Sistemas Interativos

Exercício nº1 – Flexibility

22 de março de 2021

Grupo nº 3

Filipa Alves dos Santos (A83631)

Hugo André Coelho Cardoso (A85006)

João da Cunha e Costa (A84775)

Luís Miguel Arieira Ramos (A83930)

Válter Ferreira Picas Carvalho (A84464)





Mestrado Integrado em Engenharia Informática

Universidade do Minho

Índice de conteúdos

1. Introdução

No âmbito deste exercício para a unidade curricular de Sistemas Interativos, foi atribuído ao grupo um tema para explorar, relacionado com os princípios de design de interfaces: Flexibilidade (*dialogue initiative*; *multithreading*; task migrability).

Ao longo do presente relatório, é explicado cada um destes conceitos, procurando realçar a sua relevância no contexto de desenvolvimento de aplicações orientadas a utilizadores, e são apresentados exemplos de boas práticas associadas.

2. Desenvolvimento

2.1. Flexibility

Um dos principais princípios de usabilidade de um produto a ter em conta durante o seu desenvolvimento é a sua flexibilidade. Esta propriedade diz respeito ao leque de maneiras em que o utilizador e o sistema podem trocar informação e está intimamente ligado ao conceito de eficiência, que traduz a capacidade do utilizador de tirar o melhor proveito possível do produto. De seguida, serão abordadas algumas estratégias importantes a ter em conta relativas a estes conceitos.

Ao desenvolver um produto, é necessário considerar que todos os utilizadores aprendem de maneiras diferentes e têm as suas preferências, por isso um bom partido para tornar o programa flexível é permitir um certo nível de customização, de maneira a que os utilizadores possam organizar atalhos e definições à sua maneira, dentro dos possíveis, de forma a maximizar a sua eficiência.

Os atalhos referidos acima são outra funcionalidade importante: a existência de atalhos para operações usadas frequentemente torna mais flexível a navegação pelo sistema e aumenta a eficiência do utilizador. Semelhante a isto, deve existir também uma funcionalidade de pesquisa, permitindo ao cliente encontrar mais facilmente o que procura, tanto na *home page* como em outros contextos mais específicos da aplicação.

Por fim, outra característica importante a ter em conta é a flexibilidade das próprias ferramentas disponibilizadas pelo sistema. Por exemplo, no caso de um sistema de prescrições médicas, seria má prática fixar as unidades da quantidade da receita. Esta decisão dificilmente se adequaria a todas as receitas, acabando por dificultar o trabalho do utilizador – restringiria a sua liberdade e pioraria a sua eficiência, devido à falta de flexibilidade.

2.1.1. Dialogue Initiative

A *Dialogue Initiative* tem como objetivo ajudar e guiar o utilizador, de modo que este tenha sempre o controlo, mas com algumas restrições impostas pelo sistema na entrada de diálogo.

A usabilidade depende maioritariamente do fluxo de diálogo, ou seja, de como o utilizador interage com o sistema através de linguagem natural. O processo de design foca-se principalmente no fluxo de diálogo, onde é necessário compreender o utilizador, existir uma compreensão clara dos fatores humanos assim como outros fatores que possam interferir negativamente com a usabilidade do sistema.

De seguida, podemos verificar um exemplo de *Dialogue Initiative*. O utilizador fechou o documento Word sem o gravar, surge então um modal em que o utilizador é avisado sobre o ficheiro que não está gravado. São lhe dadas três possibilidades e cabe ao utilizador a capacidade de escolher qual delas pretende.

