PARTE I

FUNDAMENTOS

CAPÍTULO 2

INTERACÇÃO HUMANO-COMPUTADOR

ÍNDICE

Prefácio.

A passagem definitiva do processamento diferido (ou "batch") ao processamento interactivo aconteceu com o aparecimento da chamada 3ª geração (1964-1971). Até então, a exploração interactiva de máquinas de 2ª geração concentrava-se num número mínimo de centros, em geral para aplicações de controlo e milita-res, sobrepondo-se os problemas de falta de eficiência e de fiabilidade dos siste-mas aos problemas relacionados com a sua falta de facilidade de utilização.

O aparecimento dos primeiros sistemas de "tempo repartido"[1], em meados dos anos 60, é assim o verdadeiro marco histórico da transição apontada. O período que se segue corresponde à primeira verdadeira "massificação" da utili-zação de computadores de forma interactiva. Ainda que estudos em áreas rela-cionadas com a IHC tenham sido feitos desde o aparecimento dos primeiros computadores, em particular devotados a questões de ergonomia, é no entanto após o aparecimento destes sistemas que surge verdadeiro interesse científico e tecnológico na área.

Historicamente, a primeira conferência na área realiza-se em 1966 (cf. [IBM 66]), as primeiras publicações periódicas surgem em 1969[2] e o primeiro livro em 1973 [Martin 73]. No entanto, a maior parte dos estudos então realizados e pu-blicados são dedicados quase exclusivamente à componente humana.

É na transição da 4ª para a 5ª geração, nos anos 80, com o aparecimento e proliferação dos computadores pessoais e estações de trabalho tecnologicamente desenvolvidas, que, verdadeiramente, a área de IHC assume importância crucial e se apresenta como uma disciplina autónoma, procurando os seus próprios métodos, técnicas e ferramentas, mas já como disciplina charneira entre as disciplinas de Factores Humanos e Engenharia da Programação.

Em síntese, enquanto o custo dos computadores foi sendo muito superior ao custo do utilizador, fosse este profissional ou não, os problemas relacionados com a facilidade da sua utilização foram sendo considerados secundários, sendo o objectivo principal do esforço científico de então a descoberta de técnicas que permitissem um maior aproveitamento dos recursos computacionais. Quando os custos da componente lógica superaram os custos da componente física, os esforços foram orientados de imediato para a procura de métodos, técnicas e ferramentas que pudessem trazer maior rigor e qualidade ao processo de con-cepção de aplicações. Não tendo sido ainda satisfatoriamente resolvido o proble-ma anterior, eis que a maioria dos utilizadores, conforme anteriores previsões, passa a ser não-profissional,

---

[1] *time-sharing*

[2] *International Journal on Man-Machine Studies* e *IEEE Trans. on Man-Machine Studies.*

exigindo tempos de aprendizagem muito curtos e rentabilização quase imediata do seu investimento em tecnologia computacio-nal. Passa-se assim de um investimento científico na exploração, para um inves-timento na concepção e implementação, de imediato seguido da necessidade de investimento na utilização. Sendo a qualidade de construção (no sentido da programação) de uma aplicação condição necessária, ainda que não suficiente, ao seu sucesso em termos de utilização, considera-se hoje em dia que só o seu real sucesso de utilização pode prescrever a sua efectiva qualidade de concepção e construção. A Interface com o Utilizador passa a ser o factor determinante do sucesso ou insucesso das aplicações interactivas.

Torna-se assim notória a dependência entre as duas fundamentais discipli-nas que suportam a área de IHC, designadamente, Factores Humanos e Enge-nharia dos Programas. Sente-se uma premente necessidade de, para além dos seus desenvolvimentos isolados visando a melhoria da qualidade dos Sistemas Interactivos, esforço cooperativo ser realizado, em consonância até com as actu-ais correntes científicas de multidisciplinaridade e transferência de conheci-mento e tecnologia entre áreas científicas.

O problema geral a resolver em IHC tem portanto, em essência, três entida-des envolvidas que devem ser estudadas segundo métodos próprios. Primeiro, o *utilizador*, humano, possuindo, intrinsecamente, percepção e inteligência, e consciente de objectivos a concretizar através de tarefas a realizar utilizando as capacidades que lhe são oferecidas pelo *sistema computacional interactivo*, a se-gunda entidade. A *Interface com o Utilizador* é a 3ª entidade, que, tal como a aplicação propriamente dita é antes de mais um produto software, deve ser ca-paz de estabelecer a mais fácil correspondência entre os objectivos do utilizador e a forma de os atingir. Os objectivos são expressos sob a forma de resultados das tarefas interactivas, sendo estas por sua vez expressas sob a forma de se-quências de comandos a executar ao nível da aplicação.

2.1.- Introdução.

J. C. Licklider[3], ainda que relativamente pouco referido, foi uma personalidade de génio da computação em geral, e um visionário da computação interactiva, pelo que a sua referência é, no mínimo, obrigatória numa tese relacionada com IHC e Sistemas Interactivos[4].

[3] n. 1915, = 1990.

[4] Aqui lhe seja prestada alguma da muita homenagem a que, ainda que postumamente, tem absoluto direito pelas inúmeras e brilhantes contribuições para as áreas de IHC e Ciências da Computação.

Historicamente, a ele se associam marcos da computação tão importantes como o desenvolvimento, nos anos 60, do primeiro sistema de *time-sharing*, construído sobre um Digital PDP-1, a direcção do projecto ARPA que, em 1965, deu origem ao aparecimento nos EUA dos primeiros programas de doutoramen-to em Ciências da Computação (designadamente em Berkeley, Carnegie-Mellon, MIT e Stanford) e, nos anos 70, às primeiras redes de computadores interacti-vos.

Em artigos publicados nos anos 60 e compilados em [Taylor 90], Licklider anteviu e sugeriu caminhos para desenvolvimentos na área de IHC. Destes, cerca de 35 anos depois, apenas alguns foram conseguidos. Em [Licklider 60], por exemplo, são lançadas as ideias base e analisados os requisitos para o de-senvolvimento de uma verdadeira simbiose humano-computador, em oposição à corrente então em vigor que via os sistemas humano-máquina como simples "extensões mecânicas do homem". Com o decorrer do tempo a "extensão mecâni-ca" acabou por dar lugar à quase completa substituição do homem em tais sis-temas, inversão tal que se pode afirmar ter passado o humano a ser um auxiliar do sistema, ou seja, uma mera "extensão humana das máquinas".

Em IHC pretende-se, de facto, que o computador seja uma "extensão mecâ-nica do homem", um seu auxiliar na realização de certas tarefas. Porém, e tendo em atenção a desproporção existente entre as capacidades cognitivas de um ser humano e de um programa de computador, os dois processadores em causa, o sonho de *simbiose* de Licklider não é ainda sequer verdadeira *interacção*[5] mas tão só *reacção*.

Em [Licklider 68] é igualmente visionada a possibilidade de usar computa-dores e redes como suporte ao "trabalho cooperativo"[6], à difusão de novidades, à partilha de informação e de conhecimento[7], bem como à construção de espaços comuns de diálogo, sendo de realçar o famoso, premonitório e preocupante pa-rágrafo, transcrito sem tradução,

> *"In a few years, men will be able to communicate more effectively through a machine than face to face."*

Por outro lado, e numa linha mais pragmática, no mesmo artigo define-se *comunicação interactiva* como consistindo de curtos "jactos" bidireccionais de diálogo (a metáfora, interessante ainda que não totalmente completa do ponto de vista do autor, é um jogo de "ping-pong") sendo a *obstrução*[8], ou seja,

---

[5] Interacção, no verdadeiro sentido do termo, implica naturalmente muito mais do que a simples troca de informação.

[6] cf. os actuais esforços em *Cooperative Software Work*, etc.

[7] cf. na actualidade, *Gopher*, *News*, *WWW*, *Internet*, etc.

[8] tradução do termo americano original *filibustering* equivalente ao termo *starvation* doutras áreas.

a não equilibrada posse do controlo do diálogo, considerada uma anomalia de comu-nicação a evitar (cf. monólogo vs. diálogo).

Finalmente, o mesmo artigo prevê igualmente que a interacção seja realiza-da, preferencialmente, com programas e modelos programados, e que seja reac-tiva e não competitiva, isto é, basicamente *assistêncial* no sentido do utilizador.

Esta visão do que poderá ser a interacção entre humanos e computadores, ainda que formulada a muitos anos de distância, parece ser, ainda hoje, extre-mamente correcta e pragmática e, como tal, até à apresentação de melhores vi-sões, comungada pelo autor.

Introduzidas estas considerações de contexto, é principal objectivo deste capítulo apresentar em síntese o real "estado da arte" na área de IHC, através da referência e análise das principais teorias, heurísticas, práticas e resultados tecnológicos conseguidos nas suas tão diversas disciplinas.

2.2.- IHC: Concepções como Disciplina.

Concepções alternativas da disciplina de IHC são, por vários autores, defendi-das e, em geral, aceites. Mas, por serem concepções, sofrem do problema de po-derem reflectir visões redutoras e unitárias, ainda que compensadas por uma maior coerência e completude. Por exemplo, uma concepção da IHC enquanto disciplina científica ou como disciplina de engenharia conduz necessariamente a perspectivas diferentes quanto ao seu modo de funcionamento, técnicas e objec-tivos.

Antes porém de se apresentarem tais concepções da área de IHC e realizar a sua análise, é importante que se defina o que deve ser entendido por *disciplina* e, neste caso particular, por *disciplina de IHC*. Segundo a maioria das definições correntes, para que uma *disciplina* possa nascer e ter existência é necessário que surja um *problema geral* a resolver, que sobre o mesmo se reuna *conheci-mento*, e *prática* seja realizada, quer tendo por base tal conhecimento quer vi-sando o enriquecimento deste. Assim, e em síntese, poder-se-á afirmar que:

Disciplina = Problema + Conhecimento + Prática.

O aumento do *conhecimento* numa disciplina é visto como o produto funda-mental da investigação realizada. Este conhecimento pode ser público (quando formalizado), privado (quando experimental) ou ainda codificado (quando ex-presso sob a forma de regras, princípios ou mesmo teorias). Conhecimento é assim, naturalmente, em qualquer das suas formas, uma característica crucial de qualquer disciplina.

A *prática* consiste, em geral, na aplicação do conhecimento adquirido na resolução do problema abordado na disciplina, ainda que por vezes tal

conheci-mento possa resultar de prática e observação anterior[9]. O *problema geral* a resolver é a característica que acaba por distinguir as diversas disciplinas. As disciplinas científicas abordam o problema científico geral da explicação funda-mentada e, em consequência, da previsibilidade ou réplica (cf. biologia, física, etc.). As disciplinas de engenharia, por outro lado, abordam o problema geral da concepção e produção.

Dada a generalidade intrínseca dos problemas, torna-se óbvio aceitar que dentro da mesma disciplina um ou mais subproblemas possam ser considera-dos, conduzindo à noção de *subdisciplina*. Torna-se assim imperioso considerar igualmente o *âmbito* de uma disciplina e, em relação a cada um dos problemas a abordar, a possibilidade de se considerarem pois *subdisciplinas*, resultantes da decomposição do problema geral (ou âmbito geral) em subproblemas (âmbitos ou contextos específicos).

Long e Dowell, em [Long e Dowell 89], propõem uma razoável definição do *âmbito* da disciplina de IHC, como sendo *"humanos e sistemas computacionais interactuando com o objectivo de realizarem trabalho de forma efectiva"*. Tal âmbito geral implica que humanos possam ser tomados como indivíduos ou até organizações, computadores como máquinas programáveis, individuais ou em rede. Tal tem igualmente implicações nas estruturas quer de humanos quer dos computadores, bem como no nível das suas interacções, mais físicas (cf. trans-formação de energia ou o seu controlo) ou mais abstractas (cf. transformação de informação). A *efectividade* corresponde ao grau de obediência aos requisitos.

O *problema* geral em IHC, caracterizado o seu âmbito, pode ser postulado como sendo o problema de *"conceber sistemas para uma interacção entre huma-nos e computadores visando a realização de trabalho efectivo"*. Assim, *concepção* pode considerar-se como o *problema da disciplina*, o que determina de imediato que os produtos finais sejam *concepções de sistemas interactivos*.

A prática em IHC procura soluções para este problema geral, subdividindo-o em dois subproblemas distintos e abordados igualmente em disciplinas distin-tas:

- *o problema da concepção de humanos interactuando com computadores;*
- *o problema da concepção de computadores interactuando com humanos;*

Em geral, estes dois subproblemas são abordados em disciplinas distintas. Os Factores Humanos (e a Ergonomia) abordam o problema de colocar humanos a interagir com computadores. A Engenharia da Programação aborda o proble-ma de conceber sistemas computacionais interactuantes com o ser humano.

Actualmente três perspectivas da área de IHC são defendidas:

---

[9] cf. *conhecimento heurístico.*

- IHC como *disciplina baseada na heurística*, logo artesanal;
- IHC como *disciplina de uma ciência aplicada*;
- IHC como *disciplina de engenharia*;

2.2.1.- IHC como Disciplina Heurística.

As disciplinas baseadas em heurísticas procuram solucionar o problema que abordam através de práticas de implementação e teste. O conhecimento assim acumulado é heurístico, logo implícito e informal. É suportado pela própria prá-tica, sendo formuladas heurísticas que são implementadas e testadas na práti-ca, sendo retidas as que, em tal contexto, conduziram a práticas bem sucedidas. Na área de IHC, projectos históricos como o VideoTexto da Prestel evoluiram usando esta metodologia [Gilligan e Long 84]. Porém, um dos maiores proble-mas que se encontram ao encarar IHC como uma disciplina heurística reside no facto de não ser efectiva, dado que as heurísticas não são operacionais, ou seja, sendo implícitas e informais não podem ser directamente usadas por quem não esteve directamente ligado ao processo da sua geração.

Esta reconhecida dificuldade na possibilidade de operacionalização de heu-rísticas é agravada pelo facto de a integração entre as áreas de Factores Hu-manos e de Engenharia da Programação envolver equipas diferentes, pelo que se torna adicionalmente complexa. Para além dos problemas de dificuldade de operacionalização e transmissão do conhecimento adquirido, a perspectiva heu-rística, não sendo testável, não garante que a prática baseada na heurística pos-sa assegurar os resultados pretendidos. Finalmente, este conhecimento não é, dadas as condições da sua obtenção, generalizável, o que relativiza o seu valor.

2.2.2.- IHC como Disciplina Científica.

IHC encarada como disciplina de uma ciência aplicada deverá usar conhecimen-to científico, sob a forma de teorias, modelos, leis, axiomas, hipóteses, etc., em suporte à prática de concepção. O conhecimento é científico, isto é, explícito e formal, podendo ser portanto provado ou, pelo menos, refutado. É ainda, ideal-mente, um conhecimento generalizável.

