HCI (Human-Computer Interaction);

O que devemos saber/conhecer:

- Os utilizadores;
- As tarefas;
- O contexto de utilização.

Devem-se aplicar:

- *Design* interactivo centralizado no utilizador;
- Princípios de usabilidade;
- Técnicas de avaliação;
- Ou seja IPM envolve o design, implementação e avaliação de sistemas interactivos no contexto da tarefa do utilizador.

Qualidades do design de interacção:

- Usabilidade;
- Utilidade;
- Satisfação;
- Comunicação;
- Sociabilidade.

- Ter o utilizador em conta, testar, envolver o utilizador no processo de desenho/design, iterar/melhorar.

- Os sistemas devem ser robustos e consistentes:

- Devem estar preparados para um grupo de utilizadores alvo;
- Devem suportar uma utilização despreocupada;
- Devem ser úteis, ou seja, ajudar a completar tarefas em vez de criar obstáculos;
- O design da interface não deve ser deixado para ultimo plano, ou seja, deve ser desenvolvido a par do sistema.

-Aceitabilidade do sistema:

- Social;
- Prática:
    - Usabilidade:
        - Fácil de aprender;
        - Fácil de utilizar;
        - Fácil de lembrar;
        - Poucos erros;
        - Dê prazer.
    - Custo;
    - Compatibilidade.

- As interfaces são difíceis de desenhar:

- O utilizador tem sempre razão (se o utilizador tem problemas com a interface então a interface é que tem problemas);
- O utilizador nem sempre tem razão (o desenho da interface não pode derivar de apenas o que eles querem. Eles não sabem o que é melhor para eles.)
- Os utilizadores não são *designers*:
    - Não desenham uma interface do nada;
    - Reagem a designs que não gostam (podem ser relutantes);
    - Deve-se apresentar os designs num modelo que o utilizador perceba (prótotipo);
- Usabilidade – quão bem conseguem os utilizadores utilizar as funcionalidades do sistema;
- Atributos de usabilidade:
    - "Aprendibilidade" – quão rápido podem os utilizadores começar a ter uma interacção efectiva e conseguir máxima performance;
    - Eficiência – Assim que o utilizador tiver aprendido o sistema deve ser conseguida a máxima produtividade possível;
    - "Memorabilidade" – deve ser fácil de lembrar;
    - Erros – deve haver uma baixa taxa de erros;
    - Satisfação – deve dar prazer utilizar.
- Para medir a usabilidade:
    - Testar os utilizadores – seleccionar utilizadores que melhor representem o alvo (fazer tarefas especificas);
    - A usabilidade é medida relativamente a certos utilizadores em certas tarefas;
    - Determinar a usabilidade do sistema com b ase em médias de valores medidos.
- Usabilidade – Aprendizagem:
    - Fácil de aprender;
    - Sistemas de aprendizagem fácil permitem ao utilizador chegar a uma boa performance em pouco tempo;
    - Seleccionar um utilizador ao acaso e verificar quanto tempo leva a chegar a determinado sitio.
    - Medir a proficiência em número de utilizadores que conseguiram completar a tarefa e em que espaço de tempo.
- Usabilidade – experiência:
    - A eficiência é referente a um utilizar *expert* (quando a curva de aprendizagem estagna);
    - A experiência pode ser descrita pelo número de horas no sistema;
    - Medir continuamente a performance em determinada tarefa.
- Usabilidade – memorabilidade:
    - Ao contrário dos utilizadores casuais os utilizadores habituais não precisam de aprender a utilizar do início;
    - Os utilizadores habituais apenas precisam de aprender a utilizar o sistema baseados em aprendizagens anteriores;
    - Tarefas muito longas tendem a ser mais difíceis de momorizar;

