7
Referências

ABRAHÃO, J. I. *Universidade de Brasília* **Reestruturação Produtiva e Variabilidade do Trabalho: Uma Abordagem da Ergonomia.** Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0102-37722000000100007&lng=es&nrm=iso&tlng=pt>. Acesso em: 20 jul. 2006 às 23h13min.

ALMEIDA, H. O.; COSTA, E.; PERKUSICH, A. **Desenvolvimento de Software para Sistemas Multiagentes** - XXV SBC, 2005.

BAUER, B.; MÜLLER, J. P.; ODELL, J. **Agent UML: Formalism for Specifying Multiagent Interaction** Springer-Verlag, Berlin, pp. 91-103, 2001.

BELLIFEMINE, F.; CAIRE, G.; POGGI, A.; RIMASSA, G. **JADE A White Paper**, 2003.

BOCIO, J.; MORENO, A.; VALLS, A. **Hospital Arrangements for a Transplant Operation using Agents**, 2001.

BOSCH, J. **Product-Line Architectures in Industry: A Case Study** – University of Karlskrona/Ronneby, 1998.

BRADSHAW, J. **An Introduction to Software Agents**. Software Agents, J. Bradshaw (ed.). AAAI/MIT Press, 1997.

BRUGALI, D.; SYCARA, K. **A Model for Reusable Agent Systems**. In: Implementing Application Frameworks: Object-Oriented Frameworks at Work. Fayad, Johnson, Schmidt (eds.) John Wiley & Co, 1999.

CAIRE, G. **JADE TUTORIAL JADE programming for Beginners**, 2003.

CHAVEZ, C. V. F. G. **Um Enfoque Baseados em Modelos para o Design Orientado a Aspectos.** Tese (Doutorado em Informática) – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 2004.

CHOREN, R.; LUCENA, C. J. P. **Modeling Multi-agent systems with ANote** Springer-Verlag 2004, 2003a.

CHOREN, R.; LUCENA, C. J. P. **The ANote Modeling Language for Agent-Oriented Specification**, 2003b.

CHOREN, R.; **Uma Linguagem de Modelagem para Sistemas Baseados em Agentes.** Tese (Doutorado em Informática) – Departamento de Informática,

Pontifícia Universidade Católica do Rio de Janeiro, 2002.

DURSCKI, R. C.; SPINOLA, M. M.; BURNETT, R. C.; REINEHR, S. S. **Linhas de Produto de Software: Riscos e Vantagens de sua Implementação** – VI Simpósio Internacional de Melhoria de Processos de Software, SIMPROS, 2004.

FAYAD, M. E. **Introduction to the Computing Surveys' Eletronic Symposium on Object-Oriented Application Frameworks** – University of Nebraska, 2000.

FAYAD, M. E.; JOHNSON, R. E. **Domain-Specific Application Frameworks: Frameworks Experience by Industry** – Wiley, 2000.

FIPA website. Foundation for Intelligent Physical Agents. Disponível em: <http://www.fipa.org>. Acesso em: 29 jun. 2006 às 18h00min.

FONTOURA, M.; PREE, W.; RUMPE, B. **UML-F: A Modeling Language for Object-Oriented Frameworks,** 2002. Disponível em: <www.almaden.ibm.com/cs/people/fontoura/papers/ecoop2000.pdf>. Acesso em: 03 fev. 2007 às 15h43min.

FONTOURA, M.; PREE, W.; RUMPE, B. **The UML Profile Framework Architectures** – Addison Wesley, 2002.

FONTOURA, M. **A Systematic Approach for Framework Development.** Tese (Doutorado em Informática) – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 1999.

FONTOURA, M. **Enhancing Framework Design and Utilization** – ACM'99 Student Research Contest (Graduate), 1999.

FROEHLICH, G.; HOOVER, H. J.; LIU, L.; SORENSON, P. **Requirements for Hooks Tools** – University of Alberta, 1998.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley, Reading, 1995.

GARCIA, A.; LUCENA, C.; CASTRO, J.; OMICINI, A.; ZAMBONELLI, F. (editors). **Software Engineering for Large-Scale Multi-Agent Systems**. Lecture Notes in Computer Science, vol. 2603, Springer-Verlag, April 2003.

GARCIA, A. F. **Objetos e Agentes: Uma Abordagem Orientada a Aspectos.** Tese (Doutorado em Informática) – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 2004.

