MI/CEI

2003/2004

Interacção Humano-Computador

Engenharia da Usabilidade

José Creissac Campos

Departamento de Informática
Universidade do Minho
jose.campos@di.uminho.pt

Bibliografia, etc.

Livros

- Interactive System Design. Newman & Lamming (1995), Addison-Wesley.
- Human-Computer Interaction, third edition. Dix et al. (2004), Pearson/Prentice-Hall.
- Human-Computer Interaction. Preece et al. (1994), Addison-Wesley.

Web

- `www.usabilitynews.com` (British HCI Group)
- `www.usabilitynet.org` (projecto financiado pela EU)
- `www.upassoc.org` (Usability Professionals' Association)
- `www.usabilityfirst.com`

Plano

1. **Motivação**
   - Porquê pensar nestas coisas?
2. Breve história da IHC
3. Usabilidade
   - O que é? / Como se mede?
4. O Utilizador
   - Teorias do comportamento humano / Modelos mentais.
5. Desenvolvimento centrado no utilizador
   - Modelos, técnicas e ferramentas.
6. Avaliação
   - Métodos empíricos / métodos analíticos.

Motivação

Fraca usabilidade pode fazer falhar até a solução tecnologicamente mais avançada...

Evolução da utilização dos computadores

anos 60 Programadores

Software elaborado por programadores para programadores.

anos 70 Utilizadores profissionais

Computadores *mainframe*. Interfaces de "linha de comando".

anos 80 Grupos vastos de utilizadores indiferenciados

Boom dos PCs. Aparecimento das interfaces gráficas (*WIMP*).

anos 90 A internet!

Acesso a informação (demasiada?) para todos.

2000+ Computação ubíqua

O computador desaparece?

- Computadores cada vez mais pequenos.

  De salas enormes passamos para computadores *invisíveis*.

- Complexidade dos sistemas desenvolvidos aumenta.

  Quanto mais o hardware evolui, mas somos tentados a tentar *atacar* problemas cada vez mais complexos.

- Em particular, a complexidade das técnicas de interacção aumenta.

  Passamos de interruptores para reconhecimento de voz e de gestos.

- Software utilizado em situações cada vez menos controladas.

  O computador está em todo o lado. Imaginem o dia mundial sem computadores...

- Deixamos de fazer software para nós para fazer software para os outros!

  Aspecto muito importante: o que é que *os outros* querem do *nosso* software?

Ponto da situação

do processo de desenvolvimento

Estudos mostram que para grandes projectos (+50,000 linhas de código):

- 1/3 dos projectos é abandonado antes de estar terminado;
- em média, a duração dos projectos ultrapassa em 50% os prazos estimados;
- produtividade média está abaixo das 10 linha de código por dia;
- em média, encontram-se 60 erros por cada 10,000 linhas de código;
- custo de manter o software ultrapassa o dobro do custo de desenvolvimento.

do produto desenvolvido

Num estudo feito nos EUA (Eason, 88):

- 40% dos sistemas instalados tiveram ganho marginal;
- 40% falharam/foram rejeitados;
- apenas 20% dos sistemas instalados atingiram os benefícios pretendidos;

Exemplos

- Sonda Mariner I, Julho de 1962
- Therac-25, final dos anos 80.
- Airbus A320 (Mulhouse-Habsheim), Junho de 1988;
- Aeroporto Internacional de Denver, início dos anos 90;
- Ariane 5, Junho de 1996;

Sonda Mariner I, Julho de 1962

Deveria ter voado até Vénus. Apenas 4 minutos após o lançamento despenhou-se no mar. Descobriu-se depois que um operador de negação lógica tinha sido omitido *por acidente* no código do programa responsável por controlar os foguetes.

Therac-25, final dos anos 80

Máquina de Raios-X totalmente controlada por software. Diversos problemas (incluindo na interface com o utilizador) provocaram a administração de radiação excessiva a vários doentes.

Airbus A320 (Mulhouse-Habsheim), Junho de 1988;

Durante um voo de demonstração um Airbus A320 aproximou-se demasiado do chão e acabou por embater em árvores quando tentava ganhar altitude. Excesso de confiança por parte dos pilotos foi a causa apontada para o acidente. Aparentemente um sistema de protecção do motor impediu que este funcionasse a toda a potencia durante a tentativa de subida.

Aeroporto Internacional de Denver, início dos anos 90

Um projecto de grande complexidade para o sistema de tratamento de bagagens envolvendo mais de 300 computadores. O projecto excedeu os prazos de tal forma que obrigou ao adiamento da abertura do aeroporto (de Outubro de 1993 até Fevereiro de 1995). O sistema tinha tantos problemas que foi necessário mais 50% do orçamento inicial para por a funcionar.

Ariane 5, Junho de 1996

Explodiu no voo inaugural devido a uma série de falhas do software de navegação. Em circunstância específicas o software *abortava* devido a uma devisão por zero. O sistema de segurança implementado consistia em ter redundância: em caso de erro os dados eram processados por outro programa (que era uma cópia do primeiro! — abordagem adequada para hardware, mas não para software). O problema já existia no Ariane 4, mas nunca se tinha manifestado!

Desenvolvimento de software não pode ser encarado como *arte*,
mas como Engenharia.
Necessitamos de métodos e ferramentas apropriados.

Erro humano?

O factor humano é crucial:

- é consensual que 60-90% de todas as falhas são atribuíveis a erro humano [Hollnagel, 1993]
- 92% das fatalidades consideradas num estudo entre 1979 e 1992 podiam ser atribuidas a problemas na interacção humano-computador (4% a causas físicas e 3% a erros de software)[MacKenzie, 1994]

Mas...

Erro de qual humano?!

Do que utiliza o sistema, ou do que o desenhou/implementou?

[Leveson, 1995]

Therac-25

Acções do utilizador colocam aparelho em estado *ilegal*.

Airbus A320 (Mulhouse-Habsheim/Nagoya)

Interferência entre acções do utilizador e acções do aparelho (problema da automação).

Um gravador de vídeo da JVC

Certas funções não podem ser utilizadas enquanto se está a ler o manual de instruções — têm *timeouts* curtos o que impede que as instruções sejam lidas a tempo. (cf. primeiras versão do Público Online!)

Uma vida de cartões

Nunca foram para o carro sem o cartão do estacionamento?

Engenharia de software tipicamente preocupa-se com a construção dos sistemas.
É necessário prestar mais atenção à forma como estes vão ser utilizados.
Interacção-Humano Computador

Plano

1. Motivação
   - Porquê pensar nestas coisas?
2. **Breve história da IHC**
3. Usabilidade
   - O que é? / Como se mede?
4. O Utilizador
   - Teorias do comportamento humano / Modelos mentais.
5. Desenvolvimento centrado no utilizador
   - Modelos, técnicas e ferramentas.
6. Avaliação
   - Métodos empíricos / métodos analíticos.

Breve história da IHC

anos 60 Programadores

Software funciona essencialmente em *batch*. Não existiam grandes preocupações com a interface.[a]

anos 70 Utilizadores profissionais

Noção de que a *interface com o utilizador* é factor importante no sucesso de um sistema. Atenção centra-se no *desempenho* dos utilizadores.

anos 80 Grupos vastos de utilizadores indiferenciados

Noção de que desempenho não é tudo (treino, práticas de trabalho, ...). Necessidade de auto-aprendizagem. Surge o termo Interacção Humano-Computador (ciências da computação + psicologia cognitiva).

anos 90 A internet!

CSCW. Aspectos sociais e organizacionais.

2000+ Computação ubíqua — Novos desafios...

[a] Nos anos 40 tinha surgido interesse na Ergonomia, mas esta lidava com as características físicas das máquinas e o seu impacto no desempenho dos operadores.

Interface com o utilizador

- "Aqueles **aspectos de um sistema** com os quais o utilizador entra em contacto" (Moran 81) ("uma linguagem de entrada para o utilizador, uma linguagem de saída para a máquina e um protocolo de interacção").
- Surge o conceito de *user-friendly* — muitas vezes pouco mais que interfaces *bonitinhas*.

Interacção Humano-Computador

- "conjunto de processos, diálogos e acções através dos quais um utilizador humano emprega e **interage com um computador**" (Baecker and Buxton 87)
- "disciplina que se ocupa do **desenho, avaliação e implementação** de sistemas de computação interactivos para uso humano e com o estudo dos principais fenómenos que os rodeiam" (ACM SIGCHI 1992).
- "HCI concerne o desenho, implementação, e avaliação de sistemas interactivos no contexto das tarefas e trabalho do utilizador". (Dix et al. 2004)

Houve uma clara evolução de conceitos! Deixou de se pensar só na interface para se pensar na interacção entre utilizador(es) e sistema(s), e daí passou-se a todo o processo de desenvolvimento.

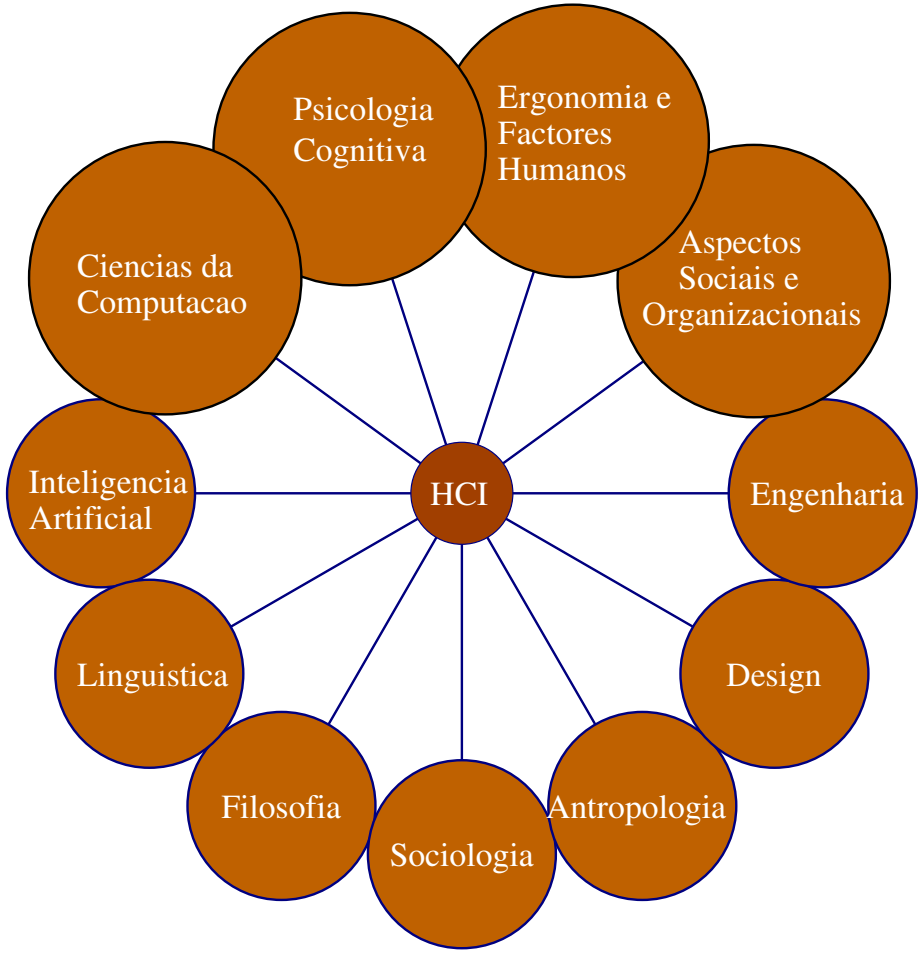
Standards

- ISO 9241-11: 1998 — *Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability*
- ISO 13407:1999 — *Human-centered design processes for interactive systems*
- ...

Processos de desenvolvimento

- Human-centered design (ISO 13407)
- User-centered design/User Engineering (IBM – www.ibm.com/easy/)
- Usage-centered design (Constantine & Lockwood Ltd. – www.foruse.com)

Uma área multidisciplinar



Objectivo é atingir boa *Usabilidade*

Usabilidade

O que é Usabilidade?

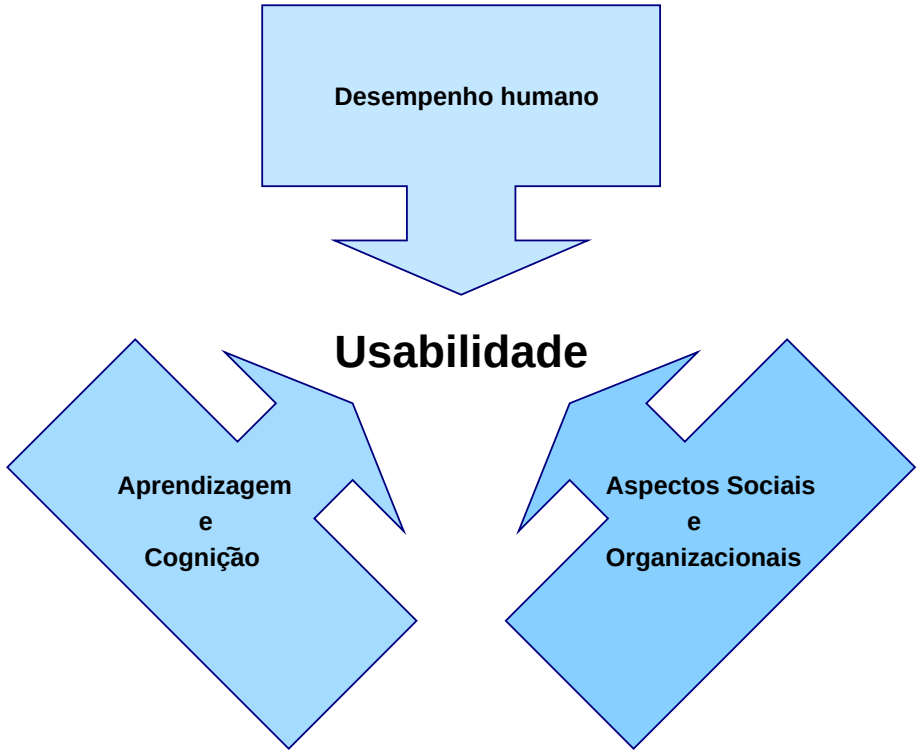
Usabilidade A eficácia, eficiência e satisfação com que utilizadores determinados atingem objectivos determinados em ambientes específicos (ISO DIS 9241-11).

- "eficácia": o utilizador consegue realizar as tarefas pretendidas;
- "eficiência": o custo de atingir os objectivos é aceitável (em termos de tempo, facilidade de aprendizagem, etc.)
- "satisfação": quão confortáveis se sentem os utilizadores com o sistema? (questão complexa).

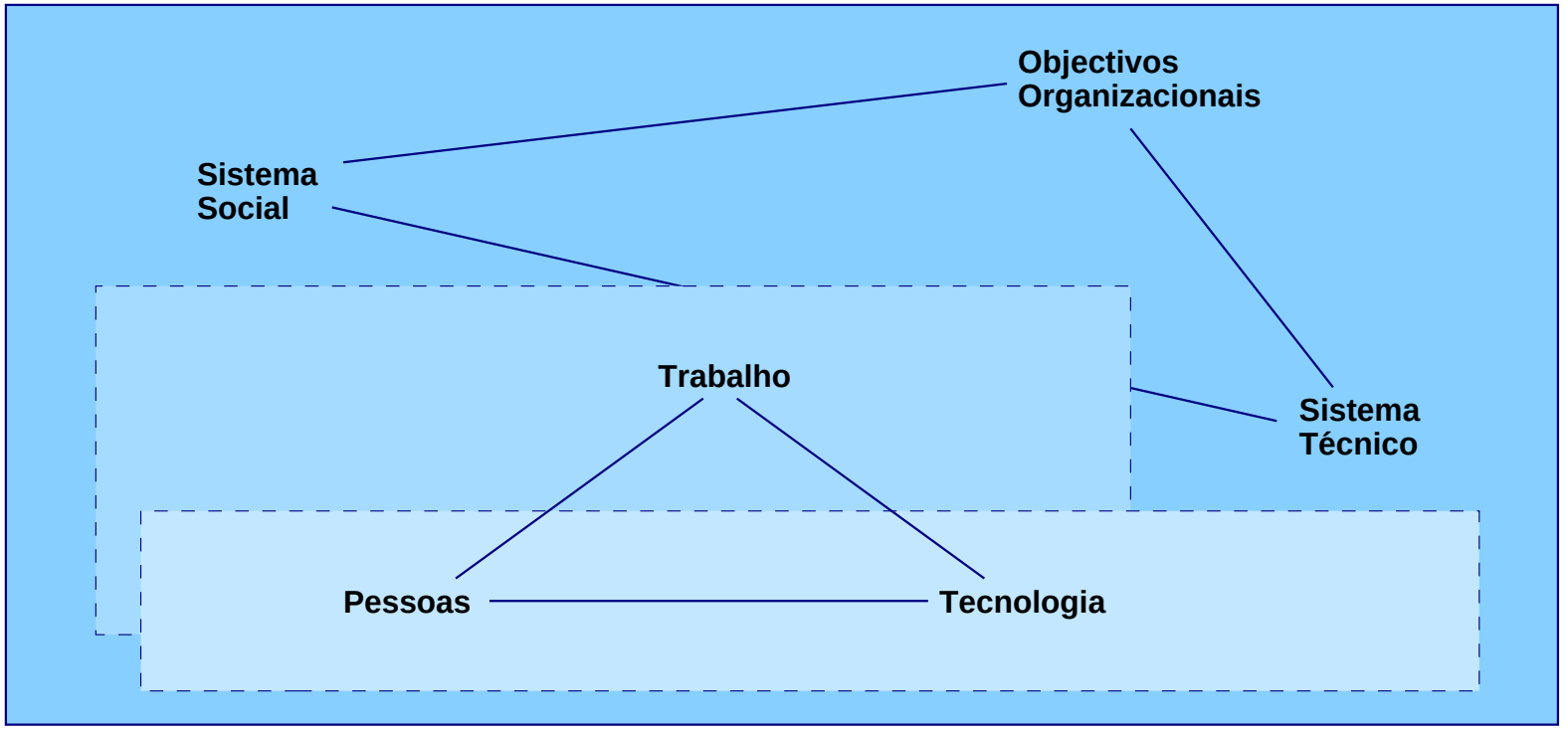
usabilidade $\neq$ *user friendliness*

- *user friendly*: fácil de usar (mas serve para alguma coisa?)

Três perspectivas complementares



Uma visão por níveis



Como atingir a Usabilidade?

Como medir a Usabilidade?

- Análise de usabilidade tradicional recorre a testes com utilizadores reais.
- Tipicamente realizada nos extremos do processo de desenvolvimento:
  - durante a análise de requisitos para analisar a concorrência;
  - na fase final de desenvolvimento para avaliar o produto desenvolvido.