    - Teste de memória – depois de terminar a tarefa pedir aos utilizadores qual seria o efeito de determinados comandos;
- Usabilidade – erros:
    - Os utilizadores devem cometer o menor número de erros possíveis enquanto utilizam o sistema;
    - Erro: acção que não leva ao objectivo;
    - A taxa de erros é medida no número de acções feitas pelo utilizador enquanto faz uma certa tarefa;
    - Alguns erros são corrigidos imediatamente pelo utilizador causando apenas uma perda de performance;
    - Erros catastróficos podem ser contabilizados separadamente dos erros menores.
- Usabilidade – satisfação:
    - Quão agradável é o sistema;
    - Medições psicofísicas (pupilas, batimentos, etc…);
    - Perguntar aos utilizadores a sua opinião subjectiva (média de várias opiniões);
    - O episódio mais difícil para o utilizador é o mais memorável;
    - Utilização de questionários;
    - Os utilizadores normalmente são positivos a não ser que a utilização seja muito má.
- Usabilidade – tradeoffs:
    - Não se deve dar sempre o mesmo peso a todos os aspectos da usabilidade;
    - Nem sempre é possível obter a pontuação máxima em todos os pontos da usabilidade (evitar erros catastróficos pode levar à perda de eficiência);
    - Tentar arranjar uma solução equilibrada definindo os atributos mais importantes com base na análise de tarefas dos utilizadores.
- Usabilidade – definição dos objectivos:
    - São fáceis de definir quando já existe competição no mercado;
    - Para sistemas sem competidor, os objectivos de usabilidade são muito difíceis de definir.
- Princípios de usabilidade:
    - Aprendizagem (fácil de aprender e chegar á máxima performance);
    - Flexibilidade (várias formas do utilizador interagir);
    - Robustez (nível de suporte para o utilizador completar as suas tarefas);
    - Outros:
        - Previsibilidade;
        - Sintetização;
        - Familiaridade;
        - Generalização;
        - Consistência;
        - Iniciativa de diálogo;
        - Multitarefa;
        - Migração de tarefas;
        - Substituição (vários inputs correctos);
        - Personalização;

        - Observável (estado do sistema);
        - Recuperação (em caso de erro);
        - Resposta do sistema;
        - Conformidade nas tarefas.

Slides 2

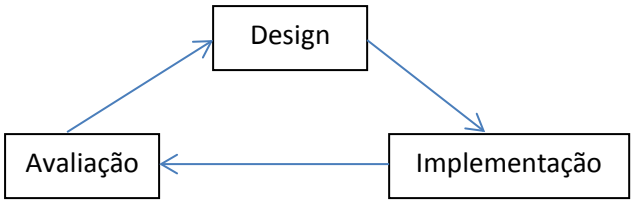
- Humano:
    - Tem capacidades limitadas de processar informação (sentidos);
    - A informação é armazenada, processada e aplicada:
        - Para resolver problemas;
        - Concluir;
        - Apreender conhecimento;
        - Erro;
    - Visão:
        - Profundidade e tamanho (perspectiva);
        - Claridade (luminância);
        - Cor (daltónicos);
        - Processamento visual> ambiguidade> solução: contexto;
        - Leitura (tamanho da letra 9 a 12, espaçamento).
    - Audição:
        - Fornece informação sobre o ambiente circundante;
        - Os humanos localizam a origem do som;
    - Toque:
        - Informação sobre o ambiente circundante;
    - Movimento:
        - Tempos de reacção:
            - Visual 200ms;
            - Auditivo 150 ms;
            - Dor 700ms.
        - Lei de Fitts – define o tempo que o utilizador leva a seleccionar o alvo no ecrã.
    - STM (short term memory);
    - LTM (long term memory):
        - Esquece aos poucos ao longo do tempo;
        - Relembrar e reconhecer;
    - Razão:
        - Dedutiva;
        - Indutiva;
        - Abdutiva.
    - Resolução de problemas:
        - Analogia;
        - Ganho de skills.