GIORGINI, P.; KOLP, M.; MYLOPOULOS, J.; PISTORE, M. **The Tropos Methodology: An Overview**, 2000.

GRISS, M. L.; FONSECA, S.; COWAN, D.; KESSLER, R. **SmartAgent: Extending the JADE Agent Behavior Model** AOSE Workshop, 2002.

HENDERSON-SELLERS, B.; GIORGINI, P.; BRESCIANI, P. **Enhancing Agent OPEN with concepts used in the Tropos methodology** – 2003 Disponivel em www.dit.unitn.it/~pgiorgio/papers/esaw03.pdf Acesso em: 20 fev. 2007 às 21h27min.

HUGET, M. P.; ODELL, J. **Representing Agent Interaction Protocols with Agent UML** - AAMAS'04, 2004.

JENNINGS, N. R.; WOOLDRIDGE, M. **Agent-Oriented Software Engineering** - in Handbook of Agent Technology (ed. J. Bradshaw) AAAI/MIT Press, 2001. (to appear).

JENNINGS, N. **Agent-Oriented Software Engineering**. Proceedings of the 12th International Conference on Industrial and Engineering Applications of Artificial Intelligence, pp. 4-10, 1999.

JENNINGS, N.; WOOLDRIDGE, M. **Agent-Oriented Software Engineering**. Handbook of Agent Technology, J. Bradshaw (ed.). AAAI/MIT Press, 2000.

JENNINGS, N. **An Agent-Based Approach for Building Complex Software Systems** – communication of the ACM vol.44 no 4, 2001.

LUCENA, C.; GARCIA, A.; ROMANOVSKY, A.; CASTRO, J.; ALENCAR, P. **Software Engineering for Multi-Agent Systems II**. Lecture Notes in Computer Science, vol. 2940, Springer-Verlag, February 2004.

MATHIAS FILHO, I. **A Documentação e a Instanciação de Frameworks Orientados a Objetos.** Tese (Doutorado em Informática) – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 2002.

MATTSSON, M.; BOSCH, J.; FAYAD, M. E. **Framework Integration Problems, Causes, Solutions** – Communications of the ACM, 1999.

MAZZIA, C.; GANGULY, P.; KIDD, M. **Healthcare Applications based on Software Agents** – MEDINFO, 2001.

NORMARK, K. **Hooks and Open Points** - Aalborg University, 1994.

OLIVEIRA, T. C. **Uma Abordagem Sistemática para a Instanciação de Frameworks Orientados a Objetos.** Tese (Doutorado em Informática) – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 2001.

OLIVEIRA, T. C.; ALENCAR, P. S. C.; LUCENA, C. J. P.; COWAN, D. D. **RDL: A Language for Framework Instantiation Representation**, 2006.

PATE SANTOS, G. N. **Um Sistema Multi-Agentes de Controle de Prescrições Médicas**. Projeto Final de Programação (Mestrado em Informática) – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro,

2006. Disponível em: <http://www.inf.puc-rio.br/~gsantos/HTML/DocProjFinal.zip>. Acesso em: 20 fev. 2007 às 21h27min.

PEÑA, J.; HINCHEY, M. G.; RUIZ-CORTÉS A. **Multiagent System Product Lines: Challenges and Benefits** – CACM vol. 49 no. 12 dezembro 2006.

POGGI, A.; RIMASSA, G.; TURCI, P. **Engineering CoMMA Multiagent System with Agent UML** WOA Workshop, 2002.

RABELO JR, A.; ROCHA, A. R.; SOUZA, A. D.; XIMENES, A. A.; LOBO, N.; CARVALHO, D.; FILHO, J. W. C. S.; OLIVEIRA, K. M.; SOUZA, L. A.; WERNECK, V. M. **Um Sistema Especialista para Diagnóstico de Cardiopatias Isquêmicas**, 1993. Disponível em: <http://www.informaticamedica.org.br/informed/isquem.htm>. Acesso em: 10 jun. 2006 às 22h30min.

SARDINHA, J. A. R. P. **MAS-School e ASCYNC: Um Método e um Framework para Construção de Agentes Inteligentes.** Tese (Doutorado em Informática) – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 2005a.

SARDINHA, J. A. R. P.; CHOREN, R.; SILVA, V. T.; MILIDIÚ, R.; LUCENA, C. J. P. **A combine specification language and development framework for agent-based application engineering**. ScienceDirect The Journal of Systems and Software 79 pp 1565-1577, 2006.