Disciplinas científicas deverão pois ser associadas à área dos Factores Humanos (cf. Psicologia, Linguística, etc.) bem como à área da Engenharia da Programação (cf. Ciências da Computação). O conhecimento de predição que pode ser herdado da Psicologia pode ser tornado de prescrição, formando con-juntos de "directivas"[10]. Porém, a maioria destas prescrições não são operacio-nalizáveis, testáveis e generalizáveis relativamente à concepção dos

[10] *guidelines.*

sistemas. Assim, a concepção de um sistema, de acordo com determinadas directivas, acaba por continuar a ser realizada segundo o ciclo *implementação-avaliação-iteração*.

O mesmo pode ser dito do conhecimento de predição que pode ser "importa-do" das Ciências da Computação para a concepção efectiva de sistemas interac-tivos. Muito deste conhecimento de predição suporta princípios de prescrição de Engenharia da Programação, tais como, por exemplo, modularidade, enca-psulamento, abstracção, extensibilidade, etc. Estes princípios gerais poderão ser usados como suporte a princípios mais específicos e adequados ao problema da concepção e construção de sistemas em IHC.

No entanto, e sendo indiscutível a importância e validade destes princípios, tal como acontece com o conhecimento oriundo da Psicologia, existe igualmente neste caso o problema da sua correcta transferência para a fase de concepção, onde questões de efectividade de utilização se levantam.

Assim, e ainda que por razões que se prendem com o carácter intrinsecamente explanatório e de predição do conhecimento gerado, esta concepção de IHC como disciplina de uma ciência aplicada apresenta várias limitações à sua aplicabilidade. Por outro lado, e ainda que os custos da geração de princípios possa ser baixo, o custo da aquisição de conhecimento é em geral elevado, tanto mais que as "directivas" geradas podem vir a mostrar não ser efectivas por se-rem demasiado específicas.

2.2.3.- IHC como Disciplina de Engenharia.

As disciplinas de engenharia abordam a resolução dos seus problemas, a concepção e a construção de produtos e sistemas, realizando especificações antes de realizar implementações. O conhecimento subjacente a uma disciplina de engenharia existe sob a forma de *princípios de engenharia*, sendo naturalmente conhecimento de prescrição. As concepções podem ser especificadas consoante as prescrições adequadas ao seu caso e, depois, as implementações verificadas relativamente ao que foi prescrito.

A concepção da IHC como disciplina de engenharia assume que o conheci-mento em IHC deva ser constituído por princípios de engenharia oriundos quer dos Factores Humanos quer da Engenharia da Programação, sendo a sua práti-ca, idealmente, baseada na especificação e posterior implementação das concep-ções. Estes princípios deverão ser formais e operacionais, resultando a sua ope-racionalidade do próprio facto de serem formais, bem como da própria forma-lização dos conceitos neles envolvidos.

As disciplinas principais de IHC, Factores Humanos e Engenharia da Programação podem, nesta concepção, possuir os seus próprios princípios e práticas. A disciplina dos Factores Humanos terá como problema fundamental a

es-pecificação e implementação de modelos do comportamento do utilizador. A En-genharia da Programação deverá encarregar-se da especificação e implementa-ção do sistema computacional que, em interacção com o utilizador, permita atingir a "performance" desejada.

Conforme salienta Thimbleby, em [Thimbleby 86], alguns autores, em parti-cular da área dos Factores Humanos, contestam esta concepção, considerando que não sendo o utilizador um sistema formal e tendo um comportamento não explicável em termos lógicos e rigorosos por ser demasiado não-determinista, não pode ser objecto de formalização.

Esta objecção resulta no entanto de uma compreensão errada dos objectivos da aplicação de métodos formais como auxiliares ao estudo da componente hu-mana. De facto, não existe a pretensão de "representar o utilizador", mas tão só, de forma abstracta mas rigorosa, fazer com que os que concebem sistemas in-teractivos prestem também atenção a certas necessidades dos utilizadores que foram já bem caracterizadas e podem ser consideradas universais.

Thimbleby introduz mesmo, em [Thimbleby 84], sem quaisquer complexos, os designados *gueps*[11], "princípios de engenharia" do utilizador, como meios de captura de princípios coloquialmente aceites como conducentes a maior facili-dade de utilização. Dos vários critérios impostos por Thimbleby sobre os poten-ciais *gueps*, salienta-se o requisito de poderem ser expressos formalmente, ain-da que devam ser facilmente comunicáveis.

A experiência mostra também, por outro lado, que o comportamento não-determinista do utilizador pode, perante regras bem definidas, "ser tornado" de-terminista, até certo grau, ou, pelo menos, previsível e controlável[12].

Na figura seguinte esquematiza-se esta concepção de IHC.

[11] cf. o título do próprio artigo.

[12] veja-se o exemplo do comportamento dos condutores perante as "regras" dos sinais luminosos.

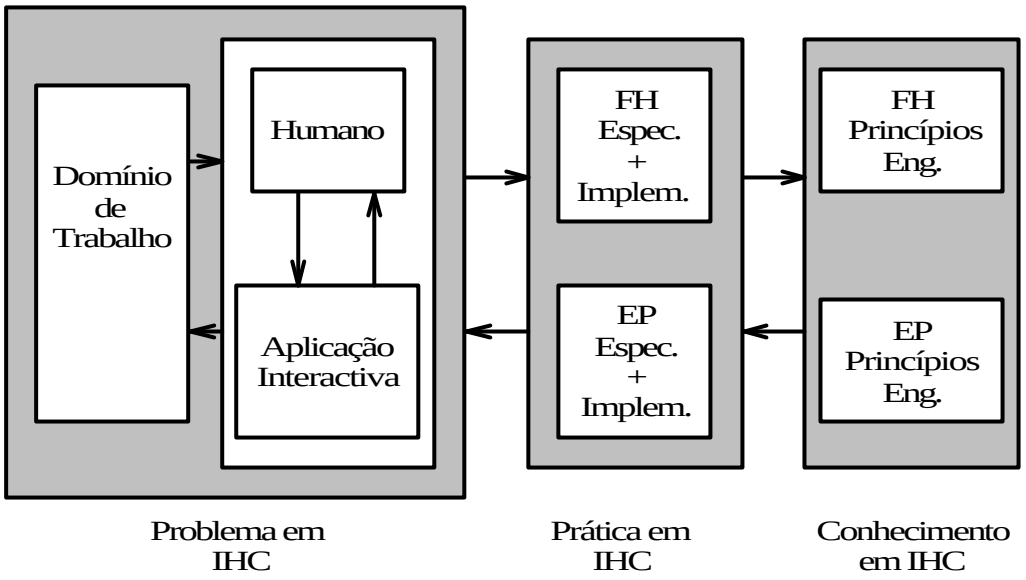

Fig. 2.1 - IHC como disciplina de Engenharia.

Para além das qualidades de conhecimento generalizável que os princípios de engenharia corporizam, da sua já referida operacionalidade e rigor, da possi-bilidade das suas subdisciplinas poderem evoluir segundo as suas próprias prá-ticas, a filosofia comum de *especificação e implementação* é favorável ao desen-volvimento de programas comuns e, principalmente, poderá constituir o tão pro-curado e necessário "elo de integração" das subdisciplinas.

No que diz respeito à componente de construção de sistemas interactivos, ou seja, IU e aplicação, é esta a concepção de IHC que se adopta e que sustenta conceptualmente a abordagem empregue, baseada na aplicação de métodos formais de especificação como ponto de partida para as implementações, com o apoio de técnicas de prototipagem, logo de concepção iterativa.

Esta complementaridade entre o emprego de métodos formais e a utilização de técnicas de prototipagem e desenho iterativo é, para o desenvolvimento de aplicações interactivas, de importância crucial. Os métodos formais têm sido tradicionalmente usados no desenvolvimento de programas sobre um conjunto fixo, ainda que por vezes incompleto, de requisitos funcionais. Mesmo neste contexto, e qualquer que seja a metodologia de desenvolvimento empregue, iteração e possibilidade de prototipagem rápida são auxiliares hoje em dia indispensáveis. Porém, reparemos que estamos a discutir o desenvolvimento de aplicações interactivas, ou seja, possuindo para além dos requisitos funcionais fixos e conhecidos à partida, requisitos de *facilidade de utilização*[13] dependentes do

[13] em inglês *usability* ou *ease of use*.

utilizador, não fixos e não conhecidos à partida, logo não formalizáveis. Torna-se assim evidente não ser viável considerar a aplicação dos métodos formais a todo o processo[14] pelo que, ainda que todo o processo possa ser visto numa perspectiva de engenharia, a componente de Factores Humanos deve recorrer a uma outra abordagem, parecendo no entanto imprescindível que esta se baseie também em iteração e prototipagem, recorrendo até a conhecimento heurístico resultante de anteriores experiências.

Definitiva e crucial parece ser, no entanto, a consideração de que o desen-volvimento de aplicações interactivas deixa de ter como simples objectivo a pro-dução de um produto a partir de requisitos fixos, mas antes a muito mais exi-gente meta de produzir um produto capaz de modificar positivamente os proces-sos de trabalho dos humanos através da sua utilização.

Em resultado, o ciclo de desenvolvimento de aplicações deve ser repensado, passando a interligar os processos de construção, divididos em diversos ciclos de reconcepção, reprototipagem, reimplementação e reavaliação, com os proces-sos de teste de utilização e de avaliação da facilidade desta. Sem técnicas e fer-ramentas que facilitem a prototipagem rápida de ambas as componentes, inter-activa e funcional, a sua interligação ainda a nível de protótipos, ou seja, sem que ainda qualquer delas tenha atingido o desenvolvimento definitivo, e sem um método que garanta a sistematização do processo, os resultados ficarão sempre aquém das expectativas dos utilizadores.

2.3.- Sistemas Computacionais Interactivos.

Ponderados que foram já os problemas, as expectativas, as limitações e as dife-rentes perspectivas inerentes à área de IHC, parece importante que se tenha agora em conta um determinado enquadramento geral ou modelo do fenómeno interactivo, relativamente ao qual um domínio verbal e conceptual possa ser definido e fixado. Comecemos no entanto por tentar uma melhor caracterização dos sistema interactivos dentro de uma perspectiva sistémica.

Um *sistema computacional interactivo* pressupõe a existência de, pelo menos, um utilizador humano e de um sistema computacional. Passaremos a chamar, genericamente, ao primeiro *utilizador*. O segundo é um sistema computacional do qual, por abstracção, nos interessa aqui fundamentalmente considerar a componente lógica, ou seja, a *aplicação*, construída por forma a que a sua fun-cionalidade possa ser acedida em *modo interactivo* (por oposição

[14] Ainda que não seja viável formalizar o subjectivo, determinados princípios de engenharia de interacção podem naturalmente ser formalizados, permitindo a realização de provas de satisfação dos mesmos.

ao *modo batch*, ainda que este não tenha que ser excluído nem o deva ser até em muitos casos).

Porque destes sistemas computacionais interactivos nos interessa centrar a atenção primordialmente na camada "software", devíamos passar a designá-los por *sistemas "software" interactivos*, assumindo-se desde já que a interacção a considerar se realiza entre um *utilizador humano* e a componente aplicacional do sistema, a *aplicação*. Outras interacções poderiam no entanto ser consideradas bastando para tal alargar os tipos de fontes produtoras de "input" para a aplica-ção.

O sentido fundamental da implementação sobre computadores de sistemas "software" interactivos (*sistemas interactivos ou aplicações interactivas*), quer se-jam de edição de texto, de cálculo, gráficos ou de processamento de dados, re-sulta da necessidade de automatizar tarefas e procedimentos que, de outro mo-do, seriam realizados de forma manual, mecânica ou por outro meio tecnoló-gico, possivelmente não tão eficiente como a tecnologia de computação hoje dis-ponível. No entanto, e independentemente da solução adoptada, características intrínsecas à normal actividade humana permanecem em essência, podendo porém variar quanto à forma e quanto ao processo.

De facto, qualquer actividade humana tem por motivação a necessidade de obtenção de um conjunto de *objectivos* num determinado *domínio*, entendido genericamente como uma área de perícia ou de conhecimento na qual se desen-volve a actividade. Ainda neste contexto geral da actividade humana, *tarefas* surgem como operações a realizar, sob a forma de manipulações sobre as enti-dades constituintes dos domínios, visando a obtenção dos objectivos ou *resulta-dos*.

Um sistema "software", interactivo ou não, começa por ser uma transposi-ção (uma representação ou modelo) das entidades constituintes de um domínio, em certos casos previamente existente e tangível, noutras entidades que poderão nunca ter uma existência real, a menos da sua existência enquanto representa-ções simbólicas em computador. Em qualquer dos casos, e aparte a diferente análise que poderia realizar-se para cada caso, designaremos por *domínio da aplicação* a este domínio "construído", para que se possa distinguir do domínio real, possivelmente pré-existente e assimilado. Trata-se portanto, agora, de ga-rantir uma correspondência o mais aproximada possível entre tais domínios, por forma a que o conhecimento que o utilizador do sistema interactivo possa pos-suir do domínio real não seja obstáculo e, antes pelo contrário, o auxilie na compreensão e manipulação do domínio virtual. Deste modo, ser-lhe-á facilitada a satisfação dos mesmos *objectivos* embora por outros meios e processos, e sob outras formas.

No caso de sistemas "software" interactivos, em que as aplicações visam su-bstituir ou automatizar domínios ou sistemas de informação previamente

exis-tentes, é de notar que aos objectivos intrínsecos da actividade se podem adicio-nar novos objectivos resultantes da busca de uma correcta e acessível activida-de de manipulação do sistema virtualizado. Devemos pois distinguir entre os objectivos para a actividade no domínio real e os objectivos para a actividade so-bre a parte simulada da realidade, devendo idealmente os segundos ser auxili-ares de "mais-valia", até então inexistentes, para a obtenção dos primeiros.

O que nunca deverá ser esquecido é que, conforme se afirmou na secção anterior, o "software" é desenvolvido com o único propósito de apoiar os huma-nos nas suas tarefas ou lazeres. Ainda que o tipo de suporte a tais tarefas possa assumir várias e diferentes formas, o princípio comum às soluções "software" é o da utilização de um modelo da actividade humana sob a forma de informação e de processos sobre essa informação.

Num sistema interactivo haverá sempre, portanto, que distinguir entre a componente puramente computacional e a componente humana, tendo em atenção que a *interacção*, enquanto processo de comunicação que possibilita a um ser humano, utilizador de um sistema computacional, obter deste, por ma-nipulação, determinados resultados dentro de uma gama possível, não deve ser um obstáculo mas antes um auxiliar.

No âmbito em que nesta tese se pretende discutir *interacção*, apenas as acti-vidades, tarefas e objectivos dos utilizadores directamente relacionados com o sistema "software" interactivo são considerados. Tal é, igualmente, o grau de abstracção da maioria dos modelos de interacção [Dix 91].

2.4.- Referenciais de Interacção.

