CAPÍTULO 4

MÉTODOS FORMAIS EM IHC

ÍNDICE

4.1.- Introdução.

A aplicação de métodos formais em IHC tem por primeiro objectivo o domínio da complexidade através da aplicação de um normal processo de abstracção, não sendo consensual nem claro o que se deve abstrair e o que se deve representar. É porém um ponto assente, ou quase[1], que a aplicação de métodos formais de especificação na concepção e desenvolvimento de sistemas interactivos é, apesar de todas as possíveis dificuldades, a abordagem correcta e indispensável à necessária introdução de rigor neste processo. No entanto, a introdução deste rigor no processo é suposto conduzir a uma melhoria da qualidade endógena do sistema, restando de seguida garantir a sua qualidade exógena, ou seja, aquela que é revelada e avaliada pelos utilizadores. Esta última apenas poderá ser real- mente atingida em colaboração com outras disciplinas.

A figura 4.1 procura sintetizar esta perspectiva, considerando-se ainda que o emprego de métodos formais ou rigorosos pode ser encarado sob três distin-tas perspectivas, não disjuntas mas antes supostamente complementares.

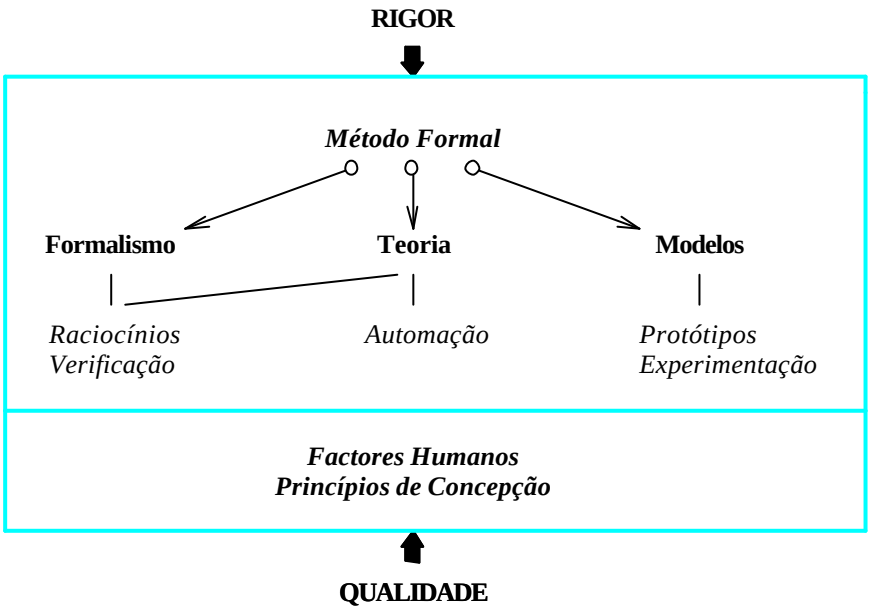

Fig. 4.1 - Rigor e Qualidade em IHC.

Em primeiro lugar, podemos considerar o simples emprego de uma notação formal de descrição, com semântica bem definida, possivelmente

[1] Os argumentos contra a utilização de métodos formais em IHC, ou seja na especificação da camada interactiva, são os mesmos apresentados relativamente à sua tradicional aplicação em Engenharia da Programação, isto é, na especificação da camada computacional.

assente numa teoria. De facto, e conforme pode ser verificado pelo resumo que se apresenta na secção seguinte, a maioria dos trabalhos realizados em IHC possuindo alguma base rigorosa, consistem no emprego de formalismos na descrição de um aspecto particular do fenómeno interactivo em si, ou de características de ferramentas e objectos usados na construção dos sistemas.

Teorias ou teorizações são os contextos necessários à compreensão completa dos processos interactivos visando a sua automatização. Para tal, modelos baseados na teoria devem ser construídos. Como vimos em capítulo anterior, não se podem referir muitas teorias da interacção ou dos sistemas interactivos, mas muitos são os diversos tipos de modelos existentes na área. Muitos destes modelos, de carácter mais rigoroso e mais operacional ou operacionalizável, são ideais para a construção de protótipos dos componentes modelados.

A disparidade da aplicação de métodos formais em IHC justifica uma mais profunda análise dos diferentes *papéis* (objectivos) e *âmbitos* (extensões) de tais aplicações. As secções seguintes são devotadas à apresentação de uma síntese dos principais trabalhos de aplicação de métodos formais a diferentes áreas da IHC. Salientar-se-ão os formalismos para a descrição de diálogos, pela sua rele-vância.

O problema da especificação de diálogos com uma elevada sensibilidade ao contexto determinado pela aplicação, e as vantagens da utilização de métodos de especificação que possibilitem a prototipagem de IU, e os requisitos para que tal técnica possa ser utilizada em IHC são analisadas na secção 4.5.

Na secção 4.6, sumarizam-se as diferentes aplicações de métodos formais ou de formalismos em IHC em relação a um referencial tridimensional para a classificação dos diferentes tipos possíveis de utilização destes métodos. Tendo por base as características representadas neste referencial, tecem-se também algumas considerações sobre as condições julgadas necessárias para que o uso de um método de especificação formal, ou de um simples formalismo, possa ser a base de um verdadeiro *método* de desenvolvimento de sistemas interactivos.

4.2.- Âmbito e Objectivos.

Os métodos formais têm vindo a ser aplicados em IHC com os mais diversos âmbitos e objectivos. No entanto, deve desde já referir-se, têm-no sido seguindo mais uma perspectiva de descrição e, eventualmente, de prova de satisfação de dadas propriedades, ou visando o desenvolvimento de protótipos, do que numa perspectiva de servirem como elemento de base para o desenvolvimento da IU, e muito menos do Sistema Interactivo.

Posta de parte qualquer hipótese de formalização do utilizador humano, ain-da que, como se viu, algumas das suas características cognitivas e de comporta-mento possam ser pelo menos representáveis, a atenção foi então dirigida para o estudo formal do fenómeno interactivo e dos componentes dos sistemas a este mais directamente associadas.

A formalização de um conjunto de propriedades de interacção que, pelo me- menos teoricamente, devem ser exibidas por um qualquer Sistema Interactivo, tem sido o objectivo de vários autores [Dix 87] [Dix e Harrison 89] [Abowd 91]. O objectivo é poder confrontar, de um modo rigoroso, as propriedades reais das IU com tais princípios e, daí, inferir níveis de usabilidade. Muitos são os princípios formalizados. Porém, dado terem por base de formalização modelos abstractos de sistemas interactivos, acaba por se tornar difícil fazer a sua verificação em IU reais.

Dix e Harrison em [Dix e Harrison 89] referem a existência de um *lapso de formalidade*[2] entre a usual especificação de requisitos em linguagem natural e as especificações funcionais realizadas em linguagem matemática. Alguns dos modelos de interacção mais abstractos, em particular os modelos PIE e REDPIE, visam exactamente estabelecer uma ponte entre requisitos informais e especifi-cações formais, através da expressão rigorosa, matemática, de propriedades in-formalmente reconhecidas e definidas pelos engenheiros de factores humanos e psicólogos. Obviamente que existirá sempre uma tradução de linguagem natural para linguagem formal e, portanto, um lapso formal há-de sempre existir. No entanto, e assumindo que quem vai verificar a satisfação dos requisitos de usa-bilidade é um psicólogo, é importante que existam modelos de interacção de base rigorosa mais orientados para estes.