SARDINHA, J. A. R. P.; MILIDIÚ, R. L.; PARANHOS, P. M.; CUNHA, P. M.; LUCENA, C. J. P. **An Agent Based Architecture for Highly Competitive Eletronic Markets**, 2005b.

SCHIMD, H. A. **Systematic Framework Design by Generalization** – Communications of the ACM, 1997.

SCHMITT, D. **A Framework Development Process for Product-Line Architectures** – Open University, 2000.

SILVA, V.; GARCIA, A.; BRANDÃO, A.; CHAVEZ, C.; LUCENA, C.; ALENCAR, P. **Taming Agents and Objects in Software Engineering**. In: Software Engineering for Large-Scale Multi-Agent Systems. A. Garcia, C. Lucena, J. Castro, A. Omicini, F. Zambonelli (eds.). Springer-Verlag: LNCS 2603, Berlin, April 2003.

TAC - Trading Agent Competition. **Game Description,** 2004. Disponível em: <http://www.sics.se/tac/page.php?id=3>. Acesso em: 14 jan. 2007 às 23h10min.

YU, E.; LIU, L. **Modelling Trust in the *i** Strategic Actors Framework**, 2001.

ZAMBONELLI, F.; JENNINGS, N.; WOOLDRIDGE, M. **Organizational Abstractions for the Analysis and Design of Multi-agent Systems**. In: Ciancarini, P., Wooldridge, M. (eds.): Agent-Oriented Software Engineering,

Springer-Verlag, 2001.

Anexo A -
Ciclo médico paciente

Para se ter idéia de como é o funcionamento de um hospital e facilitar o entendimento da aplicação CQPM, será apresentado o ciclo médico-paciente de áreas diferentes do CTI. Constantemente inclui-se no ciclo médico-paciente algum tipo de exame e/ou tratamentos como, por exemplo, exames laboratoriais (sangue urina, etc.), exames complementares (radiografias, ultra-sonografia, etc.), tratamento com fisioterapia, ou algum outro tipo de exame e/ou tratamento. Na figura abaixo pode ser entendido com mais clareza o ciclo de vida médico-paciente.

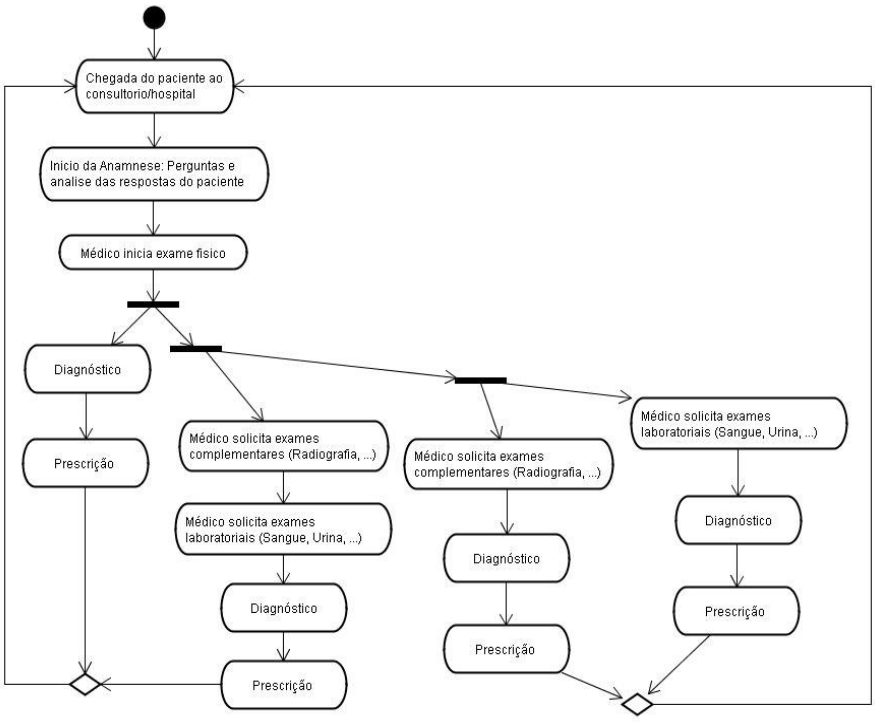

Figura 30. Ciclo de vida Médico-Paciente.

