José Eduardo Moreira Fernandes

Conceitos em Sistemas de Informação: UML e a sua adequação ao FRISCO

Universidade do Minho

2002

José Eduardo Moreira Fernandes

Conceitos em Sistemas de Informação: UML e a sua adequação ao FRISCO

Dissertação submetida à Universidade do Minho para
obtenção do grau de Mestre em Sistemas de Informação

Universidade do Minho

2002

Resumo

A área de Sistemas de Informação tem sido invadida de conceitos mal definidos e terminologia mal empregue, juntamente com metodologias e linguagens não solidamente fundamentadas. Tal tem sido motivo de preocupação para a comunidade de sistemas de informação e alvo de estudos no sentido de analisar, sugerir ou propor formas mais correctas e fundamentadas de expressar o conhecimento e de estabelecer a comunicação.

O relatório FRISCO ("Framework of Information System Concepts") realizado pelo "IFIP WG 8.1 Task Group FRISCO" é um destes estudos, no qual se propõe um esquema de conceitos entendidos como relevantes para a área de sistemas de informação, os quais permitam, juntamente com uma terminologia adequada, uma comunicação mais clara, efectiva e não ambígua na área sistemas de informação. Paralelamente, a linguagem UML ("Unified Modeling Language") surgiu e tornou-se como um standard na modelação de sistemas orientados a objectos, a qual sendo destinada essencialmente para a modelação de sistemas de software, advoga na sua especificação a flexibilidade para ser usada para a modelação de outros sistemas que não de software.

Este estudo pretende analisar até que ponto a linguagem UML é adequada para a descrição/modelação de sistemas de informação, com base no esquema de conceitos de sistemas de informação proposto pelo FRISCO.

É feita uma breve introdução a como o UML organiza e apresenta o seu metamodelo e, no que respeita ao FRISCO, também é feita uma breve apresentação focando os pressupostos ontológicos que sustentam a sua visão ontológica e a forma como foi construído o seu esquema de conceitos propostos. Para além de uma reflexão sobre várias questões que levantam dificuldades em considerar como suficiente a adequação do UML ao FRISCO, procede-se neste estudo a à análise da adequação do UML para representar cada um dos conceitos propostos pelo FRISCO. Desta análise é apresentada uma síntese das conclusões.

Este trabalho enquadra-se nos domínios de ontologias, sistemas de conceitos e linguagens de modelação orientadas a objecto, e constitui um contributo para o esclarecimento de questões relativas a conceitos utilizados na modelação de sistemas de informação.

Abstract

The Information Systems area has been invaded with no sound basis concepts and terminology jointly with methodologies and languages. This has been a motive of concern to the information systems community and several studies are being made in order to analyse, suggest or recommend more correct and well-founded ways of expressing knowledge and establishing communication.

The FRISCO ("Framework of Information System Concepts") report of the IFIP WG 8.1 Task Group FRISCO is one of these studies; it suggests a scheme of concepts viewed as relevant to the information systems area, which may enable, together with the correct terminology, a more clear, effective and not ambiguous communication. At the same time the UML ("Unified Modeling Language") language appeared and became standard in the modelling of object oriented systems which aims essentially to the modelling of software systems and has in its specification the flexibility to be used in the modelling of other systems.

This study has the intention of analysing how well is the UML language suited to be used in the description/modelling of information systems with its basis on the scheme of concepts considered relevant by FRISCO.

A brief introduction is done to the organization and to the presentation of the UML metamodel in its specification's document and, on what concerns to FRISCO, an introduction is also done focusing the ontological assumptions that sustain its ontological vision and the way the FRISCO built the schema of proposed concepts. Beyond a reflection about several issues that arise difficulties in accepting the UML as compatible to FRISCO's concepts, this study proceeds to the analysis of how well UML is suited to represent each of the FRISCO proposed concepts. From this analysis, a synthesis is elaborated.

This study is a contribution to the elucidation of questions concerning the usage of concepts in information systems' modelling.

Agradecimentos

Aos meus pais e ao meu irmão que sempre me deram o incentivo e o apoio para concretizar este objectivo e, sem os quais, tal não teria sido viável. Á minha querida Ana, pelo carinho, pela ternura, pela atenção e incentivo.

Ao meu orientador Prof. João Álvaro Carvalho que pela enorme paciência, pela sua pronta atenção, pela sua participação no esclarecimento de questões e de caminhos a percorrer e, pela sua lucidez, a qual contribuiu para a minha evolução a nível de correcção de pensamento e de expressão.

A todos aqueles que com gosto me desejaram bem e me concedem o gosto da amizade.

Com sinceridade, Obrigado.

Índice

Lista de Figuras

Lista de Quadros

1. Introdução

Esta dissertação, no contexto de uma preocupação e interesse crescente com conceitos e terminologia subjacentes às linguagens de modelação, aborda a questão da adequação das linguagens de modelação a um esquema de conceitos solidamente fundamentados. É objectivo deste trabalho analisar o quanto bem o UML ("*Unified Modeling Language"*), como linguagem de modelação, se adequa para a representação de sistemas de informação à luz do esquema de conceitos proposto no relatório[1] FRISCO ("*Framework of Information System Concepts"*) [FRISCO98].

1.1. Motivações e Enquadramento

Reconhecendo a importância do FRISCO como um estudo de referência no estabelecimento de um esquema sólido e bem fundamentado de conceitos de Sistemas de Informação e, do UML como a linguagem de modelação standard com forte aceitação no mundo académico e na indústria, o motivo deste trabalho prende-se com o facto do esquema de conceitos proposto pelo FRISCO carecer de linguagens de modelação que o ponham em prática, verificando-se a ausência de linguagens de modelação de relevo que directamente o suportem.

O esquema de conceitos do FRISCO resulta do trabalho iniciado em 1988 e concluído nos finais de 1996 pelo IFIP WG 8.1 Task Group FRISCO[2] [FRISCO98, pág.1] e, foi motivado por uma crescente preocupação sobre a situação na área de sistemas de informação, na qual se verifica a existência e utilização de conceitos desapropriados e mal definidos. Em resposta a esta preocupação, este grupo desenvolveu esforços no sentido de fornecer uma estrutura conceptual, isto é, definições (onde possível) não ambíguas, simples e claras e uma terminologia para os conceitos mais fundamentais no campo de sistemas de informação, incluindo noções de informação e comunicação, de organização e de sistema de informação.

No domínio das linguagens de modelação, encontra-se a linguagem UML ("*Unified Modeling Language")*, a qual representa nos dias de hoje a principal linguagem de modelação orientada a objectos comummente aceite. Tendo sido normalizada pelo grupo OMG[3] ("*Object Management Group*"), é uma linguagem que pretende ser aplicada em vários domínios, na modelação de uma diversidade de tipos de sistemas, incluindo a modelação de negócios. Com a aceitação do

[1] O relatório FRISCO foi originalmente finalizado em Dezembro de 1996 e disponibilizado na web para download para fins educacionais e de investigação [FRISCO98].

[2] Grupo proposto em 1987 e estabelecido em 1988.

[3] O "Object Management Group (OMG)" é definido como "(…) *an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications*".

UML, emergiu uma nova geração de ferramentas, técnicas e processos que usam a linguagem UML.

Faz-se notar que o esquema de conceitos do FRISCO e a linguagem UML tiveram para o seu desenvolvimento, abordagens distintas. Por um lado, o FRISCO com a preocupação de rigor, coerência, e sólida fundamentação teórica; por outro lado, o UML como uma linguagem que tem vindo a emergir fruto de contributos diversos e que só mais recentemente tem vindo a ser objecto de atenção e cuidado "científico".

Portanto, torna-se imperioso que face à importância e contributo do FRISCO, e à importância e utilização/expansão crescente do UML, se analise a adequação desta linguagem ao esquema de conceitos de sistemas de informação proposto pelo FRISCO. A Figura 1.1 procura ilustrar o objectivo deste trabalho.

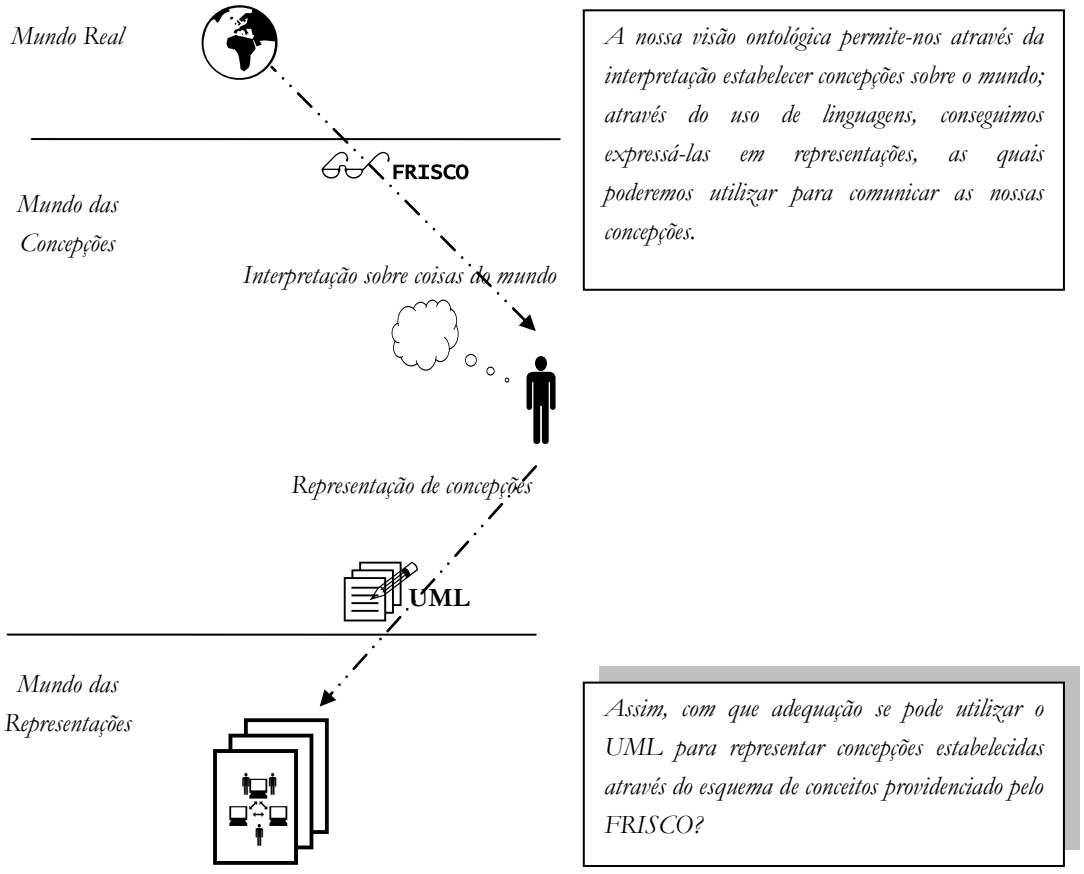

Figura 1.1 – A adequação do UML como linguagem para representação de conceitos do FRISCO

1.2. Metodologia de Trabalho

Para a concretização deste estudo, optou-se como metodologia de trabalho começar por conhecer o relatório realizado pelo "IFIP WG 8.1 Task Group FRISCO" [FRISCO98]. A aquisição deste conhecimento, envolve entre outras, a compreensão dos pressupostos que traduzem a visão ontológica estabelecida pelo FRISCO, e sobre a qual, são definidos os vários conceitos que compõem o esquema de conceitos do FRISCO.

A necessidade de compreender a linguagem UML impõe que (em virtude da complexidade subjacente), antes de se analisar o seu documento de especificação [OMG-UML01] e respectivo metamodelo, se reveja a literatura relativa à apresentação e utilização desta linguagem. Após esta revisão, procede-se ao estudo do documento de especificação do UML, com o objectivo de compreender o metamodelo fornecido, e os respectivos elementos que o compõe.

No sentido de aprofundar o conhecimento relativo a questões respeitantes a conceitos de sistemas de informação e ao estudo da adequação de uma linguagem a um esquema de conceitos, procede-se à revisão de literatura relevante.

Uma vez concretizada a revisão de literatura, a construção de conhecimento e, tendo presente a diferença de abordagens que levaram ao desenvolvimento do esquema de conceitos do FRISCO e da linguagem UML, procede-se a uma reflexão sobre várias questões que aparentam levantar dificuldades a uma adequação satisfatória do UML ao FRISCO.

Após breves considerações sobre estas questões, define-se a abordagem a ser empreendida na análise da adequação dos vários elementos do UML para representação dos vários conceitos que constituem o esquema do FRISCO. Uma vez realizada esta análise de adequação, elabora-se uma síntese dos resultados obtidos, sobre a qual se tecem considerações.

Finalmente, é feita a conclusão deste trabalho, na qual se debruça sobre o trabalho realizado, sobre as dificuldades, e sobre possíveis trabalhos a serem realizados no seguimento deste.

1.3. Estrutura da Tese

Este documento encontra-se estruturado em seis partes:

(1) *Introdução*, em que se menciona os objectivos, enquadramento, e metodologia do trabalho;

(2) *Conceitos de sistemas de informação*, em que se apresentam vários conceitos relativos a sistemas de informação e linguagens de modelação;

(3) *Breve apresentação do FRISCO*, com introdução ao trabalho desenvolvido e à forma como se chegou a um esquema de conceitos;

(4) *Breve apresentação do UML e seu metamodelo*, na qual se foca a origem do UML e a forma como se encontra organizado e expresso o metamodelo da linguagem;

(5) *Análise da adequação do UML ao FRISCO*, onde após breves considerações pertinentes a este estudo, se procede ao estudo da adequação dos elementos do UML aos conceitos do FRISCO, sintetizando e reflectindo sobre os resultados obtidos;

(6) *Conclusão*, na qual se comenta o trabalho desenvolvido, as dificuldades sentidas e, os trabalhos futuros que podem ocorrer em sequência deste trabalho realizado.

Este documento contém os anexos:

A. *Análise realizada para cada um dos conceitos do FRISCO* – anexo onde se apresenta o raciocínio desenvolvido e consequentes conclusões relativas a cada elemento do FRISCO e eventual elemento(s) UML que o possa(m) adequadamente representar.

B. *Pressupostos, apresentação informal e formal dos conceitos do FRISCO* – anexo que constitui uma sucinta compilação para consulta rápida dos elementos relevantes para este estudo.

C. *Pacotes e metamodelo do UML* – anexo que dá a conhecer o metamodelo do UML, seus elementos e a forma como se encontra organizado.

2. Conceitos de Modelação de Sistemas de Informação

Na área de sistemas de informação não existe uma referência conceptual e terminologia comummente aceite que possa ser aplicada para a definição de novos conceitos ou explicação dos conceitos existentes [FRISCO98]. Estudos efectuados revelam que *"os sistemas de conceitos das abordagens de desenvolvimento de sistemas de informação raramente estão articulados, o que contribui para a falta de integração de modelos parciais que constituem um modelo completo de sistema de informação"* [Carvalho94].

Em sintonia com estas preocupações, alguns críticos reclamam que a orientação a objectos, a qual constitui na actualidade o paradigma de crescente relevo, está menos adequada para a modelação de domínios de problema do desenvolvimento de sistemas de informação porque as abstracções fornecidas estão talhadas para representar artefactos de software e não para coisas ou aspectos do domínio do problema [Opdahl99].

2.1. Sistemas de Informação

O que se entende por "sistema de informação"? Este termo, tão fundamental na área de TSI (Tecnologias e Sistemas de Informação), pode ser interpretado de forma diferente por diferentes grupos de pessoas: como um sistema técnico implementado por tecnologia computacional; como um sistema social, tal como uma organização com as suas necessidades de informação; ou como um sistema conceptual que engloba quer um sistema técnico quer sistema social [FRISCO98].

Carvalho [Carvalho99], baseado nas definições de informação e de sistema, conclui que um sistema de informação pode ser um objecto activo que lida com (processa) informação, ou um objecto cujo propósito é informar, ou uma combinação de ambos, ou seja, um objecto que lida com informação e com propósito de informar. Considerando estas interpretações, argumenta que existem quatro tipos de objectos de interesse na área de TSI, os quais podem ser legitimamente considerados como sistemas de informação [Carvalho99], sendo cada um útil para diferentes actividades profissionais:

1) Organização (sistemas autónomos) cujo negócio (finalidade) é informar os seus clientes (SI1 na Figura 2.1);
2) Um subsistema que assegure a comunicação entre os subsistemas de gestão e operacional num sistema autónomo (SI2 na Figura 2.1);

3) Qualquer combinação de objectos activos que lidam apenas com informação e cujos agentes são computadores ou dispositivos baseados em computadores – um sistema baseado em computador (SI3 na Figura 2.1);

4) Qualquer combinação de objectos activos (processadores) que lidam apenas com informação (SI4 na Figura 2.1).

A figura seguinte esquematiza estes conceitos:

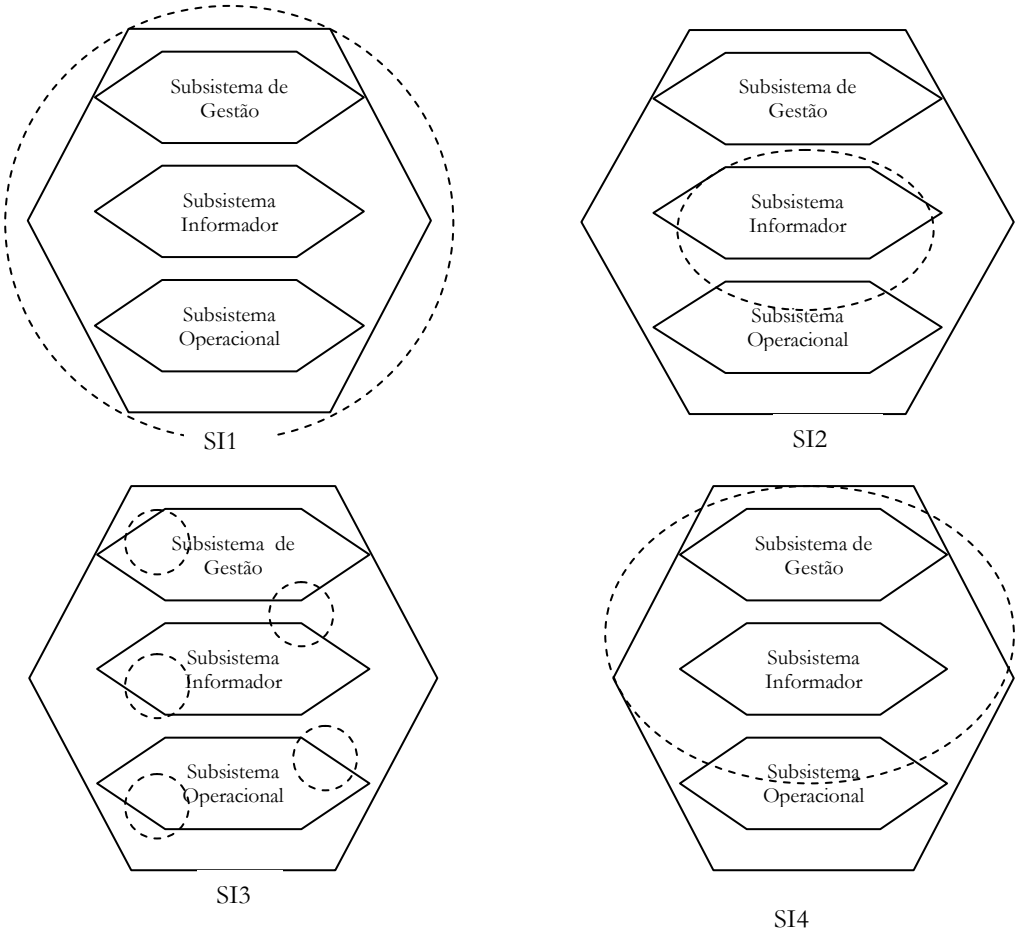

Figura 2.1 – Objectos que podem ser considerados sistemas de informação (adaptado de [Carvalho99])

Carvalho observou que, tendo em conta os vários tipos de objectos que se podem considerar como sistemas de informação, uma análise de algumas definições de sistema de informação presentes na literatura, revela que estas definições correspondem a mais do que um objecto; e conclui: "*embora em alguns casos não seja problemático, noutros conduz a uma confusão sobre o que é objecto de interesse de uma actividade profissional ou de um domínio científico. Mais ainda, esta confusão contribui para a dificuldade em estabelecer uma terminologia consensual na disciplina de TSI, o que em consequência contribui para a dificuldade de estabelecer um conhecimento fulcral partilhado da disciplina*". [Carvalho99]

Correspondendo às diferentes interpretações de sistema de informação, existem vários dialectos diferentes da nossa linguagem profissional; pessoas que aparentam interpretar sistema de informação do mesmo modo, usam conceitos e terminologia diferentes para o explicar. Por exemplo, o termo informação pode representar dados, ou a interpretação de uma mensagem, ou o conhecimento ganho aquando a recepção dos dados expressando a mensagem [FRISCO98].

As raízes do problema de diferentes significados e terminologias residem no passado, devido a diferentes facetas da área terem sido lidadas por diferentes disciplinas científicas ou "culturas", em particular as ciências de computação e ciências sociais, existindo pouca comunicação entre estas culturas. No desenvolvimento de sistemas de informação fortemente baseados em computadores, profissionais ligados às áreas computacionais negligenciaram os aspectos organizacionais, cognitivos e sociais essenciais do desenvolvimento de sistemas de informação, tendo dificilmente estado cientes do papel central da informação e da comunicação nas organizações. Por outro lado, os profissionais de áreas sociais e de organização têm sido frequentemente inconscientes da importância de considerar os aspectos formais (mais essenciais) do desenvolvimento de sistemas de informação e portanto, vagueado sem objectividade onde maior precisão seria necessária [FRISCO98].

Perante a situação acima descrita, conclui-se que, para uma comunicação correcta na área de sistemas de informação e propiciadora do desenvolvimento de métodos e linguagens mais apropriados à modelação de sistemas de informação, torna-se necessário esclarecer os conceitos relevantes e introduzir alguma clareza terminológica a esta diversidade.

2.2. Modelação de Sistemas de Informação

2.2.1. Motivos da Modelação

Para sistemas complexos, recorre-se à construção de modelos porque existem limites à capacidade humana de compreender e lidar com a complexidade; não se consegue compreender um sistema na sua totalidade de uma só vez. Assim, através da modelação, estreita-se o problema focando apenas um aspecto de cada vez; a abstracção, ênfase em detalhes relevantes enquanto ignorando outros, é a chave para a aprendizagem e comunicação.

Em particular, a modelação constitui uma parte central de todas as actividades que levam ao desenvolvimento de bom software. Constroem-se modelos para comunicar entre as partes interessadas, a estrutura e o comportamento desejado do sistema, para visualizar e controlar a arquitectura do sistema, frequentemente expondo as oportunidades para simplificação e reutilização [Booch99].

2.2.2. Esquema de Conceitos

Embora os actuais modelos e métodos de desenvolvimento empreguem técnicas poderosas e sofisticadas, alguns aspectos fundamentais da análise e desenho de sistemas de informação parecem em falta. O relatório FRISCO considera que uma das razões prende-se pelo facto dos objectivos básicos e dos princípios de sistemas de informação não terem até à altura sido analisados com profundidade suficiente, alegando que a investigação académica tem-se concentrado mais em problemas formalizáveis do que em problemas que são relevantes de uma perspectiva social e económica nas organizações. Considera igualmente que a metodologia orientada à implementação tem-se centrado mais em torno de técnicas usadas na produção eficiente de sistemas de informação, em vez de se concentrar na produção efectiva de sistemas de informação com base no impacto final destes no negócio: a adição de valor dentro da organização [FRISCO98, pág. 15].

O FRISCO consciente da problemática relativa a conceitos e terminologias, procurou com o seu trabalho, fornecer uma estrutura conceptual com definições simples, claras e não ambíguas, juntamente com terminologia apropriada para os conceitos mais fundamentais no campo de sistema de informação; procurou assentar a sua estrutura numa visão ontológica baseada em pressupostos bem estabelecidos, os quais traduzem a sua visão do mundo.

2.2.3. Metamodelo das Linguagens de Modelação

Pesquisa passada na área de sistemas de informação suscitou uma abordagem ao estudo do desenvolvimento de sistemas que realça a compreensão dos metamodelos subjacentes às linguagens de modelação empregues nos métodos de desenvolvimento de sistemas de informação [Carvalho94].

Estas linguagens de modelação, usadas para representar modelos de sistemas de informação, possuem uma propriedade comum a todas as linguagens de modelação, que é a de todas elas serem baseadas num esquema de conceitos que reflecte as suas visões particulares do "mundo" concebido. Este esquema de conceitos, na qual a linguagem de modelação é baseada e sobre a qual todas as restrições se baseiam, corresponde ao metamodelo dessa linguagem. Portanto, o metamodelo determina a forma como se vê, se concebe, ou se modela o "mundo", ou seja, expressa uma determinada visão ontológica [FRISCO98, pág. 153]

O UML no seu estado actual, define uma notação e um metamodelo, o qual estabelece os seus conceitos fundamentais, e implicitamente afirma a sua forma de "ver o mundo".

2.3. Adequação de uma Linguagem a um Esquema de Conceitos

Ao longo dos últimos anos, as metodologias de modelação orientadas a objectos têm-se tornado cada vez mais aceites como "a melhor prática" para o desenho lógico e físico de SI, implementação e manutenção. Na engenharia de software, as construções OO ("Object Oriented") idealizadas e sua utilização aparecem bem compreendidas e definidas. Contudo, a adequação da orientação a objectos nas primeiras fases do desenvolvimento (orientada a domínios do problema que o SI deve suportar e representar) não é clara. Críticos do uso de técnicas OO nas fases iniciais do desenvolvimento de SI, advogam que não tem sido feito esforço suficiente no desenvolvimento de metodologias de desenvolvimento de SI orientados a objecto que sejam igualmente boas a analisar e a representar domínios concretos de problemas [Opdahl01].

A maioria dos métodos disponíveis concentra-se ao nível da expressão de conceitos ou modelos, existindo ainda uma ênfase com os aspectos sintácticos da modelação de conceitos, enquanto que problemas relacionados com a semântica do modelo e à pragmática da utilização de modelos têm sido negligenciados [FRISCO98].

Fowler [Fowler00] também é da opinião que a maioria dos métodos OO têm pouco rigor; as suas notações apelam mais à intuição do que à definição formal, sugerindo que uma forma de aumentar o seu rigor é definindo um metamodelo.

2.3.1. Adequação do UML ao FRISCO

O grupo de trabalho FRISCO, envidou esforços no sentido de procurar esclarecer e estabelecer um conjunto definições e terminologia adequada para os conceitos mais relevantes na área de Sistemas de Informação. O UML, standard para a modelação de sistemas de software, afirma-se também como passível de ser empregue para a modelação de outros sistemas que não de software Assim, é legítimo questionar se o UML, como linguagem com conceitos subjacentes ao seu metamodelo, será adequado para fornecer representação para os conceitos de sistemas de informação tidos como relevantes pelo FRISCO na área de Sistemas de Informação. A resposta a esta questão é a motivação desta dissertação, e à qual se pretende responder no restante deste documento.

Para o estudo desta adequação torna-se conveniente considerar mais algumas noções, as quais brevemente se apresentam nas secções que se seguem.

2.3.2. Os Quatro Mundos

Opdahl e Henderson-Sellers [Opdahl01] recorrem no seu trabalho[4] ao suporte fornecido pela distinção entre quatro mundos do desenvolvimento de Sistemas de Informação [WandWeber93][5]; convém para este estudo ter presente essa distinção:

(1) O *mundo objecto* é o "Universo do Discurso" do sistema proposto, sobre o qual o SI final pode conter e manter informação;

(2) O *mundo do uso* compreende a organização que usará e incorporará o SI, como por exemplo, seus objectivos e estrutura, os indivíduos, suas crenças, hábitos de trabalho, intenções, etc.;

(3) O *mundo do desenvolvimento* compreende as pessoas envolvidas no desenvolvimento de SI, seus métodos, ferramentas e planos de trabalho, os artefactos que usam e produzem, incluindo modelos, etc;

(4) O *mundo do sistema* compreende representações intermédias de vários aspectos do artefacto software a ser construído a vários níveis de abstracção, culminando no SI final.

O mundo *objecto* e o mundo *uso* são conjuntamente referidos em [Opdahl01] como o "*domínio do problema*"[6]. Numa linguagem de modelação, como o UML, poder-se-ão (eventualmente) encontrar elementos de modelação destinados a uma utilização mais apropriada num destes mundos (ou eventualmente em mais do que um).

2.3.3. Análise Ontológica de Gramáticas de SI

Quer o UML e o FRISCO tenham ou não o mesmo objecto de sistema de informação sob consideração (ver [Carvalho99]) e/ou elementos essencialmente destinados para os mesmos mundos, para a análise da adequação deve-se ter em conta os quatro tipos de discrepâncias ontológicas tidas em conta em [Opdahl01][7] que podem prejudicar a claridade ontológica das construções de uma linguagem relativamente a uma ontologia/esquema de conceitos:

- *Sobrecarga de construções* - é quando uma construção da linguagem é interpretada como correspondendo a várias construções ontológicas;

[4] Análise da "*OPEN Modelling Language (OML)*" em termos do modelo de sistemas de informação "*Bunge-Wand-Weber (BWW)*".

[5] Relativamente a estes mundos ("*parcialmente sobrepostos*"), [Opdahl01] faz referência a [Jarke92] e [Mylopoulos92].

[6] Estes também fazem notar que "*Also, not all of the subject world may be represented by the final IS and this subset will be referred to as the scope of the application*".

[7] Relativamente a estas discrepâncias ontológicas, [Opdahl01] referencia [WandWeber93].

- *Redundância de construções* - é quando mais do que uma construção da linguagem é interpretada como correspondendo à mesma construção ontológica;
- *Excesso de construções* - é quando uma construção da linguagem não pode ser interpretada como correspondendo a uma qualquer construção ontológica;
- *Défice de construções* - é quando nenhuma construção da linguagem é interpretada como correspondendo a uma determinada construção ontológica.

Enquanto que a sobrecarga e défice de construções são consideradas como fraquezas da linguagem de modelação, o caso de redundância e de excesso não o são.

Da mesma forma que em [Opdah0l], uma avaliação ontológica relativa ao esquema de conceitos proposto pelo FRISCO pode ser feita com base em duas correspondências:

- *Correspondência de Interpretação* do UML para o FRISCO em ordem a:
  - Identificar as construções que não são orientadas ao domínio do problema;
  - Identificar construções intencionadas para representar diferentes tipos de fenómenos de domínio de problema;
  - Definir com precisão o significado de cada construção orientada ao domínio do problema em termos do tipo de fenómeno que está intencionado representar.
- *Correspondência de Representação* do FRISCO para o UML em ordem a:
  - Identificar (na linguagem) construções orientadas ao domínio de problema que podem ser redundantes porque se sobrepõem semanticamente com outras;
  - Identificar défices na linguagem, isto é construções orientadas ao domínio do problema que estão em falta.

Este estudo terá predominantemente em foco a *correspondência de representação* do FRISCO para a linguagem UML.

3. A Framework of Information System Concepts (FRISCO)

> *"Sistemas de Informação" dizem respeito ao uso de informação por pessoas ou grupos de pessoas em organizações, em particular através de sistemas baseados em computador. Uma compreensão adequada destes sistemas requer que sejam vistos no contexto das organizações que os empregam.*
>
> [FRISCO98, pág. 26]

3.1. O Relatório FRISCO

O relatório FRISCO [FRISCO98] apresenta os resultados do trabalho do grupo "IFIP WG 8.1 Task Group FRISCO", desenvolvido em consequência de uma preocupação acerca da então situação científica, educacional e industrial, a qual foi expressa no primeiro manifesto:

> *"There is a growing concern within IFIP WG 8.1 about the present situation, where too many fuzzy or ill-defined concepts are used in the information system area. Scientific as well as practice-related communication is severely distorted and hampered, due to this fuzziness and due to the frequent situation that different communication partners associate different meanings with one and the same term. There is no commonly accepted conceptual reference and terminology, to be applied for defining or explaining existing or new concepts for information systems".*
> [FRISCO98, pág. 2]

O grupo foi proposto em 1987 e estabelecido em 1988, tendo organizado três conferências sobre este assunto (em 1989 [ISCO89], em 1992 [ISCO92] e em 1995 [ISCO95]); o grupo produz em 1990 um relatório intermédio provisório [Lindgreen90]. Em 1998 divulga na web - para fins educacionais e de investigação - o relatório final que ilustra os resultados do trabalho desenvolvido.

3.2. Principais Resultados

Para os cientistas e profissionais na área de sistemas de informação, o relatório apresenta um referencial que fornece um esquema de conceitos consistente e coerente e, uma terminologia apropriada que lhes permite expressar sobre matérias na área de sistemas de informação de uma forma bem definida.

O relatório também justifica cientificamente a área de sistemas de informação, procurando enquadrá-la numa área de contexto mais geral, compreendendo a filosofia, a ontologia, a semiótica, a ciência de sistemas, a ciência da organização, e a ciência de computação.

3.3. Desenvolvimento do Estudo

A definição de um vocabulário unificado para todo o domínio de sistemas de informação afigura-se como difícil, pois existem muitos sub-domínios diferentes, preocupados com diferentes conjuntos de problemas, onde se encontram palavras de todos os dias (tal como informação, dados, comunicação, processo, conhecimento, sistema, etc.) colocadas a serviço com significados locais restritos. Idealmente, a terminologia para qualquer domínio seleccionado tem de ser colocada numa sólida fundação teórica, a qual não é construída facilmente. Existe a convicção que esta apenas poderá emergir como resultado de consenso alcançado gradualmente pela comunidade profissional que trabalha nesse domínio. Em alguns domínios, onde aparentemente existe um grau suficiente de consenso, o FRISCO tentou isolar e clarificar alguns conceitos fundamentais.

Estabelecendo que os sistemas de informação existem exclusivamente dentro de organizações para suportar o seu trabalho e satisfazer as suas necessidades de informação e de comunicação, o FRISCO considerou que, para uma melhor compreensão de sistemas de informação, ser conveniente recorrer a outras áreas de conhecimento como:

- *Ciência Organizacional*, para melhor compreensão das organizações: o que elas são, como funcionam, quais são os seus componentes, qual a sua estrutura e comportamento. As organizações podem ser vistas como sistemas organizacionais;
- *Ciência Cognitiva e Semiótica*, para melhor compreender as noções de informação e de comunicação, e assim, poder-se correctamente compreender os requisitos de informação e de comunicação dentro das organizações;
- *Ciência de Sistemas*, para melhor compreender os sistemas de informação e os sistemas organizacionais, uma vez que ambos são sistemas. Os sistemas são concepções específicas (nas mentes das pessoas) que podem ser representados em algumas linguagens de modelação;
- *Visão Ontológica e Posição Filosófica*, para que se possa ser capaz de investigar com solidez (e fundamentar correctamente qualquer conceito).

3.3.1. Opções e Princípios Orientadores

Durante o trabalho do FRISCO, descobriu-se que para o campo de sistemas de informação, não existe à priori uma fundação conceptual objectivamente determinada que se pudesse assumir e simplesmente utilizar como suporte para a construção de novos conceitos. Considerando que tal fundação só pode ser determinada inter-subjectivamente através de consenso, ao preparar a estrutura de conceitos de sistemas de informação, o FRISCO teve de inevitavelmente fazer face ao problema de identificar as várias opções respeitantes a certas facetas da área, e escolher entre estas opções de acordo com certos princípios de orientação [FIRSCO98].

Os princípios de orientação mais importantes que regeram a elaboração da estrutura de conceitos do FRISCO foram:

- consistência global;
- generalidade;
- simplicidade;
- ancorar os conceitos de informação em campos relacionados;
- uma fundação conceptual sobre a qual se possa construir.

As opções mais relevantes assumidas foram:

- posição metafísica construtivista (que reconhece que toda a concepção tem um aspecto puramente pessoal e privado, e que constrói uma realidade inter-subjectiva através da partilha das experiências individuais da realidade em acções onde se chega a consenso sobre as categorias das coisas);
- à priori, uma visão ontológica subjacente que permita definir uma fundação básica sobre a qual se possam definir os conceitos (apresentada mais à frente neste documento);
- apresentação informal e formal dos conceitos definidos.

3.4. Os Conceitos na área de Sistemas de Informação

3.4.1. A Relevância de um Conceito

Sustenta o relatório que um conceito deve ser introduzido num esquema de conceitos apenas se for relevante. No domínio de sistemas de informação, um conceito para ser relevante deve em última instância contribuir para a criação de valor nos processos das organizações; caso contrário, não é relevante. A única razão para introduzir um conceito no nosso domínio é a de criar valor através do seu uso; a contribuição de valor por um conceito estabelecido origina dos seus efeitos na organização como consequência da sua aplicação (directa ou indirecta) apropriada durante a análise e desenho de processos e seus possíveis suportes de TI, permitindo processos efectivos de comunicação [FRISCO98].

3.4.2. Mudança de Perspectiva

Na área de sistemas de informação, a visão predominante (de cariz tecnológico) é a de que os sistemas de informação baseados em TI têm o objectivo de adquirir, processar e disseminar dados. Em contraste com esta visão, o relatório FRISCO propõe uma mudança de perspectiva de concentração exclusiva em dados e informação, para passar a abranger a comunicação humana e o seu papel no suporte de criação de valor em acções dentro de uma organização, uma vez que, na maioria dos empreendimentos humanos, a adição de valor resulta de acção coordenada, e, em qualquer organização, a acção coordenada é por necessidade baseada na comunicação efectiva.

Assim, subjacente ao conjunto de conceitos fundamentais, está uma mudança de uma perspectiva limitada do *como* para um perspectiva mais larga do *porquê*, baseada no pensamento em termos de efeitos na organização pela aplicação conceptual da estrutura de conceitos. Ver os sistemas de informação principalmente como veículos para comunicação, em vez de como meio de armazenamento de dados e de processamento, requer em alguma medida, novos e complementares tipos de métodos para análise e desenho [FRISCO98].

3.4.3. Ponto de partida para uma Estrutura de Conceitos

A criação de uma estrutura para o raciocínio e eventualmente para o desenvolvimento de métodos e técnicas para análise e desenho pode ter vários pontos de partida. Na procura de tal

ponto de partida, o FRISCO colocou uma série de questões fundamentais que lhe permitiu delinear uma linha de raciocínio, questões essas que são sintetizadas no Quadro 3.1.

Domínio, Âmbito	Questão	Resposta
Economia da Organização: Aspectos valor/adição de valor organizacional	Porque se desenham e implementam sistemas de informação?	Os sistemas de informação são desenhados e implementados dentro de uma organização para suportar acção adequada de forma a que possa ser alcançada adição de valor.
Economia de Cultura: Aspectos Sociais	Qual é o contexto no qual as acções são executadas?	Uma organização constitui um sistema social, onde a acção é executada dentro de um conjunto de objectivos, normas e regras de comportamento mais ou menos bem estabelecidas.
Economia da Comunicação: Aspectos pragmáticos	Como adquirirmos a base necessária para a decisão e acção coordenada?	A base primária para a acção coordenada é estabelecida através da comunicação.
Economia da Informação Aspectos semânticos	O que é na realidade comunicado?	Exclusivamente, são comunicados concepções ou modelos.
Economia de apresentação: Aspectos sintácticos	Como são representados os modelos?	Os modelos são expressos como sentenças em linguagens definidas.
Economia de armazenamento e transmissão: Aspectos empíricos	Como são as sentenças tornadas persistentes no tempo, i.e., se manifestam?	As sentenças são codificadas em padrões (sujeitos a corrupção e ruído)
Economia da codificação e detecção: Aspectos físicos	Como são implementados os padrões?	Os padrões são produzidos como traços detectáveis em substratos físicos.

Quadro 3.1 - Uma visão da linha de raciocínio do FRISCO

3.5. Definição de Conceitos

O esquema de conceitos proposto FRISCO (especificamente relacionado com a área de "sistemas de informação") é baseado em vários pressupostos que representam a sua visão particular do mundo, a "Weltanschauung". Não pretendendo desenvolver uma teoria ontológica universal, o FRISCO restringe-se apenas à apresentação de uma visão ontológica adequada para descrever e explicar os fenómenos e conceitos mais importantes no campo de sistemas de informação.

3.5.1. Os Pressupostos que estabelecem a Visão do Mundo

Pressupondo um mundo de actores que interagem entre si, os quais, no contexto de uma estrutura organizacional, empreendem acções que se consideram ser úteis ou necessárias, o FRISCO apresenta a sua visão ontológica com base em quinze pressupostos, os quais de seguida se apresentam:

Pressuposto [a]:

O "mundo" existe, independentemente da nossa própria existência, ou das nossas capacidades cognitivas e intelectuais.

Pressuposto [b]:

Os seres humanos são capazes de observar e de perceber "partes" ou "aspectos" do "mundo" (às quais chamamos de domínio) com os seus sentidos, formando portanto percepções nas suas mentes. As percepções podem ser consideradas como padrões específicos, geralmente alterando-se com o tempo.

Pressuposto [c]:

Os seres humanos são capazes de formar concepções nas suas mentes, como resultado de percepção actual ou passada de vários processos cognitivos ou intelectuais, tais como reconhecimento, caracterização, abstracção, derivação e/ou reflexão interna. A colecção de concepções (relativamente) estáveis e (suficientemente) consistentes na mente de uma pessoa é chamado de seu conhecimento.

Pressuposto [d]:

Um domínio percebido pode ser concebido como composto por componentes identificáveis, aos quais chamamos de coisas; elas podem sobrepor-se, conterem-se umas às outras ou relacionar-se entre si de qualquer forma empírica.

Pressuposto [e]:

Algumas coisas são concebidas como tendo existência estática (estados), enquanto que outras são concebidas como alterações de alguns estados (transições). Assim, um domínio percebido pode ser concebido como tendo uma existência num contexto temporal.

Pressuposto [f]:

Algumas transições podem ser concebidas como sendo efectuadas ou provocadas por algumas coisas activas, chamadas de actores. Uma tal transição, chamada de acção, é efectuada por esse actor em coisas passivas, chamadas de actands. Uma acção racional por um actor é dita ser em procura de um objectivo. É possível que a causa de uma transição não possa ser atribuída a um actor específico, mas meramente surge devido a, ou é despoletada por, algum aspecto do estado precedente.

Pressuposto [g]:

As pessoas usam representações para comunicar as suas concepções. Essas concepções são representadas em alguma linguagem em algum meio.

Pressuposto [h]:

Para uma sociedade, é de interesse principal que qualquer domínio seja representado de forma que que exista consenso sobre este. Sem reinvidicar a presumida "existência" em qualquer sentido "mundo real" do termo, as representações resultantes com as quais toda a gente concorda, será chamada de "realidade inter-subjectiva".

Pressuposto [i]:

Podem-se formar concepções estáveis na mente de uma pessoa – como um resultado de observar o estado de alguma parte do mundo concebido ou as alterações de tal estado, ou como resultado da comunicação com outras em relação àquele. Qualquer tal concepção pode ser chamada de conhecimento sobre o mundo. A existência assumida da concepção (mental) é uma metáfora para a experiência de pessoas quando recebem comunicação significativa.

Pressuposto [j]:

As pessoas observam, experienciam e discutem representações a vários níveis semióticos, isto é, eles focam, com variada ênfase: os aspectos físicos e empíricos de uma representação, as leis para expressar tal representação, o seu significado, o seu uso e efeitos, e o ponto ao qual se pode concordar com e possivelmente derivar obrigações implícitas em qualquer afirmação sobre alguma coisa.

Pressuposto [k]:

Os actores humanos podem ter em suas mentes modelos, concepções claras, precisas e não ambíguas, os quais podem expressar na forma de denotação de modelo, numa linguagem formal ou semi-formal apropriada.

Pressuposto [l]:

Aquando a observação de algum domínio, os actores humanos frequentemente concebem coerência nesse domínio, ou afirmam a presença de tal corência. O modelo resultante, feito de modelos coerentes, é chamado de sistema. Qualquer coisa fora desse domínio será considerado como pertencendo ao ambiente do sistema.

Pressuposto [m]:

As organizações são caracterizadas por arranjos (geralmente orientados a objectivo) entre várias partes, pelas quais é possível a acção (maioritariamente co-operativa e coordenada) através de comunicação (frequentemente sistemática), a qual ocasiona a troca de informação, envolvendo mensagens significativamente compostas.

Pressuposto [n]:

Uma organização é um agrupamento de actores, juntamente com uma colecção de actands, tal que (a) é perseguido um objecivo comum ou seja mostrada alguma outra características de coerência, e (b) ocorrem interacções baseadas em comunicação. A informação é usada numa organização no contexto do seu funcionamento (acções), não só internamente mas também em relação ao seu ambiente, a sociedade no geral. Devido a este envolvimento social, as normas são directivas significativas para indivíduos na organização e a organização como um todo.

Pressuposto [o]:

O uso de informação organizacional deriva da sua prática operacional, a qual pode estar presente na forma do seu sistema organizacional (pelo "modelo de negócio"); dentro deste, pode-se encontrar embebido o que pode ser descrito como uma rede complexa de fluxos de informação que suporta as suas acções; alguns dos fluxos de informação podem ser arranjados dentro de um subsistema de informação computarizado.

Na representação de concepções os seres humanos frequentemente usam símbolos, tais como letras, caracteres, string de caracteres, para referir as suas concepções. A Figura 3.1 resume os pressupostos de [a] a [g].

Os pressupostos assumidos reflectem a visão comum do grupo, embora haja elementos [Stamper98] do grupo que não partilhem na plenitude alguns dos pressupostos.

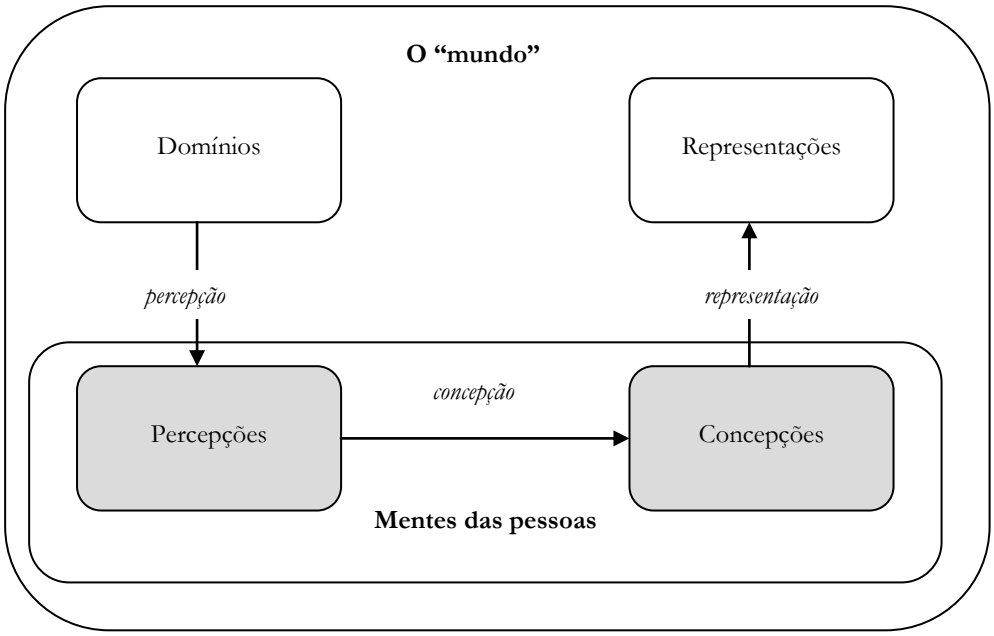

Figura 3.1 –Domínios de Percepção, Concepção e de Representação (adaptado de [Frisco98])

3.5.2. Estrutura Conceptual

Assumindo que o objectivo é alcançar uma visão coerente de todos os aspectos de informação e de comunicação num contexto organizacional, a exposição dos conceitos no relatório FRISCO é feita quer de um modo informal, quer de modo formal: o modo informal tem a intenção de facilitar a compreensão dos conceitos, destinado para estudantes e profissionais; o modo formal, com a intenção de obter maior rigor, com a intenção de se assegurar um documento científico, e destinado para aquela pessoa mais inclinada para o plano teórico.

De notar que no grupo do FRISCO não foi consensual o processo que levou a apresentação do núcleo do relatório. Na opinião de Lindgreen [Lindgreen98], foi um erro decidir que era um objectivo principal conseguir um conjunto formalizado de definições axiomáticas, decisão baseada na premissa que sem tal formalismo o relatório são seria visto como um documento científico quando medido pelas normas de um trabalho realizado no IFIP. Alega para sustentação da sua posição, que "…*providing a set of axiomatic definitions neither is necessary nor a sufficient condition to ensure scientific result*." [Lindgreen98]. Relativamente à abordagem axiomática escolhida, sustenta que existem muitos aspectos relevantes para a compreensão dos conceitos que não podem ser expressos adequadamente pelo formalismo; alega também que embora baseado em conceitos matemáticos simples, *"…many readers in practice are untrained in the notation – also among those scientifically educated – and accordingly they will have severe difficulties in grasping what is expressed by the formal statements*". Conforme poderemos constatar, enquanto que se torna de fácil compreensão os conceitos apresentados informalmente, o mesmo não se poderá afirmar da sua apresentação formal.

As críticas [Lindgrenn98] também se estendem à exposição informal dos conceitos: para além das explicações deveriam também ser fornecidos argumentos para a relevância directa ou indirecta dos conceitos no contexto organizacional e comunicativo, e, para cada conceito deveriam haver argumentos apropriados para a terminologia sugerida. De facto, na leitura do FRISCO podemos observar que embora se assinale outros termos também usados na literatura para um mesmo conceito, não existe no entanto uma justificação para a escolha de um termo em detrimento de outro para a representação de um conceito.

3.5.2.1. A Exposição Informal

O esquema de conceitos proposto pelo FRISCO está na sua totalidade exposto no anexo B, apresentando-se no Quadro 3.2 apenas um extracto da definição desses conceitos. Com a intenção de assegurar a não distorção da interpretação pretendida pelos seus autores, os conceitos são expostos na língua inglesa, língua em que foram originalmente expressos.

Conceito	Significado
Definition E1: Thing	A **thing** is any part of a conception of a domain (being itself a "part" or "aspect" of the "world"). The set of all things under consideration is the conception of that domain.
Definition E2: Predicator, Predicated thing	A **predicator** is a thing, used to characterise or qualify other things, and assumed as being "atomic", "undividable" or "elementary". A **predicated thing** is a thing being characterised or qualified by at least one predicator.
Definition E3: Relationship	A **relationship** is a special thing composed of one or several predicated thing(s), each one associated with one predicator characterising the role of that predicated thing within that relationship.
Definition E4: Set membership, Elementary thing, Composite thing	A **set membership** is a special binary relationship between a thing (the set), characterised by the special predicator called 'has-element', and another thing, characterised by the special predicator called 'is-element-of'. An **elementary thing** is a thing, not being a relationship and not being characterised by the special predicator called 'has-element'. A **composite thing** is a thing, not being an elementary thing.
Definition E5: Entity	An **entity** is a predicated thing as well as an elementary thing.
Definition E6: Type, Population, Instance	A **type** of things is a specific characterisation (e.g. a predicate) applying to all things of that type. A **population** of a type of things is a set of things, each one fulfilling the characterisation determining that type An **instance** of a type of things is an element of a population of that type.

Quadro 3.2 – Extracto da definição informal de conceitos do FRISCO

3.5.2.2. A Exposição Formal

O formalismo da estrutura de conceitos é feito com base em primitivas, axiomas, funções e definições. E em linha com o dito por Lindgreen [Lindgreen98], alguns dos conceitos introduzidos informalmente não foram formalizados, nomeadamente[8]:

Relative time [E11, D13],	Absolute time [E11, D14],	Co-action [E14],
Semiotic level [E25],	Intensional model, Extensional model [E28],	Meta-model [E29],
Communication [E37, D37],	Shared knowledge [E38, D38],	Norm [E39]

Os restantes foram formalizados e são de seguida parcialmente expressos no Quadro 3.3[9]. Poder-se-á constatar que a leitura e compreensão dos conceitos na sua apresentação formal, não é de facto tão imediata quanto a uma apresentação informal dos mesmos, sendo aconselhável a leitura destes últimos, antes de se avançar para a interpretação dos primeiros.

Conceito
Primitive P1: Thing [E1] The set of all things is denoted by **Z**
Definition D1: Relationship [E3] The set of all relationships is denoted by **R** . $\mathbf{R} = \{r \in \mathbf{Z} \mid r \subseteq \mathbf{U} \wedge \mathbf{U} = \{<q, p> \mid q, p \in \mathbf{Z}\} \wedge 1 = \lvert r \rvert < \infty\}$.
Definition D2: Predicator [E2] The set of all predicators is denoted by **P** . $\mathbf{P} = \{p \in \mathbf{Z} \mid \exists\, u \in \mathbf{U}, q \in \mathbf{Z}\ [u = <q, p>]\}$.
Definition D3: Predicated thing [E2] The set of all predicated things is denoted by **Q** . $\mathbf{Q} = \{q \in \mathbf{Z} \mid \exists\, u \in \mathbf{U}, p \in \mathbf{P}\ [u = <q, p>]\}$.
Primitives P2: Set membership predicators 'has-element' $\in$ **P** and 'is-element-of' $\in$ **P** are special predicators characterising a set and an element of a set in the context of a set membership, respectively.
Definition D4: Set membership [E4] The set of all set memberships is denoted by **SM** . $\mathbf{SM} = \{sm \in \mathbf{R} \mid$ $sm = \{<q_1, \text{has-element}>, <q_2, \text{is-element-of}>\} \wedge q_1, q_2 \in \mathbf{Q} \wedge q_1 \neq q_2\}$. Abbreviations: - $sm = (q_2 \in q_1)$; - $\{q_2, ...\} = q_1$; or $\{q_2, ...\} \subseteq q_1$; - $\wp(q_1)$ denotes the set of all subsets of q_1. - $\wp_m(q_1)$ denotes the set of all sub-multi-sets of q_1.

[8] No FRISCO, todos os conceitos encontram-se etiquetados por uma letra e um número: na apresentação informal do conceito, é utilizada a letra E; na apresentação formal do conceito, é utilizada a letra D. Para cada letra (D ou E) os números são sequencialmente atribuídos (começando no número um).

[9] No anexo B encontram-se expostas as apresentações formais e informais de todos os conceitos.

Definition D5: Elementary thing [E4] The set of all elementary things is denoted by **EZ .** **EZ** = {ez ∈ Z \ **R** | ¬ ∃ u ∈ **U** [u = <ez, has-element>]}.
Axiom A1: ∀ p ∈ **P** [p ∈ **EZ**].
Definition D6: Entity [E5] The set of all entities is denoted by **E** . **E** = **EZ** ⋂ **Q**.
Definition D7: Composite thing [E4] The set of all composite things is denoted by **CZ** . **CZ** = **Z** \ **EZ.**

Quadro 3.3 – Extracto de definição formal do FRISCO

3.6. Tópicos de interesse

Durante o trabalho desenvolvido, o FRISCO identificou três disciplinas que foram consideradas particularmente relevantes para a compreensão das fundações da área de sistemas de informação [FRISCO98, pág. 137]:

(1) *Semiótica*, a disciplina que respeita à teoria dos sinais;

(2) *Ciência de Sistemas*, disciplina que respeita a teoria de sistemas;

(3) *Ontologia*, a disciplina que diz respeito a teorias de como o mundo pode ser visto, concebido ou modelado.

