SEMINÁRIO DE PESQUISA EM ONTOLOGIA NO BRASIL
3ºONTOBRAS
30 e 31 de agosto de 2010 - Florianópolis/SC

Padrões de Modelagem e Regras de Construção de Modelos para a criação de Ontologias de Domínio Bem-Fundamentadas em OntoUML

Alex Pinheiro das Graças (NEMO/UFES) apgracas@inf.ufes.br

Giancarlo Guizzardi (NEMO/UFES) gguizzardi@inf.ufes.br

Resumo: OntoUML é uma linguagem de modelagem derivada de uma Ontologia de Fundamentação e destinada à construção de ontologias de domínio como modelos conceituais de referência. Apesar de sua crescente adoção, OntoUML tem sido reportada como uma linguagem de complexa para modeladores novatos. Este artigo apresenta contribuições teóricas e metodológicas para apoiar esses modeladores. Em primeiro lugar, o artigo explora padrões de modelagem instrínsecos à linguagem OntoUML e oriundos dos fundamentos ontológicos subjacentes à linguagem. Estes padrões são utilizados para derivar regras de construção de modelos que representam explícitamente as regras de formação destes padrões. Em segundo lugar, o artigo demonstra como essas regras podem ser materializadas em forma de um guia metodológico que, em formato de diálogo, conduz o modelador na construção de modelos bem-fundamentados nesta linguagem.
Palavras-chave: Representação de Ontologias; OntoUML; Fundamentos Ontológicos.

1. Introdução

A construção de modelos conceituais ontologicamente bem-fundamentados e, em particular, de ontologias de domínio bem-fundamentadas, é uma atividade de substancial complexidade. Como discutido em (Guizzardi e Halpin, 2008), tem-se reconhecido cada vez mais a necessidade de se ter linguagens e metodologias para construção de ontologias que incorporem distinções verdadeiramente ontológicas e diretrizes metodológicas expresivas.

Uma linguagem construída especificamente para lidar com essa questão é a OntoUML (Guizzardi, 2005). Trata-se de uma versão estendida da UML 2.0, cujo metamodelo incorpora primitivas de modelagem e uma rica axiomatização derivadas das teorias ontológicas que compõem a ontologia de fundamentação UFO (ibid.). O foco em uma linguagem desse tipo é em *expressividade* e *clareza de representação* e não em requisitos não-funcionais (e.g., tratabilidade computacional, decidibilidade, etc.). A idéia é que uma vez criada uma ontologia em OntoUML, esta, como modelo conceitual, pode então dar origem a diversas codificações, em diferentes linguagens (e.g., OWL DL, RDF, F-Logic, CASL) visando maximizar diferentes requisitos não funcionais (Guizzardi e Halpin, 2008).

OntoUML tem sido utilizada com sucesso na criação de ontologias em diversos domínios complexos como eletrofisiologia do coração (Gonçalves et al., 2009) e Petróleo e Gás (Guizzardi et al., 2010). Além disso, tanto a linguagem quanto seus conceitos subjacentes têm experimentado uma crescente adoção (Halpin, 2008)(Baumman, 2009), muito embora tal linguagem apresente um alto grau de complexidade para modeladores iniciantes.

Assim, o objetivo neste trabalho é propor contribuições teóricas e metodológicas que visam apoiar usuários iniciantes da linguagem na construção de modelos conceituais em OntoUML. A seção seguinte faz uma breve introdução a um fragmento de OntoUML relevante à discussão desenvolvida neste artigo. A seção 3 explora uma nova perspectiva de uso para OntoUML, na qual a linguagem é vista como uma Linguagem de Padrões (*Pattern Language*) tal que suas primitivas de modelagem se manifestam não como elementos isolados mas como elementos componentes de *padrões de modelagem*. Além disso, a seção propõe *regras de construção de modelos* que visam sistematizar as regras de formação desses padrões. Um guia metodológico para utilização de tais regras é apresentado na seção 4. O objetivo deste guia é conduzir em forma de diálogo a construção de modelos bem-fundamentados nesta linguagem. Por fim, a seção 5 apresenta brevemente algumas direções de trabalhos futuros.