Problemas

- Elevados custos do processo;
- Avaliação de qualidade feita demasiado tarde para permitir alterações do produto a baixo custo.

Possíveis objectivos de usabilidade

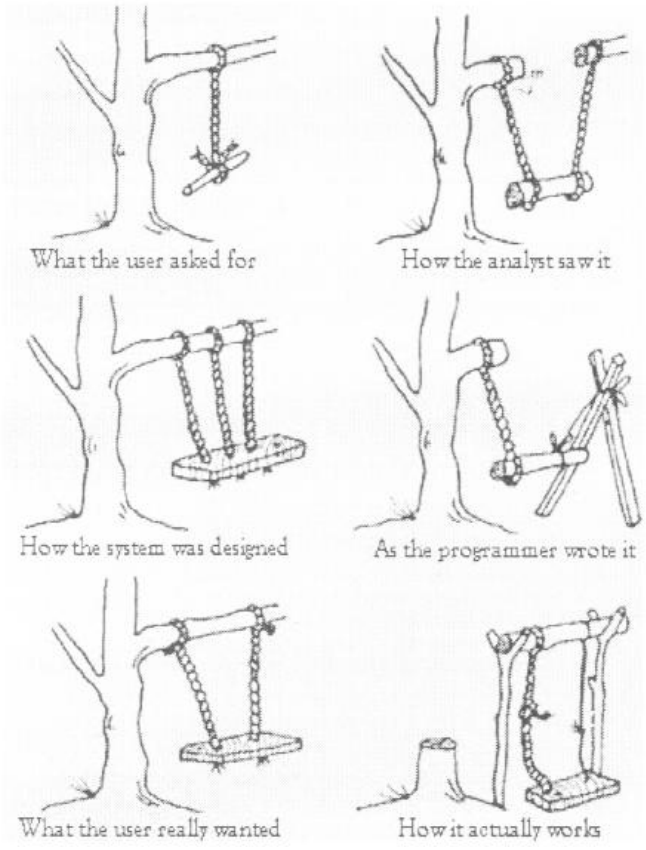
Possible measurement criteria (Tyldesley, 1988).

(1)	Time to complete task.
(2)	Percentage of task completed.
(3)	Percentage of task completed per unit time (speed metric).
(4)	Ratio of successes to failures.
(5)	Time spent on errors.
(6)	Percentage number of errors.
(7)	Percentage or number of competitors that do this better than current product.
(8)	Number of commands used.
(9)	Frequency of help or documentation use.
(10)	Time spent using help or documentation.
(11)	Percentage of favourable:unfavourable user comments.
(12)	Number of repetitions of failed commands.
(13)	Number of runs of successes and of failures.
(14)	Number of times the interface misleads the user.
(15)	Number of good and bad features recalled by users.
(16)	Number of available commands not invoked.
(17)	Number of regressive behaviours.
(18)	Number of users preferring your system.
(19)	Number of times users need to work around a problem.
(20)	Number of times the user is disrupted from a work task.
(21)	Number of times the user loses control of the system.
(22)	Number of times the user expresses frustration or satisfaction.

Engenharia da Usabilidade

Prática mostra que boa usabilidade é difícil de atingir. A Engenharia da Usabilidade procura ser uma resposta a este problema.

- A Engenharia da Usabilidade procura garantir que o produto é adequado ao fim para o qual foi desenvolvido.
- Engenharia da Usabilidade: conceitos e técnicas para definir, atingir e verificar objectivos de usabilidade dos sistemas.



Engenharia da Usabilidade

> "... um processo através do qual a usabilidade de um produto é especificada quantitativamente e *à priori*. Então, conforme o produto ou protótipos do produto são desenvolvidos, é demonstrado que eles atingem os níveis planeados de usabilidade." [Faulkner, 2000] citando [Tyldesdly, 1990]

É necessário definir

- Objectivos quantitativos de usabilidade

  Ver página 24.

- Técnicas para atingir esses objectivos

  Desenho centrado no utilizador.

- Técnicas para verificar se os objectivos foram atingidos

  Avaliação de sistemas interactivos.

É necessário identificar

- Quem são os utilizadores — A usabilidade deve ser definida em relação a um tipo específico de utilizador. Utilizadores experientes têm necessidades diferentes de utilizadores noviços.
- Que actividades pretendem/devem realizar — A usabilidade deve ser definida para tarefas específicas que o sistema deve suportar. No entanto, o sistema acabará muitas vezes por ser utilizado de formas não previstas inicialmente.
- Em que ambiente — O tipo de ambiente em que o sistema vai ser utilizado pode influenciar não só a usabilidade do sistema, mas a forma como os testes podem decorrer.

(cf. definição na página 20 / modelo na página 22)

Em resumo...

Para um produto ter sucesso deverá ser:

- **Útil**

  Realizar o que é pretendido (cf. eficácia).

- **Utilizável**

  Fazê-lo de forma fácil e natural (cf. eficiência).

- **Utilizado**

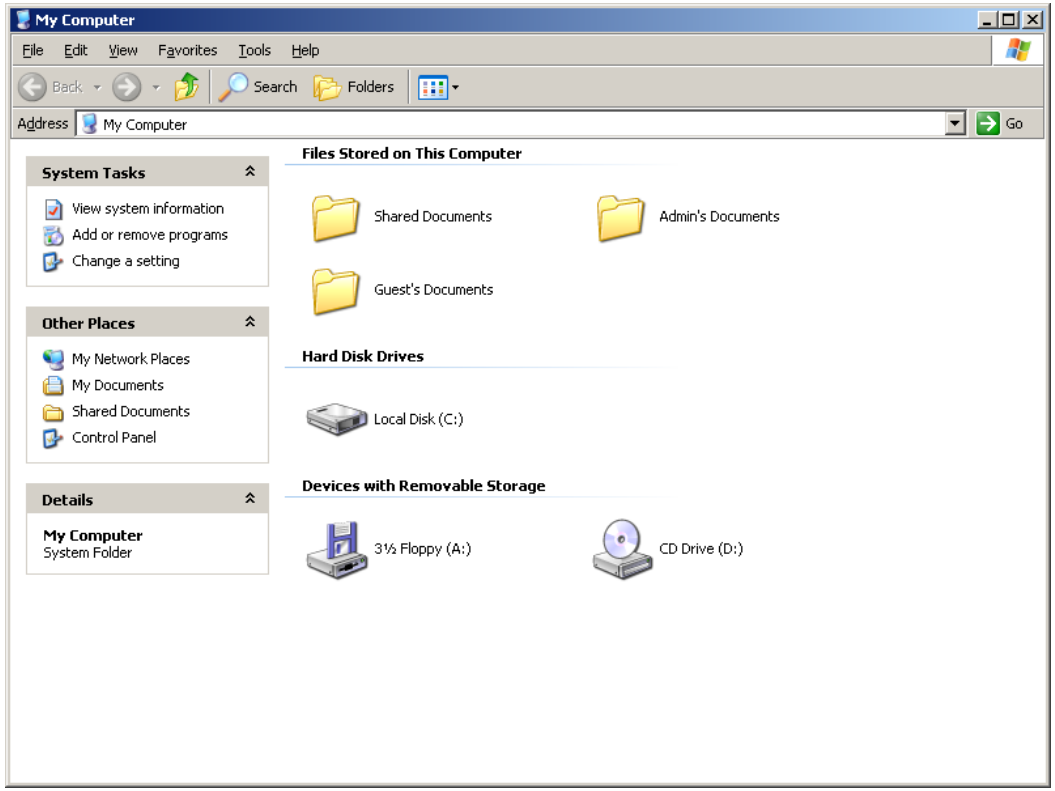
  Fazer com que as pessoas o queiram utilizar (cf. satisfação).

(comparar com definição de usabilidade na página 20)

A pensar...

Durante a semana procure identificar situações/sistemas em que surgiram problemas de usabilidade.

Exemplo



Plano

1. Motivação
   - Porquê pensar nestas coisas?
2. Breve história da IHC
3. Usabilidade
   - O que é? / Como se mede?
4. **O Utilizador**
   - Teorias do comportamento humano / Modelos mentais.
5. Desenvolvimento centrado no utilizador
   - Modelos, técnicas e ferramentas.
6. Avaliação
   - Métodos empíricos / métodos analíticos.

O Utilizador

Usabilidade: $\underbrace{\text{Utilizador} + \text{Tecnologia}}_{\textbf{Sistema Interactivo}} + \text{Trabalho} + \text{Contexto}$ (ver pag. 22)

Como medir usabilidade? / Como avaliar os sistemas?

Como projectar a usabilidade?

Temos tendência para avaliar a interface segundo os nossos próprios padrões.
Serão os mesmos do utilizador final?

Medir usabilidade / Avaliar sistemas

Com utilizadores

Nem sempre os utilizadores estão disponíveis.

Implica termos acesso ao sistema ou a um protótipo adequado.

Sem utilizadores?

- Com base em propriedades genéricas — como definimos essas propriedades?
  - com base em experiência adquirida
  - com base em teorias do comportamento humano

  Propriedades representam conhecimento sobre os utilizadores
- Com base em modelos dos utilizadores — "simula-se" o utilizador

Quer propriedades quer modelos são representações mais ou menos explícitas de conhecimento sobre como os utilizador fazem uso da tecnologia.

Que conhecimento possuimos sobre os utilizadores?

Voltaremos ao tema da avaliação mais tarde...

Como compreender o Utilizador?

É necessário:

- Compreender como é que o utilizador *interpreta* a interface?
- Compreender como é que o utilizador *se comporta* perante a interface?

Necessitamos de teorias do comportamento humano.

Tarefa da psicologia cognitiva!

Teorias do comportamento humano

Que tipos de teorias? ([Newman and Lamming, 1995])

- Teorias explicativas

  Procuram explicar o **comportamento** humano **observado**.

- Leis empíricas

  Permitem fazer **previsões** qualitativas **de desempenho**.

- Modelos dinâmicos de comportamento

  Procuram definir qual o **comportamento previsível**.

Teorias explicativas

- Procuram explicar o comportamento humano, não têm poder preditivo.
- Podem ser úteis:
  - nas fases iniciais de recolha de requisitos e análise — para explicar o comportamento observado;
  - nas fases finais de avaliação do sistema desenvolvido – para explicar diferenças entre o resultado final e o inicialmente previsto.

Exemplo

Impacto de um sistema de informação médica no contacto entre o médico e o paciente.

Leis empíricas

- Predições qualitativas de desempenho humano;
- Derivadas de observações/estudos do comportamento humano.

Exemplos

- Lei de Hick

  $T = k \log_2(n+1)$

  $T \rightarrow$ tempo necessário para escolher entre n alternativas ($k \approx 150$msec)

- Lei de Fitt

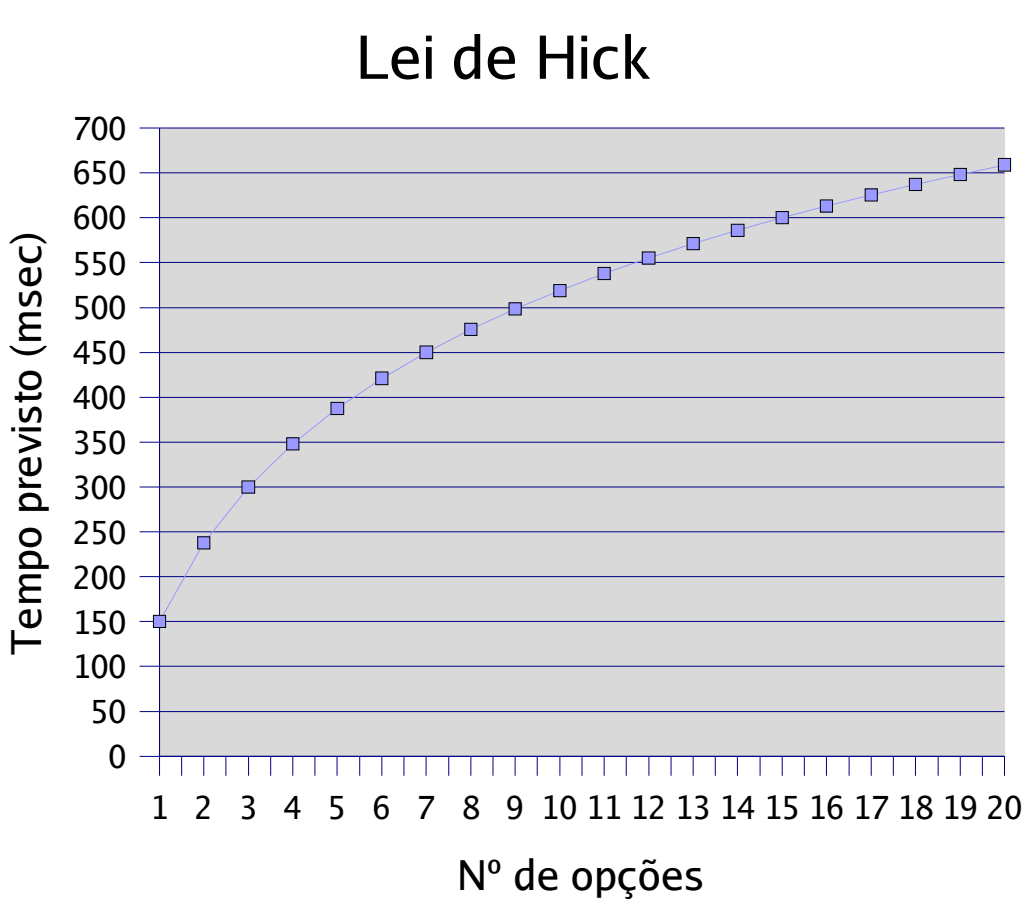
  $T_{\text{POS}} = k \log_2 \left(\frac{D}{W} + .5\right) \qquad (k \approx 100msec)$

  $T_{\text{POS}} \rightarrow$ tempo necessário para atingir um alvo de largura W partindo da distância D.

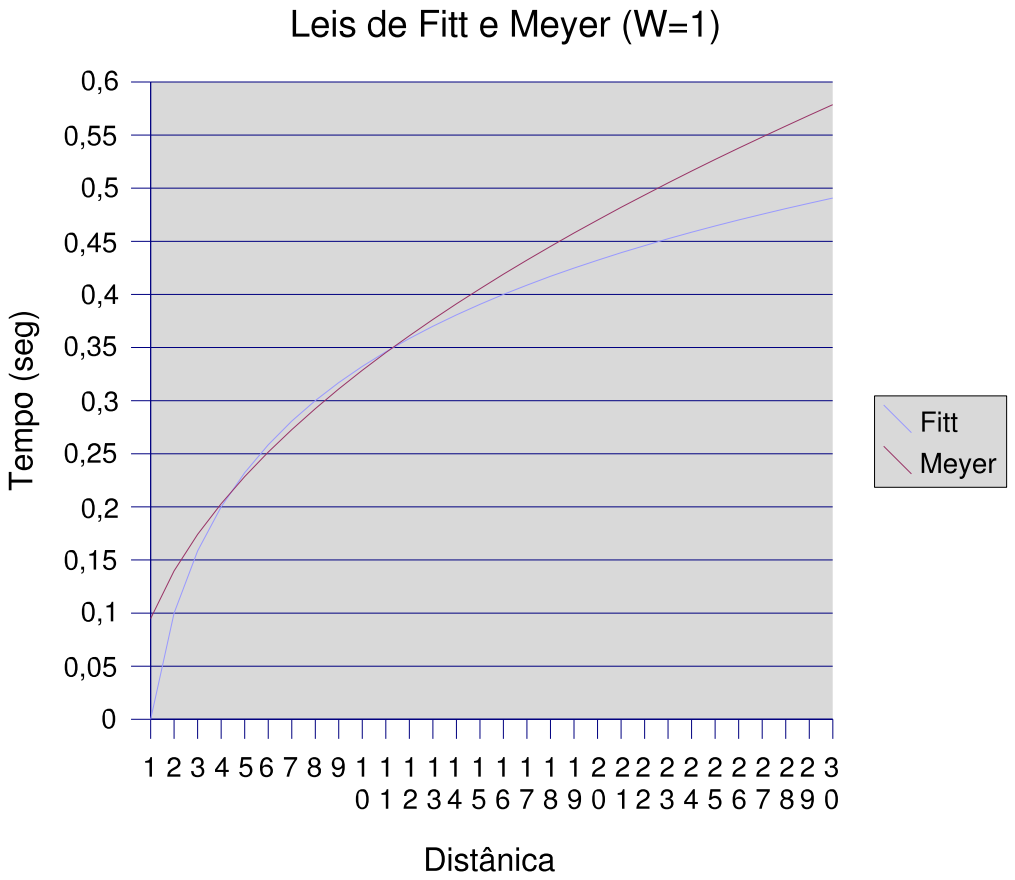
- Lei de Meyer

  $T_{\text{POS}} = A + B \times \sqrt{\frac{D}{W}} \qquad (A \approx -13msec,\ B \approx 108msec)$

  Um refinamento da Lei de Fitt para movimentos rápidos.



Como só podem ser formuladas para situações muito concretas, as leis empíricas aplicam-se normalmente a acções isoladas.


Leis de Fitt e Meyer (W=1)
Tempo (seg)
0,6
0,55
0,5
0,45
0,4
0,35
0,3
0,25
0,2
0,15
0,1
0,05
0
1 2 3 4 5 6 7 8 9 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
Distânica
Fitt
Meyer

Modelos dinâmicos

- Permitem analisar sequências de acções.
- Procuram responder às seguintes questões:
  - Que acções serão executadas?
  - Qual será o desempenho a executar determinadas acções?

Exemplo

- *Keystroke-Level Analysis*
  - Permite analisar desempenho na realização de uma tarefa conhecida.
  - Aplicável quando o método de operação é bem conhecido.
  - Útil para comparar desempenho previsto de métodos de operação alternativos.
  - Cada operação é dividida em componentes e a cada tipo de componente está associado um tempo de execução.
  - Utilizam-se heurísticas para introduzir componentes de "preparação mental" (pausas).

KLA — Keystroke-Level Analysis

Operadores:

K	premir tecla do teclado (de .08 a 1.10)
B	premir botão do rato (premir/largar .10 / clique .20)
P	apontar com o rato (ver Lei de Fitt — média de 1.1)
H	mover a mão (.40)
M	preparação mental (1.35)
R(t)	resposta do sistema

Exemplos:

- Fechar uma aplicação em Windows (short cut vs. menu)
- ...

