simples.	Comercial	(URL23)
Buoy	*Widgets* personalizados. Tratamento de eventos mais simplificado. Serializa interfaces em XML. Melhor mecanismo de representação da interface.	O posicionamento das componentes não é preciso. Não foi possível a inserção do diagrama de Gantt.	*Open-source*	(URL27)

Analisando esta tabela, conclui-se que estas ferramentas, à excepção do *Jwx Widgets* apresentam algum potencial, apesar de existirem limitações em cada uma delas.

4.4 Conclusão

Neste capítulo abordou-se as várias API's de criação de interfaces gráficas, bem como algumas ferramentas de suporte na criação das mesmas, e também foram analisadas ferramentas de criação de gráficos. Para cada um destes grupos, foi construída uma tabela comparativa a indicar as vantagens e desvantagens de cada ferramenta.

Após esta análise chegou-se à conclusão de que o Swing por si só consegue atingir todas as necessidades na criação de uma interface gráfica, não sendo necessário a utilização de uma ferramenta extra.

Quanto à ferramenta de criação de gráficos, a mais indicada tendo em conta a dificuldade de aprendizagem e respectivas funcionalidades é o JFreeChart pois contém uma programação muito simples, é *open-source*, pelo que é possível a implementação dos mecanismos de *drag and drop* para interacção com um gráfico de Gantt, disponibiliza boa documentação, compatível com o Swing e contém grande variedade de gráficos.

5 Processo de desenvolvimento

Neste capítulo procede-se à descrição do processo de desenvolvimento do protótipo. Inicialmente é ilustrada a arquitectura geral do sistema procedendo, em seguida, à descrição das funcionalidades implementadas e integradas no âmbito desta dissertação de mestrado.

5.1 Arquitectura do sistema

O sistema a desenvolver encontra-se dividido em vários módulos, ilustrado na Figura 41. O Editor de Tarefas, o Editor de Problemas, o Editor de Máquinas, e a Visualização de Resultados reflectem a interface com o utilizador. Estes módulos encontram-se interligados entre si e o módulo de escalonamento do sistema ADSyS.

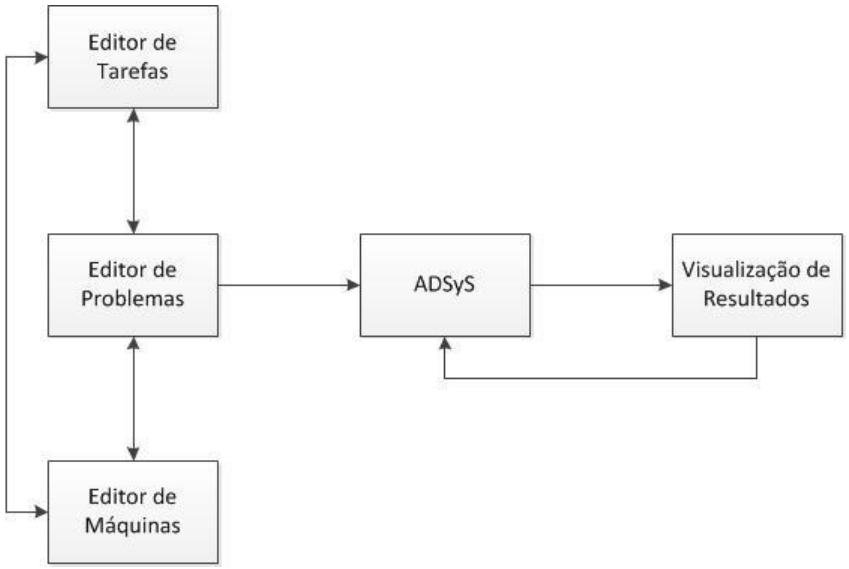

Figura 41: Arquitectura do sistema.

- O **Editor de Tarefas** é um módulo responsável pela criação de uma gama operatória (tarefa) ou pela edição de uma já existente;
- O **Editor de Problemas** é um módulo onde o utilizador define ou edita um problema de escalonamento já existente;
- O **Editor de Máquinas** é um módulo onde o utilizador define ou edita um *layout* fabril já existente.

- O módulo de **Visualização de resultados** permite ao o utilizador aceder a *outputs* (relatórios, gráficos) gerados pelo sistema.
- O modelo de escalonamento do sistema **ADSyS** é responsável pelo processamento dos dados provenientes do módulo de edição de problemas;

No âmbito deste trabalho de mestrado é desenvolvido o módulo de Edição de Máquinas o qual este será detalhado nas secções seguintes.

5.2 Análise de requisitos

No início do processo de desenvolvimento, foi necessário o levantamento dos requisitos para este módulo. Como requisitos de interface, o sistema teria que seguir ao máximo as mesmas orientações utilizadas nos outros módulos, para facilitar a interpretação, aprendizagem e utilização geral do sistema.

Seguindo essas orientações, o sistema teria que conter uma área de desenho, uma barra de menus, uma barra de ferramentas, uma janela de atributos (quando necessário), uma janela de ferramentas e uma barra de estado.

Como requisitos principais o módulo de Edição de *layouts* fabris, pretendia-se uma componente onde o utilizador desenharia a planta da fábrica, contendo a localização das máquinas, possíveis trajectos, e outros elementos gráficos. Esses elementos podem ser rectângulos, elipses, *labels* e linhas. O utilizador pode a qualquer momento alterar as propriedades visuais desses elementos tais como a cor de preenchimento, a cor da periferia, entre outras. O sistema também deve permitir a inserção de uma descrição dos elementos, de forma a proporcionar uma melhor interpretação do *layout* fabril.

5.2.1 Casos de uso

A Figura 42 ilustra o diagrama de casos de uso referente às ligações existentes entre o utilizador e o sistema. Este diagrama representa as funcionalidades principais do módulo de Edição de um *layout* fabril, e respectivas precedências. O utilizador pode abrir um ficheiro, que representa um *layout* fabril existente, utilizar os vários elementos de desenho, disponibilizados pelo módulo, para a elaboração de um *layout* fabril, e gravar esse trabalho num ficheiro. As funcionalidades de alteração das propriedades dos elementos só são disponibilizadas após a criação desses elementos.

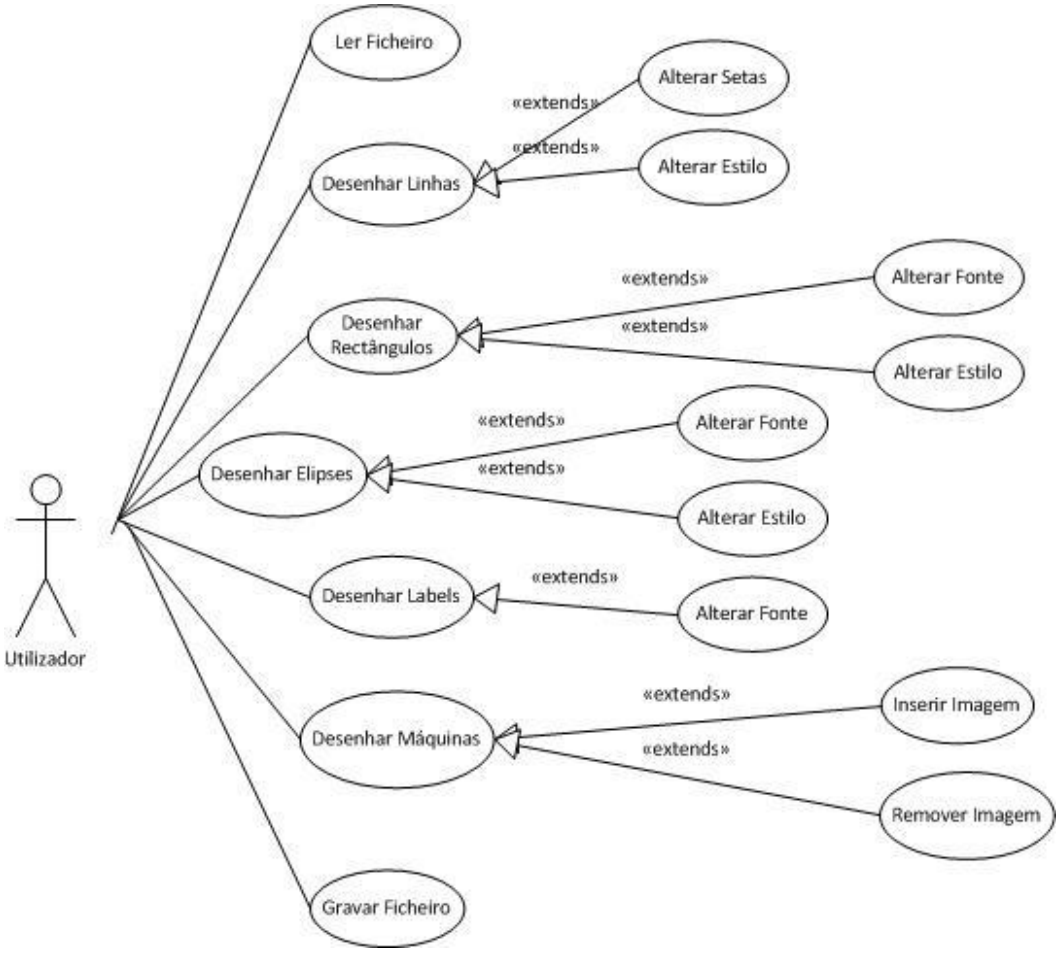

Figura 42: Diagrama de Casos de Uso.

As tabelas a seguir ilustram os casos de uso desenvolvidos no âmbito da análise de requisitos. Estes casos de uso são referentes aos processos de criação e alteração das propriedades dos vários elementos de desenho e máquinas, e também dos processos de leitura e gravação do *layout* definido pelo utilizador. Os casos de uso "Alterar Fonte" e "Alterar Estilo" contêm a mesma descrição para os vários elementos de desenho em causa, sendo estes apenas descritos uma vez.

5.2.1.1 Caso de Uso “Ler Ficheiro”

A Tabela 8 apresenta a descrição do caso de uso “Ler Ficheiro” no formato completo, onde se descreve quem pode usufruir desta funcionalidade, partes interessadas, condições e fluxo principal.

Tabela 8: Descrição do Caso de Uso “Ler Ficheiro”

Caso de Uso “Ler Ficheiro"	
Actor	Utilizador
Partes interessadas	Utilizador: Leitura do ficheiro directo e simples.
Pré-Condições	
Pós-Condições	• O ficheiro tem de conter uma estrutura válida
Fluxo Principal	1. O utilizador vai ao menu “File” e clica na opção “Open” 2. O sistema apresenta uma caixa onde o utilizador selecciona um ficheiro para ser lido. 3. O utilizador selecciona esse ficheiro. 4. O sistema lê o ficheiro e apresenta os resultados na área de desenho.
Fluxo Secundário	**3a. O utilizador cancela a selecção do ficheiro** 1. O sistema emite um aviso do sucedido. **4a. O ficheiro em causa encontra-se inválido** 1. O sistema cancela o processo de leitura. 2. O sistema emite um aviso do sucedido.

5.2.1.2 Caso de Uso "Desenhar Linha"

A Tabela 9 apresenta a descrição do caso de uso "Desenhar Linha" no formato completo.

Tabela 9: Descrição do Caso de Uso "Desenhar Linha"

Caso de Uso "Desenhar Linha"	
Actor	Utilizador
Partes interessadas	Utilizador: Desenho da linha de forma intuitiva.
Pré-Condições	• A opção de selecção da linha tem de se encontrar activa.
Pós-Condições	
Fluxo Principal	1. O utilizador escolhe a opção de criação de uma linha. 2. O utilizador define, através do mecanismo *drag and drop* o início e o fim da linha.

5.2.1.3 Caso de Uso "Desenhar Rectângulo"

A Tabela 10 apresenta a descrição do caso de uso "Desenhar Rectângulo" no formato completo.

Tabela 10: Descrição do Caso de Uso "Desenhar Rectângulo"

Caso de Uso "Desenhar Rectângulo"	
Actor	Utilizador
Partes interessadas	Utilizador: Desenho de um rectângulo de forma intuitiva.
Pré-Condições	• A opção de selecção do rectângulo tem de se encontrar activa.
Pós-Condições	
Fluxo Principal	1. O utilizador escolhe a opção de criação de um rectângulo. 2. O utilizador define, através do mecanismo *drag and drop* o comprimento e a largura do rectângulo.

5.2.1.4 Caso de Uso "Desenhar Elipse"

A Tabela 11 apresenta a descrição do caso de uso "Desenhar Elipse" no formato completo.

Tabela 11: Descrição do Caso de Uso "Desenhar Elipse"

Caso de Uso "Desenhar Elipse"	
Actor	Utilizador
Partes interessadas	Utilizador: Desenho de uma elipse de forma intuitiva.
Pré-Condições	• A opção de selecção da elipse tem de se encontrar activa.
Pós-Condições	
Fluxo Principal	1. O utilizador escolhe a opção de criação de uma elipse. 2. O utilizador define, através do mecanismo *drag and drop* o tamanho da elipse.

5.2.1.5 Caso de Uso "Desenhar *Label*"

A Tabela 12 apresenta a descrição do caso de uso "Desenhar *Label*" no formato completo.