Para cada uma destas três disciplinas, foi dedicada uma secção num capítulo de relatório[10]de forma a poderem ser lidas independentemente uma das outras, e até, independentemente do próprio relatório. Entre os tópicos focados nestas secções encontram-se, entre outros:

- Resenha histórica da semiótica, níveis de semiótica, modelação e modelos;
- Sistema, Domínio de Sistema, "Systemeering", Sistemas Organizacionais e Subsistema;
- Metamodelos, diferenças entre metamodelos e características relevantes de sistemas de informação.

3.7. Conceitos considerados neste estudo

Em jeito de conclusão deste capítulo dedicado ao FRISCO, apresentam-se no Quadro 3.4 os conceitos[11] que são considerados para este estudo de adequação.

[10] Capítulo 6 ("*Elaboration on Selectd Topics*") [FRISCO98]

[11] Os conceitos encontra-se listados de cima para baixo e da esquerda para a direita, em conformidade com a ordem com que são apresentados no relatório do FRISCO.

Conceitos do FRISCO		
Thing	Domain	System denotation
Predicator	Domain component	System component
Predicated thing	Domain environment	System environment
Relationship	Human actor	System viewer
Set membership	Perception	System representer
Elementary thing	Perceiving action	Dynamic system
Composite thing	Perceiver	Static system
Entity	Conception	Active system
Type	Conceiving action	Passive System
Population	Conceiver	Open system
Instance	Conceiving context	Closed System
Transition	Interpreting action	Sub-system
State	Interpreter	Knowledge
Pre-state	Interpreting context	Data
Post-state	Symbol	Message
State-transition structure	Alphabet	Message transfer
Composite transition	Symbolic construct	Sender
Transition occurrence	Language	Receiver
Relative time	Representation	Information
Absolute time	Representing action	Communication
Rule	Representer	Shared knowledge
Actor	Representing context	Organisational system
Action	Label	Norm
Composite action	Reference	Information system
Action occurrence	Semiotic level	Information system denotation
Co-action	Model	Computerised information sub-system (CISS)
Actand	Model denotation	
Input actand	Modelling action	
Output actand	Modeller	
Resource	Intensional model	
Action context	Extensional model	
Goal	Meta-model	
Goal-pursuing actor	System	

Quadro 3.4 – Conceitos do FRISCO considerados neste estudo

Como se poderá constatar pela análise de adequação[12] efectuada, os conceitos e definições utilizadas correspondem à apresentação informal de conceitos. Tal deve-se por se considerar a versão informal das definições como sendo a mais clara e útil para este estudo.

[12] Apresentada mais à frente neste documento, no Anexo A.

4. Unified Modeling Language (UML)

> *"In a relatively short period of time the Unified Modeling Language has emerged as the software industry's dominant modeling language."* [Kobryn99]

O UML (Unified Modeling Language) é a linguagem de modelação adoptada como standard pelo OMG (Object Management Group) para especificar, visualizar, construir, e documentar artefactos de sistemas de software, como também para modelar negócios e ou outros sistemas que não de software. Representa uma colecção das melhores práticas de engenharia na modelação orientada a objectos, com provas dadas na modelação de sistemas grandes e complexos. Constitui o resultado de vários anos de esforço, em que se focou a unificação dos métodos mais usados no mundo e a adopção de boas práticas da indústria. [OMG-UML01].

4.1. História

As linguagens de modelação orientadas a objectos apareceram algures entre meados dos anos 70 e o fim dos anos 80 assim que os metodologistas, deparando com um novo género de linguagens de programação (orientadas a objectos) e com a crescente complexidade das aplicações, começaram a experimentar abordagens alternativas à tradicional análise e desenho. Durante o período de 1989 e 1994, o número de métodos orientados a objectos aumentou de menos de 10 para mais de 50. Muitos dos utilizadores destes métodos tinham problemas em encontrar uma linguagem de modelação que satisfizesse completamente as suas necessidades, alimentando a chamada guerra dos métodos [Booch99].

O UML não foi a primeira notação para a modelação (orientada a objectos) de sistemas de software; de facto, o UML foi criado para colocar um fim à confusão entre notações concorrentes. A Figura 4.1 procura ilustrar o caminho percorrido até se chegar à aceitação do UML como um standard pelo OMG.

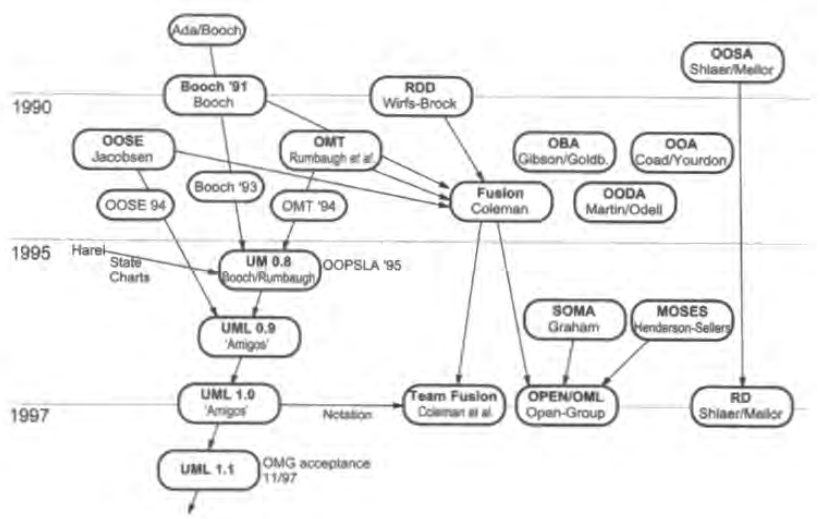

Figura 4.1 – Evolução das linguagens orientadas a objectos (extraído de [Oestereich99])

O UML começou como uma colaboração entre três dos mais proeminentes metodologistas: Grady Booch, Ivar Jacobson e James Rumbaugh. Primeiramente, Booch e Rumbaugh procuraram unificar os seus métodos com o *Unified Method v.0.8* em 1995; por esta altura, Jacobson juntou-se a eles para colaborar numa tarefa ligeiramente menos ambiciosa de unificar as suas linguagens de modelação com UML 0.9.

Os utilizadores reconheceram rapidamente as vantagens de uma linguagem de modelação comum que pudesse ser usada para visualizar, especificar, construir e documentar os artefactos de um sistema de software. Começaram a aplicar os esboços iniciais da linguagem aos diversos domínios desde as finanças e saúde, até às telecomunicações e à indústria aeroespacial. Levados por uma forte procura dos utilizadores, os vendedores de ferramentas de modelação rapidamente incluíram suporte ao UML nos seus produtos.

Ao mesmo tempo que o UML estava a tornar-se um standard de facto na indústria, uma equipa internacional de peritos de modelação assumiu a responsabilidade de tornar a linguagem um standard formal. Representando uma mistura diversificada de vendedores e integradores de sistemas, começaram em 1996 a trabalhar com os metodologistas para propor o UML como a linguagem de modelação standard para o OMG; uma proposta inicial (UML 1.0) foi apresentada ao OMG em Janeiro de 1997. Depois de nove meses de melhorias à especificação, submeteram a proposta final (UML 1.1) em Setembro de 1999, a qual o OMG adoptou como seu standard de modelação de objectos em Novembro de 1999.

É importante notar que o UML sofreu de alguns efeitos secundários indesejáveis devidos ao seu rápido percurso através do processo de submissão do OMG. Embora a infra-estrutura e a maioria da superstrutura da linguagem fossem sólidas, vários problemas significativos foram reconhecidos na altura da submissão final – "*Incomplete semantics and notations for activity graphs, Standards elements bloat; Architectural misalignement*"-. Em vez de atrasar a normalização do UML,

os proponentes resolveram endereçar alguns destes problemas na revisão seguinte da linguagem [Kobryn99].

A versão 1.2 foi mais cosmética, mas a versão 1.3 tornada pública em 1999, teve mais significado[Fowler00]. O UML 1.3 é a primeira versão madura da especificação da linguagem de modelação. Da perspectiva de um modelador, não houve muitas alterações entre UML 1.1 e UML 1.3. A maioria das melhorias feitas à linguagem foi a um nível infraestrutural, e envolveu afinação da semântica do metamodelo. Apenas foram feitas um pequeno número de alterações à notação [Kobryn99].

A versão actual da linguagem é a 1.4, estando-se à espera ao longo de 2002 de uma nova versão 2.0[13].

4.2. A Linguagem

Uma análise da literatura acerca do UML revela existirem várias formas de apresentação/introdução à linguagem, o que levanta algumas dificuldades perante a necessidade de fazer uma exposição geral e sucinta dos aspectos mais relevantes da linguagem.

Pode-se constatar que em alguma literatura (por exemplo [Fowler00; Furlan98; Penker00]) se começa por apresentar os diagramas existentes no UML (que tipos, quais os usos mais apropriados), e subjacente a estes, os elementos de modelo neles utilizados.

Outra literatura ([Penker98; Rumbaugh98]), começa por enfatizar visões do UML - "*Views show different aspects of the system that are modeled. A view is not a graph, but an abstraction consisting of a number of diagrams. Only by defining a number of views, each showing a particular aspect of the system, can a complete picture of the system be constructed.*"[Penker98] – e, para cada uma destas visões, apresentam-se os vários diagramas que nelas se podem enquadrar, e com estes, explicam-se os elementos de modelo aplicáveis.

[Booch99] afirma "*To undestand UML, you need to form a conceptual model of the language, and this requires learning three major elements: the UML's basic building blocks, the rules that dictate how those building blocks may be put together, and some common mechanisms that apply throughout the UML*"; começando assim a sua linha de apresentação com os blocos básicos de construção (em que o vocabulário do UML compreende três tipos de blocos de construção: coisas, relacionamentos, e diagramas), regras, e mecanismos comuns da linguagem.

[Silva01] segue esta linha de raciocínio, podendo-se encontrar referência à estrutura de conceitos do UML com a afirmação "*A estrutura de conceitos do UML pode ser vista através das seguintes noções: (1) coisas,...; (2) relações, …; e (3) diagramas, …. Os elementos encontram-se organizados*

[13] A qual deverá contudo surgir apenas em 2003.

consoante a sua funcionalidade ou responsabilidade. Assim, há elementos de estrutura, comportamento, agrupamento e anotação."; de forma abreviada, segue-se a apresentação destes.

E como se apresenta o documento onde se encontra a especificação da linguagem UML [OMG-UML01]? Não sendo recomendado para introdução ao UML, "*Readers looking for an introduction to the UML or object modeling should consider another source.*"[OMG-UML01, pág. 2-2], a abordagem dá ênfase à arquitectura da linguagem e ao rigor. Apresenta em primeiro lugar a semântica do UML recorrendo ao metamodelo da linguagem, o qual estruturado em pacotes, é descrito de modo semi-formal, "*The metamodel is described in a semi-formal manner using these views: abstract sintax, well-formedness rules, semantics*" [OMG-UML01, pág. 2-3]. Após a apresentação do metamodelo e semântica do UML, é apresentada a notação do UML, a qual especifica a sintaxe gráfica para a expressão da semântica descrita anteriormente pelo metamodelo do UML; aqui aparece então a apresentação dos diagramas do UML [OMG-UML01].

Portanto, conclui-se que não existe uma abordagem inicial comum para a apresentação dos conceitos mais relevantes do UML; as existentes têm um carácter de entendimento pessoal do sobre o quê e/ou como é mais relevante e apropriado para a compreensão da linguagem, pressupondo implicitamente uma particular audiência.

Fazendo parte do interesse deste trabalho a consideração dos conceitos que podem ser encontrados no metamodelo do UML, opta-se por prosseguir com a apresentação do metamodelo da linguagem em conformidade com a exposição presente no documento de especificação (o qual formaliza a semântica da modelação com o UML), remetendo para a literatura existente a pesquisa de mais informações relativas à introdução ao UML.

4.3. Extensões e Perfis do UML

Antes de se entrar na apresentação do metamodelo do UML, interessa[14] considerar os mecanismos de extensão de linguagem que o UML oferece, e em particular, o conceito de perfil incluído no UML.

Em acordo com o que se pode encontrar expresso na literatura,

> "*O UML providencia um elevado número de conceitos e notações particularmente concebidos de forma a satisfazer os requisitos típicos de modelação de software. Contudo, podem surgir situações em que se torna desejável a introdução de conceitos e/ou notações adicionais…*"
>
> [Silva01, pág. 261]

[14] Para o estudo da adequação do UML ao FRISCO.

o UML permite através dos seus mecanismos de extensão, criar extensões à linguagem, o que torna possível uma maior facilidade em modelar outros domínios que não de software, ou domínios de particular especificidade[15].

A própria especificação do UML espelhando esta necessidade,

> "*OMG expects that the UML will be tailored as new needs are discovered and for specific domains."*
>
> [OMG-UML01, pág. 1-5]

fornece, para além dos mecanismos gerais de extensão, dois perfis exemplo que procuram ilustrar como podem ser usados os mecanismos de extensão para adaptar o UML para domínios específicos: (1) Perfil UML para Processos de Desenvolvimento de Software e (2) Perfil UML para a Modelação de Negócios.

4.3.1. Perfil UML para Processos de Desenvolvimento de Software

Este perfil procura ser um exemplo de como é utilizada alguma terminologia de perfil e de notação[16], que facilite a organização dos modelos desenvolvidos ao longo do processo de desenvolvimento:

> "M*any of the stereotypes are used to give the ability to structure and categorize models and systems during different stages of the development process.*"
>
> [OMG-UML01, pág. 4-2]

Os estereótipos que são definidos neste perfil são apresentados[17] no Quadro 4.1

[15] Neste contexto é usado o termo "perfil" para designar um conjunto predefinidos de estereótipos, marcas com valor, restrições e ícones.

[16] Embora sendo baseado no "*Unified Process"*, não é uma definição completa do deste, ou de como o aplicar (para mais informação sobre este processo de desenvolvimento de software, ver [Jacobson99]).

[17] Em virtude da sua relevância no contexto deste estudo, faz-se acompanhar alguns destes conceitos com a descrição contida no documento de especificação.

Nome	Classe Base	Descrição
UseCaseModel	Model	Especifica os serviços que um sistema oferece aos seus utilizadores.
AnalysisModel	Model	
DesignModel	Model	
ImplementationModel	Model	
UseCaseSystem	Package	
AnalysisSystem	Package	
DesignSystem	Subsystem	
ImplementationSystem	Subsystem	
AnalysisPackage	Package	
DesignSubsystem	Subsystem	
ImplementationSubsystem	Subsystem	
UseCasePackage	Package	
AnalysisServicePackage	Package	
DesignServiceSubsystem	Subsystem	
Boundary	Class	Uma classe fronteira[18] reside na periferia de um sistema, mas dentro deste. Interage com actores fora do sistema como também com classes entidade, controle e outras classes fronteira dentro do sistema.
Entity	Class	Uma entidade é uma classe passiva; isto é, os seus objectos não iniciam interacções por eles mesmos. Um objecto entidade pode participar em muitas realizações diferentes de casos de uso e geralmente vive para além de qualquer interacção.
Control	Class	Uma classe de controle é uma classe cujos objectos gerem interacções entre colecções de objectos. Uma classe de controle geralmente tem comportamento que é específico para um caso de uso, e um objecto de controle geralmente não vive para além das realizações de caso de uso em que participa.
Communicate	Association	A comunicação é uma associação entre actores e casos de uso que é usada para denotar mensagens que podem ser trocadas entre eles. Também pode ser usada entre fronteira, controle, e entidade, e entre actor e fronteira.
Subscribe	Association	Uma associação "subscrição" entre duas classes denota que objectos da classe fonte (chamada de subscritor) serão notificados quando um evento em particular ocorreu em objectos da classe destino (chamada de classe publicitária). A associação inclui a especificação do conjunto de eventos que fazem com que um subscritor seja notificado.

Quadro 4.1 – Elementos do Perfil UML para o Processo de desenvolvimento de software

4.3.2. Perfil UML para a Modelação de Negócios

Este é um perfil exemplo que descreve como o UML pode ser adaptado para a modelação de negócios:

[18] Termo retirado de [Silva01, pág. 268].

"*Although all UML concepts can be brought to bear on this domain* [business modeling], *but example emphasizes common stereotypes and some useful terminology*."

[OMG-UML01, pág. 4-9]

Na especificação do UML, um sistema de negócio compreende vários modelos diferentes mas relacionados, em que os modelos são caracterizados como sendo exteriores ou interiores ao sistema de negócio que representam:

"*Exterior models are use case models and interior models are object models. A large system may be partitioned into subordinate business systems*"

[OMG-UML01, pág. 4-10]

Os estereótipos que são definidos neste perfil são apresentados no Quadro 4.2.

Nome	Classe Base	Descrição
UseCaseModel	Model	Um modelo de caso de usos é um modelo que denota os processos de negócio de um negócio e as suas interacções com entidades externas tais como clientes ou parceiros. Um modelo de caso de usos descreve: - o negócio modelado como casos de uso; - entidades externas modeladas como actores; - os relacionamentos entre entidades externas e os processo de negócio.
UseCaseSystem	Package	
UseCasePackage	Package	
ObjectModel	Model	Um modelo de objectos é um modelo cujo pacote de nível topo é um sistema de objectos que descreve as coisas no interior do próprio negócio.
ObjectSystem	Subsystem	
OrganizationUnit	Subsystem	
WorkUnit	Subsystem	Representa um subsistema que pode conter uma ou mais entidades. É um conjunto de objectos orientados a tarefa que formam um todo reconhecível ao utilizador final.
Worker	Class	É uma classe que representa uma abstracção de um humano que age dentro do sistema. Um trabalhador ("worker") interage com outros trabalhadores e manipula entidades aquando da participação em realizações de casos de uso.
CaseWorker	Class	É um caso especial de trabalhador que interage directamente com actores fora do sistema.
InternalWorker	Class	É um caso especial de trabalhador que interage com outros trabalhadores e entidades dentro do sistema.
Entity	Class	Uma entidade é uma classe passiva; isto é, os seus objectos não iniciam interacções por eles mesmos. Um objecto entidade pode participar em muitas realizações diferentes de casos de uso e geralmente vive para além de qualquer interacção.
Communicate	Association	É uma associação usada para definir que instâncias dos classificadores associados interagem.

Subscribe	Association	Uma associação "subscrição" entre duas classes denota que objectos da classe fonte (chamada de subscritor) serão notificados quando um evento em particular ocorrer em objectos da classe destino (chamada de classe publicitária). A associação inclui a especificação do conjunto de eventos que fazem com que um subscritor seja notificado.

Quadro 4.2 – Elementos do Perfil UML para a Modelação de Negócio

4.4. Organização do UML

4.4.1. Arquitectura de Quatro Camadas

Conforme expresso no documento de especificação da linguagem, o metamodelo do UML está definido como uma das camadas de uma arquitectura de metamodelação, a qual é constituída por quatro camadas: meta-metamodelo, metamodelo, modelo, e objectos de utilizador; as funções destas camadas encontram-se sumariadas na tabela que se segue.

Camada	Descrição	Exemplo
meta-metamodelo	A infra-estrutura para a arquitectura de metamodelação. Define a linguagem para especificar de metamodelos	*MetaClasse, MetaAtributo, MetaOperação*
metamodelo	Uma instância de um meta-metamodelo. Define uma linguagem para especificar um modelo.	*Classe, Atributo, Operação, Componente*
modelo	Uma instância de um metamodelo. Define conceitos relevantes de interesse de um domínio.	*Produto, Preço, definirPreço, ServidorPrincipal*
Objectos do utilizador (dados do utilizador)	Uma instância de um modelo. Define um domínio de informação específico.	*<Banana>, 50, definirPreço(50)* *<ServidorPiano>*

Quadro 4.3 – Arquitectura de metamodelação (adaptado de [OMG-UML01])

4.4.2. Estrutura do Metamodelo do UML

Devido à sua complexidade, a linguagem UML está decomposta em vários pacotes lógicos, os quais por sua vez poderão conter outros pacotes. Estes pacotes agrupam metaclasses que mostram uma forte coesão entre si e um acoplamento menos forte com metaclasses noutros pacotes.

O metamodelo está decomposto em vários pacotes[19] de topo (Figura 4.2):

[19] Por questão de clareza, e para evitar inconsistências ou traduções menos apropriadas para termos com tradução menos imediata ou adequada, a referência aos elementos que fazem parte do metamodelo é feita utilizando a língua em que foram expressos, ou seja, em Inglês.

- *"Foundation"*
- *"Behavioral Elements"*
- *"Model Management"*

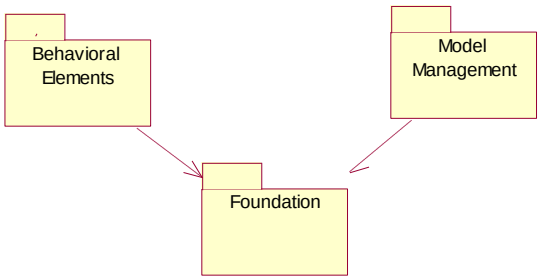

Figura 4.2 – Pacotes de topo do UML

Os pacotes "*Foundation*" e "*Behavioral Elements*" são ainda mais decompostos (Figura 4.3 e Figura 4.4):

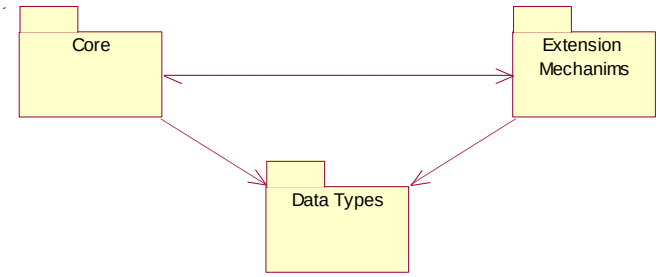

Figura 4.3 – Pacotes de Foundation

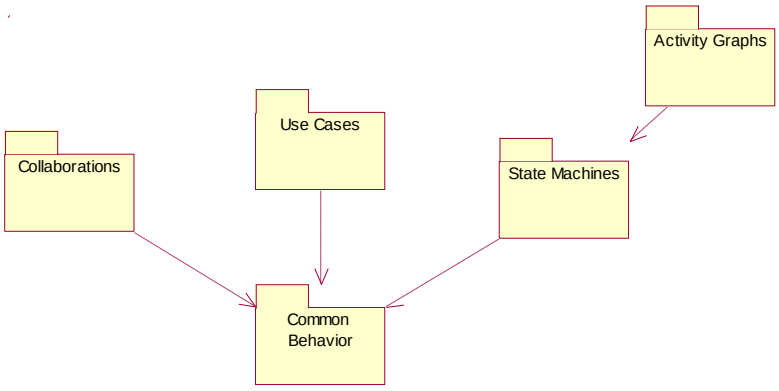

Figura 4.4 – Pacotes de Behavioral Elements

4.5. Descrição do Metamodelo

A descrição do metamodelo no documento de especificação do UML encontra-se feita de um modo semi-formal, recorrendo a uma sintaxe abstracta, a regras de boa formação, e a explicitação da semântica.

A sintaxe abstracta é fornecida como um modelo descrito num subconjunto do UML, consistindo de um diagrama de classes UML e suportado por descrição em linguagem natural. As regras de boa formação são fornecidas usando uma linguagem formal (OCL - Object Constraint Language) e linguagem natural. Finalmente, a semântica é descrita principalmente em linguagem natural, mas pode incluir alguma notação adicional, dependendo da parte do modelo sendo descrita.

Na construção do metamodelo UML foram utilizadas diferentes técnicas para especificar as construções da linguagem, usando algumas das capacidades do UML. As construções principais do UML estão expressas em **metaclasses** no metamodelo. Outras construções, em essência sendo variantes de outras, são definidas como **estereótipos** de metaclasses no metamodelo. Este mecanismo permite que a semântica da construção variante seja significativamente diferente da metaclasse base. Outra forma mais leve de definir variantes é através de **metaatributos**[20].

A descrição está nela própria contida, não sendo necessárias outras fontes de informação para ler o documento. A compreensão da especificação é prática uma vez que apenas é necessário um pequeno subconjunto de construções UML para descrever a sua semântica ("*Although this is a metacircular description*" [OMG-UML01, pág. 2-8]).

4.5.1. Formalismo da Linguagem

A descrição do metamodelo não é uma especificação completamente formal da linguagem, porque tal teria adicionado complexidade significativa sem um benefício claro. Contudo, à estrutura da linguagem é dada uma especificação precisa, o que é necessário para a inter-operacionalidade de ferramentas.

Para a sua apresentação do UML, o documento de especificação do UML recorre aos conceitos de:

- Sintaxe
- Sintaxe abstracta

[20] Pode-se por exemplo, verificar que no metamodelo do UML a construção agregação é definida por um atributo do elemento de metamodelo "*AssociationEnd*", o qual é usado para indicar se a associação é uma agregação ordinária, uma agregação composta, ou uma simples associação.

- Semântica estática
- Semântica dinâmica.

Para cada um destes conceitos, é feita de seguida uma breve introdução.

O documento de especificação considera comum para a especificação de linguagens, definir primeiro a sintaxe da linguagem e depois descrever a sua semântica estática e dinâmica. A **sintaxe** define quais as construções que existem na linguagem e como as construções são construídas em termos de outras construções. Devido a ter uma sintaxe gráfica, o documento define uma **sintaxe abstracta** da linguagem. A sintaxe concreta é então definida, fazendo-a corresponder à notação na sintaxe abstracta. A **semântica estática** de uma linguagem define como uma instância de uma construção deve ser conectada a outras instâncias para ter significado, e a **semântica dinâmica** define o significado de uma construção bem formada.

4.5.2. Estrutura da Especificação de um Pacote

Em cada pacote pode-se encontrar a especificação de:

- Sintaxe Abstracta
- Regras de Boa Formação
- Semântica

4.5.2.1. Sintaxe Abstracta

A sintaxe abstracta é apresentada em diagramas de classes que mostram as metaclasses que definem as suas construções e os seus relacionamentos. Os diagramas também ilustram algumas das regras de boa formação, principalmente requisitos de multiplicidade dos relacionamentos, e a indicação se as instâncias de uma determinada construção devem ou não estar ordenadas. Finalmente, é fornecida uma curta descrição informal em linguagem natural descrevendo cada construção.

A Figura 4.5 é um exemplo de como a sintaxe abstracta é expressa através de um diagrama de classes (neste caso, relativa a "*Backbone*" no pacote "*Core*").

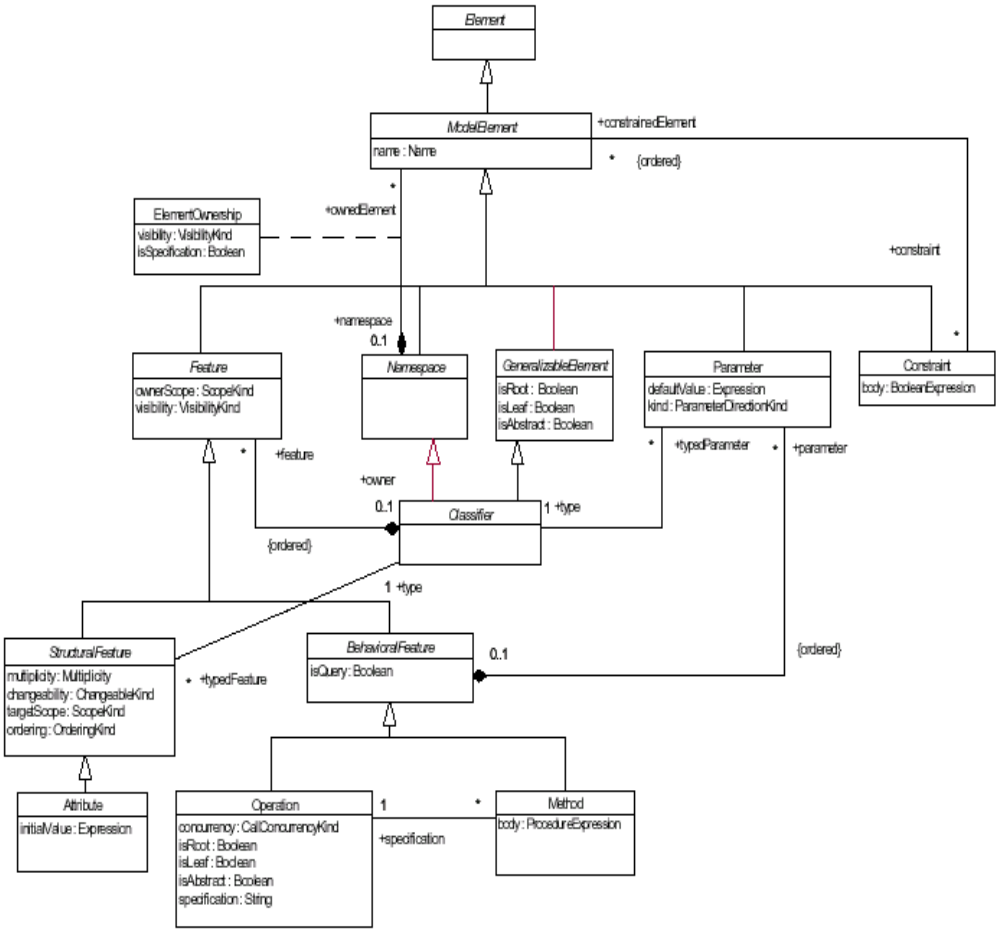

Figura 4.5 – Exemplo de diagrama de classes utilizado para expressar a sintaxe abstracta

Para cada um dos elementos incluídos na sintaxe abstracta, é apresentada uma *descrição informal*. Por exemplo para o elemento "*Class*" são fornecidas as descrições contidas no Quadro 4.4:

Class	*"A class is a description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. A class may use a set of interfaces to specify collections of operations it provides to its environment. In the metamodel, a Class describes a set of Objects sharing a collection of Features, including Operations, Attributes and Methods, that are common to the set of Objects.Furthermore, a Class may realize zero or more Interfaces; this means that its full descriptor (see Section 2.5.4.4, "Inheritance," on page 2-70 for the definition) must contain every Operation from every realized Interface (it may contain additional operations as well). …"*
Attributes isActive	*Specifies whether an Object of the Class maintains its own thread of control. If true, then an Object has its own thread of control and runs concurrently with other active Objects. Such a class is informally called an active class. If false, then Operations run in the address space and under the control of the active Object that controls the caller. Such a class is informally called a passive class.*
Stereotypes «auxiliary»	*Specifies a class that supports another more central or fundamental class, typically by implementing secondary logic or control flow. The class that the auxiliary supports may be defined explicitly using a Focus class or implicitly by a dependency relationship. Auxiliary classes are typically used together with Focus classes, and are* *particularly useful for specifying the secondary business logic or control flow of components during design. See also: «focus».*

«focus»	*Specifies a class that defines the core logic or control flow for one or more auxiliary classes that support it. Support classes may be defined explicitly using Auxiliary classes or implicitly by dependency relationships. Focus classes are typically used together with one or more Auxiliary classes, and are particularly useful for specifying the core business logic or control flow of components during design. See also: «auxiliary».*
«implementation»	*Specifies the implementation of a class in some programminglanguage (for example, C++, Smalltalk, Java) in which an instance may not have more than one class. This is in contrast to Class, for which an instance may have multiple classes at one time and may gain or lose classes over time, and an object (a child of instance) may dynamically have multiple classes. An Implementation class is said to realize a Type if it provides all of the operations defined for the Type with the same behavior as specified for the Type's operations. An Implementation Class may realize a number of different Types. Note that the physical attributes and associations of the Implementation class do not have to be the same as those of any Type it realizes and that the Implementation Class may provide methods for its operations in terms of its physical attributes and associations. See also: «type».*
«type»	*Specifies a domain of objects together with the operations applicable to the objects, without defining the physical implementation of those objects. A type may not contain any methods, maintain its own thread of control, or be nested. However, it may have attributes and associations. The associations of a Type are defined solely for the purpose of specifying the behavior of the type's operations and do not represent the implementation of state data. Although an object may have at most one Implementation Class, it may conform to multiple different Types. See also: «implementation».*
Inherited Features	*Class is a GeneralizableElement. The following elements are inherited by a child classifier, in addition to those specified under its parent, Classifier.*
isActive	*The child may be active when the parent is passive, but not vice versa. In most cases, they are the same.*

Quadro 4.4 – Exemplo da descrição informal de um elemento de metamodelo

4.5.2.2. Regras de Boa Formação

A semântica estática das metaclasses UML, excepto para restrições de multiplicidade e ordenação, são definidas como um conjunto de invariantes de uma instância da metaclasse (de notar que uma metaclasse não é requerida ter invariantes), tendo estas invariantes de ser verificadas para que uma construção tenha significado. Cada variante é expressa por uma expressão OCL juntamente com uma explicação da expressão.

Por exemplo, para o elemento "*Class*" são fornecidas as *regras de boa formação* constantes no Quadro 4.5.

[1]	If a Class is concrete, all the Operations of the Class should have a realizing Method in the full descriptor. **not** self.isAbstract **implies** self.allOperations->forAll (op | self.allMethods->exists (m | m.specification->includes(op)))
[2]	A Class can only contain Classes, Associations, Generalizations, UseCases, Constraints, Dependencies, Collaborations, DataTypes, and Interfaces as a Namespace.

```
self.allContents->forAll->(c |
c.oclIsKindOf(Class ) or
c.oclIsKindOf(Association ) or
c.oclIsKindOf(Generalization) or
c.oclIsKindOf(UseCase ) or
c.oclIsKindOf(Constraint ) or
c.oclIsKindOf(Dependency ) or
c.oclIsKindOf(Collaboration ) or
c.oclIsKindOf(DataType ) or
c.oclIsKindOf(Interface )
```

Quadro 4.5 – Exemplo de descrição formal de um elemento do metamodelo

4.5.2.3. Semântica

Os *significados das construções* são definidos em linguagem natural. Por exemplo, para o elemento "*Class*", é fornecida a descrição presente no Quadro 4.6.

Class

The purpose of a class is to declare a collection of methods, operations, and attributes that fully describe the structure and behavior of objects. All objects instantiated from a class will have attribute values matching the attributes of the full class descriptor and support the operations found in the full class descriptor. Some classes may not be directly instantiated. These classes are said to be abstract and exist only for other classes to inherit and reuse the features declared by them. No object may be a direct instance of an abstract class, although an object may be an indirect instance of one through a subclass that is non-abstract.

When a class is instantiated to create a new object, a new instance is created, which is initialized containing an attribute value for each attribute found in the full class descriptor. The object is also initialized with a connection to the list of methods in the full class descriptor.

Note – An actual implementation behaves as if there were a full class descriptor, but many clever optimizations are possible in practice.

Finally, the identity of the new object is returned to the creator. The identity of every instance in a well-formed system is unique and automatic.

A class can have generalizations to other classes. This means that the full class descriptor of a class is derived by inheritance from its own segment declaration and those of its ancestors. Generalization between classes implies substitutability; that is, an instance of a class may be used whenever an instance of a superclass is expected. If the class is specified as a root, it cannot be a subclass of other classes. Similarly, if it is specified as a leaf, no other class can be a subclass of the class.

Each attribute declared in a class has a visibility and a type. The visibility defines if the attribute is publicly available to any class, if it is only available inside the class and its subclasses (protected), if it can be used within the containing package (package), or if it can only be used inside the class (private). The targetScope of the attribute declares whether its value should be an instance (of a child) of that type or if it should be (a child of) the type itself.

There are two alternatives for the ownerScope of an attribute:

- *it may state that each object created by the class (or by its subclasses) has its own value of the attribute, or*
- *that the value is owned by the class itself.*

An attribute also declares how many attribute values should be connected to each owner (multiplicity), what the initial values should be, and if these attribute values may be changed to:

- *none - no constraint exists,*
- *frozen - the value cannot be replaced or added to once it has been initialized, or*

• addOnly - new values may be added to a set but not removed or altered.

For each operation, the operation name, the types of the parameters, and the return type(s) are specified, as well as its visibility (see above). An operation may also include a specification of the effects of its invocation. The specification can be done in several different ways (for example, with pre- and post-conditions, pseudo-code, or just plain text). Each operation declares if it is applicable to the instances, the class, or to the class itself (ownerScope). Furthermore, the operation states whether or not its application will modify the state of the object (isQuery). The operation also states whether or not the operation may be realized by a different method in a subclass (isPolymorphic). A method realizing an operation has the same signature as the operation and a body implementing the specification of the operation. Methods in descendants override and replace methods inherited from ancestors (see Section 2.5.4.4, "Inheritance," on page 2-70). Each method implements an operation declared in the class or inherited from an ancestor. The same operation may be declared more than once in a full class descriptor, but their descriptions must all match, except that the generalization properties (isRoot, IsAbstract, isLeaf) may vary, and a child operation may strengthen query properties (the child may be a query even though the parent is not). The specification of the method must match the specification of its matching operation, as defined above for operations. Furthermore, if the isQuery attribute of an operation is true, then it must also be true in any realizing method. However, if it is false in the operation, it may still be true in the method if the method does not actually modify the state to carry out the behavior required by the operation (this can only be true if the operation does not inherently modify state). The visibility of a method must match its operation.

Classes may have associations to each other. This implies that objects created by the associated classes are semantically connected; that is, that links exist between the objects, according to the requirements of the associations. See Association on the next page. Associations are inherited by subclasses.

A class may realize a set of interfaces. This means that each operation found in the full descriptor for any realized interface must be present in the full class descriptor with the same specification (see Section 2.5.4.4, "Inheritance," on page 2-70). The relationship between interface and class is not necessarily one-to-one; a class may offer several interfaces and one interface may be offered by more than one class. The same operation may be defined in multiple interfaces that a class supports; if their specifications are identical then there is no conflict; otherwise, the model is ill formed. Moreover, a class may contain additional operations besides those found in its interfaces.

A class acts as the namespace for various kinds of contained elements defined within its scope including classes, interfaces, and associations (note that this is purely a scoping construction and does not imply anything about aggregation), the contained classifiers can be used as ordinary classifiers in the container class. If a class inherits another class, the contents of the ancestor are available to its descendants if the visibility of an element is public or protected; however, if the visibility is private, then the element is not visible and therefore not available in the descendant.

Quadro 4.6 – Exemplo da definição de semântica de um elemento do metamodelo

4.5.3. A Hierarquia de Pacotes

A hierarquia de pacotes que organiza o metamodelo do UML (e os objectivos que regem cada um dos pacotes) está sintetizada no Quadro 4.7.

Pacotes	Subpacotes	Descrição
Foundation	***"The Foundation package is the language infrastructure that specifies the static structure of models."***	
	Core	*"The Core package specifies the basic concepts required for an elementary metamodel and defines an architectural backbone for attaching additional language constructs, such as metaclasses, metaassociations, and metaattributes."*
	Data Types	*"The Data Types package defines basic data structures for the language."*
	Extension Mechanisms	*"The Extension Mechanisms package specifies how model elements are customized and extended with new semantics."*
Behavioral Elements	***"This Behavioral Elements package is the language superstructure that specifies the dynamic behavior or models."***	
	Common Behavior	*"Common Behavior specifies the core concepts required for behavioral elements. "*
	Collaborations	*"The Collaborations package specifies a behavioral context for using model elements to accomplish a particular task."*
	Use Cases	*"The Use Case package specifies behavior using actors and use cases."*
	State Machines	*"The State Machines package defines behavior using finite-state transition systems."*
	Activity Graphs	*"The Activity Graphs package defines a special case of a state machine that is used to model processes."*
Model Management	***"The Model Management package specifies how model elements are organized into models, packages, subsystems, and UML profiles."***	

Quadro 4.7 – Pacotes do metamodelo do UML

No anexo C, encontram-se apresentados todos os pacotes presentes no UML, juntamente com a sintaxe abstracta neles presente.

4.6. Elementos do Metamodelo a serem considerados neste estudo

Conclui-se a apresentação da linguagem UML, com os elementos do metamodelo que se encontram definidos como metaclasses (Quadro 4.8, Quadro 4.9) e que são considerados neste estudo. Contudo, faz-se notar que no metamodelo existem outros elementos[21], os quais não sendo definidos como metaclasses no metamodelo do UML, constituem no entanto elementos que se consideram no estudo de adequação do UML ao FRISCO.

[21] Atributos, associações e papéis, estereótipos, etc.

Elemento de Metamodelo UML	Definido em Pacote	Elemento de Metamodelo UML	Definido em Pacote
ExtensionPoint	Use Cases	BooleanExpression	Data Types
Actor	Use Cases	LocationReference	Data Types
UseCase	Use Cases	Mapping	Data Types
UseCaseInstance	Use Cases	Multiplicity	Data Types
Include	Use Cases	MultiplicityRange	Data Types
Extend	Use Cases	Name	Data Types
Event	State Machines	ArgListsExpression	Data Types
Guard	State Machines	«enumeration» VisibilityKind	Data Types
StateVertex	State Machines	IterationExpression	Data Types
Pseudostate	State Machines	MappingExpression	Data Types
SynchState	State Machines	ProcedureExpression	Data Types
StubState	State Machines	Integer	Data Types
State	State Machines	ActionExpression	Data Types
CompositeState	State Machines	Expression	Data Types
SimpleState	State Machines	TimeExpression	Data Types
Transition	State Machines	«enumeration» ScopeKind	Data Types
SubmachineState	State Machines	«enumeration» PseudostateKind	Data Types
SignalEvent	State Machines	«enumeration» ParameterDirectionKind	Data Types
CallEvent	State Machines	«enumeration» OrderingKind	Data Types
TimeEvent	State Machines	«enumeration» ChangeableKind	Data Types
ChangeEvent	State Machines	«enumeration» CallConcurrencyKind	Data Types
StateMachine	State Machines	«enumeration» Boolean	Data Types
FinalState	State Machines	UnlimitedInteger	Data Types
Package	Model Management	String	Data Types
ElementImport	Model Management	«enumeration» AggregationKing	Data Types
Subsystem	Model Management	TypeExpression	Data Types
Model	Model Management	Geometry	Data Types
TaggedValue	Extension Mechanisms	ObjectSetExpression	Data Types
TagDefinition	Extension Mechanisms		
Stereotype	Extension Mechanisms		

Quadro 4.8 – Elementos do metamodelo do UML (continua no Quadro 4.9)

Elemento de Metamodelo UML	Definido em Pacote	Elemento de Metamodelo UML	Definido em Pacote
ElementResidence	Core	Signal	Common Behavior
Comment	Core	SubsystemInstance	Common Behavior
Abstraction	Core	DataValue	Common Behavior
Usage	Core	Instance	Common Behavior
Permission	Core	Stimulus	Common Behavior
Class	Core	DestroyAction	Common Behavior
Interface	Core	UninterpretedAction	Common Behavior
Node	Core	NodeInstance	Common Behavior
Dependency	Core	SendAction	Common Behavior
Artifact	Core	Object	Common Behavior
AssociationClass	Core	Exception	Common Behavior
DataType	Core	Reception	Common Behavior
Primitive	Core	Argument	Common Behavior
ProgrammingLanguageData Type	Core	Action	Common Behavior
Enumeration	Core	ActionSequence	Common Behavior
EnumerationLiteral	Core	CreateAction	Common Behavior
TemplateParameter	Core	CallAction	Common Behavior
TemplateArgument	Core	TerminateAction	Common Behavior
Component	Core	Link	Common Behavior
BehavioralFeature	Core	LinkEnd	Common Behavior
ModelElement	Core	AttributeLink	Common Behavior
ElementOwnership	Core	LinkObject	Common Behavior
Feature	Core	ComponentInstance	Common Behavior
Namespace	Core	ReturnAction	Common Behavior
GeneralizableElement	Core	Message	Collaborations
Parameter	Core	CollaborationInstanceSet	Collaborations
Constraint	Core	InteractionInstanceSet	Collaborations
Binding	Core	ClassifierRole	Collaborations
StructuralFeature	Core	AssociationEndRole	Collaborations
PresentationElement	Core	AssociationRole	Collaborations
Attribute	Core	Collaboration	Collaborations
Operation	Core	Interaction	Collaborations
Method	Core	ClassifierInState	Activity Graphs
Relationship	Core	ActivityGraph	Activity Graphs
Flow	Core	Partition	Activity Graphs
Generalization	Core	ActionState	Activity Graphs
Association	Core	ObjectFlowState	Activity Graphs
AssociationEnd	Core	CallState	Activity Graphs
Classifier	Core	SubactivityState	Activity Graphs
Element	Core		

Quadro 4.9 – Elementos do metamodelo do UML (continuação do Quadro 4.8)

5. Estudo da Adequação do UML ao FRISCO

O estudo da adequação do UML ao FRISCO não se revela tarefa fácil devido à diferença de origens e abordagens entre o UML e o FRISCO:

- O UML tem as suas origens na unificação de práticas correntes de desenvolvimento de sistemas de software orientados a objectos, com conceitos e técnicas para a especificação de artefactos e modelos de sistemas de software. Constata-se na sua especificação, através do metamodelo da linguagem, o esforço de procurar colocar em sintonia e interligação, os variados conceitos que a linguagem fornece. Pela mesma especificação, pode-se constatar que estes conceitos foram concebidos com o pensamento no desenvolvimento de sistemas de software, apesar de contudo advogar que, através de mecanismos de extensão presentes na especificação do UML, poder-se-ão modelar quaisquer outros sistemas que não de software. Não obstante a existência destes úteis mecanismos, um facto permanece verdade: a maioria do elementos presentes no UML estão pensados e organizados com a intenção de representar sistemas de software.
- O FRISCO teve motivações completamente diferentes; foi a observação da prática corrente da comunicação sobre conceitos de Sistemas de Informação que levou ao estudo das origens de sistemas de informação e à definição com rigor de um esquema de conceitos que clarificasse os conceitos relevantes na área de sistemas de informação.

5.1. Considerações gerais

Antes de se iniciar a análise da adequação dos elementos fornecidos pelo UML aos conceitos definidos no FRISCO, é importante expor e reflectir sobre várias questões que anunciam à partida, eventuais dificuldades em conseguir encontrar de forma clara, simples e imediata, a adequação satisfatória do UML ao FRISCO. Nas secções que se seguem, dão-se a conhecer estas questões, sobre as quais se tecem as necessárias considerações.

5.1.1. FRISCO, UML e Visões Ontológicas

Sempre afirmado como uma linguagem de modelação orientada a objectos,

> "*It is the visual modeling language of choice for building object-oriented and component-based systems.*"
>
> [OMG-UML01, pág. 1-3]
>
> *"The UML specifies a modeling language that incorporates the object-oriented community's consensus on core modeling concepts."*
>
> [OMG-UML01, pág1-7]

o UML incorpora naturalmente conceitos e princípios importantes da orientação aos objectos:

- objectos que representam uma coisa ou conceito do mundo real, possuidores de identidade própria, estrutura e comportamento, e que podem, estabelecer ligações entre si; colaboram através da troca de mensagens no sentido de providenciarem um pretendido comportamento;

- classes que representando colecções de objectos com a mesmas características, especificam a estrutura e comportamento dos objectos; através de relacionamentos entre classes, são especificadas as características das ligações que os objectos (considerados instâncias das respectivas classes) podem estabelecer entre si;

- herança (simples ou múltipla) entre classes e subclasses;

- princípios de encapsulamento, substituição e polimorfismo [Scott01, pág. 29].

Esta adesão ao paradigma de orientação a objectos contribui para o fundamental da visão ontológica do UML, ou seja, a sua forma de ver o "mundo"; a sua visão particular do "mundo" está reflectida, como em todas as linguagens, numa concepção de estrutura de conceitos. Esta concepção de estrutura de conceitos (e de todas as restrições sobre ela), é representada pelo metamodelo da linguagem, apresentado no documento de especificação.

O FRISCO não assume qualquer aproximação ao paradigma da orientação a objectos; apenas tece algumas afirmações que procuram expressar alguns défices ou qualidades nas abordagens orientadas a objectos:

> "*Most contemporary methods provide very weak facilities for rule extraction. Similarly, they do not allow handling large amounts of rules and structuring these in an efficient way. Object-oriented approaches suffer the same weakness, in spite of a pretence to the opposite*"
>
> [FRISCO98, pág. 19, par. 8]
>
> *"Positive trends may be observed in the emerging application of speech act theory and, to a limited extent, object-oriented methods for organisation analysis and design. "*
>
> [FRISCO98, pág. 21, par. 1]

Assim sendo, qual visão ontológica do FRISCO? O FRISCO, na procura de dependências conceptuais importantes para a compreensão da natureza dos sistemas de informação (os quais considera como existindo exclusivamente dentro de uma organização),

> "*Information systems exist exclusively within organization, to support their work, and to fulfil their information and communication requirements"*
>
> [FRISCO98, pág. 12, alínea a)]

vai embeber conhecimento da ciência da organização, ciência cognitiva, semiótica e ciência de sistemas. Baseando-se numa visão ontológica que permita identificar os conceitos relevantes, assume também para todo o raciocínio, uma posição filosófica construtivista[22].

Em conformidade com a afirmação,

> "*As a starting point, we present an overall view of the "world", including our metaphysical or philosophical position, in other words, our "Weltanschauung". It takes the form of a number of assumptions in which all concepts of interest are introduced and sketched informally*."
>
> [FRISCO98, pág. 27, par.3]

a visão ontológica assenta num conjunto de pressupostos estabelecidos pelo FRISCO. Pode-se com brevidade comentar os pressupostos do FRISCO[23]:

- os pressupostos de [a] a [c] dizem respeito ao reconhecimento de que o *mundo* existe, e de que os seres humanos possuem capacidades de *percepção* e de *concepção*;
- os pressupostos [d] e [e]: referem o mundo como sendo constituído por *coisas* que se relacionam, e que, algumas destas coisas são concebidas como tendo existência estática (estados) e outras como sendo alterações de estado (transições);
- o pressuposto [f] considera que existem transições que são efectuadas ou despoletadas por *actores* (tidos como coisas activas) sobre '*actands*'[24] (tidos como coisas passivas); estas transições são chamadas de *acções*; uma acção racional é dita ser em perseguição de um objectivo.
- os pressupostos de [g] a [j] referem que as pessoas usam *representações* para comunicar e construir uma realidade inter-subjectiva (reflecte a posição filosófica construtivista); estes pressupostos também definem conceito de *conhecimento*;

[22] Um construtivista é alguém que acredita que a "realidade" existe independentemente de qualquer observador, mas que é consciente do facto de apenas nos ser possível o acesso às nossas "concepções" mentais; para o construtivista, o relacionamento entre a realidade e a concepção é principalmente subjectivo e, pode ser sujeito a negociação entre observadores; qualquer acordo – a que se chama de "realidade inter-subjectiva" - pode ser adoptado de tempos em tempos [FRISCO98, pág. 26].

[23] Anteriormente apresentados no ponto 3.5.1 deste documento, e que se aqui encontram, em conformidade com a apresentação feita no relatório FRISCO, identificados de [a] a [o]

[24] É difícil arranjar uma tradução apropriada para este termo.

- os pressupostos [k] e [l] definem conceitos de *modelo* e de *sistema*;

- os pressupostos [m] e [n] definem e caracterizam uma *organização* como sendo um grupo de actores, juntamente com um conjunto de 'actands' tal que (a) é perseguido um objectivo comum ou é mostrada uma outra característica de coerência, e (b) as interacções são baseadas em comunicação. É esta comunicação (que engloba a troca de informação através de mensagens significativas) que permite a acção cooperativa e coordenada.

- o pressuposto [o] refere que no suporte às acções de uma organização, pode-se encontrar dentro desta uma "*rede complexa de fluxos de informação*"; alguns destes fluxos de informação podem ser colocados dentro de um "*subsistema de informação computorizado*".

Uma análise a estes pressupostos revela que a visão ontológica por estes definida, considera um mundo em que actores efectuam acções sobre "actands", sendo as acções possibilitadas pela comunicação de representações de informação através de mensagens; são considerados modelos, sistemas, organizações, e como parte destas, os sistemas de informação. Estes são os principais elementos considerados nesta visão ontológica; não são explicitamente tecidas considerações próximas das fundamentais da orientação a objectos (com a excepção para o reconhecimento de coisas que se podem relacionar, e o reconhecimento da existência de estados e de transições).

A diferença dos fundamentos base das visões ontológicas do FRISCO e do UML contribui para a dificuldade no estabelecimento de uma correspondência de representação adequada entre conceitos do FRISCO e elemento do metamodelo do UML.

5.1.2. FRISCO, UML e Sistemas de Informação

O que entende o FRISCO por "sistema de informação"? Será que objecto "sistema de informação" considerado pelo FRISCO coincide com o objecto "sistema de software" que o UML em essência se destina a representar?

Conforme já referido, é possível assumir a existência de quatro tipos de objectos distintos que se podem considerar como "sistema de informação"; consequentemente torna-se relevante colocar a questão: a que sistemas de informação se refere o FRISCO e o UML?

Relembra-se que Carvalho [Carvalho99] considera quatro tipos de objectos que podem ser chamados de "sistema de informação":

	Sistema de Informação
SI1	*Organização (sistema autónomo) cujo negócio (objectivo) é informar os seus clientes.*

SI2	*Um subsistema que assegure a comunicação entre os subsistemas de gestão e operacional num sistema autónomo.*
SI3	*Qualquer combinação de objectos activos que lidam apenas com objectos simbólicos e cujos agentes são computadores ou dispositivos baseados em computadores – um sistema baseado em computador.*
SI4	*Qualquer combinação de objectos activos (processadores) que lidam apenas com objectos simbólicos. Quando aplicado a uma organização, esta vista corresponde a todas as actividades organizacionais com excepção daquelas que lidam com matéria e energia.*

Quadro 5.1 – Objectos que podem ser considerados "sistemas de informação" (retirado de [Carvalho99])

O FRISCO define um "sistema de informação" como sendo:

> *Um sistema de informação é um subsistema de um sistema organizacional que compreende a concepção de como estão compostos os aspectos orientados à informação*[25] *e comunicação de uma organização, e de como estes operam, descrevendo portanto os arranjos e as acções (implícitas ou explícitas) orientadas à comunicação e fornecedoras de informação dentro dessa organização.*

Como enquadrar esta definição do FRISCO nas definições de sistema de informação oferecidas em [Carvalho99]? Para o FRISCO, um sistema de informação é um subsistema de um sistema organizacional, o que numa primeira análise, não irá facilmente ao encontro do objecto SI1 que Carvalho identificou como sendo sistema de informação. Põem-se de parte SI3 por apenas incluir como agentes dispositivos computacionais. Considerando a definição SI4 e a definição SI2 (SI2 que considera um sistema de informação como um subsistema de apoio aos subsistemas operacional e de gestão), o melhor enquadramento do sistema de informação do FRISCO é relativamente ao objecto SI4.

Uma pesquisa do termo "sistema de informação" no documento de especificação do UML [OMG-UML01], revela que, com a excepção no prefácio da referência,

> "*The Object Management Group, Inc. (OMG) is an international organization supported by over 800 members, including information system vendors…*"
>
> [OMG-UML01, pág. xxi]

e da referência,

> "*…OMT-2 was especially expressive for analysis and data-intensive information systems*"
>
> [OMG-UML01, pág. 1-11]

não existem quaisquer referências a sistemas de informação, o que é indicador da não consideração de aspectos fundamentais relacionados com a noção de sistema de informação.

Pode-se constatar no documento de especificação afirmações que indicam que o UML tem essencialmente como objecto de modelação um sistema de software:

[25] Nota-se aqui que o FRISCO define "informação" como o incremento de conhecimento.

"The Unified Modeling Language (UML) is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system."