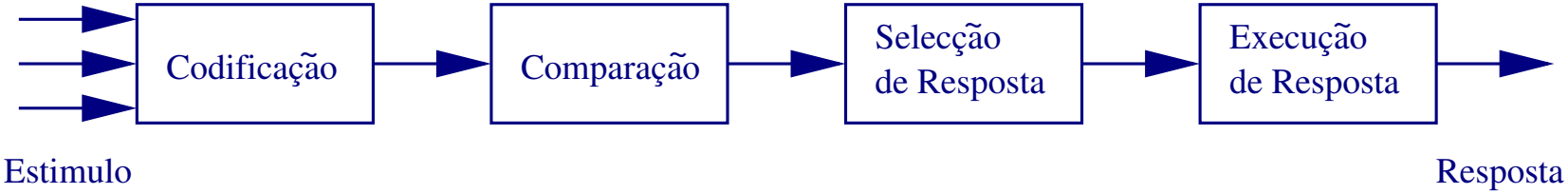
Modelos do Utilizador / Arquitecturas Cognitivas

Humanos como Processadores de Informação

As teorias anteriores focam aspectos específicos do comportamento humano. Torna-se necessário ter um modelo integrador que permita uma análise global.

Durante os anos 60 e 70 o principal paradigma na área da psicologia cognitiva foi foi o de caracterizar humanos como processadores de informação: tudo o que é recebido pelos orgãos sensoriais era considerado como informação que o cérebro processa.

Os estágios de processamento:



Duas extensões ao modelo :

- Atenção
  - Temos a capacidade para focar a nossa atenção nos estímulos que nos interessam, ignorando outros estímulos menos interessantes.
  - Quando diversos estímulos competem pela nossa atenção podemos ser distraídos por estímulos menos relevantes.
  - Claramente, o modo como usamos a nossa atenção perante uma interface tem grande impacto na eficácia com que a podemos utilizar (voltermos a este tema...)
- Memória:
  - memória sensorial;
  - memória de curto prazo;
  - memória de longo prazo.

  A interface não deve sobrecarregar a memória do utilizador (*Gone for a cup of tea syndrome*).

Memória Sensorial

- Um *buffer* para os estímulos recebidos:
  - Estímulos visuais — memória iconica (.5 sec)
  - Estímulos auditivos/aurais — *echoic memory*
  - Estímulos de "toque" — memória haptica
- Duração muito limitada;

Memória de Curto Prazo

- memória utilizada para armazenar infromação que é necessária para executar "cálculos".
- Tempo de acesso $\approx$ 70ms / degradação $\approx$ 200ms.
- Capacidade de armazenamento limitada: 7 +/- 2 unidades de informação.

  Isto não significa que os menus devam ter 7 +/- 2 entradas!...

Memória de Longo Prazo

- Memória duradoura:
  - informação factual;
  - conhecimentos experimentados;
  - regras de comportamento;
  - tudo o que *'sabemos'*.
- Capacidade de armazenamento enorme (ilimitada?)
- Acesso lento e pouca degradação (sem degradação?)
- Dois tipos de memória de longo prazo:
  - episódica — eventos e experiencias guardados sequencialmente.
  - semântica (cf. redes semânticas) — repositório estruturado de factos, conceitos e competências.

    Derivada da memória episódica.

GOMS

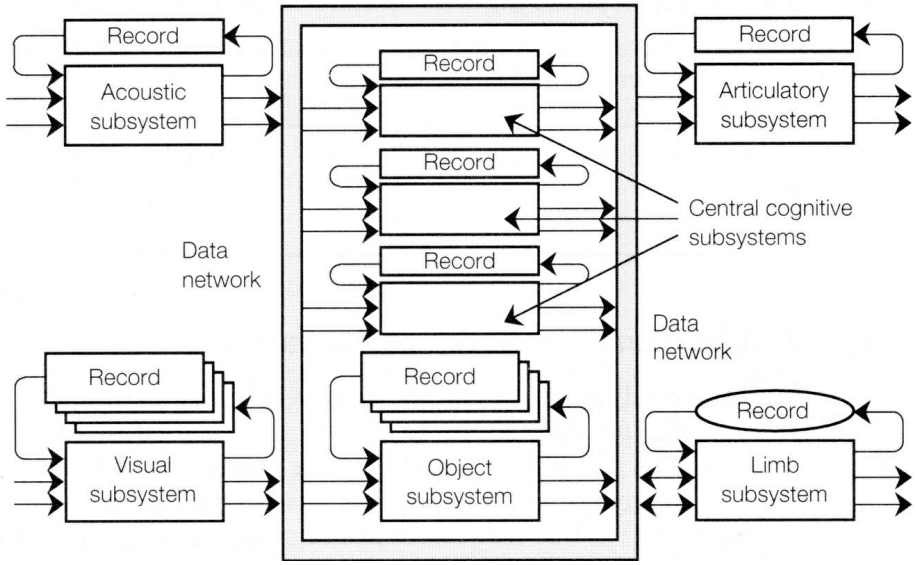
- Um dos primeiros modelos do utilizador a utilizar este tipo de abordagem foi o Modelo do Processador Humano (*Model Human Processor*).
- Este deu origem a uma família de modelos denominada GOMS (*Goals, Operations, Methods, and Selection rules*)
  - Permitem conjecturar sobre qual o método que será utilizado para atingir um objectivo
  - Permitem prever desempenho na realização de tarefas pré-definidas (através de *Keystroke-Level Analysis*)
  - Permitem analisar a complexidade de uma interface

Procurar uma linha

```
Goal: Locate Line
.     Choose Command                          ...(if not at task line)
.     [select Goal: use Line-feed Method
.             .     Goal: specify Command
.             Goal: use Quote-String Method
.             .     Goal: specify Command
.             .     Goal: specify Argument
.             Goal: use Scroll Method
.             .     Goal: specify Command
.             .     Goal: specify Argument]   ...(repeat until at task
                                                              line)
```

Interacting Cognitive Subsystems

- Procura definir uma arquitectura para o sistema cognitivo humano.



- É instanciado para cada caso concreto.
- Demasiado complexo para ser prático em termos de aplicação directa.
- Tem sido utilizado essencialmente para expressar conhecimento da psicologia cognitiva de forma mais explícita e rigorosa.

PUM — Programmable User Modelling

- Define uma arquitectura cognitiva que é *programada* com o conhecimento que o utilizador é suposto ter.
- Esta arquitectura é encarada como um mecanismo de solução de problemas através de *means-ends analysis*.

Etapas do método:

1. Identificar tarefas
2. Identificar operações conceptuais
3. Descrever o conhecimento do utilizador na *Instruction Language*
4. Analisar o modelo manual e automaticamente (voltaremos a isto...)

1. Identificar tarefas

Identificar as tarefas que o utilizador pretenderá realizar, a um nível apropriado de abstracção (nem demasiado próximas do nível físico - premir um botão - nem demasiado abrangentes - editar um texto). Exemplos: copiar um pedaço de texto, fazer *undo*, navegar na web utilizando o mecanismo de história.

2. Identificar operações conceptuais

Estas são as operações que vão permitir realizar as tarefas. Devem ser definidas em relação ao que o utilizador pretende fazer e não em relação à interface (o que quer fazer e não como pode fazer).

3. Descrever o conhecimento do utilizador na Instruction Language

Para isso é necessário identificar os *objectos* que o utilizador conhece e quais as relações entre eles. Um aspecto a considerar é como o utilizador obtém o conhecimento: através do écran, por inferência, memória, etc.

Um exemplo...

Considere um editor de texto e a tarefa de substituir pedaços de texto com outro texto presente no editor. O editor possui comandos para fazer *copy*, *cut* e *paste*.


```
OBJECTS
  text: tx-1, tx-2, tx-3.
  location: a, b, c, d.
RELATIONS
  cursor-at(location)
  buffer-contains(text)
  text-at(text, location)
  is-marked(text, location)
OPERATIONS
operation locate-cursor (location: L)
    user-purpose:        cursor-at(L)
    subgoaling-precond: locate(L)
    action:              locate-at(L)
operation mark-text-at-loc (text: TX, location: L)
    user-purpose:        is-marked(TX, L)
    subgoaling-precond: cursor-at(L)
    filtering-precond:  text-at(TX, L)
    action:              mark-text(TX)
operation cut-text-from-loc (text: TX, location: L)
    user-purpose:        not text-at(TX, L)
    subgoaling-precond: is-marked(TX,L)
    action:              cut-text
```


```
operation copy-text-to-buffer (text: TX, location: L)
    user-purpose:        buffer-contains(TX)
    subgoaling-precond:  is-marked(TX,L)
    predicted-effect:    buffer-constains(TX)
    action:              copy-to-buffer
operation oaste-text-at-loc (text: TX, location: L)
    user-purpose:        text-at(TX,L)
    subgoaling-precond:  buffer-constains(TX)
                         cursor-at(L)
    action:              paste-text
```

- Autores afirmam que a maior virtude da abordagem está na discussão gerada durante a construção dos modelos;
- Nem sempre se atinge um modelo totalmente definido dado o *custo* que isso acarreta.

Modelos exploratórios

- Nos modelos anteriores está implícito que os utilizadores sabem como atingir objectivos e que utilizarão a interface de forma *procedimental*.
- Com o aparecimento das interfaces gráficas a utilização do software passou a ser efectuada de uma forma mais *oportunista*.
- Os utilizadores tendem a explorar a interface:
  1. Definem objectivos a atingir
  2. Exploram a interface em busca de acções apropriadas
  3. Seleccionam as acções que aparentam ser apropriadas
  4. Analisam o resultado obtido (decidem se o objectivo foi atingido - se consideram que não foi, regressam a 2)

Modelos Mentais do Utilizador

- Modelos anteriores tentam explicar como o utilizador "funciona".
- Em todos eles, assume-se que o utilizador constrói uma "imagem" mental do artefacto — **Modelo Mental**.
- Modelo mental pode ser obtido a partir de:
  - experiência própria;
  - observação de outros utilizadores;
  - documentação.
- A noção de modelo mental é útil para guiar o desenvolvimento — que imagem pretendo que o utilizador tenha do artefacto?
- Dois tipos base:
  - Estáticos — a imagem que o utilizador possui sobre o estado do sistema;
  - Dinâmicos — modelos causais utilizados para prever consequências das acções.

Modelos estáticos/estruturais

- Procuram capturar a *imagem mental* que o utilizador possui/constrói sobre o estado/estrutura interna do sistema.
- Não permitem fazer previsões sobre como o utilizador vai utilizar o sistema pois lidam com o "como o sistema se organiza internamente" e não com o "como o sistema é utilizado".
- No entanto, se um utilizador possuir um modelo estrutural correcto, consegue prever o efeito das acções e, assim, descobrir como utilizar o sistema para realizar tarefas.
- Normalmente um utilizador não conseguirá derivar um modelo estrutural completo do sistema, mas necessita pelo menos de um modelo parcial para poder compreender o sistema.

Modelos dinâmicos/funcionais

Procuram capturar o conhecimento que o utilizador possui/constrói sobre como utilizar o sistema

Tipos comuns de modelos

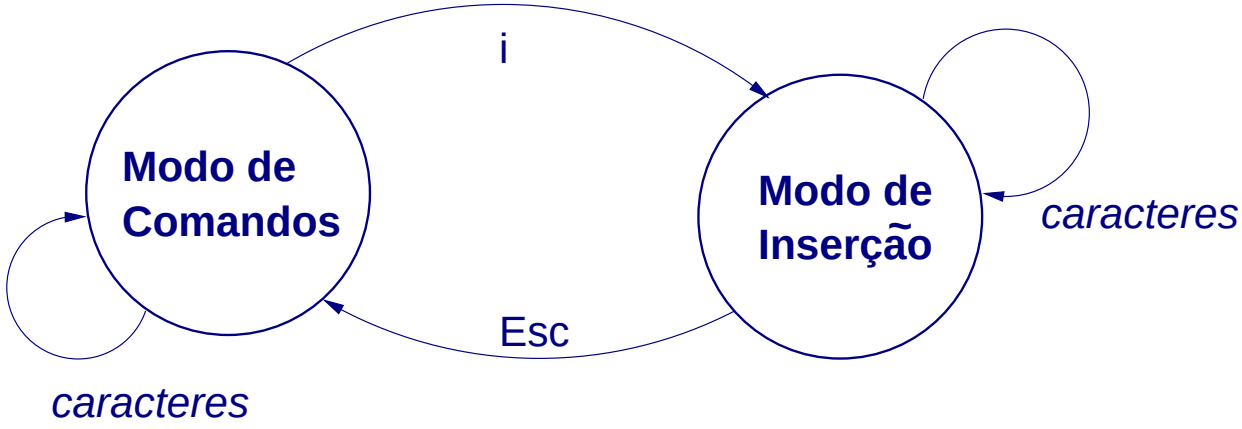
- Modelos Objecto-acção
- Máquinas de Estado
- Modelos de Mapeamento

Modelos Objecto-acção

- Surgem quando o utilizador toma consciência da existência de objectos na interface e das acções que neles pode realizar.
  - Caso típico: Interfaces por manipulação directa.

Máquinas de Estado

- Surgem quando o utilizador constrói um modelo mental baseado na observação de mudanças de estado do sistema.
  - Caso típico: Interfaces com *modos*.

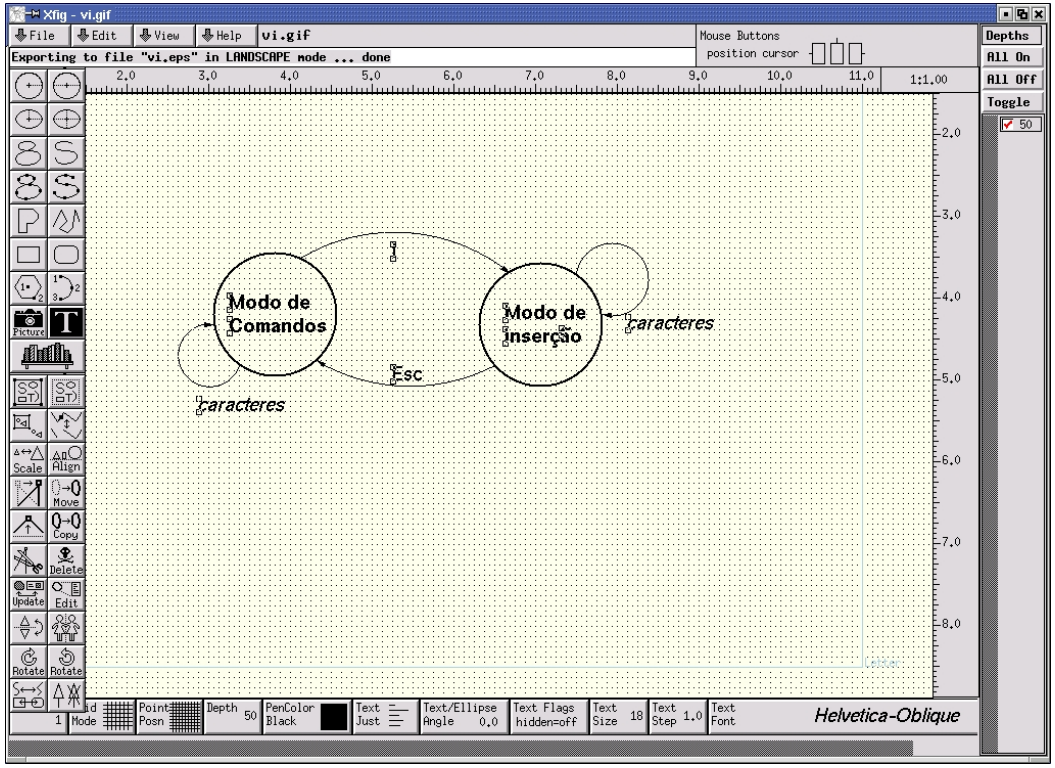


Não confundir com modelo do sistema!

Modelos de Mapeamento

- Surgem quando o utilizador realiza repetidamente a mesma sequência de acções.
  - Tratam-se de estratégias aprendidas para atingir objectivos
- Podem causar interferências com outros modelos.

Exemplo



O Utilizador — Conclusão

- Sistema Interactivo = Utilizador + Artefacto tecnológico

  Os utilizadores são uma peça fundamental num sistema interactivo.

- O utilizador deve ser considerado durante o processo de desenvolvimento do artefacto

  Vimos algumas abordagens à modelação dos utilizadores.

- Trabalho nesta área procura compreender o modo de "*funcionar*" dos humanos tendo em vista desenvolver métodos e ferramentas que permitam prever como um dado utilizador interagirá com um sistema.

Um clássico: The Psychology of Everyday Things. D Norman (1988), Basic Books. Agora chama-se "The Design..."

O objectivo é desenvolver um sistema que satisfaça as necessidades do utilizador impondo-lhe o mínimo esforço possível (de memória, de interpretação, na execução das tarefas, etc.)

GOMS — revisitado

Resumo

- Uma *família* de técnicas de modelação utilizável para modelar o comportamento dos utilizadores.
- O comportamento é modelado em termos de **Objectivos**, **Operações**, **Métodos** e **Regras de selecção**.
- Resumidamente, um modelo GOMS consiste em Métodos que são utilizados para atingir objectivos. Um método é uma sequência de Operações que devem ser executadas e de (sub)Objectivos que devem ser alcançados. Se existe mais que um método para atingir um objectivo, utilizam-se Regras de selecção para determinar qual o método a usar.

Aplicabilidade

Um modelo GOMS fornece um modelo do comportamento do utilizador durante a realização de tarefas conhecidas. O modelo podem ser utilizado para diversos fins:

Analisar se toda a funcionalidade necessária está disponível

Se tivermos uma lista dos objectivos do utilizador, podemos utilizar o modelo para verificar se existem métodos para atingir cada um desses objectivos. (Atenção: isto não significa analisar se o sistema permite ou não atingir os objectivos, antes analisar se o utilizador sabe ou não como utilizar o sistema para os atingir)

Tempos de execução

Os modelos GOMS permitem fazer previsões acerca do tempo necessário para os utilizadores atingirem determinados objectivos (assumindo que não ocorrem erros). Isto permite a comparação de diferentes alternativas de desenho e/ou a identificação de pontos fracos num dado desenho de uma interface.

Sistemas de ajuda inteligentes e wizards

Uma vez que os modelos GOMS são representações explícitas da actividade dos utilizadores, podem auxiliar o desenvolvimento de sistema de ajuda (*help*) capazes de fornecer suporte/ajuda ajustados aos objectivos do utilizador.

Objectivos

- Os Objectivos correspondem ao que o utilizador pretende atingir na utilização do sistema.
- Os objectivos podem ser definidos a vários níveis de abstracção, desde objectivos de alto-nível (preparar apontamentos) até objectivos de baixo-nível (apagar uma palavra).
- Os objectivos de alto-nível podem ser decompostos em subobjectivos, organizando-se de forma hierárquica.