Apesar de o fenómeno de *interacção* entre um utilizador e um sistema "software" ser ainda actualmente um processo com características mais de procura de coo-peração do que de real e equilibrada interacção, encontrar-lhe uma definição precisa e englobante é igualmente complexo.

Nesta secção pretende-se, para além de rever alguns dos mais importantes referenciais ou enquadramentos de interacção[15] propostos, apresentar o *referen-cial de interacção* que nesta tese é adoptado, e no contexto do qual, sempre que necessário, se introduzem e abordam algumas das principais características inerentes ao modelo de concepção de sistemas interactivos nesta mesma tese proposto.

Conforme sublinha Norman, em [Norman 84], um *enquadramento de interac-ção* não pretende revelar novidades sobre o fenómeno interactivo, mas tão só estabelecer os estágios fundamentais do processo, facilitando assim a análise dos requisitos, propriedades, características e influências que cada

[15] em inglês *interaction frameworks*.

um dos es-tágios definidos transporta para o problema mais global da concepção de siste-mas interactivos.

Neste sentido, estes referenciais são em geral representações sistémicas de sistemas interactivos, ou seja, representam com maior ou menor grau de deta-lhe ou de abstracção todas as componentes de um sistema interactivo e as suas interligações funcionais.

Adicionalmente, um referencial deste tipo permite mais facilmente encontrar uma taxonomia classificativa para os inúmeros modelos, notações e formalis-mos, de origem multidisciplinar, existentes na área e que nos propomos, de for-ma representativa, sintetizar.

2.4.1.- Ciclo Execução/Avaliação de Norman.

O referencial de interacção proposto por Norman em [Norman 84], e designado por *Ciclo Execução/Avaliação*[16], identificava quatro fases fundamentais no pro-cesso interactivo, nomeadamente: *intenção*, *selecção*, *execução* e *avaliação*. Estas quatro fases foram em trabalho posterior [Norman 86] reduzidas para duas, *exe-cução* e *avaliação*, divididas em sete estágios sequenciais e cíclicos de inter-acção, que representam as diferentes actividades operacionais e cognitivas que o utilizador deve desenvolver durante a interacção. Estas diferentes fases são re-presentadas na figura 2.2.

Como se pode ver pela figura, estabelecido pelo utilizador um *objectivo* de tarefa, o mesmo é traduzido numa *intenção* formulada. De seguida, o utilizador tem que traduzir a sua intenção numa *sequência de acções* interactivas. Estas acções são de seguida executadas recorrendo à utilização dos meios disponibi-lizados pela *interface com o utilizador*. Termina aqui a fase designada por *fase de execução* e inicia-se a fase designada por *avaliação*.

A partir deste estágio, e segundo este modelo, o utilizador deixa de ter uma atitude operacional e passa a ter uma atitude perceptiva e cognitiva.

[16] *Evaluation/Execution Cycle* no original.

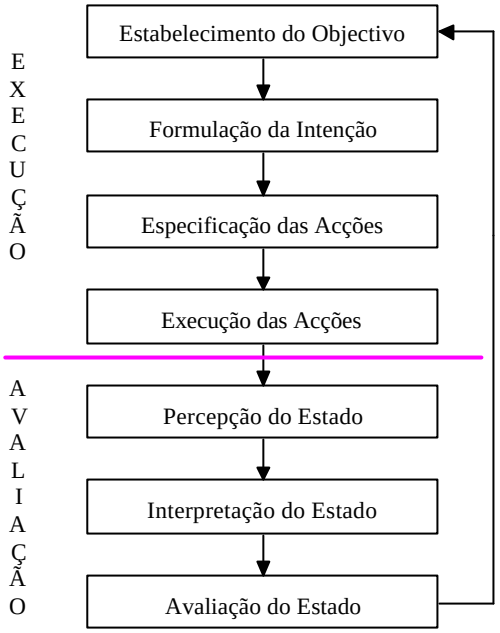

Fig. 2.2 - Ciclo Execução - Avaliação de Norman.

Assim, o utilizador começa por ter a *percepção* do resultado imediato da sua *execução*, de seguida realiza a *interpretação do estado* resultante da *especificação da sua sequência de acções*, e, finalmente, faz a *avaliação do estado do sistema* relativamente aos *objectivos estabelecidos*. A simetria entre a fase de *execução* e a fase de *avaliação* é sublinhada na figura 2.3.

Ainda que substancialmente incompleto, por considerar que o fenómeno de interacção com um sistema interactivo termina na IU, abstraindo assim da *com-ponente de construção* e concentrando-se principalmente na *componente cogni-tiva-comportamental*, o modelo de Norman é visto por muitos autores como um modelo didáctico e obrigatório, enquanto forma simples de instruir os que de-senvolvem a componente de construção sobre os factores psicológicos e cogniti-vos da interacção, tão importantes para a facilidade de utilização dos sistemas interactivos.

O modelo é, por outro lado, considerado igualmente um referencial impor-tante na área da psicologia da interacção por, deste ponto de vista, a ter dividi-do nas fases essenciais de *formulação de plano* e *observação de resultados*.

A figura 2.3 permite observar com detalhe estas duas fases e suas respecti-vas subfases. A *formulação do plano* envolve as subfases de formulação da in-tenção, especificação da acção e execução, cada uma delas com características cognitivas próprias. A obtenção de resultados passa por uma primeira subfase de percepção, seguida de uma interpretação dos resultados, a que se segue uma avaliação dos objectivos realmente atingidos e comparação com os esperados.

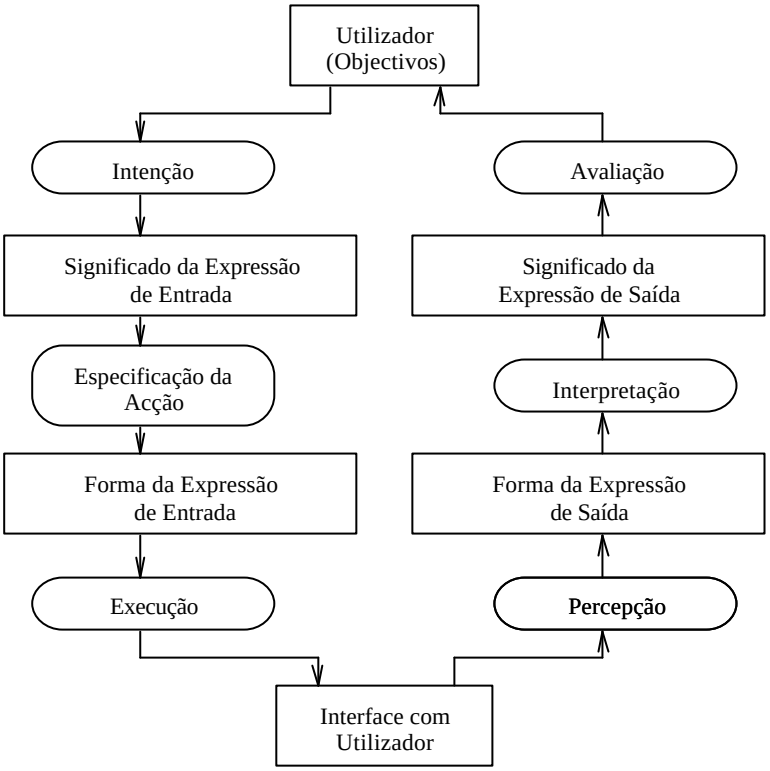

Fig. 2.3 - Ciclo de Norman - II.

De facto, a partir do modelo de Norman o trabalho de investigação na área da pesquisa formal em psicologia da interacção humano-computador passou a ser dividido em duas grandes subáreas, a primeira abordando os aspectos co-gnitivos da formulação e da execução de planos e a segunda os aspectos cogniti-vos da percepção e avaliação dos resultados obtidos [Butler et al. 89].

2.4.2.- Referencial de Barnard e Harrison.

Barnard e Harrison procuraram posteriormente estabelecer uma "ponte" entre as áreas, baseada na criação de modelos explícitos do utilizador, do sistema e da IU [Barnard e Harrison 89]. Não se tratando, na génese, de um referencial de interacção, o modelo visa compatibilizar propostas oriundas de áreas diferentes (centradas no utilizador e centradas no sistema) procurando interligá-las e compatibilizá-las, em resultado da observação das comuns incompatibilidades entre quem estuda e define modelos do utilizador e quem desenvolve modelos do sistema, e dos desvios resultantes de assunções erróneas de uns relativamente aos outros. Barnard e Harrison introduzem neste trabalho a possibilidade de se criarem modelos, do utilizador e da aplicação, possuindo a característica co-mum de se basearem em máquinas

de transição de estados. Os estados do utili-zador são vistos como *estados cognitivos* enquanto que os estados da aplicação como *estados da máquina*[17]. Uma representação diagramática deste modelo é apresentada na figura 2.4.

Qualquer interacção entre o utilizador e o sistema tem por resultado uma sequência de transições de estado em ambos. A correspondência bidireccional, salientada na figura, entre estados e modelos significa que os modelos não se limitam a definir as transições de estado. De facto, podem ser introduzidas abs-tracções nos modelos (cf. tipicamente *tarefas* no modelo do utilizador e *ciclos* no modelo da aplicação) baseadas nas transições de estado.

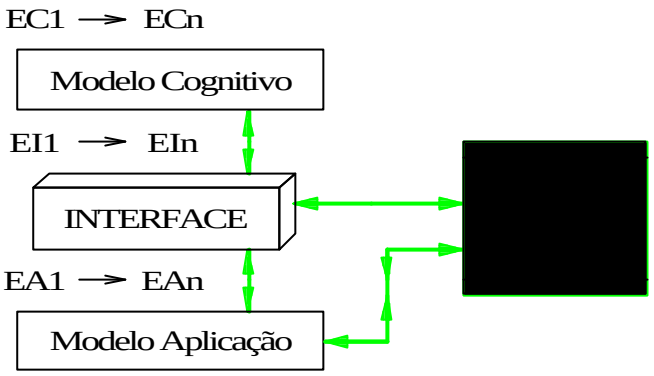

Fig. 2.4 - Modelo de Barnard e Harrison.

A inexistência de uma ligação explícita e clara entre os dois tipos de transi-ções e as abstracções introduzidas nos modelos levou à sugestão de um terceiro modelo, o *modelo de interacção*, tendo por base a ideia de que qualquer relação entre conceitos de modelação do utilizador e do sistema devem ser realizados no seu "ponto de encontro", i.é., ao nível da IU. O modelo de interacção é igualmen-te baseado em transições de estados, designados, neste caso, por *estados de in-teracção*.

Em [Finlay et al. 91] foram acrescentados *eventos* ao modelo anterior, sob a forma de etiquetas de operações ou de transições de estado, assumindo o papel de elementos de ligação entre o modelador da aplicação e o modelador do utilizador. Ao nível da interface com o utilizador, o mesmo evento pode ser rela-cionado com operações quer sobre o estado do sistema quer sobre o estado co-gnitivo do utilizador.

Em [Barnard e Harrison 91] alguns esforços são realizados no sentido de colmatar o reconhecido problema da uniformização da descrição dos modelos, em particular as dificuldades de formalização do modelo do utilizador. O proble-ma é, no entanto, de tal grau de dificuldade que, ainda que tais esforços mere-çam referência, a verdade é que rapidamente se tornam

[17] cf. *machine states* no original.

obsoletos quando somos confrontados com a sua aplicabilidade em IU de muito maior grau de complexi-dade, por serem, por exemplo, não sequenciais. Modelos como estes devem ser vistos fundamentalmente como modelos de análise, quer do fenómeno de inter-acção em si quer de requisitos, e não como modelos de concepção ou de imple-mentação, pois deixam em aberto uma enorme lacuna não superável pela tecno-logia e metodologia actualmente ao dispor.

2.4.3.- Modelos Abstractos: PIE e REDPIE.

O princípio da concepção de Sistemas Interactivos "centrada no utilizador"[18] advoga que o construtor do sistema tente estar permanentemente ajustado à percepção que o utilizador, a cada instante, possa ter da interacção. Em geral mergulhado nos inúmeros detalhes da implementação, torna-se evidentemente difícil ao construtor abstrair dos mesmos e assumir uma perspectiva mais próxi-ma do utilizador, ou seja, mais observacional.

A visão que o utilizador tem do sistema pode, em abstracto, considerar-se uma visão do tipo "caixa preta"[19]. O utilizador observa o sistema como sendo uma "máquina" desconhecida que transforma as suas entradas nos resultados que lhe são apresentados. Com base nesta perspectiva simples que o utilizador tem do sistema, modelos abstractos do tipo "caixa preta"[20] foram pois desenvol-vidos, não só para servirem de auxiliares aos construtores de sistemas interacti-vos mas também para o estudo de propriedades particulares de determinados tipos de sistemas interactivos, como por exemplo, editores de texto, gestores de "janelas" e até pequenos ambientes de programação [Dix et al. 86] [Dix e Harrison 86].

O clássico exemplo de modelos "caixa preta" é o modelo *PIE*, inicialmente apresentado em [Dix e Runciman 85] e desenvolvido posteriormente em [Dix 87] até dar origem ao mais expressivo modelo designado por *REDPIE* que se apre-senta na figura 2.5.

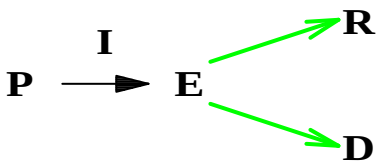

Fig. 2.5 - O modelo REDPIE.

[18] *user-centered design.*

[19] *black-box.*

[20] *black-box models.*

O modelo considera as entradas do utilizador como sendo originadas a par-tir de um conjunto de programas *P*, entendidos genericamente como sendo des- de toques de teclas a comandos. Os resultados são representados pelo conjunto *E* de efeitos resultantes, encarados igualmente de forma genérica. O sistema é completamente modelado por uma função de interpretação *I* que estabelece a correspondência entre programas e seus efeitos,

$$I : P \rightarrow E$$

O modelo *PIE* foi posteriormente refinado no modelo *REDPIE* que passou a distinguir dois subtipos no espaços de efeitos: o *"display"* e o *resultado*. O resul-tado *R* representa os efeitos correspondentes aos objectivos da interacção, desi-gnadamente as alterações de estado que ocorrem na camada computacional e que produzem os resultados. O "display" *D* representa o subconjunto dos efeitos de carácter efémero, como sejam, por exemplo, os que correspondem a altera-ções do ecrã.

A flexibilidade definicional dos espaços de *programas* e *efeitos* permite que o modelo possa ser aplicado com diferentes graus de generalidade e abstracção. Sendo principalmente um modelo abstracto da interface, serviu de base ao apa-recimento dos primeiros trabalhos de análise de propriedades de interacção usando notações formais [Sufrin e He 90].