[OMG-UML01, pág. 1-6]

"*Developing a model for an industrial-strength software system prior to its construction or renovation is as essential as having a blueprint for large building. Good models are essential for communication among project teams and to assure architectural soundness. ."*

[OMG-UML01, pág. 1-3]

"*The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems."*

[OMG-UML01, pág. 1-1]

"*The UML Profile for Business Modeling is an example profile that describes how UML can be customized for business modeling. Although all UML concepts can be brought to bear on this domain, but example emphasizes common stereotypes and some useful terminology. Note that UML can be used to model different kinds of systems (such as software systems, hardware systems, and real-world organizations)."*

[OMG-UML01, pág. 4-9]

Esta última definição em particular, feita no perfil do UML para modelação de negócios, revela que o uso do UML para modelação de outros sistemas que não os sistema de software, poderá por conveniência obrigar a usar os mecanismos de extensão do UML para definir terminologia, e, implicitamente, a expressão de conceitos relevantes para os sistemas em questão. Tal constitui uma evidência que o UML, apesar de poder ser utilizado (eventualmente recorrendo aos seus mecanismos de extensão) para modelar outros tipos de sistemas, foi pensado essencialmente para sistemas de software.

Considerando as definições de sistemas de informação propostas [Carvalho99], o sistema de software referenciado no UML corresponde ao sistema de informação SI3, podendo-se eventualmente ter como pequena diferença, o facto do UML considerar relevante, para a especificação da funcionalidade do sistema de software, o papel desempenhado por entidades externas ao sistema (as quais modeladas como actores, poderão ser não apenas dispositivos computacionais, mas também seres humanos).

O Quadro 5.2 sintetiza o objecto "sistema de informação" considerado pelo FRISCO e pelo UML, relativamente aos possíveis objectos sistemas de informação considerados em [Carvalho99].

Sistema de Informação	FRISCO	UML

SI1	*Organização (sistema autónomo) cujo negócio (objectivo) é informar os seus clientes.*		
SI2	*Um subsistema que assegure a comunicação entre os subsistemas de gestão e operacional num sistema autónomo.*		
SI3	*Qualquer combinação de objectos activos que lidam apenas com objectos simbólicos e cujos agentes são computadores ou dispositivos baseados em computadores – um sistema baseado em computador.*		**X**
SI4	*Qualquer combinação de objectos activos (processadores) que lidam apenas com objectos simbólicos.*	**X**	

Quadro 5.2 – Classificação dos SI considerados pelo FRISCO e pelo UML

Ter por objectos de interesse "sistemas de informação" distintos, não facilita a tarefa de fazer ou de esperar uma correspondência adequada entre os conceitos do FRISCO e UML.

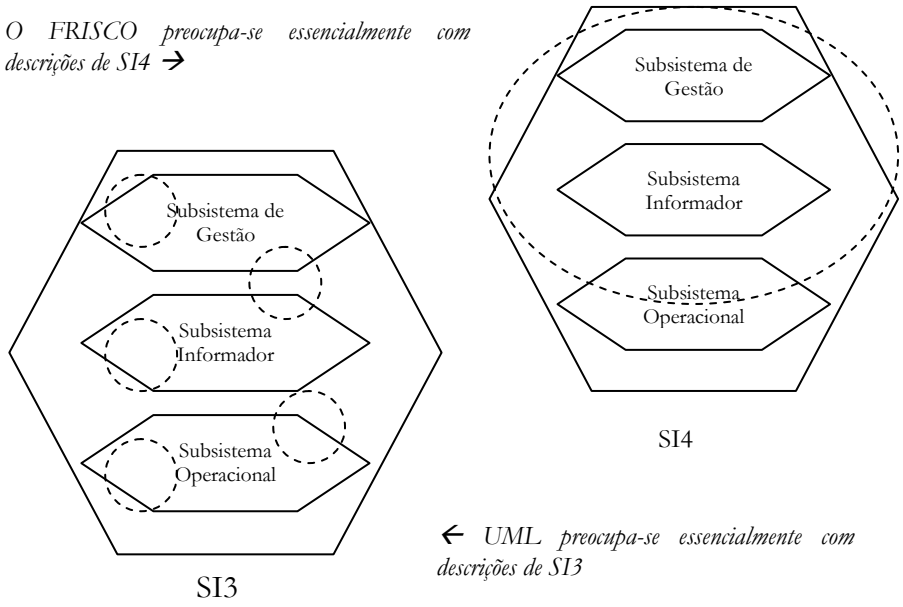

Figura 5.1 – Ilustração do enquadramento do FRISCO e UML relativamente a objectos SI

5.1.3. FRISCO, UML e os Quatro Mundos

Considerando os quatro mundos do desenvolvimento de sistemas de informação (a Figura 5.2 ilustra uma visualização destes mundos), coloque-se então a questão: para quais destes quatro mundos se preocupam principalmente o FRISCO e o UML em fornecer conceitos?

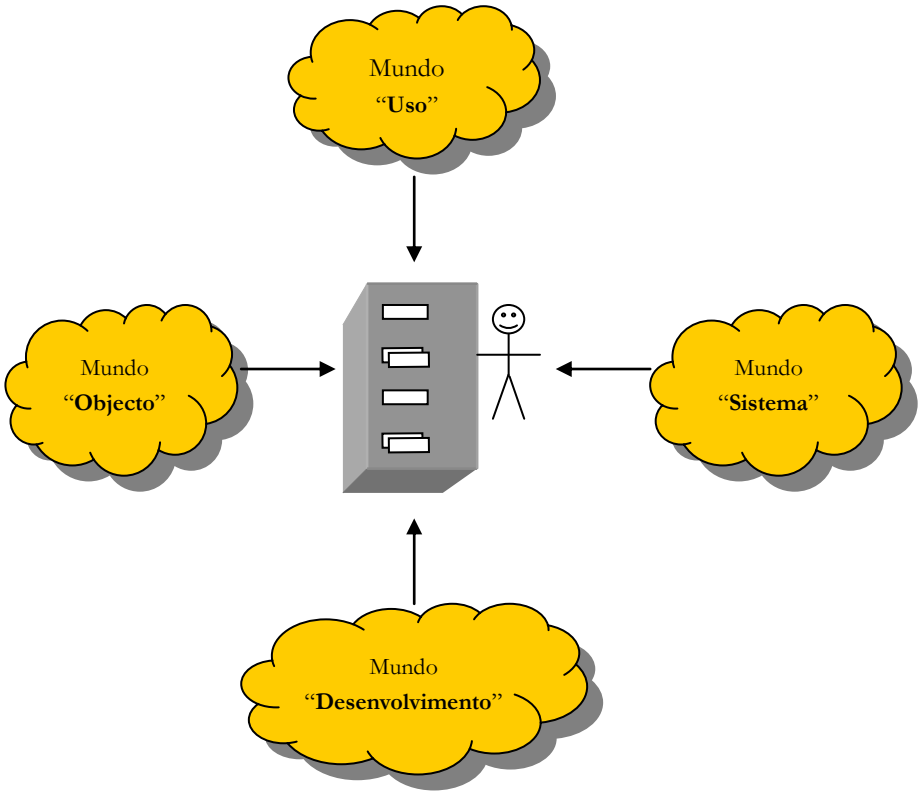

Figura 5.2 – Visualização dos quatro mundos

A resposta poderá não ser simples, devido a interpretações e considerações de carácter subjectivo e pessoal; no entanto, com mais ou menos adequação, não se estará muito longe da verdade ao dizer que, pelo que se conhece do FRISCO, este conterá elementos destinados ao mundo *objecto* e ao mundo *uso*. Pelo que se conhece do UML, este conterá elementos destinados ao mundo *objecto* e ao mundo do *sistema* (também é possível argumentar-se por outros mundos, mas no entanto, e dadas as considerações anteriores, consideram-se estes os que melhor vão de encontro com os objectivos principais e natureza do UML). O Quadro 5.3 sintetiza estas considerações.

Mundo	Caracterização do Mundo	FRISCO	UML
Objecto	*O mundo objecto é "Universo do Discurso" do sistema proposto, sobre o qual o SI final pode conter e manter informação.*	**X**	**X**
Uso	*O mundo do uso compreende a organização que usará e incorporará o SI, e.g., seus objectivos e estrutura, os indivíduos, suas crenças, hábitos de trabalho, intenções, etc.*	**X**	
Desenvolvimento	*O mundo do desenvolvimento compreende as pessoas envolvidas no desenvolvimento de SI, seus métodos, ferramentas e planos de trabalho, os artefactos que usam e produzem, incluindo modelos, etc.*		
Sistema	*O mundo do sistema compreende representações intermédias de vários aspectos do artefacto software a ser construído a vários níveis de abstracção, culminando no SI final.*		**X**

Quadro 5.3 – Enquadramento dos elementos do FRISCO e UML nos quatro mundos

Que mundos nos interessam considerar neste estudo? A resposta a esta questão poderá ser influenciada pelas nossas pretensões; atendendo que pretendemos saber o quanto bem o UML se adequa ao FRISCO, será mais apropriado considerar os conceitos nos mundos do FRISCO; para estes, procura-se encontrar no UML conceitos que melhor lhes adeqúem. Poderemos então dizer que é de interesse saber o quanto bem o UML representa os *mundos objecto* e *uso*, segundo a perspectiva do FRISCO.

5.1.4. FRISCO, UML e o Universo de Conceitos

Tendo-se nos pontos anteriores levantadas questões que nos levam a ter alguma prudência na abordagem da adequação do UML ao FRISCO, podem-se ainda levantar mais algumas a ter em conta, as quais se dão a conhecer neste e nos outros pontos que se seguem.

Seja levantada a questão: fornece o FRISCO um esquema de conceitos com pretensões de serem pragmaticamente utilizados na descrição/modelação de sistemas de informação, ou, um esquema com pretensões de ser teoricamente bem fundado e concebido (definindo conceitos meramente teóricos que nos quais se possam basear outros conceitos), o qual permita a utilização coerente e consistente de conceitos e terminologia na *comunicação* clara e não ambígua na área de sistemas de informação?[26] Ou ambos?

Para ajuda ao raciocínio, consideremos os seguintes factos. Por um lado,

- Carvalho [Carvalho94], com o objectivo de estabelecer um sistema de conceitos (COMOD - *Concepts For Information Systems Modelling*) a fim de ser utilizado na modelação de sistemas de informação, define e relaciona um conjunto de conceitos completo e pragmático, expressando-o através do que designamos por metamodelo.

- Para a modelação de negócios, Eriksson e Penker [Penker00] definem um metamodelo que expressa a sua visão dos conceitos fundamentais envolvidos no domínio do negócio;

- O UML, para modelação de sistemas de software (essencialmente), também define através de um metamodelo um conjunto de conceitos e terminologia útil[27].

Por outro lado,

- o FRISCO dá ênfase ao rigor a questões teóricas (como os acto de percepcionar, conceber e representar, como a questão de separar claramente entre conceitos e as suas denotações, como a definição de conhecimento partilhado, ou como a definição

[26] Esta questão é válida se implicitamente supusermos que o conjunto de conceitos necessários para a comunicação não tem de ser, em termos de necessidade, o mesmo conjunto para a modelação.

[27] O UML, apesar de independente de processos, sugere alguns tipos de diagramas (com utilização específica) para expressar modelos.

da sua visão ontológica através de um conjunto de pressupostos e o estabelecimento de uma posição filosófica como o construtivismo) que permitam estabelecer um esquema de conceitos coerente e bem fundado.

Procure-se através Figura 5.3 expressar graficamente a ideia:

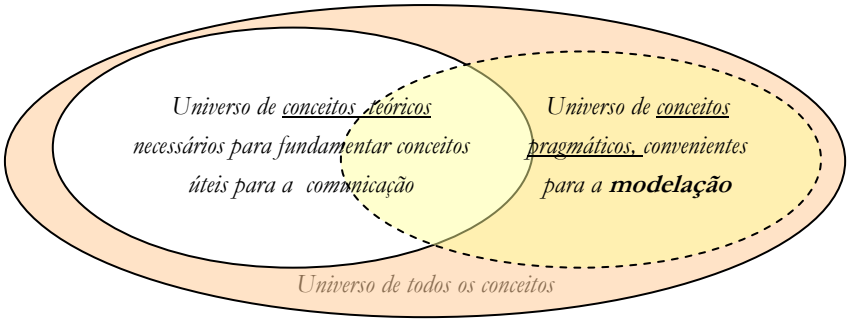

Figura 5.3 – Universos de Conceitos

Com que universo se preocupou o FRISCO? De facto, pode-se identificar como maior preocupação no FRISCO a fundamentação teórica (com inerentes pressupostos e conceitos de carácter mais teórico) relativa à área de sistemas de informação, com o objectivo de permitir uma clarificação e proposta de conceitos que tornem possível a comunicação sobre de sistemas de informação sem distorções de significado. Obviamente, ao definir conceitos de sistemas de informação, entre estes deverão encontrar-se aqueles que permitem descrever em termos genéricos[28] um sistema de informação; no entanto, serão estes suficientes para uma modelação efectiva, conveniente e útil? Talvez não. Contudo, estes conceitos gerais são relevantes, pois solidamente fundados, constituem uma referência que quaisquer pretensos esquemas ou linguagens de modelação de sistemas de informação não poderão ignorar.

5.1.5. FRISCO e as extensões ao UML

Foram apresentados neste documento, os perfis fornecidos pelo UML que permitem que esta linguagem possa com maior facilidade ser empregue para modelação de domínios mais específicos. Em reforço de que a linguagem UML necessita de ser "expandida" através dos mecanismos de extensão de linguagem, aquando da modelação de domínios específicos, apresentam-se de seguida mais alguns exemplos de extensões feitas à linguagem UML. Ver-se-á no final destas apresentações, que se torna legítimo questionar se não se poderia, através dos mecanismos de extensão do UML, desenvolver-se um perfil que adequadamente reflectisse o esquema de conceitos do FRISCO.

[28] "Genéricos" no sentido de que não são (salvo algumas excepções) consideradas "especializações" dos conceitos mais gerais, os quais são de conveniência e interesse para uma modelação concreta.

5.1.5.1. Outras Extensões ao UML

5.1.5.1.1. Extensões de Negócio de Eriksson-Penker

Eriksson e Penker [Penker00] ilustram como o UML pode[29], para além da modelação de sistemas de software, ser usado também para a modelação de negócios, sustentando que existem vantagens em utilizar os conceitos e técnicas orientadas a objectos:

> "*There are several advantages to using object-oriented concepts and techniques to a model a business: similar concepts(…)well-proven established techniques(…) standard notation(…)short learning curve(…) new and easier ways to view an organization or a business.*"
>
> [Penker00, pág.15].

Considerando que até um certo ponto, os elementos e diagramas do UML podem ser utilizados para criar modelos de negócios, tal não é tido como suficiente. Erikssson e Penker [Penker00] afirmam:

> "*However, certain concepts frequently used in business modeling are not part of the standard UML. Fortunately (…), UML as built-in extensions. By using these mechanisms it is possible to introduce and define business modelling concepts in UML*"
> [Penker00 pág. 64, par. 3]

> "*Because UML was initially to designed to describe aspects of a software system, it had to be extended to more clearly identify and visualize the important concepts of process, goals, resources and rules of a business system.*"
> [Penker00 pág. 66, par. 1]

Será interessante referir o enquadramento feito relativamente a modelos de negócios e sistemas de informação:

> "*Information systems will be designed to support these business models…*"[30]
>
> [Penker00, pág. 64, par. 4]

Eriksson e Penker [Penker00] são críticos relativamente ao perfil de Modelação de Negócios apresentado no documento de especificação do UML. Consideram que as extensões apresentadas não são fornecidas com qualquer explicação de como as usar, apenas descrevendo-as com pequenas descrições de texto (conforme se pode constatar na apresentação dos perfis efectuada no ponto 4.3 deste documento). Mais ainda, vêm estas extensões, baseadas na aplicação da modelação de casos uso para descrever negócios, como

[29] Os autores reconhecem que, alternativamente ao UML, existem outras formas de criar modelos de negócios (incluindo outras notações como IDEF0).

[30] Não se pode solidamente argumentar por um tipo de objecto que estes autores consideram como "sistema de informação", podendo-se dizer, que é possível ser a SI2 ou a SI3 (tipos de objecto em conformidade com o exposto no ponto 2.1).

sendo usadas por um produto de processos vendido por uma empresa[31]. Não partilham a perspectiva de se usar a modelação de casos de uso para a descrição de processos de negócio, apresentando mesmo as suas críticas: 3

> *"They [use cases] describe the functions provided by a black box system. However, viewing a business system as a black box doesn't offer sufficient insight into how the existing or future processes operate. (…)*"
>
> [Penker00, pág.66, último par.]

Sugerem em alternativa, um conjunto de extensões que formam uma estrutura básica de extensões de negócio para o UML, levando em conta a modelação de processos:

> "*The extensions merge UML with process modeling, so that it is easier to use UML for business modeling.*"
>
> [Penker00, pág. 67, par.1]

É proposto em [Penker00, pág. 65] um metamodelo que estabelece os conceitos básicos de negócio e seus relacionamentos; com base nestes conceitos, poderão ser construídos os modelos de negócio.

Este metamodelo é reproduzido neste documento na Figura 5.4, com o único intuito de constituir um contributo ao esclarecimento de qual a melhor abordagem a seguir para o estudo da adequação do UML ao FRISCO, incentivando-nos a reflectir se não seria de facto útil, considerar uma abordagem similar de definição de um perfil UML para o FRISCO (desta feita considerando o esquema de conceitos de sistemas de informação proposto pelo FRISCO).

[31] "Rational Software Corporation".

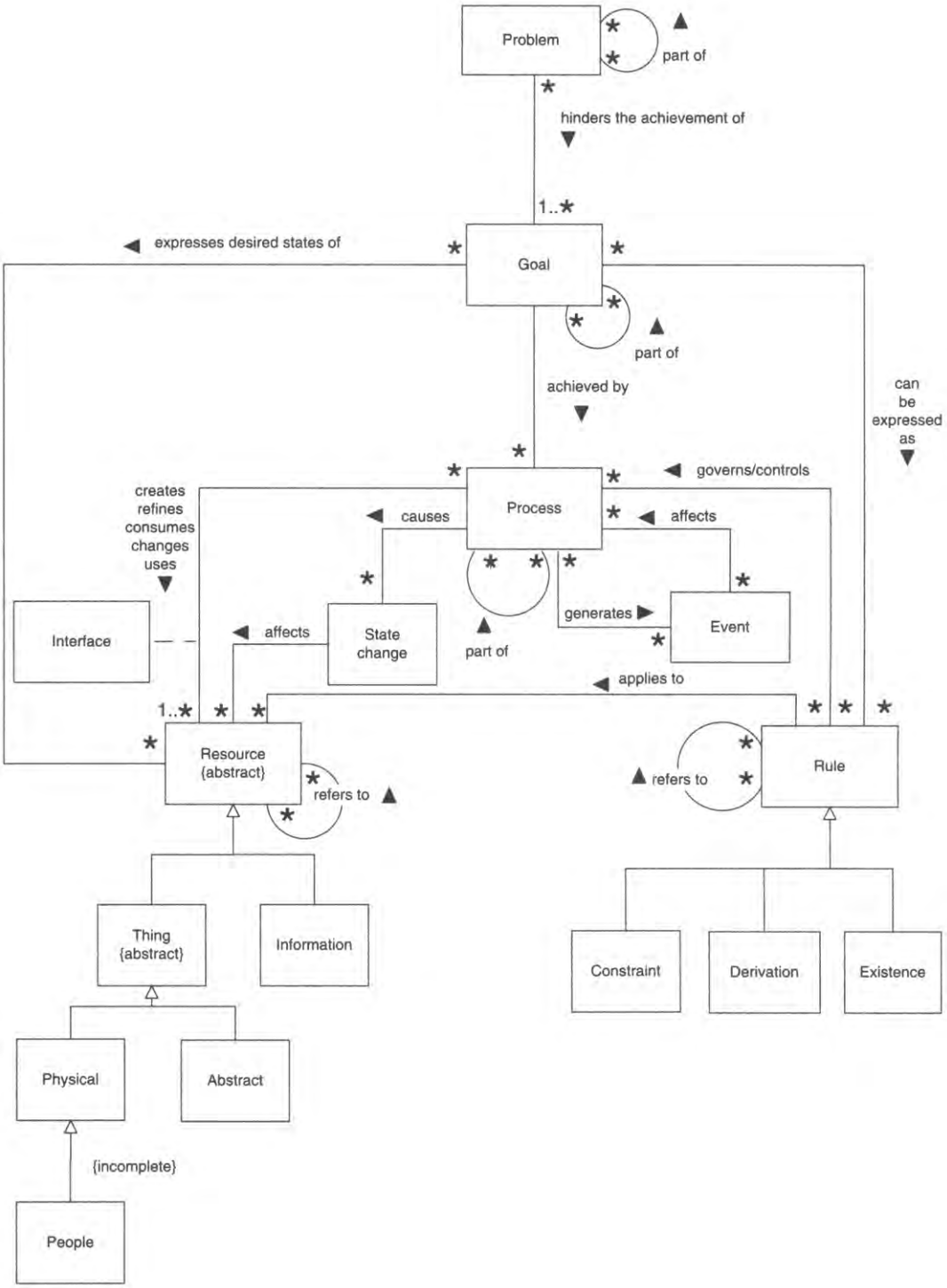

Figura 5.4 – Metamodelo de conceitos de negócio [Penker00]

5.1.5.1.2. Modelação de Aplicações Web

Em reforço da constatação de que se recorre a extensões do UML para a modelação de domínios com alguma especificidade, Conallen [Conallen99] afirma:

> "*The building blocks of UML – as they come out of the box – are just not sufficient to express the necessary subtleties of scripted Web pages as objects in a class diagram*"
>
> [Conallen99, pág. 149]

e apresenta as extensões[32] que entende necessárias para modelação de aplicações Web.

Por conseguinte, até no desenvolvimento de sistemas de software (ou seja o mesmo tipo de objecto de sistemas de informação – SI3), o UML necessita por vezes de ser expandido com extensões de forma a melhor se adequar para a modelação de um domínio mais particular.

5.1.5.2. Criação de Perfil UML para o FRISCO

Tendo em conta as considerações anteriores relativas aos objectos "sistema de informação", dos quatro mundos e dos universos de conceitos, poderemos (após as evidências apresentadas de se recorrer aos mecanismos de extensão do UML para a modelação de domínios específicos), questionar se o caminho a seguir será o de tentar estabelecer uma correspondência entre os conceitos do FRISCO e do UML, ou em alternativa, recorrer aos mecanismos de extensão para definir um perfil que permita introduzir alguma semântica e terminologia que facilite a representação dos conceitos do FRISCO com o UML. Neste estudo, não se prossegue este caminho (de definição de perfil) devido a se entender ser o interesse deste estudo focar os elementos da linguagem UML em si, e não novos elementos passíveis de definição através dos mecanismos de extensão da linguagem.

5.1.6. Síntese das Considerações

Foram apresentadas nas secções anteriores várias questões que devem ser tidas em conta no estudo que nos leve a concluir sobre a adequação do UML ao FRISCO. Faça-se uma súmula das questões levantadas nos pontos anteriores:

- Visão ontológica do FRISCO que permite ver um mundo de actores que interagem; uma visão ontológica do UML baseada nos princípios da orientação a objectos;
- Diferença entre objectos "sistema de informação" considerados pelo FRISCO e pelo UML;

[32] As quais se podem constatar em [Conallen99].

- A consideração dos quatro mundos presentes no desenvolvimento de sistemas de software;
- Ênfase do FRISCO em aspectos teóricos que sustentam um esquema de conceitos, com menos foco em aspectos pragmáticos da modelação de sistemas de informação;
- Possibilidade do uso dos mecanismos de extensão do UML (perfis).

Através das questões consideradas foi-nos permitido conhecer um conjunto de factores relevantes a este estudo, possibilitando-nos um melhor posicionamento para a tomada de decisões relativas à abordagem empreender.

5.2. Abordagem

Para se proceder à correspondência de representação dos conceitos do FRISCO por elementos da linguagem UML, foram identificadas duas possíveis abordagens.

5.2.1. Abordagens Possíveis

5.2.1.1. Correspondência de Representação Simples

Sem a consciência das questões anteriormente expostas, a tentativa de considerar de um lado os conceitos do FRISCO, e do outro lado os conceitos que se podem retirar a partir do metamodelo do UML, para que sequencialmente, do primeiro para o último elemento do FRISCO procurar o(s) elemento(s) adequado(s) no UML revelar-se-ia frustrante e consequentemente menos apropriada do que o desejável. Tal dever-se-ia certamente pelos motivos já explorados, agravados por um outro: constata-se que o metamodelo apresentado tem uma grande ênfase nos aspectos sintácticos, o que tira alguma coerência relativamente aos conceitos fundamentais relevantes para a modelação e subjacentes a uma visão ontológica. Adicionalmente, são numerosos os elementos presentes no metamodelo do UML, conforme se pode constatar no Quadro 4.8 e no Quadro 4.9 apresentados. A estes quadros, junta-se toda a informação presente na sintaxe abstracta do metamodelo do UML, o que nos dá a perceber a dimensão da informação a apreciar neste estudo.

5.2.1.2. Abordagem Alternativa

Em alternativa à correspondência de representação feita de uma forma sequencial e sistemática dos elementos do FRISCO para os elementos do UML sistemática, pode-se delinear uma outra que leve em linha de conta as várias considerações anteriormente referidas. A Figura 5.5 ilustra esta estratégia e a qual se passa de seguida a explicar e a justificar.

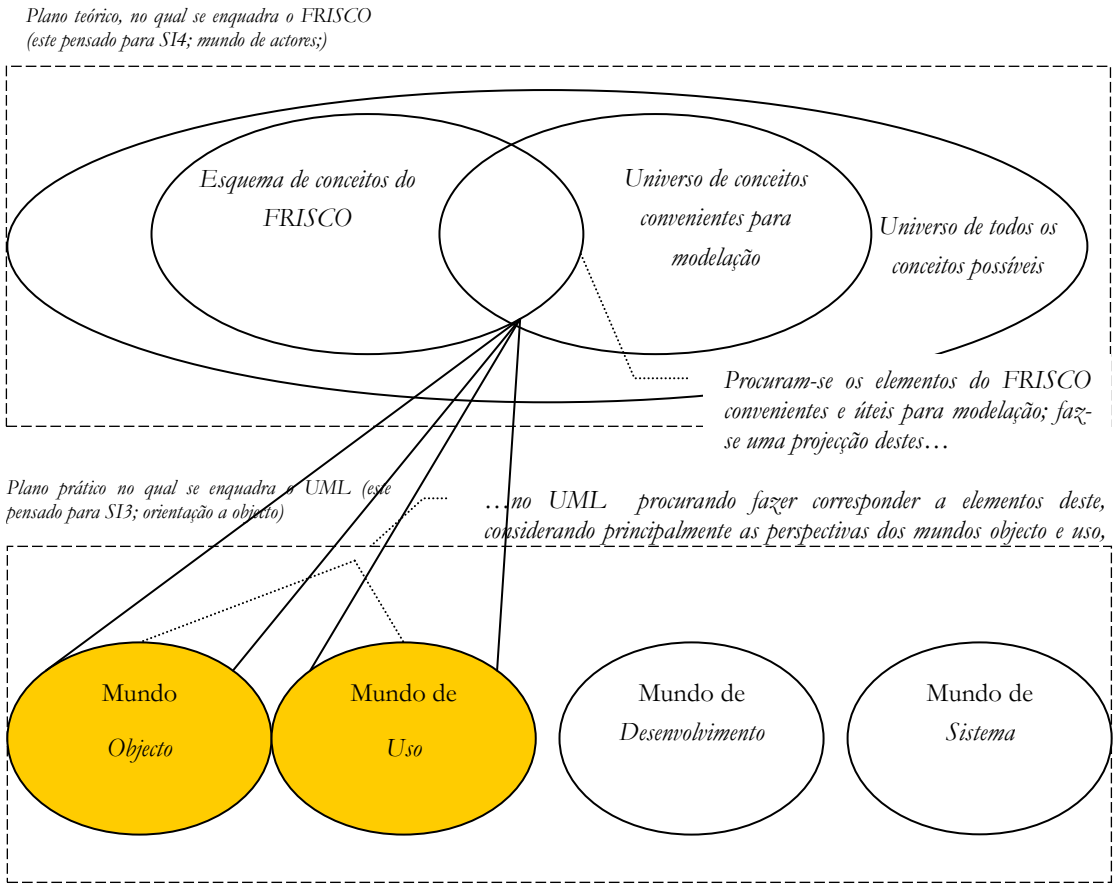

Figura 5.5– A abordagem alternativa

Sendo o UML uma linguagem essencialmente construída pela e para a prática de modelação, sem preocupações de fundamentação ontológica, e, tendo o FRISCO como preocupação principal fornecer um esquema de conceitos teoricamente fundamentados e apropriados para a comunicação (e não necessariamente todos aqueles que se possam em considerar úteis e convenientes para a modelação), interessa-nos identificar aqueles elementos do FRISCO que são úteis e passíveis de utilização na modelação; aparece assim na Figura 5.5, uma área de intersecção entre os designados "universos".

Após esta identificação, obtém-se um conjunto de conceitos (cada um deles solidamente fundamentados) para os quais se pode procurar elementos do UML que adequadamente os representem. Considerando os mundos no desenvolvimento de sistemas de informação, o UML como linguagem de modelação, poderá apresentar elementos concebidos para aplicação nestes mundos; destes elementos, interessa-nos identificar aqueles de aplicação nos mundos *objecto* e *uso* (mundos de interesse do FRISCO em conformidade com exposição feita no ponto 5.1.3). A Figura 5.5 mostra a projecção da intersecção dos elementos úteis para a modelação do FRISCO sobre os conceitos UML presentes nestes mundos. A análise desta projecção permitirá aferir a adequação do UML para representar os conceitos definidos do FRISCO. Mais ainda, fornecer-nos-á indicações da conveniência em utilizar as extensões do UML (eventualmente até de criar um perfil) para a dita representação. Sempre presentes em todo

este raciocínio, estarão as considerações relativas aos objectos de SI e aos pressupostos ontológicos.

Ocorrerá nesta altura, o pensamento de que poderá haver conceitos do FRISCO, que não sendo necessariamente de aplicação imediata na modelação, mas que contudo, sendo conceitos e terminologia da área de sistemas de informação, nos desperta também o interesse particular de saber se o UML também tem conceitos e terminologia similar. Portanto, a Figura 5.5 revela apenas uma faceta do estudo a ser feito; este deverá ser complementado por uma análise mais geral que permita revelar os conceitos e as respostas inerentes ao pensamento manifestado.

5.2.2. Abordagem Escolhida

Entre as duas abordagens expostas, escolhe-se prosseguir pela abordagem intitulada por "Correspondência de Representação Simples" em detrimento da abordagem alternativa. Embora possa à primeira vista não parecer a melhor opção, existem motivos que justificam esta escolha. Esta opção deve-se principalmente pelo facto da abordagem alternativa aparentar envolver trabalho acrescido, e de ser mais susceptível de discórdia em relação a opções a efectuar, as quais se podem traduzir por:

(1) A necessidade de separar no FRISCO, os conceitos de teóricos dos conceitos considerados úteis para a modelação;

(2) A necessidade de separar os elementos do UML nos diversos mundos considerados.

É facilmente reconhecível que estas separações têm um carácter subjectivo que influencia o resultado das separações, e consequentemente, o resultado poderá reflectir, de alguma forma, uma interpretação pessoal do que é útil ou não para a modelação.

A estes motivos, junta-se o facto de (embora reconhecendo que devido às considerações anteriormente tecidas, não seja fácil achar uma adequação dos elementos do UML ao FRISCO):

(1) A consciência das considerações anteriormente expostas ajuda a compreender porque se encontram dificuldades ao longo da sistemática procura de elementos adequados aos conceitos do FRISCO.

(2) A disposição dos conceitos pode ser a mesma que a apresentada no relatório do FRISCO, no qual são distribuídos por três grandes grupos associadas às etapas da construção do esquema conceptual de conceitos (os fundamentais, os relacionados com actor-acção-actand, e os restantes do esquema conceptual).

A ponderação destes factores levou então a que se optasse pela abordagem de correspondência de representação simples, de carácter sequencial, procurando seguir a mesma

ordem em que encontram dispostos os conceitos no relatório do FRISCO (embora por vezes, pontualmente e por conveniência, não se mantenha essa ordem). Também se agrupam os conceitos pelas consideradas "camadas" no relatório do FRISCO.

É importante notar que quando se procura um elemento no UML, esta procura é feita no metamodelo do UML, e não do documento de especificação UML em si; os elementos que definem a linguagem UML são os que estão de alguma forma presentes no seu metamodelo, quer sob a forma de metaclasses, quer sob a forma de estereótipos, atributos, papéis[33], ou outros.

No entanto, a escrita apresentada no documento de especificação [OMG-UML01] pode revelar conceitos ou terminologia que os autores do documento utilizaram ou acharam necessários utilizar para expressar o UML, mas que na verdade, só farão apenas parte da linguagem UML se de algum modo estiverem de facto presentes no metamodelo. Não obstante, ao não se encontrar (ou a não se achar viável) um elemento UML adequado para representação de um conceito no FRISCO, tem-se a atenção de procurar no documento algum conceito[34] que, embora não pertencendo ao metamodelo do UML, se ache útil referir neste estudo.

[33] Do inglês "*roles*".

[34] Por vezes ao longo dessa procura, usa-se neste documento a palavra "termo" como o mesmo significado.

5.3. Análise da Adequação

Após as considerações tecidas e a justificação da abordagem seguida, efectua-se então a análise sistemática dos conceitos do FRISCO e correspondentes elementos UML adequados à sua representação. Este ponto constitui o núcleo do trabalho envolvido neste estudo, e do qual extraem os resultados que permitirão concluir sobre a adequação do UML para representar os conceitos presentes no FRISCO.

Devido à dimensão da análise sistemática de cada um dos conceitos do FRISCO, optou-se por apresentá-la em anexo (anexo A) a este documento. Em jeito de síntese da análise efectuada, reúne-se (no Quadro 5.4) as afirmações proferidas relativas à adequação dos elementos do UML aos conceitos do FRISCO.

5.3.1. Síntese da Análise

Conceito do FRISCO	Análise de Adequação do UML relativa ao Conceito do FRISCO
Thing	O conceito de "Coisa" ("Thing") do FRISCO pode ser representado adequadamente por "*ModelElement*" do UML.
Predicator Predicated thing	Não se pode fazer adequadamente associar elementos do UML aos conceitos de "predicator" e de "predicated thing" presentes no FRISCO.
Relationship	Em rigor, um "relacionamento" ("Relationship") não pode ser correctamente representado por um elemento (ou vários) do UML, podendo-se apenas dizer que no UML existe um elemento, ligação ("*Link*"), que poderá parcialmente representar o conceito de "Relationship" do FRISCO.
Set membership Elementary Thing Composite Thing	Poder-se-á fazer corresponder o conceito "set membership" ao conceito agregação ou composição, os quais sendo variantes de uma associação, têm o objectivo de denotar um relacionamento todo/parte; a agregação permite que as suas partes possam pertencer a mais do que um todo, enquanto que numa composição as partes apenas podem pertencer a um todo. Assim, o UML distingue coisas compostas de coisas elementares através da existência ou não existência de relacionamentos de agregação de composição.
Entity	O UML não tem definido no seu metamodelo um elemento denominado por "entidade", apesar de utilizar este termo com frequência ao longo do seu documento. Uma análise ao metamodelo do UML revela que o elemento "*Instance"* é aquele que mais se aproxima em correspondência à definição de entidade ("Entity") no FRISCO.

Type Population, Instance	A correspondência assumida de "Type" para "*Classifier*" sofre de imperfeição, sendo contudo, aquela que melhor se pode fazer. Fazendo notar a observação anterior (relativa a "*Link*" como instância de "*Association*"), assume-se a correspondência de instância ("Instance") do FRISCO para "*Instance*" do UML. Uma pesquisa revela que o UML não define o termo população correspondente ao conceito "Population" (também não o define no metamodelo) para se referir à colecção de instâncias de um determinado tipo.
Transition	Assume-se a correspondência entre transição ("Transition") do FRISCO e a transição do ("*Transition")* do UML.
State Pre-State Pos-State	O elemento "*SimpleState"* é aquele que por definição de se aproximará à definição de "estado" ("State") do FRISCO. No UML, uma transição dá-se de um estado designado por estado fonte ("*source"*), para um designado por estado alvo ou destino ("*target"*).
State transition structure Composite Transition	Pode-se fazer corresponder "state transition estrutura" do FRISCO para "*Composite State*" do UML. De igual forma, atendendo à definição de "Composite transition" do FRISCO, também se considera a correspondência com *Composite State* do UML.
Transition occurrence	Não se assume a existência no UML de um conceito correspondente ao conceito de "Transition occurrence" presente no FRISCO.
Relative time Absolute time	O UML não define explicitamente conceitos que se possam fazer corresponder a estes de "tempo relativo" ("Relative time") e "tempo absoluto" ("Absolute time") do FRISCO.
Rule	Pode-se fazer corresponder para regra ("rule") do FRISCO, o conceito de constrangimento ("*Constraint*"), fazendo no entanto notar a imperfeição e limitações desta correspondência.
Actor	Com as devidas diferenças, pode-se associar o conceito de "actor" do FRISCO ao elemento "actor" presente no metamodelo do UML. Os actores no UML representam quer utilizadores humanos, quer outros sistemas.
Action Co-Action	Por coerência, faz-se corresponder "acção" ("Action") no FRISCO ao elemento "caso de uso" ("*UseCase*") do UML; uma co-acção terá de ser vista como a participação conjunta de mais do que um actor na interacção com um caso de uso.
Actand Input actand Output actand Resource	Também por coerência com a posição assumida relativamente a "actor" e "acção", e considerando a forma de o FRISCO ver o mundo como um mundo onde "actores que desenvolvem acções sobre "actands", opta-se por não fazer corresponder nenhum elemento do metamodelo do UML a "actand" do FRISCO. Consequentemente, não se consideram os conceitos de "Input actand", "Output actand" e "Resource" do FRISCO relativamente à sua representação pelo UML.
Composite action Action occurrence	Considerando o caso de uso como a melhor correspondência (apesar de limitada) para a acção, não haverá um elemento do UML adequado a este "Composite action" do FRISCO. Uma acção composta terá, por coerência ser vista como sendo composta por uma sequência/composição de casos de uso. O conceito de "ocorrência de acção" ("Action occurrence") corresponderá ao elemento do metamodelo "*UseCaseInstance"*.
Action context	No seu metamodelo, o UML não contempla nenhum elemento ao que se possa fazer corresponder à noção de "Action context".

Goal Goal-pursuing actor	Nem "goal" nem "goal-pursuing actor" do FRISCO encontram elementos no UML que os possam representar.
Domain Domain component Domain environment	Para os conceitos "domain", "domain component" e "domain environment" do FRISCO não encontram elementos adequados no metamodelo do UML para os representar. No entanto, define no seu glossário de termos, o termo "domain", e, faz uso genérico do termo "environment" no documento de especificação com mesmo significado que o conceito de ambiente do FRISCO. Relativamente ao termo "domain component" do FRISCO, não se encontra qualquer termo/conceito no UML ao qual se possa associar.
Human actor	O UML não apresenta um termo que distinga um actor humano ("Human actor") considerado no FRISCO de um actor não humano.
Perception Perceiving action Perceiver Conception Conceiving action Conceiver Conceiving context Interpreting action Interpreter Interpreting context	Todos estes termos ("Perception", "Perceiving action", "Perceiver", "Conception", "Conceiving action", "Conceiver", "Conceiving context", "Interpreting action", "Interpreter", "Interpreting context") existem no FRISCO em resultado da sua consideração da importância de distinguir o real, a percepção, a concepção e a representação. O UML não tem qualquer preocupação com conceitos a este nível.
Symbol Alphabet Symbolic construct Language	Pode-se verificar que não existe qualquer consideração relativamente à explicitação de conceitos similares a símbolo ("Symbol") e alfabeto ("Alphabet") do FRISCO; no entanto, pode-se fazer corresponder a noção de construção simbólica ("Symbolic construct") do FRISCO à noção de "construct" utilizada no documento do UML. O conceito linguagem ("Language") do FRISCO, é certamente em significado, o mesmo, quer no FRISCO quer no documento UML. Em suma, no metamodelo não se encontram elementos que possam representar estes conceitos de ("Symbol", "Alphabet", "Symbolic construct", "Language") do FRISCO.
Representation Representer Action Representer Representer Context	Poderemos fazer corresponder a "Representation" do FRISCO, o elemento do metamodelo UML "PresentationElement". Os conceitos associados aos termos "Representer", "Representer Action", "Representer Context" do FRISCO não são contemplados no UML.
Label Reference	O UML usa conceito de nome ("name"), propriedade de todos os elementos, como forma de referir a uma concepção, permitindo portanto, que se possa associar o elemento de metamodelo "*Name*" como possível representação "Label" do FRISCO. Não define no metamodelo elementos que permitam associar ao conceito "reference" do FRISCO, embora utilize e defina "reference" no seu glossário de termos,

Semiotic Level	O conceito de "Semiotic Level" do FRISCO, está a um nível teórico no qual o UML não tece considerações.
Model denotation System Denotation Information System Denotation	No FRISCO (em virtude de separar o que se concebe do que se representa), "Model denotation", "System Denotation", e "Information System Denotation", referem a representações das concepções "Model", "System" e "Information System" respectivamente. O UML em geral não se preocupa com este rigor teórico. Assim, quando é utilizado um destes termos, como por exemplo "Model", este tanto serve para referir a sua concepção como a sua representação.
Modelling action Modeller System viewer System representer	No metamodelo não existem elementos que possam representar quaisquer destes conceitos do FRISCO (Modelling action, Modeller, System viewer, System representer). Embora não os defina no seu glossário, o UML utiliza no seu documento os termos "modeler" e "modeling": o termo "modeler" para designar aquele que desenvolve actividade de modelação; "modeling" para designar esta actividade de modelação.
Model Metamodel Intensional model Extensional model	Para o conceito "Model" do FRISCO, o UML fornece o elemento "*Model*" do seu metamodelo. Para a concepção "metamodelo" ("Metamodel") do FRISCO, o UML permite representá-la através de um estereótipo «metamodel» do elemento "*Model*". Os termos "intensional model" e "extensional model" do FRISCO não se revêem em quaisquer dos elementos presentes no metamodelo.
System Sub-system System component System environment	Existe o conceito de subsistema ("*Subsystem*"), o qual permite não só representar o "sistema" ("System") do FRISCO (designado em UML por "*top-level susbsystem*"), como também um subsistema do próprio sistema, ou seja, o equivalente a "Sub-system" do FRISCO. "System component" do FRISCO não encontra elemento no UML que o possa adequadamente representar. O metamodelo não define explicitamente elemento que represente o ambiente de um sistema ("System environment" do FRISCO). Contudo, o documento de especificação utiliza o termo "ambiente do sistema" ("environment of the system" ou "system's environment") para se referir ao que envolve o sistema. Deve-se observar que em virtude de o objecto "sistema de informação" do UML ser SI3, os conceitos "System" e "Information System" sobrepõem-se no UML, tendo no contexto desta, o mesmo significado.
Dynamic system Static system, Active system, Passive System, Open system, Closed System	"Dynamic system", "Static system", "Active system", "Passive System", "Open system", "Closed System" são mais alguns exemplos de conceitos que não são considerados no UML.
Knowledge Information Data Shared knowledge	Quer no metamodelo, quer no documento de especificação do UML, não se encontram correspondências para os conceitos "Knowledge" e "Information" do FRISCO; não no metamodelo mas apenas no documento, encontra-se o termo "informação", o qual é utilizado com o significado equivalente ao conceito "dados" ("Data") definido pelo FRISCO. Sobre o conceito "Shared knowledge" do FRISCO, o UML não estabelece quaisquer termos ou considerações.

Message Message transfer Sender Receiver	"Message", "Message transfer", "Sender" e "Receiver" do FRISCO: o UML não tem presente no metamodelo um elemento claramente adequado a este conceito de "mensagem" no FRISCO (a não ser que se considere a colecção de argumentos). Pode-se ver "Message transfer" representado no metamodelo do UML pelo elemento "*Stimulus*". "Emissor" ("Sender") e "Receptor" (Receiver") do FRISCO encontram correspondência no metamodelo do UML, respectivamente, aos papéis de "Emissor" ("*sender*") e Receptor ("*receiver*") que as instâncias podem desempenhar num estímulo.
Communication	Considera-se então que o conceito "comunicação" ("communication") do FRISCO não encontra no metamodelo um elemento para o representar. Contudo, o termo "comunicação" é genericamente utilizado no documento de especificação do UML com o mesmo significado de "comunicação do FRISCO, salvaguardando contudo que no UML não é necessário que a comunicação envolva a presença de pelo menos dois actores humanos (o que para o FRISCO se torna necessário).
Organisational system Norm	Não estando no âmbito da especificação do UML, estes conceitos de "Organisational system" e "Norm" do FRISCO, não são definidos no UML.
Information system Computerised information sub-system (CISS)	Não se encontram no UML termos ou conceitos que claramente satisfaçam ou vão ao encontro das definições de "Information system" e de "Computerised Information sub-system (CISS)" do FRISCO. Na realidade, o conceito de "sistema de informação" e de "subsistema de informação computorizado" estão diluídos no mesmo conceito de "sistema físico" utilizado no documento do UML, mas não especificado no seu metamodelo.

Quadro 5.4 – Síntese da análise de adequação do UML ao FRISCO

5.3.2. Comentários aos Resultados Obtidos

A síntese exposta Quadro 5.4 pode ser ilustrada, para efeitos de melhor percepção dos resultados, por um diagrama que apresenta e relaciona os conceitos envolvidos. Este diagrama é apresentado na Figura 5.6.

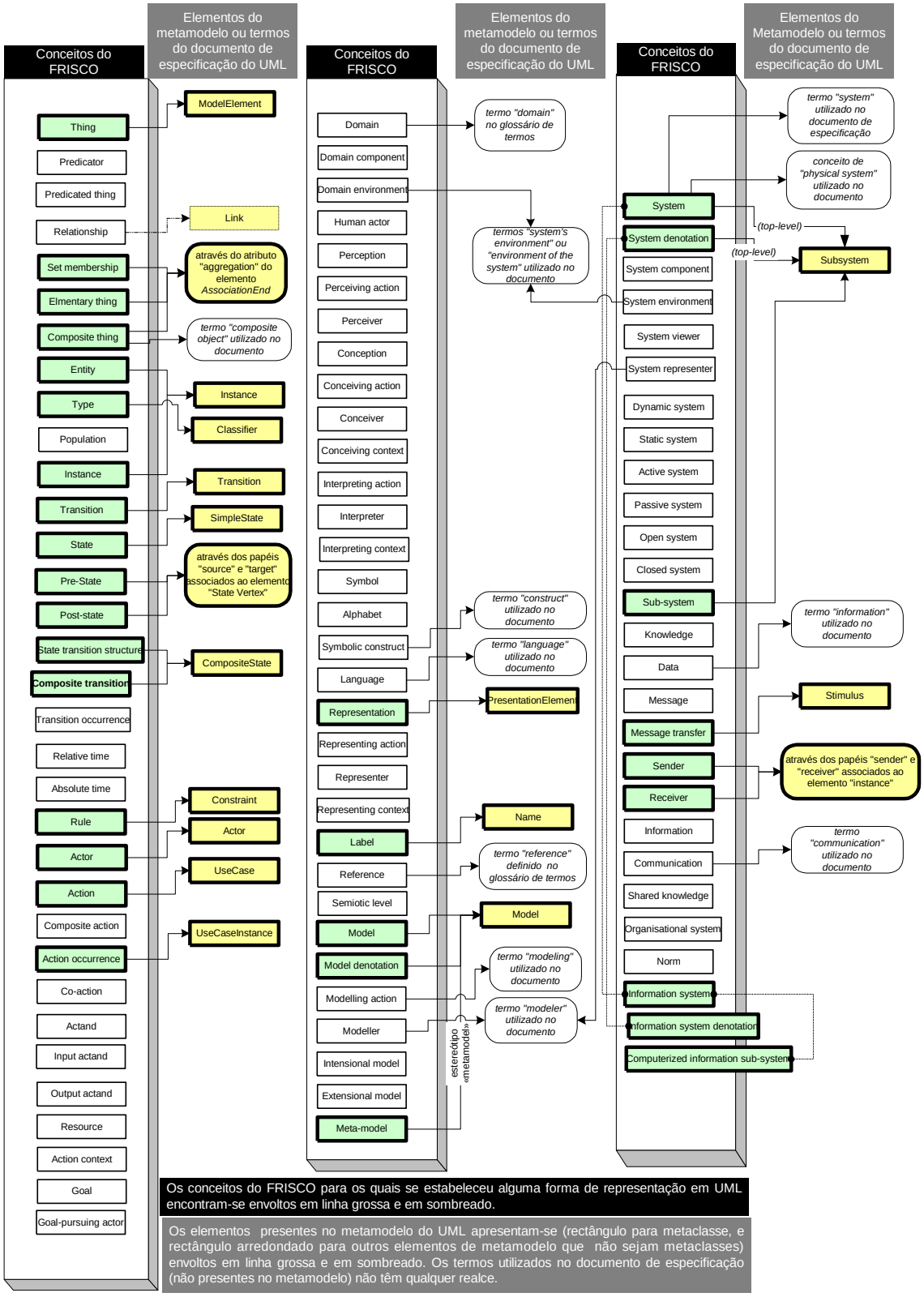

Figura 5.6 – Adequação dos elementos UML ao FRISCO

A análise da adequação efectuada revela que:

(1) Grande parte dos conceitos presentes no FRISCO não dizem propriamente respeito a conceitos de modelação. Existem muito conceitos do FRISCO para os quais não se encontram elementos adequados à sua representação; também se observa que vários conceitos do FRISCO são representados no UML pelo mesmo elemento.

(2) São utilizados conceitos (termos) no documento de especificação que, embora não estejam presentes como elementos no metamodelo da linguagem, poderiam corresponder a conceitos do FRISCO. A representação dos conceitos do FRISCO é feita em UML por *metaclasses* (na sua maioria), por *estereótipos* de metaclasses, por *atributos* de metaclasses, ou por *papéis* ("roles") que os elementos podem desempenhar numa associação.

(3) Ao contrário do que intuitivamente se poderia esperar, "Relationship" do FRISCO não coincide com a noção de "*Relationship*" do UML. Os conceitos de "Set membership", Elementary thing" e de "Composite Thing" não correspondem a elementos do metamodelo definidos como metaclasses, mas são representados implicitamente pelo atributo "*aggregation*" do elemento do "*AssociationEnd*" presente no metamodelo como uma metaclasse.

(4) O conceito "State" do FRISCO, apesar de poder ser considerado muito abrangente, faz-se correspondência de representação com "*Simple State*" do UML. Os conceitos de "Pre-state", "Pos-state" do FRISCO podem ser representados em UML através dos papéis "*source*" e "*target*" que os estados desempenham quando relacionados por uma transição. De forma similar, o mesmo acontece para os conceitos de "Sender" e "Receiver" do FRISCO, os quais são representados pelos papéis que associados ao elemento "*Instance*" do UML.

(5) "Meta-model" do FRISCO pode ser representado pelo estereótipo «metamodel» do elemento "*Model*"; É notória a não adequação do UML aos conceitos de "Knowledge", "Data" e "Information" definidos pelo FRISCO. Enquanto que o FRISCO define "System", "Information System", "Computerized information sub-system", no UML não se consegue de forma clara separar estes conceitos.

Uma vez que nesta análise se agrupou os conceitos do FRISCO em conformidade com a sua definição no relatório FRISCO, podem-se estimar quantos conceitos do FRISCO obtiveram correspondência no UML. Assim, verifica-se que 14 em 21 (66,7%) dos conceitos fundamentais encontraram elementos adequados para os representar, 3 em 12 (25%) dos conceitos de Actor, Acção e conceitos relacionados encontraram elementos no UML, e 14 em 59 (23,7%) acharam elementos UML adequados para os representar. Na totalidade, 31 em 92

(33,7%) dos conceitos do FRISCO encontraram representação através de elementos presentes no metamodelo do UML.

A observação da análise efectuada (em anexo A), permite reconhecer que a dificuldade de se adequar o UML ao FRISCO se deve às questões levantadas no ponto 5.1, as quais resultam das diferentes origens e objectivos do FRISCO e do UML.

6. Conclusão

Reconhecendo a importância do FRISCO como um estudo de referência no estabelecimento de um esquema de conceitos de Sistemas de Informação, e do UML como a linguagem de modelação standard com forte aceitação no mundo académico e na indústria, o motivo deste trabalho prendeu-se com o facto do esquema de conceitos proposto pelo FRISCO carecer de linguagens de modelação que o ponham em prática. Este estudo pretendeu analisar até que ponto a linguagem UML é adequada para a descrição/modelação de sistemas de informação, com base no esquema de conceitos de sistemas de informação proposto pelo FRISCO.

6.1. Trabalho desenvolvido e principais conclusões

Para a concretização deste estudo procurou-se conhecer o relatório produzido pelo "IFIP WG 8.1 Task Group FRISCO" e os conceitos subjacentes ao metamodelo da linguagem UML. Devido a alguma complexidade inerente à compreensão do metamodelo da linguagem UML, houve a necessidade de se recorrer a literatura que facilitasse e permitisse a compreensão do UML. Depois de analisado o documento de especificação da linguagem UML e revista outra literatura conveniente para este estudo, estabeleceu-se a abordagem a empreender para análise,. A análise foi desenvolvida procurando a melhor correspondência de representação entre os conceitos do FIRSCO e os elementos fornecidos no metamodelo da linguagem UML. Procedendo-se a uma síntese, reflectiu-se sobre os resultados obtidos.

Os resultados levam a concluir que a linguagem UML, na sua forma original (sem recorrer aos seus mecanismos de extensão) não fornece elementos adequados para representação do esquema de conceitos FRISCO. Tal vem dar razão e fundamento às considerações previamente efectuadas neste estudo. Coloca-se no entanto, a possibilidade de se recorrer aos mecanismos de extensão da linguagem UML (a exemplo do que já é feito para outros domínios e esquemas de conceitos) para se poder adequadamente representar o esquema de conceitos de sistemas de informação proposto pelo FRISCO.

6.2. Dificuldades

Podem-se, de forma sucinta, expor as principais dificuldades sentidas:

- A significativa componente teórica associada a visões ontológicas e esquemas de conceitos e, a complexidade inerente à análise de adequação;

- O esforço de leitura subjacente ao estudo e compreensão do metamodelo UML;
- O facto de não se estar em exclusividade para o desenvolvimento deste estudo, o que obrigando a constantes interrupções, vem em agravo à dimensão do esforço a ser desenvolvido.

6.3. Trabalho futuro

Verificando-se a existência na literatura de extensões para diversos domínio, levanta-se a possibilidade de, recorrendo aos mecanismos de extensão da linguagem UML, criar um perfil UML que se adeqúe à representação do esquema de conceitos do FRISCO. Tal criação poderia ser objecto de um trabalho futuro.

Com objectivos semelhantes ao deste estudo, seria interessante desenvolver um trabalho para analisar a adequação da linguagem UML à representação de outros esquemas de conceitos.

Outro trabalho seria o de reflectir sobre que alterações se poderiam fazer na especificação da linguagem UML, para que esta melhor representasse um esquema de conceitos solidamente fundamentados.