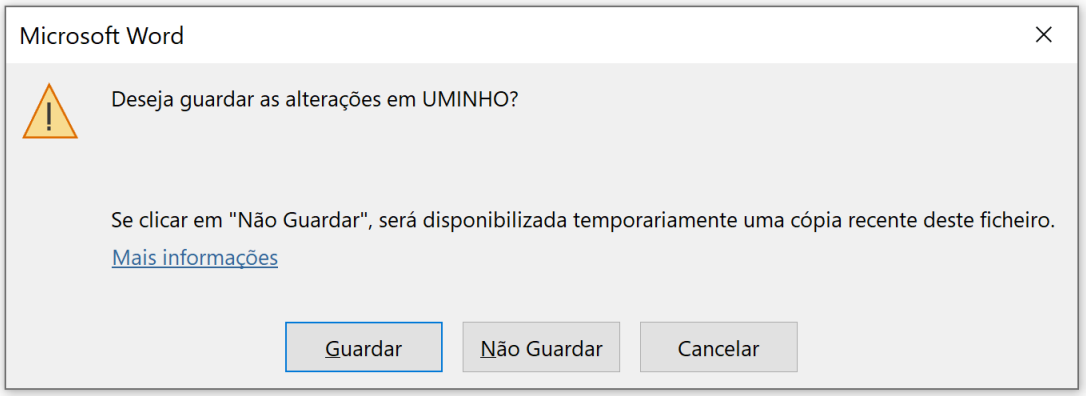

Figura 1 - Exemplo de Dialogue Initiative

Por outro lado, podem existir interfaces que não dão qualquer tipo de feedback ou realizam validação, pelo que todo o trabalho de investigação é atribuído ao utilizador. Uma situação muito comum é a existência de erros em comandos dentro de um terminal, tal como no exemplo seguinte:

Figura 2 - Exemplo em que Dialogue Initiative não é utilizada

Assim, esta interface não segue o princípio de *Dialogue Initiative*, uma vez que cabe ao utilizador realizar todo o processo de verificação e validação antes de executar o programa, assim como lembrar-se das diretorias de origem e de destino, causando um impacto negativo no ritmo de trabalho.

2.1.2. Multithreading

Multithreading é a habilidade do sistema suportar mais que uma tarefa em simultâneo. Existem dois tipos diferentes de *multithreading*, sendo um deles *Interleaved multithreading*, que permite overlap temporal de diferentes tarefas, mas que o diálogo com o utilizador seja restrito a uma só a cada instante. Já *Concurrent multithreading* permite comunicação simultânea de informação sobre diferentes tarefas.

Um exemplo disto é o *Windowing System*, que implementa ambos os tipos de *multithreading* explorados e é o software responsável por controlar as várias diferentes partes dos ecrãs do display. Este implementa *multithreading interleaved* no input, pois é possível termos várias janelas (tarefas) abertas ao mesmo tempo, mas só podemos interagir com cada uma .

Passando ao output, há um exemplo do segundo tipo de *multithreading*, através de *Multimodality with Fusion,* que permite os utilizadores interagirem com computadores através de diversas maneiras, com os seus diferentes sentidos (fala, audição, olhar, etc). Um caso específico seria quando um utilizador está a escrever num ficheiro e de repente ouve o alerta de um novo email: ambas as tarefas seriam percecionadas por ele como sendo simultâneas.

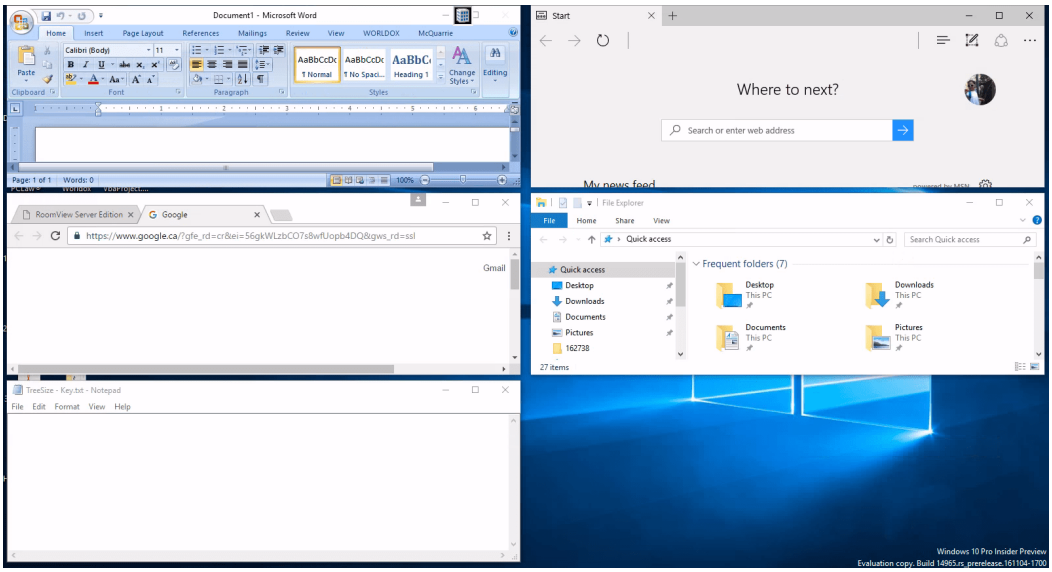

Figura 3 – Exemplo de Multithreading

No caso de um CRT terminal, só é possível executar uma tarefa de cada vez, pois depende do estado do programa. Logo, neste caso não há qualquer presença de *multithreading*.