Tabela 12: Descrição do Caso de Uso "Desenhar Label"

Caso de Uso "Desenhar *Label*"	
Actor	Utilizador
Partes interessadas	Utilizador: Desenho de uma *label* de forma intuitiva.
Pré-Condições	• A opção de selecção da *label* tem de se encontrar activa.
Pós-Condições	
Fluxo Principal	1. O utilizador escolhe a opção de criação de uma *label*. 2. O utilizador define, através do mecanismo *drag and drop* o comprimento e a largura da *label*.

5.2.1.6 Caso de Uso "Desenhar Máquina"

A Tabela 13 descreve o caso de uso "Desenhar Máquina" no formato completo.

Tabela 13: Descrição do Caso de Uso "Desenhar Máquina

Caso de Uso "Desenhar Máquina"	
Actor	Utilizador
Partes interessadas	Utilizador: Desenho de uma máquina de forma intuitiva.
Pré-Condições	• A opção de selecção da máquina deve estar activa.
Pós-Condições	
Fluxo Principal	1. O utilizador selecciona a opção de criação de uma máquina. 2. O utilizador define, através do mecanismo *drag and drop* o comprimento e a largura da máquina.

5.2.1.7 Caso de Uso "Gravar Ficheiro"

A Tabela 14 descreve o caso de uso "Gravar Ficheiro" no formato completo.

Tabela 14: Descrição do Caso de Uso "Gravar Ficheiro"

Caso de Uso "Gravar Ficheiro"	
Actor	Utilizador
Partes interessadas	Utilizador: Gravação do ficheiro directo e simples.
Pré-Condições	
Pós-Condições	• O ficheiro tem de conter uma extensão válida (XML).
Fluxo Principal	1. O utilizador vai ao menu *File* e clica na opção *Save* 2. O sistema apresenta uma caixa de diálogo onde o utilizador define um ficheiro para gravação. 3. O utilizador define o ficheiro. 4. O sistema grava os elementos desenhados pelo utilizador nesse ficheiro.
Fluxo Secundário	**3a. O utilizador cancela a escolha do ficheiro** 1. O sistema emite um aviso do sucedido.

5.2.1.8 Caso de Uso “Alterar Setas”

A Tabela 15 apresenta a descrição do Caso de Uso “Alterar Setas” no formato completo.

Tabela 15: Descrição do Caso de Uso “Alterar Setas”

Caso de Uso “Alterar Setas"	
Actor	Utilizador
Partes interessadas	Utilizador: Alteração das setas de forma rápida e intuitiva
Pré-Condições	• Uma linha tem de se encontrar seleccionada.
Pós-Condições	
Fluxo Principal	1. O utilizador escolhe uma linha da área de desenho, clica com o botão secundário do rato e clica na opção “*Edit Arrow...*”. 2. O sistema apresenta uma caixa contendo um formulário de alteração das setas de iníco e de fim. 3. O utilizador preenche esse formulário. 4. O sistema guarda as alterações efectuadas e actualiza a área de trabalho
Fluxo Secundário	***a. A qualquer momento o utilizador pode cancelar as alterações** 1. O sistema não efectua as alterações.

5.2.1.9 Caso de Uso “Alterar Estilo”

A Tabela 16 apresenta a descrição do caso de uso “Alterar Estilo” no formato completo.

Tabela 16: Descrição do Caso de Uso “Alterar Estilo”

Caso de Uso “Alterar Estilo"	
Actor	Utilizador
Partes interessadas	Utilizador: Alteração do estilo do elemento de rápida e intuitiva
Pré-Condições	• Um elemento de desenho (à excepção da máquina) tem de se encontrar seleccionado.
Pós-Condições	
Fluxo Principal	1. O utilizador escolhe um elemento de desenho da área de trabalho, clica com o botão secundário do rato e clica na opção “Edit Shape...”. 2. O sistema apresenta uma caixa contendo um formulário de alteração de propriedades relativas ao estilo do elemento. 3. O utilizador preenche esse formulário. 4. O sistema guarda as alterações efectuadas e actualiza a área de trabalho
Fluxo Secundário	***a. A qualquer momento o utilizador pode cancelar as alterações** 1. O sistema não efectua as alterações.

5.2.1.10 Caso de Uso “Alterar Fonte”

A Tabela 17 apresenta a descrição do caso de uso “Alterar Fonte” no formato completo.

Tabela 17: Descrição do Caso de Uso “Alterar Fonte”

Caso de Uso “Alterar Fonte"	
Actor	Utilizador
Partes interessadas	Utilizador: Alteração da fonte do elemento de forma rápida e intuitiva
Pré-Condições	• Um elemento de desenho (à excepção da máquina) tem de se encontrar seleccionado.
Pós-Condições	
Fluxo Principal	1. O utilizador escolhe um elemento de desenho da área de trabalho, clica com o botão secundário do rato e clica na opção “Edit Font...”. 2. O sistema apresenta uma caixa contendo um formulário de alteração de propriedades relativas à fonte do elemento. 3. O utilizador preenche esse formulário. 4. O sistema guarda as alterações efectuadas e actualiza a área de trabalho
Fluxo Secundário	**O utilizador cancela a escolha do ficheiro** 1. O sistema emite um aviso do sucedido.

5.2.1.11 Caso de Uso “Inserir Imagem”

A Tabela 18 apresenta a descrição do caso de uso “Inserir Imagem” no formato completo.

Tabela 18: Descrição do Caso de Uso “Inserir Imagem”

Caso de Uso “Inserir Imagem"	
Actor	Utilizador
Partes interessadas	Utilizador: Inserção da imagem de uma máquina de rápida e intuitiva
Pré-Condições	• O elemento máquina tem de se encontrar seleccionado.
Pós-Condições	
Fluxo Principal	1. O utilizador selecciona uma máquina da área de trabalho, clica com o botão secundário do rato e clica na opção “Add Image...”. 2. O sistema apresenta uma caixa onde o utilizador selecciona uma imagem para alteração. 3. O utilizador escolhe o ficheiro. 4. O sistema altera a imagem do elemento máquina em causa.
Fluxo Secundário	***a. A qualquer momento o utilizador pode cancelar as alterações** 1. O sistema não efectua as alterações.

5.2.1.12 Caso de Uso “Remover Imagem”

A Tabela 19 apresenta a descrição do caso de uso “Remover Imagem” no formato completo.

Tabela 19: Descrição do Caso de Uso “Remover Imagem”

Caso de Uso “Inserir Imagem"	
Actor	Utilizador
Partes interessadas	Utilizador: Remoção da imagem de uma máquina de rápida e intuitiva
Pré-Condições	• O elemento máquina tem de se encontrar seleccionado e com uma imagem atribuida.
Pós-Condições	
Fluxo Principal	1. O utilizador selecciona uma máquina da área de trabalho, clica com o botão secundário do rato e clica na opção “Remove...”. 2. O sistema remove a imagem do elemento máquina em causa.
Fluxo Secundário	

5.2.2 Diagrama de classes

A Figura 43 ilustra o diagrama de classes utilizado no âmbito da realização deste módulo. Esta estrutura é elaborada de forma a facilitar o acesso aos vários elementos de desenho. As classes *Line*, *Rectangle*, *Elipse*, *Label* e *Machine* referem-se aos elementos que são desenhados pelo utilizador. Os atributos de cada uma destas classes são as várias propriedades que cada elemento de desenho contém, contendo para cada um destes atributos, os métodos *get* e *set* para atribuição e aquisição.

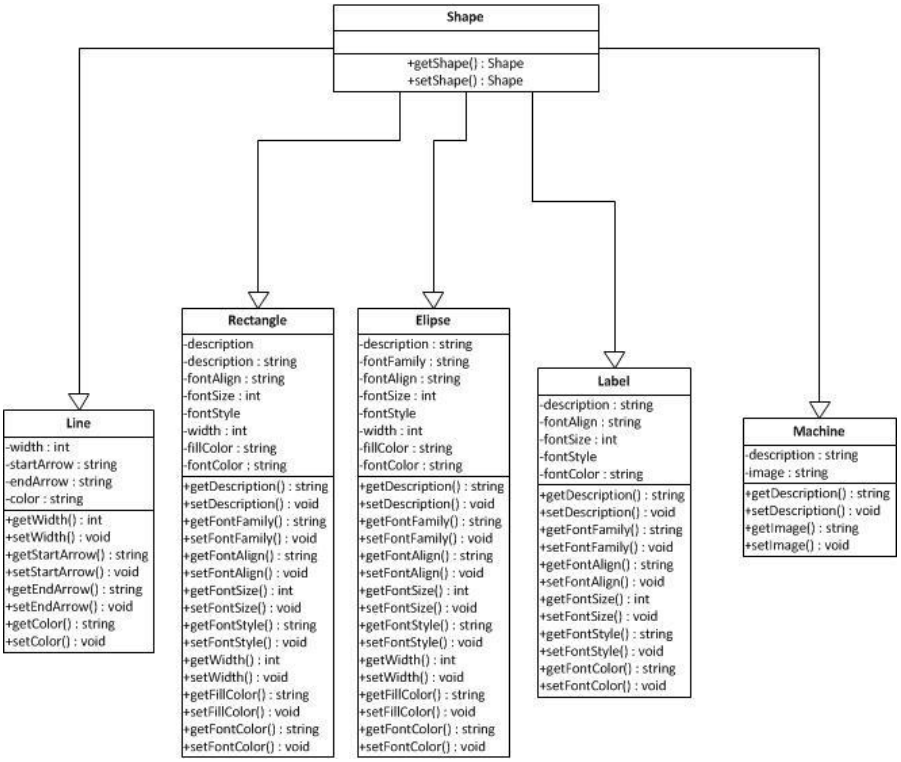

Figura 43: Diagrama de classes do módulo.

5.3 Linguagens e tecnologias utilizadas

Nesta secção apresentam-se, de uma forma breve, as ferramentas e tecnologias utilizadas no âmbito do desenvolvimento do módulo do Editor gráfico proposto, nomeadamente a linguagem Java, a ferramenta NetBeans e a API de desenho jGraph.

5.3.1 **Java**

Java é uma linguagem de programação orientada a objectos, desenvolvida, pela *Sun Microsystems* em 1995, e apresenta semelhanças com outras linguagens de programação, como por exemplo o C, C# ou C++ (URL29).

Como API para o desenvolvimento de interfaces, tal como indicado anteriormente (secção 4.4), será utilizado o Swing pela sua simplicidade e por ser bastante completo em prol das necessidades e requisitos do sistema.

5.3.2 **NetBeans**

NetBeans é um ambiente de desenvolvimento livre e *open-source*, que permite a criação de aplicações Java, C/C++, PHP, JavaScript e Groovy. Foi desenvolvido pela *Oracle Corporation*, e actualmente encontra-se na versão 7.2 (URL30).

5.3.3 **jGraph**

O JGraph é uma biblioteca *open-source* de criação de grafos, sendo a primeira versão lançada em 2001. Contém funcionalidades de criação de elementos visuais, tais como rectângulos e elipses, entre outras, e é possível alterar todo o seu aspecto visual (URL32).

5.4 Protótipo

Após o levantamento dos requisitos do sistema, foi necessário o desenvolvimento de um protótipo para uma melhor preparação para a fase de desenvolvimento.

Para a elaboração deste protótipo foram definidos critérios que influenciaram certas decisões, nomeadamente em termos de interface, pois este módulo necessitava de se integrar com os dos restantes módulos na arquitectura geral do sistema, por razões de consistência e interacção no sistema.

Como janela principal, um esboço do interface esperado e que continha os elementos definidos na secção de levantamento de requisitos (Figura 44).

- Barra de Menus;
- Barra de Ferramentas;
- Janela de Ferramentas;
- Janela de Atributos;
- Área de desenho;

- Barra de estado.

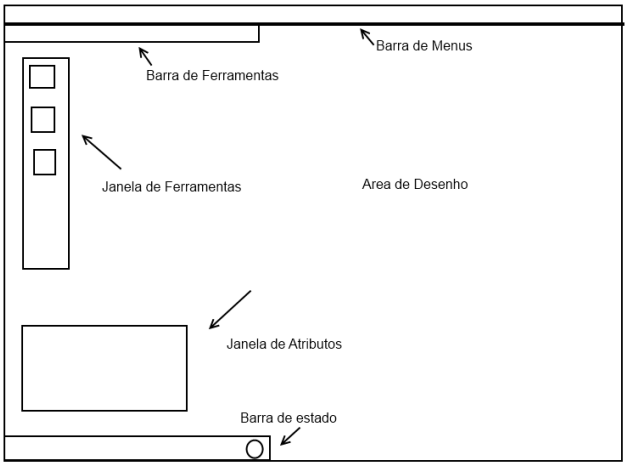

Figura 44: Esboço do interface do módulo.

A barra de menus contém operações de abrir e guardar ficheiro, copiar/cortar/colar e funcionalidades de *undo* e *redo*. A barra de ferramentas contém botões, representados por ícones, onde se efectuam as funcionalidades de copiar, cortar e colar, *undo* e *redo*, e *zoom*. A janela de ferramentas contém as funcionalidades de inserção dos vários elementos de desenho, a janela de atributos contém a descrição do elemento de desenho seleccionado e é possível a alteração dessa descrição. A barra de estado indica se o trabalho realizado se encontra num estado válido (ícone verde) ou inválido (ícone vermelho).