Um trabalho também interessante seria averiguar que esquemas de conceitos e subjacentes visões ontológicas estão presentes no actual ensino de sistemas de informação e, neste mesmo trabalho ou num outro, propor um esquema de conceitos (e visão ontológica) que permitisse uma uniformização dos conceitos no ensino de sistemas de informação.

Será que a linguagem de modelação UML pode de facto ser a base para a modelação dos diferentes e diversos domínios de problema? Poderá a possibilidade do livre uso de mecanismos de extensão levar à criação de inúmeros perfis e, no fundo, à criação de "novas linguagens"? Dever-se-iam normalizar os perfis para as várias áreas, perfis esses sustentados por uma linguagem comum (UML)? A resposta a estas questões e outras subjacentes poderiam constituir motivo de outro trabalho.

Mais estudos e trabalhos poderiam ser propostos; no entanto termina-se com a sugestão de uma área que pode ser interessante para a identificação de conceitos e para a modelação usando a linguagem UML: a área de sistemas de tempo real.

Referências

Booch99 — Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999

Carvalho94 — Carvalho, João A. e Luis Amaral, *Using an Explicit System of Concepts for Information Systems Modelling: COMOD*, Zupancic, J. e S. Wryzca (eds.), Proceedings of the 4th Intenational Conference Information Systems Development- ISD'94: Methods & Tools, Theory & Practice, Bled, Slovenia, 20-22 September 1994, pp. 93-103.

Carvalho99 — Carvalho, João A., *Information System? Which One Do You Mean?*, in Falkenberg, E., K. Lyytinen e A. Verrijn-Stuart (Eds.), *Information Systems Concepts: An Integrated Discipline Emerging* , Kluwer Academic Publishers, 2000, pp. 259-280 (Proceedings of "ISCO 4 - Information Systems Concepts: An Integrated Discipline Emerging", Leiden, Holanda, 20 a 22 de Setembro de 1999).

Conallen99 — Jim Conallen. *Building Web Applications with UML*, Addison-Wesley, 1999

Fowler00 — Martin Fowler, Kendal Scott. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 2ª Edição, Addison-Wesley, 2000

FRISCO98 — Eckhard D. Falkenberg, Wolfang Hesse, Paul Lindgreen, Bjorn E. Nilsson, J. L. Han Oei, Colette Rolland, Ronald K. Stamper, Frans Assche Alexander A. Verrijn-Stuart, Klaus Voss. *The FRISCO Report*, IFIP, 1998

Furlan98 — José David Furlan. *Modelagem de Objectos através da UML*, Makron Books, 1998

ISCO89 — Proceedings IFIP WG 8.1 Conference ISCO1, Namur, Belgium, October 1989, North-Holland, 1989

ISCO92 — Proceedings IFIP WG 8.1 Conference ISCO2, Alexandria, Egypt, April 1992, North-Holland, 1992

ISCO95 — Proceedings IFIP WG8.1 Conference ISCO3, Marburg, Germany, March 1995, Chapman & Hall, 1995

Jacobson99 — Ivar Jacobson, Grady Booch and James Rumbauch. *The Unified Software Development Process*. Addison-Wesley, 1999

Jarke92 — Jarke, M., Mylopoulos, J., Schmidt, J. W. and Vassiliou, Y., DAIDA: *An Environment for Evolving Information Systems*, ACM Transactions on Information Systems (TOIS) 10(1), 1{50 (1992).

Kobryn99 — Cris Kobryn. *UML 2001: A Standardization Odyssey*, Communications of the ACM", vol. 42, nº 10, 1999

Lindgreen90 — P. Lindgreen (ed.): A Framework of Information System Concepts, Interim Report of the IFIP WG 8.1 Task Group FRISCO, May 1990

Lindgreen98 — Paul LindGreen, *FRISCO – A Way to Insight ? Reflections on a Challenging and Eye-opening Project that Missed a Goal*, FRISCO Report, IFIP, 1998

Mylopoulos92 — Mylopoulos, J., *Knowledge Representation in Information Systems,* Tutorial Notes, CAiSE*92, The Fourth International Conference on Advanced Information Systems Engineering, Manchester/UK, 1992.

Oestereich99 — Bernd Oestereich. *Developing Software with UML: Object-Oriented analysis and design in practice*. Addison-Wesley, 1999

OMG-UML01 — OMG, *Unified Modeling Language Specification v1.4*, http://www.omg.org/technology/documents/formal/uml.htm

Opdahl01 — Andreas L. Opdahl e Brian Henderson-Sellers, *Grounding the OML metamodel in ontology*, Journal of Systems and Software 57 (2) (2001) pp. 119-143

Opdahl99 — Andreas L. Opdahl, Brian Henderson-Sellers and Frank Barbier. *An Ontological Evaluation of the OML Metamodel.* In "*Information System Concepts: An Integrated Discipline Emerging*, Eckhard D. Falkenberg and Kalle Lyytinen (eds.), IFIP/Kluwer, 1999.

Penker00 — Hans-Erik Eriksson and Magnus Penker. *Business Modeling with UML*, John Wiley &Sons, 2000

Penker98 — Hans-Erik Eriksson and Magnus Penker. *UML Toolkit*, John Wiley &Sons, 1998

Rumbaugh98 — James Rumbaugh, Ivar Jacobson, Grady Booch *The Unified Modeling Language Reference Manual*, Addison-Wesley Object Technology Series, 1998

Schneider98 — Geri Schneider, Jason P. Winters. *Applying Use Cases: A Practical Guide*, Addison-Wesley, 1998

Scott01 — Scott, Kendall. *UML Explained*, Addison-Wesley, 2001

Silva01 Alberto Silva, Carlos Videira. *UML, Metodologias e Ferramentas CASE,* Edições Centro Atlântico, 2001

Stamper98 Ronald K. Stamper. *A Dissenting Position*, FRISCO Report, IFIP 1998

Wand99 Wand, Y. and Weber, R.. *On the Ontological Expressiveness of Information Systems Analysis and Design Grammars*, Journal of Information Systems 3, 217-237 (1993).

Warmer99 Jos Warmer, Anneke Kleppe. *The Object Constraint Language: Precise Modeling with UML*, Addison-Wesley, 1999

Anexos

Apresentam-se de seguida os anexos a este documento:

Anexo A - Análise da adequação do UML ao FRISCO

A.1 - Conceitos Fundamentais

A.2 - Actor, Acções e Conceitos associados

A.3 - Restantes Conceitos

Anexo B - Pressupostos e Conceitos do FRISCO

B.1 - Pressupostos

B.2 - Apresentação Informal dos Conceitos

Anexo C - Pacotes e Elementos do Metamodelo do UML

C.1 - Pacote "Foundation

C.2 - Pacote "Behavioral Elements

C.3 - Pacotes e Elementos do Metamodelo do UML

A. Análise da adequação do UML ao FRISCO

Nas secções que se seguem, procede-se à análise dos conceitos envolvidos em cada das etapas de construção do esquema conceptual do FRISCO, etapas essas que foram[35]:

(1) Apresentação dos conceitos mais fundamentais - um conjunto de conceitos coerentes e logicamente consistentes para ver o mundo:

1) Thing
2) Predicator
3) Predicated thing
4) Relationship
5) Set membership
6) Elementary thing
7) Composite thing
8) Entity
9) Type
10) Population
11) Instance
12) Transition
13) State
14) Pre-state
15) Post-state
16) State-transition structure
17) Composite transition
18) Transition occurrence
19) Relative time
20) Absolute time
21) Rule

(2) Apresentação das noções de actor, acção e outros conceitos relacionados:

22) Actor
23) Action
24) Composite action
25) Action occurrence
26) Co-action
27) Actand
28) Input actand
29) Output actand
30) Resource
31) Action context
32) Goal
33) Goal-pursuing actor

[35] Etapas que surgem após a etapa inicial respeitante à apresentação da visão geral do mundo, referenciada por "Weltanschauung" e expressa sob a forma de um conjunto de pressupostos (é sobre esta forma de ver o mundo que o FRISCO sustenta os conceitos definidos nas restantes etapas).

(3) Apresentar os vários conceitos ultimamente necessários para o esquema conceptual:

34) Domain
35) Domain component
36) Domain environment
37) Human actor
38) Perception
39) Perceiving action
40) Perceiver
41) Conception
42) Conceiving action
43) Conceiver
44) Conceiving context
45) Interpreting action
46) Interpreter
47) Interpreting context
48) Symbol
49) Alphabet
50) Symbolic construct
51) Language
52) Representation
53) Representing action
54) Representer
55) Representing context
56) Label
57) Reference
58) Semiotic level
59) Model
60) Model denotation
61) Modelling action
62) Modeller
63) Intensional model
64) Extensional model
65) Meta-model
66) System
67) System denotation
68) System component
69) System environment
70) System viewer
71) System representer
72) Dynamic system
73) Static system
74) Active system
75) Passive System
76) Open system
77) Closed System
78) Sub-system
79) Knowledge
80) Data
81) Message
82) Message transfer
83) Sender

84) Receiver
85) Information
86) Communication
87) Shared knowledge
88) Organisational system
89) Norm
90) Information system
91) Information system denotation
92) Computerised information sub-system (CISS)

É com base nesta ordem dos conceitos que se faz a análise da adequação do UML para a representação do esquema de conceitos do FRISCO.

A.1. Conceitos Fundamentais

Estes conceitos, caracterizados como *"a coherent and logically consistent set of concepts for viewing the "world"* [OMG-UML01, pág. 27], constituem uma camada fundamental de conceitos e, são definidos para posteriormente permitirem a apresentação dos conceitos de actor e acções, e de outros conceitos relacionados. Começa-se então a análise dos conceitos pela ordem em que foram definidos no FRISCO. Para cada conceito do FRISCO apresenta-se a sua definição; de seguida apresentam-se várias considerações (recorrendo a extractos de metamodelo[36] e de definições ou afirmações presentes no UML) que permitirão encontrar (eventualmente) no UML um elemento que se adeqúe (com maior ou menor adequação) ao conceito definido pelo FRISCO.

Começa-se então pelo primeiro conceito apresentado pelo FRISCO, o conceito de "Coisa" ("Thing") do FRISCO.

Thing

> *"A **thing** is any part of a conception of a domain (being itself a "part" or "aspect" of the "world"). The set of all things under consideration is the conception of that domain."*
>
> [FRISCO98]

A procura no metamodelo do UML de um elemento que possa representar o conceito de "coisa" ("Thing") do FRISCO, leva-nos a focar a atenção na secção do metamodelo apresentado na Figura A.1:

[36] Os extractos de metamodelo apresentados usam (e em conformidade com o dito ao longo do ponto 4.5 deste documento) elementos e notação do UML (ex: classe, generalização/especialização, associação, multiplicidade, papéis, etc.)

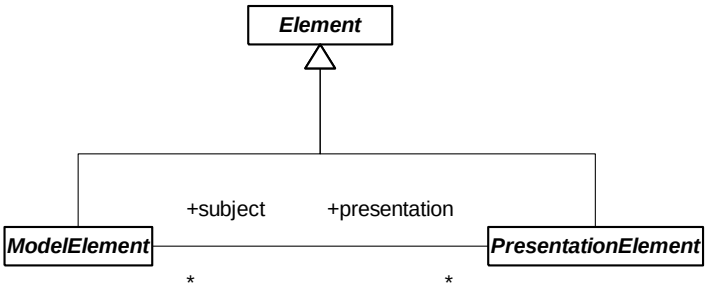

Figura A.1 – Extracto do metamodelo UML – As especializações de "Element"

"*Element*" é o elemento mais genérico que se encontra no metamodelo e a partir do qual todos os elementos são derivados. "*ModelElement*" e "*PresentationElement*" são especializações (conceito no contexto do paradigma de orientação a objectos) de "*Element*".

Também do documento de especificação, pode-se retirar as definições apresentadas para estes elementos:

> *Element (2.5.2.16)*
>
> *An element is an atomic constituent of a model.*
>
> *In the metamodel, an Element is the top metaclass in the metaclass hierarchy. It has two subclasses: ModelElement and PresentationElement. Element is an abstract metaclass.*
>
> [OMG-UML01, pág. 2-34]
>
> *ModelElement (2.5.2.27)*
>
> *A model element is an element that is an abstraction drawn from the system being modeled. Contrast with view element, which is an element whose purpose is to provide a presentation of information for human omprehension. In the metamodel, a ModelElement is a named entity in a Model. It is the base for all modeling metaclasses in the UML. All other modeling metaclasses are either direct or indirect subclasses of ModelElement.*
>
> [OMG-UML01, pág. 2-42]
>
> *PresentationElement (2.5.2.33)*
>
> *A presentation element is a textual or graphical presentation of one or more model elements.*
>
> *In the metamodel, a PresentationElement is an Element that presents a set of ModelElements to a reader. It is the base for all metaclasses used for presentation. All other metaclasses with this purpose are either direct or indirect subclasses of PresentationElement. PresentationElement is an abstract metaclass. The subclasses of this class are proper to a graphic editor tool and are not specified here. It is a stub for their future definition.*
>
> [OMG-UML01, pág. 2-48]

Pode-se observar no extracto de metamodelo do UML apresentado na Figura A.2 que é a partir de "*ModelElement"* que se encontram as primeiras especializações, enquanto que "*PresentationElement"* não existe qualquer especialização.

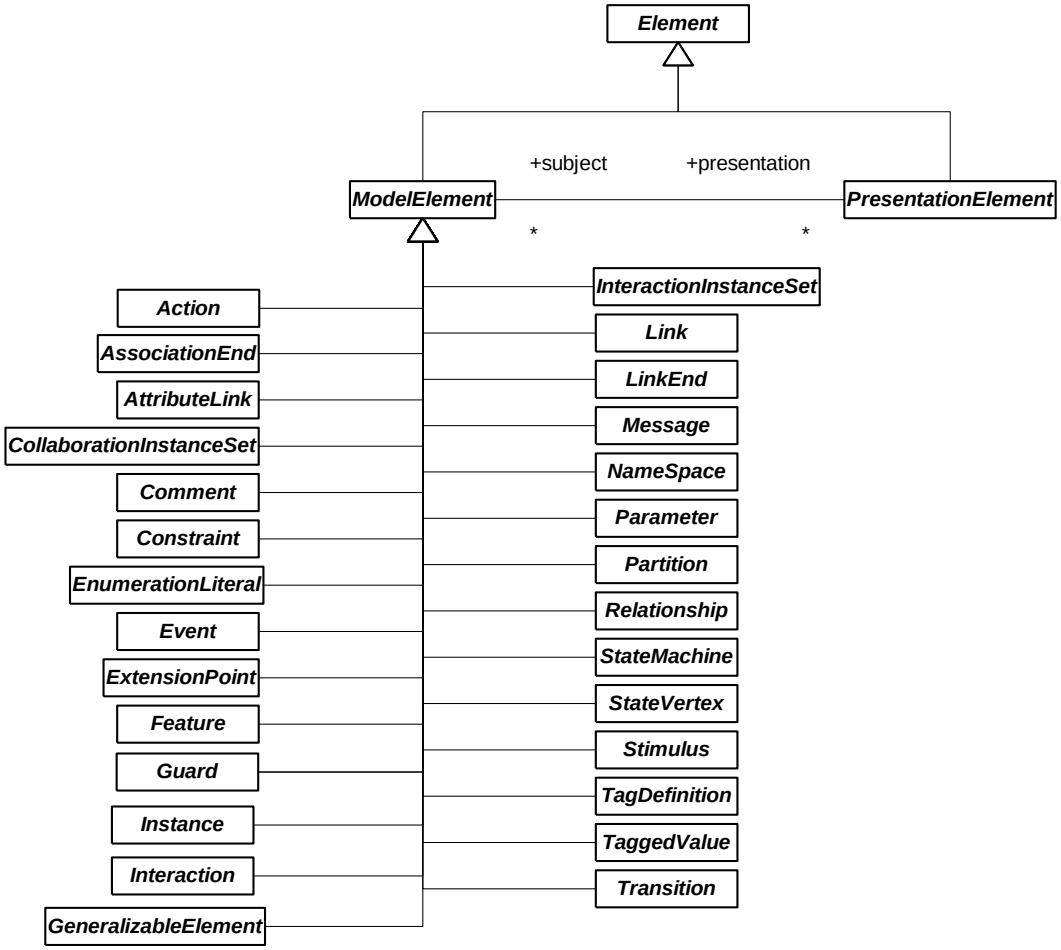

Figura A.2 – Extracto do metamodelo UML –As especilizações de "ModelElement"

Pode-se constatar que, através das especializações "*ModelElement"* e "*PresentationElement*" a partir de "*Element*", se separa a concepção da representação. Pode-se a partir daqui tirar duas conclusões:

1) O UML distingue entre concepção e representação; a concepção em si é contemplada pelo elemento de modelo "*ModelElement",* enquanto que a sua representação é feita por elemento de apresentação "*PresentationElement"*. No documento de especificação, o que se apresenta (relativo ao metamodelo) na sua parte de sintaxe e semântica corresponde aos elementos de modelo que compoêm a linguagem UML. Na secção de guia de notação do UML[37], propõe-se a representação visual para estes elementos da linguagem UML.

[37] "UML Notation Guide" [OMG-UML01, pág. 3-1]

2) *Coisa* ("*thing*") do FRISCO pode corresponder a "*ModelElement" do UML*. Porquê *"ModelElement"* e não *"Element"?* O "*Element"* permite no fundo fazer passar a noção de que existe uma diferença entre concepção e representação, indo assim ao encontro aos pressupostos em que assenta a visão ontológica do FRISCO (as pessoas usam representações para comunicar concepções). Mas que abstracções existem no mundo? Ou seja, que coisas existem no mundo (e para as quais arranjamos representações)? Uma coisa ("thing") definida no FRISCO como "*...any part of a conception of a domain*…", o que revela a preocupação com a concepção; no UML "*ModelElement"* é definido como "*... an element that is an abstraction drawn from the system being modeled."*; a consideração destas definições, conjuntamente com a constatação de que é a partir de "*ModelElement"* que o metamodelo do UML define as abstracções que se podem encontrar num domínio, leva a assumir como melhor correspondência a de "Thing" do FRISCO com "*ModelElement"* do UML.

Predicator, Predicated thing

> *A **predicator** is a thing, used to characterise or qualify other things, and assumed as being "atomic", "undividable" or "elementary". A **predicated thing** is a thing being characterised or qualified by at least one predicator.*
>
> [FRISCO98]

A intenção no FRISCO de definir "predicator" e "predicated thing" é expressa pela afirmação que precede estas definições:

> "*This definition* [thing] *is of course too general to be of any practical value. To be able to structure our conceptions in a meaningful way, we need to introduce a number of special things. Firstly, to distinguish the things we wish to characterize in some way, from the things that characterizes them, we introduce the special concept "predicator"".*

Pretende-se no FRISCO distinguir as "coisas" em si, daquilo que possa caracterizar essas coisas. Começa-se aqui a notar a diferença nos objectivos e fundamentos que constituem as visões ontológicas do FRISCO e do UML.

O FRISCO, através da especialização do conceito de "Thing", define "predicator" e "predicated thing"; no UML, a partir de "*ModelElement*", especializam-se e distinguem-se vários conceitos entre os quais se encontram classificadores, relacionamentos, acções, instâncias, ligações (links), mensagens, etc.. O UML, ao contrário do FRISCO, não tem a preocupação distinguir entre as coisas que caracterizam e as coisas que são caracterizadas; coisas caracterizadas e coisas que caracterizam estão a nível de granularidade que o UML não considera, o que se pode constatar no metamodelo do UML.

Coloca-se a questão: então, no UML, o que é que caracteriza as coisas? Relembra-se que o UML tem por base o paradigma da orientação a objectos. Neste paradigma, existe em interesse em ver todas as coisas como objectos, os quais possuindo uma estrutura e comportamento, são caracterizados atributos e por operações; os objectos podem ter ligações com outros objectos. Os objectos com estrutura e comportamento semelhantes são classificados como pertencendo a uma determinada classe, e as suas possíveis ligações com outros objectos são modelados como relacionamentos entre classes; o UML incorpora estes conceitos. Apresenta-se na Figura A.3 um extracto retirado do metamodelo do UML que permite trazer alguma luz sobre este assunto.

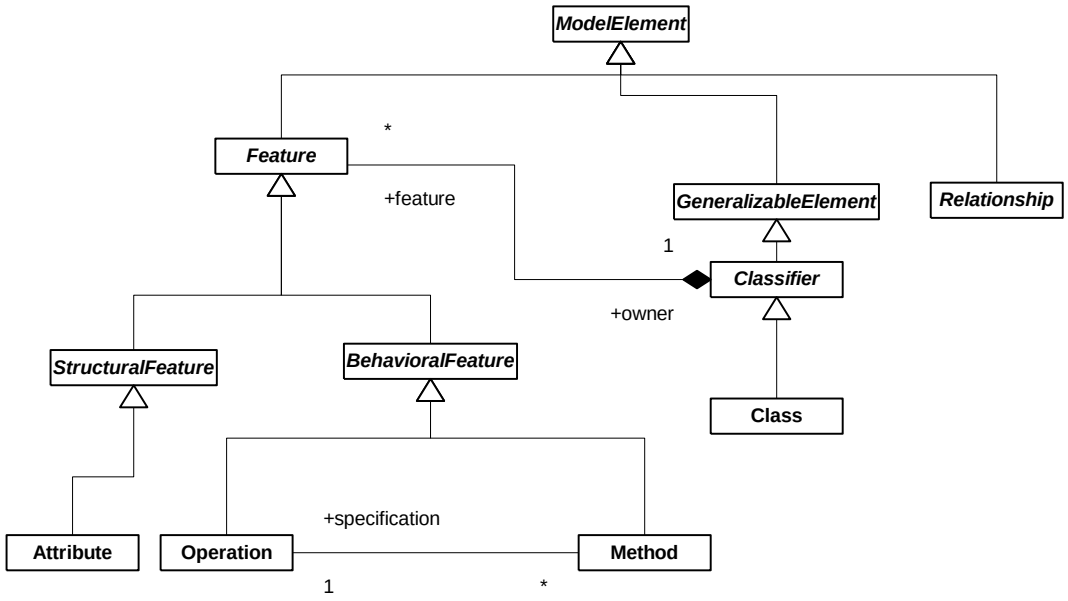

Figura A.3 – Extracto do metamodelo UML – elementos "Feature" e "Classifier"

No UML, "*Feature*" é o elemento que permite atribuir características (quer de estrutura, quer de comportamento) ao que é considerado no metamodelo de "*Classifier*" (Classificador). Um classificador é definido como:

> *"A classifier is an element that describes behavioral and structural features; it comes in several specific forms, including class, data type, interface, component, artifact, and others that are defined in other metamodel packages."*
>
> [OMG-UML01, pág. 2-28]

Através das especializações de "*Classifier*" (ver Figura A.4) é modelado aquilo que exibe características de estrutura e comportamento.

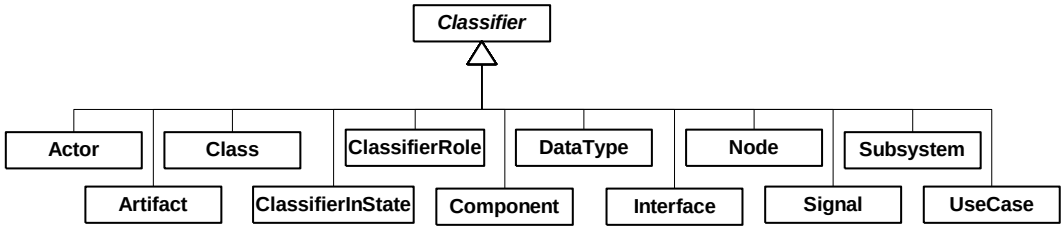

Figura A.4 – Extracto do metamodelo UML – Especializações de "Classifier"

Se atendermos a exemplos de "predicators" fornecidos no FRISCO ("*is a person*", "*is a book*", "*is the author of*" e "*is written by*"), pode-se colocar a questão: no UML, qual será o equivalente ou aproximado ao conceito do qual aqueles são exemplos? Uma resposta possível é que exemplos do género "*is a person*", "*is a book*" enquadrar-se-ão nos mecanismos de classificação ou de generalização/especialização da orientação a objectos, e, exemplos do "*is the author of*", "*is written by*" são modelados como relacionamentos (tipicamente associações) envolvendo elementos de modelo (mais propriamente classificadores).

Em suma, não se pode fazer adequadamente associar elementos do UML aos conceitos de "predicator" e de "predicated thing" presentes no FRISCO.

Relationship

*A **relationship** is a special thing composed of one or several predicated thing(s), each one associated with one predicator characterising the role of that predicated thing within that relationship.*

Examples:

- *A representation of a unary relationship:*

  {<predicated thing, predicator>}:

  Ludwig Wittgenstein is a person.

- *Two representations of one and the same binary relationship:*

  {<predicated thing 1, predicator 1>, <predicated thing 2, predicator 2>}:

  Ludwig Wittgenstein is the author of Philosophical Investigations.

  Philosophical Investigations is written by Ludwig Wittgenstein.

- *A representation of a ternary relationship:*

  A particular department orders a particular article from a particular supplier.

- *Since a relationship is also a thing, it may be characterised by predicators itself, i.e. may be related to other things:*

  A particular student attends a particular course (a binary relationship);

  A particular mark is granted for this attendance

  (a binary relationship between a thing and the first relationship).

[FRISCO98]

O UML define relacionamento ("*Relationship*") como uma conexão entre elementos de modelo:

> *Relationship(2.5.2.36)*
>
> *A relationship is a connection among model elements. In the metamodel, Relationship is a term of convenience without any specific semantics. It is abstract.*
>
> [OMG-UML01, pág. 2-49]

Pode-se constatar no metamodelo (ver Figura A.5) que, sendo "*Relationship"* definido como um elemento abstracto, possui várias especializações que podem ser utilizadas na modelação.

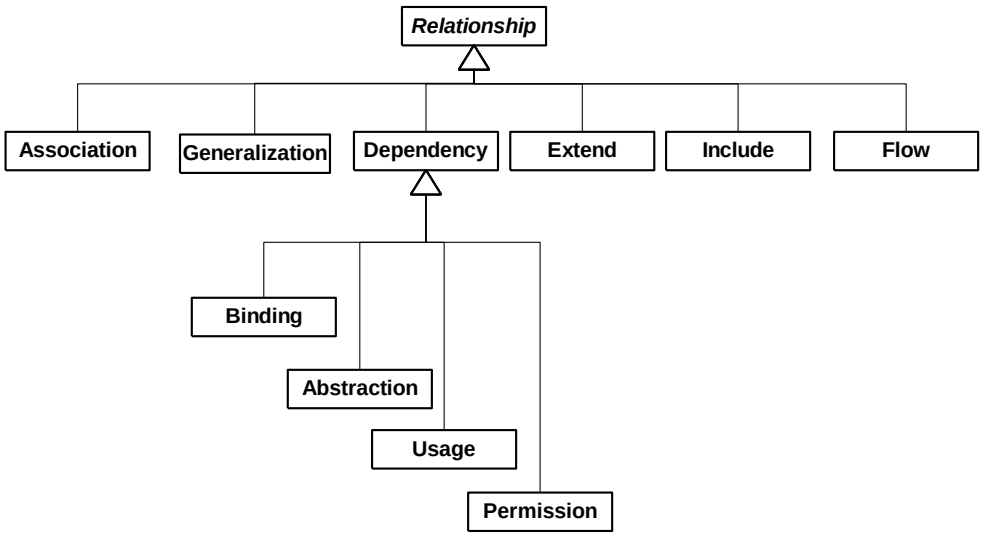

Figura A.5 – Extracto do metamodelo UML – "Relationship" e suas especializações

Faz-se notar que no metamodelo do UML, e por definição, o elemento associação ("*Association*") representa tipos de relacionamentos entre (e apenas) elementos do tipo classificador ("*Classifier*").

> *Association(2.5.2.3)*
>
> *An association defines a semantic relationship between classifiers. The instances of an association are a set of tuples relating instances of the classifiers. Each tuple value may appear at most once.*
>
> [OMG-UML01, pág. 2-19]

Outros relacionamentos envolvem, de uma forma mais geral, elementos do tipo "*ModelElement*". Por exemplo, poder-se-á constatar pelos extractos do metamodelo a seguir apresentados na Figura A.6 e na Figura A.7 que os relacionamentos "*Association"* e "*Dependency"* não envolvem o mesmo tipo de elementos.

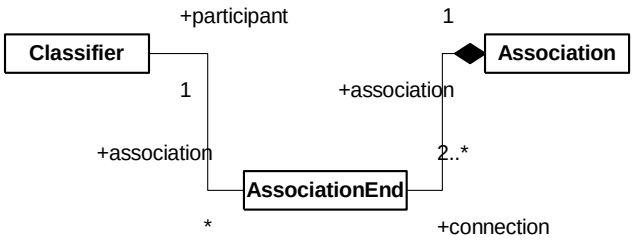

Figura A.6 – Extracto do metamodelo UML – "Association"

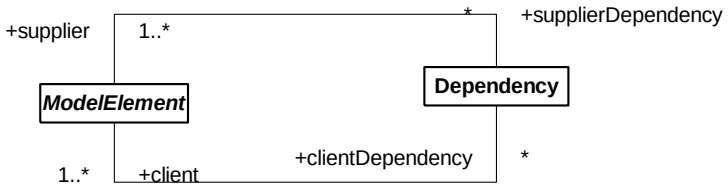

Figura A.7 – Extracto do metamodelo UML – "Dependency"

À partida, pelas definições apresentadas, poderíamos ser levados a dizer que "relacionamento" no FRISCO é adequadamente representado por "relacionamento" ("*Relationship*") no UML. No entanto, algures mais à frente do relatório do FRISCO, aquando de outra definição[38] (a de "Type"), encontra-se a seguinte afirmação:

> *We are building up our conceptual framework basically from an extensional point of view. Whenever we speak of a thing, a relationship, an entity, etc., we mean an "instance", not a "type".*

Ou seja, o "relacionamento" no FRISCO não se refere a género "tipo", mas sim a uma instância concreta. Tal inibe a possibilidade anterior, uma vez que no UML, um "relacionamento" procura estar em representação de tipos, para os quais ocorrerão instâncias.

Então o que é que se encontra no metamodelo do UML, a nível de instância, e ao qual se possa fazer corresponder a "relacionamento" do FRISCO? No que diz respeito a relacionamentos, e ao nível de significado de instância, existe apenas o elemento ligação ("*Link*").

O elemento "*Link*" do UML é definido como:

> *Link (2.9.2.12)*
>
> *The link construct is a connection between instances.*
>
> *In the metamodel Link is an instance of an Association. It has a set of LinkEnds that matches the set of AssociationEnds of the Association. A Link defines a connection between Instances.*
>
> [OMG-UML01, pág. 2-102]

[38] E mais precisamente, imediatamente antes da definição de "Type" no FRISCO.

Observa-se no metamodelo (ver Figura A.8) que "*Link*" é composto por dois ou mais "*LinkEnd*", os quais estão em representação das instâncias que participam na ligação.

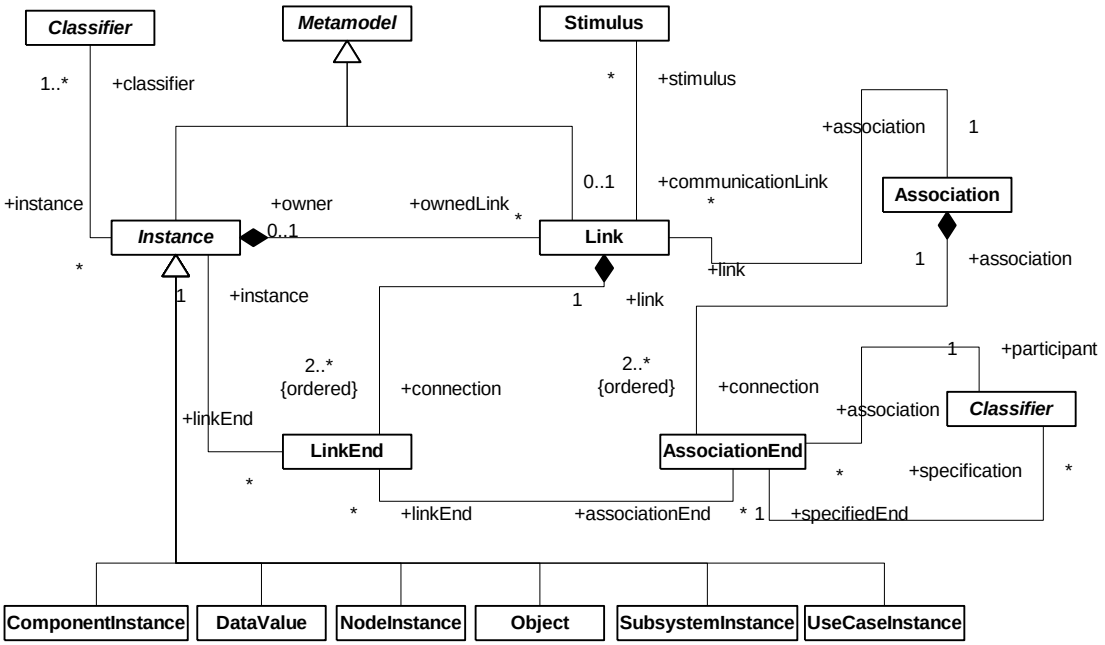

Figura A.8 – Extracto do metamodelo UML – "Link"

Constata-se que uma ligação ("*Link*") é considerada uma instância de uma associação[39], e, no metamodelo, nenhuma das outras especializações de "relacionamento" (generalização, dependência, etc.), encontram definidas no metamodelo as respectivas instâncias. Tal significa que o "relacionamento" ("Relationship") do FRISCO, como uma instância e não como um tipo, corresponderia na melhor das hipóteses, a "*Link*" do UML. Contudo, tal seria redutor, pois conforme se pode constatar pelos exemplos fornecidos com a definição de relacionamento no FRISCO ("*Ludwig Wittgenstein is a person", etc.*), o conceito de relacionamento do FRISCO abarca não só o que o UML considera como associação (e seu associado conceito de instância,"*Link*"), mas também o relacionamento subjacente ao conceito de generalização do UML, como também o implícito mecanismo da classificação, e para os quais o UML não define instância.

Consequentemente, em rigor, um "relacionamento" ("Relationship") não pode ser correctamente representado por um elemento (ou vários) do UML, podendo-se apenas dizer que no UML existe um elemento, ligação ("*Link*"), que poderá parcialmente representar o conceito de "Relationship" do FRISCO.

[39] Apesar de não estar definida no metamodelo do UML como especialização do elemento "instance". Tal não acontece porque o UML considera uma "instância" como sendo instância de um "classificador", e não de relacionamentos.

Em jeito de apontamento aparte, apenas mais uma observação relativa a relacionamentos no UML: uma transição em UML é tida como um relacionamento entre estados:

> *Transition (2.12.2.17)*
>
> *A transition is a directed relationship between a source state vertex and a target state vertex. It may be part of a compound transition, which takes the state machine from one state configuration to another, representing the complete response of the statemachine to a particular event instance.*
>
> [OMG-UML01, pág. 2-155]
>
> *transition A relationship between two states indicating that an object in the first state will perform certain specified actions and enter the second state when a specified event occurs and specified conditions are satisfied. On such a change of state, the transition is said to fire.*
>
> [OMG-UML01, pág. B-20]

No que diz respeito a um esquema de conceitos, tal leva a questionar se seria legítimo considerar uma transição como um relacionamento, e consequentemente de alguma forma, reflectir tal no metamodelo. Como complemento informativo, apresenta-se na Figura A.9 o extracto do metamodelo que ilustra o elemento "*Transition"* do UML*:*

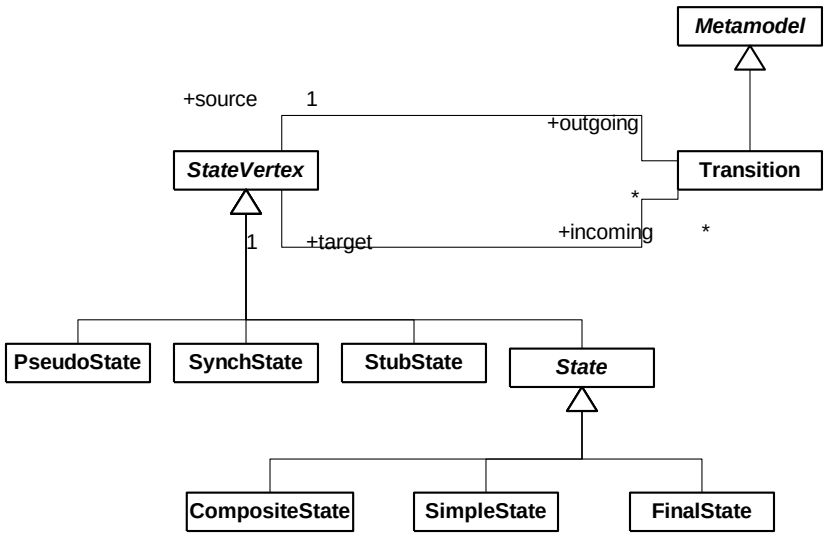

Figura A.9 – Extracto do metamodelo UML – "Transition"

Set membership, Elementary thing , Composite thing

> *A* ***set membership*** *is a special binary relationship between a thing (the set), characterized by the special predicator called 'has-element', and another thing, characterized by the special predicator called 'is-element-of'. An* ***elementary thing*** *is a thing, not being a relationship and not being characterised by the special predicator called 'has-element'. A* ***composite thing*** *is a thing, not being an elementary thing.*
>
> [FRISCO98]

Exemplos do FRISCO que ilustram estes conceitos:

- "*A convoy of ships contains a number of particular ships.*"
- *"A particular ship contains a number of ship components plus a number of relationships between these components indicating their spatial position."*

Em UML, a associação representa uma relacionamento semântico entre classificadores, e, conforme se pode constatar na especificação UML *"In UML, Associations can be of three different kinds: 1) ordinary association, 2) composite aggregate, and 3) shareable aggregate. (…)"* [OMG-UML01 pág 2-66], uma associação pode representar uma agregação, ou seja um relacionamento todo/parte.

Portanto, poder-se-á fazer corresponder o conceito "set membership" ao conceito agregação ou composição, os quais sendo variantes de uma associação, têm o objectivo de denotar um relacionamento todo/parte; a agregação permite que as suas partes possam pertencer a mais do que um todo, enquanto que numa composição as partes apenas podem pertencer a um todo.

O exemplo apresentado pela Figura A.10, ilustra uma composição. Poder-se-ia em raciocínio similar, esboçar uma estrutura semelhante para os exemplos do FRISCO anteriormente apresentados.

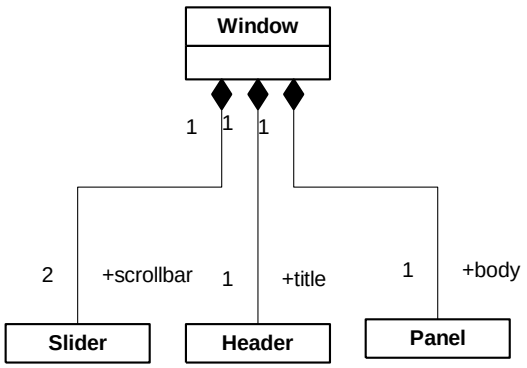

Figura A.10 – Exemplo de composição (retirado de [OMG-UML01])

No metamodelo do UML, não são definidas especializações da "associação" para representar os conceitos de "agregação" e de "composição"; em vez, estes últimos são definidos através de um valor atribuído a um atributo presente no elemento "*AssociationEnd*", mais especificamente o atributo "*aggregation*", para o qual os valores possíveis são: "*none*", "*agregate*", "*composite*".

Considerando a definição de "Elementary Thing" do FRISCO, uma coisa elementar será aquela que não é nem um relacionamento, nem possui um relacionamento de agregação ou de composição no qual desempenhe o papel de todo, e, uma coisa composta, será aquela que não for considerada elementar.

Deve-se então aqui também fazer notar que, embora não no metamodelo, existem referências a objecto composto ("*Composite Object*"), e mesmo a uma notação:

> *"A composite object represents a high-level object made of tightly-bound parts. This is an instance of a composite class, which implies the composition aggregation between the class and its parts."*
>
> *"A composite object symbol maps into an Object of the given Class with composition links to each of the Objects and Links corresponding to the class box symbols and to association path symbols directly contained within the boundary of the composite object symbol (and not contained within another deeper boundary)."*
>
> [OMG-UML01, pág. 3-67, 3-68]

Tal vai ao encontro com o anteriormente dito: uma coisa elementar será aquela não sendo relacionamento, não tem ligações ("links") de agregação ou de composição.

Portanto, o UML distingue coisas compostas de coisas elementares através da existência ou não existência de relacionamentos de agregação de composição.

Entity

> *An **entity** is a predicated thing as well as an elementary thing.*

O UML não define no seu metamodelo qualquer elemento designado por "entidade" ("Entity"). No entanto, pode-se constatar no documento do UML que o termo "entidade" é utilizado para expressar um sistema, um subsistema ou uma classe. Apresentam-se alguns excertos retirados do documento de especificação do UML:

> *"The use case construct is used to define the behavior of a system or other semantic entity without revealing the entity's internal structure."*
>
> [OMG-UML01 pág. 2-137]
>
> *"Actors model parties outside an entity, such as a system, a subsystem, or a class that interact with the entity".*
>
> [OMG-UML01 pág. 2-140]
>
> *"In the following text the term entity is used when referring to a system, a subsystem, ora class and the terms model element and element denote a subsystem or a class."*
>
> [OMG-UML01 pág. 2-141]

Coloquemos agora a questão: no UML, o que deveremos considerar como entidade? Pela definição do FRISCO, uma "coisa elementar" não é um relacionamento, e assim, uma

entidade nada terá a ver com relacionamentos. Adicionalmente, recorde-se que a definição é relativa a instâncias, e não a tipos[40].

Veja-se a definição de instância, o qual é um elemento presente no metamodelo do UML:

> *Instance (2.9.2.11)*
>
> *The instance construct defines an entity to which a set of operations can be applied and which has a state that stores the effects of the operations. In the metamodel Instance is connected to at least one Classifier that declares its structure and behavior.*
>
> [OMG-UML01 pág. 2-102]

No metamodelo do UML, "*Instance"* tem as especializações[41] ilustradas na Figura A.11.

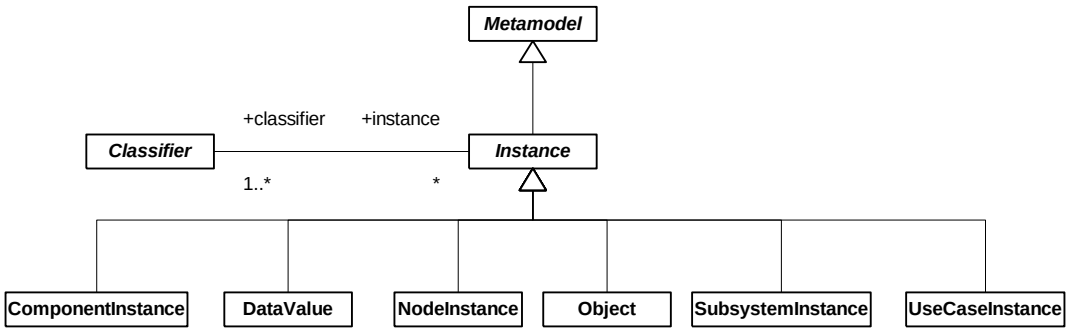

Figura A.11 – Extracto do metamodelo UML – "Instance"

Poderemos encontrar algum elemento no UML ao qual possamos fazer uma correspondência a "Entity" do FRISCO? A melhor aproximação será a "*Instance"*. No entanto esta correspondência poderá estar longe de ser pacífica; podem-se levantar algumas questões que suscitam dúvidas quanto à plenitude desta correspondência. Por exemplo, no FRISCO, uma entidade é definida como uma coisa elementar; ora uma instância possui um conjunto de atributos que ajudam a caracterizá-la, e no fundo, existirá um relacionamento de composição entre a instância e os seus atributos, o que indica que uma instância do UML, poderá ser vista à luz do FRISCO como uma coisa composta. A Figura A.12, apesar de ilustrar o exemplo para uma classe (cujas instâncias são denominadas de objectos), ajuda a ilustrar a questão:

[40] Conforme observação feita (aquando à análise do elemento "Relationship" do FRISCO na pág. 84), relativamente à construção do esquema de conceitos de um ponto de vista "extensional".

[41] Faz-se notar que no metamodelo, a instância de uma associação, "*Link*", não é definida como uma especialização do elemento do "*Instance*", mas sim como uma especialização de "*ModelElement*".

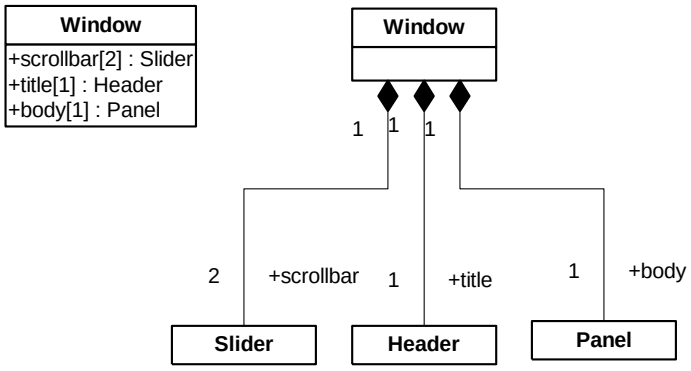

Figura A.12 – Exemplo de composição referente a atributos (retirado de [OMG-UML01])

A resposta a esta questão poderá ser encontrada no próprio FRISCO, nas notas que se seguem à definição da própria "Entity":

> "*Notes:*
>
> • *The reason to define an entity as a predicated thing is to distinguish it from a predicator.*
>
> • *Whether or not some predicated thing is considered to be elementary (i.e. an entity) is dependent on the context in which that predicated thing is used. That is, the notion of elementariness depends on the level of "granularity" chosen in a particular situation. Whenever this situation changes, the elementariness or non-element-ariness of a predicated thing may change as well."*
>
> [FRISCO98, pág 36]

Verifica-se então que a noção de elementar depende da granularidade escolhida para uma situação em particular. Podemos concluir que o UML não tem definido no seu metamodelo um elemento denominado por "entidade", apesar de utilizar este termo com frequência ao longo do seu documento. Uma análise ao metamodelo revela que o elemento "*Instance*" é aquele que mais se aproxima em correspondência à definição de entidade no FRISCO.

Type, Population, Instance

> *A **type** of things is a specific characterisation (e.g. a predicate) applying to all things of that type. A **population** of a type of things is a set of things, each one fulfilling the characterisation determining that type. An **instance** of a type of things is an element of a population of that type.*
>
> [FRISCO98]

No documento de especificação do UML é feita uma referência à dicotomia tipo-instância:

> *A major purpose of modeling is to prepare generic descriptions that describe many specific items. This is often known as the type-instance dichotomy. Many or most of the modeling concepts in UML have this dual character, usually modeled by two paired modeling elements, one represents the generic*

descriptor and the other the individual items that it describes. Examples of such pairs in UML include: Class-Object, Association-Link, Parameter-Value, Operation-Invocation, and so on.

[OMG-UML01 pág. 3-14]

No metamodelo, encontramos as definições de classes e objectos, de associações e ligações, e de outros. Em particular, pode-se retirar do metamodelo o extracto apresentado na Figura A.13:

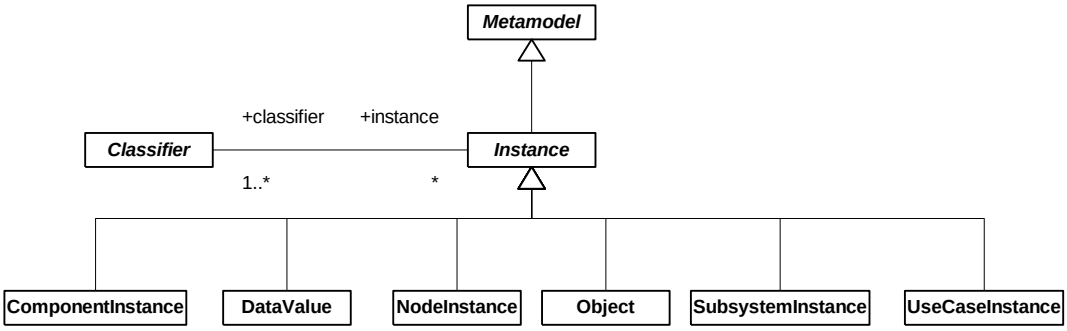

Figura A.13 – Extracto do metamodelo UML – "Instance"

Conforme ilustrado na Figura A.14, o elemento classificador ("*Classifier*") possui várias especializações:

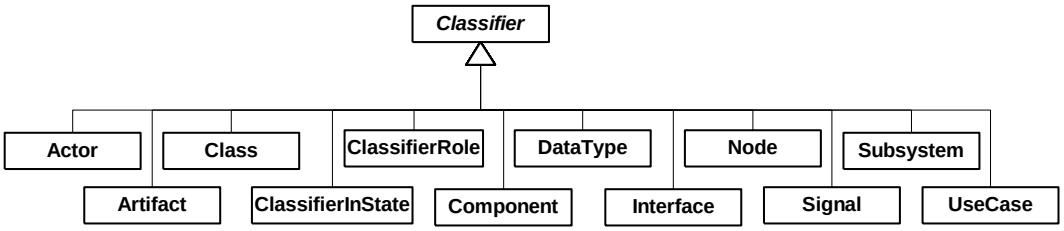

Figura A.14 – Extracto do metamodelo UML – "Classifier"

Será que se poderia dizer que o tipo ("Type") do FRISCO corresponde ao elemento classificador ("*Classifier")* do UML? Antes de afirmarmos que sim, é conveniente ter em atenção mais algumas afirmações presentes no FRISCO, as quais que indicam que devemos ter mais prudência na resposta.

Na definição de classe ("Class") no UML, pode-se encontrar:

Class (2.5.2.9)

A class is a description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. A class may use a set of interfaces to specify collections of operations it provides to its environment.

(…)

Stereotypes

> ***«type»*** *Specifies a domain of objects together with the operations applicable to the objects, without defining the physical implementation of those objects. A type may not contain any methods, maintain its own thread of control, or be nested. However, it may have attributes and associations. The associations of a Type are defined solely for the purpose of specifying the behavior of the type's operations and do not represent the implementation of state data.*
>
> [OMG-UML01 pág. 2-26, 2-27]
>
> *A class is the descriptor for a set of objects with similar structure, behavior, and relationships. The model is concerned with describing the intension of the class, that is, the rules that define it. The run-time execution provides its extension, that is, its instances. (…) A class represents a concept within the system being modeled. Classes have data structure and behavior and relationships to other elements.*
>
> [OMG-UML01 pág, 3-35]
>
> *"Classes can be stereotyped as Types or Implementation Classes (although they can be left undifferentiated as well). A Type is used to specify a domain of objects together with operations applicable to the objects without defining the physical implementation of those objects. (…) An Implementation Class defines the physical data structure (for attributes and associations) and methods of an object as implemented in traditional languages (C++, Smalltalk, etc.). An Implementation Class is said to realize a Type if it provides all of the operations defined for the Type with the same behavior as specified for the Type's operations. An Implementation Class may realize a number of different Types."*
>
> [OMG-UML01 pág. 3-49]

É dado a entender que as classes providenciam a forma de definir "tipos", os quais são usados então para especificar domínio de objectos. Poder-se-ia então talvez considerar que, à luz da afirmação anterior, o "tipo" ("Type") do FRISCO corresponderá a classe ("*Class")* do UML?

A considerar uma das duas opções, opta-se por se considerar mais adequada a correspondência de "Type" do FRISCO para "*Classifier"* do UML, devido a ser clara e explícita a associação entre "*Classifier"* e "*Instance"* (o que se poderá constatar no extracto do metamodelo ilustrado na Figura A.15), e, o elemento "*Classifier*" ser mais abrangente, pois para além de classe (*"Class"*) e a sua instância objecto ("*Object"*), existem outros tipos (como por exemplo subsistema ("*Subsystem*") e a sua instância, *SubsystemInstance*). Considera-se portanto a correspondência de "Type" para "*Class*" como uma forma particular da mais geral de "Type" para "*Classifier*".

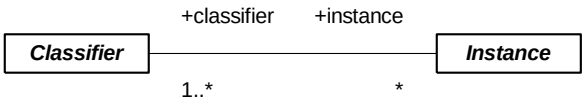

Figura A.15 – Extracto do metamodelo – "Classifier" e "Instance"

Classifier (2.5.2.10)

A classifier is an element that describes behavioral and structural features; it comes in several specific forms, including class, data type, interface, component, artifact, and others that are defined in other metamodel packages.

[OMG-UML01 pág. 2-28]

A component instance is an instance of a component that resides on a node instance.

[OMG-UML01 pág 2-100]

A node instance is an instance of a node.

[OMG-UML01 pág. 2-104]

No entanto, é importante referir que existe algo que poderá invalidar a esta opção, e que se pode expressar pela seguinte questão: será válido (por consequência de se considerar "*Classifier*" do UML adequado ao conceito "Type" do FRISCO), afirmar que "Interface" (uma especialização de "*Classifier*") é representante de um "tipo" ? Não, não é válido; se fosse, qual seria a sua possível "população"? Poderá um classificador do género "*Interface*" ter instâncias? Não, e o UML esclarece isso, na secção de "regras de boa formação"[42], ao explicitar:

AssociationEnd (2.5.3.3)

(…) The Classifier of an AssociationEnd cannot be an Interface or a DataType if the association is navigable away from that end.

[OMG-UML01 pág. p-2-53]

Conclui-se que não é perfeita a correspondência assumida de "Type" para "Classifier", sendo contudo, aquela que melhor se pode estabelecer.

No respeita a instâncias, a que corresponderá no UML o conceito de instância ("Instance") do FRISCO?

Pelo que foi dito (não só sobre "tipo", mas também por tudo o anteriormente dito), a correspondência mais propícia de "Instance" do FRISCO será para o elemento "*Instance"* do metamodelo do UML. No entanto, pode-se, relativamente ao UML, levantar uma questão com alguma pertinência: sendo no UML definido um "*Link"* como sendo uma instância de

[42] Secção "Well-formedness rules".

uma associação *("Association")*, porque é que "*Link*" não está como uma especialização de "*Instance'*?

Com a intenção de ilustrar esta questão, do metamodelo retira-se o extracto apresentado pela Figura A.16:

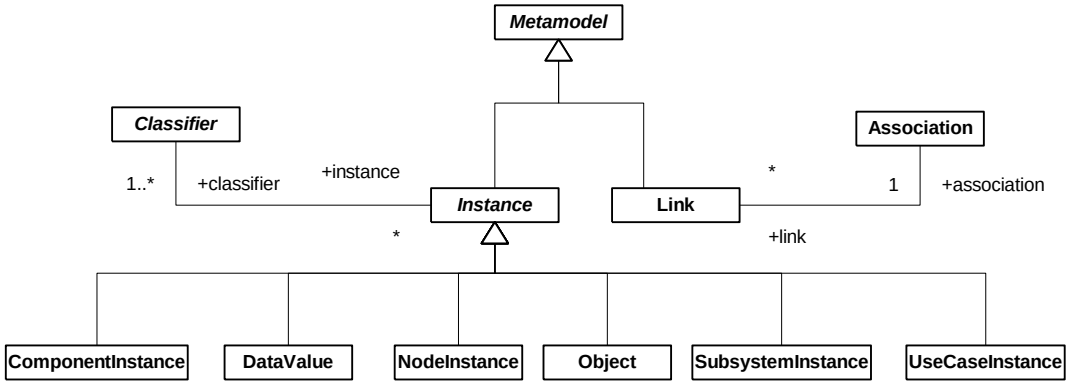

Figura A.16 – Extracto do metamodelo – "Instance" e "Link"

Relembra-se a definição do UML relativa a "*Link*"[43]:

> *Link (2.9.2.12)*
>
> *The link construct is a connection between instances.*
>
> *In the metamodel Link is an instance of an Association. It has a set of LinkEnds that matches the set of AssociationEnds of the Association. A Link defines a connection between Instances.*
>
> [OMG-UML01, pág. 2-102]

Certamente o que em geral prevaleceu na elaboração da especificação do UML, foi a ênfase na sintaxe e na correcção gramatical, em eventual (e pontual) detrimento de alguma coerência na definição de conceitos e terminologia adequada. No caso concreto, o facto do elemento "*Instance"* estar associado a um classificador, e, o "*Link*" ser relativo a uma associação (a qual é um relacionamento), deverá estar na origem do facto do elemento "*Instance*" do UML não incluir instâncias de relacionamentos (em "*Link*" é o exemplo). Poder-se-á então dizer que instâncias de classificadores são, em termos de tratamento, significativamente diferentes de instâncias de relacionamentos.