2. Pressupostos Teóricos: A Linguagem OntoUML

A linguagem OntoUML é definida pela extensão do metamodelo da linguagem UML para construção de diagrama de classes, a partir da aplicação de uma série de distinções ontológicas reunidas numa ontologia de fundamentação chamada UFO-A (Guizzardi, 2005). Contudo, este trabalho foca apenas nas distinções que refinam a primitiva *Class* de modelagem UML, representadas na UFO-A como categorias de Tipos de Objetos (*Object Type*), conforme mostra a figura 1. Mais especificamente, uma categoria particular de Tipos de Objetos chamada Tipos Sortais (Sortal Type) será abordada. Estes são considerados os tipos mais recorrentes em modelos conceituais e ontologias de domínio (Halpin, 2008)(Baumman, 2008). Para uma descrição detalhada e caracterização formal da linguagem, o leitor deve ser referir a (Guizzardi, 2005).

Além de incorporar tais distinções ontológicas, o metamodelo estendido da OntoUML contêm restrições de integridade que definem as maneiras válidas de se combinar as primitivas de

modelagem que representam estas distinções. Na seção 2.1 são discutidas brevemente as referidas restrições e distinções.

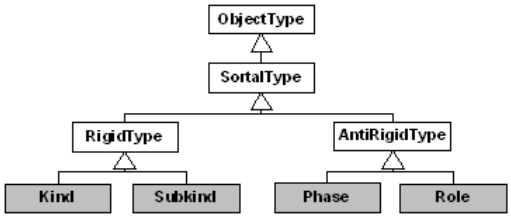

Figura 1 Distinções Ontológicas entre categorias de tipos de objetos em OntoUML

2.1 Categorias de Tipos de Objetos na OntoUML

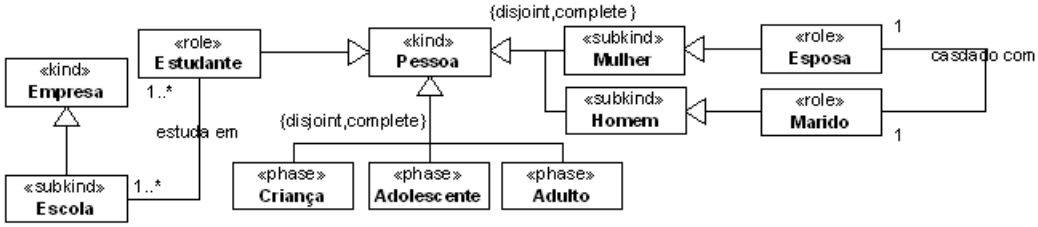

Figura 2 Exemplo de Modelo Conceitual em OntoUML

O modelo da figura 1 faz uma distinção básica entre duas categorias de Tipos de Objetos, considerando-se a (meta)propriedade ontológica chamada *Rigidez*. Um tipo T é rígido se para toda a instância x de T, x é necessariamente uma instância de T (em um sentido modal), i.e., x é uma instância de T em todas as situações possíveis (Guizzardi, 2005). Em contraste, um tipo é dito *Anti-Rígido* se para toda a instância x de T, é sempre possível que x deixe de ser uma instância de T, i.e., x pode deixar de ser uma instância de T sem deixar de existir (sem perder sua identidade) (ibid.).

Como exemplo (fig. 2), pode-se contrastar o tipo rígido Pessoa com o tipo anti-rígido Estudante. Suponha um indivíduo qualquer, nomeado João, que é instância de Pessoa e Estudante em uma determinada situação s_1. É possível existir uma situação s_2 em que João ainda exista, porém não seja mais instância de Estudante. Por outro lado, João será sempre instância de Pessoa em todas as situações em que ele existir.