Os modelos *PIE* e *REDPIE* são bons exemplos de como modelos abstractos podem ser usados na formalização de propriedades de interacção que são inde-pendentes do domínio do problema e da implementação. Porém, dado o seu ní-vel de abstracção, tais modelos não são construtivos, isto é, não fornecem qual-quer suporte que permita ao construtor derivar uma implementação a partir de uma especificação inicial.

Estes modelos da IU, de carácter fundamentalmente analítico, são igual-mente importantes pois podem considerar-se como os primeiros elos de ligação entre os Factores Humanos e a Engenharia da Programação, tendo permitido formalizar princípios básicos de facilidade de utilização dos Sistemas Interacti-vos, tal como os mesmos devem ser observados ao nível da IU.

2.4.4.- O Referencial de Abowd.

O referencial de interacção que nesta tese é adoptado foi apresentado pela primeira vez em [Abowd e Beale 91], constituindo-se como uma extensão aos modelos anteriores, e considerando quatro componentes fundamentais, designa-damente: *utilizador*, *aplicação*, *entrada* e *saída*. A figura 2.6 ilustra as relações que se estabelecem entre estas entidades, e as *distâncias* de comunicação entre elas.

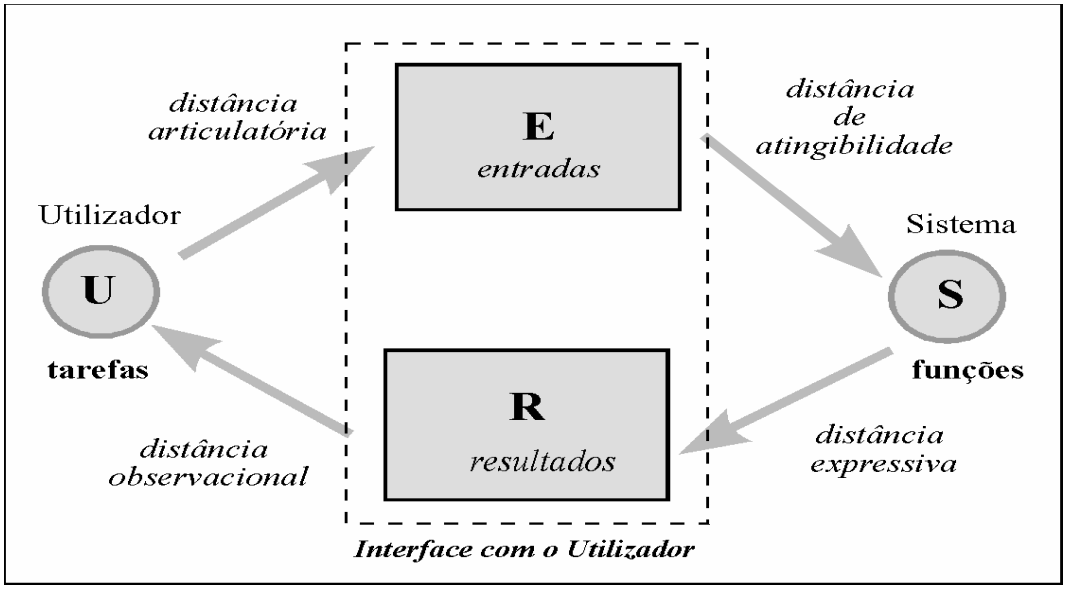

Fig. 2.6 - Referencial de Interacção de Abowd.

De base estrutural (i.é. procurando definir componentes e sua interligação) mas fortemente linguístico (i.é. vendo relações entre componentes sob uma pers-pectiva de comunicação), o modelo, que herda algumas características dos ante-riores, assume que cada componente possui a sua própria linguagem, constitu-indo estas os respectivos veículos de comunicação com os outros componentes. O utilizador usa uma linguagem de expressão de tarefas - *task language* - que é igualmente a sua linguagem para a expressão mental dos seus objectivos. As entradas (*inputs*) são expressas numa linguagem de entrada (*input language*) muito dependente dos dispositivos de entrada disponíveis (cf. "joystick", *rato*, "touch screen", botões, etc.). Os resultados (*outputs*) exprimem-se também numa linguagem de resultados (*output language*) igualmente dependente dos disposi-tivos empregues (cf. ecrã, som, etc.). O sistema, ou aplicação, reconhece uma linguagem da aplicação (*core language*) exprimindo a sua funcionalidade e as respectivas variáveis de estado. As linguagens de entrada e de saída são elemen-tos constituintes da designada Interface com o Utilizador (IU).

Neste modelo, a interacção entre um utilizador e um sistema consiste funda-mentalmente num ciclo iniciado no utilizador, seguindo o modelo de *execução-avaliação* de Norman, anteriormente introduzido. Em cada passo do ciclo, ocorre uma tradução entre linguagens, resultante da necessária comunicação entre dois dos componentes do sistema. A formulação de um objectivo é feito pelo uti-lizador usando a linguagem de tarefas. De seguida, esta formulação deve ser transcrita para a linguagem de entrada para que tal objectivo seja comunicado ao sistema interactivo. À maior ou menor dificuldade existente nesta formulação de objectivos em termos da linguagem de entrada dá-se o nome de *distância ar-ticulatória.* É pois uma medida da dificuldade do utilizador em exprimir na lin-guagem de entrada as tarefas

que, na linguagem de tarefas, são necessárias pa-ra atingir os objectivos pretendidos.

A *distância de atingibilidade* pretende medir a discrepância entre o conjunto de estados disponíveis do sistema e os que podem ser atingidos utilizando a linguagem de entrada. Por exemplo, um editor de texto pode permitir, através de funções disponíveis na sua camada aplicativa ou computacional, operações de "scrolling" contínuo, enquanto que a linguagem de entrada pode apenas permitir toques de tecla que correspondem a "scrolling" discreto.

A *distância expressiva* é uma medida do grau de diferenciação que a lingua-gem de saída fornece enquanto veículo de transmissão dos vários possíveis esta-dos do sistema. Em geral, o utilizador terá sempre que realizar uma interpreta-ção destes resultados, expressos na linguagem de saída, para que possa deter-minar se os seus objectivos foram ou não conseguidos. Um exemplo clássico de possibilidade de diminuição da distância expressiva foi o aparecimento, viabili-zado pelo desenvolvimento tecnológico ao nível dos dispositivos de saída, dos editores de texto do tipo WYSIWYG (*"what you see is what you get"*), nos quais o estado do sistema, neste caso a estrutura e formatação actual do texto que se encontra em edição, é fiel e imediatamente reproduzida no dispositivo de saída.

A *distância observacional* mede o grau de facilidade com que o utilizador in-terpreta o que, conforme os resultados apresentados na saída, aconteceu no sis-tema em função e resposta à sua actuação. Em princípio, a diminuição da dis-tância expressiva implicará uma menor distância observacional.

Num sistema interactivo ideal tais distâncias deveriam ser nulas, i.é., qual-quer natural formulação de objectivos do utilizador seria suficiente para a reali-zação das tarefas conducentes à concretização dos mesmos. O utilizador, tal co-mo no mundo real, estaria a interactuar com a tarefa a executar e não com uma interface para tal tarefa. Não sendo tal possível com a tecnologia actual, resta procurar que a IU não seja um obstáculo à concretização dos objectivos e à rea-lização das tarefas. Para tal é necessário reduzir o mais possível cada uma das distâncias enunciadas, por forma a que a distância entre a *semântica da inten-ção* (i.é. a formulação e expressão dos objectivos) e a *semântica do resultado*[21] (i.é., o julgamento ou avaliação do atingido segundo a expressão do sistema) se-ja mínima. Em geral, a esta distância entre as intenções formuladas mentalmen-te na linguagem de tarefas e expressas na linguagem de entrada, e as interpre-tações dos

[21] Em inglês esta distância é habitualmente referida como *semantics of evaluation*. Parece no entanto ao autor mais óbvio que, por simetria com a noção de *semântica da intenção*, esta seja designada por *semântica do resultado*, remetento para um processo mental extrínseco ao modelo, tal como o que origina a formulação de objectivos, neste caso a avaliação dos resultados obtidos.

resultados apresentados na linguagem de saída, designa-se por *dis-tância semântica.*

Qualquer sistema interactivo deverá ter como objectivo garantir que esta distância semântica seja nula (o que é impossível) ou, pelo menos, mínima. A figura 2.7 representa as quatro fases principais do processo interactivo conside-radas neste referencial: *articulação*, *execução*, *apresentação* e *observação*.

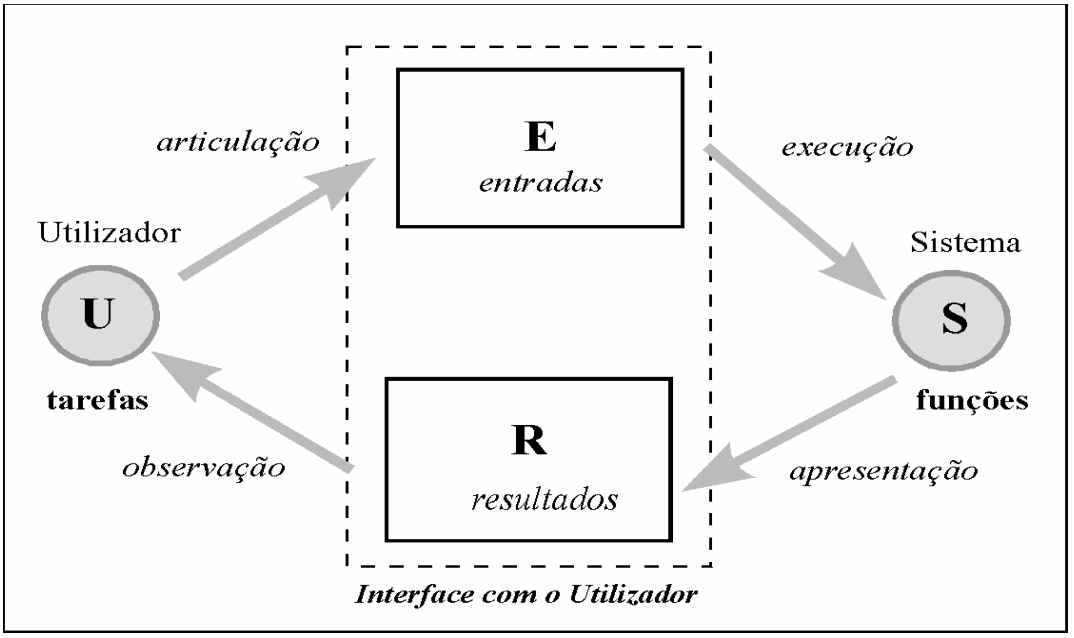

Fig. 2.7 - Modelo de Abowd - II.

Estas fases são, como se verifica, idênticas às fases propostas pelos modelos de Norman.

Sistemas interactivos multimédia e multi-modais podem ser vistos como sistemas nos quais, através de uma abordagem com adequado suporte tecnoló-gico, se enriqueceu a comunicação entre utilizador e sistema, por incorporação de outros tipos de representação de informação e outros meios de comunicação, com o objectivo geral de diminuir a distância semântica pela suposta diminui-ção das distâncias articulatória e observacional.

2.5.- Modelos de Interacção.

2.5.1.- Classificação.

Sendo a área de IHC multi-disciplinar, não poderá supreender que o número de modelos existentes seja elevado e que o seu âmbito seja tão díspar. De facto, existem modelos disponíveis para representar praticamente todas as entidades presentes num sistema interactivo. Modelos do e para o utilizador,

das suas tarefas, do auxílio que lhe deve ser dado, do diálogo que com este deve ser estabelecido, modelos para os que concebem a interacção, modelos de arquitec-tura, etc.

A tabela 2.1 apresenta uma amostragem significativa dos muitos modelos habitualmente referenciados em diferentes disciplinas ligadas a IHC.

Modelos	*De Tarefas*	
	Do Utilizador	
	Do Diálogo	*De Processamento*
	Da Aplicação ----------	
	Da Adaptação	*Do Domínio*
	Da Concepção	
	Do Sistema	
	De Arquitectura	
	Da Organização	
	Ergonómicos	
	De Auxílio ("Help")	*Auxílio ao Diálogo*
	De Ensino	*Ensino do Diálogo*
	De Meta-Diálogo ----------	*Adaptação ao Diálogo*
	De Avaliação	*Concepção do Diálogo*

Tab. 2.1 - Modelos em IHC.

O objectivo desta secção é a apresentação de um estudo sintético, basica-mente com preocupações de classificação sistemática, dos diversos modelos existentes para a análise, sob diferentes pontos de vista, dos vários fenómenos e componentes de sistemas interactivos, desde a componente cognitiva do utili-zador até à arquitectura interna da implementação, estudo esse que ajude a ca-talogar os diversos modelos propostos.

Neste sentido, a definição de *modelo* aqui adoptada é lata e resulta da reu-nião das definições propostas por [Rohr e Tauber 84] e [Oberquelle 84], sinteti-záveis nas seguintes características:

- uma *representação*, ou *descrição*, comunicável, ...
- ... as *propriedades relevantes* de uma entidade ou objecto real, ...
- ... *a um dado nível de abstracção*, ...
- ... *projectada sobre um dado meio* (p. ex. no cérebro, num papel, num programa, numa base de conhecimento, etc.).

Rohr e Tauber associam a um *modelo* uma relação de utilização descrita por **R(M, S, O)**, onde **M** é o modelo, **S** representa o sujeito a quem o modelo se destina (e/ou o possui) e **O** o objecto modelado. Nielsen, em [Nielsen 90], propõe que o par **S x O** seja visto como o *tipo* do modelo, considerando ainda duas ca-racterísticas adicionais:

- Existem sete tipos de objectos muito relevantes no desenvolvimento de um sistema interactivo, designadamente:
  - o utilizador (**U**);
  - o "desenhador" (**D**);
  - o sistema computacional (**C**);
  - o investigador de factores humanos (**R**);
  - a tarefa do utilizador (**T**);
  - o "mundo" no qual o utilizador realiza uma tarefa (**W**);
  - o manual do sistema computacional (**M**);
- Uma notação simples permite classificar cada um dos modelos, combi-nando os objectos anteriores, ainda que certas combinações sejam in-compatíveis.

Admitindo o tipo Objectos = { U, D, C, R, T, W, M }, Nielsen introduz ainda uma notação simples para a classificação de modelos, ou seja, para a definição do *tipo do modelo*, notação que se apresenta na tabela seguinte:

Tipo	**Significado**
AB	O modelo que *A* tem de *B*
A(BC) *ABC*	O modelo que *A* tem do modelo *BC*, que é o modelo que *B* tem de *C*
A+B	Modelo que representa uma combinação dos objectos *A* e *B*
A(B+C)	O modelo que *A* tem da combinação *B+C*

Tab. 2.2 - Tipos dos Modelos.

Para além desta tipificação dos modelos, outros atributos lhes podem ser associados correspondendo a mais uma dimensão de classificação. Segundo esta dimensão, os modelos podem ainda ser considerados :

- *Interiorizáveis* ou *Exteriorizáveis*.