Fazendo notar a observação anterior (relativa a "L*ink*" como instância de "*Association*"), assume-se a correspondência do conceito "Instance" do FRISCO para "*Instance"* do UML.

O UML não define no seu metamodelo (nem no seu documento) elemento adequado ao conceito "Population" do FRISCO. Para se referir à colecção de instâncias de um determinado tipo, o UML usa na sua especificação o termo "conjunto de" ("*set of*"), conforme se ver em algumas das afirmações a seguir apresentadas.

[43] Já apresentada neste documento, aquando a análise do conceito "Relationship" do FRISCO (análise feita na pág. 84 e seguintes).

An association defines a semantic relationship between classifiers. The instances of an association are a set of tuples relating instances of the classifiers.

[OMG-UML01, pág 2-19]

The Association represents a set of connections among instances of the Classifiers.

[OMG-UML01, pág. 2-20]

A class is a description of a set of objects that share the same attributes, operations, methods, relationships, and semantics.

[OMG-UML01, pág. 2-26]

Transition

A transition is a special binary relationship between two (partially or totally) different composite things, called the pre-state and the post-state of that transition, whereby at least one thing is element of the pre-state, but not of the post-state, or vice versa.

[FRISCO98]

Como ajuda na contextualização da definição de transição, consideremos novamente o extracto do metamodelo UML apresentado na Figura A.17:

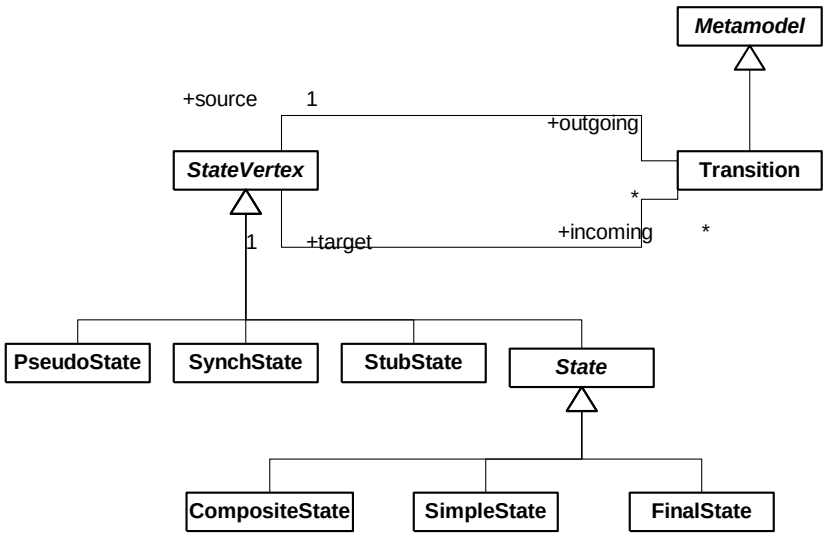

Figura A.17- Extracto do metamodelo - "Transition"

Notando a definição do elemento "transição" ("*Transition*") feita pelo UML,

Transition (2.12.2.17)

A transition is a directed relationship between a source state vertex and a target state vertex. It may be part of a compound transition, which takes the state machine from one state configuration to another, representing the complete response of the state machine to a particular event instance.

[OMG-UML01, pág. 2-155]

chega-se à conclusão que, relativamente ao conceito de transição, o UML está em consonância com o FRISCO, na medida em que também a define como um relacionamento entre dois estados. Assim, assume-se a correspondência entre transição ("Transition") do FRISCO e a transição do (*"Transition")* do UML.

Faz-se notar que, embora não sendo expresso no metamodelo, a especificação do UML caracteriza, para além das simples transições, outros tipos de transições:

- transições de alto nível ou de grupo
- transições compostas
- transições internas
- transições de acabamento

Estas transições encontram-se definidas na especificação UML pelas afirmações que a seguir se apresentam:

Transitions (2.12.4.6)

High-level transitions

Transitions originating from the boundary of composite states are called high-level or group transitions. If triggered, they result in exiting of all the substates of the composite state executing their exit actions starting with the innermost states in the active state configuration. (…)

Compound transitions

A compound transition is a derived semantic concept, represents a "semantically complete" path made of one or more transitions, originating from a set of states (as opposed to pseudo-state) and targeting a set of states. (…)

A (simple) transition connecting two states is therefore a special common case of a compound transition. (…)

Internal transitions

An internal transition executes without exiting or re-entering the state in which it is defined. This is true even if the state machine is in a nested state within this state. (…)

Completion transitions and completion events

A completion transition is a transition without an explicit trigger, although it may have a guard defined. When all transition and entry actions and activities in the currently active state are completed, a completion event instance is generated. This event is the implicit trigger for a completion transition.

[OMG-UML01 2-164, 2-165]

State

A state is a composite thing, involved as pre-state or as post-state in some transition. No element of a state may be a transition, itself.

[FRISCO98]

Tendo em conta o extracto do metamodelo UML respeitante a estados representado na Figura A.18,

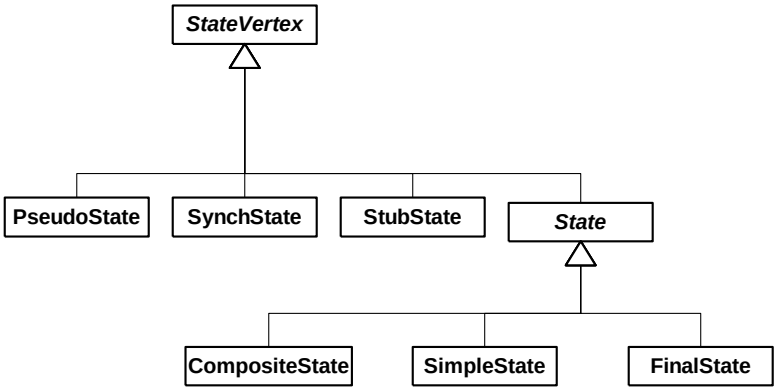

Figura A.18- Extracto do metamodelo - "State"

o que é que se deve considerar como elemento UML adequado ao conceito de estado ("State") do FRISCO? Veja-se primeiro as definições de "*StateVertex*", "*PseudoState*", "*SyncState*", "*StubState*", e teça-se de seguida breves considerações sobre estes estados, as quais permitirão colocá-los de parte na procura do elemento mais adequado à definição de estado no FRISCO.

StateVertex (2.12.2.12)

A StateVertex is an abstraction of a node in a statechart graph. In general, it can be the source or destination of any number of transitions.

[OMG-UML01, pág. 2-154]

PseudoState (2.12.2.7)

A pseudostate is an abstraction that encompasses different types of transient vertices in the state machine graph. They are used, typically, to connect multiple transitions into more complex state transitions paths. For example, by combining a transition entering a fork pseudostate with a set of transitions exiting the fork pseudostate, we get a compound transition that leads to a set of concurrent target states.

[OMG-UML01, pág. 2-151]

StubState (2.12.2.13)

A stub state can appear within a submachine state and represents an actual subvertex contained within the referenced state machine. It can serve as a source or destination of transitions that connect a state vertex in the containing state machine with a subvertex in the referenced state machine.

[OMG-UML01, pág. 2-154]

SynchState (2.12.2.15)

A synch state is a vertex used for synchronizing the concurrent regions of a state machine. It is different from a state in the sense that it is not mapped to a Boolean value (active, not active), but an integer. A synch state is used in conjunction with forks and joins to insure that one region leaves a particular state or states before another region can enter a particular state or states.

[OMG-UML01, pág. 2-155]

Debrucemo-nos sobre estes elementos introduzidos pelas expressões anteriores:

- *"StateVertex"*, representa genericamente um vértice de um grafo de um diagrama de estados; pelo facto de lhe serem definidas várias especializações - entre as quais se encontram algumas incompatíveis com a definição de estado do FRISCO[44], torna-se uma definição muito vaga e abrangente, e consequentemente, não adequada.
    - Conforme se poderá observar na especificação, "*PseudoState"* (para os quais são definidos vários tipos: "*initial pseudostate*" "*deepHistory*", "*shallowHistory*", "*join vertices*", "*fork vertices*", "*junction vertices*", "*choice vertices*") existe por conveniência de modelação, não indo propriamente ao encontro da definição que se procura como adequada à definição de estado no FRISCO.
- Pela definição, um "*StubState"* é um estado que está em representação de um outro estado pertencente a uma outra máquina de estados, o que o põe de parte ao nosso interesse.
- *"SynchState"* é definido a nível de modelação com o interesse de sincronizar regiões concorrentes de uma máquina de estados. Portanto, também não é o que se procura.

Poderá então no UML ser "*State"* o elemento mais adequado à definição de estado do FRISCO? Veremos de seguida motivos que justificam uma resposta negativa. Tenha-se novamente em conta (ver Figura A.19), a parte do metamodelo que define no UML o elemento o "*State"* e as suas especializações:

[44]O que se constatará nas considerações que se seguem.

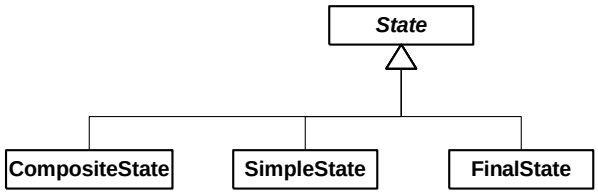

Figura A.19- Extracto do metamodelo - ""State" e suas especializações

Apresentam-se igualmente as afirmações do documento de especificação que definem "*State"*, "*CompositeState"*:

> *State (2.12.2.10)*
>
> *A state is an abstract metaclass that models a situation during which some (usually implicit) invariant condition holds. The invariant may represent a static situation such as an object waiting for some external event to occur. However, it can also model dynamic conditions such as the process of performing some activity; that is, the model element under consideration enters the state when the activity commences and leaves it as soon as the activity is completed.*
>
> [OMG-UML01, pág. 2-152]
>
> *CompositeState (2.12.2.3)*
>
> *A composite state is a state that contains other state vertices (states, pseudostates, etc.).*
>
> [OMG-UML01, pág. 2-149]

Relembrando a definição do FRISCO relativa a "estado", em especial a última parte,

> *A state is a composite thing, involved as pre-state or as post-state in some transition. No element of a state may be a transition, itself.*

nenhum elemento de um estado pode ser uma transição; ora, isto coloca de parte a possibilidade de se considerar o elemento "*CompositeState"* do UML como adequado a estado do FRISCO, e consequentemente, pela definição do metamodelo, o próprio elemento do metamodelo "*State"*; assim se justifica porque não se pode simplesmente dizer que "State" no FRISCO é adequadamente representado por "*State*" no UML. Consequentemente, relativamente ao conceito de "estado", restam dois elementos no metamodelo do UML como opções: "*SimpleState"* e "*FinalState"*.

> *SimpleState (2.12.2.9)*
>
> *A SimpleState is a state that does not have substates. It is a child of State.*
>
> [OMG-UML01, pág. 2-152]
>
> *FinalState (2.12.2.5)*

> *A special kind of state signifying that the enclosing composite state is completed. If the enclosing state is the top state, then it means that the entire state machine has completed. A final state cannot have any outgoing transitions.*
>
> [OMG-UML01, pág. 2-150]

Será consensual concluir que "*FinalState"* não corresponderá à definição do FRISCO; portanto é o elemento "*SimpleState"* aquele que por definição se aproximará à definição de "estado"[45] do FRISCO. Certamente que se poderia ficar por aqui no que diz respeito a esta adequação, mas existe algo mais a dizer:

- no UML, um estado, incluindo o "*SimpleState",* pode conter o que se designa (no UML) por "transições internas". Estas ditas transições internas, não implicam qualquer mudança de estado; "transições internas" é o nome dado a eventos que podem ocorrer enquanto se está num estado, eventos esses que, sendo tratados dentro desse estado, não despoletam transições de estado. Portanto, à luz do conceito de transição do FRISCO, estas "transições internas" não são consideradas como "transições" no FRISCO.
- Tendo em conta o esquema de conceitos do FRISCO, existe implicitamente no conceito de estado do UML, três momentos em particular poderão despoletar alguma celeuma quanto a deverem ser ou não considerados estados. No UML, existe a possibilidade [OMG-UML01, pág. 3-139] de se definir (1) uma acção a ser executada quando se entra num estado, sendo tal entrada denotada em UML com a etiqueta de acção "*entry*", (2) uma acção a ser executada após a entrada e que se desenrolará até à saída do estado, sendo tal circunstância denotada em UML com a etiqueta de acção "*do*", e, (3) uma acção a ser executada quando se sai do estado, sendo denotada em UML com a etiqueta de acção "exit". A questão que se levanta é: não poderão estes "momentos" serem considerados como estados, e a passagem de um destes momentos para outro momento, como uma transição? Fica em aberto a discussão, não se prosseguindo esta questão neste documento, no qual apenas se apresenta de seguida duas figuras: a Figura A.20, extraída da especificação do UML, que procura ilustrar esta questão (embora nela não esteja representada o equivalente ao momento 'do', mas inclui as chamadas "transições internas"), e a Figura A.21, que apresenta o extracto do metamodelo do UML que explicitamente contempla as acções relativas a estes momentos.

[45] Obs: Sugere-se a consulta das considerações tecidas ao conceito de "estado" aquando da análise efectuada ao conceito de "Actor" do FRISCO, mais à frente neste documento (pág. 113).

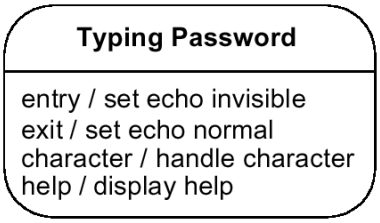

Figura A.20 – Exemplo de estado extraído de [OMG-UML01, pág. 3-140])

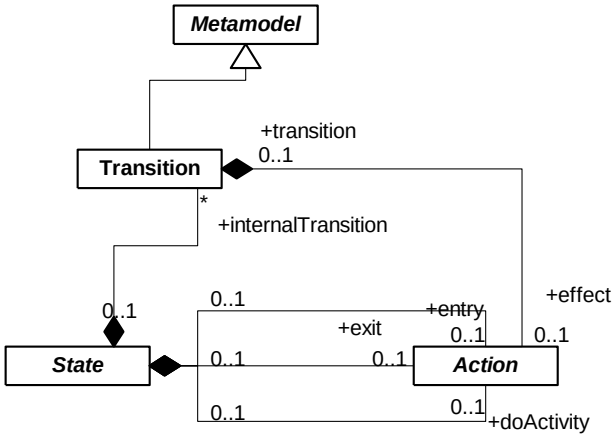

Figura A.21 – Extracto do metamodelo UML que ilustra os eventos "entry", "do", e "exit" que ocorrem num estado

Pre-state, Post-state

> *The pre-state of a transition is the state valid before that transition, and is characterised by the special predicator 'before'.*
>
> *The post-state of a transition is the state valid after that transition, and is characterised by the special predicator 'after'.*
>
> [FRISCO98]

No UML, uma transição dá-se de um estado designado por estado "fonte" ("*source*"), para um designado por estado "alvo" ou "destino" ("*target*"), o que é corroborado pela afirmação e pelo extracto de modelo (ver Figura A.22) que se encontram no documento de especificação do UML:

> *Every transition, except for internal transitions, causes exiting of a source state, and entering of the target state. These two states, which may be composite, are designated as the main source and the main target of a transition.*
>
> [OMG-UML01 2-166]

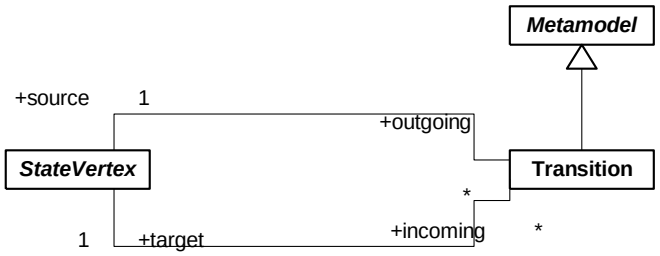

Figura A.22 – – Extracto do metamodelo – "Transition"

State-transition structure, Composite transition, Transition occurrence

Given are the transitions t_X: s1 => s2 and t_Y: s3 => s4. The following basic ***state-transition structures*** *exist in this case:*

(1) Sequence:

sequ (t_X, t_Y) is a sequence of transitions if s3 is a subset of s2.

The resulting state-transition structure has s1 as pre-state and s4 as post-state.

Longer sequences are defined as follows:

sequ (t_X, t_Y, t_Z) follows from sequ (t_X, t_Y) and sequ (t_Y, t_Z)

(2) Choice:

choice (t_X, t_Y) is a choice of transitions if the intersection of s1 and s3 is not empty.

The result is either transition t_X or t_Y, but not both.

(3) Concurrency:

concur (t_X, t_Y) are concurrent transitions if the intersection of s1 and s3 is empty.

The result is (s1 union s3) => (s2 union s4).

A ***composite transition*** *is a state-transition structure*[46] *with a unique pre-state and a unique post-state*

A ***transition occurrence*** *is a specific occurrence of a transition. A set of transition occurrences is subject to strict partial ordering.*

[FRISCO98]

Do metamodelo UML pode-se extrair a parte que diz respeito a estados compostos ("*CompositeState*") (ver Figura A.23). Constata-se que o UML permite que um estado (estado composto) possa conter outros estados (simples ou compostos) com as suas respectivas transições.

[46] O FRISCO identifica como sinónimo o termo "Processo". [FRISCO98, pág. 38]

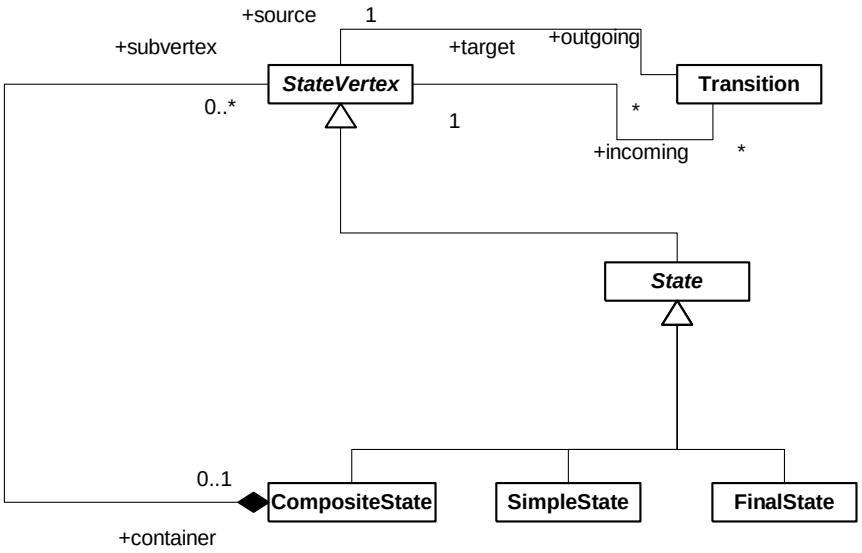

Figura A.23 – Extracto do metamodelo – "Composite State"

O UML define o conceito de estado composto e de subestado:

> *CompositeState (2.12.2.3)*
>
> *A composite state is a state that contains other state vertices (states, pseudostates, etc.). The association between the composite and the contained vertices is a composition association. Hence, a state vertex can be a part of at most one composite state. Any state enclosed within a composite state is called a substate of that composite state. It is called a direct substate when it is not contained by any other state; otherwise, it is referred to as a transitively nested substate.*
>
> [OMG-UML01, pág. 2-149]

Mais ainda, pode-se observar que o UML caracteriza um estado composto como podendo ser constituído por estados concorrentes ou por estados não concorrentes:

> *Composite States (3.76)*
>
> *Semantics (3.76.1)*
>
> *A composite state is decomposed into two or more concurrent substates (called regions) or into mutually exclusive disjoint substates. A given state may only be refined in one of these two ways. Naturally, any substate of a composite state can also be a composite state of either type.*
>
> [OMG-UML01, pág. 3-141]

Apenas com a intenção de facilitar a compreensão de estados compostos no UML, apresenta-se a Figura A.25 e a Figura A.26.

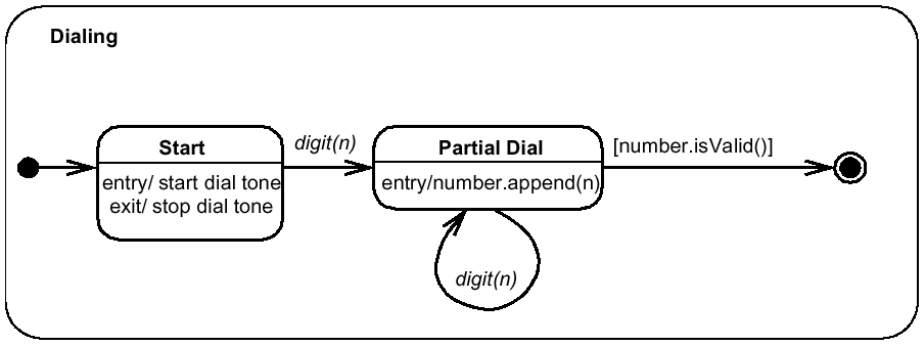

Figura A.24 – Exemplo de notação para estados sequenciais (extraído de [OMG-UML01, pág. 141])

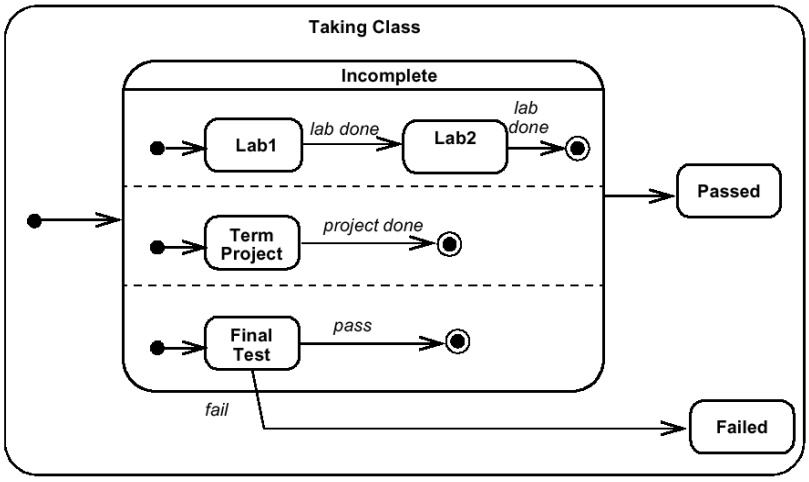

Figura A.25 – exemplo de notação para estados concorrentes (extraído de [OMG-UML01, pág. 141])

O UML também permite que a uma transição possa ser atribuída uma condição, denotada no metamodelo pelo elemento ("*Guard*"), a qual tem necessariamente ser válida para que a transição possa ocorrer:

> *Guard (2.12.2.6)*
>
> *A guard is a boolean expression that is attached to a transition as a fine-grained control over its firing. The guard is evaluated when an event instance is dispatched by the state machine. If the guard is true at that time, the transition is enabled, otherwise, it is disabled.*
>
> [OMG-UML01, pág. 2-150]

Através destas "guardas" (as quais para um dado "estado fonte" têm de ser mutuamente exclusivas) torna-se possível definir opções nas transições que podem ocorrer.

Pode-se assim dizer que o UML oferece as estruturas necessárias para a representação das "state-transitions structures" do FRISCO. Mais ainda, deve-se notar a riqueza do UML neste capítulo, uma vez que oferece mais elementos úteis para a modelação. Assume-se como válida a correspondência de "state transition strucutre" do FRISCO para "*Composite State*" do UML. De igual forma, atendendo à definição de "Composite transition" do FRISCO, também se considera a correspondência com "*Composite State"* do UML.

Contrariamente ao FRISCO, o UML no seu metamodelo não faz distinção entre transição e instância/ocorrência de transição, sendo, no documento de especificação do UML, a única referência relativa a instância de transição[47]:

> *A transition instance (such as a Stimulus or Message in a sequence diagram, a collaboration diagram, or a Transition in a state machine) may be given a name.*
>
> [OMG-UML01, pág. 3-114]

No entanto esta referência não pode ser levada à letra, na medida que, na definição de estímulo ("*Stimulus"*) ou de mensagem ("*Message"*), não aparece nada que aponte estes como instância de transição:

> *Stimulus (2.9.2.21)*
>
> *A stimulus reifies a communication between two instances.*
>
> [OMG-UML01, pág. 2-106]
>
> *Message (2.10.2.8)*
>
> *A message defines a particular communication between instances that is specified in an*
>
> *Interaction*
>
> [OMG-UML01, pág. 2-124].

No documento do FRISCO, aquando da definição de "transition occurrence", aparece uma nota a indicar que, por vezes, o termo "evento" é utilizado na literatura como sinónimo. Poderá ser este o caso?

A Figura A.26 permite observar que no metamodelo se pode encontrar um relacionamento entre transição e evento.

[47] Referência apresentada no capítulo relativo à notação da linguagem "UML Notation Guide".

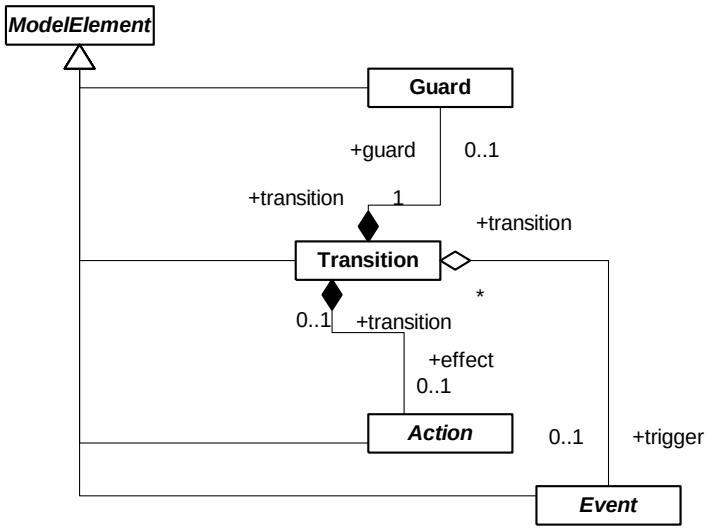

Figura A.26 – Extracto do metamodelo UML – "Event"

A definição de "evento" no UML é:

> *Event (2.12.2.4)*
>
> *An event is a specification of a type of observable occurrence. The occurrence that generates an event instance is assumed to take place at an instant in time with no duration.*
>
> *Strictly speaking, the term "event" is used to refer to the type and not to an instance of the type. However, on occasion, where the meaning is clear from the context, the term is also used to refer to an event instance.*
>
> [OMG-UML01, pág. 2-150]

Constatando que na definição anterior de evento, nada é mencionado relativamente a transição, faz-se notar que a ocorrência de uma transição é despoletada por um evento (associado a essa transição); tal ocorrência apenas acontecerá se o estado actual for o estado fonte indicado para essa transição e se for satisfeita eventual condição de guarda associada a essa transição.

> *A simple transition is a relationship between two states indicating that an instance in the first state will enter the second state and perform specific actions when a specified event occurs provided that certain specified conditions are satisfied. On such a change of state, the transition is said to "fire."*
>
> [OMG-UML01 pág. 3-146]

Portanto, a ocorrência de um evento não significa necessariamente a existência de uma transição subjacente a uma mudança de estado, pois para transições com guarda, é necessário que também seja válida a condição inerente à guarda. Adicionalmente, as "transições internas" num estado também são consumidoras de eventos, sem que haja qualquer alteração de estado (e consequentemente respectiva transição entre estados). Assim, um evento não é uma ocorrência de uma transição.

Portanto, não se assume a existência no UML de um conceito correspondente ao conceito de "Transition occurrence" presente no FRISCO.

Relative time, Absolute time

> *The strict partial order imposed on the sets of all transition occurrences is called* ***relative time****.*
>
> ***Absolute time*** *may be determined by a clock that issues (assumedly) regular pulses (transition occurrences of the clock, or clock events). An absolute time value may be assigned to some transition occurrence, by comparing that transition occurrence with the successive (absolute-time-determining) clock events. A strictly ordered set of time points can be defined on the basis of these clock events, called time axis.*
>
> [FRISCO98]

O UML não define explicitamente conceitos que se possam fazer corresponder a estes de "tempo relativo" e "tempo absoluto" do FRISCO. A título informativo, apresentam-se algumas das referências relevantes no UML relativas a tempo ou instante de tempo.

> *Event (2.12.2.4)*
>
> *An event is a specification of a type of observable occurrence. The occurrence that generates an event instance is assumed to take place at an instant in time with no duration. …"*
>
> [OMG-UML01, pág. 2-150]
>
> *Event (2.12.4.1)*
>
> *(…) No assumptions are made about the time intervals between event reception, event dispatching, and consumption. This leaves open the possibility of different semantic models such as zero-time semantics.*
>
> [OMG-UML01, pág. 2-161]
>
> *TimeEvent (2.12.2.16)*
>
> *A TimeEvent models the expiration of a specific deadline. Note that the time of occurrence of a time event instance; that is, the expiration of the deadline is the same as the time of its reception. However, it is important to note that there may be a variable delay between the time of reception and the time of dispatching (for example, due to queueing delays)…."*
>
> [OMG-UML01, pág. 2-155]
>
> *Comparison to classical statecharts (2.12.5.4)*
>
> *(…) Classical statecharts are based on the zero-time assumption, meaning transition stake zero time to execute. The whole system execution is based on synchronous steps where each step produces new events that will be processed at the next step. In object-oriented state machines, these assumptions are*

relaxed and replaced with these of software execution model, based on threads of execution and that execution of actions may take time."

Rule

A rule determines a set of permissible states and transitions in a specific context. In other terms, a rule governs a non-empty set of types of things by determining their permissible populations.

[FRISCO98]

Por interpretação simples desta definição, poder-se-ia fazer corresponder este conceito de regra ("rule") do FRISCO ao conceito UML de "máquina de estado" ("*StateMachine*"):

StateMachine (2.12.2.11)

A state machine is a specification that describes all possible behaviors of some dynamic model element. Behavior is modeled as a traversal of a graph of state nodes interconnected by one or more joined transition arcs that are triggered by the dispatching of series of event instances. During this traversal, the state machine executes a series of actions associated with various elements of the state machine.

[OMG-UML01, pág. 2-153]

Contudo, uma melhor observação à definição proposta pelo FRISCO, encontrar-se-á igualmente escrito:

Notes:

- *We use here the term 'rule' in the most generic sense, covering the widest possible spectrum, from the presumed rules governing the functioning of the "physical world" (very strong "laws of nature"), via rules of agreed behaviour in society (e.g. not always enforceable legal arrangements and business guidelines), to semantically or syntactically permissible and representations in some language (legal expressions). Rules are more or less strict or enforceable.*
- *Instead of specifying what is permissible, it may be easier in practice to formulate rules specifying what is forbidden, i.e. what the constraints are. (…)*

Examples:

The law of gravity; the laws of electromagnetism; the logical axioms that restrict statements to remain meaningful (e.g. true and false being mutually exclusive, a person being either active or inactive); mutually agreed rules of what may be entered into or changed inside a database, or passed over a network (semantic integrity, protocols, etc.).

[FRISCO98]

O UML oferece mecanismos gerais que podem ser aplicados a qualquer elemento de modelo; entre estes mecanismos encontram-se os constrangimentos ("*Constraint*") (ver Figura A.27), os quais permitem definir limitações a elementos de modelo:

Constraint and Comment (3.16)

Semantics (3.16.1)

A constraint is a semantic relationship among model elements that specifies conditions and propositions that must be maintained as true; otherwise, the system described by the model is invalid (with consequences that are outside the scope of UML). Certain kinds of constraints (such as an association "xor" constraint) are predefined in UML, others may be user-defined. A user-defined constraint is described in words in a given language, whose syntax and interpretation is a tool responsibility. A constraint represents semantic information attached to a model element, not just to a view of it.

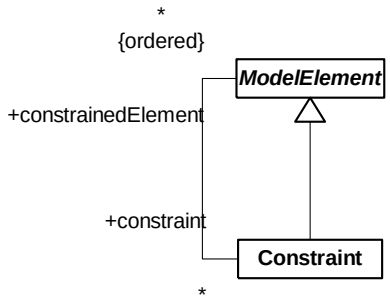

Figura A.27 – Extracto do metamodelo UML – "Constraint"

O UML permite (através de linguagem natural ou através uma linguagem formal - "Object Constraint Language (OCL)" – fornecida na própria especificação do UML), a definição de constrangimentos especificados pelo utilizador. Contudo, faz-se notar que quando se estabelecem e se caracterizam elementos ou relacionamentos entre elementos, já se estão (de alguma forma) a estabelecer regras. Por exemplo, quando se condiciona a multiplicidade de um relacionamento, obriga-se ou limita-se o número de instâncias a associadas a outra instância; quando se especificam transições, está-se a definir as regras que norteiam a transição de estados de uma entidade. Estes são exemplos de constrangimentos ou regras de comum utilização, os quais não são definidos explicitamente através do elemento "*Constraint*" dos mecanismos gerais oferecidos pelo UML.

O UML não assume explicitamente o termo regra ("rule"), nem no seu metamodelo, nem na sua especificação; embora faça uso genérico deste termo para a especificação da linguagem, não o assume como um elemento relevante na sua forma de ver o mundo. Em vez, usa (conforme anteriormente dito) o termo "*Constraint"* para definir semântica adicional a elementos de modelo:

Constraint (as extended) (2.6.2.1)

The constraint concept allows new semantics to be specified linguistically for a model element. The specification is written as an expression in a designated constraint language. The language can be

> *specially designed for writing constraints (such as OCL), a programming language, mathematical notation, or natural language.*
>
> [OMG-UML01, pág. 2-163]

Portanto, assume-se como correspondência para regra "rule" do FRISCO o conceito de constrangimento ("*Constraint*"), fazendo no entanto notar a imperfeição e limitações desta correspondência.

A.2. Actor, Acções e Conceitos associados

Embora considerando como suficiente os conceitos definidos na camada fundamental para especificar quaisquer concepções de domínios, o FRISCO, com o intuito de facilitar o tratamento da informação e da comunicação em geral, e, dentro das organizações em particular, introduz mais algumas especializações aos conceitos fundamentais anteriormente definidos [FRISCO98].

Antes de procedermos a considerações sobre os conceitos, convém relembrar aqui, mais uma vez que, em acordo com as definições de "sistema de informação "fornecidas por Carvalho [Carvalho99], o UML se destina a modelar sistemas de informação do tipo SI3, enquanto que o FRISCO essencialmente se destina modelar sistemas de informação do tipo SI4. Tal resultará numa notória dificuldade de conciliar conceitos de ambas as partes. Após esta breve recapitulação, continue-se a análise das definições do FRISCO, e a procura de terminologia adequada/elementos do metamodelo do UML.

Actor, Action

An actor is a special thing conceived as being "responsible" or "responsive" and as being able to "cause" transitions, and is therefore part of their pre-states, and, if not "destroyed" or "consumed" by the transitions, also part of their post-states.

Notes:

- *One actor can in general cause more than one transition, i.e. perform more than one action.*
- *Actors can be classified in many different ways. For our purposes, a useful classification is to distinguish between actors having the normal capabilities of human beings, which we will call human actors, and all other actors, such as other living beings or devices.*

An action is a transition involving a non-empty set of actors in its pre-state, and, if not "destroyed" or "consumed" by the action, in its post-state as well, and involving a non-empty or empty set of other things (actands) as part of its pre-state, and having a non-empty or empty set of other things (actands) in its post-state.

Examples:

- *Stock-taking (action) by a warehouse-clerk (actor) checking current stock, and producing a stock-inventory;*
- *Issuing a stock item (action) by a stock supervisor (actor), resulting in a change of stock level;*
- *Writing (action) a report by an author (actor);*
- *Expressing (action) a conception by a person (actor), in the form of a the representation.*

[FRISCO98]

Quando o FRISCO refere um actor como sendo capaz de provocar "transições" e de fazer parte do "pré-estado" e, caso não seja "consumido", também parte do "pós-estado" de uma acção, levantam-se algumas interrogações relativamente ao conceito de estado.

Será que esta noção de "estado", é idêntica em sentido, ao conceito subjacente ao elemento "*SimpleState*" do UML, anteriormente estabelecido como o mais adequado para o conceito de "estado" do FRISCO? O UML considera o conceito de estado como sendo um conceito útil para expressar "situações" particulares em que uma entidade se pode encontrar, para as quais o comportamento de uma entidade poderá ser diferente perante eventos ou estímulos idênticos. O FRISCO, com esta definição de actor, dá a entender o conceito de "estado" como tendo um significado mais "vasto", a ponto de significar "um estado actual das coisas" (a exemplo do que por vezes se pode considerar, relativamente a uma instância actual de um esquema de base de dados), e no qual, uma acção que seja empreendida resultará num novo estado de coisas. Assim, relativamente à correspondência anteriormente feita[48], poder-se-á dizer que o conceito de "*SimpleState*" do UML estará em correspondência, não plena, mas como sendo uma "especialização", de carácter mais restrito, do conceito de "estado", mais genérico, do FRISCO.

O FRISCO estabelece o conceito de actor, e de acções, de "actands", porque os considera fundamentais na sua concepção do mundo relativo a uma estrutura organizacional de carácter social, em que predomina a informação e a comunicação, e no qual os actores interagem entre si, empreendendo as acções consideradas úteis ou necessárias. Adicionalmente, o FRISCO distingue entre actores humanos e actores não humanos.

E o UML? O UML também define o conceito de actor, o qual está contemplado no seu metamodelo (ver Figura A.28).

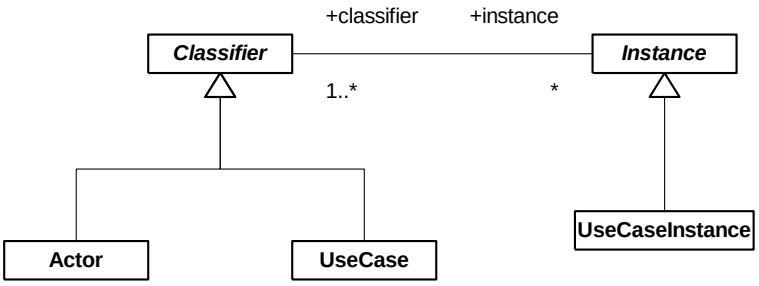

Figura A.28 – Extracto do metamodelo UML – "Actor"

Será que este elemento "Actor" do UML terá o mesmo significado e intenção do actor no FRISCO? O UML define actor como sendo:

Actor (2.11.2.1)

[48] Aquando da análise do conceito "State" do FRISCO (pág. 99)

> *An actor defines a coherent set of roles that users of an entity can play when interacting with the entity. An actor may be considered to play a separate role with regard to each use case with which it communicates.*
>
> [OMG-UML01, pág. 2-135]

Pode-se ainda encontrar na especificação outras indicações que ajudam a esclarecer o conceito de actor:

> *Actors model parties outside an entity, such as a system, a subsystem, or a class that interact with the entity. (…) If the entity is a system, the actors represent both human users and other systems.*
>
> *(…) Since an actor is outside the entity, its internal structure is not defined but only its external view as seen from t he entity. Actor instances communicate with the entity by sending and receiving message instances to and from use case instances and, at realization level, to and from objects. This is expressed by associations between the actor and the use case or the class.*
>
> [OMG-UML01, pág. 2-140]

O que é que se pode extrair de imediato destas afirmações?

- O UML considera que o actor está fora da entidade/sistema que se pretende modelar.
- O UML não faz distinção entre actores humanos e não-humanos.
- No UML o actor representa um papel, e não necessariamente um ser humano, dispositivo, ou outro sistema.
- No UML, intimamente ligado ao conceito de actor, está o conceito de "caso de uso" ("*use case*"), com o qual um actor pode ter uma associação através da qual lhe é permitido a interacção (envio e recepção de mensagens) com a entidade.

Um caso de uso está definido no metamodelo do UML (conforme se pode constatar na Figura A.28), e é definido como:

> *UseCase (2.11.2.6)*
>
> *The use case construct is used to define the behavior of a system or other semantic entity without revealing the entity's internal structure. Each use case specifies a sequence of actions, including variants, that the entity can perform, interacting with actors of the entity.*
>
> [OMG-UML01, pág. 2-137]

Pode-se ainda encontrar na especificação esclarecimentos relativos à semântica deste elemento:

> *The purpose of a use case is to define a piece of behavior of an entity without revealing the internal structure of the entity. The entity specified in this way may be a system or any model element that contains behavior, like a subsystem or a class, in a model of a system.*

> *Each use case specifies a service the entity provides to its users; that is, a specific way of using t he entity . The service, which is initiated by a user, is a complete sequence. (…)*
>
> *A use case describes the interactions between the users and the entity as well as the responses performed by the entity, as these responses are perceived from the outside of the entity.*
>
> [OMG-UML01, pág. 2-141]

Temos então para efeitos de modelação, um sistema do qual um actor pode solicitar determinadas funcionalidades sob a forma de casos de uso; a estrutura ou comportamento interno do sistema não é do conhecimento do actor. Posto de outra forma, as acções que um actor desenvolve corresponderão às funcionalidades representadas pelos casos de uso; mais ainda, este actor não tem conhecimento das coisas (ou recursos) envolvidas na acção que é empreendida, as quais são denominadas por "actands" no FRISCO.

Portanto, pode-se depreender que a visão primária do UML sobre o mundo não é propriamente como um conjunto acções, empreendidas por actores, sobre "actands", mas sim, como um conjunto de acções que podem ser empreendidas por actores, sem pretender incluir a este nível, os "actands".

Podemos concluir que o UML, para além de identificar no seu metamodelo um conjunto de elementos relevantes para a modelação de um domínio, tem embutido naquele, uma forma própria de ver os "acontecimentos" no mundo, a qual não vai propriamente ao encontro da visão exposta pelo FRISCO.

Em jeito de síntese, o UML, tem o conceito de actor; esse actor pode ser ou não humano; pode iniciar e estar envolvido em acções, em que estas acções estão representadas por casos de uso; contudo, o que está envolvido nas acções não é considerado de relevância como elemento para modelação na perspectiva do actor relativamente à funcionalidade solicitada ao sistema. Assim, com as devidas diferenças, pode-se associar o conceito de "actor" do FRISCO ao elemento "*Actor*" presente no metamodelo do UML e, por coerência, fazer corresponder "acção" no FRISCO ao elemento "caso de uso" ("*UseCase*"), igualmente presente no metamodelo do UML. No entanto, considerando a forma de o FRISCO ver o mundo como um "actores que desenvolvem acções sobre actands", opta-se por não fazer corresponder nenhum elemento do metamodelo do UML ao conceito de "actand" do FRISCO.

Relativamente ao conceito de acção, podem ainda ser tecidas outras considerações. Por exemplo, não se poderá deixar de referir que UML também possui o conceito de acção ("*Action"*), o qual é enquadrado no metamodelo do UML (ver Figura A.29):

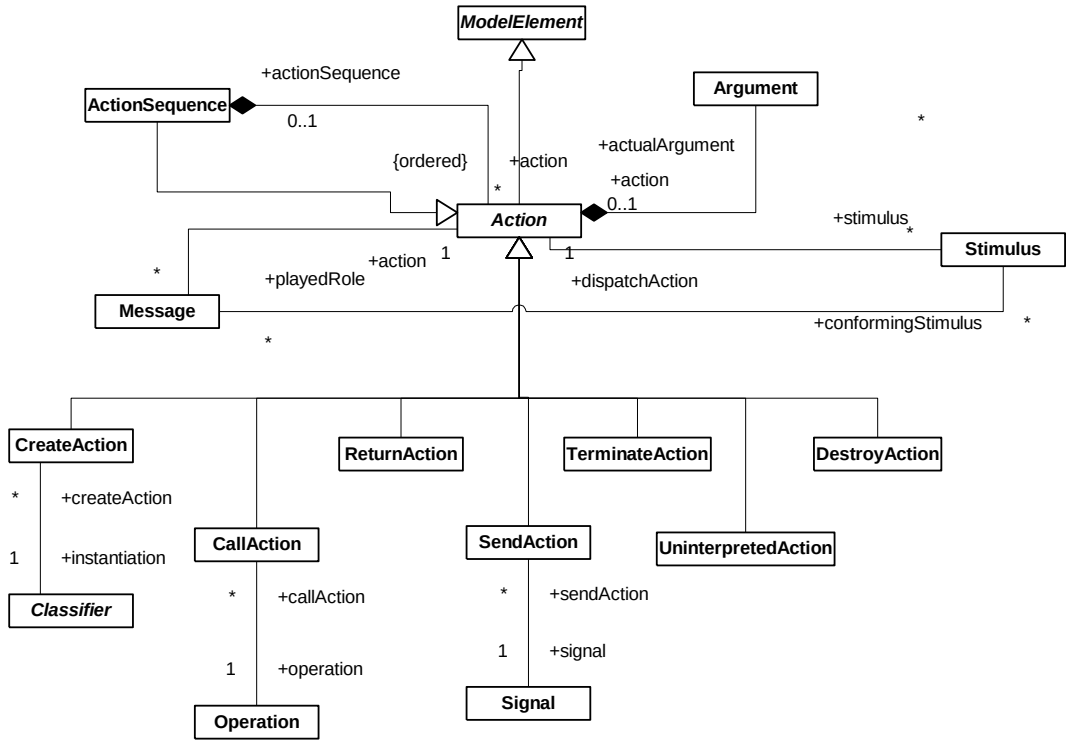

Figura A.29 – Extracto do metamodelo UML – "Action"

Será que se poderia considerar este como adequado ao conceito de "acção" do FRISCO? Veja-se o conceito de acção:

> *Action (2.9.2.1)*
>
> *An action is a specification of an executable statement that forms an abstraction of a computational procedure that results in a change in the state of the model, and can be realized by sending a message to an object or modifying a link or a value of an attribute.*
>
> [OMG-UML01, pág. 2-98]

Esta definição não faz qualquer referência a actores; as suas especializações reforçam o seu carácter computacional:

> *CallAction (2.9.2.5)*
>
> *A call action is an action resulting in an invocation of an operation on an instance. A call action can be synchronous or asynchronous, indicating whether the operation is invoked synchronously or asynchronously.*
>
> [OMG-UML01, pág. 2-100]
>
> *CreateAction (2.9.2.7)*
>
> *A create action is an action resulting in the creation of an instance of some classifier.*
>
> *DestroyAction (2.9.2.8)*

A destroy action is an action that results in the destruction of an object specified in the action.

[OMG-UML01, pág. 2-101]

ReturnAction (2.9.2.18)

A return action is an action that results in returning a value to a caller.

SendAction (2.9.2.19)

A send action is an action that results in the (asynchronous) sending of a signal. The signal can be directed to a set of receivers via an objectSetExpression, or sent implicitly to an unspecified set of receivers, defined by some external mechanism.

[OMG-UML01, pág. 2-105]

TerminateAction (2.9.2.23)

A terminate action results in self-destruction of an object.

[OMG-UML01, pág. 2-107]

Constata-se assim que o conceito de "acção" ("*Action"*) do UML (concebido no âmbito de um SI3) não se coaduna com o conceito de "Acção" proposto pelo FRISCO.

No extracto de metamodelo do UML apresentado na Figura A.29, são ilustrados relacionamentos entre acções, mensagens e estímulos. Quais os significados destes relacionamentos? Um estímulo representa a comunicação real que ocorre entre duas instâncias, enquanto que a mensagem, constitui a especificação de um estímulo, conforme podemos constatar na especificação do UML:

Stimulus (2.9.2.21)
A stimulus reifies a communication between two instances.

[OMG-UML01, pág. 2-106]

Message and Stimulus (3.63)

Semantics (3.63.1)

A Stimulus is a communication between two Instances that conveys information with the expectation that action will ensue. A Stimulus will cause an Operation to be invoked, raise a Signal, or cause an Instance to be created or destroyed. A Message is a specification of Stimulus; that is, it specifies the roles that the sender and the receiver Instances must conform to, as well as the Action which will, when executed, dispatch a Stimulus that conforms to the Message.

[OMG-UML01, pág. 3-111]

Outra consideração relativamente ao conceito de "acção", é facto de o UML possuir o conceito de diagrama de actividade, e de oferecer no seu metamodelo elementos propícios para utilização nestes diagramas. Será que o que vulgarmente denominamos por actividade poderia corresponder a "acção" do FRISCO? Na verdade, não. Temos de relembrar que uma

acção está intimamente ligada com o conceito de actor; e, se levarmos em conta que aquilo que chamamos de actividade, é na realidade no UML uma especialização (indirecta) do conceito de "estado" (*State*) – à qual é dada o nome de "*ActionState"* –, fica posta de parte esta possibilidade. A Figura A.30 ilustra o elemento "*ActionState*" presente no metamodelo:

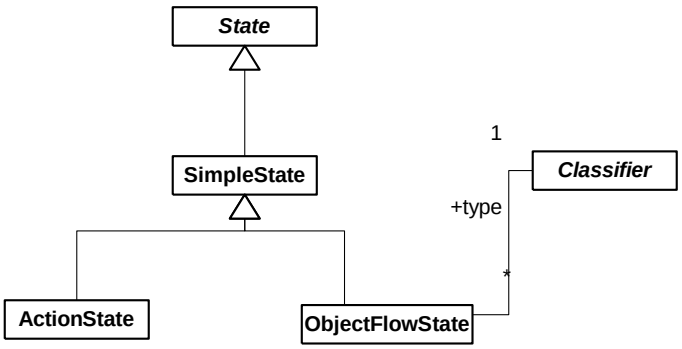

Figura A.30- Extracto do metamodelo - "ActionState"

A própria definição de diagrama de actividades no UML vai ao encontro do que foi dito:

> *Activity Diagram (3.84)*
>
> *Semantics (3.84.1)*
>
> *An activity graph is a variation of a state machine in which the states represent the performance of actions or subactivities and the transitions are triggered by the completion of the actions or subactivities. It represents a state machine of a procedure itself.*
>
> [OMG-UML01, pág. 3-156]

Conclui-se que a haver um conceito que melhor se coadune com o conceito de acção do FRISCO (o qual é apresentado numa visão do mundo no qual um conjunto de actores interagem, desenvolvendo acções que terão algum efeito sobre "actands"), esse conceito deverá ser o de casos de uso ("*UseCase"*), com todas as considerações anteriormente mencionadas.

Deve-se também dizer algo mais sobre os diagramas de actividade. Estes diagramas são de facto úteis para a modelação de um fluxo de actividades, actividades essas representadas por estados de acção, os quais podem, através de "pistas" ("swimlanes"), serem atribuídos em termos de responsabilidade ou de ocorrência (ou sob outro critério), a entidades. Mais ainda, juntamente com este fluxo de actividades, é possível representar um fluxo de "objectos" criados ou utilizados pelas actividades representadas neste diagrama. Poder-se-á então dizer que se considerarmos as ditas entidades como actores, as ditas actividades como acções, e os ditos objectos como possíveis "actands", então (possivelmente) encontraremos no UML conceitos que nos permitem estabelecer alguma correspondência com os conceitos de "actores-acções-actands". No entanto, levanta-se a questão: será que pelo facto deste caso de uso poder ser detalhado (numa perspectiva interna, nas várias actividades/acções que

empreendidas numa determinada ordem satisfazem no seu todo o caso de uso), poderemos considerar os conceitos envolvidos neste detalhe como apropriados?

Fica lançada a questão, deixando no entanto em contraponto com esta possibilidade, a consciência que um actor no UML é visto como sendo externo ao sistema, o que no FRISCO, não é assim. Adicionalmente, a forma como no metamodelo do UML são dispostos e definidos os conceitos respeitantes a diagramas de actividade, não contribui para que esta possibilidade fosse, à partida, tida como a mais apropriada. Estando intimamente ligados os conceitos de "actor" e de "acção", não se conseguir fazer corresponder correctamente o conceito de "actor", é indício que existirão dificuldades em estabelecer uma correspondência adequada relativa ao conceito de "acção"de acção.

Composite action, Action occurrence, Co-action

> *A composite action is a composite transition with the same conditions as applying for the notion of action.*
>
> *An action occurrence is a transition occurrence with the same conditions as applying for the notion of action.*
>
> *A co-action is a special action performed by more than one actor in a co-ordinated way, pursuing a common goal.*
>
> [FRISCO98]

Depois do dito sobre acção, o que dizer sobre "acção composta" ("Composite action")? Considerando o caso de uso como a melhor correspondência (apesar de limitada) para a acção, não haverá um elemento do UML adequado a este "Composite action" do FRISCO. Uma acção composta terá, por coerência ser vista como sendo composta por uma sequência/composição de casos de uso.

O conceito de "ocorrência de acção" ("Action occurrence") corresponderá ao elemento do metamodelo "*UseCaseInstance"* (Figura A.31),

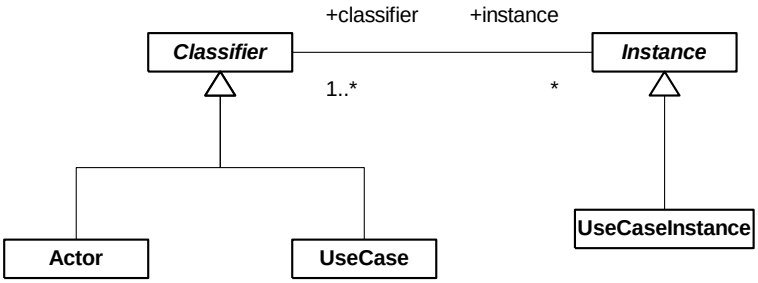

Figura A.31 – Extracto do metamodelo – "UseCaseInstance"

e, uma co-acção ("Co-action") terá der ser vista como a participação conjunta de mais do que um actor na interacção com um caso de uso.

Actand, Input actand, Output actand, Resource

> *An actand is a thing involved in the pre-state or post-state of an action, not considered as an actor for that action.*
>
> *An input actand is a part of the pre-state of an action, excluding the actors.*
>
> *An output actand is a part of the post-state of an action, excluding the actors.*
>
> *The pre-state of an action, i.e. the union of the set of actors and the set of input actands of that action, is called its resources.*
>
> [FRISCO98]

Tendo presentes as considerações tecidas na definição de actor, não se encontra no UML o conceito que adequadamente vá ao encontro do conceito de "actand" definido no FRISCO. Não quer isto dizer que, no UML, não exista forma de representar este conceito, mas tal envolveria recorrer aos mecanismos de extensão da linguagem (por exemplo, estereotipar uma classe com o título de "actand" poderia ser uma solução), e, não poderia ser feito isoladamente sem ter a atenção aos conceitos de "actor" e de "acção" com os quais se relaciona.