Algumas das críticas à aplicação de métodos formais em IHC acabam mesmo por reverter em seu favor. Uma classe usual de críticas baseia-se na ideia de que, ao favorecerem a abstracção do detalhe, os métodos formais aca-bam por não reconhecer que a usabilidade de um sistema interactivo depende de uma miríade de detalhes presentes aos mais diversos níveis de concepção, que só uma técnica iterativa de concepção poderá auxiliar a acertar [Carroll 90]. Por técnica iterativa de concepção pretende-se, neste contexto, experimentação e concepção empírica, sob o argumento óbvio de que num processo de concepção o problema não consiste em não se saber *ainda* tudo, mas antes em ser certo que *nunca* se poderá saber o suficiente.

Porém, a experimentação está actualmente ligada à noção de protótipo ou modelo executável, e a possibilidade da sua construção expedita ligada ao em-prego de formalismos de base rigorosa. O ciclo de argumentação é, portanto, de novo fechado sobre o emprego de uma notação formal.

---

[2] designado por *formality gap.*

Saliente-se, em todo o caso, o facto de que o emprego de métodos formais em IHC, de acordo aliás com a perspectiva de IHC apresentada na secção 2.2, não pode nunca ser encarada como condição suficiente para a qualidade dos sistemas mas antes, e unicamente, como uma condição considerada necessária. Em reforço desta ideia, podemos inclusivamente salientar o facto conhecido de que a maioria dos métodos e modelos formais empregues em IHC não têm por objectivo uma abordagem construtiva ao problema, dado não proporcionarem as regras e o suporte necessários à construção de um sistema interactivo, ou IU, a partir da especificação formal das suas componentes.

Na verdade, espera-se que o trabalho que se apresenta nesta tese possa dar algum contributo à utilização construtiva dos métodos formais na concepção e desenvolvimento de IU.

4.3.- Aplicações dos Métodos Formais em IHC.

Procurando justificar algumas das afirmações anteriores quanto à aplicação de métodos formais em diversas áreas e problemas de IHC, referem-se nesta sec-ção, a título de exemplo e sem preocupações de completude, alguns dos traba-lhos que exemplificam o alargado âmbito da sua aplicação. A atenção será no entanto centrada na aplicação de métodos formais na especificação de diálogos.

A formalização dos sistemas gráficos GKS e PHIGS com vista a uma melhor compreensão do seu funcionamento, a análise da sua alegada compatibilidade, e do comportamento dos dispositivos de entrada e saída, tem sido realizada por Duce usando VDM, Z ou OBJ, e, mais recentemente, LOTOS (cf. [Duce et al. 88], [Duce e Damnjanovic 92] e [Duce e Patterno 93]).

No campo da especificação da funcionalidade e de algumas propriedades dos primeiros sistemas interactivos, i.é., sistemas de edição de texto, referem-se as especificações produzidas por Sufrin [Sufrin 82] e por Martins [Martins 87a], ambas em Z, sendo a primeira uma especificação de um editor de texto fictício, enquanto que a segunda se refere a um editor de texto, o SPY, já implementado à data da especificação. O mecanismo de “buffering” deste editor de texto foi objecto de análise em [Abowd 91] numa perspectiva concorrente, ou seja, tendo por base um modelo de agentes.

Alguns autores usaram formalismos para a especificação da funcionalidade de objectos a serem usados na construção de IU, tais como janelas, menus, caixas de diálogo, e outros. Nesta área são de referir historicamente os trabalhos de Studer [Studer 84], que usa VDM para a especificação formal de uma IU baseada numa rede de menus, de Cardelli e Pike [Cardelli e Pike 85] que desen-volve uma linguagem baseada em CSP para a descrição de interacção concor-rente, ainda que ao nível dos dispositivos físicos e, mais recentemente, a notação Lean Cuisine [Apperley e

Spence 89] para a descrição estrutural e atributiva do comportamento e apresentação de sistemas de menus complexos. Exemplos de especificações formais de "window managers", não muito abundantes, podem ser encontradas em [Martins 86], [Took 86] e [Lamas 94], curiosamente todas realizadas usando a linguagem Z.

Mais recentemente, os métodos formais têm recebido uma grande atenção nas áreas dos sistemas hipermédia e multimédia. Nos primeiros, a descrição da estrutura da informação e, consequentemente, das diversas possibilidades de *navegação*[3] têm merecido uma abordagem formal, em geral baseada em Redes de Petri e suas extensões. Nos sistemas multimédia, por seu lado, os requisitos de expressividade impostos aos formalismos de especificação são levados ao extremo. Nestes sistemas, para além da impossibilidade de se abstrair da noção de estado, existe a necessidade de serem consideradas restrições temporais e também a sincronização entre meios de informação, por exemplo, audio e video. Dentre os trabalhos mais recentes na área multimédia que apresentam este tipo de preocupações de formalidade, é de referir, a nível externo, [Bowman et al. 94], onde são analisados diferentes formalismos satisfazendo os requisitos de descrição atrás referidos e sugerida a linguagem LOTOS, enquanto que a nível interno, em [Bernardo 94] uma linguagem de especificação que é um subcon-junto de CSP é desenvolvida, bem como o ambiente de suporte à sua execução.

No entanto, ao longo dos anos, a componente dos Sistemas Interactivos que maior tratamento formal recebeu foi, sem dúvida, o diálogo. Porque a clareza e grau de expressividade da descrição do diálogo estão em muito dependentes do formalismo empregue, e porque estas descrições são muito importantes para uma possível geração automática dos controladores de diálogo das IU, são a seguir referidas e analisadas as diferentes abordagens que têm sido empregues.

4.4.- Especificação Formal de Diálogos.

4.4.1.- Definição Informal de Diálogo.

Para que se possam compreender as diferentes características, e âmbito, das propostas desenvolvidas para a formalização da componente responsável pelo controlo do *diálogo* entre um utilizador e um sistema interactivo (o *controlador do diálogo*), comecemos por fixar uma definição abrangente do que deve enten-der-se *diálogo* num sistema interactivo. Adopta-se, no contexto desta tese, a definição, comum aliás, de *diálogo* como sendo a observável e bidireccional troca de símbolos e acções entre um humano e um sistema computacional, cumprindo a *interface com o utilizador* a função de suporte

[3] *browsing.*

software e *hardware* para a realização de tal troca, i.é., *comunicação* [Hartson e Hix 89].

Num diálogo são identificáveis, assim, pontos particulares no tempo em que informação é transmitida de um "interlocutor" ao outro. Estes pontos são desi-gnados por *interacções* ou *pontos de interacção*, correspondendo a "input" ou "output" que é realizado no sistema interactivo. Considerando cada uma destas interacções como um evento no sistema, a ordem temporal da sua ocorrência, sequencializada ou não, consiste na designada *estrutura* ou *fluxo do diálogo*.