Operadores

- Os operadores são as acções perceptuais, motoras ou cognitivas empregues pelos utilizadores para atingir objectivos (duplo click do rato, premir uma tecla, etc.).
- Os operadores são elementos atómicos (não decomponíveis).
- Assume-se que cada operador demora um tempo constante e independente do contexto a ser executado pelo utilizador.

Métodos

- Os Métodos são procedimentos que definem como atingir os objectivos.
- Um método é essencialmente um algoritmo que o utilizador aprendeu e que determina a sequência de subobjectivos e operadores necessária para atingir um objectivo.

Regras de selecção

- As Regras de selecção definem que métodos devem ser utilizados para atingir um dado objectivo, num dado contexto.
- As regras de selecção representam o conhecimento do utilizador sobre qual o método que deve ser utilizado.
- As regras de selecção tomam a forma de expressões condicionais (por exemplo, **se** a palavra a apagar está está a menos de três linhas de distância do cursos, **então** aplicar o método X, **senão** aplicar o método Y)

Variantes

CMN-GOMS

- A versão original (de Card, Morn, and Newell) é conhecida como CMN-GOMS.
- Os modelos CMN-GOMS podem ser representados em forma de programa tornando possível não só a sua análise como a sua execução.
- Ver um exemplo na página 46.

NGOMSL — Natural GOMS Language

- A NGOMSL fornece uma notação baseada em linguagem natural para a representação de modelos GOMS.
- Os modelos NGOMSL são baseados numa teoria cognitiva: CTT (cognitive complexity theory). Esta permite a incorporação de operadores *internos* (por exemplo, manipulação da memória de trabalho). Deste modo, NGOMSL pode ser utilizada para estimar tempos de aprendizagem de tarefas.

NGOMSL — exemplo

```
  Method for goal: Cut text
  Step 1.  Accomplish goal: Highlight text.
  Step 2.  Return that the command is CUT, and
             accomplish goal: Issue a command.
  Step 3.  Return with goal accomplished.
...
Selection rule set for goal: Highlight text
  If text-is word, then accomplish goal: Highlight word.
  If text-is arbitrary, then accomplish goal: Highlight arbitrary text.
  Return with goal accomplished.
...
Method for goal: Highlight arbitrary text
  Step 1.  Determine position of beginning of text (1.20 sec)
  Step 2.  Move cursor to beginning of text        (1.10 sec)
  Step 3.  Click mouse button.                     (0.20 sec)
  Step 4.  Move cursor to end of text.             (1.10 sec)
  Step 5.  Shift-click mouse button.               (0.48 sec)
  Step 6.  Verify that correct text is highlighted (1.20 sec)
  Step 7.  Return with goal accomplished.
```

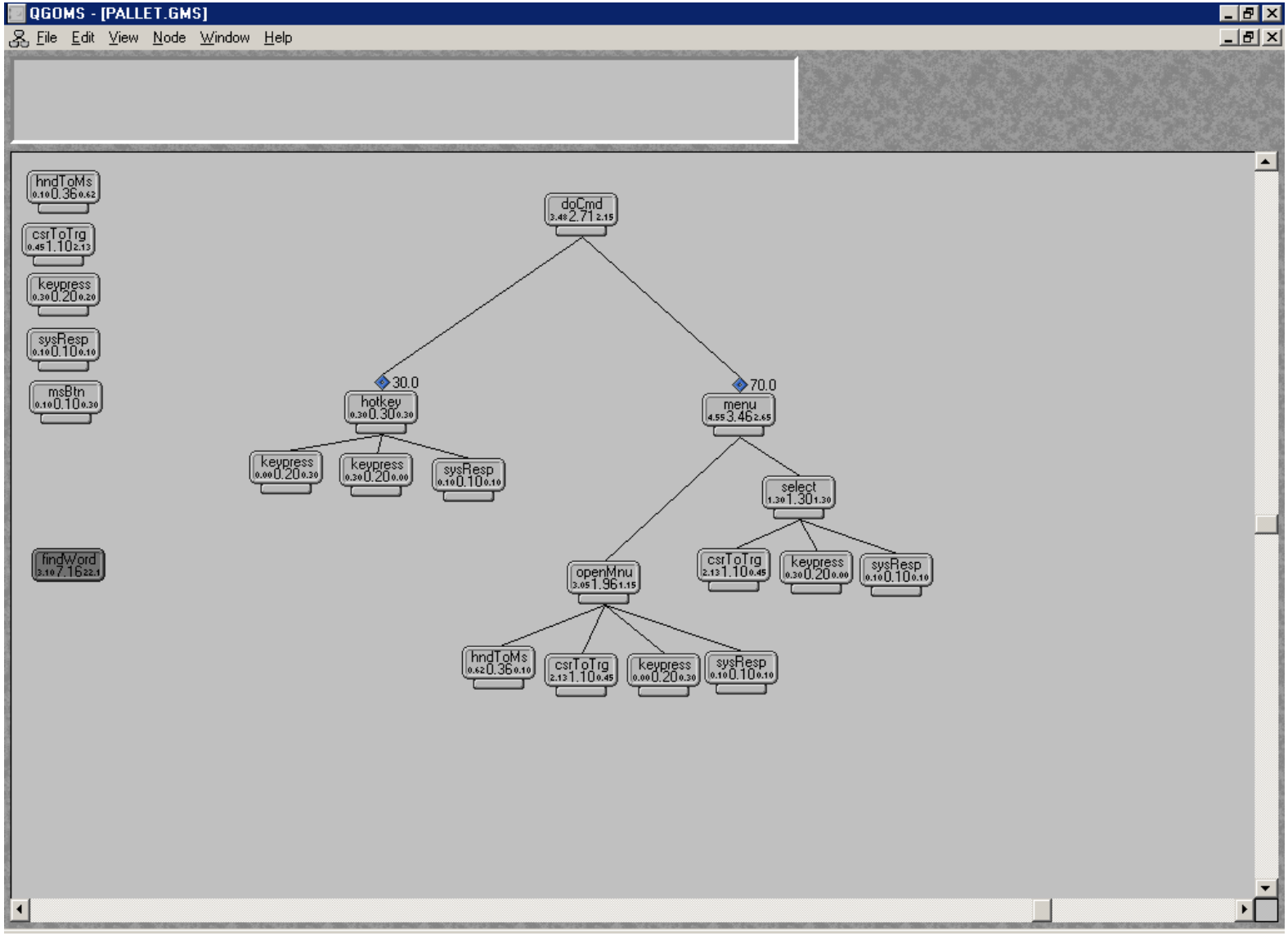
QGOMS

- QGOMS (*Quick and dirty GOMS*) é uma ferramenta para o desenvolvimento e análise de modelos GOMS.
- Como o nome indica, QGOMS utiliza uma versão simplificada da técnica original.
- Os modelos são representados graficamente numa estrutura hierárquica em forma de árvore.
  - Objectivos são representados pelos nodos internos da árvore;
  - Operadores são representados pelas folhas da árvore;
  - Métodos estão representados de forma implícita na estrutura da árvore;
  - Regras de selecção não são representadas explicitamente, em sua substituição é possível associar probabilidades às sub-árvores para indicar a frequência com que serão utilizadas.

Testar o QGOMS:

1. Fazer download de http://cob.isu.edu/qgoms/ e instalar.
2. Estudar exemplo.
3. Modelar um processo de login.

A ferramenta QGOMS



Tempos de realização para operadores (em segundos)

K Premir tecla ou botão: .28

Dependendo do utilizador os valores podem variar desde .08 até 1.20.

P Apontar o rato a um alvo no écran: 1.10

O valor varia com a distância (cf. lei de Fitt) — entre .8 e 1.5 num écran normal.
O valor 1.1 é um valor médio.

Este valor não inclui a acção de premir o botão do rato (.2)

H Colocar a(s) mão(s) no teclado ou outro dispositivo: .40

Características adicionais da ferramenta

- Permite calcular o tempo necessário para atingir os objectivos.
- Adiciona .20 por cada sub-objectivo no modelo (preparação mental!).
- Permite definir variáveis globais com tempos pré-definidos para diferentes operadores.
- Permite desactivar sub(árvores) de modo a que não sejam consideradas na análise.

Análise

- Analisar funcionalidade

  Análise tem que ser feita por inspecção visual dos modelos — deve existir uma árvore para cada um dos objectivos do utilizador.

  Análise de consistência é também feita por inspecção analisando se as (sub)árvores de uma dada tarefa são idênticas (quando a tarefa aparece em vários pontos do modelo).

- Tempos de execução

  São calculados automaticamente pela ferramenta. É possível analisar diferentes cenários manipulando os tempos associados aos operadores, as probabilidades dos nodos, ou desactivando sub-árvores.

- Calcula o tempo de aprendizagem com base num valor de 30s por nodo. É possível marcar nodos como aprendidos.

mas...

não fornece suporte para o desenvolvimento de modelos cognitivamente plausíveis.

GOMS — Críticas e Limitações

Opiniões?

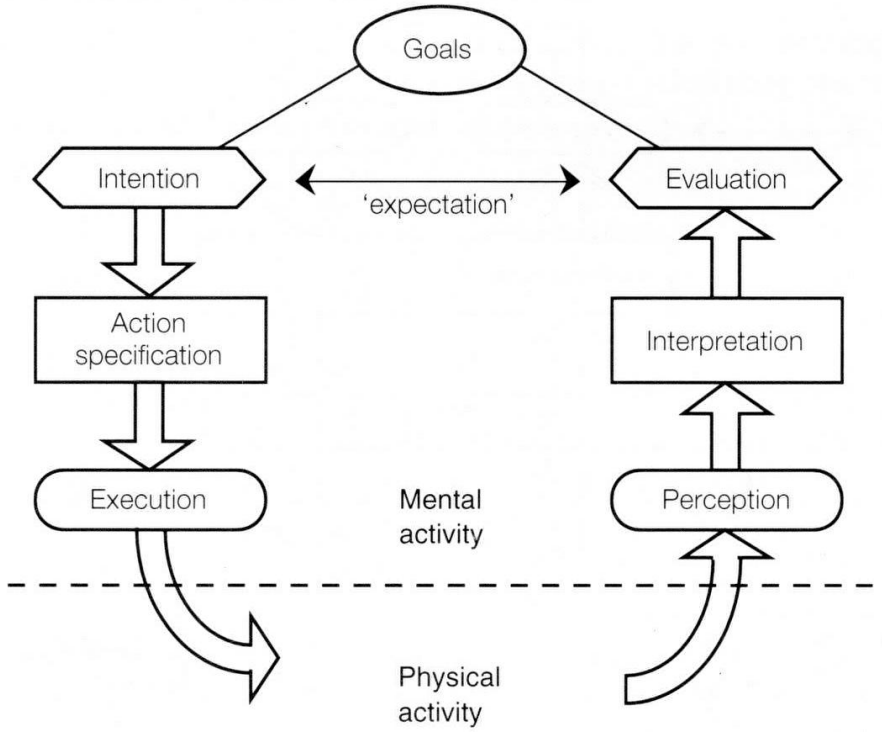
- As previsões são apenas válidas para utilizadores *peritos* que não comentam erros (e dependem fortemente da validade das previsões de tempos utilizadas para os Operadores, Regras, etc.)
  - Até os peritos cometerão erros de tempos a tempos — uma boa teoria deveria permitir analisar a probabilidade de acontecerem erros por parte de utilizadores já habituados à interface (exemplo: máquinas de pagamento nos parques de estacionamento).
  - Utilizadores noviços necessitarão de aprender a utilizar o sistema — uma boa teoria deveria permitir analisar dificuldades de aprendizagem (NGOMSL procura responder a esta crítica).
- Toda a modelação é baseada na noção de que o utilizador utiliza o sistema para atingir objectivos — isto deixa de fora formas de utilização exploratórias (exemplo: muita da navegação na web).
- Não fornece nenhuma indicação relativamente à agradabilidade de utilização.

- Baseia-se num modelo sequencial da cognição, assumindo que o utilizador realiza uma actividade de cada vez (propostas como NGOMSL ou CPM-GOMS procuram responder a esta crítica).
- Em geral, desenvolver um modelo GOMS para um sistema de média dimensão revela-se uma actividade complexa e demorada.
- Existe a tendência para modelar como o sistema deve ser utilizado e não tanto como é utilizado na realidade — entramos no campos dos modelos de tarefas.

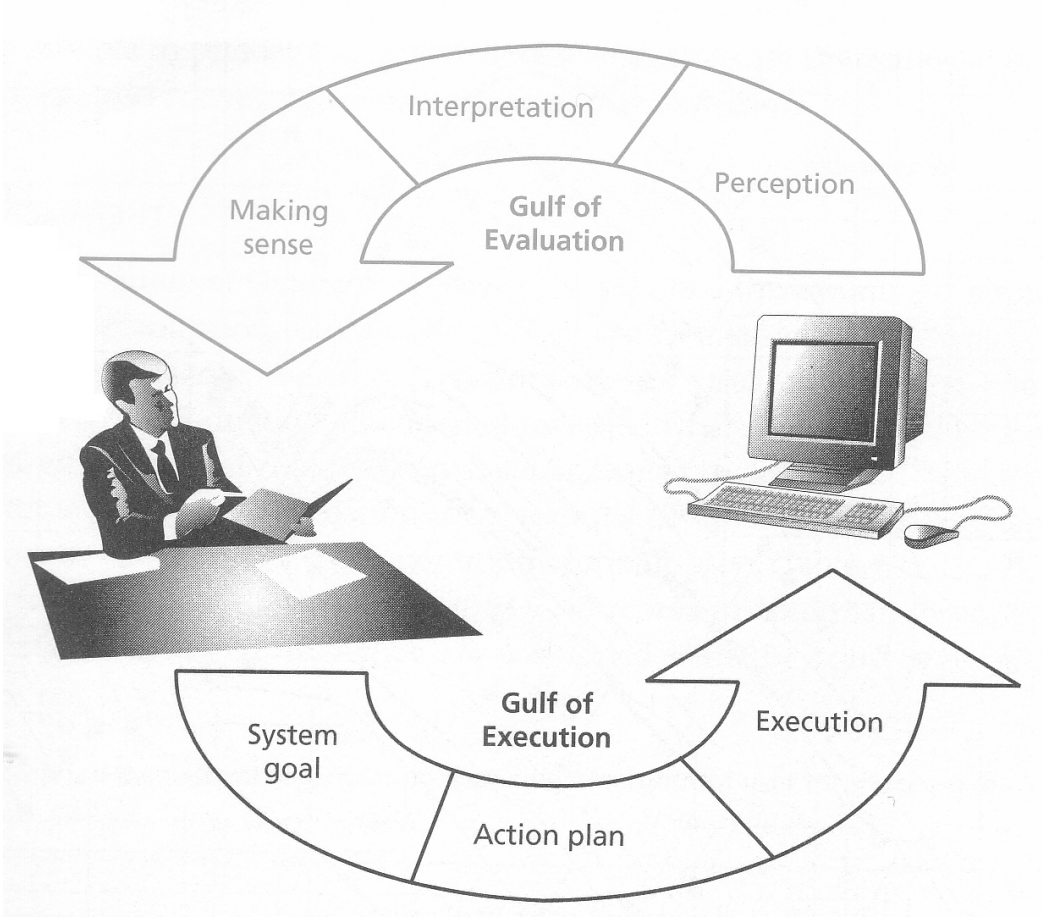
Modelos da Interacção

- Não tenta modelar o funcionamento do processo, mas antes a sua manifestação.
- Proporciona um enquadramento para compreender como as acções do utilizador se relacionam com os seus objectivos e com o sistema físico.

Modelo de Norman [Norman, 1988]

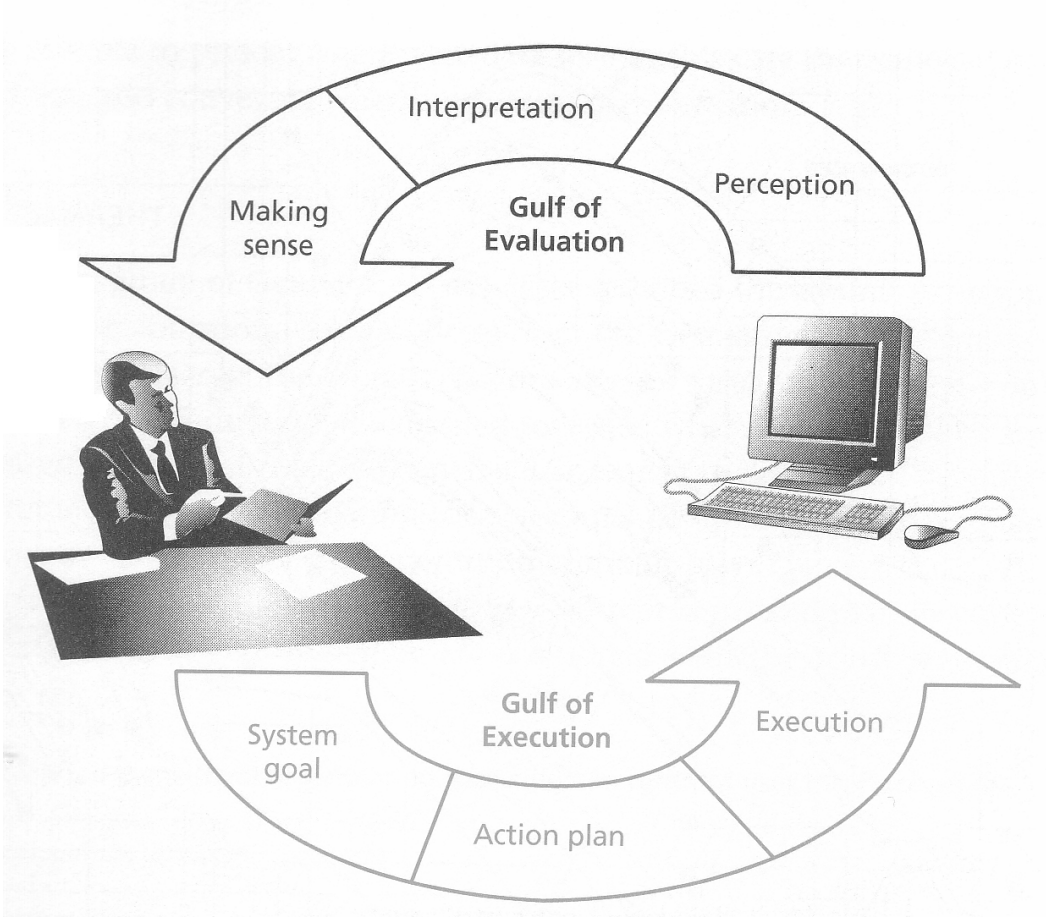


Golfo de Execução



- Esforço que o utilizador tem que realizar para efectuar determinada tarefa.
- Distância entre os objectivos do utilizador e a forma como pode atingi-los.
- Como diminui-lo?