Os tipos rígidos podem ser relacionados em cadeias de relações taxonômicas. Por exemplo, os tipos rígidos Homem e Pessoa são relacionados de forma que o segundo é uma generalização do primeiro. A raiz de uma cadeia de generalização envolvendo tipos rígidos é chamada de

um *Kind* e os tipos rígidos que especializam um *Kind* são chamados de *Subkind*. Como demonstrado em (ibid.), (R1) todo objeto deve ser instância de exatamente um *Kind*. Como consequência, (R2) para qualquer outro tipo T no modelo, ou T é um *Kind* ou T é uma especialização de exatamente um *Kind* (ibid.). *Subkinds* de um Tipo K tipicamente fazem parte do que é chamado de *Subkind Partition*. (R3) Um *Subkind Partition* $\langle SK_1...SK_n \rangle$ define de fato uma partição do tipo K que tais subkinds especializam, ou seja: (i) em qualquer situação, qualquer instância de SK_i é uma instância de K; (ii) qualquer instância de K é instância de exatamente um SK_i (ibid.). Em UML, e consequentemente OntoUML, partições são representadas pelo que é chamado de um Conjunto de Generalizações (*Generalization Set*) definido com os meta-atributos Disjunto e Completo (*Disjoint, Complete*)(ibid.). Na figura 2, os *subkinds* ⟨Homem, Mulher⟩ definem uma partição do tipo Pessoa.

Entre os tipos anti-rígidos, duas categorias são reconhecidas: *Phases* e *Roles*. Em ambos os casos, temos que suas instâncias podem passar a instanciar tal tipo, ou deixar de instanciá-lo, de forma dinâmica. No entanto, enquanto no caso de *Phases* isto se dá através da mudança de *propriedades intrínsecas* do indivíduo, no caso de *Roles*, a instanciação dinâmica é estabelecida pela mudança de uma *propriedade relacional* do objeto. Na figura 2, podemos contrastar as *Phases* Criança, Adolescente e Adulto, com os *Roles* Estudante, Marido e Esposa. Por exemplo, uma Pessoa passa a (ou deixa de) instanciar os tipos Criança, Adolescente ou Adulto caso sua idade (propriedade intrínseca) mude para estabelecer um valor dentro (ou fora) de uma determinada faixa etária. Em contraste, uma Pessoa passa a (ou deixa de) instanciar o tipo Estudante caso estabeleça um relacionamento com pelo menos um indivíduo do tipo Escola; assim como um Homem passa a instanciar o tipo Marido caso estabeleça uma relação com um indivíduo do tipo Esposa. Portanto *Roles*, além de serem anti-rígidos, possuem uma meta-propriedade (ausente em *Phases*) chamada de *Dependência Relacional*. Como consequência, (R4) qualquer *Role* deve ser representado conectado a uma relação que expressa esta propriedade relacional (e.g., “matriculado em” ou “empregado por” na fig. 2) e esta relação deve ter em sua outra terminação uma cardinalidade mínima ≥ 1 (ibid.).

Como discutido em (ibid.), (R5) *Phases* sempre ocorrem em uma partição, chamada *Phase Partition* de um tipo T. Incorporando restrições análogas aos itens (i) e (ii) da *Subkind*

Partition, para uma *Phase Partition* $\langle P_1...P_n \rangle$ tem-se que: (i) em qualquer situação, qualquer instância de P_i é uma instância de T; (ii) qualquer instância de T é instância de exatamente um P_i; Além disso, a seguinte restrição adicional é definida: (iii) para qualquer instância x de T e qualquer P_i, existe uma situação possível na qual x instância P_i (ibid.). Na figura 2, as *Phases* ⟨Criança, Adolescente, Adulto⟩ definem uma partição do tipo Pessoa.

Por fim, como provado em (ibid.), (R6) tipos rígidos não podem especializar tipos anti-rígidos.

3. Padrões de Modelagem e Regras de Formação de Modelos

O objetivo desta seção é demonstrar que, ao contrário de linguagens ontologicamente neutras como UML, EER ou OWL, OntoUML é uma linguagem de padrões (*Pattern Language*). Em outras palavras, quando um modelo é construído em OntoUML, a linguagem induz o modelador a trabalhar com a combinação de um conjunto de padrões de modelagem, ao invés de trabalhar com primitivas de modelagem de menor granularidade como Classe, Associação, Especialização, etc. Como será demonstrado aqui, a escolha por modelar um elemento do domínio usando uma das categorias representadas na linguagem implica na necessidade de representação de outros elementos associados. Nessa seção é também demonstrado como regras de formação de modelos podem ser diretamente derivadas desses padrões de modelagem. A hipótese considerada nesta abordagem (e ilustrada na próxima seção) é a de que a cada passo de modelagem, o espaço de solução (*Solution Space*) caracterizando as possibilidades de escolha primitivas de modelagem a serem utilizadas é reduzido. Assim, reduz-se a sobrecarga cognitiva do modelador e, consequentemente, a complexidade do processo de construção de modelos OntoUML.