O diálogo é, em geral, caracterizado também pelo seu *estilo*, i.é., pelos meca- nismos de manipulação e transmissão de informação, sendo de distinguir as lin-guagens de comandos, os menus, os mecanismos de questão-resposta, gráficos, "caixas de diálogo", linguagem natural, preenchimento de "impressos"[4], a mani-pulação directa, etc., sendo hoje em dia comum que os diálogos apresentem um misto de alguns destes estilos.

Por outro lado, o *modo* como o diálogo é realizado é mais uma característica a ser também considerada, dada a emergência de interfaces multi-modais, ou seja, possuindo diversos dispositivos de I/O para interacção.

O *conteúdo* e a *forma* da informação são igualmente aspectos importantes do diálogo, sendo já tecnologicamente possível dispor-se de informação sob varia-das e sofisticadas formas de apresentação e estruturação (cf. sistemas de hiper-texto e sistemas multimédia).

Adicionalmente, e dado que a definição anterior não caracteriza, quer à luz do referencial de sistema interactivo apresentado no capítulo 2, quer à luz do modelo estrutural de Seeheim [Pfaff 85], a amplitude de cada interacção, isto é, se a comunicação é tratada apenas ao nível da IU, ou provoca modificações de estado na camada computacional, parece importante introduzir a noção de *pas-so de diálogo*. Este é entendido explicitamente como uma sequência de entradas válidas do utilizador, ou apresentação de valores do sistema, para os quais não é necessária qualquer acção particular ao nível da camada computacional. Assim, é de admitir que entradas do *utilizador* possam ter interpretação e trata-mento apenas a nível da camada interactiva, ainda que, para tal, esta deva pos-suir informação adicional[5].

Finalmente, quanto ao modo da sua estruturação e execução, diálogos são classificados em geral como *sequenciais*, ou mono-caminho, que são diálogos entre um utilizador e uma só tarefa ou actividade, seguindo um trajecto linear (ainda que, eventualmente, resultante de sucessivas escolhas entre diferentes possíveis opções colocadas em cada instante) e diálogos

---

[4] *form-filling*

[5] Como veremos mais tarde, um Modelo da Aplicação mais rico poderá facilitar este tipo de acções ao nível da camada interactiva e diminuir *overheads* de comunicação entre camadas.

multi-caminho[6], nos quais diversas actividades ou tarefas estão acessíveis ao utilizador para diálogo, podendo o diálogo prosseguir em qualquer delas de forma sequencial e indepen-dente, e a comutação de contexto realizar-se a qualquer instante, sem perda de controlo. A independência destas tarefas leva a que alguns autores designem este tipo de diálogos por *diálogos assíncronos*.

A especificação de um diálogo consiste não só na especificação do fluxo do diálogo, mas também dos efeitos deste na camada computacional e na camada interactiva, do tratamento de erros e do auxílio disponível, dependendo no en-tanto da abrangência dos formalismos a perspectiva de maior interesse.

4.4.2.- Formalização do Diálogo.

Diagramas de Transição de Estados.

Os Diagramas de Transição de Estados (DTE) são uma representação pictórica das Redes de Transição de Estados (RTE), um modelo baseado em teoria de grafos e máquinas de estados finitos (FSM)[7], muito utilizado em Ciências da Computação na representação do espaço de estados acessíveis a uma máquina.

Os DTE são constituídos por dois tipos básicos de entidades: os *nodos*, que são representados por círculos, e os *arcos*, que são representados por linhas, em geral orientadas, formando um grafo orientado. Os nodos são associados a *esta-dos particulares* do sistema representado (visto como uma máquina de estados) enquanto que os arcos representam as ocorrências particulares que podem levar o sistema de um estado a outro - uma *transição de estado*.

Quando utilizados na descrição de diálogos, cada estado de um DTE repre-senta um estado do diálogo entre o utilizador e o sistema interactivo, enquanto que cada arco especifica uma acção do utilizador, representada em abstracto por um *token* de entrada, bem como, em versões aumentadas, por outras acções que, segundo [Singer 79], tanto podem ser feitas corresponder a alterações no estado da camada computacional como a resultados por esta enviados ao utili-zador. Os arcos determinam portanto a possível evolução do diálogo. Um cami-nho entre dois nodos, i.é. uma sequência de arcos entre dois nodos, representa a sequência de entradas válidas do utilizador para que o diálogo possa evoluir de um estado a outro. Caminhos entre estados iniciais e finais correspondem a *passos completos de diálogo*. Os DTE

---

[6] atrás utilizou-se o termo inglês usual, *multi-thread*, com o mesmo sentido.

[7] FSM de *finite state machines* [Minsky 72].

especificam portanto diálogos como sendo uma sequência de estados ou, numa visão alternativa mas equivalente, como as sequências de acções ou eventos que produzem tal sequência de estados.

A primeira utilização de DTE simples na descrição de diálogos foi proposta por Parnas [Parnas 69]. Encontrando-se o sistema num dado estado, ou nodo, os arcos de saída desse nodo são associados a entradas válidas do utilizador. A cada entrada válida associa-se uma transição para um novo estado.

Conforme se salientou atrás, na sua versão mais simples, os DTE não têm capacidade expressiva para representar mais do que as acções do utilizador. Extensões foram entretanto sugeridas procurando incorporar a representação das respostas do sistema, quer sob a forma de acções associadas aos arcos quer como acções associadas aos estados [Woods 70], diferenciando-se até o seu tipo [Singer 79].

A insuficiente estruturação dos DTE, de que resulta uma tendência para a explosão de estados, pode ser resolvida recorrendo a *subdiagramas*. Um arco num diagrama pode corresponder à invocação de um subdiagrama. Uma divisão lógica em subdiagramas sugerida por Green em [Green 86], granularidade que é igualmente seguida nos *Guiões de Interacção* apresentados nesta tese, consiste em criar um subdiagrama para cada um dos comandos da interface com o utili-zador. Subdiagramas para a introdução de operandos de determinado tipo são igualmente ser definidos. Uma outra possível divisão dos diagramas é proposta por Jacob em [Jacob 85], que, seguindo o modelo linguístico de interacção suge-rido em [Foley e van Dam 82], propõe que as camadas léxica e sintáctica sejam tratadas por diferentes grupos de diagramas, sendo então os diagramas léxicos invocados pelos sintácticos.

Diagramas de Transição Recursivos [Green 86,] são um caso particular de DTE nos quais os subdiagramas podem invocar-se a si próprios. Esta possibili-dade aumenta o poder expressivo dos DTE, tornando-os expressivamente equi-valentes a gramáticas independentes do contexto. São porém relativamente com-plexos de utilizar.

A importância de se especificarem diálogos nos quais certas transições só podem ocorrer se determinadas *condições,* quer de puro contexto de interacção quer relacionadas com o estado da aplicação, se verificarem - designados *diálo- gos dependentes do contexto* - conduziu à introdução de inúmeras extensões aos DTE principalmente no sentido de os prover de "memória", i.é., de variáveis ca-pazes de armazenar informação relevante para o controlo do diálogo, designada-mente, dados sobre o próprio estado do diálogo e dados sobre o estado da apli-cação[8]. Surgem assim os *Diagramas de Transição Aumentados* ou *Redes de Transição Aumentadas* (cf. o original inglês

[8] Os autores dividem as suas opiniões não tanto quanto à informação a representar no DTE mas antes quanto às restrições de acesso que devem ser impostas.