A edição das propriedades dos elementos, apresentada na Figura 45, estaria acessível através de um menu de contexto, despoletado pela acção sobre o botão direito do rato.

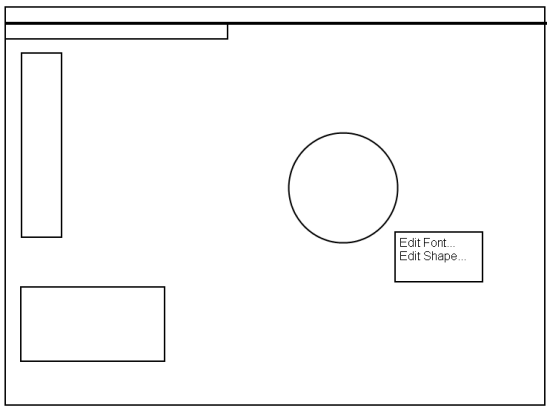

Figura 45: Esboço da janela de edição de propriedades.

5.5 Editor de *layout* fabril

Com a estrutura e o esboço especificado, juntamente com as funcionalidades identificadas, procedeu-se então ao desenvolvimento do protótipo de edição de uma planta fabril. A Figura 46 ilustra a interface da janela principal deste módulo.

O título da janela indica o nome do ficheiro que está a ser editado. Caso seja apresentado um símbolo (*) posterior ao nome, significa que existem alterações na área de desenho ou na janela de atributos. Sempre que o utilizador guarda as alterações efectuadas, esse símbolo desaparece.

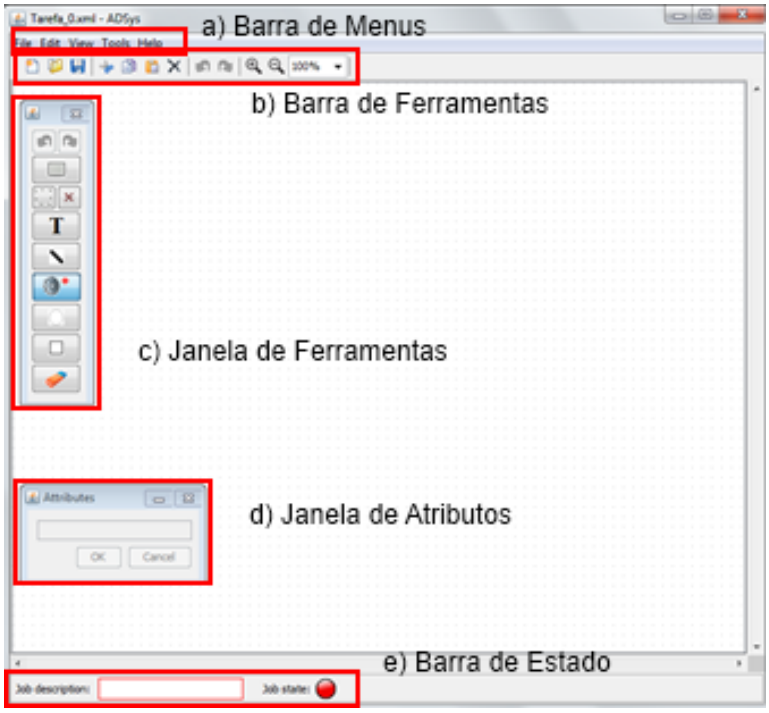

Figura 46: Interface principal do sistema.

A barra de menus, apresentada na Figura 46 (a), contém menus diferentes, cada um com funcionalidades diferentes:

- **Menu File**: contém operações de criação de um novo ficheiro, abertura de um ficheiro já existente, guardar o ficheiro que se encontra em alteração e sair do sistema.
- **Menu Edit**: contém funcionalidades de *Undo* e *Redo*, cortar, copiar, colar e eliminar, e também de selecção e remoção da selecção dos vários elementos da área de desenho do *layout* fabril.

- **Menu Tools**: este menu contém as funcionalidades de mostrar ou esconder as janelas de ferramentas e de atributos.
- **Menu Help**: contém informação acerca do sistema.

A barra de ferramentas, apresentada na Figura 46 (b), contém funcionalidades que são consistentes com os outros módulos, que permitem ao utilizador:

- Criar um novo ficheiro, abrir um ficheiro já existente, guardar as alterações realizadas num ficheiro;
- Cortar, Copiar, Colar e eliminar elementos de desenho;
- Avançar e recuar nas acções realizadas pelo utilizador;
- Efectuar *zoom-in*, *zoom-out* ou definir um nível de *zoom* personalizado;

As funcionalidades de *zoom* permitem ao utilizador a definição de um nível de detalhe do módulo. O utilizador pode utilizar os botões representados por lupas para aumentar ou diminuir gradualmente o nível de detalhe, escolher o mesmo na *Combo Box*, onde contém valores de *zoom* pré-definidos, ou introduzir manualmente um valor. A opção *All* define uma escala de *zoom* automaticamente, para que todo o conteúdo seja visível.

A janela de ferramentas, ilustrada na Figura 46 (c), contém as funcionalidades necessárias para o desenho de uma planta fabril personalizável. Aqui é possível criar máquinas e vários elementos de desenho, tais como *labels*, elipses, rectângulos e linhas. Também contém funcionalidades de selecção única, múltipla, remoção de selecção e remoção de elementos da área de desenho. Nas secções seguintes serão descritas as funcionalidades de cada um destes elementos.

A janela de atributos, ilustrada na Figura 46 (d), contém a descrição de um elemento (máquina, *label*, elipse ou rectângulo) seleccionado, onde é possível alterar a mesma.

A barra de estado, ilustrada na Figura 46 (e), contém uma descrição da planta e de um semáforo, o qual fica vermelho caso algum atributo obrigatório não tenha sido preenchido, tal como a descrição de uma máquina ou a descrição da planta.

Quando o utilizador decide criar um elemento de desenho, o sistema contém a funcionalidade de *drag and drop* para uma criação mais eficiente, mais eficaz e mais

intuitiva para o utilizador. A Figura 47 ilustra um exemplo da criação de um elemento de desenho.

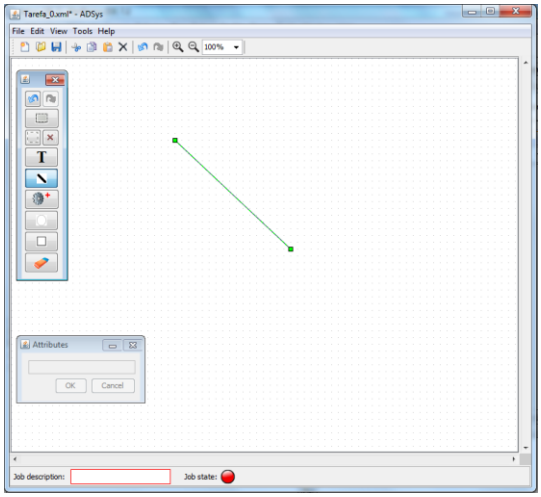

Figura 47: Criação de um elemento.

5.5.1 Edição de labels

Esta funcionalidade, tal como o nome indica, permite a edição visual das *labels* existentes na área de trabalho. Quando o utilizador acciona o botão direito do rato, aparece um *Pop-up Menu* contendo a opção "*Edit Font…*" (Figura 48).

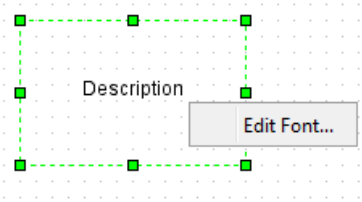

Figura 48: Edição de uma label.

Ao seleccionar a representada, aparece uma caixa de diálogo, ilustrada na Figura 49, onde o utilizador pode alterar as propriedades da fonte da *label* respectiva, tais como, o tipo de letra, o alinhamento, o estilo, o tamanho e a cor. Sempre que o utilizador clica nesta opção já se encontram seleccionadas as opções referentes às propriedades da *label* seleccionada.

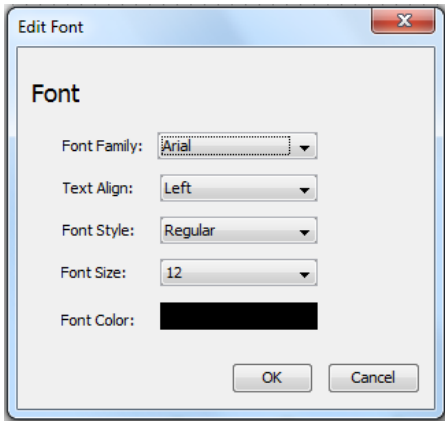

Figura 49:Edição das propriedades do texto.

5.5.2 **Edição de linhas**

A edição de linhas permite a edição de todas as propriedades relacionadas com o seu estilo visual, tais como, a cor da linha, a espessura e o tipo de seta das extremidades das linhas. O utilizador, ao accionar o botão direito do rato é confrontado com duas opções: "*Edit Arrows…*" e "*Edit Shape…*" (Figura 50).

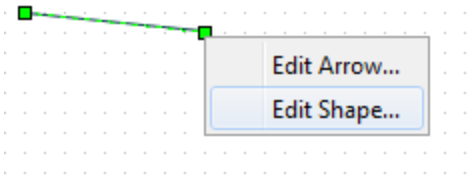

Figura 50: Edição de linhas.

Na opção '*Edit Arrows*', o utilizador define a aparência das setas iniciais e finais das linhas, recorrendo a uma caixa de diálogo (Figura 51).

Na opção '*Edit Shape*', o utilizador define a espessura da linha, bem como a respectiva cor (Figura 51).

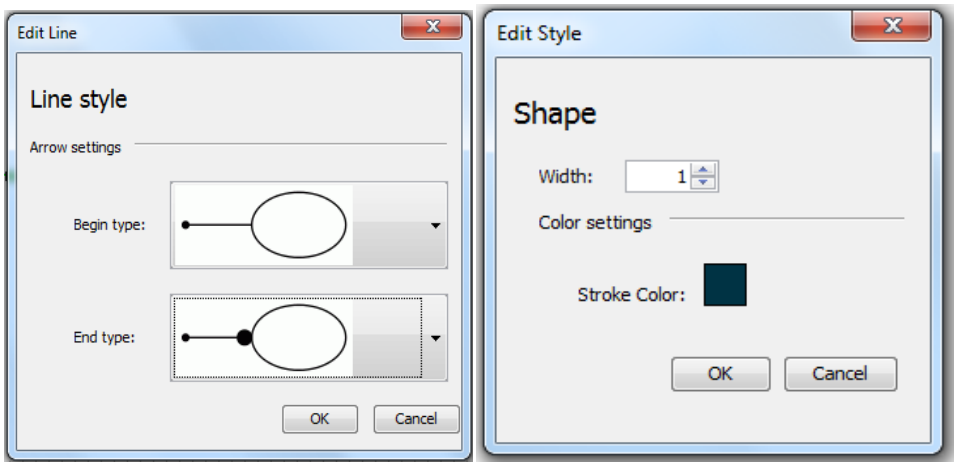

Figura 51: Edição das setas inicial e final das linhas e estilo das linhas

5.5.3 **Edição de máquinas**

Esta funcionalidade serve para a inserção ou remoção de uma imagem a ser atribuída à máquina seleccionada. As opções correspondentes surgem quando o utilizador clica com o botão secundário do rato, e a opção de remoção só se encontra activada quando já existe uma imagem atribuída à máquina. A Figura 52 ilustra esta funcionalidade.

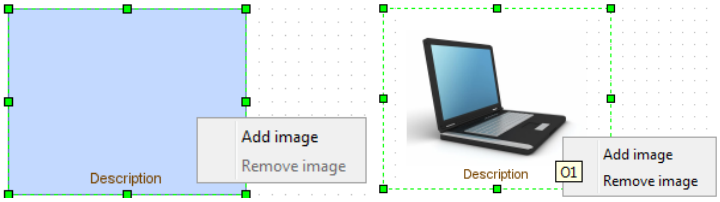

Figura 52: a) Edição de uma máquina sem imagem. b) Edição com imagem

Ambas as opções encontram.se activadas quando uma imagem já se encontra atribuída, pois é possível a alteração de uma imagem.

Uma das validações necessárias para uma correcta definição de um *layout* fabril é a existência de uma descrição nas máquinas. Caso isto não aconteça, o semáforo representado na barra de estado é ilustrado com uma cor vermelha, e o utilizador ao passar por cima de uma máquina que não contenha uma descrição, aparecerá um *tooltip* descrevendo o problema encontrado nessa máquina, representado na Figura 53.

Figura 53: Validação de máquinas.

5.5.4 **Edição de elipses**

Esta funcionalidade permite a edição das componentes visuais das elipses tais como o texto e as várias propriedades da forma (Figura 54).

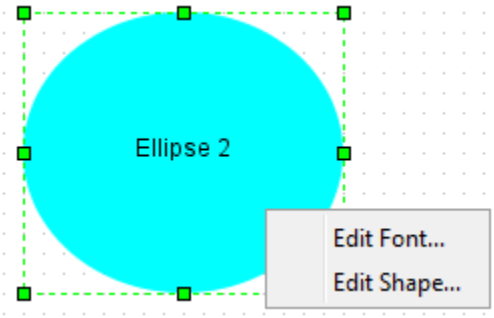

Figura 54: Edição de uma elipse.