Interiorizáveis são os modelos que têm existência no interior do desti-natário do mesmo, seja humano ou computador, e que nem sempre possuem uma representação comunicável.

Exteriorizáveis são os modelos que podem ser comunicados pois po-dem ser representados explicitamente numa determinada notação.

O critério não é disjunto.

- *Distribuídos* ou *Estruturais*.

  Distribuídos são os modelos que consistem de diferentes factos e ex-plicações acerca do objecto descrito, sem qualquer organização ou foco particular. Este critério é discreto.

  Estruturais são os modelos que são construídos e organizados em função de determinado princípio, podendo ainda assim ser exteriori-záveis ou não.

- *Genéricos* ou *Instanciados*, o que classifica o grau de refinamento de um modelo desenvolvido de forma genérica para uma situação particular, ou seja, o seu grau de especialização. É um critério contínuo.
- *Gerais* ou *Específicos*, o que procura caracterizar o grau de abrangência ou âmbito do modelo relativamente ao número de objectos descritos.
- *Descritivos, Analíticos* e de *Simulação*, o que procura caracterizar o grau crescente da sua formalização, desde os informais aos executáveis.

Assim, os modelos de interacção passam a ser, no mínimo, bidimensionais, sendo a sua primeira dimensão o seu tipo, ou seja a determinação do que mode-lam e para quem, e a sua segunda dimensão a "forma" como o fazem.

Propomo-nos aqui, usando este referencial de classificação, apresentar e classificar alguns dos mais relevantes modelos existentes em IHC. Organiza-se esta apresentação por classes de modelos segundo o objecto modelado, sendo considerados os seguintes objectos fundamentais: o utilizador, o sistema com-putacional e o diálogo. Referiremos ainda modelos para construção de IU.

2.5.2.- Modelos do Utilizador.

Modelos Cognitivos.

Os modelos do utilizador mais interessantes e familiares aos psicólogos que se dedicam ao estudo da IHC são, principalmente, os *modelos cognitivos do*

utiliza-dor. Estes modelos são, em geral, modelos do tipo ***RU***, por serem representações construídas pelos, e para os, investigadores, procurando emular partes relevan-tes do processo mental do utilizador durante o processo interactivo, com o ob-jectivo de mais facilmente compreender e prever o seu comportamento, a forma como aprende, as suas capacidades operacionais, etc. São em geral modelos ex-teriorizáveis, estruturais, genéricos e gerais, podendo ser ou não estáticos. Adicionalmente, visam sobretudo auxiliar no processo de análise e não no pro-cesso de construção.

Adicionando-lhes um atributo que represente o seu *propósito* mais específi-co, podemos considerar modelos cognitivos do utilizador com os propósitos par-ticulares que se apresentam na tabela seguinte, ainda que, em geral, mais do que um propósito possa ser referido.

Propósito do Modelo	*Análise da Actividade Cognitiva* *Análise do Conhecimento sobre as Tarefas no mundo real* *Análise das Estruturas Cognitivas do Utilizador* *Previsão de Complexidade*

Tab. 2.3 - Propósitos dos Modelos Cognitivos.

Quanto às características do utilizador, são em geral tidas em consideração nestes modelos, em maior ou menor grau, as que se apresentam na tabela se-guinte :

Características Utilizador	*Psico-motoras* *Possibilidades cognitivas* *Capacidade de Aprendizagem e Conhecimento adquirido* *Experiência* *"Performance"* *Articulação*

Tab. 2.4 - Características do Utilizador.

A maioria destes modelos cognitivos do utilizador visam sobretudo auxiliar a compreensão do processo de operacionalização, ou seja, de tradução do conhe-cimento em acção e expressão, i.é., *articulação interactiva*. O objectivo será sem-pre reduzir o não-determinismo intrínseco ao utilizador, pelo aumento do grau de previsibilidade dentro do contexto do que é modelado.

Em [Green et al. 87] estes modelos são ainda divididos em duas classes: *modelos de competência* e *modelos de "performance"*. Os *modelos de competência* permitem prever comportamento legal sem no entanto darem qualquer indicação operacional sobre como o mesmo pode ser conseguido, ou seja, sem ligação à fase articulatória. Os *modelos de "performance"*, para além de descreverem as necessárias sequências comportamentais, descrevem igualmente o que o utili-zador tem que saber para as realizar, bem como tal conhecimento é empregue na execução efectiva das tarefas.

Esta relação intrínseca, e forte, entre *conhecimento* e *execução das tarefas,* faz com que a maioria dos modelos cognitivos sejam referidos, em geral, como *modelos de tarefa*[22], pelo que será nessa qualidade que serão aqui apresentados.

Modelos de Tarefa.

Os *modelos de tarefa* são casos particulares dos modelos cognitivos anterior-mente apresentados. Um detalhado estudo comparativo dos mais significativos *modelos de tarefa* é apresentado em [Simon 88]. Aqui, dado o seu relativo inte-resse no contexto desta tese, sintetizam-se as características dos modelos mais conhecidos.

Dentro da subclasse destes modelos que visam analisar o conhecimento do utilizador sobre as actividades do mundo real, são de referir em particular CLG (*Command Language Grammar*) [Moran 81], TAKD (*Task Analysis for Knowledge Descriptions*) [Johnson et al. 85] e TKS (*Task Knowledge Structures*) [Johnson et al. 88], um desenvolvimento do TAKD.

CLG é um modelo de raiz linguística que usa quatro gramáticas distintas para descrever tarefas, semântica, sintaxe e léxico. As suas características hie-rárquicas e a tentativa de representar todo o sistema envolvido na interacção torna demasiado complexa a sua utilização e difíceis de compreender as liga-ções entre os diferentes níveis.

[22] *task models.*

TKS, é um desenvolvimento do TAKD, que se baseia na ideia de que uma en-tidade designada *task knowledge structure* (Tks) representará os diferentes tipos de conhecimentos necessários à realização de uma dada tarefa pelo utilizador. Dentro de cada Tks existem diferentes representações de conhecimento. Noções como *plano de actividade* e *papel* são pontos de partida para a definição dos *ob-jectivos* e *sub-objectivos* do utilizador. Relações entre Tks e um vocabulário de acções é igualmente disponibilizado. O método pressupõe ainda três níveis de refinamento muito complexos e não claramente formalizados. Ao primeiro nível Tks são refinadas para o GTM (*General Task Model*) onde se representam os *papéis*, *objectivos*, *acções*, *estratégias* e *definições de objectos* como uma série de "frames" de conhecimento. De seguida, o modelo GTM é transformado num mo-delo STM (*Specific Task Model*) também baseado em "frames" mas já com liga-ções à camada computacional. Finalmente, a terceira fase consiste na transfor-mação do modelo STM no modelo SIM (*Specific Interface Model*). O SIM represen-ta em abstracto a interacção. O modelo, ainda que abrangente, parece demasia-do complexo dado procurar incluir preocupações mais relacionadas com aspec-tos cognitivos e de expressão de conhecimentos, com questões ligadas à imple-mentação da IU.

Dos mais antigos e conhecidos *modelos de tarefas* para análise e previsão do comportamento do utilizador, salientam-se os modelos desenvolvidos por Card, Moran e Newell, designadamente o *Keystroke Level Model* (KLM) e o *Model Hu-man Processor* (MHP) [Card et al. 83].

O MHP é um modelo muito genérico, de grande grau de parametrização e pouco conhecimento representado, que possibilita a realização de previsões gerais sobre o tempo necessário à realização de certas tarefas, possibilitando a introdução de parâmetros tais como capacidade e velocidade cognitiva, capaci-dade de percepção e capacidade motora do constituinte cognitivo. Sendo gené-rico, o modelo permite a expressão e análise de um grande conjunto de fenóme-nos psicológicos associados ao processo interactivo, não apresentando porém capacidade para acomodar o tratamento de comportamento flexível e de erros. Ainda que contemplando as três áreas de análise preferencial, perceptiva, cogni-tiva e motora, a sua representação é, em geral, considerada superficial.

O KLM possui uma representação mínima do processo cognitivo. O modelo representa fundamentalmente acção com base num restrito conjunto de opera-dores psico-motores, tais como "toques de tecla" e "apontar" no contexto de editores de texto. O modelo não necessita de "operacionalizar" o conhecimento dado concentrar-se na previsão do tempo de execução a partir de uma sequên-cia de operações fornecidas. Parametrização relativamente a utilizadores inex-perientes e profissionais é igualmente concedida pelo modelo. Note-se no entan-to que as previsões são, nestes modelos, de carácter basicamente temporal.

Não sendo as previsões de comportamento dos utilizadores em termos tem-porais o principal objectivo do investigador, mas antes, por exemplo, a medida da complexidade das tarefas, então outros modelos deverão ser usados.

Os modelos que se apresentam a seguir são exemplos de uma classe de mo-delos que visam produzir previsões sobre alguns aspectos da complexidade da interacção. Duas subclasses são reconhecíveis dentro desta classe de modelos: os *modelos gramaticais*, tais como RFG (*Reisner´s Formal Grammar*) [Reisner 81a] e TAG (*Task-Action Grammar*) [Payne e Green 86], e os *modelos periciais*, tais como GOMS (*Goals, Operations, Methods and Selections*) [Card et al. 83], CCT (*Cognitive Complexity Theory*) [Kieras e Polson 85] e PUM (*Programmable User Model*) [Young e Whittington 90].

Os *modelos gramaticais* são, fundamentalmente, *modelos de competência*, concentrando-se na operacionalização do conhecimento recorrendo à utilização de gramáticas generativas. Deste modo, estes modelos podem gerar, para uma dada tarefa, todas as frases legais, encaradas como sequências de acções necessárias à sua realização. Uma produção corresponderá a uma regra de transformação de uma tarefa-objectivo na sequência de acções elementares a executar. Pelo facto de, com nível de abstracção maior ou menor, estes modelos fazerem referência à linguagem observável ao nível da interface, são muitas vezes, erroneamente, citados como modelos de especificação do diálogo. É no entanto de salientar que, em geral, estes modelos não suportam a descrição de comportamento flexível, ou seja, diferentes alternativas de expressão das tare-fas, nem comportamento de erro, limitando-se a analisar comportamento legal e rígido.

RFG usa a clássica *Backus-Naur Form* (BNF) para descrever tais regras e medir o grau de complexidade articulatória. No entanto, RFG realiza a análise e medida da complexidade apenas a nível sintáctico, o que é, num contexto em que se referem modelos cognitivos, insuficiente.

TAG é um dos modelos cognitivos de base gramatical mais importantes e referidos, dado ser considerado um dos modelos com possibilidades de poder vir a estabelecer a ligação entre Psicologia e IHC. Para além de preocupações de medida de complexidade, TAG aborda também as questões de *coerência* da lin-guagem de entrada, descrevendo a estrutura das tarefas e a sequência de ac-ções necessária à realização dessas tarefas.

Os *modelos periciais* operacionalizam o conhecimento através da representa-ção de funções cognitivas, procurando de algum modo construir representações da mente do utilizador baseadas em conhecidas técnicas de representação de conhecimento, sendo posteriormente tais representações usadas em análises diversas do processo cognitivo, em geral, porém, sem permitirem inferir razões de falha de *performance*.

2.5.3.- Modelos da Aplicação para o Utilizador.

Metáforas.

Em IHC, uma *metáfora* pode ser considerada um modelo simplificado do modelo conceptual do sistema, visando em geral uma unificação do conhecimento que facilite a compreensão do modelo do sistema recorrendo a analogias ou alego-rias. Trata-se de uma estratégia que visa empregar conceitos familiares aos humanos, introduzindo-os em conjunto com novos conceitos, no contexto da interacção. A grande vantagem das metáforas consiste em permitirem ao projectista explorar o vasto conhecimento que o utilizador possui do mundo real, por forma a facilitar-lhe a aprendizagem dos mecanismos de interacção.

Os modelos metafóricos podem ser do tipo ***MMC,*** quando são usados em manuais para descrever o modelo conceptual contido no mesmo, ou do tipo ***UUC,*** quando se trata de uma metáfora internalizada pelo utilizador. Em todo o caso, são "modelos de modelos", o que os coloca ao nível dos meta-modelos.

Um dos mais bem conhecidos e bem sucedidos modelos metafóricos é o de-signado *desktop model*[23]. Outros são no entanto igualmente bem sucedidos e comuns tais como os modelos da folha de cálculo (*spread-sheet model*), modelos de desenho e pintura (*paint & draw models*) e modelos de composição documen-tal [Carroll et al. 88].

Alguns autores apontam o facto de que modelos metafóricos não capturam completamente a funcionalidade de um sistema dado ser este diferente do sistema metafórico de referência. Sendo tal observação verdadeira, ela não é suficientemente forte para pôr em causa a utilidade dos modelos metafóricos. De facto, conforme é revelado no estudo de Sein e Bostrom em [Sein e Bostrom 89], os modelos metafóricos podem ser preciosos auxiliares do processo de aprendizagem, ainda que o efeito observável seja muitas vezes pequeno. O que se pode concluir é que os modelos metafóricos, ainda que úteis só por si, não dispensam a consideração dos modelos funcionais.

Vem a propósito referir que os mecanismos de suporte à interacção que se apresentarão no capítulo 5, designados *arquétipos de interacção*, foram desen-volvidos, e são apresentados, recorrendo à ideia metafórica de *interacção dirigida pela estrutura*, procurando capitalizar sobre a semelhança conceptual e funcio-nal encontrada com o bem mais generalizado e conhecido paradigma de *edição dirigida pela sintaxe*, instanciado nos hoje usuais *editores estruturados*.

O alto nível de abstracção recomendado na descrição do nível das tarefas e das metáforas, e até a possibilidade de reutilização de anteriores descrições, constituem requisitos que indiciam uma possível adequação da

[23] que, dada a exiguidade do espaço físico real, por vezes se designa por *airplane-seat model.*

aplicação de tipos abstractos de dados especificados algebricamente, em vez dos usuais mo-delos de base gramatical. Esta intuição sai reforçada pelo aparecimento relativa-mente recente de trabalhos defendendo esta perspectiva [Bass et al. 92].

Modelos Compostos.

Determinados modelos têm mais do que um objecto a ser modelado. São pois designados *modelos compostos*. A título de exemplo, alguns dos modelos cogniti-vos anteriormente referidos, tais como GOMS [Card et al. 83] e PUM [Young e Whittington 90], são modelos compostos do tipo ***R(U+C)*** pois constituem uma re-presentação, para o investigador, da interacção definida entre o utilizador e o sistema.

Já CCT [Kieras e Polson 85] é um modelo do tipo ***C(U+C)*** pois é um modelo em computador da interacção entre utilizador e sistema, destinado a realizar simulações relacionadas com medidas de complexidade.

2.5.4.- Modelos de Arquitectura "Software".

Especificação da IU.

Um formalismo de especificação de IU é um modelo da IU, tal como é visto pelo investigador ou pelo implementador. É um modelo ***RC*** ou ***DC***.