Slides 2

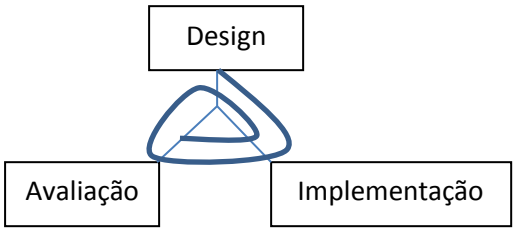
- O processo de design:
  - Técnicas científicas estruturadas;
  - O desenho do software envolve utilizadores e programador;
- Modelo cascata - pensar primeiro programar depois (ciclo de vida do software):
  1. Especificação dos requisitos;
  2. Desenho;
  3. Programação;
  4. Teste e integração;
  5. Manutenção;
  6. (com ou sem feedback, subida na cascata).
- 1º Especificação dos requisitos:
  - O designer e o cliente tentam chegar a um acordo no que deve ser implementado;
  - Envolve obter informação dos utilizadores;
  - Trabalhar aspectos do domínio:
    - Que funções deve o software realizar;
    - Informações sobre em que ambiente deve operar;
    - Que pessoas irá afectar;
    - Relações com outros produtos existentes;
  - Utilizar uma linguagem que utilizadores e programadores percebam.
- 2º Desenho:
  - Estabelecer como o sistema deve ser implementado de forma a seguir os requisitos;
  - Decomposição do sistema em componentes;
  - Requisitos: Funcionais (serviços do sistema) e não funcionais (como os serviços são fornecidos).
  - Refinar componentes da arquitectura de acordo com os requisitos não funcionais;
  - Objectivo: produzir uma descrição detalhada de cada componente e a sua integração.
- 3º Programação:
  - Implementação e teste de módulos individuais numa linguagem de programação.
- 4º Teste e integração:
  - Quando as componentes estiverem prontas são individualmente testadas e integradas;
  - Os testes de aceitação podem ser feitos com os clientes de forma a garantir que segue os requisitos;
  - Pode ser necessário certificar o sistema por uma autoridade externa.
- 5º Manutenção:
  - Depois de lançamento o sistema fica em estado de manutenção até que seja lançada a ultima versão;

- Detecção e correcção de erros;
- Revisão do sistema para satisfazer requisitos que não fora tido em conta;
- Validação:
  - Deve demonstrar que em todas as fase de vida do software (SLC) os requisitos do consumidor são satisfeitos;
  - No desenho de sistemas interactivos a validação de requisitos IPM á chamado de avaliação.
- Verificação: pode ocorrer entre a passagem de duas fases.

- O modelo em cascata não é apropriado para o desenho de interfaces em sistemas interactivos:
  - Os utilizadores só participam na especificação dos requisitos e na fase de teste;
  - A detecção tardia de erros pode levar a rectificações caras e demoradas;
  - Não suporta processos realmente interactivos.
- Desenho interactivo (cada iteração uma versão):
-



- Modelo espiral:



  - Várias iterações (custo e qualidade aumentam a cada iteração);
  - A primeira iteração pode ser feita em papel (baixo custo - protótipo);
  - Mais iterações (melhores interfaces);
- Design racional:
  - Informação que explica porque é que os sistemas computacionais são como são:
    - Informação arquitectural e estrutural;
    - Informação funcional e do comportamento;
  - Vantagens:
    - Comunicação entre todos os elementos da equipa de desenvolvimento durante o ciclo de vida;
    - Reutilização do conhecimento em situações idênticas;
    - Força a maiores cuidados nas opções de design;
    - Mostra o critério para resolver tradeoffs;
    - Organiza as diferentes possibilidades de desenho alternativas;