Figura 4 - Exemplo que Multithreading não é utilizada

2.1.3. Task Migrability

Task Migrability significa alternar a responsabilidade de execução de tarefas entre o utilizador e o sistema.

Um exemplo disto são os programas de verificação ortográfica. Sistemas com esta funcionalidade oferecerem sugestões de correção de palavras para evitar a verificação manual por parte do utilizador. No entanto, é "perigoso" a execução destes programas sem a interação do utilizador, uma vez que o sistema não consegue prever as intenções do mesmo.

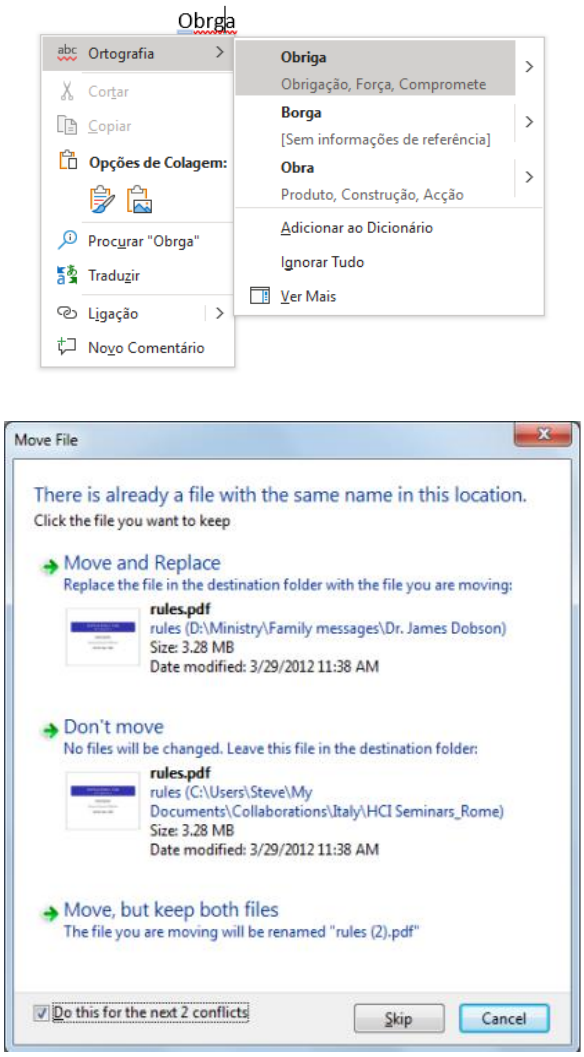

Figura 5 – Exemplos de Task Migrability

No entanto, existem interfaces que realizam operações que podem impactar negativamente o utilizador sem a sua confirmação. Um exemplo são as atualizações automáticas do Windows, que tipicamente são realizadas em plano de fundo automaticamente enquanto o utilizador pode estar a utilizar o computador, utilizando largura de banda que poderia ser necessária para uma outra atividade, tal como acesso a serviços de *streaming*, conferências, entre outros.