O formalismo de especificação, pelas suas capacidades, é genérico, exteriori-zável e, de preferência, estrutural. Uma especificação concreta, produzida usan-do o formalismo, é um modelo instanciado. Se, adicionalmente, a linguagem de especificação for executável, teremos mesmo um protótipo da IU, ou seja, um modelo de simulação.

SGIU e "Toolkits".

Os conhecidos Sistemas de Gestão da Interface com o Utilizador (SGIU)[24] [Pfaff 85], que se apresentam com mais detalhe na secção 2.6, são modelos do tipo ***CC*** ou ***DC***, pois são modelos em computador da IU, que é um subsistema computa-cional, ainda que se possam considerar igualmente

[24] do inglês *User Interface Management System* (UIMS).

modelos para o construtor da IU. São modelos genéricos pois representam classes de possíveis interfaces ainda que, por instanciação, possam produzir interfaces particulares.

Os "toolkits" [Myers 89a] são, por seu lado, modelos ***DC*** pois representam modelos de interacção destinados ao construtor da IU.

Modelos ou Arquitecturas de Implementação.

Modelos ou Arquitecturas de Implementação são modelos que visam facilitar a tarefa do implementador de sistemas interactivos fornecendo-lhe um referencial para o desenvolvimento.

Várias arquitecturas lógicas e funcionais e vários modelos "software" foram desenvolvidos com o objectivo específico de fornecerem orientações para a con-cepção de IU, em particular, no contexto dos SGIU. Sendo a separação da IU dos aspectos de computação específicos da camada computacional um dos prin-cipais objectivos da utilização de SGIU, objectivo genericamente válido, é impor-tante que se distingam diferentes tipos de *separação* para análise, e que se pro-cure verificar qual ou quais são mais significativos.

Para a facilidade de implementação e de manutenção de aplicações interac-tivas, a *separação física*, ou seja, a separação do código da IU do código da apli-cação, eventualmente colocado em diferentes ficheiros, que irão integrar dife-rentes módulos que poderão originar diferentes processos, e até "correr" em máquinas distintas, é de grande importância. A forma final do código não é rele-vante pois alterações e manutenções não se fazem sobre o código objecto das aplicações mas sobre o código fonte. Daí a importância da estruturação e locali-dade bem definida deste. A separação física não é, em si, um problema para os SGIU, pelo que o que tem interesse em ser considerado, em resultado deste tipo de separação, é a complexidade da interacção a estabelecer entre os componen-tes.

Porém, *separação lógica* tem a ver com a separação para a descrição ou especificação, assumindo desde logo como resultado uma simplificação da complexidade e dos mecanismos de interacção. Esta separação lógica, ainda que advogada, não tem tido implementações demonstrativas ao nível dos SGIU. Trata-se tipicamente de um problema técnico de implementação por resolver, mas que, como a maioria dos autores concorda, não deve impedir a aplicação do princípio, mesmo que tal implique, como em geral implica, degradações de "performance". O importante é que se permita uma visão separada e de alto nível na fase de concepção e desenvolvimento de cada uma das camadas, em adequação até ao tipo de aplicação em questão, garantindo-se, através de meca-nismos próprios, a respectiva implementação.

Este ponto de vista do autor é corroborado por [Shevlin e Neelamkavil 90] onde são introduzidos os conceitos de *separação virtual* (a que deve ser

oferecida pelo SGIU) e de *representação intermédia*, sendo da competência do SGIU tradu-zir as especificações produzidas, seguindo um de vários possíveis modelos lógi-cos, em representações intermédias posteriormente convertíveis no modelo físico implementado.

O conhecido modelo de Seeheim, apresentado na figura 2.8, foi a primeira arquitectura genérica sugerida para SGIU, como resultado do Workshop de See-heim sobre *User Interface Management Systems* [Pfaff 85].

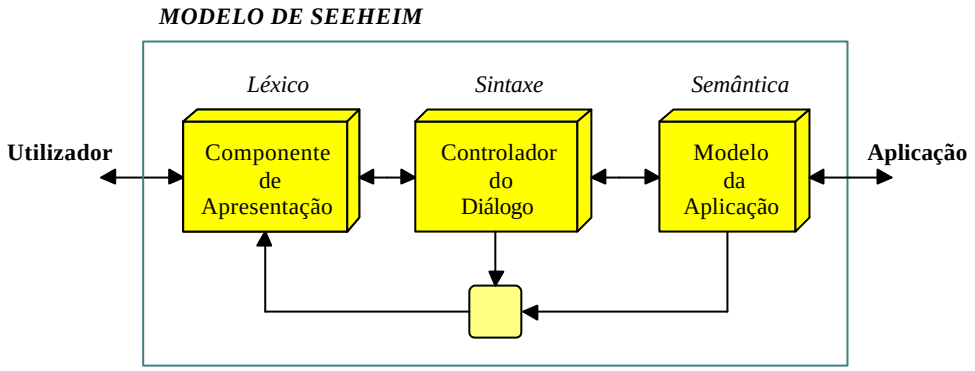

Fig. 2.8 - Modelo de Seeheim.

Distinguem-se neste modelo as três componentes fundamentais de uma IU, designadamente, a *componente de apresentação*, a *componente de controlo do diálogo* e a componente de interface com a aplicação, ou *modelo da aplicação*. A componente de apresentação é a responsável pela gestão dos detalhes de I/O. A componente de controlo do diálogo tem a responsabilidade da correcta sequen-ciação do diálogo. A componente de interface com a aplicação (ou modelo da aplicação) tem a missão de estabelecer a correcta ligação às funções e dados da camada computacional. Estes componentes podem ser feitos corresponder, e são-no em geral, aos diferentes níveis do modelo linguístico de Foley e van Dam [Foley e van Dam 82], designadamente, conforme se indica na figura, *o nível lé-xico* à apresentação, o *nível sintáctico* ao diálogo e o *nível semântico* ao modelo da aplicação ou mesmo a esta própria.

Sendo um modelo inicial de referência, de tipo lógico, ou seja, fomentando a *separação lógica* das componentes, é natural que o modelo de Seeheim apresen-te pois diversas deficiências e principalmente insuficiências quando interpretado como modelo de referência para se conseguir a *separação física* nas implemen-tações. Muitas das críticas apontadas ao modelo por certos autores [Coutaz 90] [Abowd 91] resultam deste erro de interpretação e, em consequência, de uma perspectiva demasiado exigente.

Sendo genérico e lógico, i.é., apenas procurando sugerir o idealmente atin-gível, o modelo não especifica, por exemplo, o tipo de informação que flui entre componentes, nem o tipo de abordagem e notação a usar na descrição

de cada componente, estabelecendo apenas uma subdivisão de preocupações funcionais e/ou de implementação. No entanto, enquanto referencial, o modelo é de grande utilidade dado que, pelos menos em termos de componentes a considerar numa IU (qualquer que seja a sua posterior designação), definiu um referencial padro-nizado a que praticamente todos os autores se referem e com o qual estabelecem correspondências [Duce et al. 91].

Curiosamente, e conforme se afirmou atrás, as principais críticas ao modelo acabam por surgir não pela lógica de separação proposta, mas antes pela sua (erroneamente inferida porque abusiva) falta de "performance" [Coutaz 90]. Ou-tra ineficiência habitualmente apontada ao modelo consiste em afirmar que o mesmo não fornece qualquer tipo de auxílio ao desenvolvimento modular e com-posicional da IU [Abowd 91]. Porém, enquanto modelo e não método ou metodo-logia, tal apreciação não parece ter grande sentido. Será que quando ensinamos algoritmos, seguindo por exemplo o modelo imperativo, temos a preocupação ou a possibilidade de indicar um algoritmo ideal para o desenvolvimento de algori-tmos? E se não, corresponde tal constatação à conclusão de que o modelo algo-rítmico imperativo é inexequível?

Adicionalmente, o modelo foi-se tornando, segundo uma leitura funcional, inadequado e, como tal, criticado por supostas ineficiências em certos contextos particularmente exigentes, bem como em relação a certos desenvolvimentos tecnológicos entretanto conseguidos. Por exemplo, é uma crítica comum numa leitura funcional do modelo, considerar-se que, por ser o modelo de tipo sequen-cial, então a invocação da semântica da aplicação apenas pode ser feita após a introdução de expressões de entrada completas, o que, segundo uns, inviabiliza a possibilidade de se ter "retorno semântico contínuo" associado ao nível léxico, e segundo outros apenas degrada a "performance" em tais circunstâncias, sendo tal relação tão importante em aplicações gráficas ou baseadas no paradigma de "manipulação directa" [Shneiderman 83]. Tal crítica resulta do facto de o modelo não explicitar, logo igualmente não impor, qualquer solução relativamente a co-mo e onde tal adicional suporte semântico poderia ser considerado. Em [Dance et al. 87] e [Myers 89a] são propostas duas soluções distintas para a explícita consideração de uma componente adicional de *suporte semântico*, no primeiro caso incorporada no controlador e no segundo associada ao modelo da aplica-ção, ambas indicadas como soluções sem grandes custos quanto à degradação de desempenho.

A não consideração explícita da tecnologia de interacção hoje em dia dispo-nível sob a forma de "toolkits" é um ponto adicional de crítica ao modelo, tendo todas estas considerações levado à formulação de um modelo adicional, o mode-lo de Arch-Slinky [Bass et al. 91] que se apresenta na figura 2.9, onde proble-mas de ligação ao domínio semântico da aplicação e

de utilização de tecnologia de interacção são explicitamente considerados através da inclusão de compo-nentes específicas com tais funções.

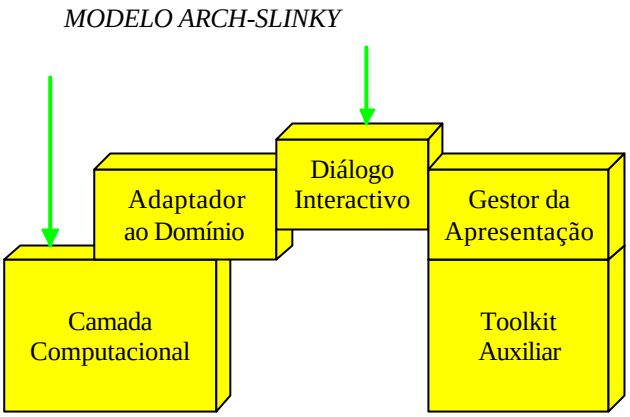

Fig. 2.9 - Modelo Arch-Slinky.

Este modelo de natureza funcional, baseado no modelo de Seeheim mas satisfazendo princípios identificados e considerados indispensáveis segundo os requisitos formulados em [Bass et al. 92a], considera agora cinco componentes. A camada computacional deve controlar, manipular e aceder a dados básicos do domínio da aplicação e executar as funções do domínio da aplicação, logo sem qualquer alteração. O *"toolkit auxiliar"*, a componente de interacção, implementa os objectos de interacção, ou seja, a interacção com o utilizador, seja esta física ou lógica. O *gestor da apresentação* deve ser encarado como um mediador, ou "buffer", entre a componente de diálogo e a componente de interacção. Ele deverá conter uma colecção de objectos a utilizar pela componente de diálogo em completa independência da sua apresentação. A componente de *diálogo interactivo* deve assumir a responsabilidade pela sequenciação das entradas, quer para o utilizador quer para a porção da aplicação, bem como a coerência entre *views*, e ainda a correcta correspondência entre os formalismos empregues para a especificação dos domínios de base, ou seja, da camada computacional, e os formalismos específicos empregues para a descrição da camada interactiva. O *adaptador ao domínio* é um componente de mediação entre a componente de diálogo e a camada computacional. Tem por objectivo representar e gerir a infor-mação "downloaded" da camada computacional, por forma a permitir um mais eficiente "retorno semântico" e melhor "performance" em geral. Admite-se neste caso uma "leitura" não sequencial do modelo.

Os modelos de Seeheim e de Arch-Slinky, enquanto modelos genéricos e ló-gicos, reflectem apenas um tipo de preocupações e de soluções nas quais o

prin-cípio de separação é associado a uma *ligação ténue*[25] entre os diversos compo-nentes.

Uma perspectiva modular, de granularidade menor mas visando maior com-posicionalidade, é a que resulta dos designados *modelos de agentes*. A ideia de base nestes modelos consiste em considerar um sistema interactivo como uma *colecção de agentes cooperantes*. Nas arquitecturas de agentes, a ideia chave consiste em considerar a construção de sistemas interactivos através da utiliza-ção de um esquema composicional que, a partir da criação de pequenos módu-los (*agentes*) e da sua composição, permita a construção de módulos mais com-plexos. Para que tal seja possível é necessário considerar inicialmente um pa-drão de decomposição adequado a todos os níveis da IU. Este padrão de decom-posição envolve em geral um número reduzido (de facto 3, conforme o modelo de Seeheim) de tipos de componentes com funcionalidades específicas. No modelo MVC [Goldberg e Robson 83] (ver Fig. 2.10) estas designam-se por *model*, *view* e *controller*. No modelo PAC [Coutaz 87] (ver Fig. 2.11) estas designam-se por *presentation*, *abstraction* e *control*. No modelo MoDE [Shan 90] estas são desi-gnadas por *appearance*, *semantics* and *interaction*.

De salientar que existe uma forte ligação entre estes modelos de agentes e o paradigma dos objectos [Cox 86] [Meyer 88], podendo até afirmar-se que tais modelos surgem em resultado da existência do paradigma. É pois igualmente importante salientar que, em tais modelos de agentes, e porque a separação en-tre dados e funções é eliminada, problemas quanto à localização ideal de dados e funções são substancialmente reduzidos.

Um dos mais antigos e referenciados modelos de interacção baseado em agentes é o modelo MVC[26] implementado no ambiente Smalltalk [Goldberg e Robson 83] [Krasner e Pope 88].

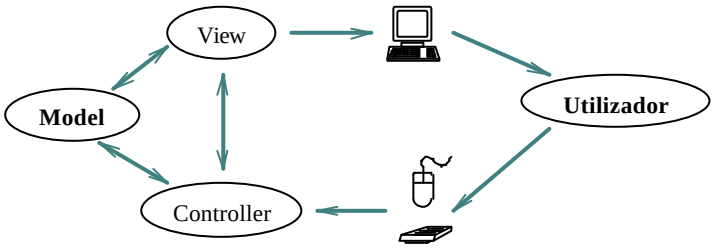

Fig. 2.10 - Modelo MVC.

[25] *loose coupling*

[26] Nas mais comuns implementações de Smalltalk, cf. Smalltalk/V e Smalltalk/W da Digitalk, o modelo MVC é implementado por classes com nomes diferentes. Assim, a *View* é representada pela classe *Pane* e a *Controller* é representada pela classe *Dispatcher*. O modelo torna-se pois MDP em vez de MVC. Os mecanismos de interligação são porém, fundamentalmente, os mesmos.