O escopo do sistema proposto é mostrado na figura a seguir:

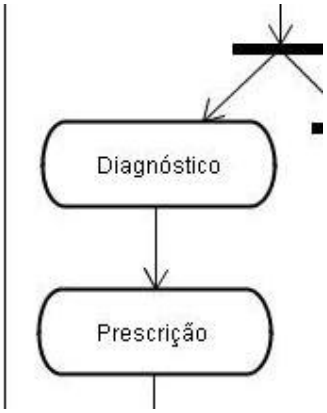

Figura 31. Escopo do sistema.

O diagnóstico de uma doença se desdobra em outros dois processos para que o diagnóstico esteja completo. A figura a seguir mostra o processo citado e seus subprocessos.

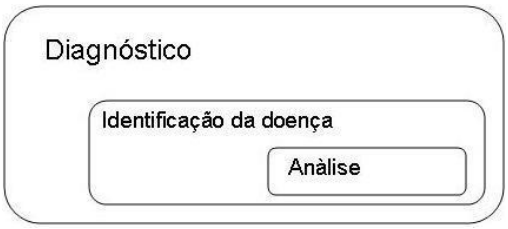

Figura 32. O processo de diagnóstico e seus subprocessos.

Dentro do processo de diagnóstico é necessário passar por dois processos: o primeiro de identificação da doença, onde o médico, através de exames, (físicos, laboratoriais e/ou complementares) consegue definir qual a doença em questão; no segundo processo é feita a analise do caso para então definir qual o estágio da doença.

Após o diagnóstico faz-se necessária à definição de um medicamento. O processo de prescrição provê três processos internos: um para a definição de um medicamento, onde o médico analisa e define qual deve ser o medicamento utilizado para o tratamento; neste processo é definida a dosagem do medicamento definido anteriormente, para isto o médico deve calcular qual a dosagem ideal para o paciente (este é o ponto que mostra toda a variabilidade que se deseja estudar); por último é feita à definição do tempo de uso do medicamento, onde o médico deve estipular o tempo necessário para a provável cura da doença. Com isto completam-se os processos de uma prescrição.

A seguir pode ser vista a figura que mostra o processo citado anteriormente e seus subprocessos.

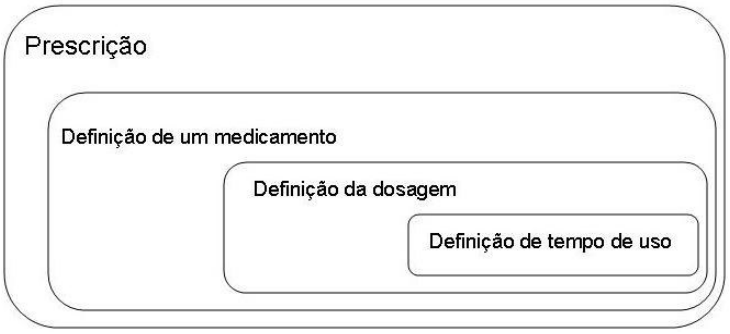

Figura 33. O processo de prescrição e seus subprocessos.

Uma das tarefas mais comuns realizadas pelos médicos é a prescrição medicamentosa, que exige do médico um alto grau de conhecimento técnico, sobre os medicamentos, além de aspectos relacionados ao paciente (como por exemplo, alergias) e um dos problemas mais comuns durante este processo é o erro em dosagem de medicamentos (principalmente em crianças).

Anexo B -
Modelagem CQPM com as linguagens apresentadas

Neste anexo são apresentadas as modelagens do agente dosador com as linguagens ANote, AUML e Topos.

A seguir será apresentada a modelagem em AUML de um agente da aplicação CQPM (“agente dosador”). A modelagem constará apenas do diagrama de protocolo já que a variabilidade em um agente software que é o que se pretende estudar se encontra apenas neste diagrama.

A figura a seguir mostra o “agente dosador” modelado com AUML.