Augmented Transition Networks, de sigla ATN), nos quais são introduzidas variáveis - *registos* - e condições à reali-zação de certas transições. Variáveis podem ser associadas quer a estados quer a arcos, podendo aí ser manipuladas. As expressões condicionais envolvendo tais variáveis são naturalmente associadas aos arcos que representam as transi-ções que condicionam. ATNs aumentam o poder expressivo das RTEs até ao ní-vel de uma Máquina de Turing.

Os ATN, com todo o seu poder expressivo, têm sido extensamente aplicados na especificação de diálogos. Casey e Dasarathy em [Casey e Dasarathy 82] usa-ram ATNs na especificação de requisitos funcionais, modelação e validação de interfaces para sistemas de tempo real. *Checkpoints* são propostos como sendo nodos especiais onde determinadas condições de contexto são testadas. *Timers* são introduzidos como atributos temporais associados a estados, para efeitos de satisfação de dados requisitos de desempenho temporal. Assim, atingido um es-tado contendo um temporizador, este é activado, correspondendo, por exemplo, ao tempo máximo de espera pela próxima entrada do utilizador. Findo este tem- po, por exemplo, a transição seguinte pode ser realizada automaticamente.

Em [Wasserman 85] os ATN são usados para especificar a linguagem de en-trada do utilizador, as acções semânticas da aplicação, bem como caracterísi-cas da apresentação (em terminal VT100). No sistema RAPID/USE, desenvolvido para a construção de IU para Sistemas de Informação, [Wasserman e Shewmake 85] e [Wasserman et al. 86], estas extensões são usadas intensivamente, poden-do os diagramas ser gerados automaticamente a partir de uma descrição textual de nodos e arcos. O objectivo principal é a geração automática de protótipos da IU, sendo ainda de salientar algumas preocupações de desenvolvimento metodo-lógico de sistemas de informação interactivos, ainda que a partir de requisitos não formalmente especificados.

Em [Jacob 85] a ideia de representar os ATN de forma textual e de desenvol-ver um interpretador de tal linguagem, para que se possam construir protótipos da interface, é também sugerida, tendo por objectivo principal a prototipagem, e não tanto o desenvolvimento da IU final.

Como salienta Green em [Green 86], um dos maiores problemas com os ATN consiste na sua impossibilidade de especificar, de uma forma simples e clara, os tratamentos adequados a realizar sempre que, num dado estado, surgem entra-das inesperadas, ou seja inválidas. A maioria das abordagens usando ATN
assu-me a solução de ignorar as entradas não especificadas. Outras usam etiquetas especiais em arcos (*wild cards*) representando o conjunto de entradas inválidas e conduzindo a um estado único onde a recuperação do erro é tentada. Olsen, em [Olsen 84], introduz para o tratamento deste tipo de anomalias um conjunto de estados de excepção - *pervasive states* - que dão acesso a um subdiagrama correspondente ao tratamento de erros.

Porém, a maior dificuldade dos ATN consiste na sua natural incapacidade de expressar outros tipos de diálogos para além dos diálogos sequenciais. De facto, estando em cada momento um único estado activo ou em foco, diálogos assíncronos não podem ser especificados usando ATNs. No entanto, mesmo esta incapacidade expressiva foi parcialmente superada em [Jacob 86], onde são propostos pontos especiais de transferência de controlo entre subdiagramas, sendo tal transferência de controlo suportado por um mecanismo operacional tendo por base o modelo das *corrotinas*. Deste modo, diálogos assíncronos po-dem ser representados, o que é demonstrado através da construção de um pro-tótipo funcional de uma IU com tais características.

Cockton, em [Cockton 85], discute um conjunto de deficiências dos ATN, em particular relacionadas com a explosão de arcos e a forma de utilização de sub-redes, que conduziu à sua própria proposta de GTN, *Generative Transition Networks* [Cockton 90], alegadamente com o mesmo poder expressivo dos ATN mas com vantagens de economia, dado possuirem construções, designadas *geradores*, que declarativamente representam arcos (e nodos) sem necessidade de os enumerar. Esta notação não tem no entanto um suporte operacional, ou seja, não é executável.

Todavia, com os ATN a interacção é tratada ao nível do evento que produz a transição de estado pelo que, sendo a granularidade tão baixa, especificações tendem a tornar-se complexas, tanto mais que, na maioria dos casos, o diálogo é especificado como um todo. Os sub-diagramas são, em geral, a resposta para a falta de modularidade, implicando no entanto preocupações de consistência, dada a possibilidade de se realizarem invocações recursivas dos sub-diagramas com resultados difíceis de compreender.

A utilização dos ATN para a especificação de diálogos não é no entanto com-pletamente posta de parte. Eles são aplicáveis, tanto mais que possuem uma re-presentação diagramática associada, devendo no entanto ser encontrada a gra-nularidade correcta ou construções modulares externas de que estes fazem par-te. Deste modo, o fluxo de controlo entre os ATN seria controlado por um meca-nismo externo, com semântica operacional própria, especificando os ATN o flu-xo de eventos em cada componente.

Jacob em [Jacob 86] delega num mecanismo não óbvio, porque não explíci-to, de corrotinas o controlo das transferências de fluxo entre os ATN que repre-sentam os comportamentos internos de objectos da apresentação.

Os Guiões de Interacção que se propõem nesta tese, na linha da estratégia atrás enunciada, usam uma forma simples de ATN para especificar o comporta-mento interactivo interno a cada unidade modular - um *guião* -, permitindo de seguida que construções de mais alto nível, mas de semântica e sintaxe bem explícita, componham estas unidades em unidades de comportamento mais complexo. Os ATN especificam assim micro-diálogos,

sem problemas de explo-são de estados, e as construções de mais alto nível fazem a composição destes em diálogos que podem ser, por exemplo, paralelos ou multi-caminho, mas que têm uma expressão clara das suas características.

Gramáticas.

A perspectiva de que a interacção humano-computador é, fundamentalmente, um diálogo que se estabelece com base na utilização de duas linguagens - a do utilizador, ou de *input*, e a do sistema, ou de *output* -, cada uma delas com o seu léxico, a sua sintaxe e respectiva semântica, e a evidente necessidade da sua tradução, conduziu, naturalmente, a uma associação directa às bem estu-dadas técnicas de especificação de linguagens de programação e à utilização de *gramáticas independentes do contexto,* cuja sintaxe é descrita usando BNF [Naur 60], sendo tal perspectiva devida inicialmente a Foley em [Foley e Wallace 74].