O modelo institui três componentes, correspondentes, mais uma vez, a cada uma das entidades indispensáveis num sistema interactivo. O componente desi-gnado por *Model* representa a estrutura de dados cujo conteúdo é manipulado e apresentado interactivamente. Esta estrutura de dados pode coincidir com o es-tado completo da aplicação ou ser apenas parte deste. A componente designada por *View* corresponde à componente de gestão da apresentação. O componente designado por *Controller* é o responsável pela interpretação dos estímulos de en-trada. As componentes *View* e *Controller* comunicam directamente com a com-ponente *Model* e vice-versa.

A sincronização dos estados do *Model* e da *View*, garantindo que qualquer alteração do estado da estrutura de dados associada à *View* é de imediato por esta reflectida na apresentação, é garantida pelo *mecanismo de dependências* existente em Smalltalk. Segundo tal mecanismo, objectos podem ser tornados dependentes de outros através de uma simples declaração. Estabelecida esta dependência, fica (quase) automaticamente garantida a sincronização entre objectos, dado que sempre que um objecto alterar o seu estado pode notificar todos ou apenas alguns dos seus dependentes de tal alteração. Ainda que os detalhes deste processo não sejam relevantes neste contexto, é no entanto importante referir que entre *Views* e *Models*, ou seja, entre apresentações e estruturas de dados, a criação desta dependência é automática, sendo no entan-to necessário garantir no código da aplicação a activação dos métodos que per-mitem a sua real implementação. Curiosamente até, é de referir que a activação do sofisticado mecanismo de dependências começa pelo envio da mensagem *changed* a si próprio por qualquer objecto que pretenda notificar os seus depen-dentes de que o seu estado interno foi alterado.

Note-se ainda que o modelo MVC, para além de ser um modelo lógico, pos-sui uma implementação de suporte, o que não acontece com todos os modelos de agentes. Por outro lado, o modelo MVC é homogéneo, já que qualquer das su-as componentes é descrita na mesma notação, i.é., na linguagem de programa-ção por objectos Smalltalk. Assim, uma aplicação interactiva pode ser construí-da através da criação de uma hierarquia de triplos MVC.

Tal como alguns outros modelos, por exemplo GWUIMS [Sibert et al. 86] ou IMAGES [Marques et al. 91], o modelo MVC deve ser visto como um modelo arquitectural de interacção baseado no modelo de objectos, e não como uma implementação em objectos de qualquer modelo de SGIU.

Finalmente, dentro desta classe de modelos baseados em agentes, é de referir o modelo PAC (*Presentation-Abstraction-Control*) [Coutaz 87], que é um modelo de concepção, sem implementação subjacente, baseado no modelo MVC.

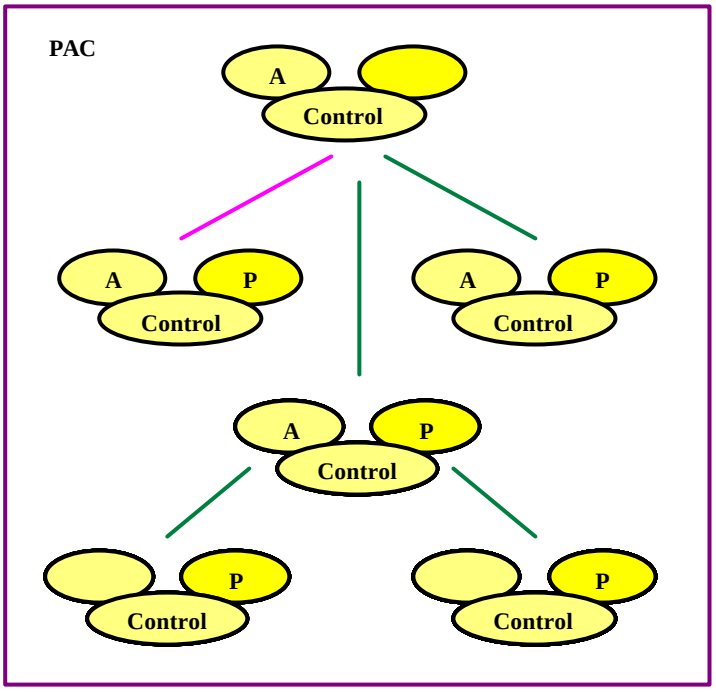

Fig. 2.11 - Modelo PAC.

O modelo PAC, como a figura 2.11 deixa entender, é um modelo conceptual que representa um sistema interactivo como uma hierarquia de agentes PAC. Cada agente triplo representa três componentes de *ligação forte*[27]. A componente designada *Abstraction* contém partes relevantes da aplicação. A componente *Presentation* contém partes relevantes da interacção, enquanto que a componen-te *Control* realiza, ao nível de cada agente, a correspondência entre as duas ou-tras componentes, bem como a compatibilidade entre diferentes agentes.

Note-se na Fig. 2.11 que o agente de mais alto nível não possui a componen-te de apresentação. Tal resulta do facto de que, em geral, o agente de mais alto nível representa a camada computacional do sistema à qual se associa algum controlo. Os outros agentes representam a decomposição da componente de *apresentação* do sistema. A decomposição da apresentação e do correspondente diálogo, e o seu relacionamento 1 para 1 ao longo dos agentes, mostra que a se-paração de preocupações não foi uma motivação no PAC. Como se pode verificar pela figura, a partir do agente raiz existe uma partição funcional intermédia, em que os agentes são compostos pelos três componentes, até se atingir o nível dos agentes mais relacionados com a interacção ao nível físico, que aparecem nas folhas da hierarquia contendo apenas componentes de apresentação e controlo. A distribuição do controlo do diálogo pelos agentes intermédios, juntamente com as componentes de apresentação, apresenta só por si, uma grande dificuldade de concepção.

[27] *tightly coupled.*

Juntando-lhe a necessidade de distribuição da funcionalidade e da ligação coerente desta aos outros componentes, parece ser notório o grau de complexidade de utilização do modelo, ainda que tal complexidade seja difícil de poder ser medida em concreto[28].

Outras arquitecturas de carácter mais específico, em geral servindo de base a implementações de SGIU, têm sido desenvolvidas. Took, em [Took 90], apre-senta uma aquitectura especializada visando garantir o bom desempenho das IU por "manipulação directa", designada *Active Medium*. Alty em [Alty 90] apresenta uma completa revisão contemplando um grande número destas arquitecturas.

2.5.5.- Modelação do Diálogo.

Diálogo, no contexto da interacção humano-aplicação, é a componente desta comunicação que pode ser externamente observável como uma bidireccional troca de símbolos realizada entre ambos os componentes em tempo real. A IU é a componente do sistema interactivo que tem a responsabilidade de fornecer o adequado contexto para que o conjunto de todos os possíveis diálogos seja, para ambos, dotados de significado e coerente. É ainda a IU o meio através do qual a troca de tais símbolos se processa.

O *princípio da separação* da componente interactiva da componente compu-tacional vem, adicionalmente, induzir a necessidade de se considerarem dois tipos distintos de diálogos. O *diálogo externo* ou *de superfície* (o realmente obser-vável), entre o utilizador e a componente interactiva, e o *diálogo interno* (não observável), entre a camada interactiva e a camada computacional.

Como foi já salientado anteriormente, a gestão deste diálogo, ou seja do fluxo da comunicação, pode ser implementada de várias formas, conduzindo, consoante a componente onde se localiza tal controlo, aos designados *controlo computacional*[29] (feito na aplicação), *controlo interactivo*[30] (na camada interac-tiva), *misto*[31] (em ambas) ou *balanceado*[32] (numa componente arbitral).

A designada *flexibilidade de interacção* diz respeito à extensão das possibili-dades oferecidas pela IU às escolhas do utilizador durante a interacção. Distin-guem-se-lhe actualmente as propriedades seguintes:

---

[28] Utilizando o modelo MVC, o autor já implementou, e supervisionou a implementação de complexos sistemas interactivos (cf. SPIOO [Pina 88]), enquanto que usando o modelo PAC não o poderá de momento fazer. De facto, enquanto que o modelo MVC tem a si associada uma implementação no sistema Smalltalk, e não só, tal não acontece com o modelo PAC que é apenas um modelo de concepção.

[29] *Computation Dominant Control.*

[30] *Dialogue Dominant Control.*

[31] *Mixed Control.*

[32] *Balanced Control* no inglês, que poderia ser igualmente traduzido por *Controlo Equilibrado.*

- "non pre-emptiveness", ou seja, fornecer sempre ao utilizador uma possí-vel acção seguinte;
- "multi-threading", ou seja, a possibilidade de execução em concorrência, ou simples "interleaving", de tarefas;
- *atingibilidade*, i.é., permitir a navegação para um qualquer estado obser-vável qualquer que seja o estado corrente;
- *multi-modalidade*, ou seja, a escolha relativa a canais de comunicação (meios de I/O);
- *multiplicidade de representação*, que corresponde à capacidade de certos sistemas permitirem representações distintas para elementos do seu esta-do (por exemplo um termómetro digital ou analógico);
- *reutilização de entradas e de resultados*, i.é., resultados apresentados poderem ser manipulados e usados de novo como entradas, e entradas anteriores poderem ser reutilizadas;
- *adaptação ao utilizador*[33], seja de forma estática (adaptação ou configura-bilidade) seja de forma dinâmica (adaptatividade);
- *migrabilidade*, i.é., suporte à transferência da iniciativa entre o utilizador e o sistema, por decisão daquele.

A maior parte destes princípios tem influência directa na concepção e implementação do diálogo interactivo bem como nos formalismos empregues para a sua descrição. Recordando os modelos de Norman [Norman 84] e de Abowd [Abowd e Beale 91] apresentados anteriormente, torna-se evidente que a consideração de algumas destas características é de grande importância para a diminuição das distâncias articulatória e observacional. Haverá no entanto sempre que ponderar o esforço necessário à sua efectiva implementação.

O *diálogo* é a componente da interacção que mais tem sido alvo de estudo pelos investigadores, daí resultando a existência de um grande conjunto de mo-delos, de base não formal e formal, para a sua representação. Dado que se dedi-ca a secção 4.4 à apresentação detalhada dos mais significativos modelos for-mais utilizados na especificação do diálogo, limitamo-nos aqui a apresentar as diferentes classes de abordagens geralmente reconhecidas.

[33] *Customizability* e *Adaptivity*.

Modelação do Diálogo	*Gramáticas* *Diagramas de Transição de Estados* *Eventos* *Sistemas de Produção* *Redes de Petri* *Processos Sequenciais Comunicantes* *Objectos*

Tab. 2.5 - Modelos do Diálogo.

A tabela 2.5 sintetiza sete das principais classes de formalismos e técnicas usualmente empregues em SGIU, ou em simples ambientes de geração ou de-senvolvimento de IU, para a descrição dos diálogos interno e externo. Uma aná-lise mais aprofundada das qualidades e capacidades descritivas de cada uma destas classses de formalismos é diferida para a secção 4.4. De salientar no entanto, desde já, que a maioria dos formalismos empregues na descrição de diálogos têm abordado principalmente o fluxo de entrada, tendo sido o "output", por motivos relacionados com o seu "baixo nível" semântico, quase completa-mente esquecido.

2.6.- Tecnologia de Interacção.

2.6.1.- Evolução Geral.

A inovação e evolução tecnológica dos dispositivos físicos, designadamente ecrãs, *rato*, "joystick", *luva*, "sound blasters", etc., possibilitou o aumento dos meios, da velocidade e da *largura de banda*[34] da comunicação bidireccional entre utilizadores e aplicações, pelo que a tecnologia lógica teve igualmente que, em resposta, evoluir por forma a poder explorar as crescentes capacidades dos dispositivos físicos que foram colocadas ao seu dispor. Interfaces com o utiliza-dor baseadas na manipulação directa de representações gráficas de entidades da aplicação[35] passaram a ser tecnologicamente viáveis [Shneiderman 83] tendo surgido diversos ambientes interactivos baseados neste estilo de interacção (tecnologia WIMP[36]), designadamente Star [Smith et al. 82], Smalltalk [Goldberg e Robson 83], Apple Lisa [Williams 83] e Macintosh [Williams 84].

---

[34] do inglês *bandwidth.*

[35] *direct manipulation user interfaces.*

[36] *WIMP = Windows, Icons, Mouse, Pop-Up menus.*

Na área da Computação Gráfica o desenvolvimento tecnológico permitiu o aparecimento, ou a viabilização tecnológica, de aplicações de elevada exigência em termos de recursos físicos, tais como sistemas CAD/CAM (2D e 3D), aplica-ções de animação por computador, simuladores e ainda os emergentes sistemas de *realidade virtual* [Bryson 92] [Kalawsky 93].

Imagens (e desenhos) são já guardadas em bases de dados de computadores pessoais e apresentadas em ecrã em resultado de operações de procura. Sons são igualmente registados e guardados em bases de dados e incorporados em texto convencional (cf. correio electrónico em NeXTStep [Webster 89]). A comuni-cação passa a poder ser realizada usando meios mais próximos da comunicação real que se realiza entre humanos. A comunicação interactiva passa assim a po-der ser *multi-média.* Por outro lado, diferentes modos de expressão de entradas passaram igualmente a ser possíveis, permitindo comunicação usando diferen-tes canais, ou seja, interacção *multi-modal*[37]. Como seria de esperar (cf. *curva de aprendizagem* atrás referida) esta evolução tecnológica provocou o aparecimento de certas subdisciplinas dedicadas ao estudo da psicologia, ergonomia, metodo-logia e aplicabilidade de sistemas baseados na possibilidade tecnológica de re-presentação, manipulação e comunicação destas novas formas de informação em computador.

Esta tese não colocará ênfase particular na abordagem dos problemas específicos dos Sistemas Interactivos Multimédia, antes se dedicando a aspec-tos metodológicos do desenvolvimento de Sistemas Interactivos em geral. Tal não impede que, posteriormente, se possam analisar algumas das contribuições aqui apresentadas à luz das necessidades impostas por sistemas mais sofistica-dos quanto ao tipo de informação comunicada entre os agentes interactivos.

Na secção seguinte passam-se em revista os principais tipos de ferramentas de apoio ao desenvolvimento de sistemas interactivos e, em particular, ferra-mentas para a construção de Interfaces com o Utilizador.

2.6.2.- Tecnologia "Software" actual.

Conforme se analisou nas secções anteriores, a separação entre a funcionalida-de da camada computacional e da camada interactiva num sistema interactivo é, independentemente das dificuldades da sua real obtenção[38], um dos mais importantes e bem aceites princípios em IHC, sendo conhecido como o *princípio da separação* [Casey e Dasarathy 82], o qual

---

[37] *multi-modal.*

[38] Como se sabe, a necessidade de aumento de *feedback* semântico, sentida por exemplo nas interfaces por manipulação directa, conduz à necessidade de diminuir tal separação ou de construir mecanismos adicionais que a mantenham.

procura garantir a *independência do diálogo* [Ehrich e Hartson 81], ou seja, o maior isolamento possível entre as camadas interactiva e funcional.