Utilizando o mesmo processo, o botão direito do rato espoleta um *Pop-up Menu* com duas opções: "*Edit Font*...", apresentando a mesma caixa de edição da fonte indicada na edição de *labels*, e "*Edit Shape*...", uma caixa de diálogo semelhante à das linhas, mas com uma propriedade extra de cor de preenchimento, ilustrada na Figura 55.

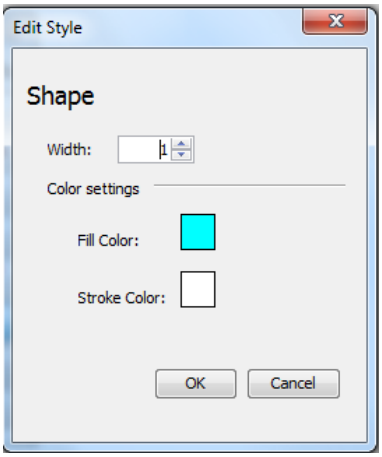

Figura 55: Edição do estilo de uma elipse.

5.5.5 Edição de rectângulos

Esta componente contém as mesmas características que a edição de elipses, ou seja, contém as funcionalidades de edição de texto e edição das propriedades da forma, ambas estas funcionalidades acessíveis através do clique com a tecla direita do rato.

5.5.6 Ferramenta de remoção de elementos

O sistema permite a remoção de elementos de desenho já criados. A utilização de um clique para a activação desta ferramenta permite a remoção de apenas um elemento de desenho. Após esse procedimento, a opção anteriormente seleccionada é activada. Se o utilizador pretender remover múltiplos elementos sem necessitar constantemente de seleccionar a opção de remoção de desenho, essa funcionalidade é accionada efectuando um duplo clique na ferramenta de remoção.

5.5.7 Operações de *undo* e *redo*

O sistema permite o avanço e o recuo de acções realizadas pelo utilizador. Desta forma aumenta a segurança e o conforto do utilizador, na medida em que pode usufruir do sistema de uma forma mais liberal. Estas operações abrangem a criação e remoção de elementos, e o posicionamento e redimensionamento dos mesmos, e estão acessíveis na barra de menus, na barra de ferramentas, na janela de ferramentas ou através da utilização das teclas CTRL+Z (*undo*) e CTRL+Y (*redo*).

5.5.8 Operações de selecção de elementos

O sistema permite a selecção de elementos para a personalização dos mesmos. Possibilita vários tipos de selecção: selecção normal, selecção em área, *select all* e *select none*. Selecção única permite a selecção de um elemento de desenho. A utilização da tecla CTRL permite a selecção múltipla de vários elementos. Selecção em área permite a selecção de vários elementos de desenho definindo uma área, e todos os elementos de desenho encontrados dentro dessa área são seleccionados. *Select all* permite a selecção de todos os elementos de desenho existentes na área de trabalho e *select none* remove a selecção dos elementos seleccionados.

5.5.9 **Operações de *clipboard***

O sistema permite operações de *clipboard* tais como o cortar, copiar, colar. Estas podem ser efectuadas em apenas um elemento, ou num grupo de elementos seleccionados.

5.5.10 **Replicação de geometria**

O sistema permite a replicação de geometria, copiando elementos seleccionados. Para além das funcionalidades de copiar e colar, seleccionado os elementos em causa, premindo a tecla CTRL e arrastando os respectivos elementos para outra zona, esses mantêm o seu posicionamento e uma réplica desses elementos é criada na zona especificada pelo utilizador.

5.5.11 **Exportação do *layout* fabril elaborado**

Esta funcionalidade serve para guardar toda a área de trabalho desenvolvida pelo utilizador num ficheiro XML. O excerto ilustrado na Figura 56 mostra a estrutura do ficheiro XML correspondente. Este ficheiro representa um *layout* fabril com uma, e contém 5 elementos (forma): uma *label*, uma máquina, uma elipse, um rectângulo e uma linha.

O elemento *chao* contém dois atributos: a descrição (*description*) que, tal como o nome indica, contém a descrição da planta fabril; o estado (*state*) indica se a planta criada se encontra num estado válido ou inválido. Neste caso encontra-se inválido pois o atributo *description* do elemento *chao* encontra-se vazio.

Todas as formas contêm vários atributos que são comuns entre eles, tais como a posição da forma (*xi*, *xf*, *yi*, *yf*), a forma (*rectangulo*, *elipse*, etc.), o tamanho (*height* e *width*) e uma identificação (*id*).

```
<chao description="" state="invalid">
   <formas>
      <forma align="center" fillColor="none" fontColor="#000000"
         fontFamily="Arial" fontSize="12" fontStyle="0" forma="label"
         height="20.0" id="O1" strokeColor="none" strokeWidth="0"
         value="Descrição" width="60.0" xCoord="150.0" yCoord="70.0"/>
      <forma forma="maquina" height="100.0" id="O2" value="Máquina 1"
         width="100.0" xCoord="330.0" yCoord="110.0"/>
      <forma align="center" fillColor="#0000ff" fontColor="#ff0033"
         fontFamily="Times New Roman" fontSize="16" fontStyle="0"
         forma="elipse" height="100.0" id="O3" strokeColor="#ffff00"
         strokeWidth="3" value="Elipse" width="100.0" xCoord="200.0" yCoord="240.0"/>
      <forma align="left" fillColor="#ff6600" fontColor="#00ffff"
         fontFamily="Arial" fontSize="20" fontStyle="3"
         forma="rectangulo" height="100.0" id="O5"
         strokeColor="#666600" strokeWidth="3" value="Rectângulo"
         width="110.0" xCoord="320.0" yCoord="230.0"/>
      <forma endArrow="block" forma="linha" height="0.0" id="P1"
         startArrow="classic" strokeColor="#003344" strokeWidth="1"
         value="" width="0.0" xf="260.0" xi="150.0" yf="200.0" yi="110.0"/>
   </formas>
</chao>
```

Figura 56: Exemplo de um ficheiro XML.

5.5.12 **Exemplo elaborado de um *layout* fabril**

Para um melhor análise da potencialidade deste módulo foi criado um exemplo completo de um *layout* fabril (Figura 57). Este foi utilizado na sessão de avaliação de usabilidade do sistema.

Neste exemplo foram utilizadas 7 máquinas, cada uma destas contendo diferentes imagens e descrições e a ordem de funcionamento das máquinas é representada por setas. Os restantes elementos são postos de trabalho onde os fabricantes suportam a elaboração das tarefas.

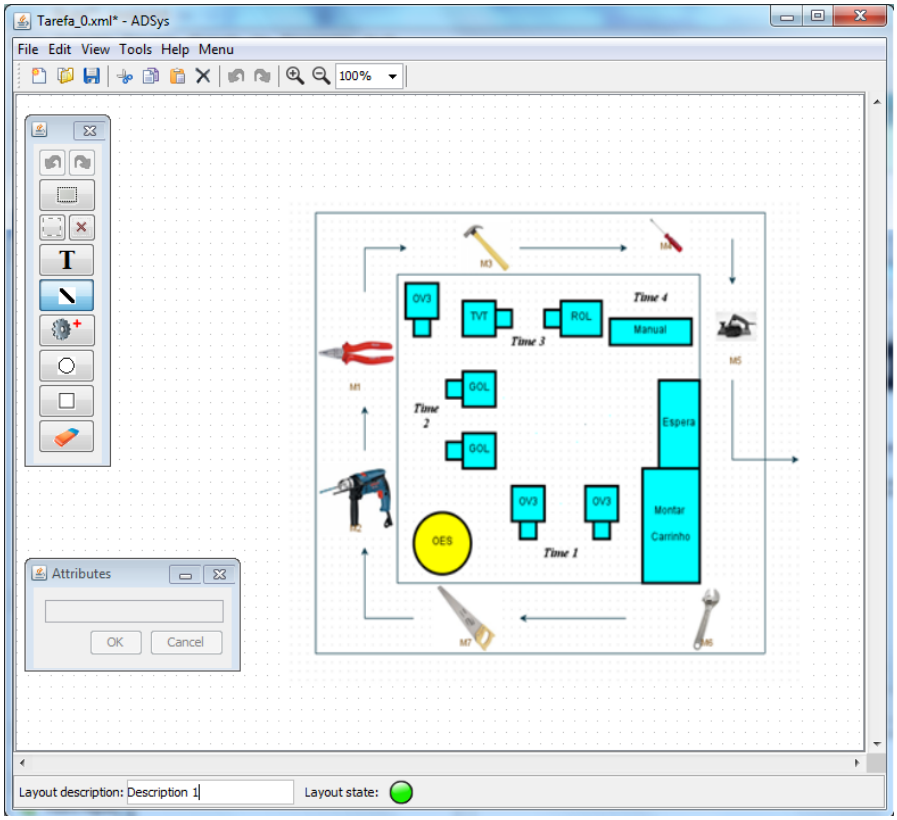

Figura 57: Exemplo de um layout fabril utilizando o módulo desenvolvido.

5.6 Conclusão

Neste capítulo procedeu-se ao desenvolvimento do protótipo de edição de um *layout* fabril. Inicialmente efectuou-se um levantamento de requisitos, descrevendo casos de uso e diagrama de classes, seguido da elaboração de um protótipo. Por fim procedeu-se à descrição das várias funcionalidades e aspectos visuais, acompanhados com figuras, à exemplificação de um ficheiro XML e a um exemplo elaborado de utilização deste módulo, exemplo este que foi utilizado no âmbito da avaliação de usabilidade.

6 Análise de Usabilidade

Neste capítulo é descrito o estudo ao módulo de edição do *layout* fabril, desenvolvido no âmbito desta tese de mestrado, através da concretização de uma análise de usabilidade. Por fim serão indicadas algumas melhorias com base num inquérito elaborado para vários utilizadores, e nas respectivas sugestões de melhoria.

6.1 Plano de análise de usabilidade

Após o desenvolvimento do módulo de edição do *layout* fabril, as necessidades funcionais dos utilizadores foram concretizadas. Mas também é necessário elaborar um estudo deste módulo ao nível da interacção com o utilizador. Para tal foi elaborado um conjunto de critérios, a serem preenchidos pelos avaliandos, de modo a recolher uma avaliação ao nível de usabilidade do sistema. Estes critérios foram definidos de modo a poder recolher informação ao nível de eficiência, erros, aprendizagem, memorabilidade e satisfação, e são avaliados para as funcionalidades deste módulo, sendo elas, de um modo mais abrangente:

- O *layout* do sistema;
- A inserção e remoção de elementos de desenho;
- A alteração das propriedades dos elementos;

6.1.1 Avaliação do *layout* do sistema

Para a avaliação do *layout* do sistema são referidos aspectos tais como o seu aspecto visual e o nível de satisfação do sistema. Em seguida são avaliados outros conceitos ao nível de aprendizagem, intuição dos ícones usados (se os ícones são de fácil compreensão), consistência e facilidade de integração do utilizador com o *layout* proposto.

6.1.2 Avaliação dos processos de inserção e remoção dos elementos de desenho

Nesta secção faz-se um levantamento dos vários aspectos a serem analisados relativamente aos processos de inserção e remoção dos elementos de desenho, tendo em conta os vários princípios de *design* de interacção. De referir:

- Interpretação e utilização dos *palette* menus;

- Inserção de linhas, máquinas e de outros elementos de desenho (*labels*, elipses, rectângulos, quadrados);
- Manipulação dos elementos de desenho ao nível de alteração da posição dos mesmos.

6.1.3 Avaliação do processo de alteração das propriedades dos elementos de desenho

Nesta secção faz-se um levantamento dos vários aspectos a serem analisados relativamente ao processo de alteração das propriedades dos elementos de desenho. De um modo genérico é elaborada uma análise às seguintes componentes:

- Uso do botão direito para a exibição do *pop-up* menu de alteração das propriedades;
- Clareza na descrição dos menus;
- Nível de satisfação da caixa de edição das propriedades dos elementos;
- Facilidade de alteração das propriedades;
- Adequação dos estilos de interacção utilizados.

6.2 Metodologia para a sessão de avaliação de usabilidade

Após o levantamento dos vários aspectos a serem analisados ao nível de interacção do utilizador com o sistema, foi criado um guião, que se encontra no Anexo 1, duma sessão informal de avaliação da usabilidade do sistema. Esse documento é composto várias componentes: introdução, descrevendo os objectivos da sessão de avaliação e o roteiro; um teste de eficiência a ser realizado utilizando o módulo desenvolvido; e um questionário a ser elaborado após a realização do teste de eficiência.

A introdução tem como objectivo contextualizar o utilizador nas tarefas que estes devem realizar e o esclarecimento do questionário para assegurar a veracidade e validação das respostas.

O teste de eficiência consiste na realização de um conjunto de tarefas de modo a que o resultado se aproxime ao máximo do pretendido. Neste caso, os participantes devem elaborar um *layout* fabril, utilizando elementos de desenho disponibilizados pelo módulo.

O questionário, preenchido após a realização do teste de eficiência, pretende obter uma opinião pessoal sobre vários aspectos de usabilidade a nível de aspecto visual e funcionalidades. Numa fase final são elaboradas questões relativamente aos cinco aspectos de usabilidade: eficiência, aprendizagem, memorabilidade, erros e satisfação (Nielsen, 1993). Também é disponibilizado um espaço dedicado a comentários, sugestões ou críticas relativamente ao módulo de edição.