        - Permite capturar informação contextual.
- Desenho centrado no utilizador:
    - Necessidades;
    - Capacidades;
    - Contexto;
    - Trabalho;
    - Tarefas.
- Análise:
    - Utilizador (quem são os utilizadores?);
    - Tarefas (o que é que o utilizador precisa de fazer?);
    - Recolher dados do utilizador (características e necessidades):
        - Questionários, entrevistas, observação;
        - Obstáculos:
            - Isolar utilizadores da equipa de desenvolvimento;
            - Utilizadores difíceis;
            - Língua diferente.
    - Análise de tarefas:
        - O que é que os utilizadores fazem?;
        - Porque é que o fazem?;
        - Como é que o fazem?;
        - O que devem saber?;
        - Que ferramentas usam?;
        - A análise de tarefas envolve conhecimento:
            - Declarativo (objectos e relações);
            - Processual (sequencia de tarefas, objectivos e subobjectivos).
        - O método de análise de tarefas consiste em observar, juntar listas de frases e acções não estruturadas e organiza-las utilizando uma notação ou diagramas;
        - Objectivos, pré-condições e subtarefas;
        - Observação directa, procedimentos errados;
    - Entrevistas com os utilizadores:
        - Estruturadas;
        - Não estruturadas;
        - Semi-estruturadas.
    - Design participativo:
        - Os utilizadores tornam-se menos representativos quando eles percebem a estrutura do sistema proposto.
    - Logging (cliques do rato, localização, logs de transacção);
    - Estudar produtos concorrentes;
    - Análise do domínio:
        - Identificar coisas importantes no domínio (pessoas, objectos físicos, objectos de informação);
        - Determinar relações importantes entre coisas;
        - Identificar uma multiplicidade de relações;
        - Feedback ao utilizador e análise de tarefas.

- Descrição das tarefas:
  - Baseado em narrativas que descrevem:
    - Actores;
    - Objectivos;
    - Ferramentas;
    - Pensamentos/acções/eventos (sequência) para conseguir os objectivos.
  - Cenários:
    - Descrição informal narrativa;
    - Utiliza vocabulário do utilizador;
    - Referências repetitivas a um objecto ou comportamento podem sugerir a sua importância ou relevância no contexto;
    - Cenários a descrever a situação actual podem ajudar a definir novos cenários;
    - Fornecer casos de teste.
  - Para criar cenários:
    - Imaginar os conceitos fundamentais;
    - Fazer estudos de campo;
    - Estudar artefactos e dispositivos relacionados com as tarefas;
    - Questionários/entrevistas;
    - Sumarizar conclusões;
    - Construir um perfil do utilizador;
    - Escrever cenários.
- Análise das tarefas:
  - Hierarquia (HTA) – decomposição hierárquica de tarefas, gráfica ou textual;
  - Parar a decomposição quando não podem ser adicionados elementos relevantes;
  - Análise de tarefas baseada no conhecimento:
    - Consultar um expert;
    - Pedir aos utilizadores para ordenarem as tarefas (pode depender do objectivo);
  - Análise de tarefas para descrição do conhecimento (TAKD):
    - Utiliza uma forma especial de taxonomia chamada TDH (task descriptive hierarqy):
      - XOR os objectos devem pertencer a 1 e só 1 ramo;
      - AND os objectos devem pertencer a todos os ramos;
      - OR os objectos devem pertencer a pelo menos um ramo;
      - Não é possível distinguir os vários objectos.
  - KRG (Knoledge representaion grammar) – cada objecto é representado por um caminho único na hierarquia.
  - Modelos entidade relação;
  - Modelos workflow;
  - Diagramas de fluxo.