Nesta abordagem, os símbolos terminais da gramática correspondem aos *tokens* de entrada, gerados pela camada de apresentação, sendo associados às acções do utilizador. A interpretação de um símbolo de entrada depende da sequência de símbolos de entrada anteriores. Esse contexto é especificado recorrendo à escrita de *produções,* nas quais símbolos não-terminais são associados a sequências particulares de símbolos terminais ou não-terminais. Cada produção especifica assim a estrutura de uma frase correcta da linguagem de entrada, ou seja, a correcta sequência de entradas do utilizador necessária à formulação da sua intenção, e ao despoletar da acção semântica corresponden-te[9].

Em [Reisner 81b], gramáticas independentes do contexto são usadas na descrição formal de *linguagens de acção* (cf. *action languages* na definição original) representando as acções físicas que o utilizador deve executar, usando os dispositivos fornecidos pela interface (cf. teclas, cursor, "joystick", etc.), para comunicar com o sistema interactivo. Condições de contexto e restrições semânticas são descritas informalmente sob a forma de anotações textuais adicionais em notas de rodapé. Sendo, como se referiu em capítulo anterior, as preocupações de Reisner mais centradas na área da análise da complexidade das linguagens de interacção e factores humanos do que na descrição formal de diálogos, conforme os seus trabalhos posteriores comprovam, o formalismo gramatical é usado fundamentalmente como mecanismo de especificação da linguagem de entrada, seu léxico e sua sintaxe, sem quaisquer preocupações com as ligações sintaxe-semântica. Outras propostas baseadas em gramáticas tais como CLG [Moran 81], GOMS

[9] Naturalmente, não se tratando neste contexto da geração de código, mas antes, assumindo um caso típico, da invocação de uma rotina da aplicação, o que, em qualquer caso implica a introdução de extensões à utilização de BNF standard.

[Card et al. 83] e TAG [Payne e Green 86], ainda que especifiquem a linguagem de diálogo, estão igualmente mais ligadas aos aspectos cognitivos e articulatórios.

É igualmente de sublinhar que a maioria destas abordagens de base gramatical são orientadas à linguagem de entrada (entradas do utilizador) sendo os resultados ("output") vistos como efeitos resultantes das *acções semânticas*. O primeiro trabalho onde é realizada a distinção entre a linguagem de entrada e a linguagem de saída, ou seja, entre a linguagem do utilizador e a linguagem do sistema, e o seu tratamento conjunto, foi apresentado por Shneiderman, em [Shneiderman 82], sob a designação de *Multiparty Grammars*. Nestas, uma úni-ca gramática especifica as duas linguagens, sendo estabelecidos pontos de mudança do controlo do diálogo, etiquetando as produções e misturando-as em função das mudanças de controlo do diálogo a especificar.

A necessidade de se introduzirem acções semânticas não apenas no final do reconhecimento de uma frase da linguagem de interacção, mas com uma maior granularidade, i.é., a meio de um reconhecimento, por forma a descrever-se o tão importante "feedback", faz com que outros formalismos gramaticais mais sofisticados devam ser empregues.

Por outro lado, os formalismos gramaticais não acomodam facilmente a descrição de interacção sensível ao contexto, dado descreverem a linguagem de interacção de uma forma estática. A gramática faz a descrição de todas as frases correctas da linguagem interactiva, sem no entanto descrever as condições em que as mesmas podem ou não ser aceites para reconhecimento. A possibilidade de se decorarem as produções com condições, ou *guardas*, contendo tal tipo de informação foi experimentada no formalismo designado *Gramáticas Interactivas Guardadas* [Martins et al. 90]. Uma implementação deste formalismo usando gramáticas de atributos possibilitou a prototipagem rápida de IU nele descritas. Porém, o resultado neste caso conseguido pode ser apenas considerado pontual, e apenas generalizável para IU puramente sequenciais.

Os formalismos gramaticais, não só pelas insuficiências já apontadas, mas também pela falta de clareza na expressão do fluxo de controlo, que pode ser até dependente do tipo de "parser", e porque não especificam diálogos assíncronos, são hoje em dia pouco ou nada utilizados na especificação de diálogos.

Modelos de Eventos.

A maioria dos actuais SGIU e a maioria das ferramentas para construção de tecnologia de interacção para IU gráficas baseiam-se no *modelo de eventos*. O modelo de eventos assume uma visão do diálogo com um sistema interactivo

muito baseada na ideia de que um sistema interactivo é, em essência, um sistema *reactivo* do tipo *evento-resposta*[10].

Os principais componentes do modelo são os *eventos* e os *processadores de eventos*[11]. Eventos resultam tipicamente de acções do utilizador realizadas sobre a IU. Para cada um dos possíveis eventos é definido um *processador de evento* que descreve as acções a realizar e as alterações a produzir nas várias variáveis que representam o estado da IU. Dado que a interpretação do evento depende do estado da IU à sua ocorrência, estes processadores de eventos assumem em geral a forma de uma estrutura multi-condicional, i.é., por casos.

As primeiras abordagens ao modelo reactivo para IU consideravam que os eventos poderiam ser tratados por *sistemas de produções*[12] [Hopgood e Duce 80], especificando-se as reacções de um componente do sistema aos eventos que era capaz de reconhecer por um conjunto de regras *condição-acção*, escritas de uma forma declarativa. Influenciado por este trabalho e procurando acrescentar modularidade à abordagem, o autor desenvolveu um formalismo baseado em sistemas de produções, os *Communicating Production Systems (CPS)* [Martins 87b] e depois uma linguagem de suporte - MEDUSA - [Martins 88b], linguagem que serviu de base à construção do sistema SPIOO [Pina 88], um sistema escrito em Smalltalk para a prototipagem e monitorização de IU, permitindo configura-ção dinâmica. Estas experiências baseadas no modelo de eventos e em sistemas de produções representando o comportamento de agentes interactivos, permiti-ram ao autor concluir pelo efectivo poder do modelo, enquanto modelo de implementação. Contudo, enquanto formalismo para a descrição do fluxo dos diálogos apresenta um grau de clareza e expressividade bastante baixo, apesar da facilidade da descrição, quase de tipo declarativo.

Green, em [Green 86], compara o poder expressivo dos ATN, gramáticas e modelo de eventos, concluindo que o modelo de eventos é o mais apropriado para ser usado como modelo de implementação, já que, sistematicamente, todos os outros podem ser neste convertidos. O modelo de eventos, ou seja, eventos, "event-loops" e "event-handlers", encontra-se implícito na maioria dos "toolkits" e ferramentas para a geração da IU anteriormente apresentados.

No entanto, o modelo de eventos enquanto modelo de descrição, apresenta diversas insuficiências. Por exemplo, se se pretender determinar que eventos podem despoletar uma dada acção, tal pode apenas ser feito pesquisando todos os processadores de eventos. Por outro lado, o conjunto de estados do sistema não é explícito, por ser distribuído. E se outras insuficiências descritivas pode-riam ser apontadas ao modelo de eventos, na

[10] *event-response system.*

[11] *event-handlers.*

[12] *production systems*.

perspectiva do autor o modelo é muito mais um modelo de implementação de diálogos (aliás bem sucedido) do que um modelo de descrição dos mesmos a alto nível. De facto, o modelo liga muito bem com a *tecnologia de objectos*, em cujo caso eventos são associados a mensagens para objectos, sendo estes capazes de os tratar no contexto do seu estado interno. A título de exemplo, um objecto do tipo "window"
pode aceitar e interpretar um conjunto de eventos relacionados com o seu redimensionamento, reposicionamento, fecho, etc. Esta ligação do modelo de eventos ao modelo de objectos tem a vantagem de proporcionar uma maior estruturação, fornecendo os objectos um grau de encapsulamento e localidade que permite melhor situar os pontos de tratamento dos eventos e dos seus efeitos, melhorando assim o ní-vel descritivo do modelo base.