3.1 *Phases*

Como discutido anteriormente, *Phases* se manifestam sempre como parte de uma *Phase Partition*, cuja superclasse comum é necessariamente um *Sortal* S. Este padrão é demonstrado na figura 3.a. Dessa forma, podemos criar uma regra de formação de modelos tal que toda vez que uma *Phase* P for instanciada (uma classe em OntoUML for estereotipada como *phase*), a regra R_P descrita abaixo se aplica. As tabelas a seguir definem as regras de forma que os itens agrupados por um colchete são mutuamente excludentes.

R_P (X)	(i) X deve ser conectado como subclasse a uma *Phase Partition* existente PP disjunta e completa; (ii) Deve ser criada uma nova *Phase Partition* PP disjunta e completa, X deve ser conectado como subclasse a PP. a. PP deve ser conectado a um tipo S existente, tal que S será a superclasse comum aos componentes de PP, e S ∈ {*kind*, *subkind*, *phase*, *role*}; b. Um novo tipo S ∈ {*kind*, *subkind*, *phase*, *role*} deverá ser criado, S deverá ser conectado a PP como superclasse, e uma das seguintes regras deverá ser aplicada a S: R_{SK} (seção 3.3), R_P, R_R (seção 3.2.), caso S seja um *Subkind*, uma *Phase*, ou um *Role*, respectivamente. c. Devem ser criadas novas *phases* $P_i..P_n$ complementares à X tal que cada uma dessas phases deve ser conectada a PP como subclasse;

3.2 *Roles*

*A*ssim como *Phases*, *Roles* representam especializações (possivelmente sucessivas) de um *Kind*, cuja condição de restrição é uma propriedade relacional. Desta forma, *Roles* se manifestam seguindo o padrão demonstrado na figura 3.b. A regra de formação correspondente ao tipo *role* é definida abaixo.

R_R (X)	(i) X deve ser conectado como subtipo a um tipo S existente | S ∈ {*kind*, *subkind*, *phase*, *role*}; (ii) Deve ser criado um novo tipo S ∈ {*kind*, *subkind*, *phase*, *role*}, S deverá ser conectada a X como supertipo, e uma das seguintes regras deverá ser aplicada a S: R_{SK} (seção 3.3), R_P (seção 3.1), R_R, caso S seja um *Subkind*, uma *Phase*, ou um *Role*, respectivamente. (iii) Caso X já não esteja associado a um outro tipo T através de uma associação R e com cardinalidade mínima ≥ 1 na terminação ligada a T, então R deverá ser criada; a associação R deverá ser conectada em uma terminação (*association end*) ao tipo X e em outra terminação ao tipo Q (tal que Q ≠ X). A terminação associada ao tipo Q deve ter cardinalidade mínima ≥ 1; (iv) O tipo Q pode ser um tipo existente ou deve ser criado. No último caso, uma das seguintes regras deverá ser aplicada a Q: R_{SK} (seção 2.3), R_P, R_R (seção 2.2.), caso Q seja um *Subkind*, uma *Phase*, ou um *Role*, respectivamente.

3.3 *Subkinds*

Subkinds podem se manifestar de duas formas: como uma especialização de um tipo S, ou como parte de um conjunto de generalização (*generalization set*) que tem como superclasse comum um tipo S. Estes padrões são ilustrados nas figuras 3.c e 3.d, respectivamente. A regra de formação correspondente ao tipo *subkind* é definida em seguida.