A sessão de avaliação tem como intuito a recolha dos dados relativos a este módulo, os quais serão tratados posteriormente. A maior parte das questões são respondidas entre os valores 1 e 5, mas a análise de eficiência, ou seja, o tempo que o utilizador necessita para a elaboração do *layout* fabril é um valor numérico representado em minutos.

6.3 Sessão de avaliação do protótipo

No dia 25 de Outubro de 2012, no GECAD pelas 9 horas, procedeu-se à sessão de avaliação do protótipo. Nesta sessão pretendia-se uma análise, através de demonstração, experimentação e preenchimento de um inquérito, do módulo desenvolvido no âmbito deste trabalho de mestrado. Esta sessão contou com vários participantes: docentes do ISEP, elementos que participaram no projecto ADSyS e alunos. Segundo Robert A Virzi (Virzi, 1992), 80% de problemas de usabilidade são encontrados na utilização de 4 ou 5 elementos para avaliação do protótipo. Nesta sessão de avaliação do protótipo participaram sete elementos: cinco com um envolvimento directo no projecto ADSyS e dois sem qualquer relacionamento. O envolvimento destes dois utilizadores que não obtinham nenhuma experiência prévia acabou por constituir uma mais-valia nas sugestões e críticas deste módulo. A Figura 58 (Virzi, 1992) ilustra um gráfico, representando a proporção de problemas encontrados em função dos sujeitos de avaliação do protótipo. Analisando esta figura, e com base no número de participantes da sessão de avaliação de usabilidade do protótipo, é possível identificar cerca de 90% dos problemas.

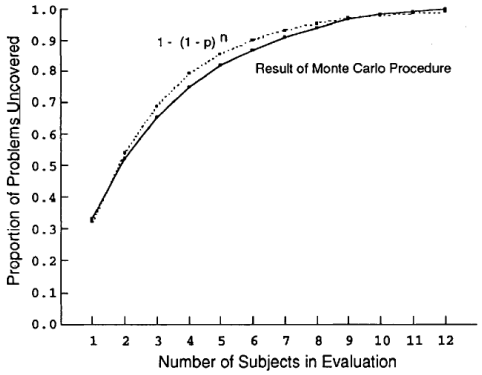

Figura 58:Proporção de problemas encontrados em função dos sujeitos (Virzi, 1992).

6.4 Análise de resultados

Nesta secção efectua-se uma apresentação e análise dos resultados obtidos no questionário realizado após a realização do teste de eficiência do módulo. A classificação das respostas obtidas variam entre dois valores, onde o valor máximo é 5 (cinco), o valor médio é 3 (três) e o valor mínimo é 1 (um).

Os resultados a serem apresentados pertencem ao inquérito realizado pelos participantes. A classificação média das avaliações dos mesmos é ilustrada nos seguintes gráficos.

6.4.1 Aspecto Visual

Relativamente ao aspecto visual os utilizadores encararam de forma positiva os pontos simplicidade do aspecto visual e adequação das cores utilizadas, embora no último ponto tivessem sido sugeridas algumas melhorias (Figura 59).

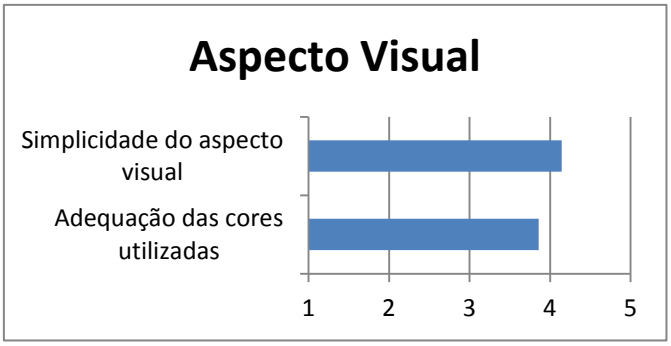

Figura 59: Classificação média do aspecto visual.

No gráfico da Figura 60 encontra-se a classificação média atribuída pelos participantes ao nível da interpretação da barra de menus.

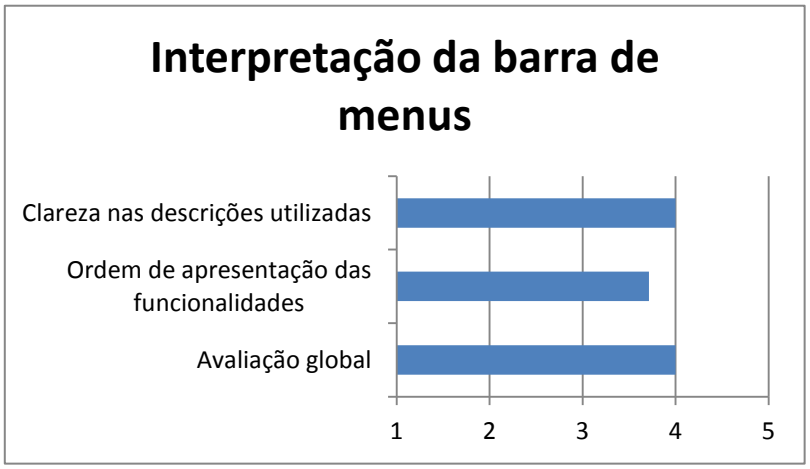

Figura 60: Classificação média dos elementos associados à barra de menus.

Este gráfico sugere que os utilizadores, a nível do aspecto da barra de menus, obtiveram uma experiência positiva, mas que o ponto "Ordem de apresentação das funcionalidades" requer algumas modificações.

A Figura 61 ilustra pontos de avaliação do módulo relativamente ao aspecto da barra de ferramentas, e respectivas classificações atribuídas pelos participantes da sessão de avaliação. Efectuando uma análise, os utilizadores obtiveram uma experiência positiva na utilização da barra de ferramentas, encontrando-se todos estes pontos acima de quatro valores.

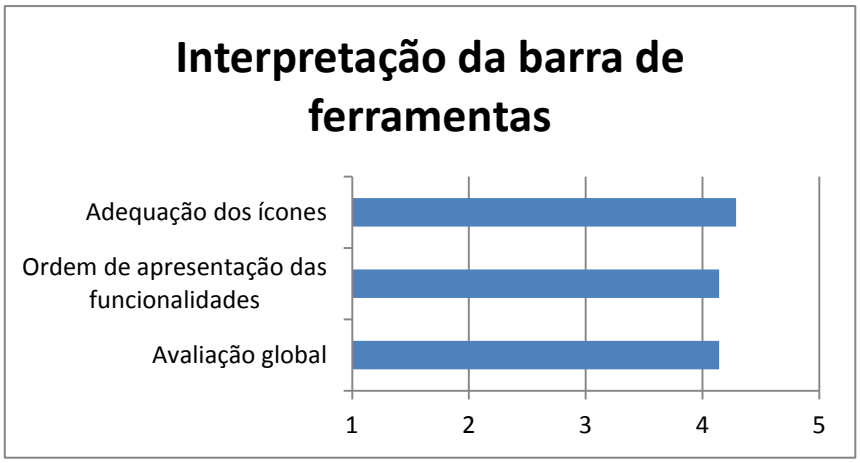

Figura 61: Classificação média dos elementos relativos à barra de ferramentas.

A Figura 62 ilustra um gráfico de barras onde se apresentam os pontos referentes à interpretação da janela de ferramentas, nomeadamente acerca dos ícones existentes na mesma.

Estes pontos obtiveram uma classificação bastante positiva por parte dos utilizadores. O ponto "Interpretação dos ícones de inserção de elementos de desenho" foi o que obteve pior classificação, ainda que positiva.

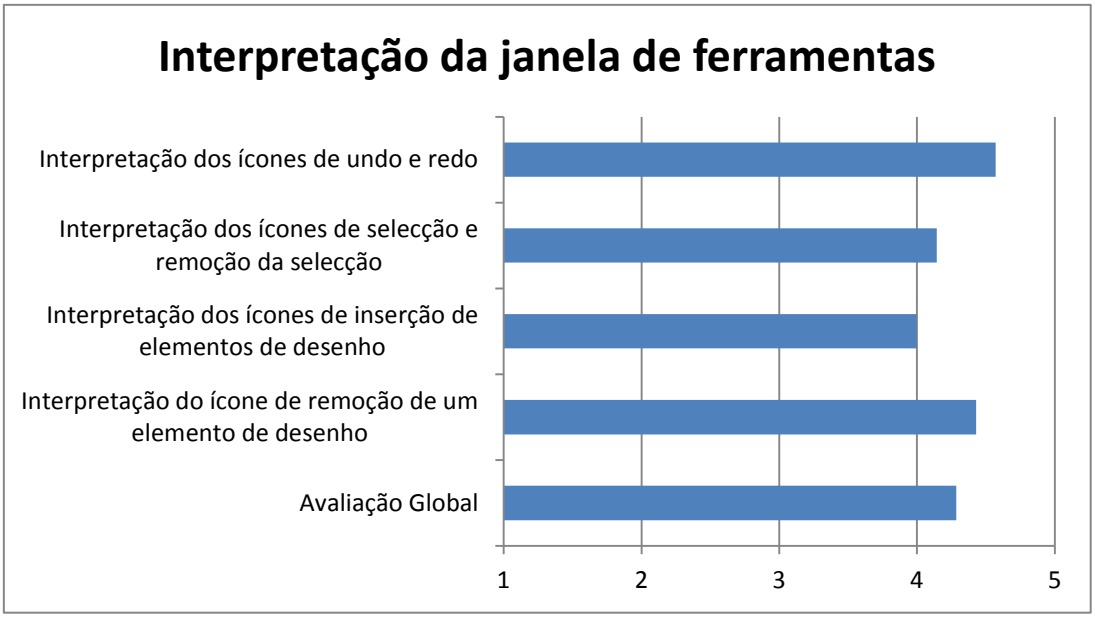

Figura 62: Classificação média dos elementos relativos à janela de ferramentas.

A Figura 63 ilustra um gráfico onde são apresentados aspectos referentes à interpretação da barra de estado, nomeadamente a interpretação do semáforo e do contorno vermelho e verde na caixa te texto existente nessa barra. Estas propriedades obtiveram nota positiva, embora não superior a quatro, o que identifica a necessidade de serem efectuadas algumas melhorias nestes aspectos.

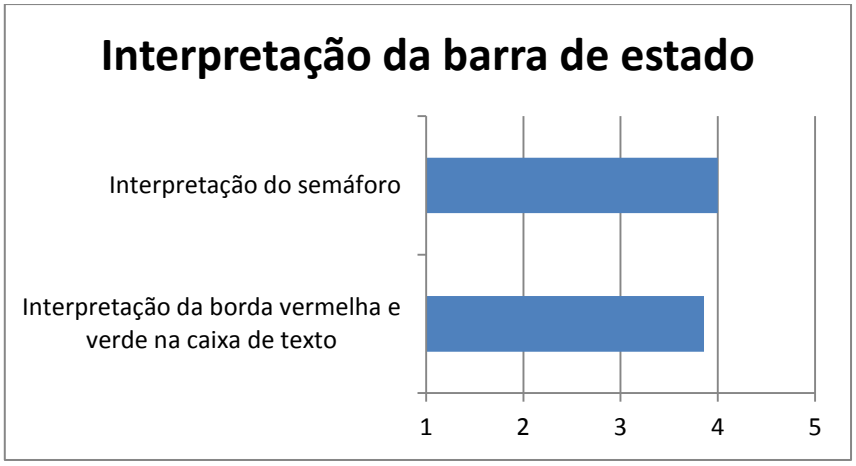

Figura 63: Classificação média dos elementos da barra de estado.

A Figura 64 ilustra um gráfico onde se classifica o aspecto visual do módulo de uma forma global, tendo em conta todos os pontos analisados anteriormente. Este resultado sugere que se efectue algumas melhorias quanto aos vários aspectos indicados anteriormente.

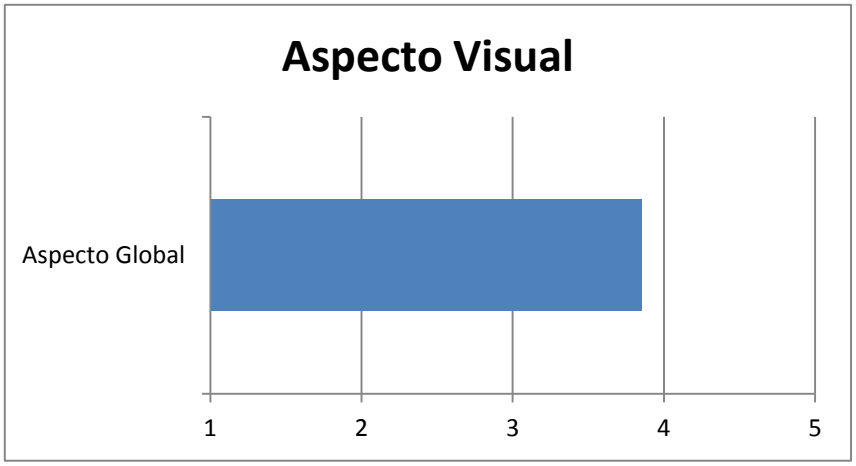

Figura 64: Classificação média dos elementos relativos ao aspecto visual.