Slides 3

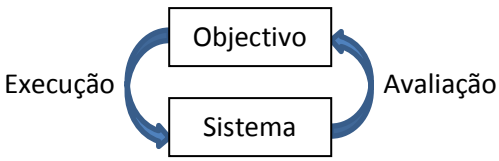
- Design iterativo e iterativo em espiral vs em cascata ou cascata com feedback (tem verificação);
- Protótipo = barato;
- Deve colocar-se o processo de design em frente ao desenvolvimento (processo de design explícito) o custo em fases avançadas pode aumentar devido ao fraco design;
- Tem de se conseguir o design certo e o design, certo.
- Esboçar (Sketching);
    - Rápido;
    - Barato;
    - Sem vocabulário;
    - Detalhe mínimo;
    - Pouca precisão;
    - Sugere e explora não confirma;
    - Ambíguo.
- Esboço no design de interacção:
    - Análogo ao tradicional;
    - Partilha todos os atributos chave;
    - Mais feel do que look;
    - Deve ser mais dinâmico.
- Sketching vs protótipo:
    - Sugere vs descreve;
    - Explora vs refina;
    - Pergunta vs responde;
    - Propõe vs testa;
    - Provoca vs resolve.
- Os esboços e protótipos projectam o produto que as pessoas querem, precisam e gostam;
- Para garantir algumas funcionalidades do design é necessário testá-los com utilizadores reais;
- Protótipo:
    - Mais rápido;
    - Mais barato;
    - Design em paralelo;
    - Mais simples;
    - Mais fácil de modificar, deitar fora;
    - Encoraja a reflexão no design;
    - Design centrado no utilizador (envolver o utilizador).
    - Fidelidade:
        - Baixa – omite detalhes usa fracos materiais;
        - Alta – Mais parecido ao produto final;
    - O uso de protótipos em papel (mockup) é mais rápido que protótipos computacionais;

    - Os protótipos podem ser de três tipos:
        - Deitar fora;
        - Incremental (adicionar novas funcionalidades ao anterior);
        - Evolucionário (tem como base o anterior).
- Análise do protótipo (equipa):
    - Computador;
    - Facilitador;
    - Observador.
- Protótipo em papel:
    - Interactivo, simula a janela e os seus elementos;
    - Interacção natural, apontar o dedo ou escrever;
    - O designer simula o efeito do computador;
    - Baixo look'n'feel, lei de Fitt;
    - Mais rápido do que no computador;
    - Fácil de alterar;
    - Foca-se no design global (sem detalhes);
    - Todos podem contribuir.
    - Limitado (cores, fontes);
    - Pouco dinâmico, pouco feedback.
- As pessoas são relutantes em criticar o design;
- Sketching: explorar o conceito, teste informal;
- Prototipo: teste de usabilidade;
- Design: conseguir o design certo (apropriado);
- Engenharia de usabilidade: conseguir o design certo (correcto);
- Protótipo computacional = protótipo em papel + layout do ecrã + cores fontes e icons + feedback interactivo + lei de Fitt (controlo do tamanho e distância entre controlos).

Slides 5

- Modelos conceptuais – as pessoas formam modelos conceptuais de como as coisa funcionam;
    - Fonte:
        - Causalidade;
        - Familiaridade com dispositivos similares;
        - Instruções;
        - Interacção.
- Um bom modelo conceptual permite-nos prever os efeitos das nossas acções e a relação entre os controlos e o resultado.
- Modelos:
    - Sistema – como o sistema funciona;
    - Interface – modelo do sistema apresentado ao utilizador;
    - Utilizador – modelo de como o utilizador pensa que o sistema funciona;
    - Design – modelo que o designer pretende para a interface;

- Tarefas do designer:
  - Escolher um modelo conceptual apropriado;
  - Comunicá-lo correctamente ao utilizador.
- Como comunicar o modelo ao utilizador:
  - Affordances (dar a ideia através das propriedades da coisa);
  - Mapeamento;
  - Visibilidade (as acções possíveis devem ser explicitas tal como o estado corrente e o resultado da acção);
  - Feedback (as acções devem ter um feedback imediato);
  - Restrições (restringir as possibilidades ao necessário).
- Metáforas – ajudam o utilizador a escolher o modelo conceptual (pedir emprestado um modelo conceptual).
  - Algumas tarefas não se ajustam á metáfora;
  - Diferenças culturais.
- Interacção – processo de transferência de informação;
- Os modelos de interacção ajudam a perceber a interacção e os problemas de raiz (funciona como framework para comparação de diferentes estilos de interacção).
  - Tradução entre o que o utilizador faz e o que quer que o sistema faça.
- Modelo de Norman:
  - Estabelecer objectivo;
  - Formar intensão;
  - Especificar a sequencia de acções;
  - Executar a acção;
  - Verificar o estado do sistema;
  - Interpretar o estado do sistema;
  - Avaliar o estado do sistema segundo as intenções e objectivos.