Outros Modelos e Notações.

Para além dos modelos mais convencionais existentes em Ciências da Computa- ção e usados na especificação de diálogos, outros, mais particulares e menos standardizados, foram sendo desenvolvidos com tal objectivo.

Mallgren, em [Mallgren 83], usa álgebras de eventos na especificação formal de primitivas de saída em linguagens gráficas interactivas. Anderson, em [Anderson 85], usa o formalismo de especificação algébrica CLEAR [Burstall e Goguen 81], com extensões tais como expressões regulares, gramáticas e se-mântica denotacional, na definição de um formalismo de especificação de diálogos. Chi, em [Chi 85], compara diferentes métodos de base algébrica para a especificação de diálogos, acabando por considerar as "expressões de fluxo"[13] de Shaw [Shaw 80], que são extensões às expressões regulares, como dos mais expressivos ainda que, possivelmente, não executável. A utilização de Redes de Petri e suas extensões na especificação de diálogos é analisada em [Biljon 88] e, mais recentemente, Redes de Petri por Objectos [Bastide e Palanque 90] foram propostas para especificação de diálogos assíncronos, sendo em [Viegas 94] apresentado um sistema para a sua geração automática, partindo de uma des-crição da rede que é construída usando uma interface gráfica.

UAN (*User Action Notation*) [Siochi e Hartson 89] é, como o próprio nome indica, uma notação para descrição das acções físicas, a nível de dispositivos, que um utilizador deve realizar durante a interacção. Os resultados de cada uma das acções atómicas do utilizador que produzem efeitos ao nível do estado da IU são também descritos de uma forma abstracta. UAN é uma das poucas notações de descrição do *diálogo externo*, podendo, dada a sua

[13] *flow expressions*

simplicidade, ser usada quer por utilizadores[14] quer por implementadores. UAN não estabelece qualquer ligação à camada computacional, sendo apenas uma notação de des-crição.

Marshall, em [Marshall 86], apresenta talvez o primeiro trabalho onde se juntam preocupações de associar descrições de diálogo com descrições formais da camada computacional. Marshall usa uma notação gráfica para a descrição do fluxo do diálogo, os "statecharts" de Harel [Harel 87], e VDM para a descrição da aplicação. Os "statecharts" têm por construção básica uma "box" que pode ser uma operação, a composição de operações, iteradores, estados de "wait", etc., sendo estas "caixas" ligadas entre si de forma sequêncial, e compostas em "caixas" de mais alto nível. As "caixas" ditas *terminais* são depois associadas a operações da aplicação. A notação não possibilita a expressão de diálogos multi-caminho, mas extensões em tal sentido podem ser facilmente acrescentadas.

Marshall associa ainda aos diagramas um conjunto de variáveis globais que servem para representar o estado da IU e as entradas do utilizador. Esta separação apresenta-se-nos como vantajosa. Porém, sendo em VDM o estado da aplicação global, ainda que cada operação declare a que variáveis deste acede para escrita ou leitura, o método não estabelece uma ligação clara entre acções de diálogo, i.é., do utilizador, e os respectivos efeitos na camada computacional.

A proposta de Marshall, ainda que sendo interessante e em conformidade com algumas das nossas próprias preocupações quanto à necessidade de se desenvolverem contextos que suportem a concepção integrada da IU e da aplicação, acaba por não apresentar soluções claras para vários problemas que são, neste sentido, cruciais, designadamente: a granularidade a que se trata o diálogo; modularidade e composicionalidade das estruturas interactivas básicas; descrição clara dos pontos onde se realiza a interferência entre camadas e seus efeitos numa e noutra e, finalmente, se se pretende método, regras de concepção e de transferência de informação. Anotam-se aqui tais observações já que, neste caso, o que é proposto não é simplesmente uma notação de descrição, como as que atrás se analisaram, mas já o esboço de um método integrador.

SPI[15] [Alexander 87] é uma linguagem de especificação e prototipagem de controladores de diálogo onde a estruturação de eventos é realizada usando CSP (cf. a linguagem *eventCSP* desenvolvida) e a camada computacional especificada em *metoo* [Henderson e Minkowitz 86]. Sequenciação, iteração, escolha e parale-lismo são contemplado na notação que tem poder expressivo para especificar diálogos ramificados. A ligação à camada computacional é

[14] Um estudo analítico realizado pelo autor a nível de alunos de Mestrado demonstrou, pelos resultados apresentados, não só a facilidade de compreensão da notação, como a capacidade desta de expressar diferenças de complexidade articulatória entre ferramentas.

[15] *Specifying and Prototyping Interaction*.

neste método descri-ta em estruturas da linguagem *eventISL,* escritas em *metoo,* onde se declaram as condições de aceitação dos eventos, as alterações produzidas no estado, entra-das e saídas, sendo a associação entre eventos e alterações de estado mais ex-plícita e mais clara que no método anterior. Para além de protótipos na lingua-gem *metoo,* protótipos em C podem também ser construídos.

Uma ferramenta para a prototipagem da linguagem *eventCSP* está também disponível para que se possam analisar os *traços* de eventos, ainda que, na óptica do autor, sem grande utilidade dada a inexistência de contexto, ou seja, sem consideração das restrições que surgem ao considerar-se o estado da aplicação. A notação herda ainda algumas das reconhecidas insuficiências do CSP, designadamente a origem e destino dos eventos, que deveriam facilmente ser reconhecidos como devidos a entradas externas, ou para sincronização de processos ou originados por outros eventos. O SPI é, no entanto, uma das abordagens mais completas para a especificação de diálogos, possuindo ainda a vantagem de ser executável. No entanto, nenhum método foi à mesma associado o que se traduz, em termos práticos, por alguma dificuldade de utilização.

Refira-se finalmente a *linguagem de agentes* de Abowd [Abowd 90], na qual *agentes* são os elementos básicos de interacção, sendo um sistema interactivo modelado como um conjunto de agentes sequenciais comunicantes. Cada agente tem, neste caso, especificado o seu comportamento interno, ou seja, o conjunto dos seus possíveis estados internos e operações, o que é feito numa extensão ao Z, a especificação do conjunto de canais de comunicação para entrada e saída, e a especificação do comportamento externo, ou seja, os eventos em que o agente pode participar, realizada usando CSP. A linguagem garante composicionalidade dos agentes e combinação independente, por intercalação[16], ou dependente, por sincronização.