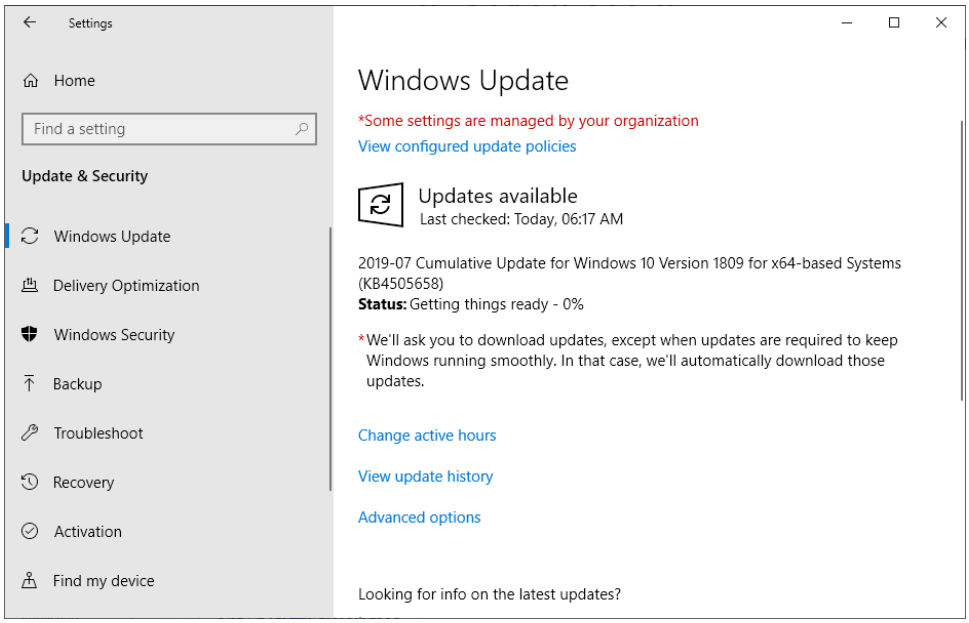

Figura 6 - Exemplo em que Task Migrability não é utilizada

2.2. Interface Design Patterns

2.2.1. Dialogue Initiative

Direct Manipulation

O objetivo deste *design pattern* é a representação contínua de objetos em que efeitos da modificação são imediatamente visíveis.

É uma das características mais importantes dos IDE's ou editores de texto mais sofisticados, por exemplo o Visual Studio Code, visto que permitem uma visão direta da árvore de diretorias e os respetivos ficheiros, permitindo que cada um ou mais sejam abertos de forma rápida uma vez que basta selecioná-los dentro do editor, ao contrário da edição com editores mais simplificados como o Notepad do Windows, que forçam o utilizador a abrir um ficheiro de cada vez e especificando a diretoria completa de cada vez que o faz.

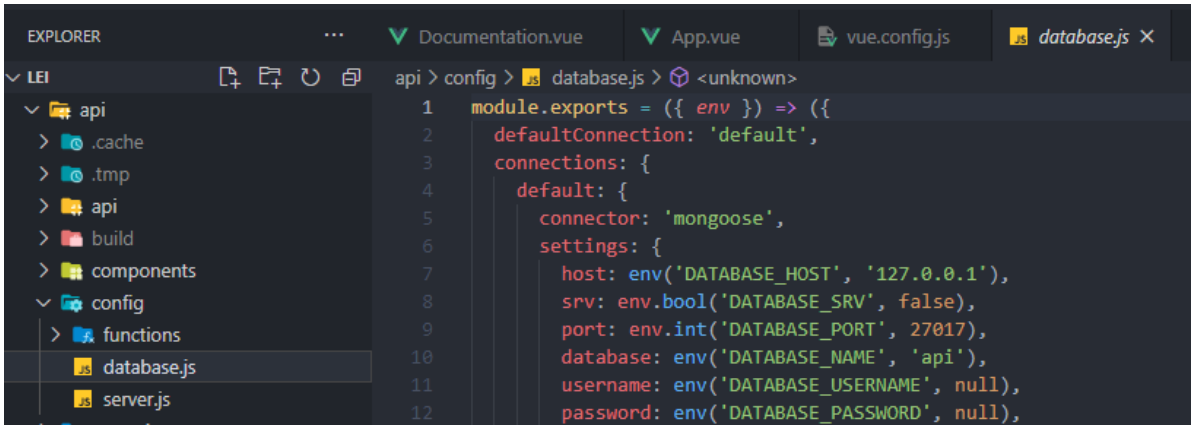

Figura 7 – Exemplo de Direct Manipulation

Assim, faz parte do "Dialogue Initiative" e, mais concretamente, é "*user-controlled*" uma vez que permite ao utilizador realizar a tarefa que pretenda, seja esta eliminar, editar, visualizar ou criar um conjunto de ficheiros ou diretorias como outras operações tais como compilações (caso o projeto seja relacionado com software), entre outros. Minimiza, deste modo, o controlo que o sistema impõe sobre o utilizador e aumenta a fluidez de ritmo de trabalho.

2.2.2. Multithreading

Dashboard

Este *design pattern* é utilizado para mostrar ao utilizador informação distinta simultaneamente através de mecanismos de representação gráficas, onde é disponibilizada informação sob a forma de estatística assim como dados analíticos.

Deste modo, o utilizador pode mais eficientemente analisar uma vasta quantidade de informação devido à sua representação gráfica, ao contrário de outras alternativas tais como a representação por

texto puro, que implicaria um maior investimento de tempo do utilizador na compreensão da informação.