6.4.2 Inserção de elementos de desenho

Nesta secção é elaborada uma análise aos resultados obtidos nas funcionalidades de inserção de elementos de desenhos, ilustrando um gráfico onde se classificam os vários pontos referentes a estas funcionalidades, acompanhados com uma descrição.

A Figura 65 ilustra um gráfico onde são apresentados e classificados os pontos referentes às funcionalidades de inserção de elementos de desenho. Analisando o

gráfico conclui-se que os utilizadores obtiveram uma experiência positiva na realização das várias funcionalidades de inserção de elementos de desenho. A inserção de elipses e rectângulos foi mais fácil para o utilizador do que a inserção de máquinas, linhas ou *labels*. A funcionalidade *drag and drop* obteve pior classificação, de onde se se conclui a necessidade de melhorias nesta funcionalidade.

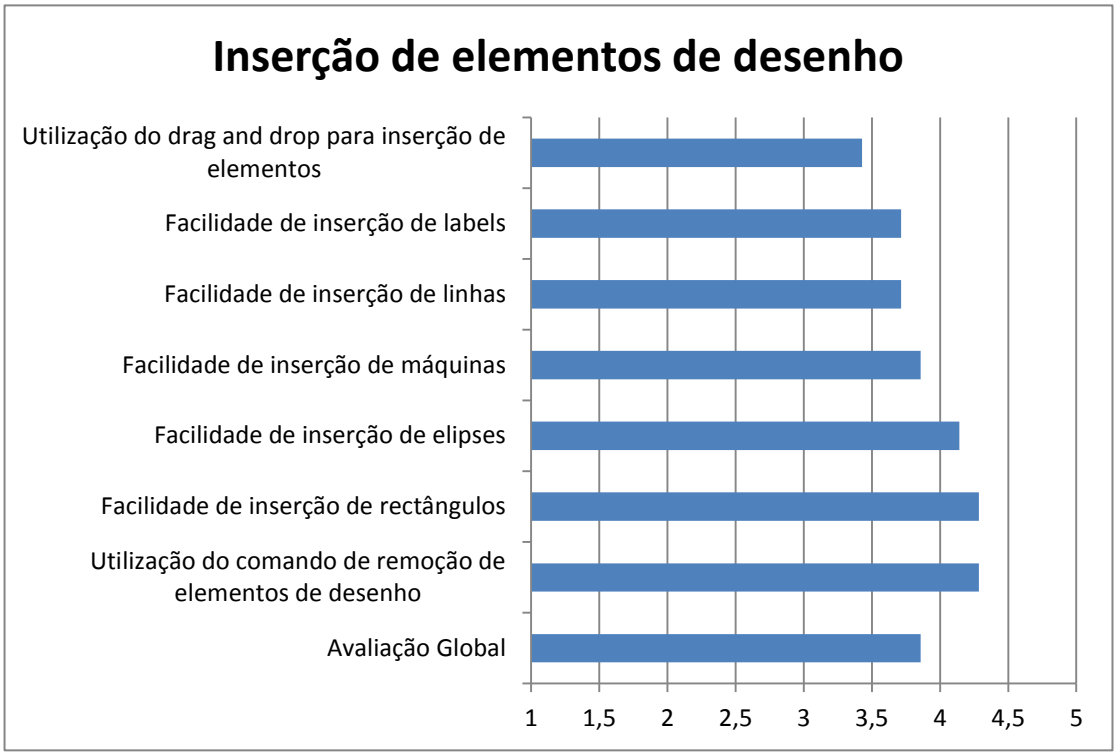

Figura 65: Classificação média das funcionalidades de inserção de elementos.

6.4.3 **Alteração das propriedades dos elementos de desenho**

Nesta secção são analisadas e classificadas as funcionalidades de alteração das propriedades dos vários elementos de desenhos, acompanhados com gráficos onde são ilustradas as classificações atribuídas pelos participantes da sessão de avaliação e de uma descrição.

A Figura 66 ilustra um gráfico onde são apresentadas e classificadas, pelos participantes da sessão de avaliação, as funcionalidades referentes à alteração das propriedades das *labels*. A avaliação global por parte dos utilizadores é positiva, mas sugere a necessidade de efectuar algumas melhorias.

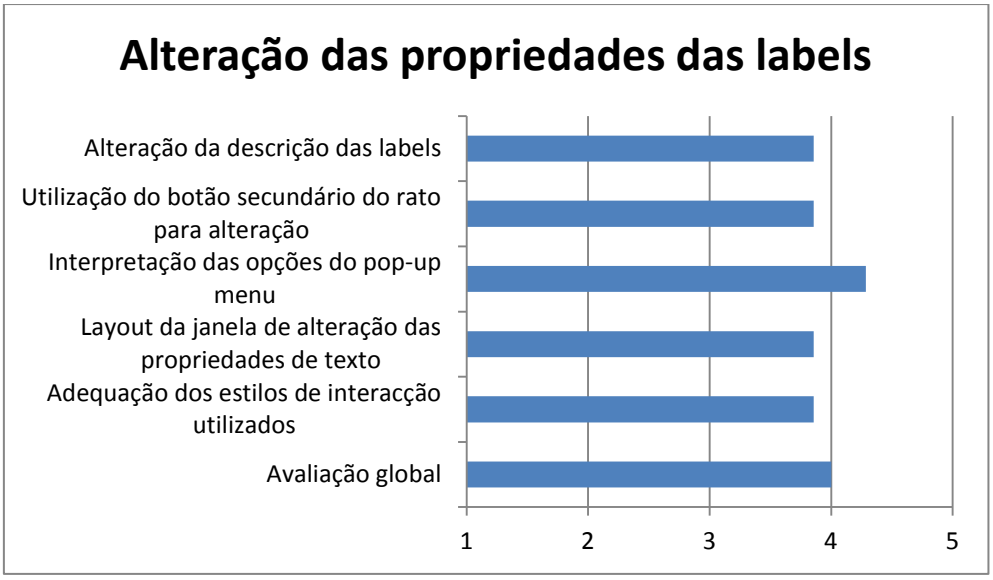

Figura 66: Classificação média dos elementos de alteração de labels.

Os participantes, de um modo geral, obtiveram uma impressão positiva, mas para a maior parte destas funcionalidades foram sugeridas alterações e melhorias, principalmente no ponto "Interpretação dos ícones de alteração das setas de início e de fim", que obteve a pior classificação. O ponto "Facilidade de alteração da cor da linha" obteve melhor classificação. Os restantes pontos são positivos, mas não muito positivos, pelo que são requeridas melhorias (Figura 67).

Figura 67: Classificação média dos elementos de alteração de labels.

A Figura 68 ilustra um gráfico onde se apresentam e classificam vários pontos referentes a todo o processo de alteração das propriedades das elipses e rectângulos, visto estes conterem exactamente as mesmas funcionalidades.

Os participantes, de um ponto de vista geral, encararam esta funcionalidade de forma positiva mas com algumas divergências em alguns aspectos. O ponto "Facilidade de alteração das propriedades da cor" obteve a melhor classificação. O ponto "Facilidade de alteração de propriedades da cor" não foi classificado por um utilizador.

Figura 68: Classificação média dos elementos de alteração de elipses e rectângulos.

A Figura 69 ilustra a avaliação global do módulo de edição das várias propriedades dos vários elementos. Conclui-se que os utilizadores obtiveram uma experiência positiva, mas que são necessárias melhorias para uma melhor usabilidade deste módulo.

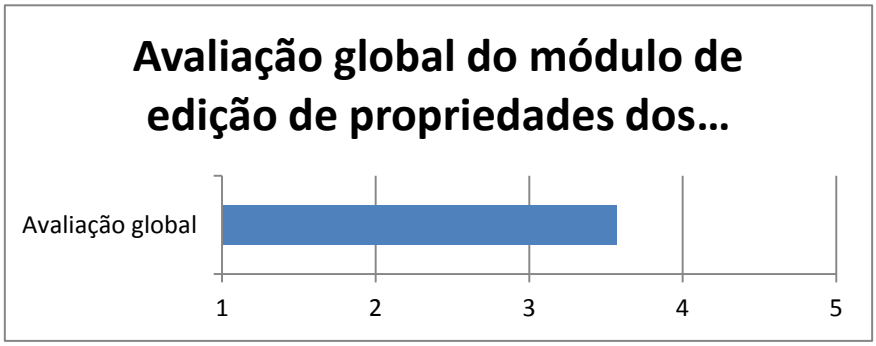

Figura 69: Classificação média da avaliação global.

6.4.4 **Aspecto Geral**

Nesta secção são apresentados e classificados os cinco aspectos de usabilidade (Nielsen,1993). A eficiência é determinada consoante o tempo de execução do teste de eficiência por parte de cada participante da sessão de avaliação. A Figura 70 ilustra, para cada utilizador, o tempo despendido na realização deste teste de eficiência. Os participantes realizaram o teste de uma forma rápida, embora alguns mais rapidamente do que outros, de onde se conclui que o sistema é eficiente. O participante 4 levou mais tempo a realizar o teste de eficiência pois este esperava que a funcionalidade de copiar formatação de um elemento encontrasse desenvolvido.

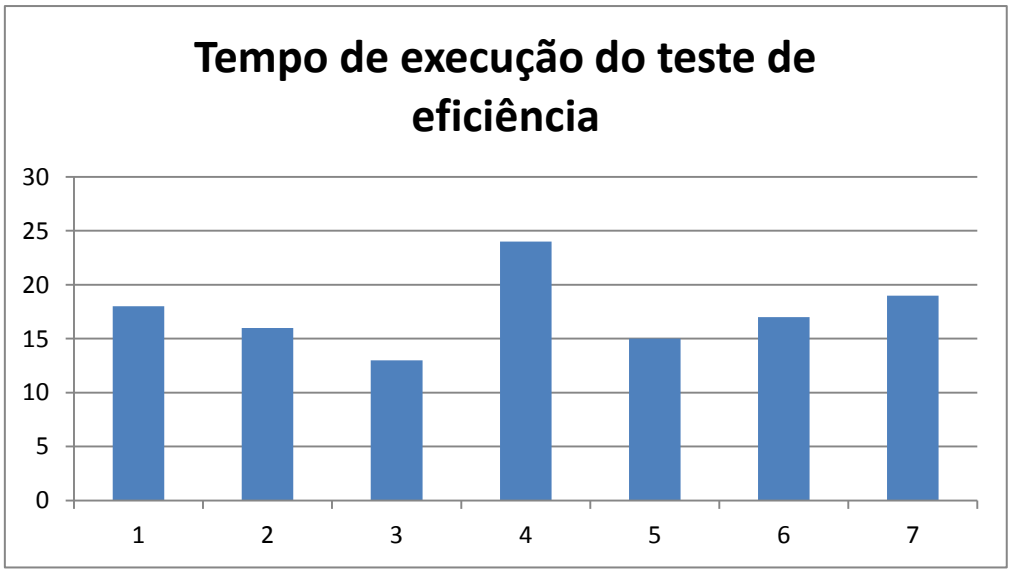

Figura 70: Tempo de execução do teste de eficiência de cada participante.

A Figura 71 ilustra um gráfico onde se apresentam e classificam quatro aspectos relativamente ao módulo em geral. Com este gráfico conclui-se que o sistema é pouco amigável no tratamento de erros efectuados pelo utilizado. Em contra partida em termos de memorabilidade e aprendizagem, o sistema é intuitivo, consistente e fácil de utilizar.

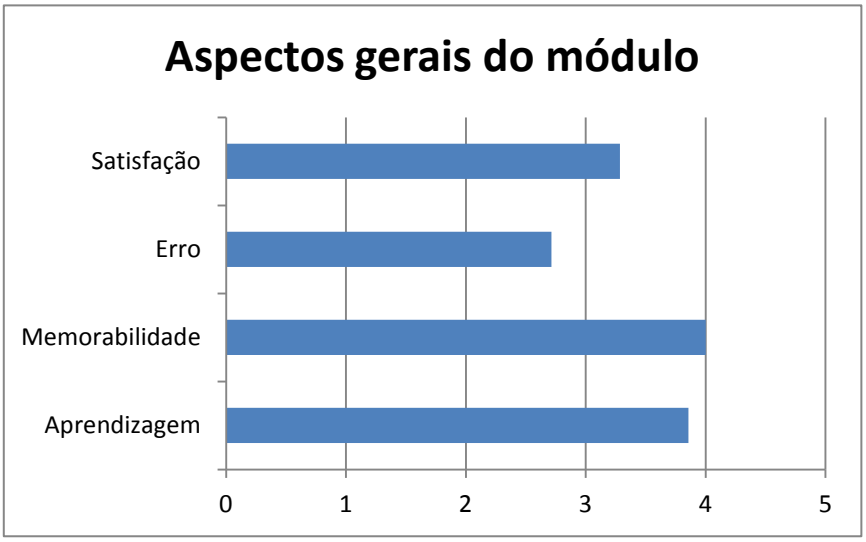

Figura 71: Classificação média dos aspectos gerais do módulo.

Em seguida foi elaborada uma tabela onde constam críticas e sugestões mais construtivas e relevantes, seguidas da sua gravidade. A Tabela 20 representa a lista de sugestões ou críticas definidas pelos participantes. Grande parte das sugestões são de gravidade média, o que causa alguma preocupação na realização destas melhorias/críticas/sugestões, embora algumas sendo de implementação mais fácil que outras. Também foi encontrado um problema de alta gravidade, nomeadamente na funcionalidade de leitura do ficheiro XML. A sua resolução implica maior prioridade.