- Principios de bom design (modelo de Norman):
  - Visibilidade;
  - Feedback;
  - Bom mapeamento (metáforas);
  - Um bom modelo conceptual.
- Se um erro é possível alguém o vai cometer (modelo de Norman);
- Evitar erros:
  - Eliminar modos;
  - Modo visível;
  - Máxima consistência;
  - Suporte a recuperação de erros.
- Framework de interacção Abowd e Beale é uma extensão do modelo de Norman onde o sistema é incluído de forma explicita:
  - Execução: utilizador – articulação – input – performance – sistema;
  - Avaliação: sistema – apresentação – output – observação – utilizador;
- O input potencia uma fácil articulação.

Slides 6

- Regras de desenho:
    - Sugerem métodos para aumentar a usabilidade do software ou produto;
    - Podem ser caracterizadas por dois tipos:
        - Autoridade (indicação se é uma regra que tem de ser seguida ou se é apenas uma sugestão);
        - Generalidade (indicação se a regra pode ser aplicada a muitas situações de desenho ou apenas numa situação em particular);
    - Standards (+autoridade, -generalidade);
    - Guidelines (-autoridade, +generalidade);
    - Quanto mais geral for uma regra mais provável é de estar em conflito (tradeoffs entre regras);
- As 8 regras de ouro de Scheiderman:
    1. Lutar pela consistência;
    2. Permitir que utilizadores frequentes utilizem atalhos;
    3. Oferecer feedback informativo;
    4. Utilizar diálogos para conter erros;
    5. Oferecer prevenção e tratamento de erros;
    6. Permitir facilmente retroceder acções;
    7. Suportar controlo interno local;
    8. Reduzir a sobrecarga de memória a curto prazo.
- Os 7 princípios de Norman:
    1. Utilizar tanto o conhecimento do mundo como o da cabeça;
    2. Simplificar a estrutura das tarefas;
    3. Fazer as coisas visíveis (ligar avaliação á execução);
    4. Fazer mapeamentos correctos;
    5. Explorar o poder das restrições naturais e artificiais;
    6. Desenhar para o erro;
    7. Quando tudo o resto falha starndardizar.
- Reutilizar conhecimento sobre soluções de design com sucesso;
- Utilizar consistência nas cores;
- Evitar o uso de cores saturadas em pequenas coisas;
- A cor pode afectar a percepção temporal ou espacial dos objectos;
- A utilização de várias combinações de cores saturadas pode alterar a percepção das mesmas;
- A cor vermelha dá uma profundidade diferente da azul;
- Deve-se utilizar no máximo 6 cores;
- Devem-se utilizar cores segundo as convenções culturais;
- Utilizar consistência nas cores;
- Utilizar redundância no código das cores com outra dimensão gráfica;
- Não utilizar código de cores em pequenos elementos gráficos;

- Utilizar cinzento neutro onde é necessário o julgamento de cores;
- Passamos mais tempo a olhar para GUI do que a trabalhar na mesma;
- Guidelines:
    - Simplicidade;
    - Contraste;
    - Espaços brancos;
    - Balanço;
    - Alinhamento.
- Técnicas para a simplicidade:
    - Remover elementos não essenciais;
    - Regularidade (usar padrões regulares, limita as variações);
    - Deixar o mesmo elemento desempenhar várias funções.
- Os designs simples são reconhecidos e percebidos com o mínimo esforço consiente;
- Espaços brancos:
    - Separam;
    - Estruturam;
    - Enaltecem.