R_{SK} (X)	(i) X deve ser conectado como subclasse a um *generalization set* existente GS, tal que GS é disjunto e completo, e os componentes de GS são todos *subkinds*;

	(ii) X deve ser conectado a um tipo S existente, tal que S ∈ {*kind*, *subkind*}. Se existir outro *subkind* SK conectado ou que deve ser criado e conectado como subclasse de S então é criado um *generalization set* GS disjunto e completo, tal que X e SK fazem parte de GS, e S é conectado a GS como supertipo comum de seus componentes; (iii) Deve ser criado um novo tipo S ∈ {*kind*, *subkind*}, S deverá ser conectada a X como superclasse, e caso S seja um *subkind*, a regra R_{SK} é aplicada a S. Se existir outro *subkind* SK que deve ser criado e conectado como subclasse de S então é criado um *generalization set* GS disjunto e completo, tal que X e SK fazem parte de GS, e S é conectado a GS como supertipo comum de seus componentes;

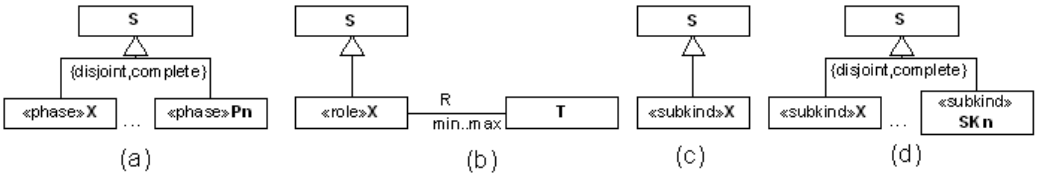

Figura 3 Padrão de Modelagem da Linguagem OntoUML para *Phases* (a), *Roles* (b) e *Subkinds* (c-d)

4. Ilustração de um Guia Metodológico baseado nas Regras de Formação Propostas

O objetivo dessa seção é exemplificar a aplicação das regras de formação de modelos OntoUML propostas na seção 3, usando como ilustração o modelo da figura 2. A execução destas regras é manifestada através de um diálogo com o modelador, servindo como guia metodológico de apoio ao usuário na construção do modelo. Os modelos resultantes da aplicação de cada um dos passos descritos a seguir são ilustrados na figura 4.

1. Suponha que o primeiro tipo incluído pelo usuário no modelo é o tipo *Criança*, e que este tipo é estereotipado como *Phase*.
2. A regra R_P *(Criança)* é então executada. Como não existe qualquer outro elemento do modelo, então o item (ii-b) é executado: *Criança é um subtipo de que? Resposta: Pessoa.* Em seguida, o item (ii-c) é executado: *Quais seriam as outras fases de Pessoa complementares a Criança? R: Adolescente, Adulto.* Por fim: *Existe um tipo T do qual Pessoa é um subtipo? R: Não.* Como conseqüência, nenhuma regra adicional é aplicada sobre *Pessoa*.
3. Suponha que o próximo tipo incluído seja *Escola*, e que este tipo é estereotipado como *subkind*.

4. A regra R_{SK} *(Escola)* é então executada. Como não existe uma partição formada por *subkinds* no modelo, as possibilidades de execução são os itens (ii) e (iii). Item (ii) é eliminado pela seguinte pergunta: *Escola é um tipo de Pessoa? R: Não.* Item (iii) é então executado: *Então, Escola é um subtipo de que? R: Empresa. Existe algum outro subtipo de Empresa e complementar a Escola que deve ser incluído no modelo? R: Não. Existe um tipo T do qual Empresa é um subtipo? R: Não.*
5. Suponha que o próximo tipo incluído seja *Estudante*, e que este tipo seja estereotipado como *Role*.
6. A regra R_R *(Estudante)* é então executada. As possibilidades de execução são os itens (i) e (ii). Item (i) é confirmado pela seguinte pergunta: *Estudante é um tipo de Empresa, Escola, Pessoa, Criança, Adolescente, ou Adulto? R: Pessoa.* Item (iii) é então executado: *Estudante é um papel que Pessoa desempenha quando relacionada com o que? Escola, Empresa, Criança, Adolescente ou Adulto? R: Escola. Como se chamaria esta relação? R: estuda em. Um Estudante estuda em no mínimo quantas Escolas? R: 1. Existe um máximo? R: Não. Uma Escola está relacionada com no mínimo quantos Estudantes? R: 1. Existe um máximo? R: Não.*
7. Suponha que o próximo tipo incluído seja *Mulher*, e que este tipo seja estereotipado como *subKind*.
8. A regra R_{SK} *(Mulher)* é então executada. Existe neste momento somente um único *generalization set* formado somente por *subkinds*. O item (i) é então descartado pela seguinte pergunta: *Mulher é um subtipo de Empresa complementar a Escola? R: Não.* As possibilidades de execução são os itens (ii) e (iii). Item (ii) é confirmado pelas seguintes perguntas: *Mulher é um subtipo de Pessoa? R: Sim. Existe algum subtipo adicional de Pessoa complementar a Mulher que deve ser representado? R: Homem.*
9. Suponha que o próximo tipo incluído seja *Marido*, e que este tipo seja estereotipado como *Role*.
10. A regra R_R *(Marido)* é então executada. As possibilidades de execução são os itens (i) e (ii). Item (i) é confirmado pela seguinte pergunta: *Marido é um tipo de Empresa, Escola, Pessoa, Homem, Mulher, Criança, Adolescente, ou Adulto? R: Homem.* Item (iii) é então executado: *Marido é um papel que Homem desempenha quando relacionado com o que? Empresa, Escola, Pessoa, Mulher, Criança, Adolescente, Adulto? R: Esposa.*