Golfo de Avaliação



- Esforço que o utilizador tem que realizar para perceber a interface.
- Distância entre a informação que a interface fornece e aquela que o utilizador pretende.
- Como diminui-lo?

Metáforas

- Permitem explorar conhecimento pré-existente para diminuir os Golfos da Avaliação/Execução (exemplo: processador de texto vs. máquina de escrever).
- Podem surgir problemas:
  - quando partes da interface divergem da metáfora (ejectar um disco num Mac!);
  - quando partes da interface não tem correspondência na metáfora (máquina de escrever não tem *itálico*).

Dois tipos de metáforas:

- Verbais

  Utilizadas para explicar a interface;
- Virtuais

  A interface é a metáfora (cf. *desktop*).

Propriedades Genéricas — Princípios da IHC

- Os modelos da psicologia cognitiva permitem a definição de um conjunto de princípios para o desenvolvimento de sistemas interactivos.
- Estes princípios devem ser aplicados com cuidado pois o contexto especifico em que estão a ser aplicados pode comprometer a sua validade.

Alguns princípios comumente aceites:

- Consistência
- Compatibilidade
- Previsibilidade
- Adaptabilidade
- Economia e Prevenção de Erros
- Reversibilidade
- Estrutura

Consistência

- Similitude nos padrões de execução de tarefas, representação da informação, etc.
- A consistência permite reduzir o esforço de aprendizagem.
- Exemplo: guias de estilo do Windows, Mac, etc.

Compatibilidade

- Similitude entre o que o utilizador espera e o que a interface fornece.
- Consistência é interna à interface; Compatibilidade diz respeito à relação da interface com o utilizador — objectivos são semelhantes.
- Exemplo: utilização de metáforas, guias de estilo (para manter consistência entre diferentes aplicações e assim aumentar a compatibilidade).

Previsibilidade

- O utilizador deve ser capaz de perceber quais as acções que são possíveis (e qual o seu efeito).
- Tem um sub-princípio: observabilidade (para poder prever o comportamento do sistema, o utilizador tem que ser capaz de observar qual o seu estado).
- Exemplo: marcar o objecto que está seleccionado numa aplicação de desenho.

Adaptabilidade

- A interface deve adaptar-se ao utilizador. O utilizador, e não a interface, deve deter o controlo. (O que acontece quando temos mais que um utilizador?)
- Adaptabilidade em excesso pode degradar a consistência (exemplo: os menus adaptáveis do Windows).
- Exemplo: diferentes níveis de suporte para utilizadores com diferentes capacidades.

Economia e Prevenção de Erros

- A interface deve permitir ao utilizador realizar o trabalho com o mínimo de esforço.
- Um sub-aspecto é prevenir erros para evitar tarefas de recuperação dos mesmos (mas isto pode, também, tornar a interface menos económica).
- Exemplo: fornecimento de atalhos / impedir que ficheiros sejam apagados sem que existam cópias de segurança.

Reversibilidade

- A interface deve permitir ao utilizador reverter o estado do sistema quando comete erros.
- Isto é menos restritivo que a prevenção de erros e permite uma maior Adaptabilidade.
- Exemplo: comandos de *undo*

Estrutura

- A interface deve estar estruturada de modo a não sobrecarregar o utilizador com informação.
- A estrutura da interface deve ser compatível com o modo como os utilizadores organizam o seu próprio conhecimento (cf. modelos mentais).
- A estrutura da interface deve ser compatível com a estrutura interna do artefacto que suporta.

Regras de desenho

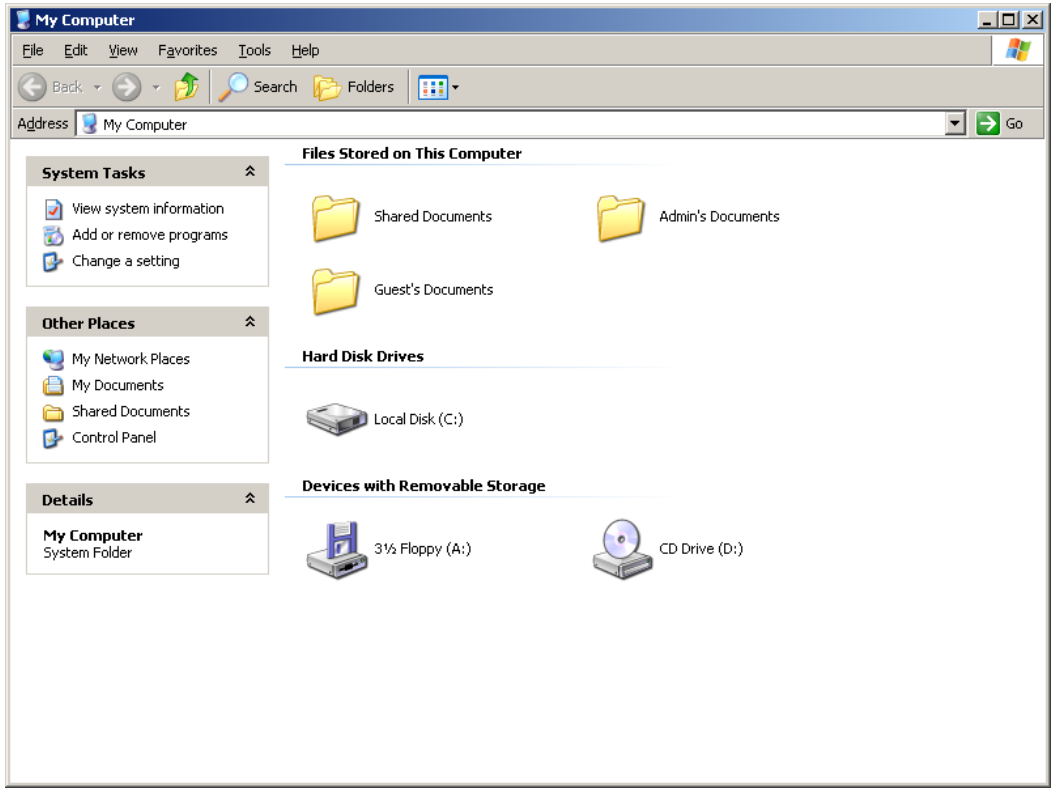
- Os princípios são orientações genéricas para o desenvolvimento e avaliação de sistemas interactivos.
- Estes princípios dão origem a regras concretas de desenho.
- Infelizmente a definição um conjunto de regras aplicável a qualquer tipo de sistema é uma tarefa impossível — tudo depende muito do utilizador e do contexto de utilização do sistema a ser desenvolvido.

Plano

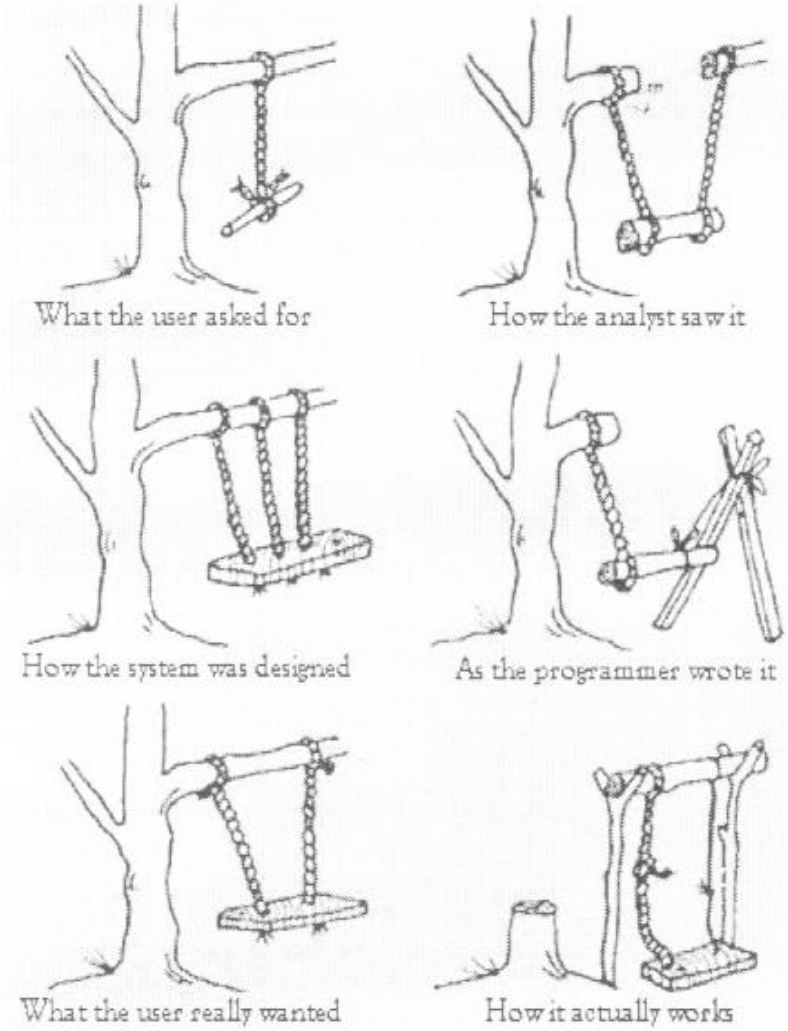
- Motivação
- O utilizador
- **Desenvolvimento centrado no utilizador**
  - O processo de desenvolvimento de software
  - Análise de Requisitos;
  - Modelação de Tarefas;
  - Modelação de Informação;
  - Modelação de Interacção;
  - *Guidelines*.
- Avaliação de sistemas interactivos

Desenvolvimento centrado no utilizador

Como evitar problemas de usabilidade?



Desenvolvimento centrado no utilizador



Como integrar questões de usabilidade no desenvolvimento de sistemas com uma componente interactiva?

- O que é que os utilizadores querem?

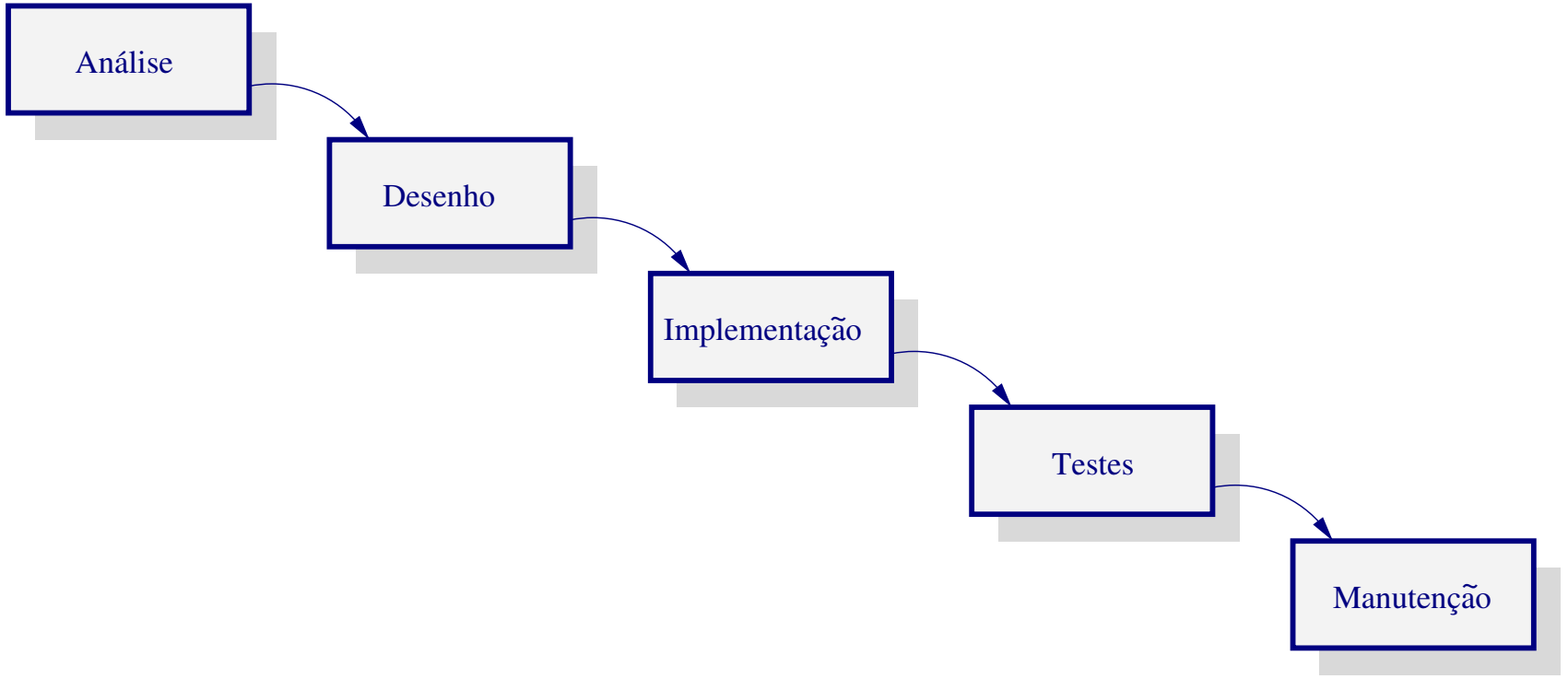
É necessário compreender:

- que tarefas o sistema deve suportar
- como sistema e utilizador *encaixam*

Um dos maiores desafios é construir o sistema *certo*.

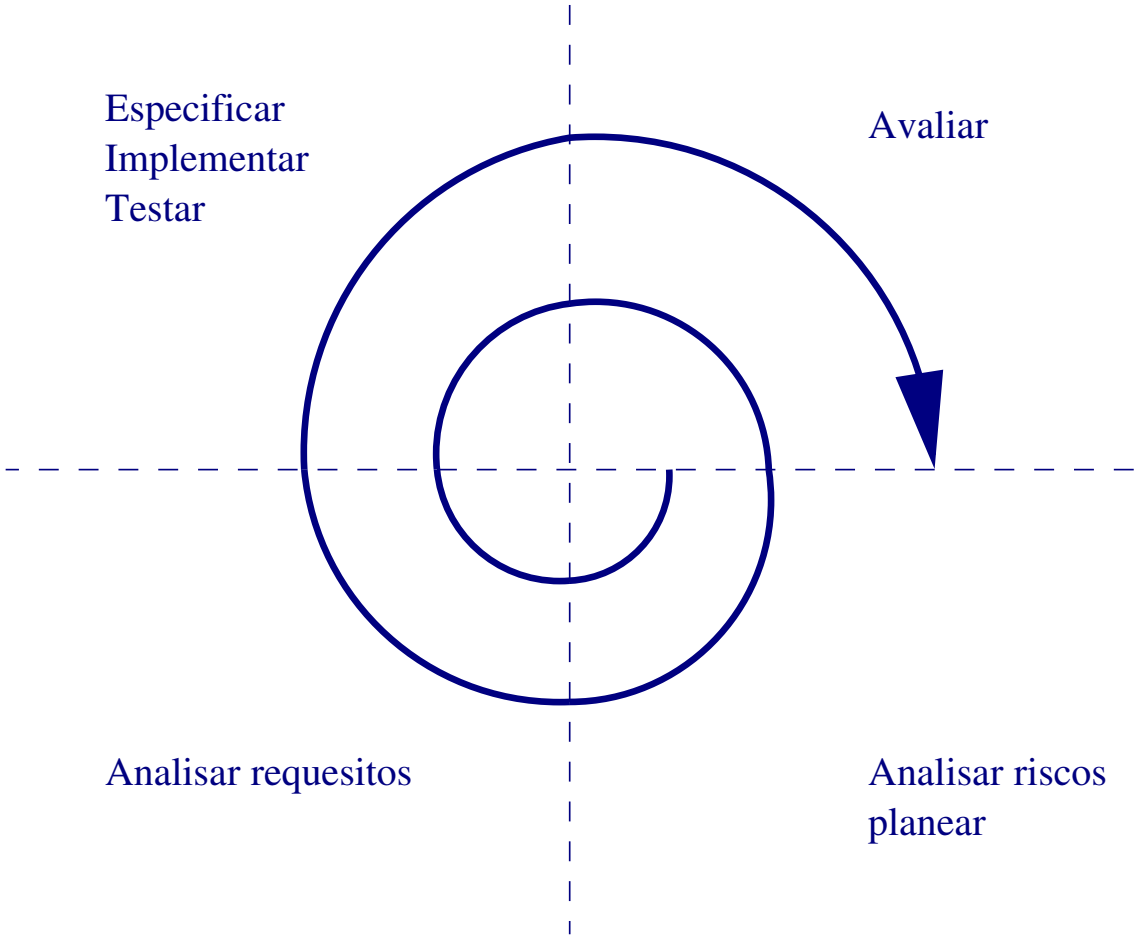
Métodos de desenvolvimento

Modelo Waterfall



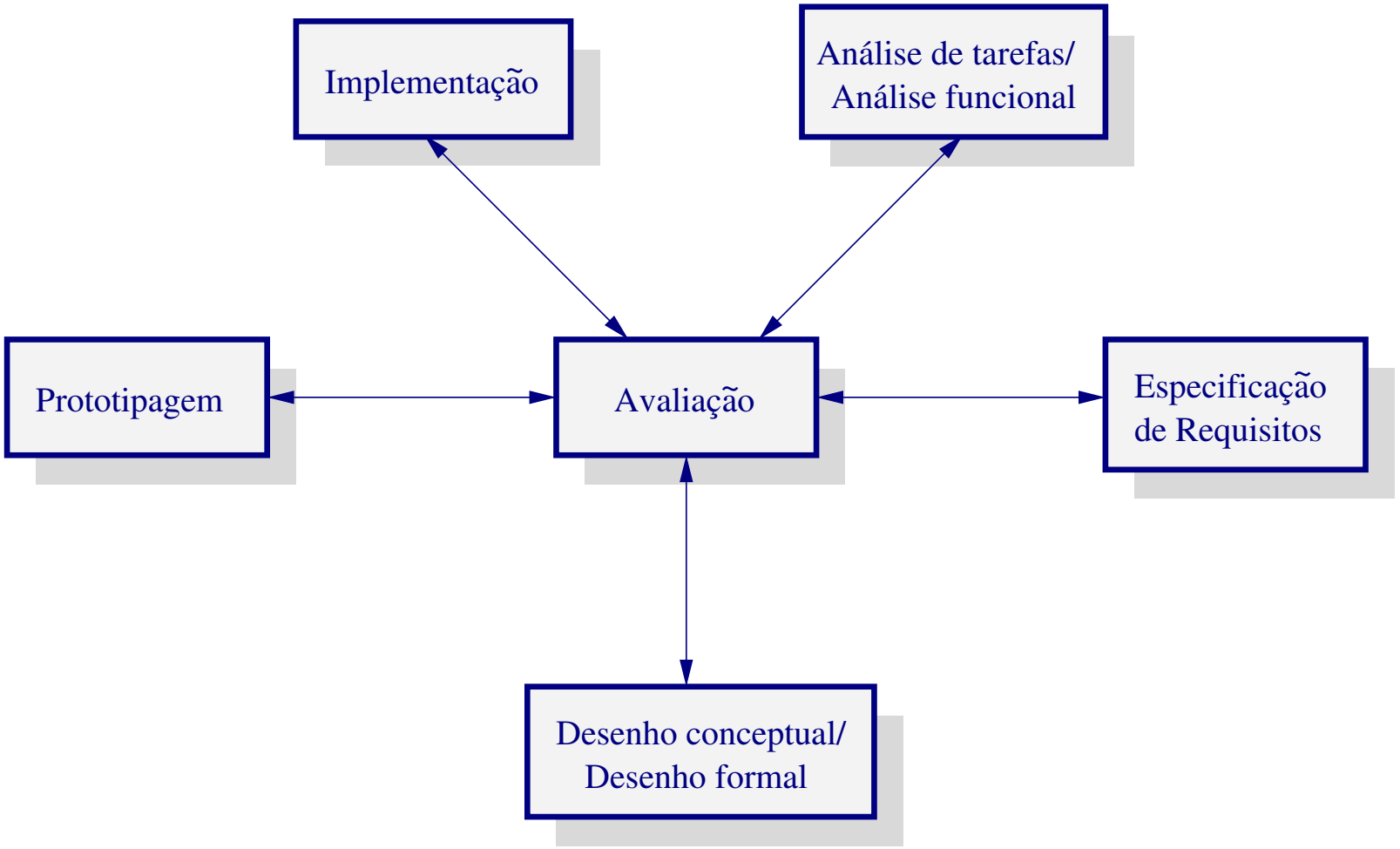
- Define o método como uma série de etapas executadas sequencialmente.
- Assume que é sempre possível tomar as decisões mais correctas — tarefa difícil, principalmente no que diz respeito à usabilidade.