Tabela 20: Levantamento de sugestões / críticas ao protótipo

Descrição da sugestão/crítica	Gravidade
Edição de propriedades também acessível na barra de ferramentas.	Média
Possibilidade de copiar formatação e estilo de um elemento	Média
Importação do ficheiro XML não se encontrava em total funcionamento.	Alta
Alteração da orientação da linha.	Média
Definição de um tamanho mínimo para os elementos de desenho	Média
Janelas de edição das propriedades centradas.	Baixa
Alterar o nome da propriedade "Shape" para não enganar o utilizador.	Média
Alteração das imagens de selecção de tipo de seta.	Média
Bugs no mecanismo "*undo*" / "*redo*".	Alta
Melhoramento de *tooltips.*	Média
Caixa de atributos a vermelho quando a descrição está em falta.	Baixa
Transparência dos ícones da janela de ferramentas.	Baixa
Edição de vários objectos simultaneamente.	Média
Funcionalidade "*repeat*"	Média
Possibilidade de criação de *layers*	Baixa
Possibilidade de criação de linhas poligonais	Baixa

6.5 Conclusão

Neste capítulo procedeu-se à análise de usabilidade do protótipo de edição de um *layout* fabril desenvolvido no âmbito deste trabalho de mestrado. Esta análise implicou a elaboração de um inquérito relacionado com aspectos de usabilidade a ser preenchido, na sessão de avaliação da usabilidade deste protótipo, pelos respectivos participantes desta sessão. Com base nestes resultados foram elaborados gráficos e

uma tabela onde, no primeiro, para cada grupo de funcionalidades, se indicava a classificação média de um grupo de funcionalidades; no segundo efectuou-se um levantamento de algumas sugestões e críticas elaboradas pelos participantes da sessão de avaliação da usabilidade do protótipo. Estas sugestões poderão ser tidas em conta como trabalho futuro.

7 Conclusões

Neste capítulo procede-se à elaboração do resumo do relatório, efectuando uma sistematização dos aspectos tratados nos capítulos anteriores. Também são indicados os objectivos realizados no âmbito deste trabalho de mestrado, indicando também funcionalidades a serem inseridas ou melhoradas.

Interfaces e usabilidade são conceitos de extrema importância não só para o desenvolvedor de *software*, como para o utilizador. Um sistema com fraca usabilidade é muito mais improvável de obter sucesso que um sistema com uma usabilidade boa. Existem normas, linhas de orientação e guias de estilos aos quais servem como suporte ao desenvolvimento de interfaces, melhorando a usabilidade do sistema. Na criação de interfaces também é recomendado o uso apropriado de estilos de interacção, para o utilizador não se sentir confuso na sua utilização. Analisou-se a metodologia de desenvolvimento em estrela, sendo esta iterativa, mais centrada a avaliação, com maior escolha de transição entre fases e realizando testes de usabilidade nessas transições.

Existem várias aplicações dedicadas ao controlo e planeamento da produção. Efectuou-se um estudo de várias, dando relevo às funcionalidades e à interface de cada uma, e tomando como ponto de partida as suas limitações no desenvolvimento deste trabalho de mestrado.

O desenvolvimento do módulo de edição de um *layout* fabril, em relação aos outros sistemas existentes, vem trazer vantagens, na medida em que existe uma maior personalização na criação de uma planta fabril.

A análise de usabilidade a este módulo foi elaborada no sentido de obter impressões, através da realização de um teste de eficiência e do preenchimento de um inquérito, no sentido de analisar melhorias, sugestões ou críticas a serem tidas em conta o refinamento deste módulo.

7.1 Objectivos realizados

Neste trabalho de mestrado pretendia-se o desenvolvimento de um editor de *layout* fabril, utilizando um processo de desenho similar à ferramenta Microsoft Paint. Ou seja, o utilizador recorrendo a *palette menus* teria que desenhar, através do conceito

drag and drop, elementos de desenho, tais como rectângulos, elipses, linhas e *labels*, e um elemento próprio de um *layout* fabril: máquinas. Para cada elemento de desenho seria necessário a sua personalização, alterando várias propriedades tais como o texto e o seu aspecto visual (cores e espessura de contorno, inserção de tipos de setas, entre outros). Neste módulo também se pretendia o armazenamento do *layout* num ficheiro em XML e a respectiva abertura do mesmo. Estas funcionalidades foram todas desenvolvidas com sucesso.

7.2 Limitações e trabalho futuro

O trabalho desenvolvido, apresenta alguns aspectos de usabilidade que poderão influenciar negativamente o utilizador. Grande parte dos aspectos foram detectados, ilustrados e analisados na Tabela 20 na secção 6.4 e serão tidos em conta como trabalho futuro. Sendo assim foi elaborada uma lista de funcionalidades a ser melhoradas ou adicionadas.

Na barra de ferramentas, seria vantajoso a adição de edição das propriedades dos elementos de desenho, de modo a satisfazer vários tipos de utilizadores. As funcionalidades de cópia de formatação e estilo poderão ser inseridas na medida em que facilita o utilizador na realização e na eficiência das tarefas. No *pop-up* menu dos elementos em que é possível a alteração das propriedades relativas ao estilo, a descrição do menu *Edit Shape* deverá ser alterada para *Edit Style* para não induzir em erro por parte do utilizador. O sistema também deverá permitir a alteração da orientação da linha de um modo rápido, sem necessidade de esforço extra por parte do utilizador (trocar a seta de início com a de fim manualmente).

Na criação de elementos de desenho, o utilizador pode, por descuido, definir o tamanho do elemento de desenho em causa bastante reduzido. Assim, dificulta o utilizador no controlo dos elementos criados (o elemento desenhado é tão pequeno que torna-se complicado a sua visualização). O sistema deve definir um tamanho mínimo desses elementos na eventualidade deste tipo de erros ocorrerem.

As caixas de diálogo de edição das propriedades dos elementos encontram-se no canto superior esquerdo, o que requer algum esforço extra por parte do utilizador. Estas devem-se encontrar centradas, com vista a colmatar este problema de usabilidade.

Os ícones de alteração das setas de início e de fim causam alguma confusão ao utilizador, pois focam-se mais no objecto circular que propriamente no tipo da seta. O tamanho deste objecto circular deve ser reduzido.

Os mecanismos de *undo* e *redo* não se encontram em total funcionamento. Não permite esses mecanismos quando o utilizador altera as propriedades de um elemento. Esta funcionalidade deve ser melhorada de modo a permitir o avanço e o recuo nas funcionalidades de edição de propriedades.

Os *tooltips* devem ser melhorados com vista a informar mais o utilizador. A caixa de atributos apresenta um defeito, pois em certos casos, quando se encontra vazia a borda não se encontra com uma cor vermelha.

Os ícones da janela de ferramentas devem ser alterados, removendo o fundo branco com vista a melhorar o aspecto do sistema.

O sistema também deve possibilitar a edição de vários objectos para um uso mais eficiente do sistema, criação de *layers* se possível e também a criação de linhas poligonais.

Estas melhorias irão certamente trazer benefícios ao nível da usabilidade do módulo de edição gráfico.

Referências Bibliográficas

Ana Madureira, Filipe Santos e Ivo Pereira, Self-Managing Agents for Dynamic Scheduling in Manufacturing, GECCO'2008 (Proc. of Genetic and Evolutionary Computation Conference 2008, Atlanta, Georgia (EUA), 12 a 16 de Julho de 2008.

Ana Madureira, Ivo Pereira, Nelson Sousa, Paulo Ávila, João Bastos, Scheduling a Cutting and Treatment Stainless Steel Sheet Line with Self-Management Capabilities, *Computational Intelligence for Engineering Systems, Intelligent Systems, Control and Automation: Science and Engineering, pp. 34-47, Springer Netherlands, ISBN: 978-94-007-0093-2, 2011*

Deborah Hix, Rex Hartson, *Developing User Interfaces.* Canadá, 1993

Jakob Nielsen, Usability Engineering, Morgan Kaufmann, Inc. San Francisco, 1993

Jide Common Layer Developer Guide, n.d. - http://www.jidesoft.com/products/JIDE_Common_Layer_Developer_Guide.pdf (acedido em 14.09.2012)

Pedro Campos e Nuno Nunes, Introdução ao Java Swing e AWT, n.d. http://cee.uma.pt/people/faculty/pedro.campos/docs/guia-IHM.pdf (acedido em 17.11.2011)

Helen Sharp, Yvonne Rogers, Jenny Preece, *Interaction Design: Beyond Human-Computer Interaction.* Wiley & Sons. 2011

Softi9, Izaro ERP Sistema Integrado de Gestão, n.d. http://www.softi9.pt/images/download/doc34.pdf (acedido em 09.09.2012)

Softi9-2, Izaro APS Escalonamento e Optimização da Produção, n.d. http://www.softi9.pt/images/download/doc34.pdf (acedido em 09.09.2012)

Wilbert O. Galitz, The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques, 3rd Edition, ISBN: 978-0-470-05342-3, 2007.

Webgrafia

URL1, n.d., *MES Sistrade* - http://www.sistrade.pt/pt/Solucoes/mis-erp-sistrade-scada-sfc.htm (acedido em 26.05.2012)

URL2, n.d., *Lekin*, - http://community.stern.nyu.edu/om/software/lekin/ (acedido em 26.05.2012)

URL3, n.d., *Gecad – News* - http://www.gecad.isep.ipp.pt/gecad/Pages/Presentation/AllNews.aspx?NewsNumber=104#ctl00_cph_page_content_l_104 (acedido em 13.09.2012)

URL4, 2012, *Softi9* - http://www.softi9.pt/products/details.php?id=APSpor (acedido em 29.05.2012)

URL5, n.d., *VarChart JGantt Overview* - http://www.netronic.com/gantt/gantt-charts.html (acedido em 17.11.2011)

URL6, n.d., Gantt Charts - http://www.netronic.com/gantt/gantt-charts/gantt-charts-software.html (acedido em 17.11.2011)

URL7, n.d., JFreeChart - http://www.jfree.org/jfreechart/ (acedido em 17.11.2011)

URL8, Vadim, 2007, Copy text from a MessageBox - http://vkreynin.wordpress.com/2007/10/02/copy-text-from-a-messagebox/ (acedido em 23.11.2011)

URL9, n.d., JFreeChart Samples - http://www.jfree.org/jfreechart/samples.html (acedido em 17.11.2011)

URL10, n.d., BeeSoft - JavaGantt overview - http://www.beesoft.eu/products/jgantt/overview.php (acedido em 17.11.2011)

URL11, n.d., BeeSoft - Purchase JavaGantt - http://www.beesoft.eu/products/jgantt/purchase.php (acedido em 17.11.2011)

URL12, n.d., Jide Products, - http://www.jidesoft.com/products/gantt.htm (acedido em 18.11.2011)

URL13, n.d., Jide Purchase - http://www.jidesoft.com/purchase/order.htm (acedido em 18.11.2011)

URL14, n.d., Abstract Window Toolkit (AWT), - http://docs.oracle.com/javase/6/docs/technotes/guides/awt/ (acedido em 17.11.2011)

URL15, Josh Fletcher, n.d., AWT vs Swing - http://edn.embarcadero.com/article/26970 (acedido em 17.11.2011)

URL16, 2012, Standard Widget Toolkit - http://pt.wikipedia.org/wiki/Standard_Widget_Toolkit (acedido em 17.11.2011)

URL17, josefx, 2011, Java Desktop application: SWT vs. Swing - http://stackoverflow.com/questions/2306190/java-desktop-application-swt-vs-swing (acedido em 17.11.2011)

URL18, n.d., Overview | Qt Jambi - http://old.qt-jambi.org/about/overview/ (acedido em 17.11.2011)

URL19, n.d., Binding Technology | Qt Jambi - http://old.qt-jambi.org/material/binding-technology/ (acedido em 17.11.2011)

URL20, Amit G Mendapara, n.d., Introduction - http://jwx.sourceforge.net/README.html (acedido em 17.11.2011)

URL21, Amit G Mendapara, n.d., wx - http://jwx.sourceforge.net/api/index.html (acedido em 17.11.2011)

URL22, Amit G Mendapara, n.d., Build Instructions - http://jwx.sourceforge.net/BUILD.html#5 (acedido em 17.11.2011)

URL23, n.d., Jide Software - http://www.jidesoft.com/ (acedido em 17.11.2011)

URL24, n.d., Jide Products - http://www.jidesoft.com/products/ (acedido em 01.12.2011)

URL25, n.d., Jide Order - http://www.jidesoft.com/purchase/order.htm (acedido em 17.11.2011)

URL26, n.d., Jide Software Purchase - http://www.jidesoft.com/purchase/ (acedido em 01.12.2011)

URL27, n.d., Buoy - http://buoy.sourceforge.net/ (acedido em 17.11.2011)

URL28, n.d., Balise Buoy GUI Builder - http://balise.sourceforge.net/ (acedido em 01.12.2011)

URL29, n.d., Java - http://www.java.com/en/download/faq/whatis_java.xml (acedido em 06.09.2012)

URL30, n.d., NetBeans IDE - http://netbeans.org/ (acedido em 07.09.2012)

URL31, n.d., moodlePT - http://moodlept.net.educom.pt/mod/book/view.php?id=408&chapterid=119 (acedido em 07.09.2012)

URL32, n.d., jGraph - http://www.jgraph.com/jgraph.html (acedido em 07.09.2012)

URL33, n.d. Principles of user interface design, n.d. - http://yeloo.wordpress.com/interactivity/principles-of-user-interface-design/ (acedido em 05.05.2012)

URL34, 2011, An Argument Against Pie Menus - http://elementaryos.org/journal/argument-against-pie-menus (acedido em 05.05.2012)

URL35, 2007, Show Input Dialog Box - http://www.roseindia.net/java/example/java/swing/ShowInputDialog.shtml (acedido em 05.05.2012)

Anexo 1 Guia de sessão de avaliação

Guião da sessão para a avaliação da usabilidade do protótipo de Edição de *Layout* Fabril.