Uma das principais consequências da generalizada adopção deste princípio foi o aparecimento de muita e diversificada tecnologia de interacção, ou seja, tecnologia dedicada à construção da camada interactiva de aplicações, designa-damente "toolkits"[39], "frameworks"[40], SGIU e GIU (Geradores de IU).

Vejamos de seguida as características principais destes diferentes tipos de tecnologia de interacção.

"Toolkits".

Entende-se normalmente por "toolkit" uma biblioteca de objectos interactivos e funções, tendo por objectivo facilitar a construção e gestão de *apresentações* e *I/O*, proporcionando abstracção relativamente aos detalhes do equipamento fí-sico. Enquanto tal, "toolkits" não impõem qualquer método ou princípios para a construção da IU. Assim, "toolkits" nem suportam nem inviabilizam o princípio da separação, ficando a obediência ao princípio, ou não, a ser um critério dos que o utilizam para construir IU.

Ainda que genericamente destinados à gestão e controlo de apresentações, funcionalidades distintas podem ser associadas aos "toolkits":

- Sistemas Gráficos[41], são "toolkits" que oferecem conjuntos de primitivas para I/O, estruturação e manipulação de representações de informação gráfica bi e tri-dimensional. GKS[42] [IEEE CG 84] e PHIGS[43] [Brown 85] são os dois sistemas gráficos mais standardizados. Ainda que com ca-racterísticas gráficas diferentes, em resultado do PHIGS ser uma evolu-ção do GKS, ambos se mostram insuficientes quanto às capacidades oferecidas para a construção de IU. Acabaram por determinar, deste mo-do, o aparecimento de novas gerações de "toolkits" mais centrados neste tipo de preocupações.

- "Sistemas de Janelas"[44] são "toolkits" que possuem bibliotecas com fun-ções capazes de realizar o controlo e a gestão das capacidades de

---

[39] De momento, parece ao autor preferível o termo original inglês à tradução por "caixa de ferramentas" ou ainda pela mais específica, e orientada a ambientes de objectos, "biblioteca de componentes". Uma boa hipótese seria mesmo "biblioteca de ferramentas", incluindo objectos interactivos e funcionalidade.

[40] por vezes traduzido como "infraestrutura", "enquadramento" ou "contexto", não tem ainda consenso suficiente para ser inequivocamente traduzido.

[41] por vezes referidos como "pacotes gráficos", do inglês *graphics packages*.

[42] *Graphics Kernel System*.

[43] *Programmers Hierarchical Interactive Graphics System.*

[44] *Window Systems.*

I/O de estações de trabalho, permitindo a "multiplexagem" do ecrã físico em dispositivos lógicos - as *janelas* -, e controlando os serviços de "input" e "output" respectivos [Hopgood et al. 85]. Sistemas de Janelas encontram a sua génese em Palo Alto na Xerox PARC, aquando do desenvolvimento do ambiente Smalltalk. A evolução, a partir destes iniciais sistemas de mono-utilizador, mono-tarefa e espaço de endereçamento simples, para plataformas (cf. UNIX?) com características de tipo multi-processo e multi-espaço de endereçamento, levantou problemas de sincronização entre processos acedendo a recursos partilhados (tais como o ecrã, o *rato* e o teclado) que conduziram às duas grandes classes de implemen-tação de sistemas de janelas: os "kernel-based", baseados na ligação do código da aplicação às rotinas gráficas e de gestão (cf. por exemplo SunWindows [Rosenthal 87]) e os "server-based", onde os serviços de gestão de janelas se concentram num servidor de rede, podendo qualquer cliente, por RPC[45], aceder aos mesmos usando um protocolo de baixo nível (cf. X [Scheifler e Gettys 86]) ou de alto nível (cf. NeWs [Sun 86], onde a linguagem Postscript [Adobe 86] é usada como protocolo). Em [Stern 87], as características principais dos sistemas X, NeWs, MacApp e Windows são analisadas e avaliadas, de uma forma exemplar-mente sintética e rigorosa.

? Outros "toolkits" têm por função fundamental facilitar a construção de apresentações com determinado "look & feel" para as IU, fornecendo um conjunto de objectos interactivos[46], com *aparência* e *comportamento* particulares, tais como menus, botões, "list panes", "scroll bars", "combo boxes", caixas de diálogo, etc. Alguns destes "UI toolkits" são apenas camadas de mais alto-nível sobre sistemas de janelas, que oferecendo apenas mecanismos básicos, com vantagens em flexibilidade, abdicam de políticas de IU, permitindo que clientes particulares, os "Gestores de Janelas"[47], implícitos nestes "UI toolkits", possam definir *apresentações de janelas*, formas de manipulação, etc. Limitando-nos, a título de exemplo, ao sistema X, as IU finais muito irão depender do "UI toolkit" adoptado, dentre, por exemplo, Athena, Andrew, e Xt [Rosenthal 87], ou ainda Xol, Xview (ambos para OPEN LOOK [AT&T 88] mas com APIs diferentes) [Probst 88] e OSF/Motif Toolkit [OSF 89].

Um "toolkit" é por vezes considerado dividido em duas camadas, em par-ticular, o *widget set* e a *camada intrinsics*. O *widget set* é a colecção de técnicas de interacção disponíveis que definem o *look & feel*. A

[45] *Remote procedure call.*

[46] *widgets* ou *gadgets.*

[47] *Window Managers.*

camada *intrinsics* (cf. Xtk [McCormack 88]) fornece os serviços comuns e as té-cnicas de implementação, em geral, quer recorrendo a interfaces proce-dimentais, quer recorrendo a implementações em objectos.

? Outros ainda, fornecem linguagens de alto nível que, não tendo por ob-jectivo uma melhoria das apresentações, visam sobretudo facilitar o pro-cesso da sua construção sobre um dado "toolkit", como UIL [DEC 88] e Tcl/Tk [Ousterhout 91].

? Ainda que o número de "toolkits" e "windowing systems" tenha vindo a diminuir em resultado de esforços de standardização, a conversão de IU de uma tecnologia para outra é ainda um processo complexo. Assim, a solução foi o desenvolvimento de "virtual toolkits", possuindo *widgets* virtuais que podem ser feitos corresponder aos *widgets* de um qualquer "toolkits" real. O seu código poderá ser ligado ao código de um qualquer "toolkit" real. Um exemplo significativo é o XVT [XVT 91], que fornece uma interface C ou C++ que permite abstrair das diferenças entre os mais comuns "toolkits" para os mais diferentes ambientes.

? "Toolkits" são implícitos em ambientes mais sofisticados tais como os apresentados a seguir, designadamente, os "application frameworks" e os SGIU.

"Frameworks" e GIU.

Um "framework" ou "application framework" é um "esqueleto" de código que implementa de forma genérica uma interface com o utilizador padronizada. O implementador deverá depois fazer a ligação deste código com as especificidades da aplicação a desenvolver. Trata-se portanto de um processo de construção de aplicações interactivas baseado em reutilização, instanciação e, em certos casos, adaptação. Um "framework" é, portanto, uma arquitectura incompleta: é mais detalhada do que um modelo mas mais genérica que uma implementação.

Por exemplo, num Macintosh as aplicações interactivas são tão semelhantes do ponto de vista da sua apresentação e do seu comportamento interactivo por-que são todas construídas usando o "application framework" designado MacApp [Schmucker 86]. No entanto, o mais antigo "application framework", ainda que não tão conhecido dos utilizadores finais, é o do Smalltalk, baseado no modelo MVC (*model-view-controller*) [Goldberg e Robson 83] [Krasner e Pope 88].

Exemplos mais recentes são ambientes tais como o NeXT Interface Builder [Garfinkel e Mahoney 93], VisualBasic [Burgess 93] e VisualC++

[Young 93] (apresentados e analisados em [Martins e Moura 94]), ou UIMX [VisualEdge 90] e Prototyper [SmethersBarnes 90], que capitalizam na dificuldade de utilização dos "toolkits" e "application frameworks" disponíveis, e fornecem linguagens de mais alto nível para a construção de IU. Algumas destas ferramentas são consi-deradas até como exemplos de GIU (Geradores de IU)[48] pelo facto de fornecerem mecanismos de alto-nível para a geração de IU sobre mecanismos já existentes.

SGIU (UIMS).

Os SGIU são ambientes compostos de duas componentes fundamentais para a criação e gestão de IU: a componente de especificação e prototipagem da IU e o núcleo de "run-time". A componente de concepção, desenvolvimento e prototi-pagem da IU é designada por UIDE[49], sendo composta por "ferramentas" gráfi-cas de edição, de especificação de diálogos, de avaliação, etc. O núcleo de "run-time" designa-se por UIS[50] e deve suportar a execução da IU criada, por exem-plo, gerindo a comunicação com a camada funcional. Assim, o UIS fornece um "framework" com base no qual os construtores do diálogo podem criar a IU. Da ligação desta IU à camada computacional, com base no suporte do UIS, resulta o definitivo e executável Sistema Interactivo.

SGIU foram, desde os anos 80, encarados como as grandes soluções como ambientes de auxílio à construção de sistemas interactivos. Inúmeros SGIU foram sendo desenvolvidos não tendo no entanto nenhum ultrapassado a fase experimental ou de produto interno. De muito maior divulgação e sucesso têm sido, indiscutivelmente, os seus subconjuntos, que atrás designámos por "toolkits" e "frameworks", bem como alguns sistemas de geração automática de IU (GIU). Em [Coutaz 89] é apresentada uma reflexão sobre as expectativas e possíveis razões do insucesso dos SGIU, a principal das quais será talvez terem procurado um grau de parametrização difícil de atingir. Por outro lado, a grande diversidade de soluções encontradas nos SGIU não terá permitido a constatação de verdadeiras soluções genéricas, sendo porém de registar alguns esforços nesse sentido [Carlsen 92].

Porém, e dada a inegável importância de muitas das soluções encontradas na implementação de SGIU, indicam-se a seguir os SGIU mais referenciados, classificados em função do modo como o diálogo com o utilizador é definido.

A maioria dos SGIU baseiam-se em representações do diálogo realizadas usando formalismos ou linguagens particulares. Um dos mais antigos SGIU referenciado é o sistema Tiger [Kasic 82] que usa uma árvore de menus para

---

[48] cf. *Interface Builders* ou *Interface Generators.*

[49] de *User Interface Development Environment*.

[50] de *User Interface System.*

o controlo do diálogo. São vários os SGIU que baseiam a descrição de diálogos em diagramas de transição de estados, tais como Connect [Alty 84] e RAPID/USE [Wasserman e Shewmake 82], enquanto que GMENUS [Duce et al. 91] usa gra-fos designados "User Interaction Points". O Syngraph [Olsen e Dempsey 83] e o Cousin [Hayes e Szeleky 83] são exemplos de SGIU que usam gramáticas para a descrição do diálogo.

GIGA [Bordegoni et al. 90] usa ATN e Petri Nets para a implementação do diálogo e de componentes de apresentação, associando eventos a certos *tokens* da Rede de Petri.

SGIU baseados no paradigma de objectos têm tido uma grande proliferação nos últimos anos. DMS [Hartson et al. 84], GWIMS [Sibert et al. 86] e Images [Simões e Marques 87] são alguns exemplos. GUISE [Bousse 92] é um "appli-cation framework" baseado em Smalltalk que pretende eliminar algumas defi-ciências de ligação entre componentes encontradas no modelo MVC.

Sassafras [Hill 86] baseia-se no modelo de eventos e na linguagem ERL (Event-Response Language) que especifica o diálogo sob a forma de agentes cujo comportamento é especificado por regras *condição-acção* activadas pela ocorrên-cia de eventos reconhecidos pelo agente.

Menulay [Buxton et al. 83], Trillium [Henderson 86], Peridot [Myers 87], Garnet [Myers 89b] e Tube [Hill e Herrman 89] permitem uma definição da apresentação da IU por manipulação gráfica directa, tal como sucede já em ambientes mais populares tais como em NeXT Interface Builder, em VisualBasic e VisualC++. Em [Gomes 91], a arquitectura OO-AGES, baseada no paradigma dos objectos e visando suportar IU com este nível de exigência, é proposta.

MIKE [Olsen 86], IDL [Foley 87], UIDE [Foley et al. 89], ITS [Wiecha et al. 90] e Chiron [Taylor e Johnson 93] são SGIU que procuram a geração automática da IU a partir de especificações de alto nível e de vários tipos de informações "reti-radas" da camada computacional. Por serem, juntamente com GIGA [Bordegoni et al. 90], abordagens em princípio comparáveis, na motivação, na forma ou no conteúdo, ao sistema GAMA-X desenvolvido no contexto do trabalho nesta tese apresentado, haverá o cuidado de, após a apresentação do sistema GAMA-X, se realizar a análise comparativa possível.

Em [Myers 92] pode ser encontrada uma das mais actualizadas revisões da tecnologia disponível para a construção de IU.

2.7.- Sumário.

Fez-se neste capítulo uma síntese do estado actual do desenvolvimentos na área de IHC, tendo-se navegado ao longo das principais subdisciplinas no sentido utilizador-aplicação.

Foram analisadas algumas das visões possíveis da área de IHC enquanto disciplina, tendo-se concluído que, no âmbito desta tese, a perspectiva mais adequada é a que assume IHC como uma disciplina de Engenharia.

Enquadramentos possíveis para o estudo abstracto, não construtivo, do fenómeno interactivo foram apresentados, com o objectivo de permitir a clarifica-ção e a fixação de algum vocabulário, de conceitos e de contextos.

Os principais modelos existentes em IHC foram de seguida sumariados e classificados tendo em consideração o objecto modelado e o destinatário do processo de modelação. Já claramente numa área onde existe grande envolvi-mento da camada computacional, apresentaram-se igualmente modelos e técni-cas vocacionadas especificamente à representação e estudo dos diálogos entre utilizador e aplicação.

Finalmente, e cobrindo já preocupações de implementação, fez-se referência à principal tecnologia actualmente disponível, quer para a gestão de dispositivos físicos de interacção, quer para a construção da IU de sistemas interactivos. Quanto a esta tecnologia, poder-se-á, em resumo, dizer que SGIU representam abstracções de controlo e de diálogo, "Sistemas de Janelas" são abstracções de recursos e do estado interactivo, “application frameworks” são aplicações inter-activas incompletas que, por instanciação geram a desejada aplicação e “tool-kits”, ao seu nível mais baixo, são abstracções dos dispositivos físicos.

O objectivo geral consiste em diminuir o tempo e o esforço de construção de IU e Sistemas Interactivos, por reutilização de código de interacção já existente, e procurando garantir qualidade, pelo menos de apresentação, da IU final. Quanto ao processo ou método de o fazer com alguma sistematização, parece poder concluir-se serem poucos os trabalhos que revelam preocupações em tal sentido.