Assim, por coerência com o anteriormente dito, não se faz corresponder o conceito "actand" nem a nenhum elemento do metamodelo do UML, nem a nenhum conceito (ou termo) referenciado na especificação. Consequentemente, o mesmo acontece para os conceitos "input actand", "output actand" e "resource".

Action context

> *The action context of an action is a special, optional part of the pre-state of that action, qualifying the context or situation in which that action is performed, and determining or modifying at least one of its output actands.*
>
> [FRISCO98]

O documento de especificação do UML apresenta por vezes a utilização do termo contexto ("context") com o sentido comum utilizado na linguagem corrente; mais ainda, sendo "context" um termo utilizado com frequência, o documento fornece no glossário de termos uma definição para o termo contexto ("context"), cuja consideração de ser ou não condizente com a noção de "action context" do FRISCO poderá eventualmente ser subjectiva e dependente de interpretação pessoal:

> *context A view of a set of related modeling elements for a particular purpose, such as specifying an operation.*
>
> [OMG-UML01, pág. B-6]

Fica no entanto registado que no seu metamodelo, o UML não contempla nenhum elemento que se possa fazer corresponder à noção de "Action context" do FRISCO.

Goal, Goal-pursuing actor

> *The goal of an action is a special input actand of that action, pursued by the actors of that action and stating the desired output state intensionally.*
>
> *A goal-pursuing actor is an actor performing an action, who deliberately aims at a specific goal when involved in that action.*
>
> [FRISCO98]

No metamodelo do UML, não se encontra nenhum elemento que se adeqúe ao conceito "objectivo" ("goal") do FRISCO. Consequentemente, não se poderá dizer que no UML, um actor tem ou não tem um objectivo ("goal-pursuing actor").

Na prática, em UML, quando um actor interage com um caso de uso (sendo este último definido como uma peça coerente de funcionalidade que traduz algo de valor para um utilizador), presume-se que o actor sabe o tipo de resultado ao empreender uma acção de invocação de um caso de uso; então, subjacente à utilização de uma funcionalidade representada por um caso de uso, um actor tem um objectivo. Poder-se-á assim assumir que todo o actor tem um objectivo; contudo este não tem representação explícita num elemento de metamodelo do UML (a este propósito, recorda-se o metamodelo proposto por Penker [Penker00] para a modelação de negócios[49], o qual define explicitamente um elemento para a representação de um "goal" associado a um processo de negócio).

Portanto, nem "goal" nem "goal-pursuing actor" do FRISCO encontram elementos no UML que os possam representar.

A.3. Restantes Conceitos

Com base nos conceitos anteriores foram então definidos os conceitos relevantes em sistemas de informação.

Domain, Domain component, Domain environment

> *A **domain** comprises any "part" or "aspect" of the "world" under consideration.*
>
> *Examples:*
> *•An enterprise (viewed as part of society) composed of departments.*
> *•A department (of that enterprise) composed of employees.*
> *•The recently engaged employees (a domain usually overlapping a number of departments).*
>
> *A **domain component** is any "part" or "aspect" of that domain.*
>
> *Examples:*

[49] Apresentado anteriormente neste documento no ponto 5.1.5.1.1.

- *An employee of a department.*
- *An employee out of the set of the recently engaged employees.*
- *A department of an enterprise.*

*A **domain environment** is the "world" without that domain.*

[FRISCO98]

No UML, não existe qualquer elemento de metamodelo que se refira a este domínio ("domain"). No entanto, define no seu glossário o termo "domínio":

> ***domain*** *An area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area.*
>
> [OMG-UML01, pág. B-8]

Podem-se também extrair do documento, algumas expressões que revelam a terminologia utilizada para referir à concepção de domínio:

> *analysis - The part of the software development process whose primary purpose is to formulate a model of the problem domain. Analysis focuses what to do, design focuses on how to do it.*
>
> [OMG-UML01, pág. B-3]
>
> *A model is an instance of a metamodel. The primary responsibility of the model layer is to define a language that describes an information domain.*
>
> *The primary responsibility of the user objects layer is to describe a specific information domain.*
>
> [OMG-UML01, pág. 2-5]
>
> *A package can be used to define a framework, specifying a reusable architecture for all or part of a system. Frameworks may include reusable classes, patterns or templates. When frameworks are specialized for an application domain, they are sometimes referred to as application frameworks.*
>
> [OMG-UML01, pág. 2-198]
>
> *Classes can be stereotyped as Types or Implementation Classes (although they can be left undifferentiated as well). A Type is used to specify a domain of objects together with operations applicable to the objects without defining the physical implementation of those objects.*
>
> [OMG-UML01, pág. 3-49]

Pela leitura do documento de especificação, observa-se que o UML faz uso genérico do termo "domínio" ("domain"), não se encontrando para além do próprio termo "domínio", definição ou termo específico que se equipare ao termo "componente de domínio" ("domain component") do FRISCO.

Relativamente ao ambiente do sistema ("environment") do FRISCO, embora não se encontre expresso no seu metamodelo do UML ou definido em qualquer outra parte, o documento de

especificação utiliza o mesmo termo ambiente ("environment") com o mesmo significado, conforme se pode constatar pelas expressões a seguir apresentadas:

> *Class (2.5.2.9)*
>
> *A class is a description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. A class may use a set of interfaces to specify collections of operations it provides to its environment.*
>
> [OMG-UML01, pág. 2-26]
>
> *The interaction between a use case and its actors can also be presented in collaboration diagrams for specification of the interactions between the entity containing the use case and the entity's environment.*
>
> [OMG-UML01, pág. 2-142]
>
> *The model may also contain model elements describing relevant parts of the system's environment. The environment is typically modeled by actors and their interfaces.*
>
> [OMG-UML01, pág. 2-202]

Em suma, para os conceitos "domain", "domain component" e "domain environment" do FRISCO, não se encontram elementos adequados no metamodelo do UML para os representar. No entanto, o UML define "domain" no seu glossário de termos, e faz uso genérico do termo "environment" (no documento de especificação) com o mesmo significado que o conceito de ambiente do FRISCO. Relativamente ao termo "domain component" do FRISCO, não se encontra qualquer conceito no UML ao qual se o possa associar.

Human actor

> *A **human actor** is a responsible actor with the capabilities and liabilities of a normal human being, in particular capable of performing perceiving actions, conceiving actions and representing actions.*
>
> [FRISCO98]

Conforme já referido, o UML não apresenta um termo que distinga um actor humano de um actor não-humano.

Perception, Perceiving action, Perceiver, Conception, Conceiving action, Conceiver, Conceiving context, Interpreting action, Interpreter, Interpreting context

> *A **perception** is a special actand resulting from an action whereby a human actor observes a domain with his senses, and forms a specific (static, non-time-varying, or dynamic, time-varying) pattern of visual, auditory or other sensations of it in his mind.*

*A **perceiving action** is a special action of a human actor having a domain as input actand and a perception as output actand.*

*A **perceiver** is a human actor involved in a perceiving action.*

*A **conception** is a special actand resulting from an action whereby a human actor aims at interpreting a perception in his mind, possibly in a specific action context.*

*A **conceiving action** is a special action of a human actor having a perception and possibly some action context as input actand(s) and a conception as output actand.*

*A **conceiver** is a human actor involved in a conceiving action.*

*A **conceiving context** is the action context of a conceiving action.*

*An **interpreting action** is the sequence of a perceiving action performed on a domain, resulting in a perception of that domain, followed by a conceiving action performed on that perception, resulting in a conception.*

*An **interpreter** is a human actor performing an interpreting action.*

*An **interpreting context** is the action context of an interpreting action.*

[FRISCO98]

Todos estes termos existem no FRISCO em resultado da sua consideração da importância de distinguir o real, a percepção, a concepção e a representação. O UML não tem qualquer preocupação com conceitos a este nível.

Symbol, Alphabet, Symbolic construct, Language

*A **symbol** is a special entity used as an undividable element of a representation in a language.*

*An **alphabet** of a language is a non-empty and finite set of symbols.*

*A **symbolic construct** is a non-empty and finite "arrangement" of symbols taken from an alphabet. In the one-dimensional case, an arrangement is just a sequence of symbols (a "sentence"). In the n-dimensional case (n>1), it may be any arrangement of its constituting symbols in the n-dimensional space. Provided one considers the elements of arrangement (such as sequence) belonging to the alphabet, a symbolic construct is a non-empty and finite set of symbols.*

*A **language** is a non-empty set of permissible symbolic constructs. The permissible symbolic constructs in a language are determined either extensionally by enumeration or intensionally by a set of rules. The rules of a language may be syntactic ("grammar") as well as semantic ("semantic rules").*

[FRISCO98]

Sendo essencialmente uma linguagem de modelação de sistemas de software, é certo que dificilmente o metamodelo do UML contemplará estes conceitos do FRISCO; o metamodelo do UML define elementos que possam ser utilmente empregues na modelação ou necessários

para uma definição correcta da sintaxe da própria linguagem. Pode-se no entanto, visto que o documento de especificação especifica uma linguagem, tentar perceber o que os autores do documento referem relativamente a estes conceitos. O único local do documento que trata de explicar como a linguagem está construída, é na secção respeitante a formalismo da linguagem[50], a qual ilustra como a linguagem é apresentada no documento. Nessa secção encontram-se referências relativas à sintaxe e semântica de uma linguagem:

> *"A common technique for specification of languages is to first define the syntax of the language and then to describe its static and dynamic semantics.*
>
> *"The syntax defines what constructs exist in the language and how the constructs are built up in terms of other constructs.*
>
> *Sometimes, especially if the language has a graphic syntax, it is important to define the syntax in a notation independent way, that is, to define the abstract syntax of the language.*
>
> *The concrete syntax is then defined by mapping the notation onto the abstract syntax.*
>
> *The static semantics of a language define how an instance of a construct should be connected to other instances to be meaningful, and the dynamic semantics define the meaning of a well-formed construct."*
>
> *"In constructing the UML metamodel different techniques have been used to specify language constructs, using some of the capabilities of UML. The main language constructs are reified into metaclasses in the metamodel."*
>
> [OMG-UML01, pág. 2-8]

Através destas referências, verifica-se que não existe qualquer consideração relativamente aos conceitos símbolo ("Symbol") e alfabeto ("Alphabet") do FRISCO, e que se pode corresponder a noção de construção simbólica ("Symbolic construct") do FRISCO à noção de "construct" utilizada no documento do UML. O conceito linguagem ("Language") do FRISCO, é certamente em significado, o mesmo, quer no FRISCO quer no documento UML. Fica, apesar de tudo, a certeza que no metamodelo não se encontram elementos que possam adequadamente representar estes conceitos de ("Symbol", "Alphabet", "Symbolic construct", "Language") do FRISCO.

Representation, Representing action, Representer, Representing context

> *A representation is a special actand describing some conception(s) in a language, resulting from an action whereby a human actor aims at describing his conception(s), possibly in a specific action context.*
>
> *A representing action is a special action of a human actor having a conception and possibly some action context as input actand(s) and a representation as output actand.*
>
> *A representer is a human actor involved in a representing action.*

[50] "*2.3.1 Levels of Formalism*" [OMG-UML01, pág. 2-8]

A representing context is the action context of a representing action.

[FRISCO98]

Conforme já referido, o UML não tem preocupação de foro teórico a este nível; contudo, se fosse colocada a questão: "O UML separa o conceito da sua representação?", a resposta teria de ser afirmativa. Um dos motivos já foi anteriormente referido neste documento: o facto de no metamodelo do UML estar expresso (ver Figura A.32) que a um elemento de modelo (conceito) pode estar associada uma representação (através de "*PresentationElement*"):

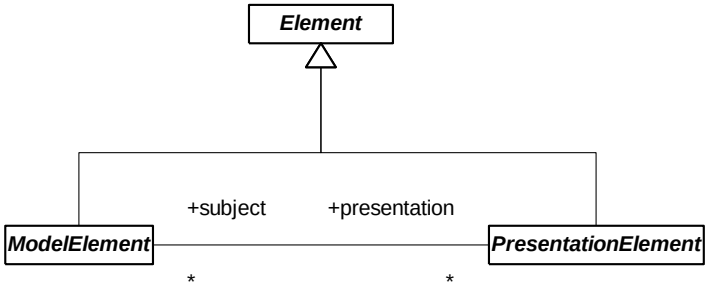

Figura A.32 – Extracto do metamodelo do UML – "PresentationElement"

Outro motivo que nos sustentaria a resposta, é o facto de o UML apresentar o seu metamodelo uma sintaxe denominada por "sintaxe abstracta", a qual procura expressar os elementos (e regras que regem a sua utilização conjunta) independentemente da notação, sendo esta notação apresentada e feita corresponder aos elementos de modelo em capítulo próprio[51];

Poderemos então fazer corresponder a "Representation" do FRISCO, o elemento do metamodelo UML "*PresentationElement*". Os conceitos associados aos termos "Representer", "Representer Action", "Representer Context" do FRISCO não são contemplados no UML.

Label, Reference

*A **label** is a special entity being an elementary representation and used for referring to some conception in an elementary way.*

Note:

We can distinguish between various kinds of labels, in particular between labels used for referring to entities (entity labels), and labels used for referring to predicators (predicator labels).

Examples:
•The string of characters 'Ludwig Wittgenstein' (an entity label)
•The string of characters 'is a person' (a predicator label)

[51] Apresentada no capítulo 3 do documento de especificação, "*3 – UML Notation Guide*" [UML OMG, pág. 3-1]

> *A **reference** is a special binary relationship between a conception and a representation used to refer to that conception.*
>
> [FRISCO98]

O FRISCO define "Label" como sendo uma "representação elementar" utilizada para referir a alguma concepção. Pode-se colocar a questão: como é que o UML se refere a uma concepção em particular? A procura elementos no documento de especificação que permitam responder a esta questão, leva-nos a considerar alguns factos:

Considere-se o seguinte extracto do metamodelo do UML, ilustrado pela Figura A.33:

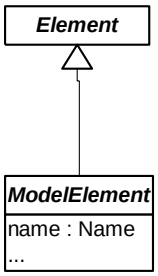

Figura A.33 – Extracto do metamodelo do UML – campo "name" de "ModelElement"

Confome se pode observar na Figura A.33, todo o elemento tem um atributo denominado "*name*", o qual é definido como podendo conter um valor do tipo "*Name*". O tipo "*Name*" (ver Figura A.34) é definido, entre outros, no metamodelo do UML[52] no pacote respeitante a tipos de dados:

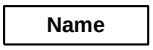

Figura A.34 – Extracto do metamodelo do UML – "Name"

A definição deste elemento do metamodelo ("*Name*") é:

> *Name (2.7.2.17)*
>
> *In the metamodel a Name defines a token that is used for naming ModelElements. A name is represented as a String.*
>
> [OMG-UML01, pág. 2-90]

sendo também definido no glossário de termos como

> ***name** A string used to identify a model element.*

[52] Na figura 2-11 do pacote "Data Types" [OMG-UML01, pág. 2-86].

Pode-se assumir que para referir a elementos de modelo, o UML usa o "nome" (*name*) associado (como atributo) ao elemento de modelo; esta conclusão vai de encontro à nota de rodapé contida no FRISCO relativamente ao conceito de "Label":

> *Synonym: 'Elementary sign token'*
>
> *Warning: The term 'Name' is sometimes used in this meaning and sometimes in various other meanings in the literature!*
>
> [FRISCO98]

Constata-se que o UML também define, não no metamodelo mas apenas no capítulo respeitante à notação da linguagem, o termo "*Label*"; no entanto, este é apenas usado como um termo de notação, não necessariamente como representação de uma concepção em particular:

> *Label (3.8)*
>
> *A label is a string that is attached to a graphic symbol.*
>
> *Semantics (3.8.1)*
>
> *A label is a term for a particular use of a string on a diagram. It is purely a notational term.*
>
> [OMG-UML01, pág. 3-10]

Relativamente a "Reference" do FRISCO, o UML não tendo contemplado o conceito no metamodelo, faz no entanto uso deste termo, definindo-o no glossário de termos como:

> *reference* *1. A denotation of a model element.*
>
> *2. A named slot within a classifier that facilitates navigation to other classifiers. Synonym: pointer.*
>
> [OMG-UML01, pág. B-16]

Desta forma, o UML também faz uso de "referências", as quais representam "apontadores" para, ou estão em representação de, outros elementos de modelo.

Em síntese, o UML usa o atributo "nome" ("*name*"), propriedade de todos os elementos, como representação para referir a uma concepção, permitindo portanto, que se possa associar o elemento de metamodelo "*Name*" como possível representação de "Label" do FRISCO. O UML não define no metamodelo elementos que permitam associar ao conceito "reference" do FRISCO, embora defina "reference"no seu glossário de termos e o utilize do documento de especificação.

Semiotic level

> *The **semiotic level** of a representation is the aspect considered in representing it. The semiotic levels are: physical, empirical, syntactical, semantical, pragmatic, and social.*

[FRISCO98]

O conceito de "Semiotic Level" do FRISCO está a um nível teórico no qual o UML não tece considerações.

Model denotation, System denotation, Information system denotation

> *A **model denotation** is a precise and unambiguous representation of a model, in some appropriate formal or semi-formal language.*
>
> *A **system denotation** is a precise and unambiguous representation of a system.*
>
> *An **information system** denotation is a precise and unambiguous representation of an information system.*
>
> [FRISCO98]

No FRISCO (em virtude de separar o que se concebe do que se representa), "Model denotation", "System Denotation", e "Information System Denotation", referem a representações das concepções "Model", "System" e "Information System" respectivamente.

Conforme dito aquando a análise[53] do conceito "Representation" do FRISCO, o UML distingue entre a concepção de um elemento de modelo ("*ModelElement*"), e a sua representação (através do elemento "*PresentationElement*").

Em particular[54], o UML distingue entre o "sistema físico" real e a sua possível representação (através do elemento de metamodelo "*Subsystem*"). Consequentemente, pode-se dizer que o documento de especificação (mas não no metamodelo) quando se refere a "physical system", estará a referir ao conceito "System" do FRISCO, e que o elemento "*subsystem*"[55] se refere a "System Denotation" do FRISCO.

Considerando a Figura 3.1 apresentada no ponto 3.5.1, pode-se dizer que o UML não se preocupa com a definição de elementos que levem em conta este rigor teórico. Assim, quando é utilizado um destes termos no UML (como por exemplo "Model"), este tanto serve para referir a sua concepção como a sua representação.

[53] Feita neste documento, na pág. 126.

[54] Em conformidade com o exposto aquando a análise do conceito "System" do FRISCO, na pág. 136.

[55] Mais precisamente "*(top-level) Subsystem*".

Modelling action, Modeller, System viewer, System representer

> *A **modelling action** is the sequence of a perceiving action performed on some domain, followed by a conceiving action on that perception, resulting in a model, and followed by a representing action on that model, resulting in a model denotation.*
>
> *A **modeller** is a human actor performing a modelling action.*
>
> *A **system viewer** is a human actor perceiving and conceiving a domain as a system. A system viewer recognises the system, by its distinction from the system environment, by its coherence, and because of its systemic properties.*
>
> *A **system representer** is a human actor representing a system in some language.*
>
> [FRISCO98]

O documento de especificação do UML usa os termos "*Modeler*" e "*Modeling*": "*Modeler*" para referir a quem desenvolve a actividade de modelação; "*Modeling*" para referir a acção que alguém empreende quando se encontra a modelar um sistema.

> *These diagrams, along with supporting documentation, are the primary artifacts that a modeler sees, although the UML and supporting tools will provide for a number of derivative views.*
>
> [OMG-UML01, pág. 1-2]
>
> *Introduction (2.1)*
>
> *Purpose and Scope (2.1.1)*
>
> *The primary audience for this detailed description consists of the OMG, other standards organizations, tool builders, metamodelers, methodologists, and expert modelers.*
>
> [OMG-UML01, pág. 2-2]
>
> *Concrete metamodel constructs are instantiable and reflect the modelling constructs used by object modelers (cf. metamodelers).*
>
> [OMG-UML01, pág. 2-12]

Tendo presente as definições do FRISCO relativas aos conceitos "System representer" e "System viewer", observa-se que UML não faz distinção entre estes; enquanto que o primeiro termo ainda pode ser visto como estando embebido no termo mais geral de "modeler", o segundo termo, "System viewer", não é considerado no UML.

Encontram-se no UML[56] referências a outros termos, os quais não representando conceitos do seu metamodelo, acarretam o nível de abstracção no qual se desenvolve uma actividade de modelação, nomeadamente os termos "Designer" ou "Business Analyst", conforme se pode

[56] Em "Unified Software Development Process" [Jacobson99], são considerados outros termos como "System Analyst", "Use-Case Engineer", "Component Engineer", "Architect" entre outros.

constatar pelas afirmações abaixo expostas; contudo, não existe qualquer intenção no UML de definir estes papéis.

> *Operation (2.5.2.30)*
>
> (…)
>
> *It is the responsibility of the system designer to ensure that deadlocks do not occur due to simultaneous blocks.*"
>
> [OMG-UML01, pág. 2-46]
>
> *Model (2.14.2.3)*
>
> *A model captures a view of a physical system. It is an abstraction of the physical system, with a certain purpose.*
>
> *(…)*
>
> *For example, a use-case model may be defined from the viewpoint of a business analyst stakeholder.*
>
> [OMG-UML01, pág. 2-189]

Pode-se concluir que no metamodelo não existem elementos que possam representar quaisquer destes elementos. Embora não os defina no seu glossário, o UML utiliza no seu documento os termos "modeler" e "modeling": o termo "modeler" para designar aquele que desenvolve actividade de modelação; o termo "modeling" para designar a actividade de modelação.

Pode-se aqui fazer uma pequena observação ao FRISCO: se uma acção "Modelling action" resulta num "Model Denotation", como se designa a acção que resulta num "System Denotation"? A resposta é que o FRISCO, apesar de preocupar-se em definir conceitos e termos como "System representer", "System viewer", não define um termo específico para esta acção; tal leva a que pela definição de sistema "*A system is a special model(…)*", seja também usado o termo "Modelling action".

Model, Intensional model, Extensional model , Meta-model

> *A **model** is a purposely abstracted, clear, precise and unambiguous conception.*
>
> > *Examples:*
> > *- An Entity-Relationship diagram (graphical representation) may be used to denote a model of the information-oriented aspects of an organisation.*
> > *- A Petri Net (graphics or algebraic expression) is often used to denote a model of the dynamics (or behaviour) of an information system.*
> > *- The documentation accompanying an application program (natural language + diagrams) provides a stylised description of its functionality, but usually does not constitute its model in any true sense.*

*An **intensional model** is that part of a model comprising the possibilities and necessities of a domain only, i.e. the types and rules.*

*An **extensional model** is that part of a model containing a specific population of the types in the corresponding intensional model, whereby this population must obey all rules determined in that intensional model.*

*A **meta-model** is a model of the conceptual foundation of a language, consisting of a set of basic concepts, and a set of rules determining the set of possible models denotable in that language.*

[FRISCO98]

O metamodelo do UML contempla o conceito de modelo. A Figura A.35 ilustra como o elemento "*Model*" é expresso no metamodelo.

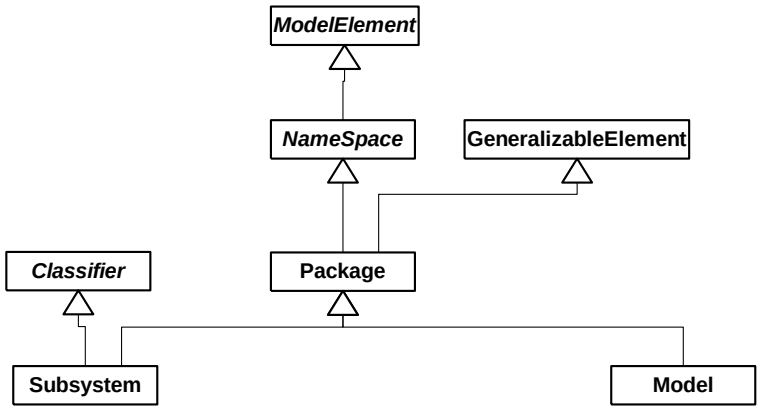

Figura A.35 – Extracto do metamodelo do UML – "Model"

Pode-se ainda encontrar no documento da especificação a definição de modelo ("*Model*"):

Model (2.14.2.3)

A model captures a view of a physical system. It is an abstraction of the physical system, with a certain purpose. This purpose determines what is to be included in the model and what is irrelevant. Thus the model completely describes those aspects of the physical system that are relevant to the purpose of the model, at the appropriate level of detail.

(…)

Different Models can be defined for the same physical system, where each model represents a view of the physical system defined by its purpose and abstraction level (for example, an analysis model, a design model, an implementation model). (…)

Stereotypes

«systemModel» A systemModel is a stereotyped model that contains a collection of models of the same physical system. A systemModel also contains all relationships and constraints between model elements contained in different models.

> *«metamodel» A metamodel is a stereotyped model denoting that the model is an abstraction of another model; that is, it is a model of a model. Hence, if M2 is a model of the model M1, then M2 is a metamodel of M1. It follows then that classes in M1 are instances of metaclasses in M2. The stereotype can be recursively applied, as in the case of a 4-layer metamodelo architecture.*
>
> [OMG-UML01, pág. 2-189]

A título meramente elucidativo, apresenta-se duas figuras (Figura A.36 e Figura A.37) extraídas do documento de especificação:

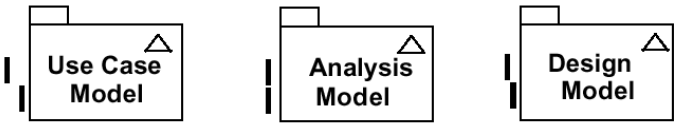

Figura A.36 – Três vistas de um sistema físico,;cada uma representada por um modelo ([OMG-UML01,pág.3-25])

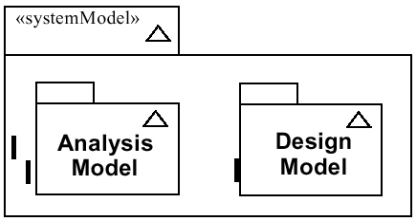

Figura A.37 – Modelo de sistema contendo modelos de análise e de desenho (extraído de [OMG-UML01, pág. 3-25])

Pela definição de modelo[57] apresentada, pode-se assumir que a concepção de modelo no UML, é consistente com a concepção de modelo do FRISCO.

Por norma, os modelos criados com UML são todos de uma perspectiva intencional; no entanto, existe também a possibilidade de ilustrar por vezes, o estado das instâncias e as ligações entre si. Por exemplo, enquanto que através de um diagrama de classes se especifica a estrutura dos objectos, e, as associações que representam a especificação das ligações que os objectos dos vários tipos podem estabelecer entre si, um diagrama de objectos, representa instâncias e ligações entre instâncias que se verificam num dado instante.

> *Object Diagram (3.20)*
>
> *An object diagram is a graph of instances, including objects and data values. A static object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time. The use of object diagrams is fairly limited, mainly to show examples of data structures.*
>
> [OMG-UML01, pág. 3-35]

Pode-se ainda encontrar a única referência aos termos "intension" e "extension"[58] no documento de especificação do UML:

[57] Faz-se aqui notar que, um elemento "*Model*" estereótipado como «*systemModel*» , representa "uma colecção de modelos", e não "uma colecção de TODOS os modelos", do mesmo sistema físico.

Class (3.22)

A class is the descriptor for a set of objects with similar structure, behavior, and relationships. The model is concerned with describing the intension of the class, that is, the rules that define it. The run-time execution provides its extension, that is, its instances.

[OMG-UML01, pág. 3-35]

Apesar destas considerações, o UML não define nem no seu metamodelo, nem no documento que o engloba, termos para os quais se possam fazer corresponder com clareza aos conceitos "intensional model" e "extensional model". Assume-se que em geral, no UML, a concepção de modelo "*Model*" tem de alguma forma implícito o conceito de "intensional model" no FRISCO.

Em conformidade com a linguagem utilizada no documento de especificação,

A metamodel is an instance of a meta-metamodel. The primary responsibility of the metamodel layer is to define a language for specifying models.

[OMG-UML01, pág. 2-5]

o termo "metamodelo" utilizado no UML está em consonância com o significado do conceito "metamodelo" definido pelo FRISCO. Mais ainda, conforme os extractos anteriormente apresentados, o UML fornece no seu metamodelo, através do elemento "*Model*", um estereótipo designado por «*metamodel*» que permite que um modelo seja considerado como um metamodelo. Pode-se então usar o termo metamodelo como a mesma concepção, quer no FRISCO, quer no UML.

Concluindo, para representar o conceito "Model" do FRISCO, o UML fornece o elemento "*Model*" do seu metamodelo. Para representar o conceito "metamodelo" do FRISCO, o UML permite representá-la através de um estereótipo «metamodel» do elemento "*Model*". Os termos "intensional model" e "extensional model" do FRISCO não se revêem em quaisquer dos elementos presentes no metamodelo.

System, System component, System environment, Sub-system

*A **system** is a special model, whereby all the things contained in that model (all the system components) are transitively coherent, i.e. all of them are directly or indirectly related to each other form a coherent whole. A system is conceived as having assigned to it, as a whole, a specific characterisation (the so-called "systemic properties").*

*A **system component** is a non-empty set of things being contained in that system.*

*The **system environment** of a system is the set of all things not being contained in that system.*

[58] Não existe nenhuma referência aos termos "intensional" ou "extensional".

> *Any **sub-system** S' of a larger system S is a system, itself. The set of all sub-system components of S' is a proper subset of the set of all system components of S.*
>
> [FRISCO98]

A procura no metamodelo de um elemento que identifique claramente o conceito de sistema[59] revela-se infrutífera. Assim, feita uma pesquisa no documento de especificação do UML, surgem algumas afirmações que nos dão indicações relevantes:

> *Models are used to capture different views of a physical system. Packages are used within a Model to group ModelElements. A Subsystem represents a behavioral unit in the physical system.*
>
> *(…) it is necessary to clearly distinguish between the physical system being modeled; that is, the subject of the model and the model element that represent the physical system in the model. For this reason, we consistently use the term physical system when we want to indicate the former, and the term (top-level) subsystem when we want to indicate the latter. An example of a physical system is a credit card service, which includes software, hardware, and wetware (people). The UML model for this physical system might consist of a top-level subsystem called CreditCardService, which is decomposed into subsystems for Authorization, Credit, and Billing.*
>
> [OMG-UML01, pág. 2-187]

Um "sistema físico" ("physical system") é o termo utilizado do documento de especificação do UML para designar o sistema do mundo real e para o qual se pretende estabelecer um modelo do sistema. Encontra-se no glossário de termos do documento de especificação do UML a definição de sistema físico:

> *physical system* *1. The subject of a model.*
>
> *2. A collection of connected physical units, which can include software, hardware and people, that are organized to accomplish a specific purpose. A physical system can be described by one or more models, possibly from different viewpoints. Contrast: system*
>
> [OMG-UML01, pág. B-15]

O elemento de modelo que representa, em termos de modelação, o sistema físico do mundo real, é designado por subsistema de "nível topo" [60] ("top-level subsystem").

Também no mesmo glossário de termos, aparece uma referência ao termo "sistema":

> *system* *A top-level subsystem in a model. Contrast: physical system.*
>
> [OMG-UML01, pág. B-19]

[59] Deve-se dizer que em virtude de o objecto "sistema de informação" do UML ser SI3 (conforme visto no ponto 5.1.2), os conceitos "System" e "Information System" sobrepõem-se no UML, tendo no contexto desta, o mesmo significado.

[60] Este é um termo designado neste documento, por se entender aqui apropriado.

Pela Figura A.38, pode-se confirmar no extracto do metamodelo que, ao contrário do termo sistema ("System"), o termo "Subsistema" ("*Subsystem*") denota um elemento presente no metamodelo do UML:

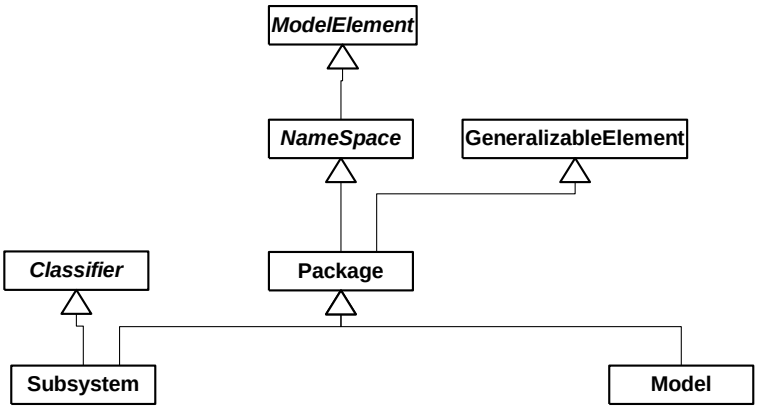

Figura A.38 – Extracto de metamodelo do UML – "Subsystem"

Subsistema é definido como:

> *Subsystem (2.14.2.5)*
>
> *A subsystem is a grouping of model elements that represents a behavioral unit in a physical system. A subsystem offers interfaces and has operations. In addition, the model elements of a subsystem are partitioned into specification and realization elements, where the former, together with the operations of the subsystem, are realized by the latter.*
>
> [OMG-UML01, pág. 2-192]

Apenas a título informativo (e de curiosidade) apresenta-se a notação proposta para representar um subsistema:

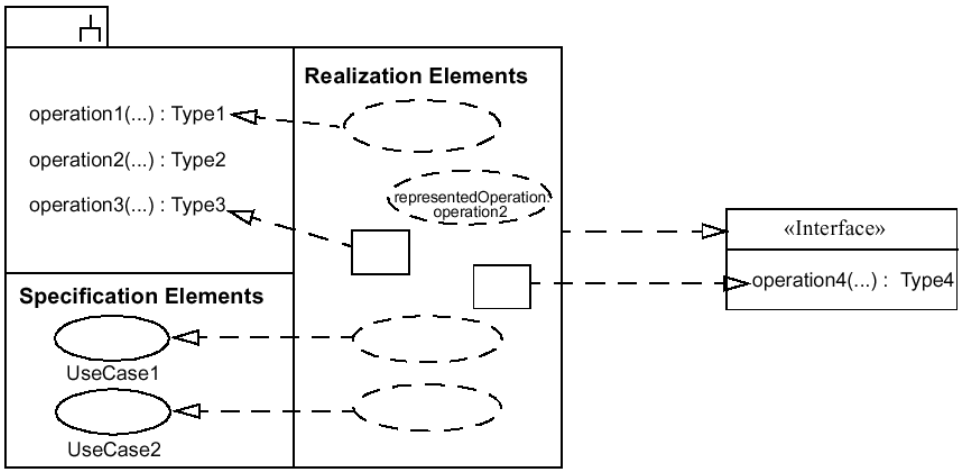

Figura A.39 – Notação de subsistema (extraída de [OMG-UML01, pág. 3-23])

Pelo até aqui dito, o UML não tem um elemento de modelo especificamente concebido para a concepção "sistema" do FRISCO. Tem é o conceito de subsistema, o qual permite não só

representar o "sistema" (designado em UML por "*top-level susbsystem*"), como também um subsistema do próprio sistema, ou seja, o equivalente a "subsystem" do FRISCO.

Faz-se a observação de que o documento de especificação do UML usa com frequência o termo "*system*" e não "*top-level subsystem*" para referenciar o sistema ao qual respeita a actividade de modelação que se está a desenvolver. Tal leva a questionar se o considerado "*top-level susbsystem*" vai de encontro à terminologia vulgarmente utilizada na comunicação. Eis alguns exemplos:

> *A component represents a modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces.*
>
> [OMG-UML01, pág. 2-31]
>
> *A model element is an element that is an abstraction drawn from the system being modeled.*
>
> [OMG-UML01, pág. 2-42]
>
> *The purpose of a model is to describe the possible states of a system and their behavior. The state of a system comprises objects, values, and links.*
>
> [OMG-UML01, pág. 2-71]
>
> *The Use Cases package is a subpackage of the Behavioral Elements package. It specifies the concepts used for definition of the functionality of an entity like a system.*
>
> *(…)*
>
> *The elements in the Use Cases package are primarily used to define the behavior of an entity, like a system or a subsystem, without specifying its internal structure.*
>
> [OMG-UML01, pág. 2-134]
>
> *The use case construct is used to define the behavior of a system or other semantic entity without revealing the entity's internal structure.*
>
> [OMG-UML01, pág. 2-137]
>
> *Actors model parties outside an entity, such as a system, a subsystem, or a class that i nt eract wi t h t he entity .*
>
> [OMG-UML01, pág. 2-140]

No que respeita ao conceito de "system component" do FRISCO, o qual é definido como "*a non-empty set of things being contained in that system* ", também não se encontra um termo adequado no UML; contudo, encontra-se no metamodelo do UML um elemento que permite representar um agrupamento de elementos de modelo, e que se dá pelo nome de "pacote" ("*package*"). O elemento pacote ("*Package")* é tido no documento de especificação como:

> *Package (2.14.2.4)*

> *A package is a grouping of model elements.*
>
> [OMG-UML01, pág. 2-190]
>
> *package A general purpose mechanism for organizing elements into groups. Packages may be nested within other packages.*
>
> [OMG-UML01, pág. B-14]

Embora também exista no UML o conceito de "*Component*":

> *component A modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces. A component is typically specified by one or more classifiers (e.g., implementation classes) that reside on it, and may be implemented by one or more artifacts (e.g., binary, executable, or script files).*
>
> [OMG-UML01, pág. B-5]

este não é considerado adequado relativamente à definição de "system component" do FRISCO.

Portanto, coloca-se a questão: podíamos considerar "pacote" ("package") do UML, como um elemento possível para representação do conceito "system component" do FRISCO, uma vez que este último é definido com "um conjunto não vazio de coisas" do subsistema? Talvez, mas neste documento opta-se por não o fazer devido ao conceito de "package", sendo um elemento para agrupamento de outros elementos de modelo, é demasiado abrangente (sendo possível a partir deste conceito derivar os conceitos de "modelo" e "subsistema") e genérico, o que o torna inapropriado para associar ao conceito mais limitado de "componente de sistema". Assim, "system component" não encontra elemento no UML que o possa adequadamente representar.

O conceito de ambiente de sistema "system environment" do FRISCO diz respeito a tudo o que não faz parte do sistema. Como é que o UML se refere a este conceito?

> *Event instances are generated as a result of some action either within the system or in the environment surrounding the system.*
>
> [OMG-UML01, pág. 2-161]
>
> *A Model also contains a set of ModelElements that represents the environment of the system, typically Actors, together with their interrelationships, such as Dependencies, Generalizations, and Constraints.*
>
> [OMG-UML01, pág. 2-189]
>
> *The model may also contain model elements describing relevant parts of the system's environment. The environment is typically modeled by actors and their interfaces. As these are external to the physical system, they reside outside the package/subsystem hierarchy. They may be collected in a separate package, or owned directly by the model.*

[OMG-UML01, pág. 2-202]

O metamodelo do UML não define explicitamente elemento que represente o ambiente de um sistema "system environment" do FRISCO. Contudo, o documento de especificação utiliza o termo "ambiente do sistema" ("environment of the system" ou "system's environment") para se referir ao que envolve o sistema.

Faz-se notar que a linguagem em si, não definindo terminologia ou elemento do metamodelo relativo ao ambiente do sistema, tem embebido no seu metamodelo um elemento que permite representar elementos do ambiente do sistema: o elemento actor ("*Actor*"), o que se pode confirmar em algumas das expressões anteriormente transcritas.

Dynamic system, Static system, Active system, Passive System, Open system, Closed System

> *A **dynamic system** is conceived as capable of undergoing change, i.e. some of the system components are transitions.*
>
> *A system that is not dynamic is called a **static system**.*
>
> *An **active system** is conceived as capable of doing something, i.e. some of the system components are actors performing actions on actands.*
>
> *A system is called a **passive system** if it is not active.*
>
> *An **open system** is conceived as one which may respond to external messages or triggers, i.e. there may be transitions within the system due external causes coming from the system environment.*
>
> *A system is called a **closed system** if the system environment cannot cause any transitions within the system.*
>
> [FRISCO98]

Estes são mais alguns exemplos de conceitos que não são considerados no UML, certamente devido ao seu objecto de interesse ser a constituição interna de um SI3. Assim, quer no metamodelo, quer no documento de especificação, não se encontram elementos relevantes que se possam associar a estes termos do FRISCO, derivados do conceito de sistema.

Knowledge, Information, Data, Shared knowledge

> ***Knowledge** is a relatively stable and sufficiently consistent set of conceptions possessed by single human actors.*
>
> ***Information** is the knowledge increment brought about by a receiving action in a message transfer, i.e. it is the difference between the conceptions interpreted from a received message and the knowledge before the receiving action.*

The term ***data*** *denotes any set of representations of knowledge, expressed in a language.*

Shared knowledge *is that knowledge of the individuals in a group of human actors, which they assume to be identical (or at least similar) to that of the others, as resulting from the negotiation process implicit in some communication.*

[FRISCO98]

O FRISCO distingue muito claramente os conceitos de conhecimento, dados e informação. O "conhecimento" ("Knowledge") é definido pelo FRISCO como um conjunto de concepções relativamente estáveis e suficientemente possuído por actores humanos; o UML não tem conceito ou termo para este "conhecimento" possuído por humanos. No documento de especificação do UML, relativamente a "conhecimento", apenas se encontra uma referência no glossário de termos, na entrada respeitante ao termo domínio:

domain An area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area.

[OMG-UML01, pág. B-8]

No FRISCO, "Informação" é definido como o aumento de conhecimento trazido por uma mensagem. O UML, para além de não considerar o conceito "conhecimento", também não considera qualquer conceito que se adeqúe ao conceito de "Informação" do FRISCO. Não no metamodelo do UML, mas sim no seu documento de especificação, é empregue o termo "informação" ("information") com o mesmo significado de "dados" ("Data") do FRISCO, conforme se pode constatar em algumas das expressões retiradas do documento:

message A specification of the conveyance of information from one instance to another, with the expectation that activity will ensue.

[OMG-UML01, pág. B-11]

stimulus The passing of information from one instance to another, such as raising a signal or invoking an operation. The receipt of a signal is normally considered an event.

[OMG-UML01, pág. B-18]

They [Tag definitions e tag values] *may be used to represent properties such as management information (author, due date, status), code generation information (optimizationLevel, containerClass).*

[OMG-UML01, pág. 2-75]

Strings - Present various kinds of information in an "unparsed" form.

[OMG-UML01, pág. 3-7]

Not all modeling information is presented most usefully in a graphical notation. Some information is best presented in a textual or tabular format.

[OMG-UML01, pág. 3-8]

Por vezes, o termo "dados" ("data") é utilizado no documento de especificação do UML, mas com uma frequência muito menor relativamente ao termo "informação". Extractos representativos desta utilização são:

> *User objects (a.k.a. user data) are an instance of a model. The primary responsibility of the user objects layer is to describe a specific information domain.*
>
> [OMG-UML01, pág. 2-5]
>
> *A Class defines the data structure of Objects, although some Classes may be abstract; that is, no Objects can be created directly from them.*
>
> [OMG-UML01, pág. 2-26]
>
> *The data content of an object comprises one value for each attribute in its full class descriptor (and nothing more).*
>
> [OMG-UML01, pág. 2-71]
>
> *An object diagram is a graph of instances, including objects and data values.*
>
> [OMG-UML01, pág. 3-35]

Conclui-se que, quer no metamodelo, quer no documento de especificação do UML, não se encontram correspondências para os conceitos "Knowledge" e "Information" do FRISCO; encontra-se sim, não no metamodelo mas apenas no documento, o termo "informação", o qual é utilizado com o significado equivalente a "dados" ("Data") pelo FRISCO. Sobre o conceito "Shared knowledge" do FRISCO, o UML não estabelece quaisquer termos ou considerações.

Message, Message transfer, Sender, Receiver

> *A message is data, transmitted by one actor (the sender) via a channel (a medium), and intended for a non-empty set of other actors (the receivers).*
>
> *A message transfer is a sequence of actions, the sending action by the sender and the receiving actions by the receivers, whereby the input actand of the sending action is the message to be sent, whereby the output actand of the sending action, being (in the simplest case) equal to the input actand of the receiving action, is the message on the channel, and whereby the output actand of the receiving action is the message received.*
>
> *A sender is an actor sending a message.*
>
> *A receiver is an actor receiving a message.*
>
> [FRISCO98]

A Figura A.40 permite observar que o UML contempla no seu metamodelo um elemento denominado de "mensagem" ("*Message*"):

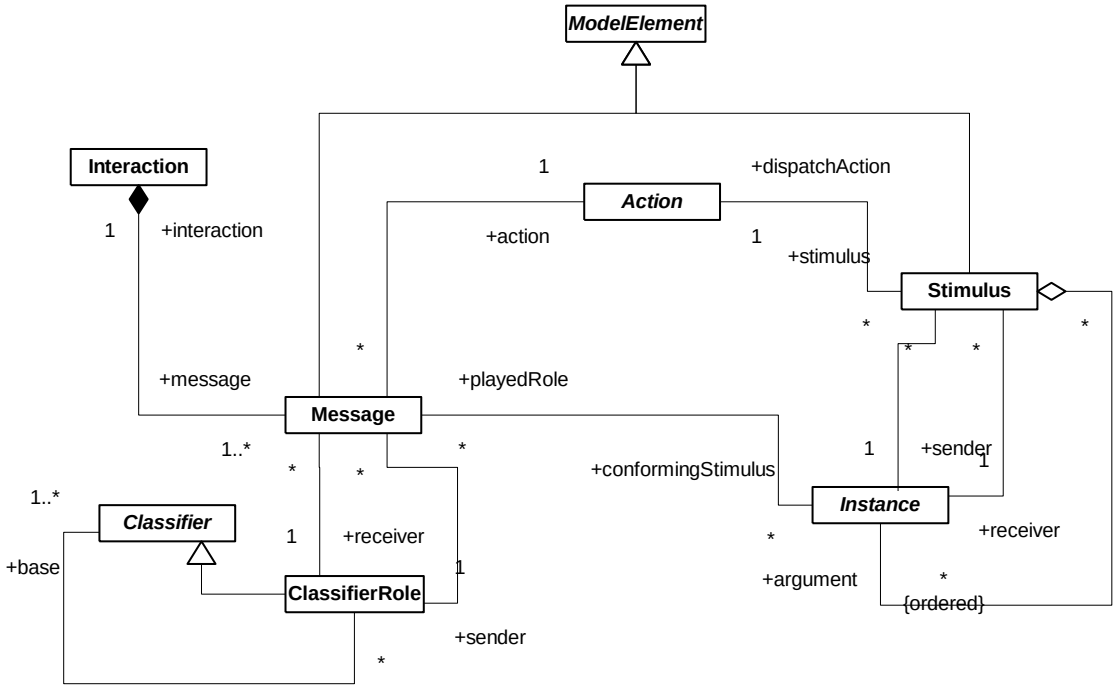

Figura A.40 – Extracto de metamodelo do UML – "Message"

O elemento "*Message*" do metamodelo é definido como:

> *Message (2.10.2.8)*
>
> *A message defines a particular communication between instances that is specified in an interaction.*
>
> *In the metamodel a Message defines one specific kind of communication in an Interaction. (…) The Message specifies not only the kind of communication, but also the roles of the sender and the receiver, the dispatching Action, and the role played by the communication Link. Furthermore, the Message defines the relative sequencing of Messages within the Interaction.*
>
> [OMG-UML01, pág. 2-124]

Pode-se ainda encontrar a explicação da relação entre mensagem ("*Message*") e estímulo ("*Stimulus*") ilustrada no metamodelo:

> *Message and Stimulus (3.63)*
>
> *Semantics (3.63.1)*
>
> *A Stimulus is a communication between two Instances that conveys information with the expectation that action will ensue. A Stimulus will cause an Operation to be invoked, raise a Signal, or cause an Instance to be created or destroyed. A Message is a specification of Stimulus; that is, it specifies the*

> *roles that the sender and the receiver Instances must conform to, as well as the Action which will, when executed, dispatch a Stimulus that conforms to the Message.*
>
> [OMG-UML01, pág. 3-111]

Verifica-se assim que uma mensagem define uma comunicação e constitui a especificação de um estímulo, o qual por sua vez é definido como acto de passagem de informação "*passing of information*" numa comunicação efectiva entre duas instâncias: uma no papel de emissor ("sender"), e outra no papel de "receptor" ("receiver").

> *stimulus The passing of information from one instance to another, such as raising a signal or invoking an operation. The receipt of a signal is normally considered an event.*
>
> [OMG-UML01, pág. B-18]

Sendo um estímulo um "acto" de passagem de informação entre intervenientes, e "mensagem" a sua especificação, a que corresponderá o conceito "Mensagem" do FRISCO, cuja definição começa por "*A message is data, transmitted by...'*? De facto, a procura de algum rigor leva a que não se possa fazer assumir como adequada a correspondência de "Mensagem" ("Message") do FRISCO quer para "Mensagem" ("*Message*"), quer para "Estímulo" ("*Stimulus*") do UML.

Uma análise atenta ao metamodelo (ver Figura A.41) revela onde poderá estar a informação que é passada entre instâncias:

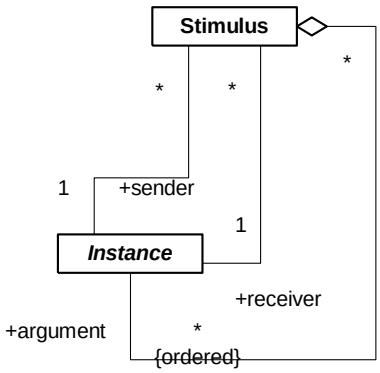

Figura A.41 – Extracto de metamodelo do UML –"Stimulus"

Na verdade, um estímulo tem associadas instâncias: uma no papel de "emissor", outra no papel de "receptor", e outras (zero, uma ou mais) no papel de "argumento". É neste papel de argumento que é feita a passagem de informação entre um emissor e receptor, o que pode ser verificado, para além de através do metamodelo, pela afirmação que se segue:

> *The Stimulus contains references to the sender and the receiver Instances playing the sender role and the receiver role of the Message, as well as a sequence of references to Instances being the result of evaluating the argument expressions of the dispatching Action.*

[OMG-UML01, pág. 3-124]

Após estas considerações, o que considerar como adequado para o conceito "Mensagem" definido no FRISCO? Na realidade, o UML não tem presente no metamodelo um elemento claramente adequado ao conceito de "mensagem" do FRISCO (a não ser que se considere a eventual colecção de argumentos que pode estar associado a um estímulo). Os termos "mensagem" e "estímulo" do UML, representam acções de transferência de informação, em que o último é a concretização do que é especificado no primeiro. Portanto, estes elementos do UML estarão mais adequados para representar o conceito "Message Transfer" do FRISCO[61]. Pode-se numa perspectiva de especificação, ver "Message Transfer" representado no UML pelo elemento "*Message*", ou numa perspectiva da real passagem de informação, ver "Message transfer" representado no UML pelo elemento "*Stimulus*", ambos presentes no metamodelo do UML. No entanto, uma vez que o FRISCO define os seus conceitos de forma em extensão[62], o elemento escolhido é o estímulo "S*timilus*".

"Emissor" ("Sender") e "Receptor" (Receiver") do FRISCO encontram correspondência no metamodelo do UML, respectivamente, aos papéis de "emissor" ("*sender*") e de "receptor" ("*receiver*") que as instâncias podem desempenhar quando envolvidas num estímulo[63].

Communication

> *Communication is an exchange of messages, i.e. a sequence of mutual and alternating message transfers between at least two human actors, called communication partners, whereby these messages represent some knowledge and are expressed in languages understood by all communication partners, and whereby some amount of knowledge about the domain of communication and about the action context and the goal of the communication is made present in all communication partners.*
>
> [FRISCO98]

O conceito de "comunicação" definido no FRISCO, tem a particularidade de exigir que na comunicação, estejam presentes, pelo menos dois actores humanos. Ora, conforme já por várias vezes dito, o UML está essencialmente concebido para sistemas SI3[64]; sobre os actores humanos (considerados como pertencentes ao ambiente do sistema a ser modelado), existe principalmente o interesse em modelar a sua comunicação com o sistema, e não propriamente, a comunicação entre aqueles. No UML, não sendo definido no metamodelo do UML um elemento que possa ser claramente associado a este conceito de "comunicação" do FRISCO, o termo "comunicação" no documento de especificação do UML está associado ao conceito

[61] Recorde-se que "mensagem" é definido no FRISCO como "…is data", e já se viu (pág. 141) que "Data" do FRISCO corresponderá ao termo "*information*" utilizado genericamente no documento de especificação do UML.

[62] Conforme observado neste documento aquando a análise do elemento "Type" do FRISCO (pág. 92).

[63] Ou mensagem, numa perspectiva de especificação.

[64] Em conformidade com o ponto 5.1.2 deste documento.

de estímulo, e é usado como termo vulgar sem atribuição especial de significado. Apresentam-se a seguir algumas afirmações extraídas do documento de especificação do UML que vêm em reforço do aqui dito.

Stimulus (2.9.2.21)

A stimulus reifies a communication between two instances.

In the metamodel Stimulus is a communication; that is, a Signal sent to an Instance, or an invocation of an Operation.

[OMG-UML01, pág. 2-106]

The stimulus uses a link between the sender and the receiver for communication.

[OMG-UML01, pág. 2-114]

Collaborations (2.10) (…)

The description of cooperating Instances involves two aspects: 1) the structural description of the participants, and 2) the description of their communication patterns.

[OMG-UML01, pág. 2-115]

An Interaction is defined in the context of a Collaboration. It specifies the communication patterns between the roles in the Collaboration. More precisely, it contains a set of partially ordered Messages, each specifying one communication;

[OMG-UML01, pág. 2-116]

The Interactions defined within the Collaboration specify the communication pattern between the Instances when they perform the behavior specified in the Operation or the UseCase.

[OMG-UML01, pág. 2-117]

Interaction (2.10.2.6)

An interaction specifies the communication between instances performing a specific task. Each interaction is defined in the context of a collaboration.

In the metamodel an Interaction contains a set of Messages specifying the communication between a set of Instances conforming to the ClassifierRoles of the owning Collaboration.

[OMG-UML01, pág. 2-123]

Message (2.10.2.8)

A message defines a particular communication between instances that is specified in an interaction.

[OMG-UML01, pág. 2-124]

> *A use-case instance is a performance of a use case, initiated by a message instance from an instance of an actor. As a response the use-case instance performs a sequence of actions as specified by the use case, like communicating with actor instances, not necessarily only the initiating one.*
>
> [OMG-UML01, pág. 2-142]

Considera-se então que o conceito "comunicação" ("communication") do FRISCO não encontra no metamodelo um elemento para o representar. Contudo, o termo "comunicação" é genericamente utilizado no documento de especificação do UML com o mesmo significado de "comunicação do FRISCO, salvaguardando contudo que no UML não é necessário que a comunicação envolva a presença de pelo menos dois actores humanos (o que para o FRISCO se torna necessário).

Organisational system, Norm

> *An organisational system is a special kind of system, being normally dynamic, active and open, and comprising the conception of how an organisation is composed (i.e. of specific actors and actands) and how it operates (i.e. performing specific actions in pursuit of organisational goals, guided by organisational rules and informed by internal and external communication), where its systemic properties are that it responds to (certain kinds of) changes caused by the system environment and, itself, causes (certain kinds of) changes in the system environment.*
>
> *Norms are socially agreed rules affecting and to a large extent directing the actions within an organisational system.*
>
> [FRISCO98]

Não fazendo parte do âmbito da especificação do UML, os conceitos de "Organisational system" e "Norm" do FRISCO, não são contemplados no UML.