Modelo em espiral



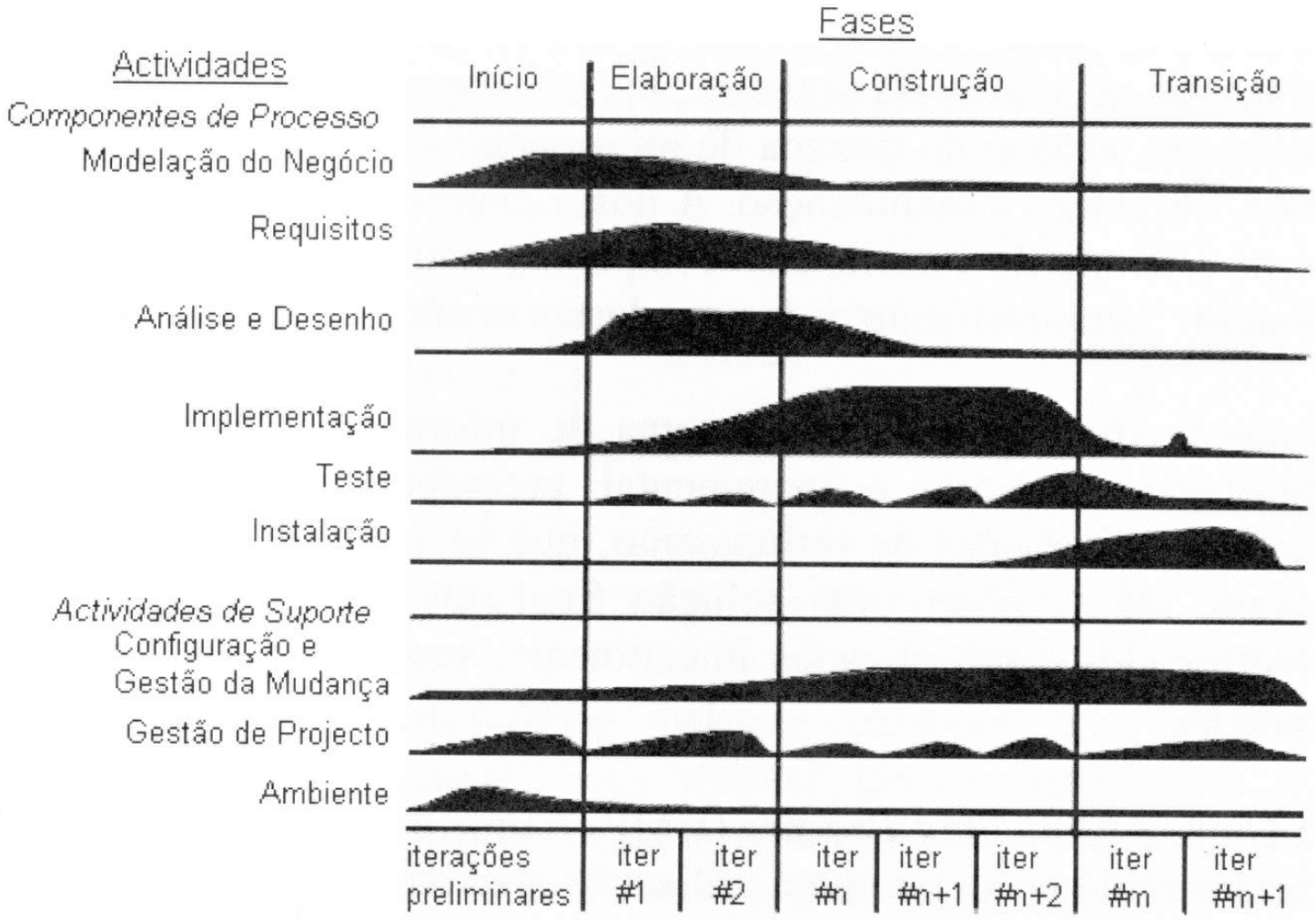
- Neste modelo reconhece-se a necessidade de iterar para controlar riscos.
- Na fase de avaliação deve ser tida em conta a questão da usabilidade.

Modelo em Estrela



- Este modelo coloca em destaque a necessidade de avaliar o sistema em todas as fases de desenvolvimento.
- Isto é particularmente relevante no que diz respeito à usabilidade.

RUP — Rational Unified Process



Início

- Identificar problema
- Decidir âmbito e natureza do projecto
- Fazer estudo de viabilidade

Resultado da fase: decisão de avançar com o projecto.

Elaboração (Análise/Concepção Lógica)

- O que vamos construir (quais os requisitos?)
- Como vamos fazê-lo? (qual a arquitectura?)
- Que tecnologias vamos utilizar?

Resultado da fase: uma arquitectura geral (conceptual) do artefacto.

Construção (Concepção Física/Implementação)

- Processo iterativo e incremental
- Em cada iteração: análise/especificação/codificação/teste/integração de parte do sistema

Resultado da fase: o artefacto!

Transição

- Acertos finais e instalação
- Optimização, formação.

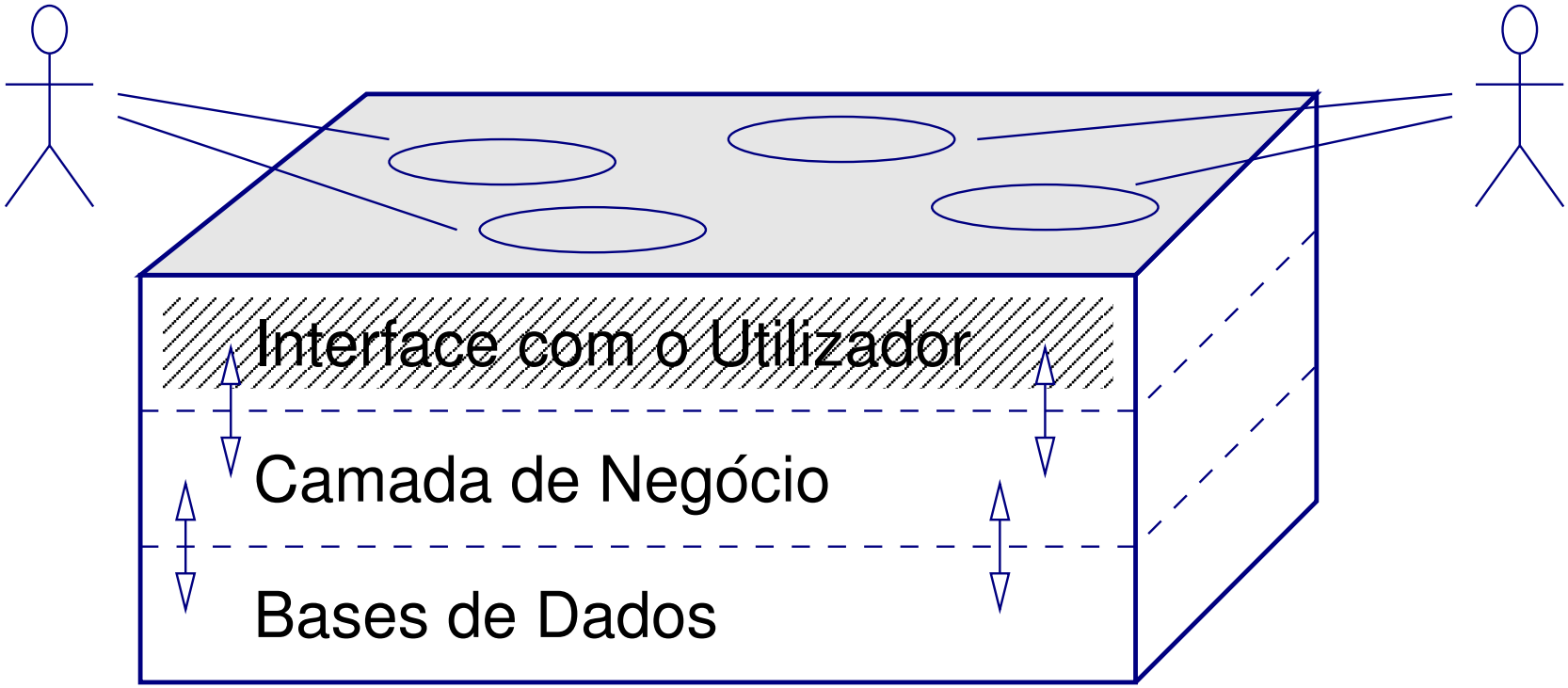
Resultado da fase: um artefacto instalado e 100% funcional.

Questão :

100% funcional para quem?

Para quem o desenvolveu/instalou, ou para quem o vai utilizar?!

Modelo de Três Camadas

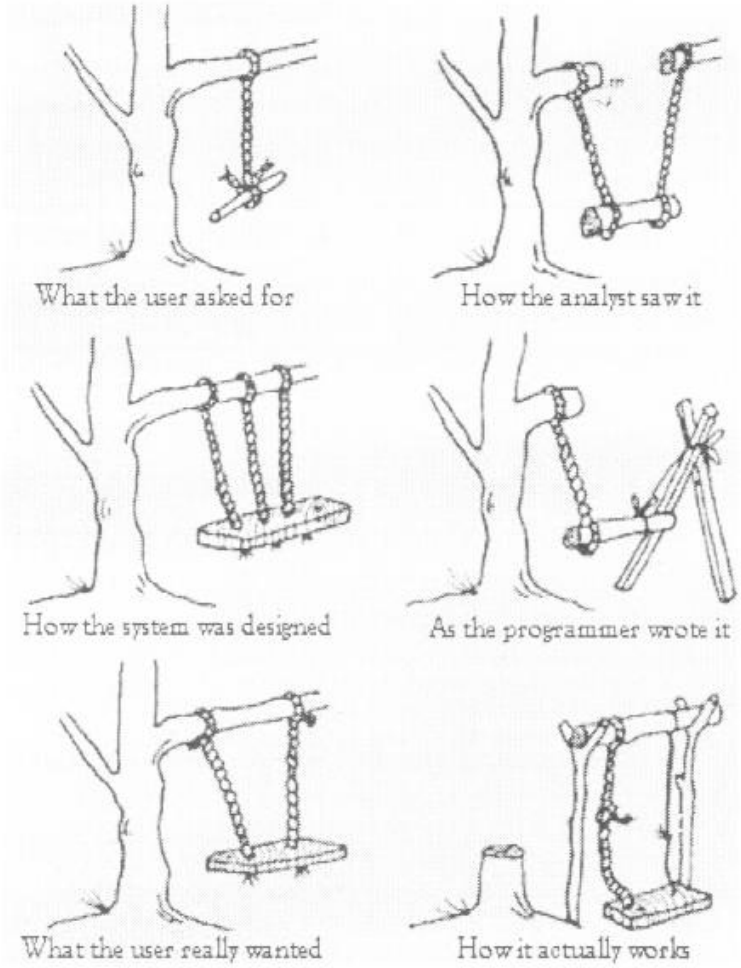


IHC e Engenharia de Software estão demasiado separadas...

Considerar o utilizador durante o desenvolvimento

- Identificar os **utilizadores**;
- Identificar os **requisitos** (dos utilizadores);
- Analisar e desenhar as **tarefas**;
- Analisar e desenhar a **informação**;
- Analisar e desenhar a **interacção**;
- **Implementar** a camada interactiva;
- **Testar** a usabilidade (só agora?!).

Conhecer os Utilizadores! Conhecer as Tarefas!



Conhecer os Utilizadores

- Quem vai utilizar o software? — tem implicações nos objectivos de usabilidade e em como eles podem ser atingidos.
  - Classes de utilizadores;
  - Níveis de perícia.

Classes de Utilizadores

- Utilizadores directos — utilizam o sistema eles próprios;
- Utilizadores indirectos — pedem a outros que utilizem o sistema por eles;
- Utilizadores remotos — não utilizam o sistema directamente mas dependem dos resultados por ele produzidos;
- Utilizadores de apoio (técnico) — são responsáveis pelo bom funcionamento do sistema.

Outro ponto a considerar: os utilizadores estão *obrigados* a utilizar o sistema ou têm poder de escolha?

Classes de utilizadores vs. Actores no UML

Níveis de perícia

- Utilizadores noviços — pouca ou nenhuma experiência / pode não estar motivado para aprender;
  - têm necessidade de *feedback* frequente;
  - preferem que lhes sejam apresentadas opções;
  - sentem receio de causar alguma avaria no sistema.
- Utilizadores intermédios — possuem características tanto de noviços, como de peritos;
  - necessitam de bons sistemas de ajuda e documentação;
- Utilizadores peritos — conhecem bem o sistema;
  - têm necessidade de atalhos;
  - preferem sistemas que possam costumizar à sua vontade.

Mas, quão perito é um perito? Utilizadores tendem a especializar.

Níveis de perícia — o factor avaria

- Utilizador noviço — alguém que tem medo de carregar numa tecla e avariar o computador;
- Utilizador intermédio — alguém que avariou o computador e não o sabe arranjar;
- Utilizador perito — alguém que avaria o computador dos outros!

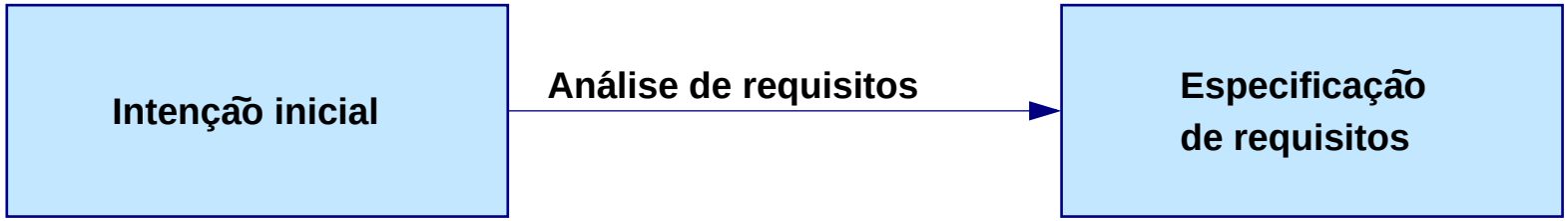
É tudo uma questão de auto-confiança!

Atenção

Estas categorias são meramente indicativas, não as aplicar de forma rígida.

Lista: pag. 91

Análise de Requisitos



Os requisitos definem o sistema a construir.

Tipos de requisitos:

- Funcionais — o que deve o sistema fazer? (não detalhar demasiado).
- de Dados — que dados são manipulados pelo sistema?
- de Usabilidade — que características de usabilidade deve o sistema ter?
  1. Identificar/estudar os utilizadores;
  2. Identificar/estudar as tarefas a serem executadas;
  3. Definir objectivos de usabilidade.
- outros requisitos não funcionais.

Identificar/estudar os utilizadores e as tarefas

Métodos para estudo dos utilizadores/tarefas:

- Realizar entrevistas

  O método base. É rápido e não necessita de muito planeamento. No entanto, nem sempre é possível ter acesso directo aos utilizadores e nem sempre as descrições que os utilizadores fazem correspondem à realidade.

- Observação dos utilizadores;

  Permitem capturar o modo como as actividades são realizadas. No entanto, o método utilizado pode interferir com a actividade a ser observada (cf. Princípio da Incerteza). Técnicas: gravação vídeo e posterior transcrição / "Think aloud"; Observação passiva; Estudos etnográficos.

- Utilizar questionários;

  É necessário um planeamento detalhado. Questionários devem ser simples, não ambíguos, e devem fornecer dados precisos e relevantes.

- Incluir utilizadores na equipe de desenvolvimento;

  É necessário cuidado pois nem sempre os utilizadores serão *fiáveis*.

Tarefas a serem executadas

- Analisar como as tarefas são executas actualmente.
- Faz a ponte para a fase seguinte: Modelação de Tarefas.

No desenvolvimento de um sistema interactivo os requisitos estão em constante evolução.

Definir objectivos de usabilidade

Possíveis objectivos de usabilidade (ver página 20):

Possible measurement criteria (Tyldesley, 1988).

(1)	Time to complete task.
(2)	Percentage of task completed.
(3)	Percentage of task completed per unit time (speed metric).
(4)	Ratio of successes to failures.
(5)	Time spent on errors.
(6)	Percentage number of errors.
(7)	Percentage or number of competitors that do this better than current product.
(8)	Number of commands used.
(9)	Frequency of help or documentation use.
(10)	Time spent using help or documentation.
(11)	Percentage of favourable:unfavourable user comments.
(12)	Number of repetitions of failed commands.
(13)	Number of runs of successes and of failures.
(14)	Number of times the interface misleads the user.
(15)	Number of good and bad features recalled by users.
(16)	Number of available commands not invoked.
(17)	Number of regressive behaviours.
(18)	Number of users preferring your system.
(19)	Number of times users need to work around a problem.
(20)	Number of times the user is disrupted from a work task.
(21)	Number of times the user loses control of the system.
(22)	Number of times the user expresses frustration or satisfaction.