Slides 7

- Ícones bem desenhados:
    - Simples, ocupam pouco espaço no ecrã;
    - Facilmente e rapidamente reconhecidos;
    - Facilmente memorizáveis;
    - Ajudam a fazer interfaces internacionais.
- Princípios de desenho:
    - Coerência/consistência;
    - Legibilidade;
    - Reconhecimento, relembrar, familiar;
    - Pouca cor (desenhar a P/B depois adicionar cor).
- Desenvolver uma linguagem icónica;
- Teste de intuitividade:
    - Perguntar ao utilizador, o que é?, o que representa?.
- Teste de usabilidade:
    - Os ícones deverão ser posicionados a formar uma interface. Pedir ao utilizador para explicar o que vai fazendo.
- Diálogos:
    - Léxicos – ícones/formas, som;
    - Sintáctico – ordem e estrutura dos inputs/outputs;
    - Semânticos – resultado duma conversa nas estruturas internas de dados.
- Rede de transição de estados (STN):
    - Estados;
    - Transições;
    - Acções/respostas.

    - Diálogos concorrentes;
- Decomposição hierárquica:
    - Facilita a representação de redes complexas;
    - Ajuda a construir protótipos (estado = ecrã).
- Redes de Petri:
    - Places (estados);
    - Transicções;
    - Contadores (marca o estado corrente);
- Diagramas de estados;
- Diagramas de fluxos;
- Diagramas de design Jackson Estruturados;
- Notação textual;
- 3 Arquitecturas conceptuais do sistema de janelas:
    - Replicar gestão dos múltiplos processos dentro de cada aplicação (problemas de sincronização e reduzida portabilidade);
    - Implementação do papel de gestão dentro do kernel (centraliza a tarefa de gestão libertando-a de aplicações individuais);
    - Gestão por uma aplicação em separado (máxima portabilidade).
- Toolkits:
    - Oferecem um conjunto de objectos interactivos ou widgets com um comportamento por defeito;
    - Ajudam a sincronizar o input e output;
    - Promovem a consistência look'n'feel;
    - Limitados;
    - Difíceis de utilizar por não programadores;
- Os utilizadores interagem com a interface, as suas acções são comunicadas à aplicação;
- A aplicação deve responder.
- A separação entre a semântica da aplicação e a apresentação melhora:
    - Portabilidade;
    - Reusabilidade;
    - Interfaces múltiplas = flexibilidade;
    - Personalização;
- Modelo MVC:
    - Model – representa a semântica da aplicação, o estado da aplicação e o seu comportamento;
    - View – gere o output gráfico ou textual da aplicação (output);
    - Controller – controla e gere o input (interacção com o utilizador).
    - Separa o frontend do backend;
    - Permite múltiplas views para os mesmos dados da aplicação;
    - Permite que as views/controllers sejam reutilizados noutros modelos;
    - Cada par view/controller está associado a apenas 1 modelo.

Slides 7

- Modelos preditivos:
    - Avaliar como as pessoas interagem com as interfaces;
    - Mede a performance do utilizador sem testá-la;
    - Útil quando não é possível executar testes com o utilizador.
- GOMS (Objectivos, operadores, métodos, regras de selecção) - modela o comportamento do utilizador em termos de:
    - Objectivos – o que o utilizador quer conseguir (decomposição em sub-objectivos);
    - Operadores – acções cognitivas que devem ser executadas para conseguir o objectivo;
    - Métodos – procedimentos que descrevem como conseguir os objectivos;
    - Regras de selecção – determinam o método a utilizar quando existem vários possíveis.
- Estes modelos podem ser utilizados para:
    - Verificar funcionalidades (verificar que existe um método para garantir o termino de todos os objectivos);
    - Prever tempos de execução;
    - Sistemas de ajuda.
- Modelo Keystroke level = GOMS simplificado, faz previsões dos tempos de execução (soma de vários valores/operadores que constituem um método);
- CMN-GOMS – feito através do Keystroke level model adicionando sub-objectivos;
- NGOMSL – considera a estimativa do tempo de aprendizagem;
- CPM-GOMS – permite especificar tarefas em paralelo;
    - A sequência que produz o caminho mais longo chamasse caminho critico.
- GOMS (vantagens):
    - Ajuda a encontrar problemas de usabilidade;
    - Poupa tempo e recursos;
    - Fácil de construir um modelo simples;
    - Permite fazer previsões;
    - Permite descrever;
    - Serve de guia ao desenvolvimento.
- Avaliação:
    - Funcionalidades do sistema;
    - Impacto da interface no utilizador;
    - Identificação de problemas específicos do sistema.
- Heurísticas de usabilidade de Nielsen:
    1. Corresponde ao mundo real (usa uma linguagem comum, permite sinónimos ou metáforas);
    2. Consistência e standards (principio da surpresa mínima);
    3. Ajuda e documentação;
    4. Controlo e liberdade do utilizador;
    5. Visibilidade do estado do sistema;