Information system, Computerised information sub-system (CISS)

> *An **information system** is a sub-system of an organisational system, comprising the conception of how the communication- and information–oriented aspects of an organisation are composed (e.g. of specific communicating, information-providing and/or information-seeking actors, and of specific information-oriented actands) and how these operate, thus describing the (explicit and/or implicit) communication-oriented and information–providing actions and arrangements existing within that organisation.*
>
> *A **computerised information sub-system** is a sub-system of an information system, whereby all actions within that sub-system are performed by one or several computer(s).*
>
> [FRISCO98]

Não se encontram no UML termos ou conceitos que claramente satisfaçam ou vão ao encontro destas definições de "Information system" e de "Computerised information sub-system (CISS)" do FRISCO. A título de curiosidade, as únicas referências a "sistema de informação" no documento de especificação, vem no prefácio do documento aquando da apresentação do OMG, e na apresentação dos precursores do UML:

> *Preface*
>
> *About the Object Management Group (OMG)*
>
> *The Object Management Group, Inc. (OMG) is an international organization supported by over 800 members, including information system vendors, software developers and users.*
>
> [OMG-UML01, pág. xxi]
>
> *UML 0.8 - 0.91(1.6.1)*
>
> *Precursors to UML (1.6.1.1)*
>
> *(…)OMT-2 was especially expressive for analysis and data-intensive information systems.*
>
> [OMG-UML01, pág. 1-11]

No documento de especificação do UML pode-se assumir o conceito de "sistema de informação"[65] e de "sub-sistema de informação computorizado" diluídos no mesmo conceito de "sistema físico"[66]:

> *physical system* *1. The subject of a model.*
>
> *2. A collection of connected physical units, which can include software, hardware and people, that are organized to accomplish a specific purpose. A physical system can be described by one or more models, possibly from different viewpoints.*
>
> [OMG-UML01, pág. B-15]

Este "sistema físico" (que poderá ser visto como um sistema de informação) é o que tido no UML como objecto de modelação e de construção aquando da modelação de um sistema; todos os elementos que compõem a linguagem UML foram criados com o objectivo de facilitar a modelação destes sistemas, os quais são essencialmente caracterizados por serem "*software intensive*".

> *Scope of the UML (1.5)*
>
> *The Unified Modeling Language (UML) is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system.*

[65] Ver nota de rodapé nº 59, a qual afirma que se pode, no contexto do UML, considerar os termos "System" e Information System" com o mesmo significado.

[66] Definição apresentada neste documento aquando da análise ao conceito "System" do FRISCO (pág. 136)

[OMG-UML01, pág. 1-6]

Pode-se naturalmente colocar a questão: se por parte do FRISCO existem os conceitos "information system" e "computerised information sub-system", e se por parte do UML existe o termo "physical system", não de poderia estabelecer alguma correspondência entre eles? Atendendo a que num "CISS" do FRISCO todas as acções são desenvolvidas por "computadores", e num "sistema físico" do UML, são consideradas, para além dos computadores, as pessoas, pode-se tentar visualizar abrangência dos conceitos:

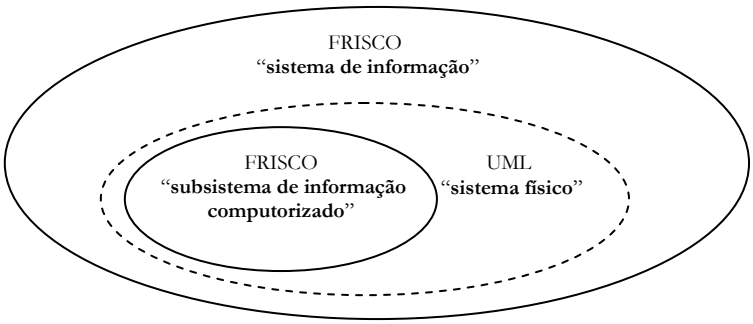

Figura A.42 – relação entre noções de sistemas do FRISCO e do UML

A partir destas abrangências, podem-se tirar ilações sobre as correspondências: (1) não se pode dizer que qualquer um dos conceitos do FRISCO é "equivalente" a "sistema físico" do UML; (2) dizer que existe uma proximidade entre "subsistema de informação computorizado" e "sistema físico" do UML, ou entre "sistema de informação" do FRISCO e "sistema físico" do UML, depende de alguma subjectividade de carácter pessoal; (3) pelo facto do UML não ter qualquer consideração válida relativa a sistemas de informação, e, por se afirmar ser uma linguagem para modelar essencialmente "…*a software intensive system*"[OMG-UML01, pág 1-6], tende-se a assumir que o "sistema físico" do UML está mais próximo do conceito de "subsistema de informação computorizado" do FRISCO.

B. Pressupostos e Conceitos do FRISCO

B.1. Pressupostos

Pressuposto [a]:
O "mundo" existe, independentemente da nossa própria existência, ou das nossas capacidades cognitivas e intelectuais.

Pressuposto [b]:
Os seres humanos são capazes de observar e de perceber "partes" ou "aspectos" do "mundo" (às quais chamamos de domínio) com os seus sentidos, formando portanto percepções nas suas mentes. As percepções podem ser consideradas como padrões específicos, geralmente alterando-se com o tempo.

Pressuposto [c]:
Os seres humanos são capazes de formar concepções nas suas mentes, como resultado de percepção actual ou passada de vários processos cognitivos ou intelectuais, tais como reconhecimento, caracterização, abstracção, derivação e/ou reflexão interna. A colecção de concepções (relativamente) estáveis e (suficientemente) consistentes na mente de uma pessoa é chamado de seu conhecimento.

Pressuposto [d]:
Um domínio percebido pode ser concebido como composto por componentes identificáveis, aos quais chamamos de coisas; elas podem sobrepor-se, conterem-se umas às outras ou relacionar-se entre si de qualquer forma empírica.

Pressuposto [e]:
Algumas coisas são concebidas como tendo existência estática (estados), enquanto que outras são concebidas como alterações de alguns estados (transições). Assim, um domínio percebido pode ser concebido como tendo uma existência num contexto temporal.

Pressuposto [f]:
Algumas transições podem ser concebidas como sendo efectuadas ou provocadas por algumas coisas activas, chamadas de actores. Uma tal transição, chamada de acção, é efectuada por esse actor em coisas passivas, chamadas de actands. Uma acção racional por um actor é dita ser em procura de um objectivo. É possível que a causa de uma transição não possa ser atribuída a um actor específico, mas meramente surge devido a, ou é despoletada por, algum aspecto do estado precedente.

Pressuposto [g]:
As pessoas usam representações para comunicar as suas concepções. Essas concepções são representadas em alguma linguagem em algum meio.

Pressuposto [h]:
Para uma sociedade, é de interesse principal que qualquer domínio seja representado de forma que que exista consenso sobre este. Sem reinvidicar a presumida "existência" em qualquer sentido "mundo real" do termo, as representações resultantes com as quais toda a gente concorda, será chamada de "realidade inter-subjectiva".

Pressuposto [i]:
Podem-se formar concepções estáveis na mente de uma pessoa – como um resultado de observar o estado de alguma parte do mundo concebido, ou as alterações de tal estado, ou como resultado da comunicação com outras em relação àquele. Qualquer tal concepção pode ser chamada de conhecimento sobre o mundo. A existência assumida da concepção (mental) é uma metáfora para a experiência de pessoas quando recebem comunicação significativa.

Pressuposto [j]:
As pessoas observam, experienciam e discutem representações a vários níveis semióticos, isto é, eles focam, com variada ênfase: os aspectos físicos e empíricos de uma representação, as leis para expressar tal representação, o seu significado, o seu uso e efeitos, e o ponto ao qual se pode concordar com e possivelmente derivar obrigações implícitas em qualquer afirmação sobre alguma coisa.

Pressuposto [k]:
Os actores humanos podem ter em suas mentes modelos, concepções claras, precisas e não ambíguas, os quais podem expressar na forma de denotação de modelo, numa linguagem formal ou semi-formal apropriada.

Pressuposto [l]:
Aquando a observação de algum domínio, os actores humanos frequentemente concebem coerência nesse domínio, ou afirmam a presença de tal corência. O modelo resultante, feito de modelos coerentes, é chamado de sistema. Qualquer coisa fora desse domínio será considerado como pertencendo ao ambiente do sistema.

Pressuposto [m]:
As organizações são caracterizadas por arranjos (geralmente orientados a objectivo) entre várias partes, pelas quais é possível a acção (maioritariamente co-operativa e coordenada) através de comunicação (frequentemente sistemática), a qual ocasiona a troca de informação, envolvendo mensagens significativamente compostas.

Pressuposto [n]:
Uma organização é um agrupamento de actores, juntamente com uma colecção de actands, tal que (a) é perseguido um objecivo comum ou seja mostrada alguma outra características de coerência, e (b) ocorrem interacções baseadas em comunicação. A informação é usada numa organização no contexto do seu funcionamento (acções), não só internamente mas também em relação ao seu ambiente, a sociedade no geral. Devido a este envolvimento social, as normas são directivas significativas para indivíduos na organização e a organização como um todo.

Pressuposto [o]:
O uso de informação organizacional deriva da sua prática operacional, a qual pode estar presente na forma do seu sistema organizacional (pelo "modelo de negócio"); dentro deste, pode-se encontrar embebido o que pode ser descrito como uma rede complexa de fluxos de informação que suporta as suas acções; alguns dos fluxos de informação podem ser arranjados dentro de um subsistema de informação computorizado.

B.2. Apresentação Informal dos Conceitos

Definição	Significado
Definition E1: **Thing**	A **thing** is any part of a conception of a domain (being itself a "part" or "aspect" of the "world"). The set of all things under consideration is the conception of that domain.
Definition E2: **Predicator, Predicated thing**	A **predicator** is a thing, used to characterise or qualify other things, and assumed as being "atomic", "undividable" or "elementary".
	A **predicated thing** is a thing being characterised or qualified by at least one predicator.
Definition E3: **Relationship**	A **relationship** is a special thing composed of one or several predicated thing(s), each one associated with one predicator characterising the role of that predicated thing within that relationship.
Definition E4: **Set membership, Elementary thing, Composite thing**	A **set membership** is a special binary relationship between a thing (the set), characterised by the special predicator called 'has-element', and another thing, characterised by the special predicator called 'is-element-of'. An **elementary thing** is a thing, not being a relationship and not being characterised by the special predicator called 'has-element'. A **composite thing** is a thing, not being an elementary thing.
Definition E5: **Entity**	An **entity** is a predicated thing as well as an elementary thing.
Definition E6: **Type, Population, Instance**	A **type** of things is a specific characterisation (e.g. a predicate) applying to all things of that type. A **population** of a type of things is a set of things, each one fulfilling the characterisation determining that type An **instance** of a type of things is an element of a population of that type.
Definition E7: **Transition, State, Pre-state, Post-state**	A **transition** is a special binary relationship between two (partially or totally) different composite things, called the pre-state and the post-state of that transition, whereby at least one thing is element of the pre-state, but not of the post-state, or vice versa. A **state** is a composite thing, involved as pre-state or as post-state in some transition. No element of a state may be a transition, itself. The **pre-state** of a transition is the state valid before that transition, and is characterised by the special predicator 'before'. The **post-state** of a transition is the state valid after that transition, and is characterised by the special predicator 'after'.

	Shorthand notation: transition: pre-state => post-state
Definition E8: **State-transition structure**	Given are the transitions t_x: s_1 => s_2 and t_y: s_3 => s_4. The following basic **state-transition structure**s exist in this case: Sequence: sequ (tx, ty) is a sequence of transitions if s3 is a subset of s2. The resulting state-transition structure has s_1 as pre-state and s_4 as post-state. Longer sequences are defined as follows: sequ (t_x, t_y, t_z) follows from sequ (t_x, t_y) and sequ (t_y, t_z) **(2) Choice:** choice (t_x, t_y) is a choice of transitions if the intersection of s_1 and s_3 is not empty. The result is either transition t_x or t_y, but not both. **(3) Concurrency:** concur (t_x, t_y) are concurrent transitions if the intersection of s_1 and s_3 is empty. The result is (s_1 union s_3) => (s_2 union s_4).
Definition E9: **Composite transition**	A **composite transition** is a state-transition structure with a unique pre-state and a unique post-state.
Definition E10: **Transition occurrence**	A **transition occurrence** is a specific occurrence of a transition. A set of transition occurrences is subject to strict partial ordering.
Definition E11: **Relative time, Absolute time**	The strict partial order imposed on the sets of all transition occurrences is called **relative time**. **Absolute time** may be determined by a clock that issues (assumedly) regular pulses (transition occurrences of the clock, or clock events). An absolute time value may be assigned to some transition occurrence, by comparing that transition occurrence with the successive (absolute-time-determining) clock events. A strictly ordered set of time points can be defined on the basis of these clock events, called time axis.
Definition E12: **Rule**	A **rule** determines a set of permissible states and transitions in a specific context. In other terms, a rule governs a non-empty set of types of things by determining their permissible populations
Definition E13: **Actor**	An **actor** is a special thing conceived as being "responsible" or "responsive" and as being able to "cause" transitions, and is therefore part of their pre-states, and, if not "destroyed" or "consumed" by the transitions, also part of their post-states.
Definition E14: **Action, Composite action, Action occurrence, Co-action**	An **action** is a transition involving a non-empty set of actors in its pre-state, and, if not "destroyed" or "consumed" by the action, in its post-state as well, and involving a non-empty or empty set of other things (actands) as part of its pre–state, and having a non-empty or empty set of other things (actands) in its post–state. A **composite action** is a composite transition with the same conditions as applying for the notion of action. An **action occurrence** is a transition occurrence with the same conditions as applying for the notion of action.

	A **co-action** is a special action performed by more than one actor in a co-ordinated way, pursuing a common goal.
Definition E15: **Actand, Input actand, Output actand, Resource**	An **actand** is a thing involved in the pre-state or post-state of an action, not considered as an actor for that action. An **input actand** is a part of the pre-state of an action, excluding the actors. An **output actand** is a part of the post-state of an action, excluding the actors. The pre-state of an action, i.e. the union of the set of actors and the set of input actands of that action, is called its **resource**s.
Definition E16: **Action context**	The **action context** of an action is a special, optional part of the pre-state of that action, qualifying the context or situation in which that action is performed, and determining or modifying at least one of its output actands.
Definition E17: **Goal, Goal-pursuing actor**	The **goal** of an action is a special input actand of that action, pursued by the actors of that action and stating the desired output state intensionally. A **goal-pursuing actor** is an actor performing an action, who deliberately aims at a specific goal when involved in that action.
Definition E18: **Domain, Domain component, Domain environment**	A **domain** comprises any "part" or "aspect" of the "world" under consideration. A **domain component** is any "part" or "aspect" of that domain. A **domain environment** is the "world" without that domain.
Definition E19: **Human actor, Perception, Perceiving action, Perceiver**	A **human actor** is a responsible actor with the capabilities and liabilities of a normal human being, in particular capable of performing perceiving actions, conceiving actions and representing actions. A **perception** is a special actand resulting from an action whereby a human actor observes a domain with his senses, and forms a specific (static, non-time-varying, or dynamic, time-varying) pattern of visual, auditory or other sensations of it in his mind. A **perceiving action** is a special action of a human actor having a domain as input actand and a perception as output actand. A **perceiver** is a human actor involved in a perceiving action.
Definition E20: **Conception, Conceiving action, Conceiver, Conceiving context**	A **conception** is a special actand resulting from an action whereby a human actor aims at interpreting a perception in his mind, possibly in a specific action context.

	A **conceiving action** is a special action of a human actor having a perception and possibly some action context as input actand(s) and a conception as output actand. A **conceiver** is a human actor involved in a conceiving action. A **conceiving context** is the action context of a conceiving action.
Definition E21: **Interpreting action, Interpreter, Interpreting context**	An **interpreting action** is the sequence of a perceiving action performed on a domain, resulting in a perception of that domain, followed by a conceiving action performed on that perception, resulting in a conception. An **interpreter** is a human actor performing an interpreting action. An **interpreting context** is the action context of an interpreting action.
Definition E22: **Symbol, Alphabet, Symbolic construct, Language**	A **symbol** is a special entity used as an undividable element of a representation in a language. An **alphabet** of a language is a non-empty and finite set of symbols. A **symbolic construct** is a non-empty and finite "arrangement" of symbols taken from an alphabet. In the one-dimensional case, an arrangement is just a sequence of symbols (a "sentence"). In the n-dimensional case (n>1), it may be any arrangement of its constituting symbols in the n-dimensional space. Provided one considers the elements of arrangement (such as sequence) belonging to the alphabet, a symbolic construct is a non-empty and finite set of symbols. A **language** is a non-empty set of permissible symbolic constructs. The permissible symbolic constructs in a language are determined either extensionally by enumeration or intensionally by a set of rules. The rules of a language may be syntactic ("grammar") as well as semantic ("semantic rules").
Definition E23: **Representation, Representing action, Representer, Representing context**	A **representation** is a special actand describing some conception(s) in a language, resulting from an action whereby a human actor aims at describing his conception(s), possibly in a specific action context. A **representing action** is a special action of a human actor having a conception and possibly some action context as input actand(s) and a representation as output actand. A **representer** is a human actor involved in a representing action. A **representing context** is the action context of a representing action. **Definition E24: Label, Reference** A **label** is a special entity being an elementary representation and used for referring to some conception in an elementary way. A **reference** is a special binary relationship between a conception and a representation used to refer to that conception.

Definition E25: **Semiotic level**	The **semiotic level** of a representation is the aspect considered in representing it. The semiotic levels are: physical, empirical, syntactical, semantical, pragmatic, and social.
Definition E26: **Model, Model denotation**	A **model** is a purposely abstracted, clear, precise and unambiguous conception. A **model denotation** is a precise and unambiguous representation of a model, in some appropriate formal or semi-formal language.
Definition E27: **Modelling action, Modeller**	A **modelling action** is the sequence of a perceiving action performed on some domain, followed by a conceiving action on that perception, resulting in a model, and followed by a representing action on that model, resulting in a model denotation. A **modeller** is a human actor performing a modelling action.
Definition E28: **Intensional model, Extensional model**	An **intensional model** is that part of a model comprising the possibilities and necessities of a domain only, i.e. the types and rules. An **extensional model** is that part of a model containing a specific population of the types in the corresponding intensional model, whereby this population must obey all rules determined in that intensional model.
Definition E29: **Meta-model**	A **meta-model** is a model of the conceptual foundation of a language, consisting of a set of basic concepts, and a set of rules determining the set of possible models denotable in that language.
Definition E30: **System, System denotation, System component, System environment, System viewer, System representer**	A **system** is a special model, whereby all the things contained in that model (all the system components) are transitively coherent, i.e. all of them are directly or indirectly related to each other form a *coherent* whole. A system is conceived as having assigned to it, as a whole, a specific characterisation (the so-called "*systemic properties*"). A **system denotation** is a precise and unambiguous representation of a system. A **system component** is a non-empty set of things being contained in that system. The **system environment** of a system is the set of all things not being contained in that system. A **system viewer** is a human actor perceiving and conceiving a domain as a system. A system viewer recognises the system, by its distinction from the system environment, by its coherence, and because of its systemic properties. A **system representer** is a human actor representing a system in some language.

Definition E31: **Dynamic system, Static system, Active system, Passive System, Open system, Closed System**	A **dynamic system** is conceived as capable of undergoing change, i.e. some of the system components are transitions. A system that is not dynamic is called a **static system**. An **active system** is conceived as capable of doing something, i.e. some of the system components are actors performing actions on actands. A system is called a **passive system** if it is not active. An **open system** is conceived as one which may respond to external messages or triggers, i.e. there may be transitions within the system due external causes coming from the system environment. A system is called a **closed system** if the system environment cannot cause any transitions within the system.
Definition E32: **Sub-system**	Any **sub–system** S' of a larger system S is a system, itself. The set of all sub-system components of S' is a proper sub-set of the set of all system components of S.
Definition E33: **Knowledge**	**Knowledge** is a relatively stable and sufficiently consistent set of conceptions possessed by single human actors.
Definition E34: **Data**	The term **data** denotes any set of representations of knowledge, expressed in a language.
Definition E35: **Message, Message transfer, Sender, Receiver**	A **message** is data, transmitted by one actor (the sender) via a channel (a medium), and intended for a non-empty set of other actors (the receivers). A **message transfer** is a sequence of actions, the sending action by the sender and the receiving actions by the receivers, whereby the input actand of the sending action is the message to be sent, whereby the output actand of the sending action, being (in the simplest case) equal to the input actand of the receiving action, is the message on the channel, and whereby the output actand of the receiving action is the message received. A **sender** is an actor sending a message. A **receiver** is an actor receiving a message.
Definition E36: **Information**	**Information** is the personal knowledge increment brought about by a receiving action in a message transfer, i.e. it is the difference between the conceptions interpreted from a received message and the personal knowledge before the receiving action.
Definition E37: **Communication**	**Communication** is an exchange of messages, i.e. a sequence of mutual and alternating message transfers between at least two human actors, called communication partners, whereby these messages represent some knowledge and are expressed in languages understood by all communication partners, and whereby some amount of knowledge about the domain of communication and about the action context and the goal of the communication is made present in all communication partners.

Definition E38: **Shared knowledge**	**Shared knowledge** is that knowledge of the individuals in a group of human actors, which they assume to be identical (or at least similar) to that of the others, as resulting from the negotiation process implicit in some communication.
Definition E39: **Organisational system, Norm**	An **organisational system** is a special kind of system, being normally dynamic, active and open, and comprising the conception of how an organisation is composed (i.e. of specific actors and actands) and how it operates (i.e. performing specific actions in pursuit of organisational goals, guided by organisational rules and informed by internal and external communication), where its systemic properties are that it responds to (certain kinds of) changes caused by the system environment and, itself, causes (certain kinds of) changes in the system environment. **Norm**s are socially agreed rules affecting and to a large extent directing the actions within an organisational system.
Definition E40: **Information system, Information system denotation**	An **information system** is a sub-system of an organisational system, comprising the conception of how the communication- and information–oriented aspects of an organisation are composed (e.g. of specific communicating, information-providing and/or information-seeking actors, and of specific information-oriented actands) and how these operate, thus describing the (explicit and/or implicit) communication-oriented and information–providing actions and arrangements existing within that organisation. An **information system denotation** is a precise and unambiguous representation of an information system.
Definition E41: **Computerised information sub-system (CISS)**	A **computerised information sub-system** is a sub-system of an information system, whereby all actions within that sub-system are performed by one or several computer(s).

B.3. Apresentação Formal dos Conceitos

Primitive P1: Thing [E1]

The set of all things is denoted by **Z**

Definition D1: Relationship [E3]

The set of all relationships is denoted by **R** .

R = {r ∈ **Z** | r ⊆ **U** ∧ **U** = {<q, p> | q, p ∈ **Z**} ∧ 1 = |r| < ∞}.

Definition D2: Predicator [E2]

The set of all predicators is denoted by **P** .

P = {p ∈ **Z** | ∃ u ∈ **U**, q ∈ **Z** [u = <q, p>]}.

Definition D3: Predicated thing [E2]

The set of all predicated things is denoted by **Q** .

Q = {q ∈ **Z** | ∃ u ∈ **U**, p ∈ **P** [u = <q, p>]}.

Primitives P2: Set membership predicators

'has-element' ∈ **P** and 'is-element-of' ∈ **P** are special predicators characterising a set and an element of a set in the context of a set membership, respectively.

Definition D4: Set membership [E4]

The set of all set memberships is denoted by **SM** .

SM = {sm ∈ **R** |

sm = {<q_1, has-element>, <q_2, is-element-of>} ∧ q_1, q_2 ∈ **Q** ∧ $q_1 \neq q_2$}.

Abbreviations:

- sm = ($q_2 \in q_1$);
- {q_2, ...} = q_1; or {q_2, ...} ⊆ q_1;
- $\wp(q_1)$ denotes the set of all subsets of q_1.
- $\wp_m(q_1)$ denotes the set of all sub-multi-sets of q_1.

Definition D5: Elementary thing [E4]

The set of all elementary things is denoted by **EZ .**

EZ = {ez ∈ Z \ **R** | ¬ ∃ u ∈ **U** [u = <ez, has-element>]}.

Axiom A1:

∀ p ∈ **P** [p ∈ **EZ**].

Definition D6: Entity [E5]

The set of all entities is denoted by **E** .

E = **EZ** ⋂ **Q**.

Definition D7: Composite thing [E4]

The set of all composite things is denoted by **CZ** .

CZ = **Z** \ **EZ.**

Function F1: Elementof

Let Elementof: **CZ** ∈ $\wp$(**Z**) be a function from composite things to sets of things,

where Elementof (cz) = {z ∈ **Z** | ∃ sm ∈ **SM**

[sm = {<cz, has-element>, <z, is-element-of>}]}

Function F2: Elementsof

Let Elementsof: **CZ** → $\wp$(**Z**) be a function from composite things to sets of things,

where Elementsof (cz) = ∅ if cz ∈ **EZ**,

else Elementsof (cz) = Elementof (cz) ⋃ {Elementsof (z) | z ∈ Elementof (cz)}.

Axiom A2:

∀cz ∈ **CZ** [cz ∉ Elementsof (cz)].

Function F3: Predthingin

Let Predthingin: **R** → $\wp$(**Q**) be a function from relationships to sets of predicated things,

where Predthingin (r) = {q ∈ **Q** | ∃ p ∈ **P** [<q, p> ∈ r]}.

Function F4: Predthingsin

Let Predthingsin: **Q** → $\wp$(**Q**) be a function from predicated things to sets of predicated things,

where Predthingsin (r) = ∅ if r ∈ **E**,

else Predthingsin (r) = Predthingin (r) ⋃ {Predthingsin (q) | q ∈ Predthingin (r)}.

Primitives P3: Transition predicators

'before' ∈ **P** and 'after' ∈ **P** denote special predicators for defining transitions, whereby the first one characterises the pre-state of a transition, and the second one the post-state.

Definition D8: Transition [E7]

The set of all transitions is denoted by **T** .

T = {t ∈ **R** | ∃ s_b, s_a ∈ **CZ** [t = {<s_b, before>, <s_a, after>} ∧ s_b ≠ s_a]}.

Abbreviation: t: s_b ⇒ s_a.

Definition D9: State [E7]

The set of all states is denoted by **S** .

S = {s ∈ **CZ** \ **T** | ∃ t ∈ **T** [s ∈ Predthingin (t)]}.

Function F5: Prestateof

Let Prestateof: **T** → **S** be a function from transitions to states,

where Prestateof (t: s_b ⇒ s_a) = s_b denotes the (pre-)state before the transition t.

Function F6: Poststateof

Let Poststateof: $\mathbf{T} \rightarrow \mathbf{S}$ be a function from transitions to states,
where Poststateof (t: $s_b \Rightarrow s_a$) = s_a denotes the (post-)state after the transition t.

Definition D10: State-transition structure [E8]

If t_x: $s_1 \Rightarrow s_2$, t_y: $s_3 \Rightarrow s_4$ are transitions, then the following basic state-transition structures exist:

(1) Sequence:

sequ (t_x, t_y) if $s_3 \subseteq s_2$.

The resulting state-transition structure has s_1 as pre-state and s_4 as post-state .

Longer sequences are defined as follows:

sequ (t_x, t_y, t_z) $\Leftarrow$ sequ (t_x, t_y) $\wedge$ sequ (t_y, t_z).

(2) Choice:

choice (t_x, t_y) if $s_1 \cap s_3 \neq \varnothing$. The result is either transition t_x or t_y, but not both.

(3) Concurrency:

concur (t_x, t_y) if $s_1 \cap s_3 = \varnothing$. The result is $(s_1 \cup s_3) \Rightarrow (s_2 \cup s_4)$.

Let **ST** denote the set of all state-transition structures formed by these rules.

Definition D11: Composite transition [E9]

The set of all composite transitions is denoted by **CT** .

$\mathbf{CT} = \mathbf{ST} \cap \mathbf{T}$.

Definition D12: Transition occurrence [E10]

(1) A transition t: $s_1 \Rightarrow s_2$ is enabled to occur in state s if $s_1 \subseteq s$.

(2) If a transition t: $s_1 \Rightarrow s_2$ occurs in state s,
then s is changed to the new state $s' = (s \setminus s_1) \cup s_2$.

The occurrence of a transition t: $s_1 \Rightarrow s_2$ in state s leading to state s' is denoted as: occ(t): $s \rightarrow s'$.

Definition D15: Type, Population, Instance [E6]

The set of all types of predicated or composite things is denoted by **TY** .

$\mathbf{TY} \subseteq \wp(\mathbf{Q} \cup \mathbf{CZ})$.

The set of all populations of types of predicated or composite things is denoted by **PO**.

$\mathbf{PO} = \{po \in \wp(\mathbf{Q} \cup \mathbf{CZ}) \mid \exists\ ty \in \mathbf{TY}\ [po \subseteq ty]\}$.

The set of all instances of types of predicated or composite things is denoted by **IN**.

$\mathbf{IN} = \{in \in \mathbf{Q} \cup \mathbf{CZ} \mid \exists\ ty \in \mathbf{TY}\ [in \in ty]\}$.

Definition D16: Rule [E12]

The set of all rules is denoted by **F** .

$\mathbf{F} \subseteq \wp(\mathbf{S}) \cup \wp(\mathbf{T})$.

Primitives P4: Actor-characterising predicators

Let 'performing' $\in$ **P** denote the predicator indicating the capability of a thing to bring about or perform a transition.

Let 'performed-by' $\in$ **P** denote the predicator indicating that a transition is performed by a thing.

Definition D17: Action [E14]

The set of all actions is denoted by **N** .

$\mathbf{N} = \{t \in \mathbf{T} \mid \exists\, q \in Q\ \exists\, r \in \mathbf{R}$

$[r = \{<q, \text{performing}>, <t, \text{performed-by}>\} \wedge q \in \text{Prestateof}(t)]\}$.

Definition D18: Composite action [E14]

The set of all composite actions is denoted by **CN** .

$\mathbf{CN} = \mathbf{ST} \cap \mathbf{N}$.

Function F7: Actorof

Let Actorof: $\mathbf{N} \rightarrow \wp(\mathbf{Q})$ be a function determining the actors performing an action,

where Actorof (n) = $\{q \in \mathbf{Q} \mid$

$q \in \text{Prestateof}(n) \wedge \exists\, r \in \mathbf{R}\ [r = \{<q, \text{performing}>, <n, \text{performed-by}>\}\}$.

Definition D19: Actor [E13]

The set of all actors is denoted by **O** .

$\mathbf{O} = \{q \in \mathbf{Q} \mid \exists\, n \in \mathbf{N}\ [q \in \text{Actorof}(n)]\}$.

Function F8: Inputof

Let Inputof: $\mathbf{N} \rightarrow \wp(\mathbf{Z})$ be a function determining the input actands of an action,

where Inputof (n) = $\{z \in \text{Prestateof}(n) \mid \neg (z \in \text{Actorof}(n))\}$.

Function F9: Outputof

Let Outputof: $\mathbf{N} \rightarrow \wp(\mathbf{Z})$ be a function determining the output actands of an action,

where Outputof (n) = $\{z \in \text{Poststateof}(n) \mid \neg (z \in \text{Actorof}(n))\}$.

Definition D20: Actand [E15]

The set of all actands is denoted by **D** .

$\mathbf{D} = \{z \in \mathbf{Z} \mid \exists\, n \in \mathbf{N}\ [z \in \text{Inputof}(n) \vee z \in \text{Outputof}(n)]\}$.

Definition D21: Resource [E15]

The set of all resources is denoted by **RS** .

$\mathbf{RS} = \{z \in \mathbf{Z} \mid \exists\, n \in \mathbf{N}\ [z \in \text{Prestateof}(n)]\}$.

Primitive P5: Context-characterising predicator

Let 'is-context' ∈ **P** denote the predicator indicating the characteristic of being a specific context.

Definition D22: Action context [E16]

The set of action contexts is denoted by **X** .

$\mathbf{X} = \{d \in \mathbf{D} \mid \exists\, n \in \mathbf{N}\ \exists\, r \in \mathbf{R}\ [r = \{<d, \text{is-context}>\} \wedge d \in \text{Inputof}(n)]\}$.

Primitives P6: Goal-pursuing predicators

Let 'pursued-by' ∈ **P** denote the predicator indicating the pursuit of a goal of an actor performing an action, to achieve the desired output state of that action.

Let 'pursuing' ∈ **P** denote the predicator indicating an actor pursuing a goal.

Definition D23: Goal [E17]

The set of goals is denoted by **G** .

G = {d ∈ **D** | ∃ n ∈ **N** ∃ o ∈ **O** ∃ r ∈ **R**
[r = {<d, pursued-by>, <o, pursuing>} ∧ d ∈ Inputof (n) ∧ o ∈ Actorof (n)]}.

Primitive P7: Human-characterising predicator

'is-human' ∈ **P** denotes the predicator characterising any human being or any being with equivalent capabilities and liabilities.

Primitive P8: Domain-characterising predicator

'is-domain' ∈ **P** denotes the predicator characterising any domain.

Primitive P9: Perception-characterising predicator

'is-perception' ∈ **P** denotes the predicator characterising any perception.

Primitive P10: Conception-characterising predicator

'is-conception' ∈ **P** denotes the predicator characterising any conception.

Primitive P11: Representation-characterising predicator

'is-repres' ∈ **P** denotes the predicator characterising any representation.

Definition D24: Human actor [E19]

OH = {o ∈ **O** | ∃ r ∈ **R** [r = {<o, is-human>}]} denotes the set of all human actors.

Definition D25: Domain, Perception, Perceiving action, Perceiver [E19]

DD = {dd ∈ **D** | ∃ r ∈ **R** [r = {<dd, is-domain>}]} denotes the set of all domains.

DP = {dp ∈ **D** | ∃ r ∈ **R** [r = {<dp, is-perception>}]} denotes the set of all perceptions.

NP = {n ∈ **N** | ∃! dd ∈ **DD** ∃! dp ∈ **DP** [dd = Inputof (n) ∧ dp = Outputof (n)]}
denotes the set of all perceiving actions.

OP = {o ∈ **OH** | ∃ n ∈ **NP** [o ∈ Actorof (n)]} denotes the set of all perceivers.

Definition D26: Conception, Conceiving action, Conceiver, Conceiving context [E20]

DC = {dc ∈ **D** | ∃ r ∈ **R** [r = {<dc, is-conception>}]} denotes the set of all conceptions.

NC = {n ∈ **N** | ∃! dp ∈ **DP** ∃! dc ∈ **DC** [dp = Inputof (n) ∧ dc = Outputof (n)]}
denotes the set of all conceiving actions.

OC = {o ∈ **OH** | ∃ n ∈ **NC** [o ∈ Actorof (n)]} denotes the set of all conceivers.

XC = {x ∈ **X** | ∃ n ∈ **NC** [x ∈ Inputof (n)]} denotes the set of all conceiving contexts.

Definition D27: Interpreting action, Interpreter, Interpreting context [E21]

NI denotes the set of all interpreting actions.

NI = {n ∈ **N** | ∃ np ∈ **NP** ∃ nc ∈ **NC** [n = sequ (np, nc) ∧
Inputof (n) = Inputof (np) ∧ Outputof (np) = Inputof (nc) ∧
Outputof (nc) = Outputof (n) ∧
Actorof (n) = Actorof (np) = Actorof (nc)]}.

OI = {o ∈ **OH** | ∃ n ∈ **NI** [o ∈ Actorof (n)]} denotes the set of all interpreters.

XI = {x ∈ **X** | ∃ n ∈ **NI** [x ∈ Inputof (n)]} denotes the set of all interpreting contexts.

Definition D28: Representation [E23], Symbol, Alphabet, Symbolic construct, Language [E22]

DR = {dr ∈ **D** | ∃ r ∈ **R** [r = {<dr, is-repres>}} denotes the set of all representations.

SB = **E** ⋂ **DR** denotes the set of all symbols.

AB = {ab ∈ $\wp$(**SB**) | 1 = |ab| < ∞} denotes the set of all alphabets.

SC = {sc ∈ $\wp_m$(ab) | ab ∈ **AB** ∧ 1 = |sc| < ∞} denotes the set of all symbolic constructs.

L denotes the set of all languages.

L = {l ∈ $\wp$(**SC**) | ∀ sc ∈ l ∃ F ⊆ **F** [sc |= F ∧ ∃! ab ∈ **AB** [sc ∈ $\wp$(ab)]]} \ ∅ .

The permissible symbolic constructs of a language can be determined in two ways:

(a) by enumerating all of them (determination by extension), or

(b) by defining a set of rules F ⊆ ***F*** on the set of symbolic constructs l ∈ $\wp$(**SC**), where each f (sc ∈ l) ∈ F holds iff sc is a permissible symbolic construct (determination by intension).

Primitives P12: Reference predicators

'referred-to-by' ∈ **P** denotes the predicator characterising a conception within a reference.

'referring-to' ∈ **P** denotes the predicator characterising a representation within a reference.

Definition D29: Representing action, Representer, Representing context [E23], Reference, Label [E24]

NR denotes the set of all representing actions.

NR = {n ∈ **N** | ∃! dc ∈ **DC** ∃! l ∈ **L** ∃! dr ∈ **DR**
[dc = Inputof (n) ∧ l = Inputof (n) ∧ dr = Outputof (n)]}.

OR = {o ∈ **OH** | ∃ n ∈ **NR** [o ∈ Actorof (n)]} denotes the set of all representers.

XR = {x ∈ **X** | ∃ n ∈ **NR** [x ∈ Inputof (n)]} denotes the set of all representing contexts.

RF denotes the set of all references.

RF = {rf ∈ **R** | ∃ dc ∈ **DC** ∃ dr ∈ **DR**
[rf = {<dc, referred-to-by>, <dr, referring-to>}]}.

LB denotes the set of all labels.

LB = {lb ∈ **E** ⋂ **DR** | ∃ rf ∈ **RF** ∃ dc ∈ **DC**
[rf = {<dc, referred-to-by>, <lb, referring-to>}]}.

Primitive P13: Model-characterising predicator

'is-model' ∈ **P** denotes the predicator characterising the features of being clear, precise, unambiguous and purposely abstracted.

Definition D30: Model [E26]

MC = {d ∈ **DC** | ∃ r ∈ **R** [r = {<d, is-model>}]} denotes the set of all models.

Function F10: Concof

Let Concof: **DR→DC** be a function determining the conceptions behind representations.

where Concof (dr) = (dc ∈ **DC** | ∃ n ∈ **NR** [dc = Inputof (n) ∧ dr = Outputof (n)]).

Definition D31: Model denotation [E26]

MD denotes the set of all model denotations.

MD = {md ∈ **DR** | ∃ mc ∈ **MC** [mc = Concof (md)]}.

Definition D32: Modelling action, Modeller [E27]

MN denotes the set of all modelling actions.

MN = {n ∈ **N** | ∃ np ∈ **NP** ∃ nc ∈ **NC** ∃ nr ∈ **NR** [n = sequ (np, nc, nr) ∧
Inputof (n) = Inputof (np) ∧ Outputof (np) = Inputof (nc) ∧
Outputof (nc) = Inputof (nr) ∈ **MC** ∧ Outputof (nr) = Outputof (n) ∈ **MD** ∧
Actorof (n) = Actorof (np) = Actorof (nc) = Actorof (nr)]}.

MOH = {o ∈ **OH** | ∃ n ∈ **MN** [o ∈ Actorof (n)]} denotes the set of all modellers.

Function F11: Knowlof

Let Knowlof: **OH**→**DC** be a function determining the knowledge of a human actor,
where Knowlof (oh) = {k ∈ ℘(**DC**) |
∀ ke ∈ Elementsof (k) ∃ nc ∈ **NC** [ke = Output (nc) ∧ oh = Actorof (nc)]}.

Definition D33: Knowledge [E33]

K denotes the set of all (personal) knowledge.

K = {k ∈ ℘(**DC**) | ∃ oh ∈ **OH** [k = Knowlof (oh)]}.

Definition D34: Data [E34]

DT ⊆ ℘(**DR**) denotes the set of all data.

Primitives P14: Message-related predicators

'sending' ∈ **P** denotes the predicator characterising any sending action.

'receiving' ∈ **P** denotes the predicator characterising any receiving action.

Definition D35: Sending action, Receiving action, Message transfer, Sender, Receiver, Message [E35]

MNS denotes the set of all sending actions.

MNS = {mns ∈ **N** | ∃ r ∈ **R** [r = {<mns, sending>}] ∧
Inputof (mns) ∈ **DT** ∧ Outputof (mns) ∈ **DT** ∧
∀ i ∈ Inputof (mns) ∀ o ∈ Outputof (mns) [i, o ∈ **DT**]}.

MNR denotes the set of all receiving actions.

MNR = {mnr ∈ **N** | ∃ r ∈ **R** [r = {<mnr, receiving>}] ∧
Inputof (mnr) ∈ **DT** ∧ Outputof (mnr) ∈ **DT** ∧
∀ i ∈ Inputof (mnr) ∀ o ∈ Outputof (mnr) [i, o ∈ **DT**]}.

MNT denotes the set of all message transfers.

MNT = {mnt ∈ **N** | ∃ mns ∈ **MNS**, mnr ∈ **MNR** [mnt = sequ (mns, mnr) ∧
Inputof (mnt) = Inputof (mns) ∧Outputof (mns) = Inputof (mnr) ∧
Outputof (mnr) = Outputof (mnt) ∧ Actorof (mns) ≠ Actorof (mnr)]}.

MOS = {o ∈ **O** | ∃ mns ∈ **MNS** [o = Actorof (mns)]} denotes the set of all senders.

MOR = {o ∈ **O** | ∃ mnr ∈ **MNR** [o = Actorof (mnr)]} denotes the set of all receivers.

MDS denotes the set of all messages to be sent.

MDS = {dt ∈ **DT** | ∃ mnt ∈ **MNT** [dt = Inputof (mnt)]}.

MDC denotes the set of all messages on a channel (between sender and receiver).

MDC = {dt ∈ **DT** | ∃ mns ∈ **MNS** ∃ mnr ∈ **MNR**
[dt = Outputof (mns) ∧ dt = Inputof (mnr)]}.

MDR denotes the set of all messages received.

MDR = {dt ∈ **DT** | ∃ mnt ∈ **MNT** [dt = Outputof (mnt)]}.

MES = **MDS** ⋃ **MDC** ⋃ **MDR** denotes the set of all messages.

Definition D36: Information [E36]

I denotes the set of all information.

$I = \{i \in \wp(DC) \mid \exists\, mnr \in \mathbf{MNR}\ \exists\, oh \in \mathbf{OH}$
$[i = Concof(Outputof(mnr)) \setminus Knowlof(oh) \wedge oh = Actorof(mnr)]\}$

Definition D39: Coherence of a set of relationships

RC denotes the set of all sets of directly coherent relationships.

$\mathbf{RC} = \{rc \in \wp(\mathbf{R}) \mid \exists\, rp \in \wp(\mathbf{R} \times \mathbf{R})\ [\forall <r_x, r_y> \in rp\ [r_x \in rc \wedge r_y \in rc \wedge r_x \neq r_y \wedge$
$Predthingsin(r_x) \cap Predthingsin(r_y) \neq \varnothing]]\}$.

RTC denotes the set of all sets of transitively coherent relationships.

$\mathbf{RTC} = \{rtc \in \wp(\mathbf{R}) \mid \exists\, rc \in \mathbf{RC}\ [(<r_x, r_y> \in rc \wedge <r_y, r_z> \in rc) \Rightarrow <r_x, r_z> \in rtc \wedge$
$(<r_x, r_y> \in rtc \wedge <r_y, r_z> \in rtc) \Rightarrow <r_x, r_z> \in rtc]\}$.

Definition D40: System [E30]

SY denotes the set of all systems.

$\mathbf{SY} = \{sy \in \mathbf{MC}\} \wedge$
$\{sy \in \mathbf{RTC} \mid \exists\, r_{sp} \in \mathbf{R}\ [sy \in Predthingsin(r_{sp}) \wedge r_{sp} \notin sy] \wedge 1 < |sy| \leq \infty\}$.

Definition D41: Sub-system [E32]

SS denotes the set of all system-sub-system pairs.

$\mathbf{SS} = \{<sy_b, sy_p> \in \mathbf{SY} \times \mathbf{SY} \mid Elementsof(sy_b) \subset Elementsof(sy_p)\}$.

Definition D42: Dynamic System, Static system [E31]

SYD denotes the set of all dynamic systems.

$\mathbf{SYD} = \{sy \in SY \mid \exists\, t \in T\ [t \in Elementsof(sy)]\}$.

SYS denotes the set of all static systems.

$\mathbf{SYS} = \mathbf{SY} \setminus \mathbf{SYD}$.

Definition D43: Active system, Passive system [E31]

SYA denotes the set of all active systems.

$\mathbf{SYA} = \{sy \in \mathbf{SY} \mid \exists\, n \in \mathbf{N}\ [n \in Elementsof(sy)]\}$.

SYP denotes the set of all passive systems.

$\mathbf{SYP} = \mathbf{SY} \setminus \mathbf{SYA}$.

Definition D44: Open system, Closed system [E31]

SYO denotes the set of all open systems.

$\mathbf{SYO} = \{sy \in \mathbf{SY} \mid \exists\, t_e, t_i \in \mathbf{T}$
$[sequ(t_e, t_i) \wedge Prestateof(t_i) \subseteq Poststateof(t_e) \wedge t_i \in sy \wedge t_e \notin sy]\}$.

SYC denotes the set of all closed systems.

$\mathbf{SYC} = \mathbf{SY} \setminus \mathbf{SYO}$.

Definition D45: Organisational system [E39]

OS denotes the set of all organisational systems.

$\mathbf{OS} = \{os \in \mathbf{SYO} \cap \mathbf{SYD} \cap \mathbf{SYA}) \mid Elementsof(os) \subseteq (\mathbf{O} \cup \mathbf{N} \cup \mathbf{D}) \wedge \exists\, g \in \mathbf{G}\ \exists\, r_{sp} \in \mathbf{R}$
$[r_{sp} = \{<(Elementsof(os) \cap \mathbf{O}), pursuing>, <g, pursued\text{-}by>\}]\}$.

Definition D46: Knowledge/data-processing action

NKD denotes the set of all knowledge- and data-processing actions.

$\mathbf{NKD} = \{n \in \mathbf{N} \mid \exists!\, d_{ik}, d_{ok} \in (\mathbf{K} \cup \mathbf{DT})\ [d_{ik} = Inputof(n) \wedge d_{ok} = Outputof(n)]\}$.

NDT denotes the set of all data-processing actions.

$\mathbf{NDT} = \{n \in \mathbf{N} \mid \exists!\, d_{ik}, d_{ok} \in (\mathbf{DT})\ [d_{ik} = Inputof(n) \wedge d_{ok} = Outputof(n)]\}$.

Definition D47: Information system [E40]

IS denotes the set of all information systems.

$$\mathbf{IS} = \{is \in (\mathbf{SYO} \cap \mathbf{SYA}) \mid \text{Elementsof}(is) \subseteq (\mathbf{O} \cup \mathbf{NKD} \cup \mathbf{DT} \cup \mathbf{K}) \wedge \exists g \in \mathbf{G}\ \exists r_{sp} \in \mathbf{R}\ [r_{sp} = \{<(\text{Elementsof}(is) \cap \mathbf{O}), \text{pursuing}>, <g, \text{pursued-by}>\}]\}.$$

Primitive P15: Computer-characterising predicator

'is-comp' ∈ **P** denotes the predicator characterising an actor as a computer.

Definition D48: Computerised actor

$$\mathbf{OCO} = \{o \in \mathbf{O} \setminus \mathbf{OH} \mid \exists r \in \mathbf{R}\ [r = \{<o, \text{is-comp}>\}]\}$$

denotes the set of all computerised actors.

Definition D49: Computerised information sub-system [E41]

$$\mathbf{CIS} = \{cis \in \mathbf{IS} \mid \text{Elementsof}(cis) \subseteq (\mathbf{OCO} \cup \mathbf{NDT} \cup \mathbf{DT})\}$$

denotes the set of all computerised information sub-systems.

C. Pacotes e Elementos do Metamodelo do UML

Fazendo-se uma breve introdução aos objectivos de cada um dos pacotes[67], apresenta-se nesta secção a sintaxe abstracta que se encontra nos vários pacotes que organizam a linguagem.

C.1. Pacote "Foundation"

O pacote "*Foundation*" é a infra-estrutura da linguagem que especifica a estrutura estática dos modelos, sendo decomposto nos pacotes "*Core*", "*Data Types*", e "*Extension Mechanisms*" (Figura C.1).

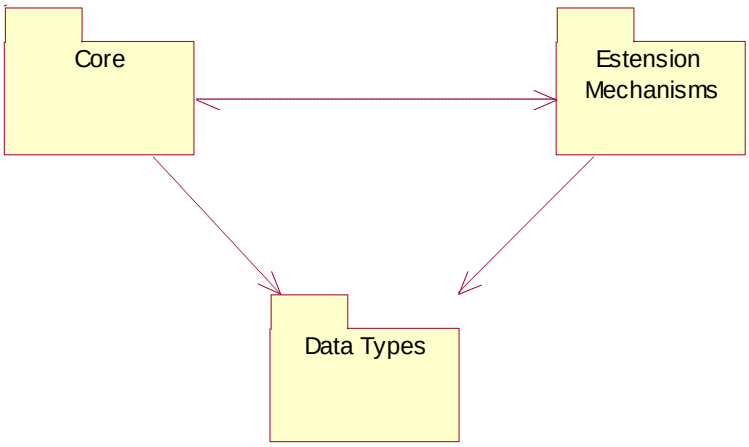

Figura C.1 – Pacotes contidos no pacote "*Foundation*"

Pacote "Core"

O pacote Core especifica os conceitos básicos necessários para um metamodelo elementar e define um *backbone* arquitectural para anexar construções de linguagem adicionais, tais como metaclasses, metaaasociações, e metaatributos.