(processo de desenvolvimento: pag. 91)

Modelação de Tarefas

Tarefa → uma sequência de actividades executada para atingir um objectivo.

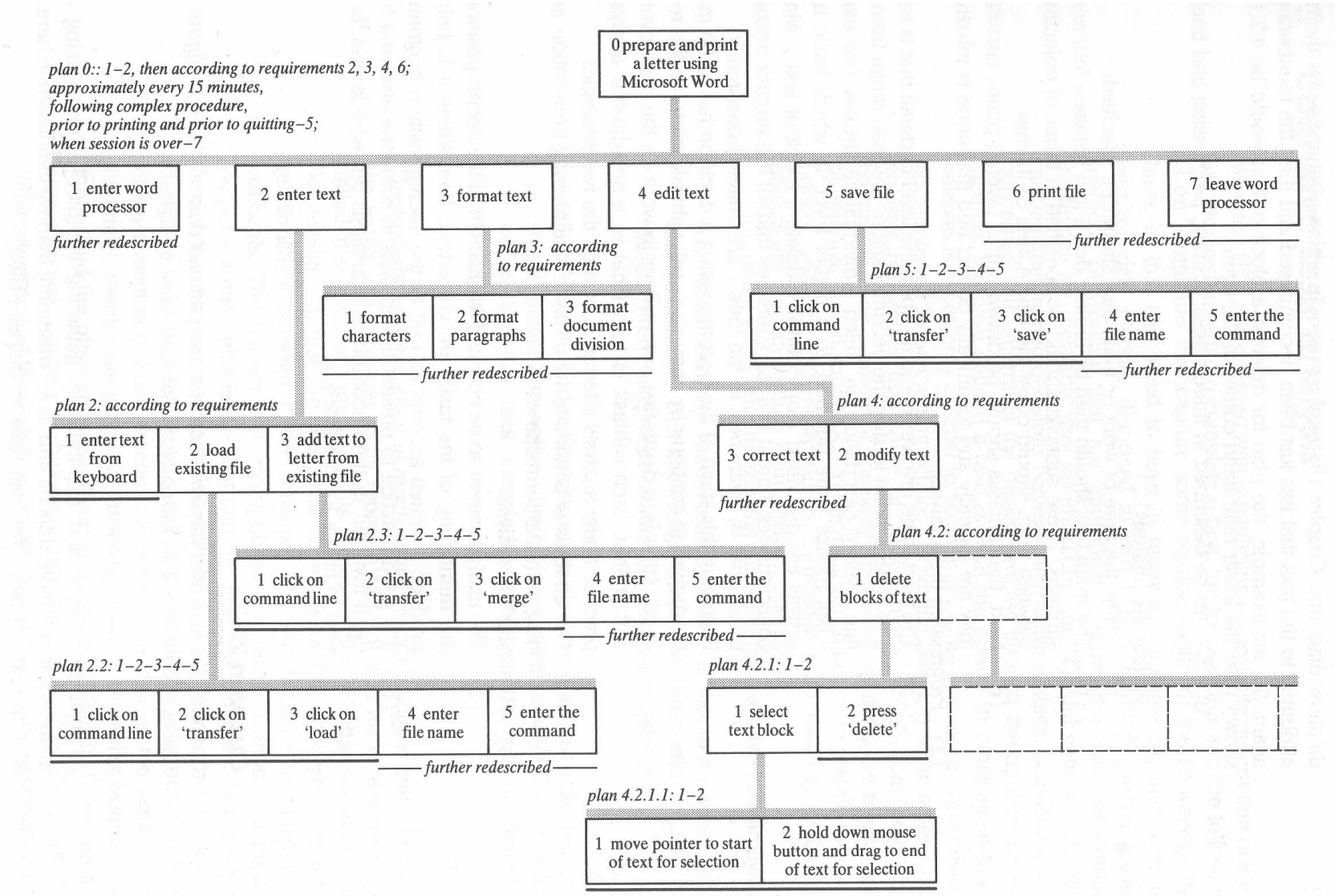
- Na fase anterior identificou-se como as tarefas são executadas actualmente.
- Nesta fase define-se como as tarefas passarão a ser executadas no sistema a ser desenvolvido.

Tarefas devem ser:

- Eficazes
- Compreensíveis (começar a pensar aqui)
- Satisfatórias(!)

Atenção: Trabalhar a partir das tarefas identificadas na fase anterior, mas evitar ficar preso a elas.

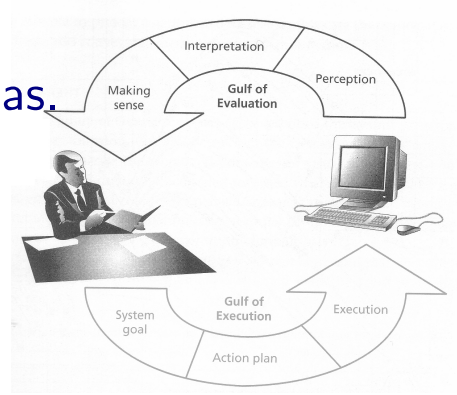
Hierarchical Task Analysis



Alocação de Funções

- Numa primeira fase as tarefas são definidas sem a preocupação de saber quem as executará (utilizador ou software).
- Numa segunda fase é necessário decidir quem executa que actividades — alocação de funções.
- Este processo ajuda a definir a funcionalidade do sistema (em particular ajuda a que se implemente apenas aquilo que é necessário!).

Modelação de Informação



- Desenhar uma interface que suporte as tarefas identificadas.

Aspectos a considerar

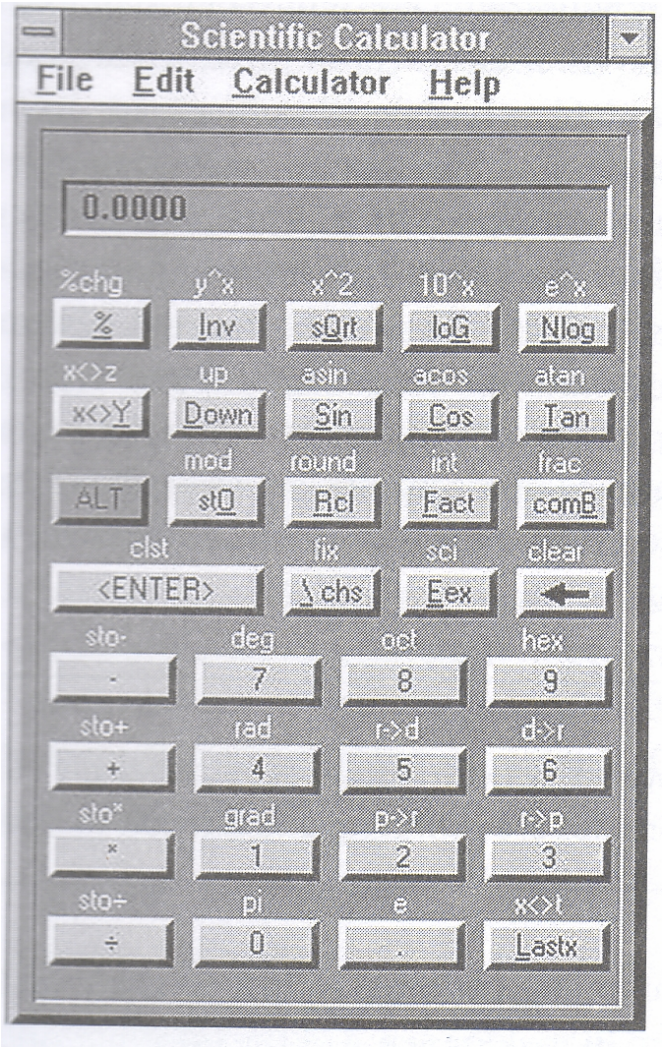
- Para cada tarefa, que informação é relevante?
- Como representar o estado do sistema? — noção base: percepção
  - visibilidade vs. observabilidade
- Consistência e coerência da interface.
- *Overdesign* — "Informação" presente na interface deve ser a necessária, demasiados embelezamentos vão tornar a interface "pesada" e/ou confusa.

Percepção

- Visão construtivista vs. visão ecológica
  - Visão construtivista
    - A percepção socorre-se de representações e da memória (aproveitar conhecimento pré-existente).
    - O que vemos não é uma *fotografia* do mundo, mas um modelo obtido por selecção e transformação da informação percepcionada.
    - Gestalt — capacidade dos nossos sentidos "organizarem" os estímulos recebidos: proximidade / similitude / fecho (*closure*) / continuidade / simetria
  - Visão ecológica
    - A percepção é um processo em que a informação é simplesmente detectada (e não construída).
    - A nossa imagem do mundo é construida através de exploração activa desse mundo.
    - Noção de *affordance*.

Exemplo I

Exemplo II


Scientific Calculator
File Edit Calculator Help
0.0000
%chg y^x x^2 10^x e^x
% Inv sQrt loG Nlog
x<>z up asin acos atan
x<>Y Down Sin Cos Tan
mod round int frac
ALT stO Rcl Fact comB
clst fix sci clear
<ENTER> \ chs Eex
sto- deg oct hex
- 7 8 9
sto+ rad r->d d->r
+ 4 5 6
sto* grad p->r r->p
* 1 2 3
sto÷ pi e x<>t
÷ 0 . Lastx

Modelação de Interacção

- Desenhar uma interface que suporte as tarefas identificadas. (intimamente ligada à anterior)
- Para cada tarefa planear como será executada.

Aspectos a considerar

- Que tecnologia utilizar?
- Como é que a interface torna aparentes as acções disponíveis?
- Qual o grau de *feedback* fornecido?
- Qual o grau de suporte à exploração do sistema? (*undo*)
- Tratamento de erros.
- Desempenho (valores por omissão, *short-cuts*).

Guidelines

- As *Guidelines* fornecem conselho na solução de problemas.
- As *Guidelines* têm duas origens principais: psicologia e experiência prática.
- Não confundir *Guidelines* com regras de desenho!

  Guidelines são princípios genéricos, as regras são específicas.
  - Exemplo de guideline: fale a linguagem do utilizador.
  - Exemplo de regra: em Portugal as datas devem usar o formato DD/MM/AA.

Utilidade das Guidelines

- Introdução de novos conceitos;
- Auxílio na escolha de alternativas;
- Proposta de estratégias para resolver problemas;
- Auxílio na avaliação da interface.

Limitações das Guidelines

- Como escolher as mais apropriadas?
- Como aplicá-las correctamente?
- Como aplicar múltiplas *Guidelines* (potencialmente contraditórias).

Guidelines para sistemas multimédia (www.emmus.org)

- Forneça um desenho simples
- Procure a consistência
- Forneça *feedback* informativo
- Minimize a carga de memória do utilizador
- Forneça "fecho"
- Forneça *short cuts* para os utilizadores frequentes
- Forneça um bom suporte para o tratamento de erros
- Forneça *undo* e pontos de saída claramente identificados

Algumas outras fontes de Guidelines

- Designing the User Interface: Strategies for Effective Human-Computer Interaction. B. Shneiderman (1992), Addison-Wesley.
- Principles and Guidelines in Software User Interface Design. D. J. Mayhew, Prentice-Hall.

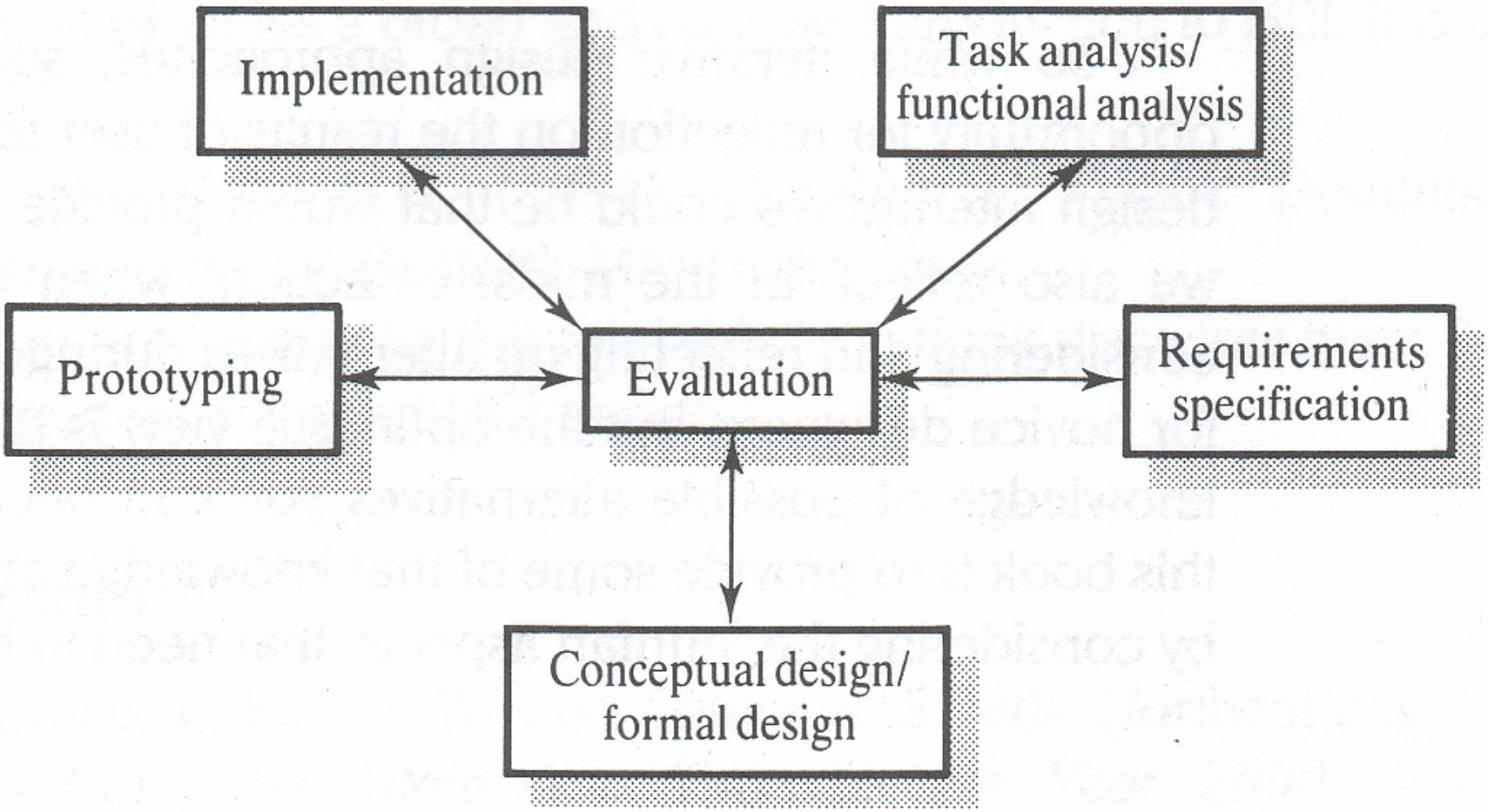
Desenvolvimento centrado no utilizador — Conclusão

- A função principal do software é auxiliar o utilizador a realizar determinadas tarefas.
- Assim, o software deve ser desenvolvido *à volta* dessas tarefas.
- Um software com boa usabilidade deverá suportar as tarefas que o utilizador pretende realizar (quer suportando processos de trabalho já existentes, ou introduzindo novos processos de trabalho).
- Em geral, um software com boa usabilidade terá os *Gulf of Evaluation* e *Gulf of Execution* tão pequenos quanto possível.
- Foram apresentadas as principais etapas de um método de desenvolvimento centrado no utilizador.

Plano

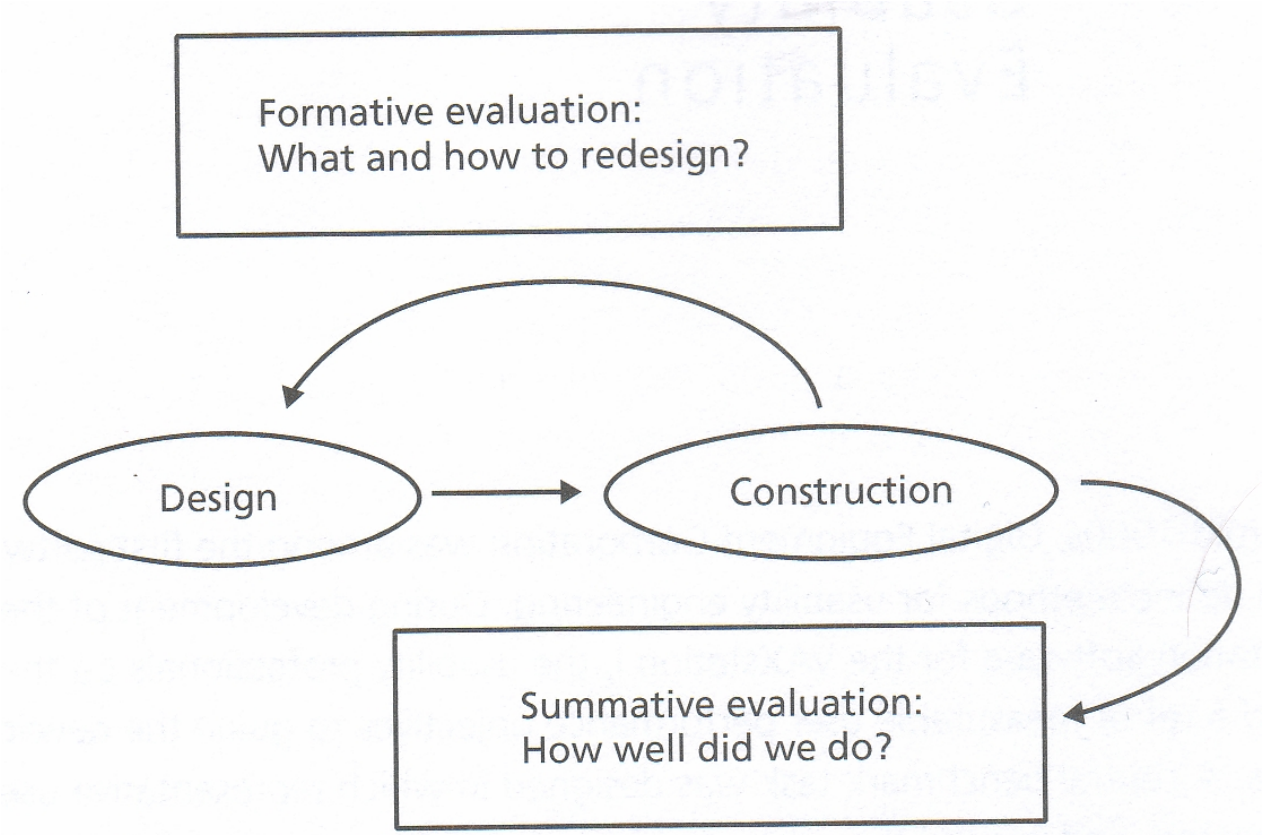
- Motivação
- O utilizador
- Desenvolvimento centrado no utilizador
- **Avaliação de sistemas interactivos**
  - Métodos empíricos;
  - Métodos analíticos.