Figura 8 – Exemplo de Dashboard

É um *design pattern* pertencente ao "Multithreading" uma vez que permite ao utilizador interagir com vários elementos simultaneamente, neste caso, estes seriam os vários dados representados visualmente. Deste modo, consegue fazer várias análises estatísticas/analíticas sobre dados distintos num local centralizado.

2.2.3. Task Migrability

Auto Complete

O propósito deste *design pattern* é dar ao utilizador ajuda no contexto de pesquisas tornando-as mais eficientes, uma vez que ele pode-se esquecer, escrever de forma incorreta ou até mesmo prever o que o utilizador pretende procurar.

Assim, o utilizador deve conseguir selecionar dentro de uma lista de resultados (finita) aquele que ele considera mais relevante para a sua pesquisa, obtendo assim uma maior velocidade e exatidão neste processo, uma vez que o utilizador não necessita de escrever (na maioria dos casos) a informação que necessita na totalidade, basta segmentos da mesma e deixar que o mecanismo lhe sugira o restante.

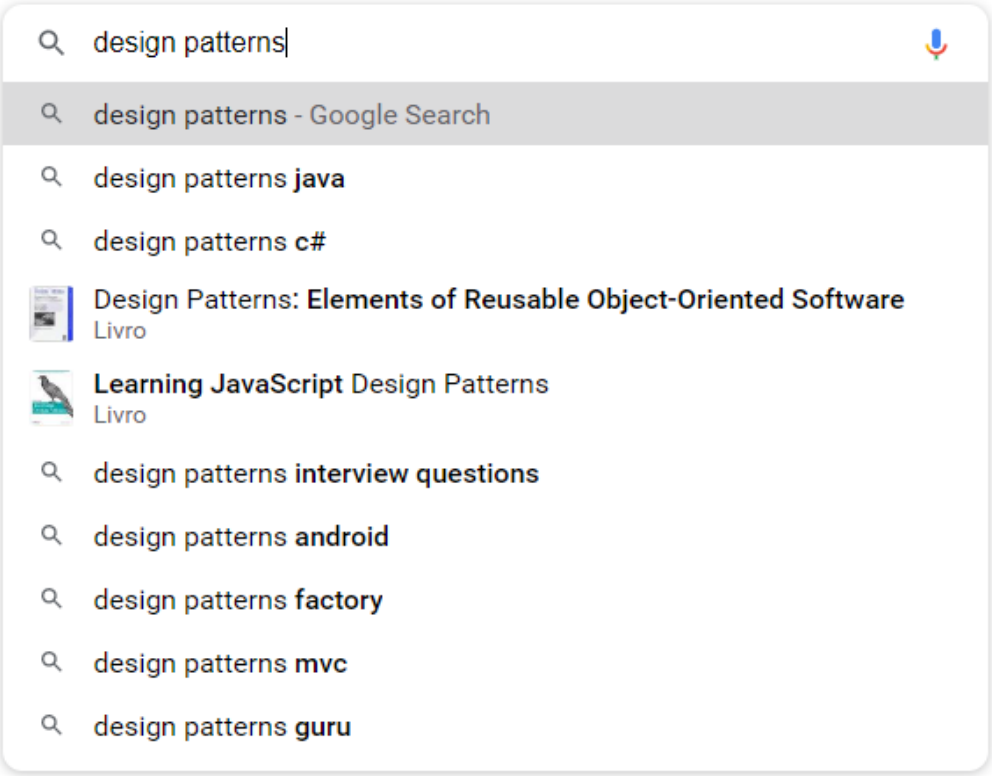

Figura 9 – Exemplo de Auto Complete

Este *design pattern* encaixa no *Task Migrability* uma vez que o sistema não toma decisões automáticas pelo utilizador, pelo contrário, dá o controlo da pesquisa ao utilizador, sugerindo apenas quais os termos que melhor encaixam na sua pesquisa, sendo o utilizador quem decide se os pretende utilizar e não o sistema.

3. Conclusão

Concluído este exercício prático, é de referir que a realização desta pesquisa permitiu ao grupo entender, de forma mais aprofundada, a utilidade e relevância dos fundamentos de design de interfaces, em concreto a flexibilidade.

O desenvolvimento deste trabalho decorreu como planeado, satisfazendo todos os requerimentos pedidos.