O formalismo é, em resultado destas características, muito expressivo, mas, conforme Abowd afirma, é de momento assumidamente não executável, sendo assim apenas utilizável como ferramenta formal de análise. De facto, diversas propriedades dos sistemas interactivos têm sido estudadas usando a *linguagem de agentes*. A especificação formal, *a posteriori,* de sistemas interactivos, sob a forma de sistemas multi-agente, tem sido outra das grandes áreas de aplicação do formalismo [Abowd 91].

No entanto, a modelação de sistemas interactivos com base nestes agentes apresenta algumas idiossincrasias, em resultado de uma perspectiva demasiado distribuída quer dos dados quer do controlo. Esta observação não se aplica aos modelos de objectos, nos quais, para além dos *objectos* (ou *actores*), existem mecanismos semânticos tais como herança e polimorfismo. Nos modelos planos de agentes, ou seja, sem hierarquia, a distribuição é

[16] *interleaving.*

apenas horizontal, pelo que o estado da aplicação é disperso pelos mini-estados internos dos agentes de uma forma difícil de compreender. A divisão do sistema em agentes apresenta por vezes divisões difíceis de compreender. Por exemplo, e seguindo exemplos do próprio Abowd, num sistema de gestão de "janelas" podemos encontrar agentes tais como *button*, *keyboard*, *MouseMove*, *Window*, *WindowState*, *OpenClose*, *Resize*, etc. Aceitam-se as razões para este tipo de divisão do sistema. Porém, dada a complexidade destes formalismos de agentes e processos, parecem ser indispensáveis regras apropriadas para a sua fácil e correcta aplicação, de mo-mento inexistentes.

4.5.- Prototipagem Rápida em IHC.

A técnica da chamada *prototipagem rápida* é, quaisquer que sejam as suas formas de realização, uma técnica imprescindível em Engenharia da Programa-ção, pois permite que os requisitos funcionais dos sistemas possam ser iterati-vamente ajustados com o cliente final, numa fase inicial de projecto. Assim, uma técnica de "baixo custo" poderá permitir, pela rápida detecção de erros na fase de captura de requisitos, evitar mais onerosos custos, dado que, quanto mais cedo no ciclo de vida do desenvolvimento das aplicações os erros puderem ser encontrados, menor custo terá a sua recuperação. Os mesmos argumentos podem ser aplicados à utilização de protótipos em IHC, considerando agora a figura do utilizador final em vez da do cliente. Myers, em [Myers 92], indica até que em resultado de um estudo exaustivo foi possível concluir que 87% das IU das aplicações interactivas são desenvolvidas de forma iterativa, e que a percen-tagem de tempo de desenvolvimento da IU no tempo total de desenvolvimento do sistema se situará entre 50% e 60%.

As abordagens formais não podem, nem têm tal objectivo, garantir só por si boas soluções "à primeira". Uma abordagem iterativa na concepção dos sistemas interactivos, e em particular das IU, é pois indispensável, e muitos são os auto-res que a defendem. Alguns, cf. [Lewis 90], acreditam mesmo que apenas as téc-nicas de prototipagem rápida de IU podem conduzir ao aumento da usabilidade de sistemas interactivos[17].

A figura 4.2 procura mostrar qual a interpretação correcta a dar à ideia de se construirem protótipos da IU. A questão que se coloca é, basicamente, o que se deve entender por um protótipo de uma IU. A maioria das abordagens à pro-totipagem de IU, mesmo a nível de grandes construtores de equipamentos e fabricantes de “software”, consiste no desenvolvimento de uma IU cuja apresen-tação seja já semelhante à que o utilizador final desejaria ter no seu sistema in-teractivo. Porém, e aceitando esta

[17] Lewis classifica esta escola como a escola do *“process is paramount”* sublinhando a atenção dada à melhoria dos processos de concepção.

perspectiva, um erro enorme está a ser come-tido, e que resulta do facto de que não é, em geral, a apresentação da IU, só por si, que determina a sua aceitação ou rejeição, mas a ligação entre a apresenta-ção e a informação por esta fornecida. Se recordarmos o ciclo de Execução-Avaliação de Norman em ligação ao referencial de Abowd, apresentados no capítulo 2, verifica-se que a informação transmitida pela apresentação pode ser mais importante semanticamente que a própria apresentação.

A questão seguinte consiste, então, em determinar-se qual a informação a dar ao utilizador, com alguma independência da respectiva forma, e donde tal informação pode ser obtida.

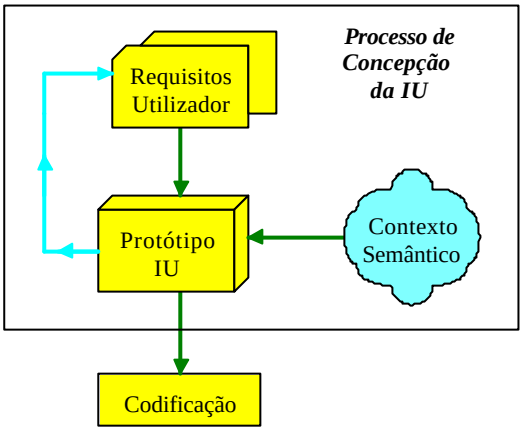

Fig. 4.2 - Processo de Concepção da IU.

O autor defende que se à utilização de métodos ou formalismos de descrição de IU se puderem associar técnicas de prototipagem, executando tais descrições num dado contexto, conforme se procura ilustrar na figura 4.2, se encontram reunidas as condições para uma verificação estática, e também para uma verifi-cação dinâmica, de requisitos de interacção de forma exemplar. A aplicação de métodos formais permite que, ainda numa fase muito inicial do projecto, se pos-sam recolher as informações fundamentais sobre o contexto semântico, de modo a que a construção do protótipo da IU não se limite a uma prototipagem da apresentação (ou seja apenas afinação do “look”) mas, o que é muito mais rele-vante, contemple também aspectos de sensibilidade da IU ao contexto (ou seja o “feel”). Por exemplo, sem informação semântica não é possível construir um protótipo de uma IU na qual os menus possam ter, desde logo, opções dinami-camente activadas e desactivadas com o decorrer da interacção. Também não é possível programar correctamente a interacção com uma "caixa de diálogo" sem serem, à partida, conhecidas as restrições associadas aos valores dos objectos que, com esta, se estão a criar. Os argumentos de um comando são em geral introduzidos por uma ordem determinada pela construção da IU,

podendo no entanto esta ordem ser flexibilizada, ou não, resultando este tipo de informação também da especificação formal.

Em resumo, o contexto semântico implícito na especificação da aplicação, ainda que este seja apenas uma abstracção do contexto final construído após refinamento, é uma informação auxiliar indispensável para uma efectiva e mais adequada utilização da técnica de prototipagem da IU. Os princípios de concep-ção advogados pelos peritos de factores humanos poderão ser de imediato verifi-cados sobre uma IU que, desde já, apresenta um comportamento igual ao que terá no final do desenvolvimento do sistema. Modificações numa qualquer das camadas, por força de modificações de requisitos, são, caso um método sistemá-tico seja associado à ideia, facilmente mantidas coerentes entre si. Alguma tec-nologia deverá ser construída no sentido de facilitar este processo.