Sintaxe Abstracta

A sintaxe abstracta do pacote Core é expressa em notação gráfica nas figuras que se seguem, e que mostram os:

- Elementos de modelo que definem o backbone estrutural do metamodelo (Figura C.2)
- Elementos de modelo que definem relacionamentos (Figura C.3)

[67] Todas as figuras apresentadas neste anexo são extraídas ou adaptadas do metamodelo apresentado no documento de especificação do UML [OMG-UML01]

- Elementos de modelo que definem dependências (Figura C.4)
- Elementos de modelo que definem vários tipos de classificadores (Figura C.5)
- Elementos auxiliares para parâmetros template, apresentação e comentários. (Figura C.6)

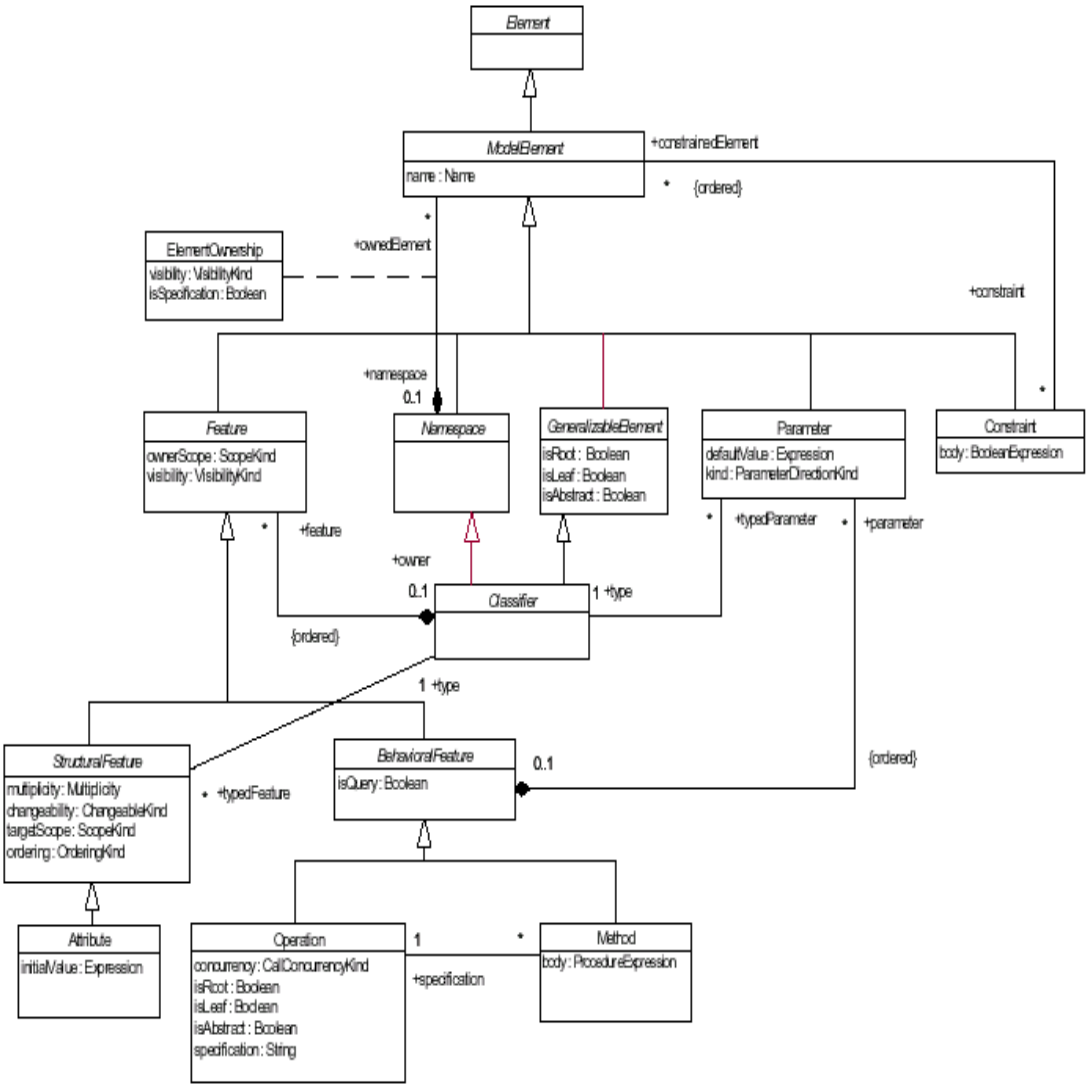

Figura C.2 – Pacote Core – Backbone

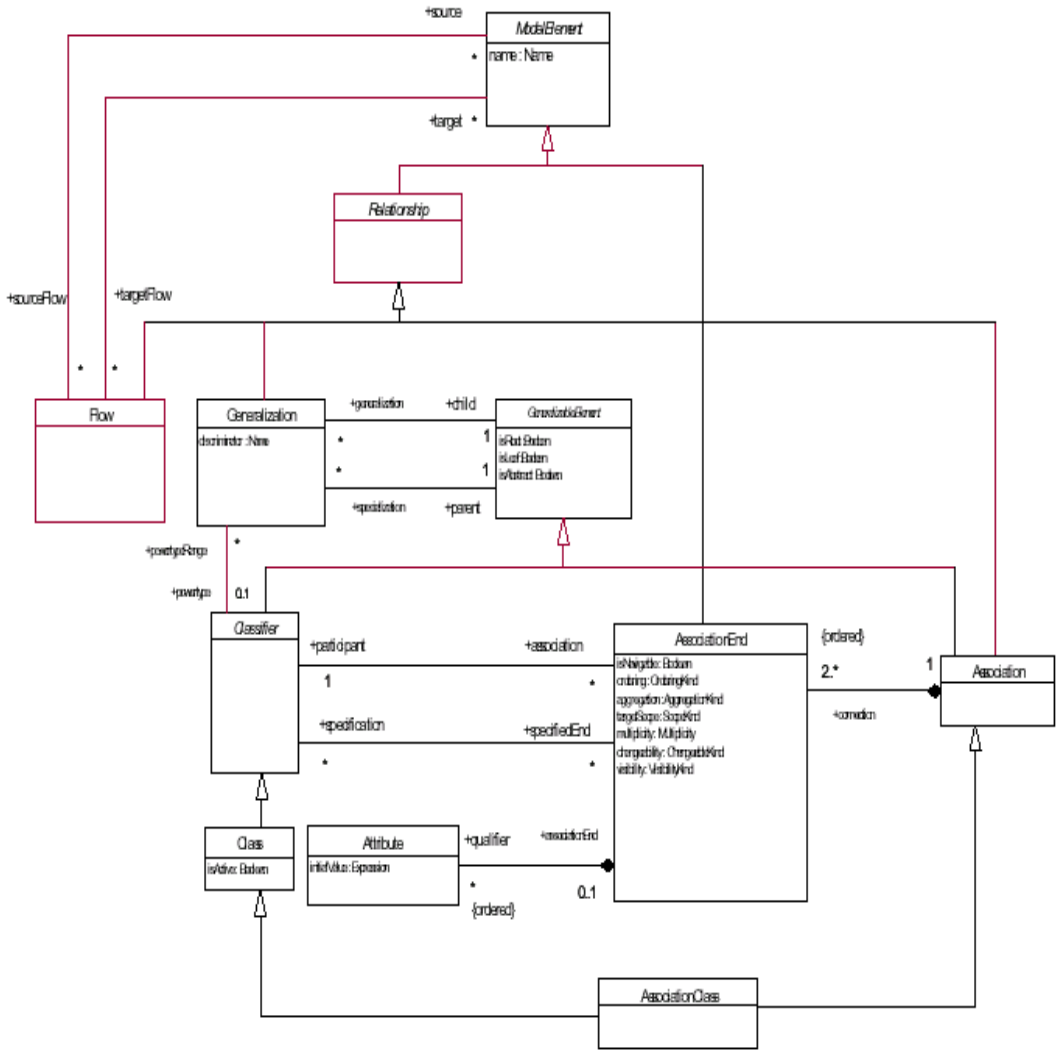

Figura C.3 –Pacote Core – Relçationships

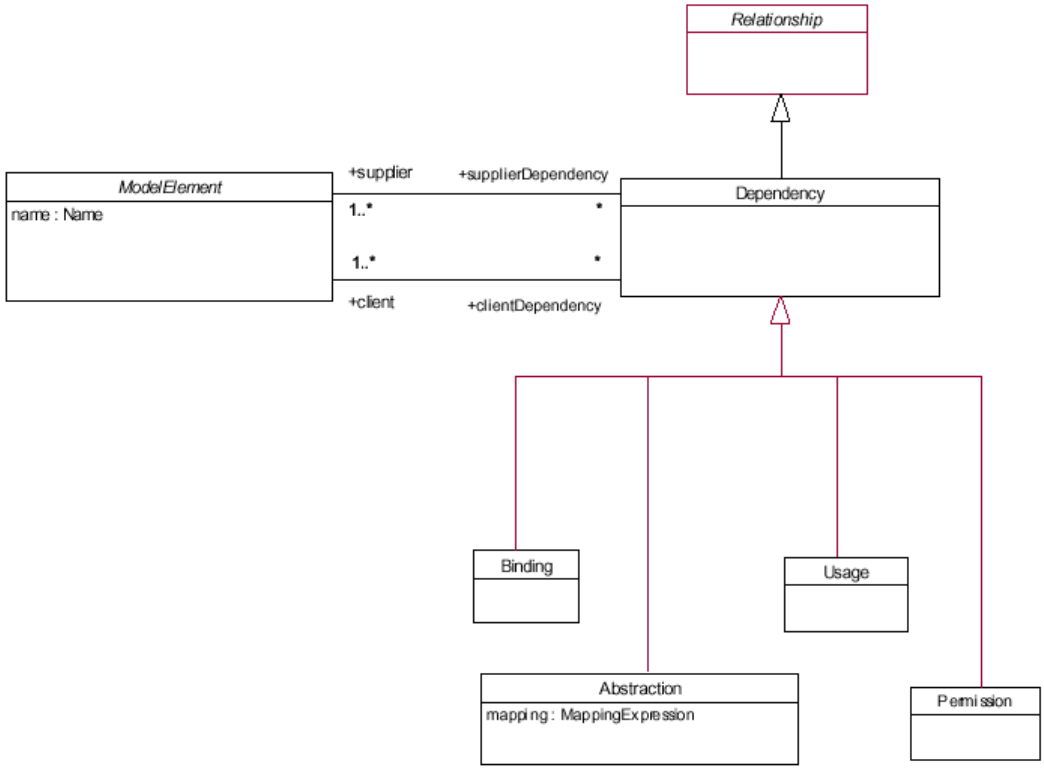

Figura C.4 – Pacote Core – Dependencies

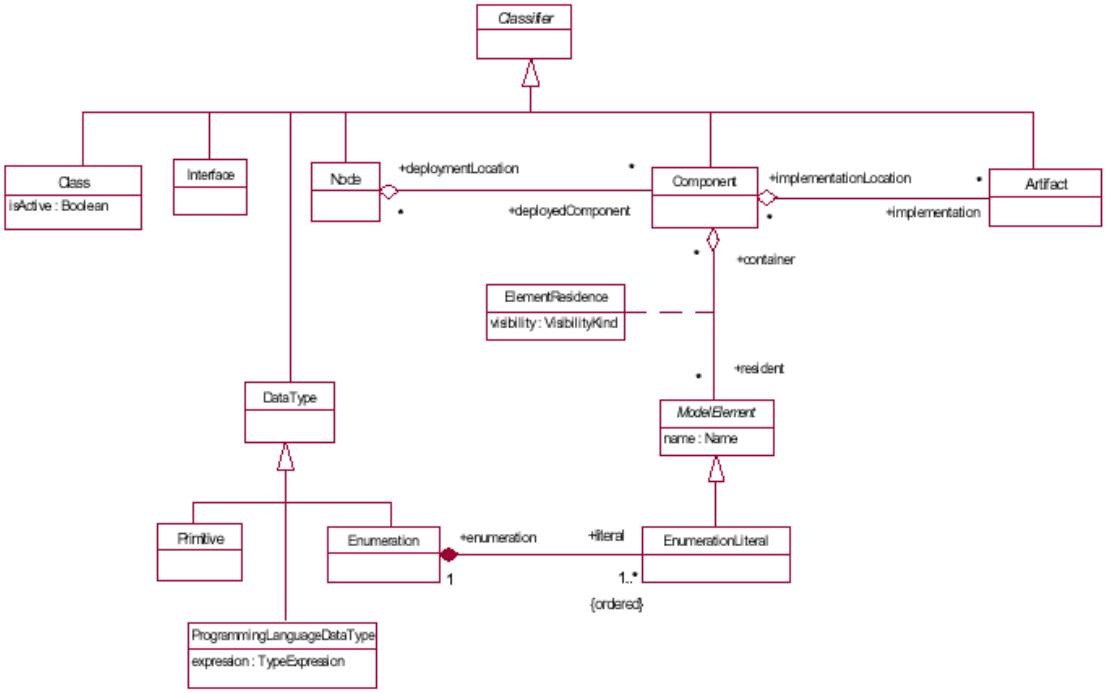

Figura C.5 – Pacote Core – Classifiers

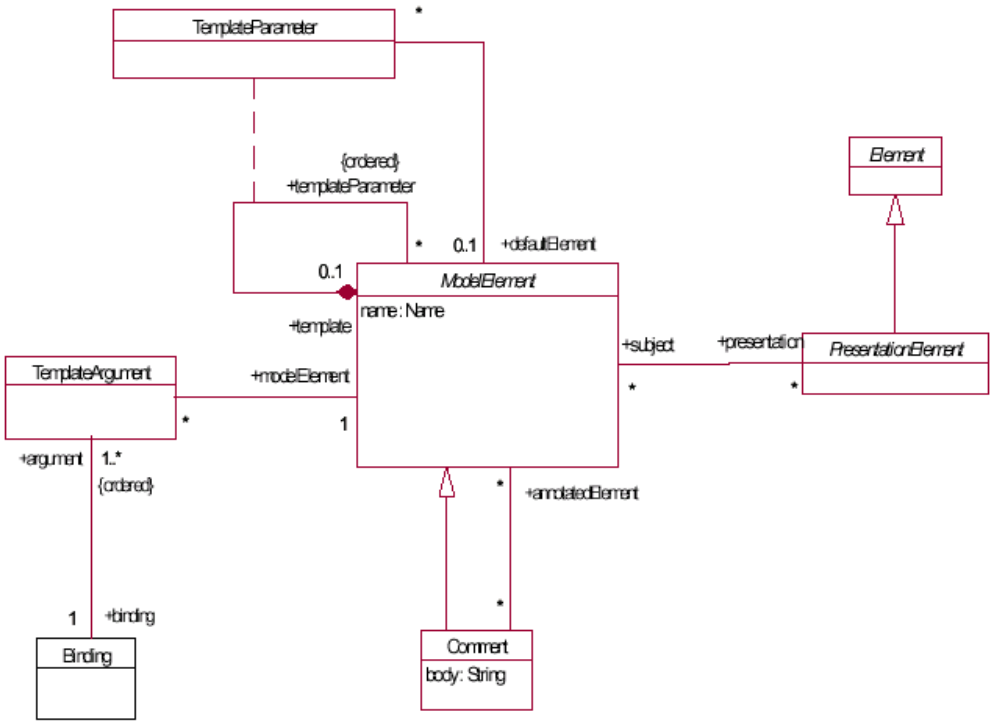

Figura C.6 – Pacote Core – Auxiliary Elements

Pacote "Data Type"

O pacote Data Types especifica os diferentes tipos de dados que são usados para definir o UML.

A sintaxe abstracta deste pacote é expressa através das figuras (Figura C.7 e Figura C.8) que se seguem.

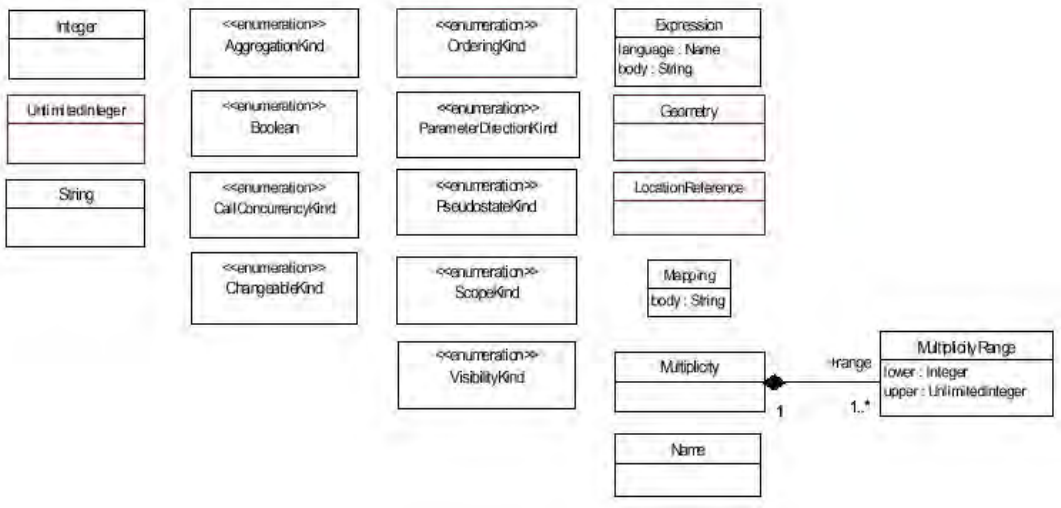

Figura C.7 – Pacote Data Types – Main

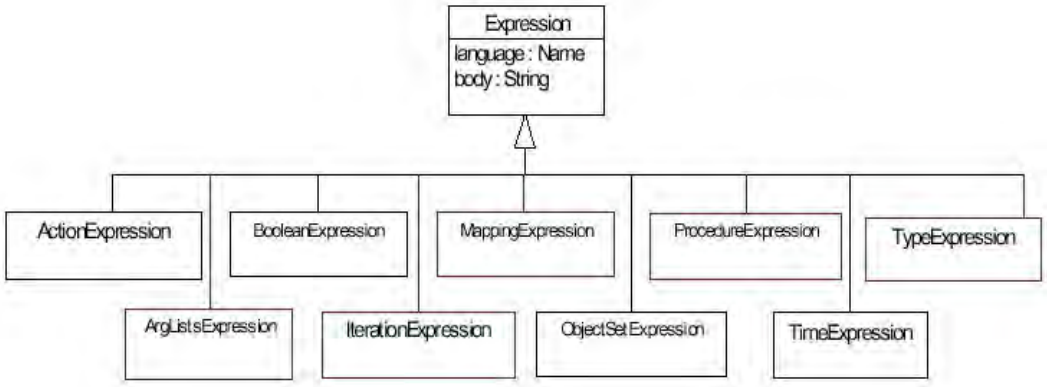

Figura C.8 – Pacote Data Types – Expressions

Pacote de "Extension Mechanisms"

O pacote de mecanismos de extensão ("*Extension Mechanisms*") especifica como os elementos de modelo são configurados e expandidos com nova semântica através do uso de estereótipos, restrições e definições de etiquetas.

As extensões têm de ser estritamente aditivas à semântica do standard UML, e portanto, não contradizer ou entrar em conflito com a semântica standard. De facto, os mecanismos de extensão são um meio de refinar à semântica standard do UML, e não de suportar extensão arbitrária da semântica.

A sintaxe abstracta do pacote Extension Mechanisms é expressa em notação gráfica pela Figura C.9 que se segue.

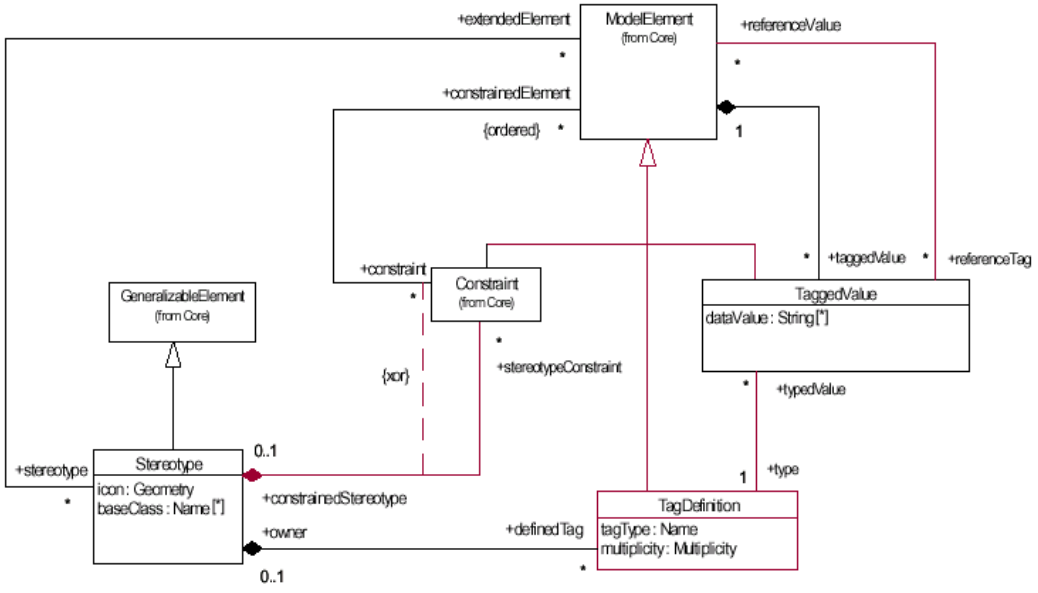

Figura C.9 – Pacote Extension Mechanisms

C.2. Pacote "Behavioral Elements"

O pacote Behavioral Elements é a superestrutura da linguagem que especifica o comportamento dinâmico dos modelos. É decomposto nos pacotes ilustrados pelaFigura C.10:

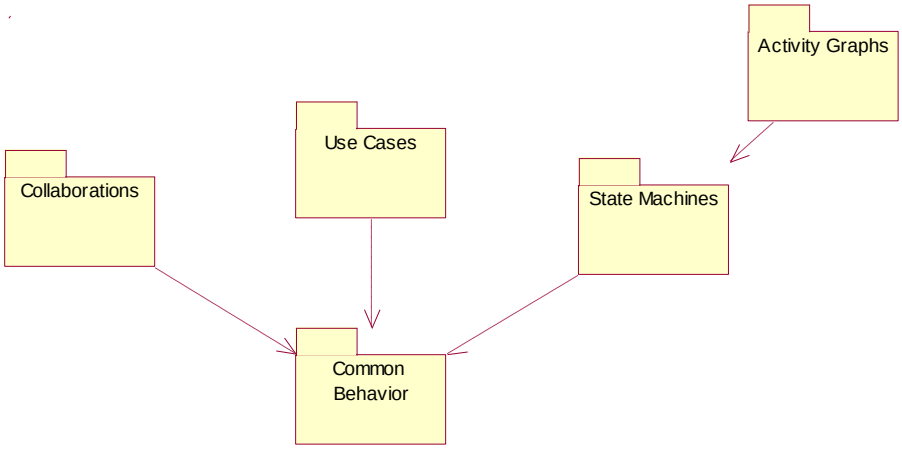

Figura C.10 – Pactoes de "Behavioral Elements"

Pacote "Common Behavior"

O pacote "*Common Behavior*" é o mais fundamental dos subpacotes que constituem o pacote "*Behavioral Elements*". Especifica os conceitos nucleares requeridos para os elementos dinâmicos e fornece a infra-estrutura para suportar Colaborações, Máquina de Estados e Casos de Uso.

A sintaxe abstracta para o pacote Common Behavior é expresso em notação gráfica pelas seguintes figuras:

- Elementos de modelo que definem Sinais e Recepções (Figura C.11)
- Elementos de modelo que especificam Acções (Figura C.12)
- Elementos de modelo que definem Instâncias (Figura C.13)
- Elementos de modelo que definem Ligações ("*links*") (Figura C.14)

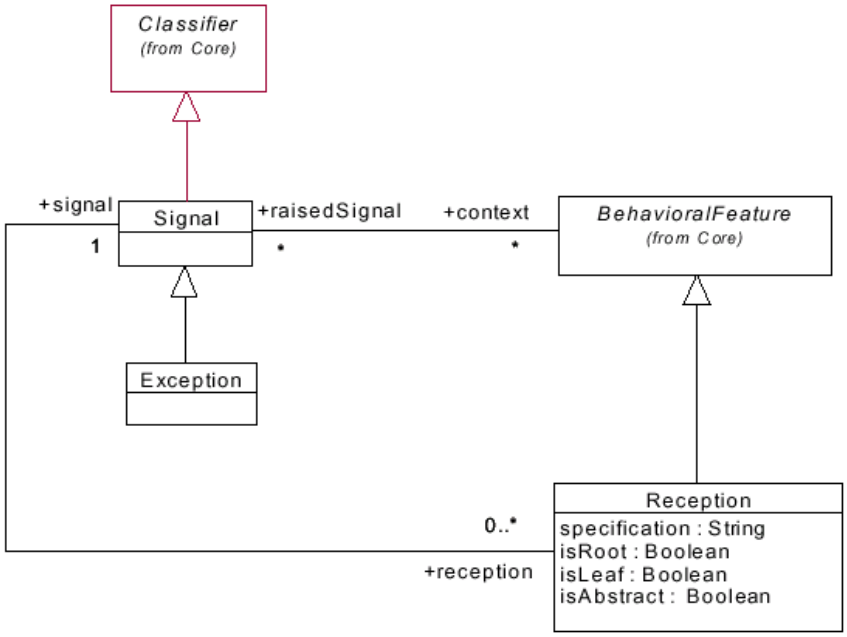

Figura C.11 – Pacote Common Behavior – Signals

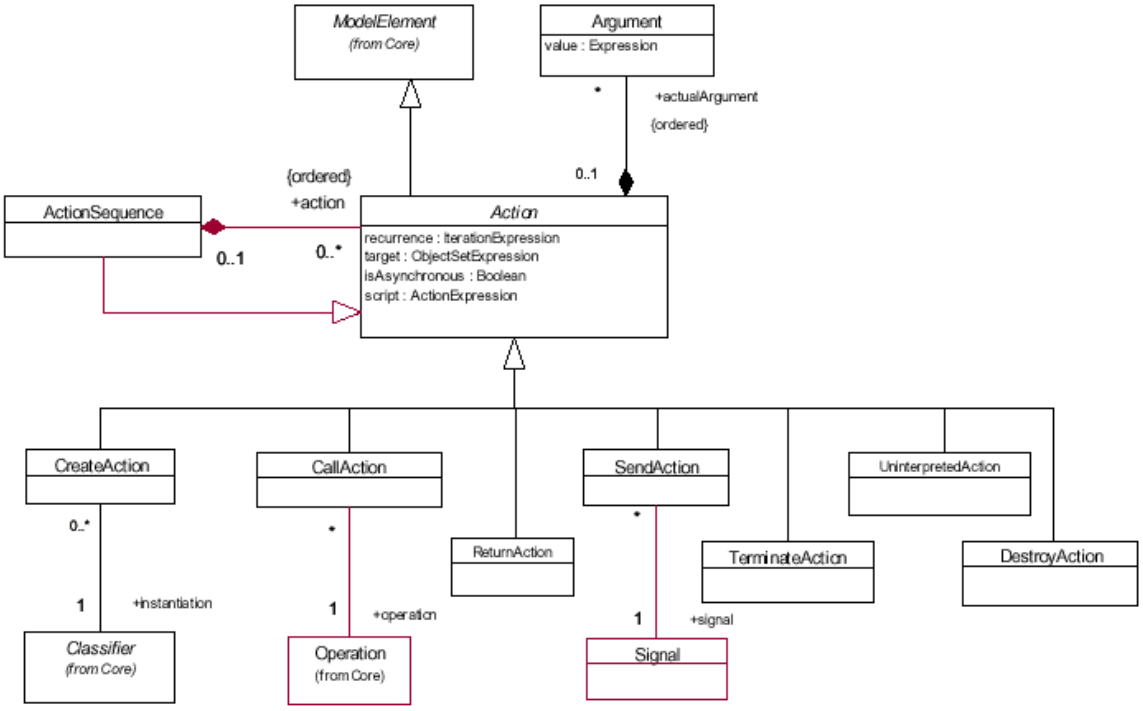

Figura C.12 – Pacote Common Behavior – Actions

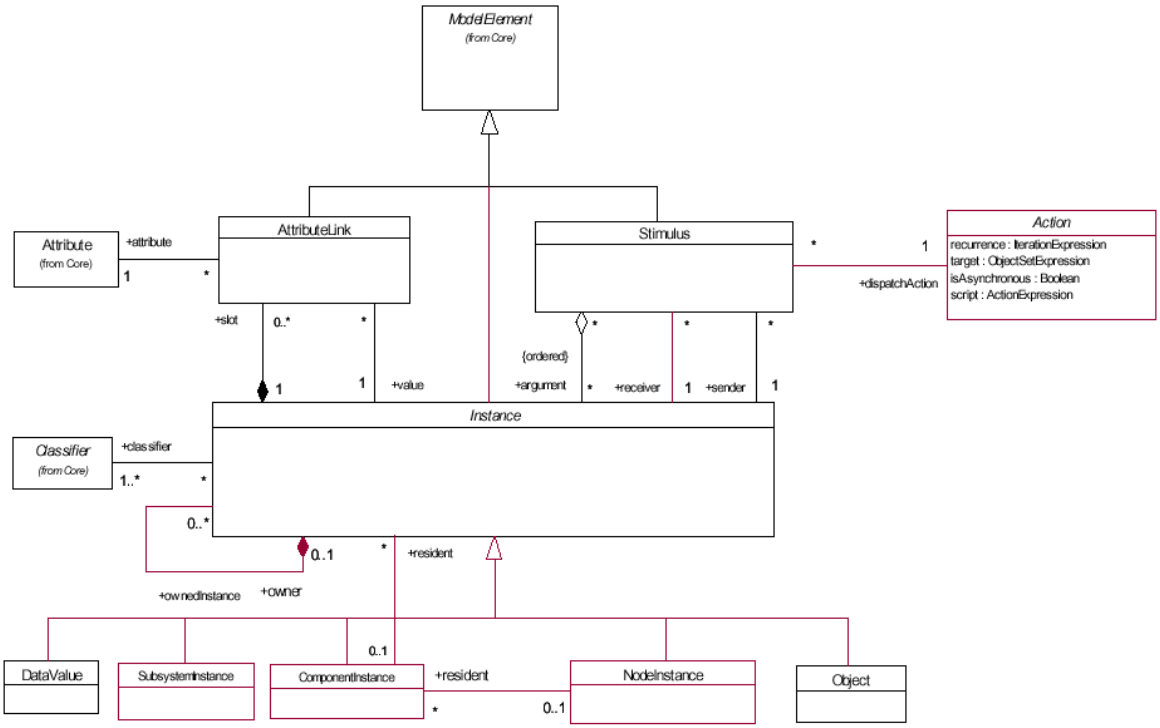

Figura C.13 – Pacote Common Behavior – Instances

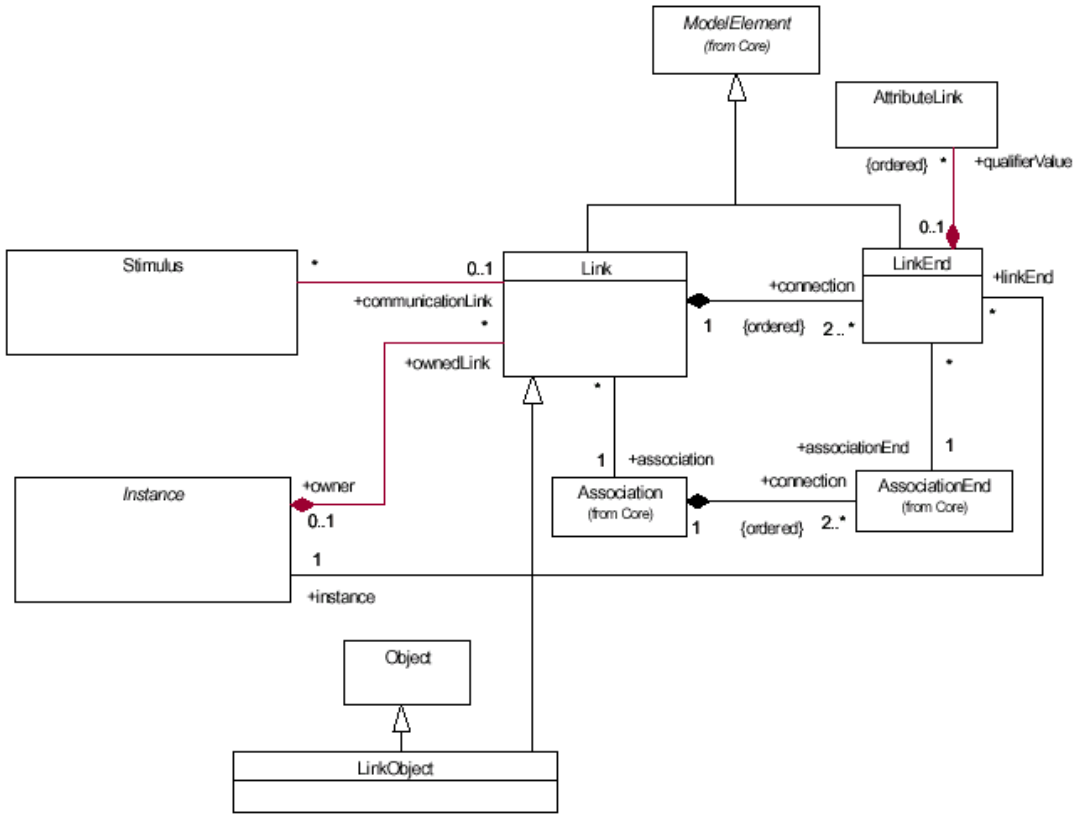

Figura C.14 – Pacote Common Behavior – Links

Pacote "Collaborations"

A descrição de instâncias cooperantes envolve dois aspectos: 1) a descrição estrutural dos participantes, e 2) a descrição dos seus padrões de comunicação. A estrutura dos participantes que desempenham papéis é chamada de "Colaboração". O padrão de comunicação efectuado pelas instâncias que desempenham os papéis em ordem a concretizar uma tarefa é chamada de "Interacção". O comportamento é implementado por conjuntos de instâncias que trocam estímulo dentro de uma interacção global. Uma interacção é definida no contexto de uma colaboração.

A sintaxe abstracta para o pacote Colaborações ("*Collaborations*")é expresso em notação gráfica nas figuras:

- " Collaborations – Roles" (Figura C.15)
- "Collaborations – Interactions" (Figura C.16)
- "Collaborations – Instances" (Figura C.17)

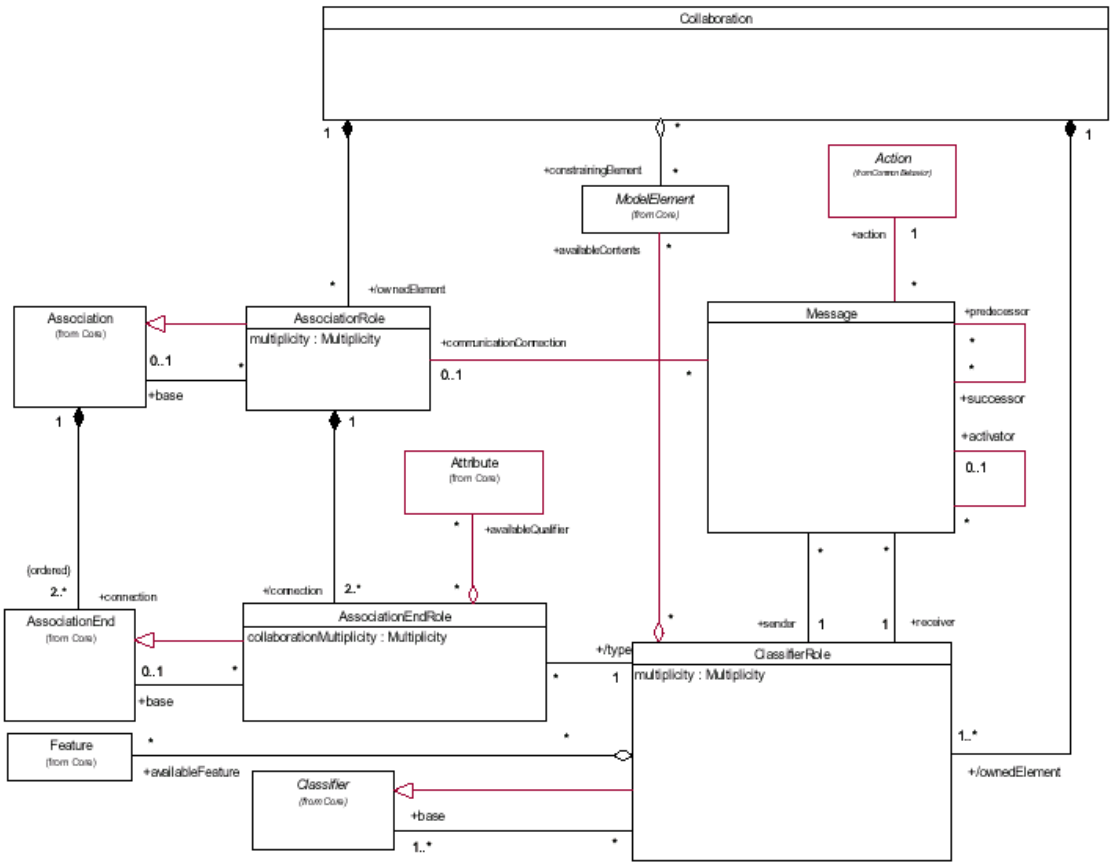

Figura C.15 – Pacote Collaboratoions – Roles

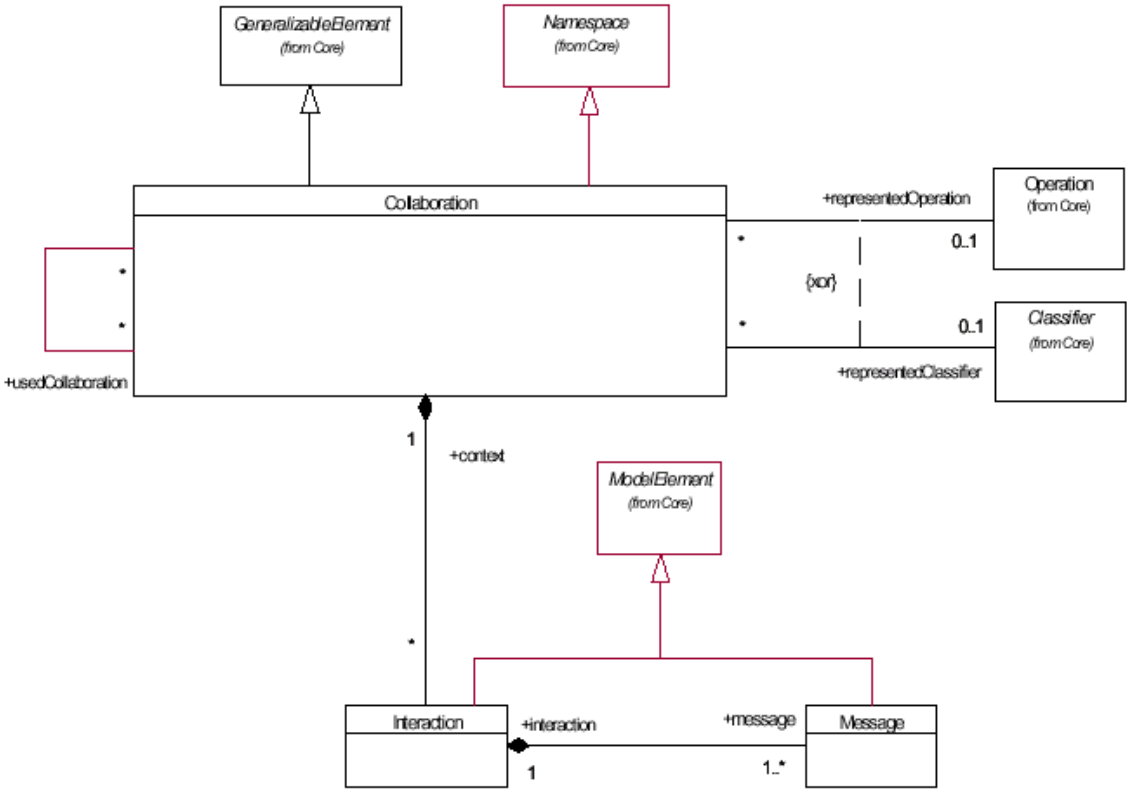

Figura C.16 – Pacote Collaborations – Interactions

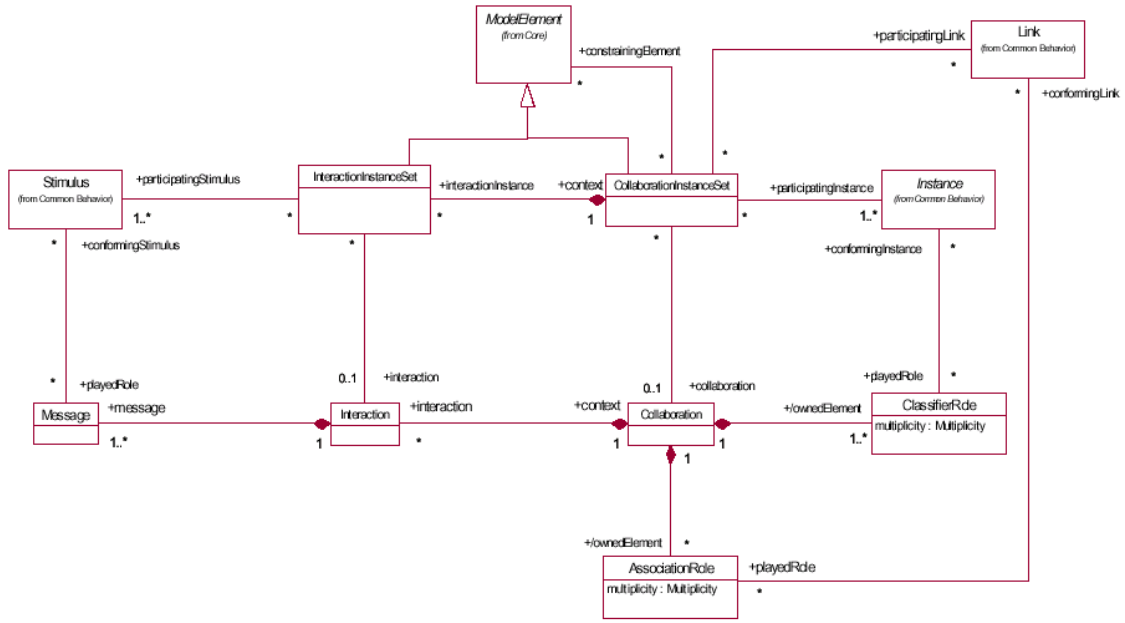

Figura C.17 – Pacote Collaborations – Instances

Pacote "Use Cases"

Os elementos no pacote caso de uso ("*Use Case*") são principalmente usados para definir o comportamento de uma entidade como um sistema ou um subsistema sem especificar a sua estrutura interna. Os elementos chave neste pacote são "*UseCase*" e "*Actor*". Instâncias de casos de uso e instâncias de actores interagem quando os serviços da entidade são utilizados.

A sintaxe abstracta para o pacote casos de uso é expresso em notação gráfica na Figura C.18 que a seguir se apresenta.

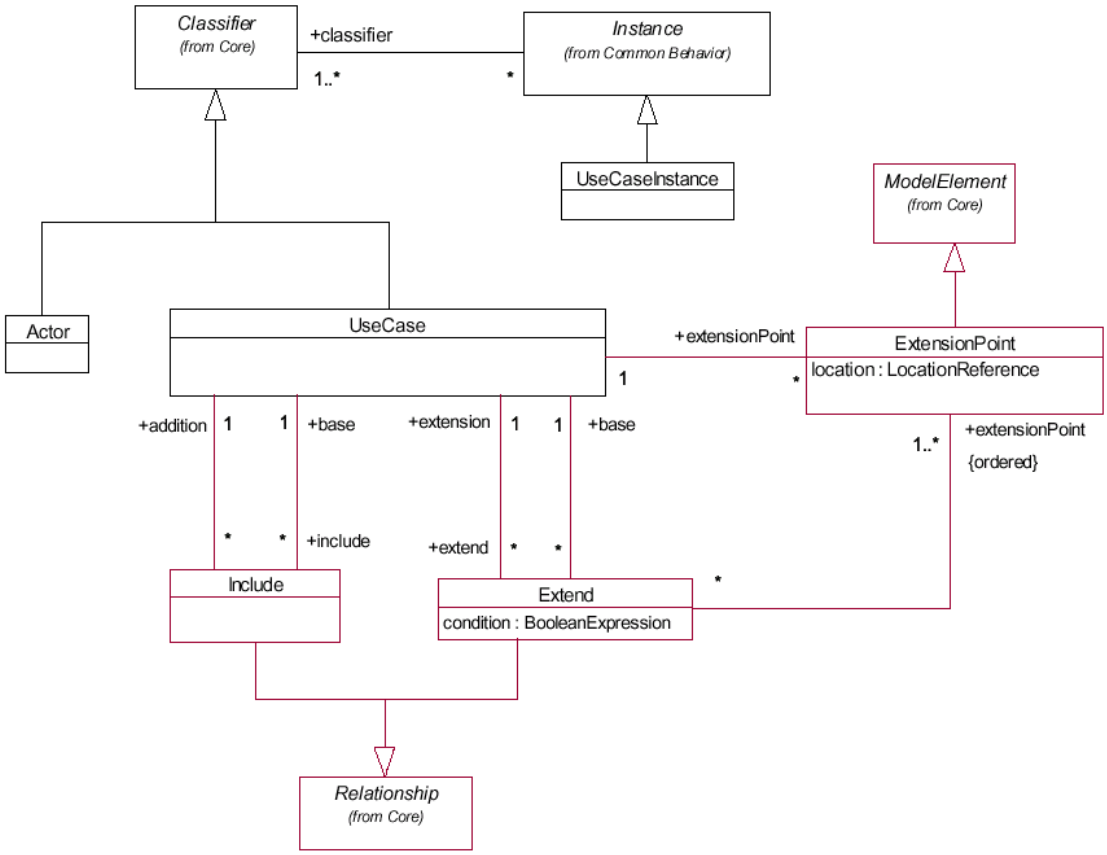

Figura C.18 – Pacote Use Cases

Pacote "State Machines"

O pacote "máquina de estados ("*State Machine*") especifica um conjunto de conceitos que podem ser utilizados para a modelação de comportamento discreto através de sistemas de transição de estados finitos.

A sintaxe abstracta para as máquinas de estado é expresso nas figuras descrevem:

- Todos os conceitos de grafos de máquinas de estado (Figura C.19)
- Os eventos que podem despoletar comportamento da máquina de estados. (Figura C.20)

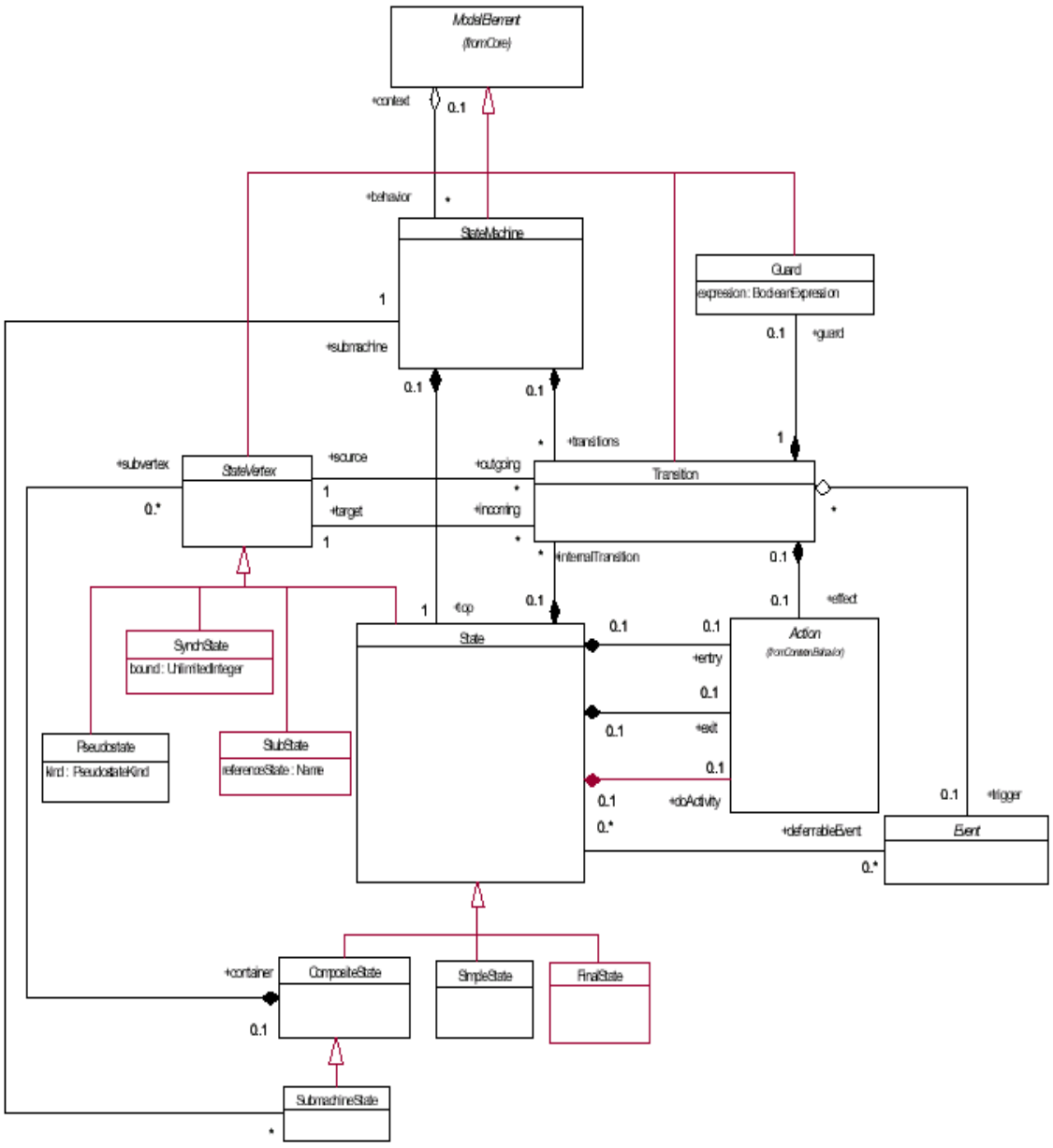

Figura C.19 – State Machines – Main

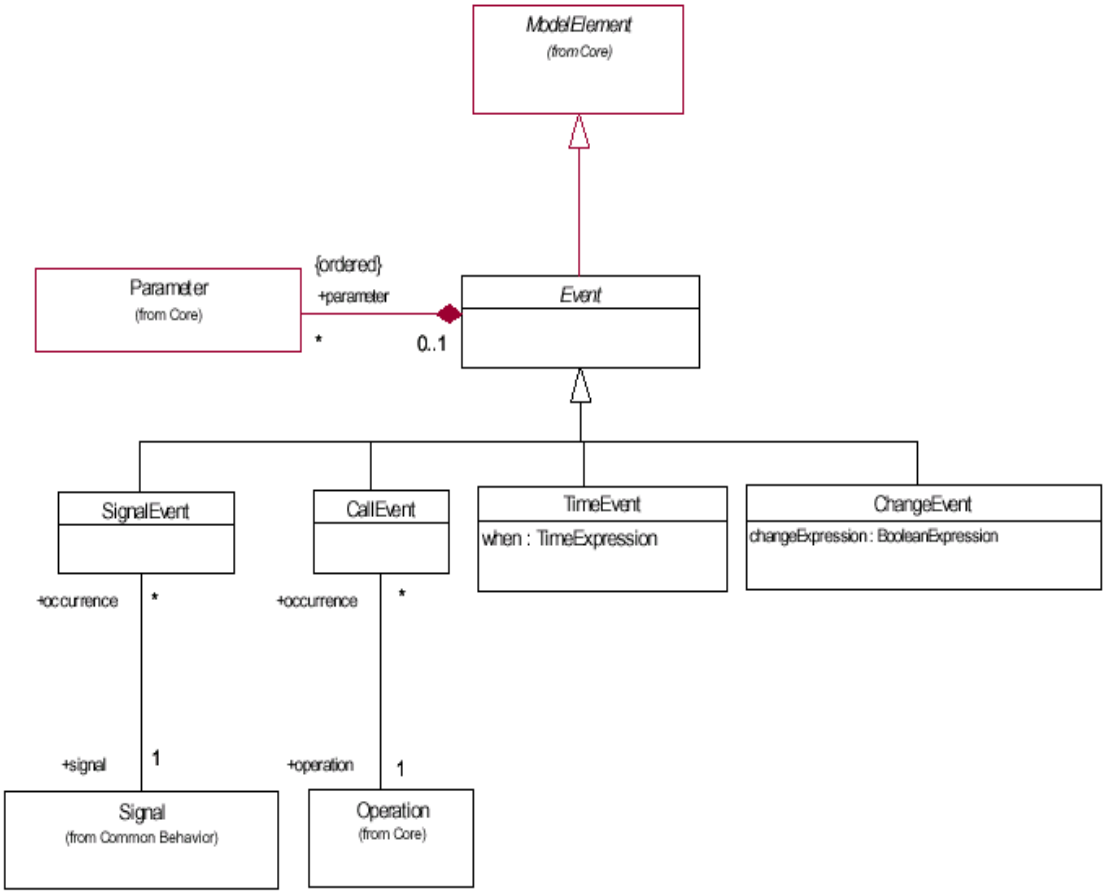

Figura C.20 – Pacote State Machines – Events

Pacote "Activity Graphs"

O pacote grafos de actividade ("*Activity Graphs*") define uma vista expandida do pacote "*State Machine*". Máquinas de Estado e Grafos de Actividade são ambos essencialmente sistemas de transição de estados, e partilham muitos elementos metamodelo.

Deve ser notado que a extensão grafos de actividade tem pouca semântica própria, a qual deve ser compreendida no contexto do pacote "*State Machine*".

A sintaxe abstracta para os grafos de actividade é expresso em notação gráfica pela Figura C.21.

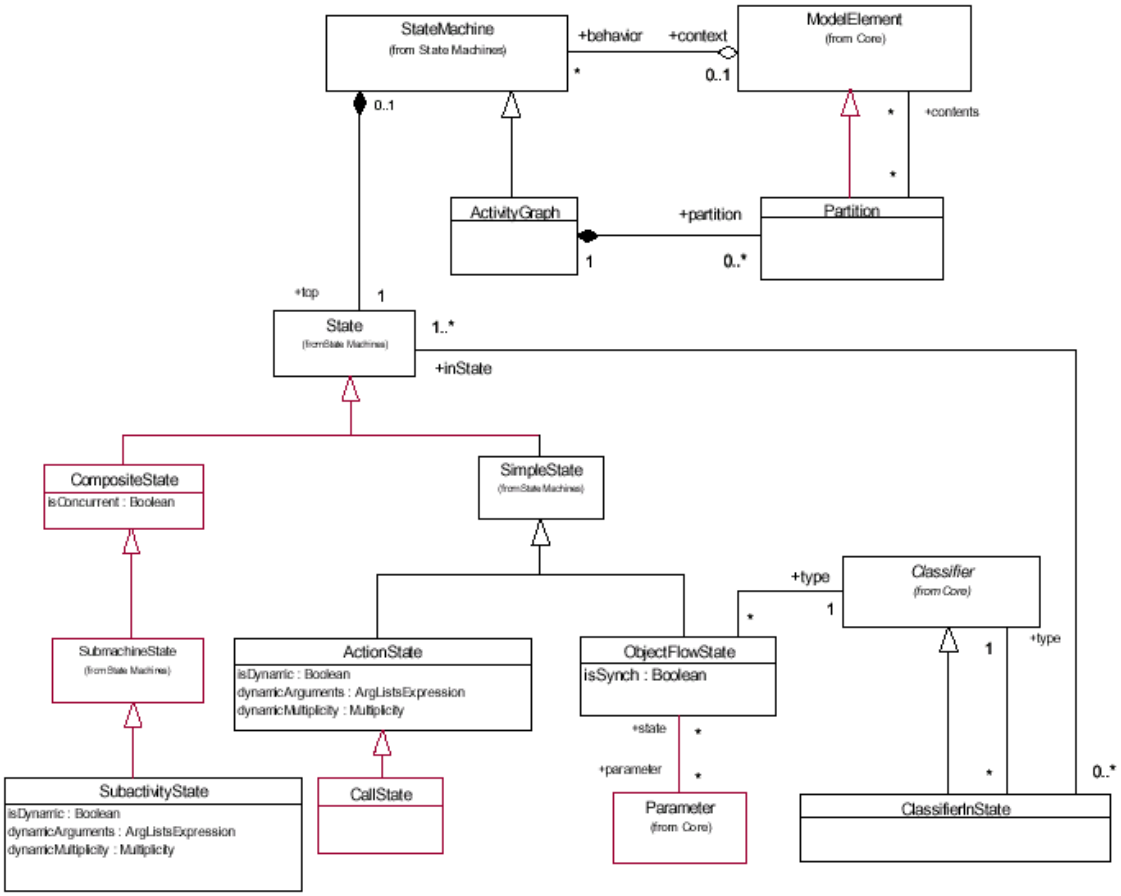

Figura C.21 – Pacote Acivity Graphs

C.3. Pacote "Model Management"

Os modelos são usados para capturar vistas diferentes de um sistema físico. Os pacotes são usados dentro de um modelo para agrupar elementos de modelo. Um subsistema representa uma unidade comportamental no sistema físico. Os Perfis UML são pacotes dedicados para agrupar extensões UML.

A sintaxe abstracta do pacote de Gestão de Modelo ("*Model Management*") é expresso na notação gráfica expressa na Figura C.22.

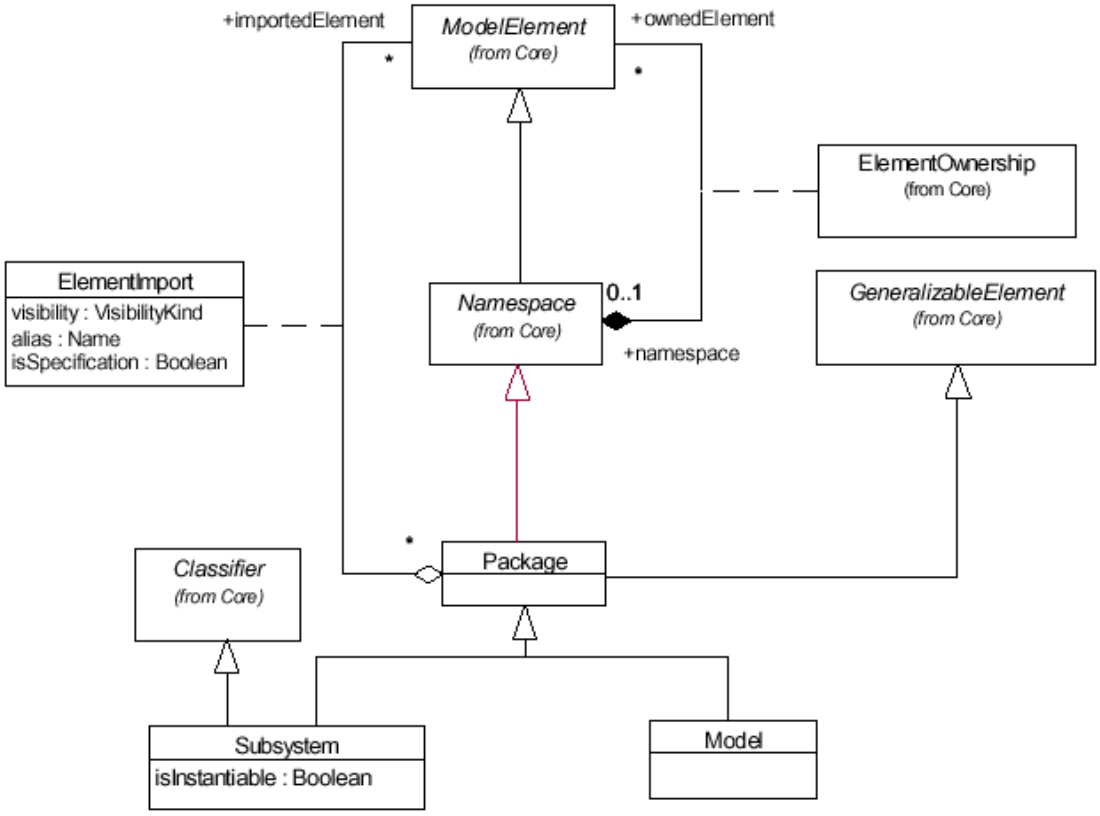

Figura C.22 – Pacote Model Management