Avaliação de sistemas interactivos



Que objectivos para a avaliação?

Garantir a qualidade do sistema!



Como definir qualidade?

- Cumpre *guidelines*? (Quais?)
- Obedece aos requisitos (de usabilidade)?
- **Os utilizadores conseguem utilizar o sistemas de forma eficaz, eficiente e agradável?**

Cumprir requisitos $\neq$ satisfazer os utilizadores.

O que queremos fazer?

- Compreender o problema e os utilizadores.
  - Que pretendem os utilizadores e que problemas sentem?
- Trabalhar para um objectivo (usabilidade).
  - Qual a melhor alternativa de desenho?
- Respeitar as normas.
  - O sistema cumpre as normas? (ISO, livros de estilo, etc.)
- Comparar sistemas.
  - Somos melhores ou piores que a competição?

Para objectivos de avaliação diferentes, métodos de avaliação diferentes.

É necessário identificar

- Quem são os utilizadores — A usabilidade deve ser definida em relação a um tipo específico de utilizador. Utilizadores experientes têm necessidades diferentes de utilizadores noviços.
- Que actividades pretendem/devem realizar — A usabilidade deve ser definida para tarefas específicas que o sistema deve suportar. No entanto, o sistema acabará muitas vezes por ser utilizado de formas não previstas inicialmente.
- Em que ambiente — O tipo de ambiente em que o sistema vai ser utilizado pode influenciar não só a usabilidade do sistema, mas a forma como os testes podem decorrer.
- Qual é o artefacto — A partida será o sistema desenvolvido, mas nas fases inicias de desenvolvimento o artefacto ainda não está totalmente definido.

(cf. modelo da página 22.)

Tipos de métodos

Métodos empíricos (Experimentais)

- Prototipagem e testes com utilizadores reais.
- Úteis para validar o sistema sob condições reais de utilização
- Dispendiosos, requerem muito tempo e organização

Métodos analíticos

- Baseados na análise de modelos dos utilizadores/sistema.
- Úteis na validação de decisões nas fases iniciais de desenvolvimento
- Tradicionalmente utilizados de forma manual e, mais ou menos, informal.

Métodos empíricos

- A análise de usabilidade é realizada recorrendo a utilizadores reais.
- O custo de aplicação é normalmente muito elevado.
- O principal objectivo é avaliar a usabilidade do sistema a partir da observação de como ele é utilizado pelos utilizadores finais.
  - Por este motivos são também chamados de **métodos interpretativos**.
- Podem ser realizados com o sistema final (avaliação sumativa) ou com um protótipo (avaliação formativa).
- Muitos dos métodos podem ser utilizados também na fase de recolha de requisitos de usabilidade.

Principais métodos

- Estudos de campo
  - Observação dos utilizadores em situações reais (com ou sem interacção entre utilizadores e avaliadores - *contextual enquiry* e análise participativa vs. abordagem etnográfica)
  - Útil tanto na fase de levantamento de requisitos como na fase de análise sumativa.
  - Questão: até que ponto a presença do avaliador altera o comportamento dos utilizadores?
- Análise em laboratório
  - Permite avaliar a interacção entre utilizador e sistema em condições mais controladas.
  - Normalmente utilizado para avaliar aspectos muito precisos da interface.
  - Questão: até que ponto as condições laboratoriais reflectem com precisão as condições reais de utilização? (*O estranho caso da dactilógrafa!*)

Métodos analíticos

- A análise da usabilidade é realizada sem recorrer a utilizadores reais.
- O custo de aplicação é muito mais baixo que o dos métodos empíricos.
- O principal objectivo é prever potenciais problemas de usabilidade (o que é diferente de avaliar a usabilidade).
  - Por este motivos são também chamados de **métodos preditivos**.

Principais métodos

- Métodos de inspecção
  - Avaliação Heurística
  - *Cognitive Walkthroughs*
- Métodos baseados em modelos

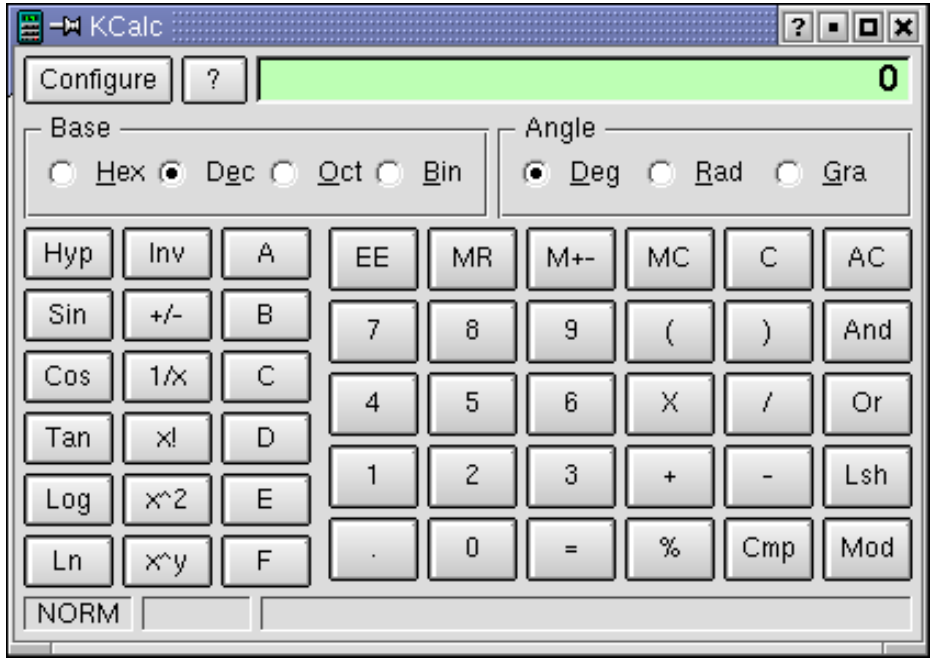
Avaliação Heurística

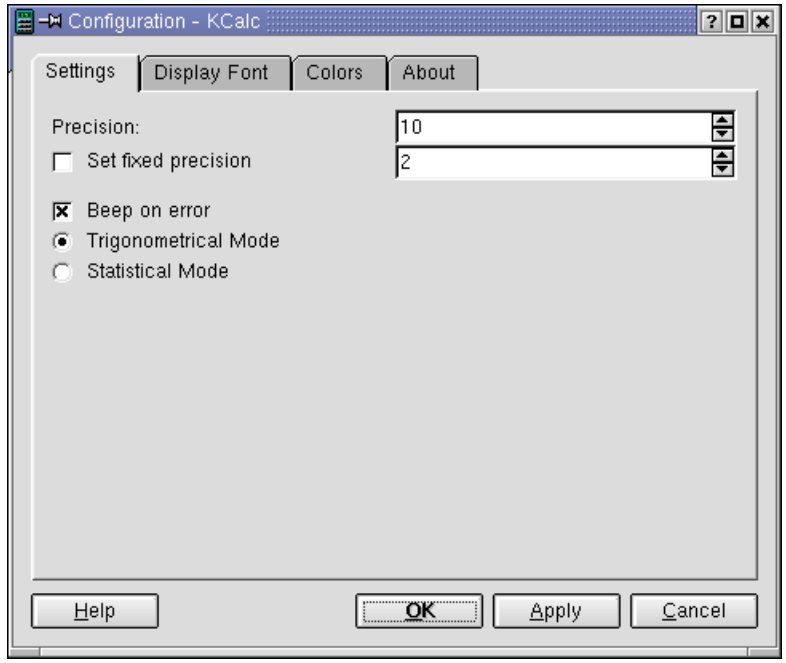
- Uma **equipa de avaliadores** realiza a análise.
  - idealmente os avaliadores deverão ser exteriores à equipa de desenvolvimento
  - cada avaliador realiza a sua análise de forma independente e os resultados são agregados no final
- Um conjunto de **heurísticas** é utilizado para guiar a análise.
  - Heurística: Metodologia que conduz à descoberta científica ou a resolução de problemas.
  - Neste contexto *heurísticas* podem ser vistas como *guidelines* de alto-nível.
- Pode ser realizada desde cedo no processo de desenvolvimento.
- Procedimento:
  - inspeccionar o fluxo da interface de écran para écran
  - inspeccionar, cada écran, face às heurísticas

Heurísticas propostas por Nielsen

- Utilizar diálogo simples e natural
- Falar a linguagem dos utilizadores
- Minimizar a carga de memória dos utilizadores
- Ser consistente
- Fornecer *feedback*
- Fornecer saídas claramente assinaladas
- Fornecer atalhos (*short cuts*)
- Fornecer boas mensagens de erro
- Prevenir os erros

Calculadora na página 105.


KCalc
Configure
?
0
Base
Hex
Dec
Oct
Bin
Angle
Deg
Rad
Gra
Hyp
Inv
A
EE
MR
M+-
MC
C
AC
Sin
+/-
B
7
8
9
(
)
And
Cos
1/x
C
4
5
6
X
/
Or
Tan
x!
D
1
2
3
+
-
Lsh
Log
x^2
E
.
0
=
%
Cmp
Mod
Ln
x^y
F
NORM


Configuration - KCalc
Settings
Display Font
Colors
About
Precision:
10
Set fixed precision
2
Beep on error
Trigonometrical Mode
Statistical Mode
Help
OK
Apply
Cancel

Cognitive Walkthroughs

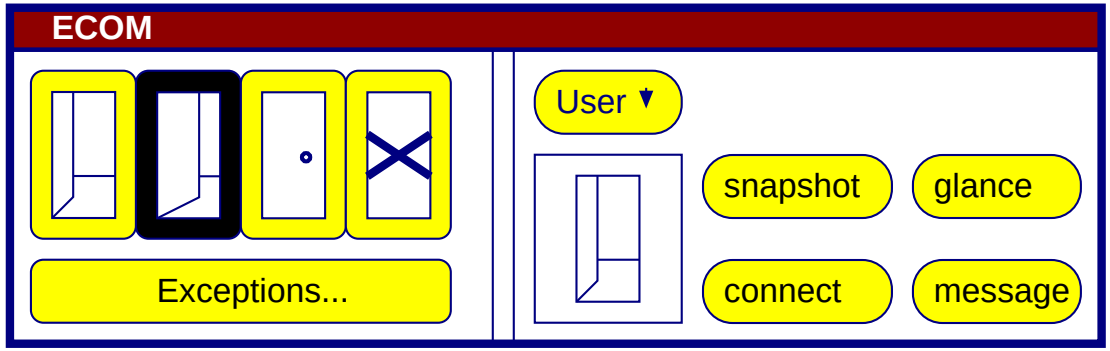
- Fornecem um método de analisar sistemas em termos de *aprendizagem exploratória*. (cf. teorias exploratórias do comportamento, página 52)
- Úteis para analisar sistemas que vão ser utilizados por utilizadores sem treino prévio.
- Fornece resposta à questão: Até que ponto consegue este sistema *guiar* um utilizador não treinado na execução de uma tarefa?
- Procedimento – simular os passos da teria exploratória de comportamento:

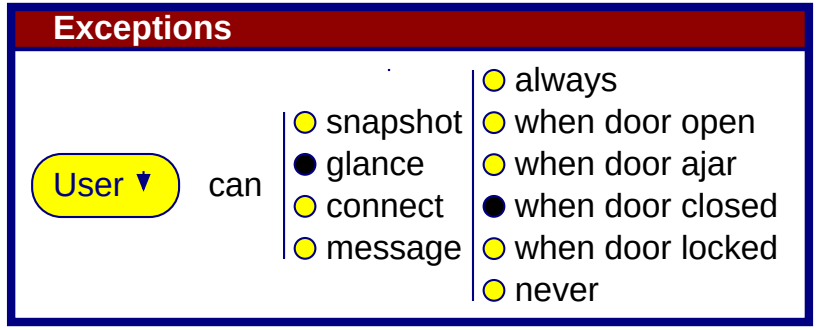
  Q1 A acção correcta é suficientemente evidente para o utilizador?

  Q2 Irá o utilizador associar as acções correctas ao que pretende fazer?

  Q3 Irá o utilizador interpretar de forma correcta a resposta do sistema à acção escolhida? (saberá se fez a escolha correcta?)

Exemplo: ECOM





Utilizar *Cognitive Walkthroughs* e Avaliação Heurística para analisar o sistema...

Métodos baseados em modelos

Os métodos anteriores levantam dois problemas:

- Os resultados estão dependentes da *qualidade* dos avaliadores.
- A análise é muito centrada em aspectos *locais* de usabilidade.

A utilização de modelos para realizar a análise pretende torná-la mais formal e repetível. Em contrapartida, o tipo de análise que se pode fazer fica mais restringido.

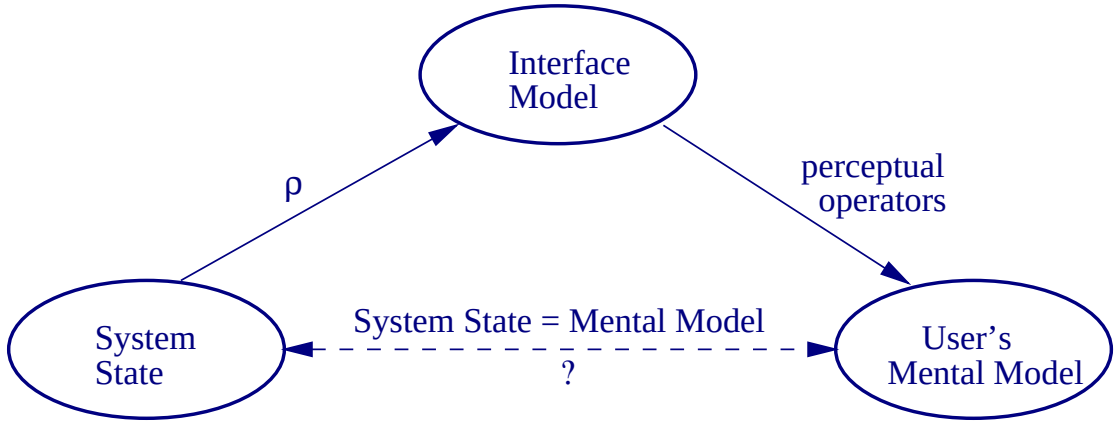
Baseados em modelos do utilizador

GOMS e KTA (ver página 39).

- Analisa-se o desempenho do utilizador, mas não do sistema...

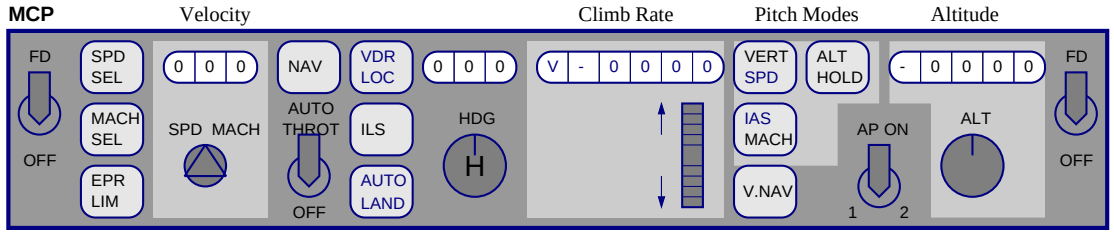
Baseados em modelos do sistema

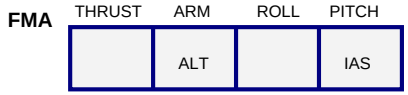
- Análise da apresentação
  - Desenvolvem-se modelos do estado do sistema, da apresentação do sistema e do modelo mental do utilizador e analisa-se a realação entre eles.



- Análise do comportamento
  - Constrói-se um modelo do comportamento do sistema e utilizam-se ferramentas de raciocínio automatizado para analisar esse comportamento.
  - Modelos são normalmente máquinas de estados finitas.

Exemplo: MCP – Mode Control Panel de um MD88





Será que o sistema é previsível?

Métodos analíticos – conclusões

- Aplicáveis desde as fases iniciais do desenvolvimento.
- Relativamente baratos de aplicar.
- Úteis para avaliar diferentes alternativas de desenvolvimento.
- Métodos baseados em inspecção: dependem muito da competência dos avaliadores e tendem a fazer optimização local da usabilidade.
- Métodos baseados na análise formal de modelos: permitem uma análise mais exaustiva, mas o tipo de análise que é possível fica mais restrita e é necessário fazer as perguntas certas. É necessária competência na área da verificação formal...
- Diferentes métodos permitem diferentes tipos de análise e requerem diferentes tipos de competências – Há que saber escolher o método apropriado.

Avaliação de sistemas interactivos — Conclusão

- Métodos empíricos são os mais dispendiosos, mas também aqueles em que mais problemas de usabilidade são identificados.
- Técnicas de *walkthrough* são úteis nas fases iniciais de desenvolvimento, onde diferentes alternativas devem ser consideradas.
- Avaliação heurística é pouco dispendiosa e eficaz, mas deve ser realizada por mais que um perito.
- Os métodos baseados em análise formal de modelos permitem obter resultados quantitativos, mas podem ser difíceis de aplicar (à excepção do GOMS, as abordagens são ainda muito recentes — mais trabalho é necessário).

Conclusão

Procurou dar-se uma visão geral sobre a questão da usabilidade no desenvolvimento de software interactivo.

Principal mensagem: conheça e pense nos utilizadores!

Para qualquer contacto: `jose.campos@di.uminho.pt`

References

[Faulkner, 2000] Faulkner, X. (2000). *Usability Engineering*. Grassroots Series. Palgrave, Hampshire, New York.

[Hollnagel, 1993] Hollnagel, E. (1993). *Human reliability analysis: context and control*. Academic Press, London.

[Leveson, 1995] Leveson, N. (1995). *Safeware: System Safety and Computers*. Addison-Wesley Publishing Company, Inc.

[MacKenzie, 1994] MacKenzie, D. (1994). Computer related accidental death: an empirical exploration. *Science and Public Policy*, 21(4):233–248.

[Newman and Lamming, 1995] Newman, W. M. and Lamming, M. G. (1995). *Interactive System Design*. Addison-Wesley.

[Norman, 1988] Norman, D. E. (1988). *The Psychology of Everyday Things*. Basic Book Inc.

[Tyldesdly, 1990] Tyldesdly, D. (1990). Employing usability engineering in the development of office products. In Preece and Keller, editors, *Human-Computer*

Interaction. Prentice Hall, Hemel Hempstead.