Esta é a abordagem que nesta tese se propõe, apoiada por um método que, tendo por base a especificação formal da camada computacional, vai permitir que a prototipagem de IU se faça exactamente segundo os requisitos que atrás foram enunciados.

4.6.- Sumário.

Apresentou-se neste capítulo uma revisão da literatura em métodos formais aplicados à disciplina de IHC. Foram feitas considerações sobre as vantagens da aplicação desses métodos aos diversos problemas existentes na área de IHC. Domínio da complexidade, clareza, coerência, completude, rigor, possibilidade de raciocínio, etc., são muitos dos atributos intrinsecamente associados ao seu emprego e que são automaticamente herdados.

Podem reconhecer-se, após a análise dos objectivos e perspectivas dos vari-ados trabalhos realizados em IHC empregando métodos formais, os três grandes eixos de utilização que se representam na figura 4.3.

Em primeiro lugar é de distinguir o eixo que representa a relação temporal entre a especificação e o especificado. Num extremo do eixo, a especificação é posterior à criação do especificado, sendo pois uma *descrição rigorosa*. A grande maioria dos trabalhos anteriormente referidos caem nesta categoria, podendo em muitos casos considerar-se que são trabalhos de "reverse engineering"[18]. No outro extremo do eixo, temos as especificações realizadas antes da construção do objecto especificado, e que constituem *prescrições rigorosas* (engenharia directa). A especificação de um controlador do diálogo, por exemplo, pode ser vista como a prescrição da sua funcionalidade e do seu comportamento. Estas especificações são documentos de grande importância para o implementador.

[18] engenharia reversa.

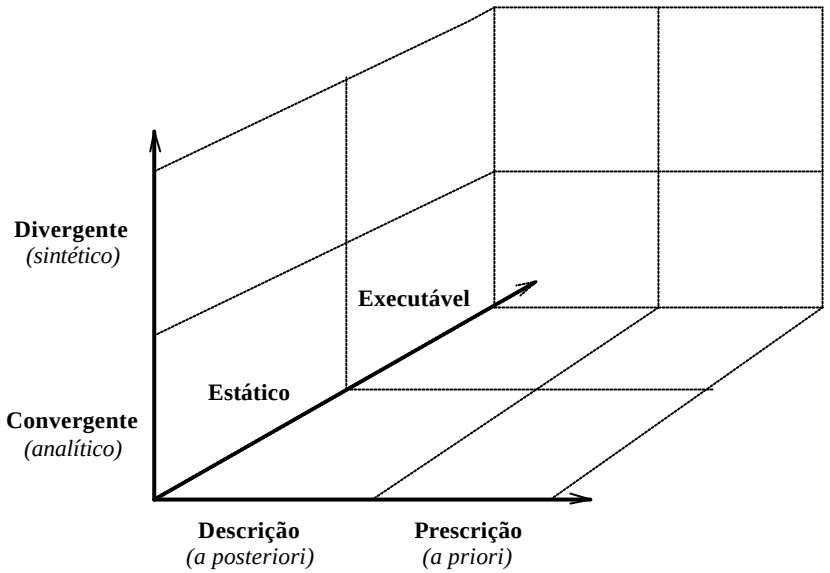

Fig. 4.3 - Aplicação de Métodos Formais em IHC: Objectivos.

Num outro eixo distingue-se o grau de abstracção, e consequente utilização da especificação produzida, distinguindo-se entre especificações *convergentes* (ou *analíticas)*, e *divergentes* (ou *sintéticas*). As *analíticas* são especificações de alto nível de abstracção, centradas num número conciso de propriedades, tendo a possibilidade de, sem interferências, permitirem estudos analíticos precisos. A maioria dos modelos cognitivos e de competência (cf. Reisner, TAG, CLG, etc.) são especificações analíticas. Modelos como PIE e REDPIE são simultaneamente analíticos e prescritivos.

Especificações *sintéticas* são as que expressam de uma forma construtiva a concepção de um sistema particular, constituindo, em geral, a base para o seu desenvolvimento. Em geral, especificações sintéticas são também prescritivas.

O terceiro eixo representa a possibilidade do formalismo de especificação ser ou não executável e, portanto, a possibilidade de se construírem protótipos dos sistemas especificados. Por exemplo, em SPI são produzidas especificações pres-critivas, sintéticas e executáveis. Na intersecção destas três características po-deremos encontrar as condições necessárias para o desenvolvimento de métodos para a construção de sistemas interactivos.

É ainda de notar que, do ponto de vista dos sistemas, são dois os principais objectos das especificações: a *funcionalidade* e o *comportamento*. Duas classes de formalismos para a especificação da funcionalidade de um sistema são em geral empregues. Os métodos baseados em modelos, como VDM, Z e CAMILA, e os algébricos, como CLEAR e Larch. Especificações baseadas em modelos são mais adequadas para a especificação de sistemas interactivos com estruturas de dados complexas e um conjunto de operações relativamente simples, como por exemplo editores de texto, bases de dados

ou sistemas CASE, enquanto que os métodos algébricos, dado representarem o estado implicitamente nas operações, são mais adequados para aplicações de características inversas. Em qualquer dos casos a funcionalidade é expressa indicando como as operações geram ou manipulam os valores do tipo de dados.

O comportamento de um sistema, definido como o conjunto das sequências possíveis das suas transições de estado, tanto pode ser especificado num forma-lismo baseado em estados, onde operações são correspondências entre estados e têm a si associadas pré e pós condições, como usando processos e eventos em que estes podem participar, abstraindo das transições internas de estado e im-pondo restrições nas sequências de eventos (*traço*s) possíveis.

Notações combinando especificação da funcionalidade com especificação de comportamento, e não procurando inferir comportamento da funcionalidade, têm sido empregues em IHC (cf. SPI, Agentes de Abowd, LOTOS, etc.).

É ainda importante salientar que o emprego de métodos formais no desenho de sistemas interactivos não implica, só por si, que se passe a ter um método com tal objectivo. Como se viu, muitas são as notações com semântica rigorosa desenvolvidas para a descrição de sistemas interactivos. Porém, em geral, regras e directivas para o seu emprego, em particular considerando a camada compu-tacional não são apresentadas. Ou seja, apresentam-se notações, mas não são apresentados métodos que facilitem a sua utilização.

Dijkstra, em [Dijkstra 76], chama a atenção para este facto, transmitindo a ideia de que, ainda que notações possam ser equivalentes em poder expressivo, o que é determinante para o seu sucesso é a facilidade com que os interessados as podem compreender e empregar.

Finalmente, é de salientar a quase total inexistência de especificações da apresentação, excepto, como se referiu, alguns trabalhos na área dos sistemas de "janelas" e relacionados com menus. Uma explicação desta lacuna pode ser baseada em duas ordens de razões. Em primeiro lugar, porquê especificar o que já se tem implementado em numerosas "ferramentas"? Em segundo lugar, uma especificação formal da apresentação tem uma semântica de tão baixo nível (cf. coordenadas, atributos de cor, etc.) que quase coincide com a implementação. É, no entanto, indiscutível, eventualmente por razões como as atrás sugeridas, que o “output” tem sido esquecido nas preocupações de aplicação de métodos formais, ou simples notações, em IHC.