Sessão para a avaliação da usabilidade

Questionário após a experimentação do protótipo ADSyS (QT1)

Primeiro teste de eficiência (BM1)

ADSyS – Edição de um *layout* fabril.

Sessão para a avaliação de usabilidade

Obrigado por ter aceitado participar nesta experiência. O seu objectivo principal consiste na avaliação das ideias subjacentes ao desenvolvimento do protótipo de edição de um *layout* fabril, um sistema vocacionado para a modelação visual de uma planta fabril, cuja utilização se baseia directamente no desenho assistido por computador.

O roteiro desta sessão está descrito na Tabela 1, indicando-se os tempos estimados para a duração de cada uma das tarefas previstas, num total esperado de cerca de 1 hora.

	Recepção (Descrição dos trabalhos)	5m
	Apresentação audiovisual e no posto de trabalho do protótipo de edição de um *layout* fabril., seguida de discussão	20m
	Realização de um exemplo de um *layout* fabril. (**BM1**)	30m
	Questionário sobre o protótipo de Edição de um *layout* fabril. (**QT1**)	5m

Tabela 1 – Roteiro da sessão

A sessão será conduzida pelo Eng.º Paulo Taveira, o qual estará ao vosso dispor para todo e qualquer esclarecimento.

O questionário apresenta uma escala de 1 (valor mínimo) a 5 (valor máximo).

Pretende-se averiguar a utilidade e a facilidade de utilização do sistema de edição do *layout* fabril, pelo que todos os comentários e sugestões serão bem-vindos. Durante a execução das tarefas "pense em voz alta".

Não há respostas certas ou erradas. Não se sinta inibido para apontar aspectos negativos ou positivos, para enunciar expectativas frustradas ou recompensadas.

Para finalizar gostaríamos de lhe agradecer o tempo e o esforço despendidos.

Questionário após a experimentação do protótipo de Edição de *layout* fabril (ADSyS)

Aspecto Visual

1. Adequação das cores utilizadas.	Péssimo |_|_|_|_|_| Excelente
2. Simplicidade do aspecto visual.	Péssimo |_|_|_|_|_| Excelente
3. Interpretação da barra de menus. 3.1. Clareza nas descrições utilizadas. 3.2. Ordem de apresentação das funcionalidades. 3.3. Avaliação global.	 Péssimo |_|_|_|_|_| Excelente Péssimo |_|_|_|_|_| Excelente Péssimo |_|_|_|_|_| Excelente
4. Interpretação da barra de ferramentas. 4.1. Adequação dos ícones. 4.2. Ordem de apresentação das funcionalidades. 4.3. Avaliação global.	 Péssimo |_|_|_|_|_| Excelente Péssimo |_|_|_|_|_| Excelente Péssimo |_|_|_|_|_| Excelente
5. Interpretação da janela de ferramentas. 5.1. Interpretação dos ícones de *undo* e *redo*. 5.2. Interpretação dos ícones de selecção e remoção da selecção. 5.3. Interpretação dos ícones de inserção de elementos de desenho 5.4. Interpretação do ícone de remoção de um elemento de desenho. 5.5. Avaliação global.	 Péssimo |_|_|_|_|_| Excelente Péssimo |_|_|_|_|_| Excelente Péssimo |_|_|_|_|_| Excelente Péssimo |_|_|_|_|_| Excelente Péssimo |_|_|_|_|_| Excelente
6. Interpretação da janela de atributos. 6.1. Interpretação da caixa de texto. 6.2. Interpretação da borda vermelha da caixa de texto.	 Péssimo |_|_|_|_|_| Excelente Péssimo |_|_|_|_|_| Excelente

6.3. Adequação dos estilos de interacção utilizados (botões, *ComboBoxes*, etc.) 6.4. Avaliação global.	Péssimo |_|_|_|_|_| Excelente Péssimo |_|_|_|_|_| Excelente
7. Interpretação da barra de estado. 7.1. Interpretação do semáforo. 7.2. Interpretação da borda vermelha e verde na caixa de texto.	 Péssimo |_|_|_|_|_| Excelente Péssimo |_|_|_|_|_| Excelente
8. Clareza da descrição dos *tooltips* dos botões.	Péssimo |_|_|_|_|_| Excelente
9. Aspecto global.	Péssimo |_|_|_|_|_| Excelente

Inserção de Elementos de Desenho

10. Utilização do *drag and drop* para inserção de elementos	Péssimo |_|_|_|_|_| Excelente
11. Facilidade de inserção de *labels*.	Péssimo |_|_|_|_|_| Excelente
12. Facilidade de inserção de linhas.	Péssimo |_|_|_|_|_| Excelente
13. Facilidade de inserção de máquinas.	Péssimo |_|_|_|_|_| Excelente
14. Facilidade de inserção de elipses.	Péssimo |_|_|_|_|_| Excelente
15. Facilidade de inserção de rectângulos.	Péssimo |_|_|_|_|_| Excelente
16. Utilização do comando de remoção de elementos de desenho.	Péssimo |_|_|_|_|_| Excelente
17. Avaliação global.	Péssimo |_|_|_|_|_| Excelente

Alteração das Propriedades dos Elementos de Desenho

18. Alteração das propriedades das *labels*. 18.1. Alteração da descrição das *labels*. 18.2. Utilização do botão secundário do rato para alteração. 18.3. Interpretação das opções do *pop-up* menu.	 Péssimo |_|_|_|_|_| Excelente Péssimo |_|_|_|_|_| Excelente Péssimo |_|_|_|_|_| Excelente

18.4. *Layout* da janela de alteração das propriedades de texto.	Péssimo |_|_|_|_|_| Excelente
18.5. Adequação dos estilos de interacção utilizados.	Péssimo |_|_|_|_|_| Excelente
18.6. Avaliação global.	Péssimo |_|_|_|_|_| Excelente
19. Alteração das propriedades das linhas.	
19.1. Utilização do botão secundário do rato para alteração do aspecto.	Péssimo |_|_|_|_|_| Excelente
19.2. Interpretação das opções do *pop-up* menu.	Péssimo |_|_|_|_|_| Excelente
19.3. *Layout* da janela de alteração das setas de início e de fim.	Péssimo |_|_|_|_|_| Excelente
19.4. Interpretação dos ícones de alteração das setas de início e de fim.	Péssimo |_|_|_|_|_| Excelente
19.5. *Layout* da janela de alteração do estilo.	Péssimo |_|_|_|_|_| Excelente
19.6. Adequação dos estilos de interacção utilizados.	Péssimo |_|_|_|_|_| Excelente
19.7. Facilidade de alteração da cor da linha.	Péssimo |_|_|_|_|_| Excelente
Avaliação global.	
20. Alteração das propriedades das máquinas.	
20.1. Alteração das respectivas descrições.	Péssimo |_|_|_|_|_| Excelente
20.2. Utilização do botão secundário do rato para alteração do aspecto.	Péssimo |_|_|_|_|_| Excelente
20.3. Interpretação das opções do *pop-up* menu.	Péssimo |_|_|_|_|_| Excelente
20.4. *Layout* da janela de alteração da fonte.	Péssimo |_|_|_|_|_| Excelente
20.5. Adequação das componentes utilizadas na janela de alteração da fonte.	Péssimo |_|_|_|_|_| Excelente
20.6. *Layout* da janela de alteração do estilo.	Péssimo |_|_|_|_|_| Excelente
20.7. Adequação dos estilos de interacção utilizados.	Péssimo |_|_|_|_|_| Excelente
20.8. Facilidade de alteração de propriedades da cor.	Péssimo |_|_|_|_|_| Excelente
20.9. Avaliação global.	Péssimo |_|_|_|_|_| Excelente

21. Avaliação global.	Péssimo |_|_|_|_|_| Excelente

Aspecto Geral

1. Eficiência.	Duração: _____
2. Aprendizagem.	Péssimo |_|_|_|_|_| Excelente
3. Memorabilidade.	Péssimo |_|_|_|_|_| Excelente
4. Erro.	Péssimo |_|_|_|_|_| Excelente
5. Satisfação.	Péssimo |_|_|_|_|_| Excelente
6. Críticas e sugestões ao sistema de edição de *layout* fabril.	

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Data __/__/____

Participante n.º __

Teste de eficiência (BM1)

Neste exercício pretende-se que o utilizador elabore um *layout* fabril, utilizando o sistema de edição de um *layout* fabril. O resultado do *layout* fabril desenhado pelo utilizador deve aproximar-se do representado na Figura 1. Não hesite em colocar questões ou em pedir ajuda sempre que entender necessário.

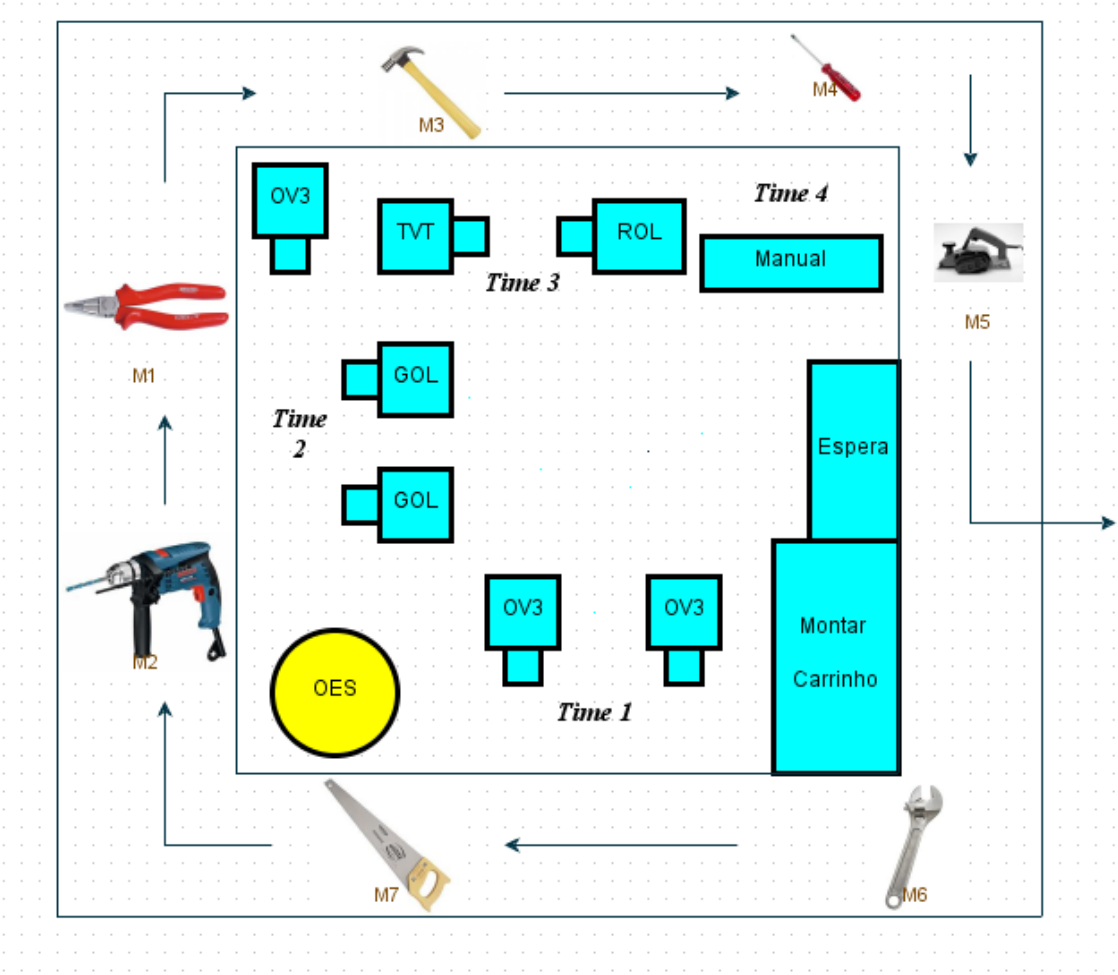

Figura 1 – Teste de eficiência

Observações:

As máquinas devem conter imagens de acordo com a Figura 1.

Formate os rectângulos e elipses de modo a que a espessura da linha de contorno seja 3. As inscrições "Time X" devem ser realizadas em itálico, negrito, tamanho 14 e com o tipo de letra *Times New Roman*. A cor de fundo das elipses devem ser amarela.

Mude a descrição do *layout* para “My layout 1”.

Quando entender que finalizou a realização deste teste, avise o guia para marcação do tempo despendido.