Como se chamaria esta relação? R: casado com. Um Marido é casado com no mínimo quantas Esposas?R: 1. Existe um máximo?R: 1 Uma Esposa está relacionada com no mínimo quantos Maridos?R: 1. Existe um máximo? R: 1.

11. A regra R_R *(Esposa)* é então executada. As possibilidades de execução são os itens (i) e (ii). Item (i) é confirmado pela seguinte pergunta: *Esposa é um tipo de Empresa, Escola, Pessoa, Homem, Mulher, Criança, Adolescente, Adulto ou Marido? R: Mulher.* Como *Esposa* já está conectada a *Marido*, o item (iii) não é executado.

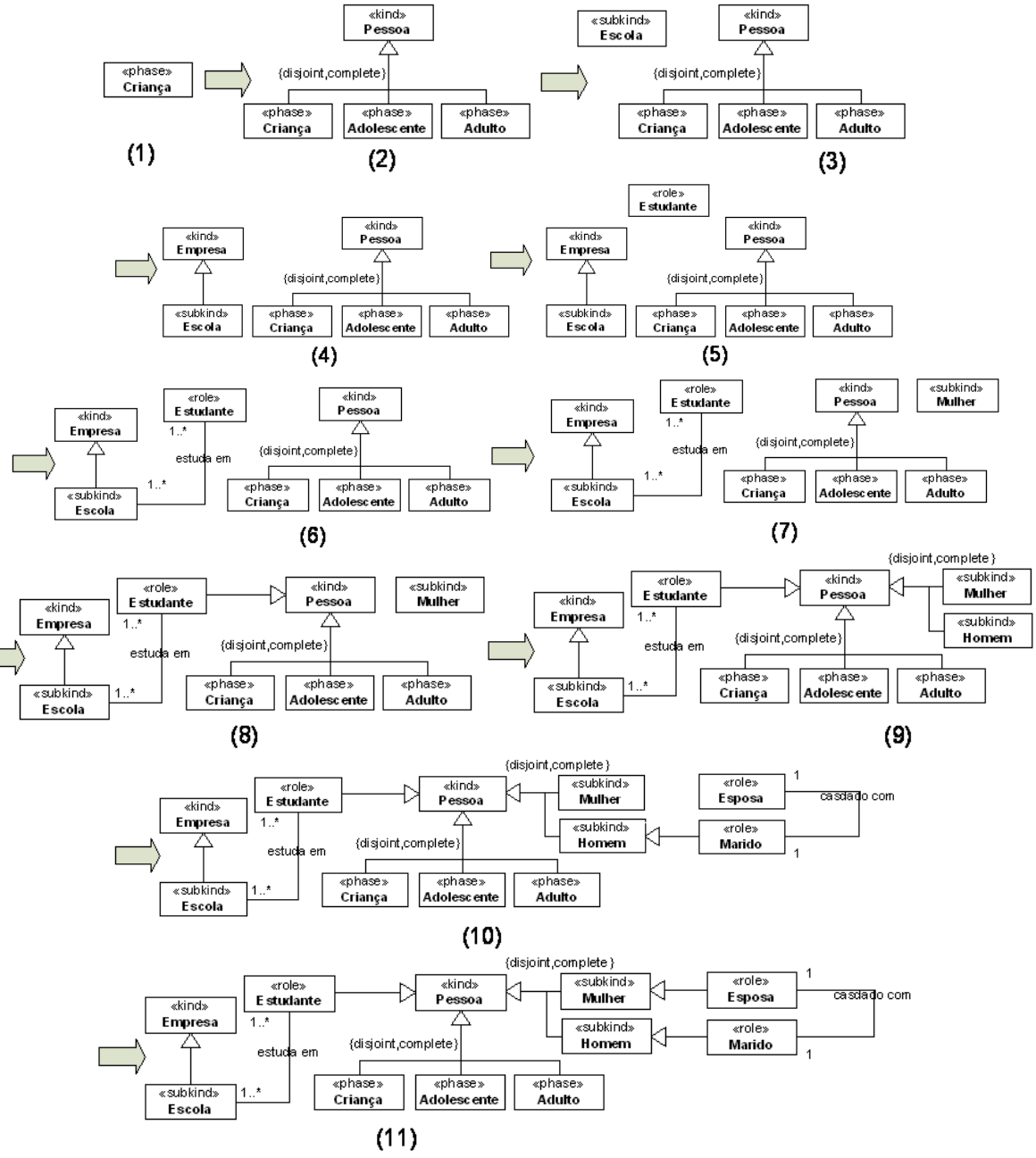

Figura4 Passos de formação de um modelo OntoUML pela aplicação das Regras de Construção propostas

5. Trabalhos Futuros

A sistematização das regras de formação indutiva dos padrões de modelagem explorada neste artigo (bem como a organização destas regras em forma de guia metodológico) tem como objetivo final servir como base para a construção de assistentes automatizados de modelagem (*Wizards*) para modeladores novatos. Em trabalhos futuros, a contribuição teórica apresentada neste artigo (padrões e regras de formação) deverá ser ampliada em duas direções: (i) explorar padrões de modelagem contemplados no conjunto total de primitivas de modelagem da linguagem; (ii) explorar regras que devem ser ativadas pela *alteração de modelos*. Uma vez realizados esses passos, pretende-se implementar o conjunto completo destas regras, bem como as diretrizes metodológicas delas