6. Flexibilidade e fácil utilização (atalhos macros);
7. Prevenção de erros;
8. Reconhecimento ajuda no conhecimento;
9. Reportar erros, diagnóstico e recuperação;
10. Design simples e minimalista (linguagem concisa).

- Heurísticas de Nielsen:
  - 1,2,3 É o esperado;
  - 4,5,6 O utilizador manda;
  - 7,8,9 Gestão dos erros;
  - 10 Mantém simples;
- Podemos ir além das heurísticas de Nielsen para a avaliação:
  - Affordances, visibilidade, lei de Fitt, princípios das cores, etc..
- Poucos avaliadores detectam poucos problemas de usabilidade, muitos é demasiado dispendioso;
- Avaliação heurística:
  - 1º Treino (conhecer a equipa de design e os avaliadores);
  - 2º Avaliação (os avaliadores trabalham separadamente, identificam os problemas mas não a sua severidade);
  - 3º Classificação da severidade (frequência, impacto, persistência) (cosmético, menor, maior, catastrófico);
  - 4º Discussão de resultados (brainstorm = soluções).
  - Aplica-se a sketches ou protótipos;
  - É barata;
  - Rápida;
  - Identifica muitos problemas: maiores e menores;
  - Difícil de identificar elementos em falta num sketch;
  - Difícil encontrar problemas relacionados com o domínio do problema.
- Princípios de Tog:
  1. Antecipação;
  2. Autonomia;
  3. Cor para cegos;
  4. Consistência;
  5. Defaults;
  6. Eficiência do utilizador;
  7. Interfaces exploráveis;
  8. Lei de Fitt;
  9. Objectos Humano-Inteface;
  10. Redução da latência;
  11. Aprendizagem;
  12. Tradeoffs limitados;
  13. Metáforas;
  14. Proteger o trabalho do utilizador;
  15. Legibilidade;
  16. Verificação do estado;

17. Interfaces visíveis.

- Os utilizadores preferem aprender a explorar (caminho cognitivo);
- Avaliação formativa:
    - Identificar os problemas de usabilidade a serem corrigidos na próxima iteração;
    - Avaliar o protótipo ou a implementação num ambiente controlado;
    - Observações qualitativas;
    - Papeis:
        - Utilizador (elabora as tarefas, pensa alto);
        - Facilitador (descreve e distribui as tarefas);
        - Observador (não diz nada, tira notas sobre o comportamento do utilizador);
- Experiencias controladas:
    - Validade interna: o efeito produzido nas variáveis de output é causado por variações nas variáveis de input;
    - Validade externa: o efeito observado pode ser generalizado para o mundo real.
- Questionários:
    - Linguagem simples;
    - Escalares:
        - Avalia as características e atitudes do utilizador;
        - Saber o julgamento dos detalhes da interface pelo utilizador;
    - Escalas:
        - Nominais;
        - Ordinais;
        - Intervalares;
        - Rácios.
- Tipos de interacção:
    - Linha de comandos;
    - Menus;
    - Forms;
    - Folha de cálculo (tabelas);
    - Pergunta/resposta;
    - Língua natural;
    - WIMP (Windows, icons, menus, pointers);
    - Point e click;