derivadas em uma versão do atual *OntoUML Editor* (Benevides e Guizzardi, 2009). Uma outra direção a ser explorada nesta pesquisa é a realização de trabalhos empíricos para verificar a hipótese de pesquisa subjacente a este trabalho, ou seja, se a proposta metodológica apresentada aqui (e suas extensões) de fato contribuem para a diminuição da complexidade do processo de construção desses modelos por modeladores novatos, bem como para a melhoria da qualidade dos modelos resultantes.

Referências

Bauman, B. T., Prying Apart Semantics and Implementation: Generating XML Schemata directly from ontologically sound conceptual models, Balisage Markup Conference, 2009.

Benevides, A.B.; Guizzardi, G. A Model-Based Tool for Conceptual Modeling and Domain Ontology Engineering in OntoUML, LNBPI, Vol. 24, 2009. ISSN 1865-1356.

Gonçalves, B. N.; Zamborlini, V.; Guizzardi, G. An Ontological Analysis of the Electrocardiogram. RECIIS, Vol. 3, no. 1, 2009. ISSN 1981-6286.

Guizzardi, G.; Lopes, M.; Baião, F.; Falbo, R. On the importance of truly ontological representation languages, IJISMD, 2010. ISSN: 1947-8186.

Guizzardi, G.; Halpin, T., Ontological Foundations for Conceptual Modeling, Applied Ontology, 2008, ISSN 1570-5838.

Guizzardi, Giancarlo. 2005. Ontological Foundations for Structural Conceptual Models, Universal Press, Holanda, 2005. ISBN 1381-3617.

Halpin, T., Morgan, T., Information Modeling and Relational Dababases, Morgan Kaufman, 2008. ISBN 1558606726