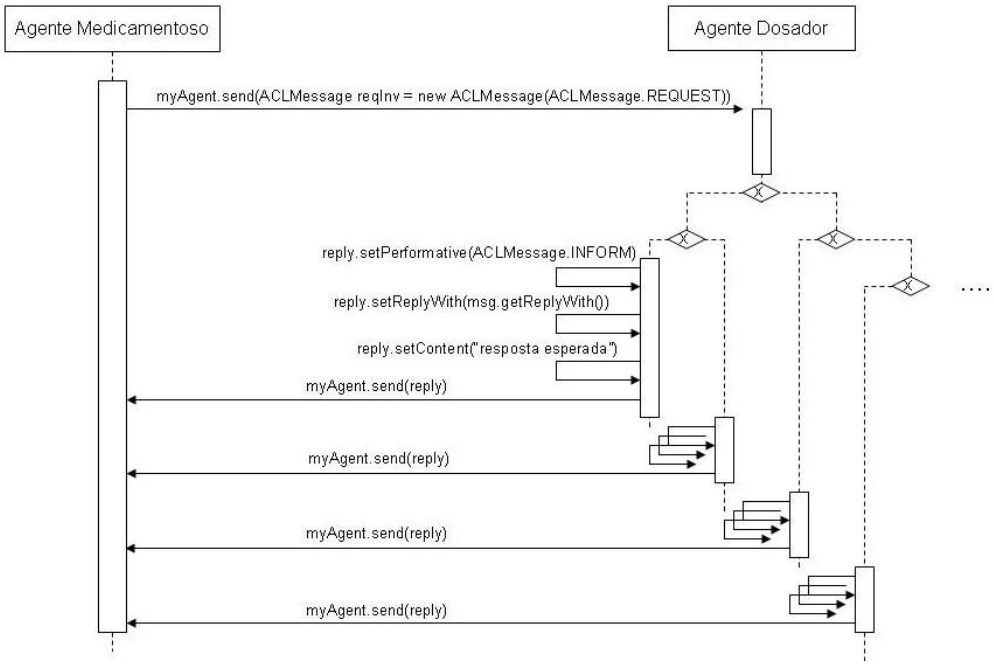

Figura 34. Modelo parcial de interação de protocolos da AUML para o agente dosador da aplicação estudo de caso CQPM.

A modelagem apresentada com a linguagem Tropos traz o diagrama de objetivo parcial do “agente dosador”.

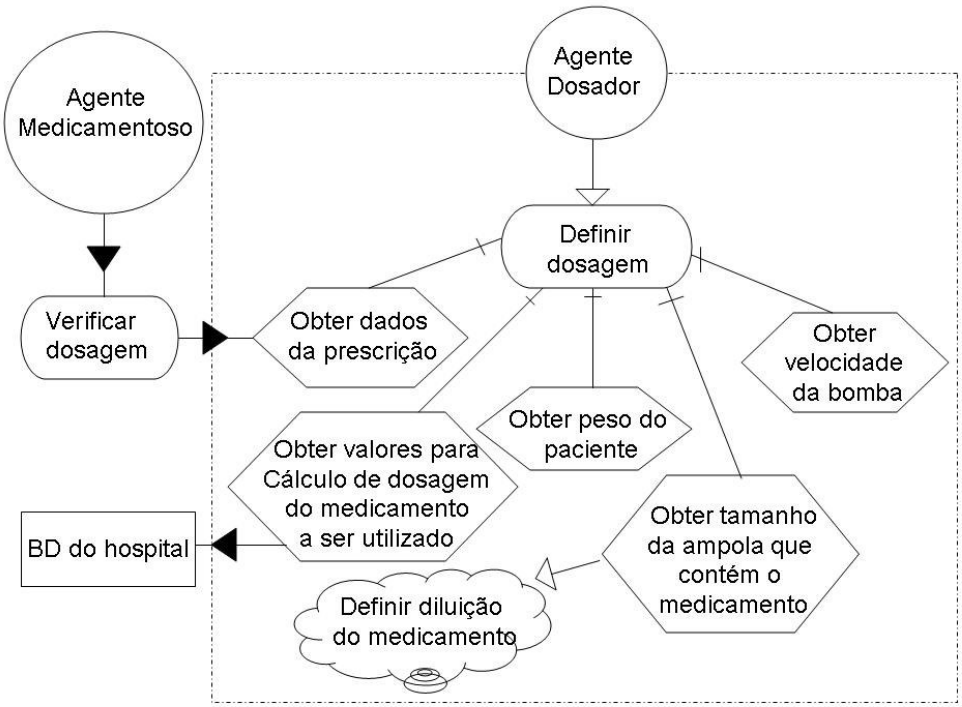

Figura 35. Modelo parcial de objetivo do agente dosador da aplicação estudo de caso CQPM.

A figura a seguir representa a modelagem com o ANote através dos diagrama *planning view* e o diagrama *scenario view* de um agente do sistema CQPM que será discutido com detalhes mais adiante.

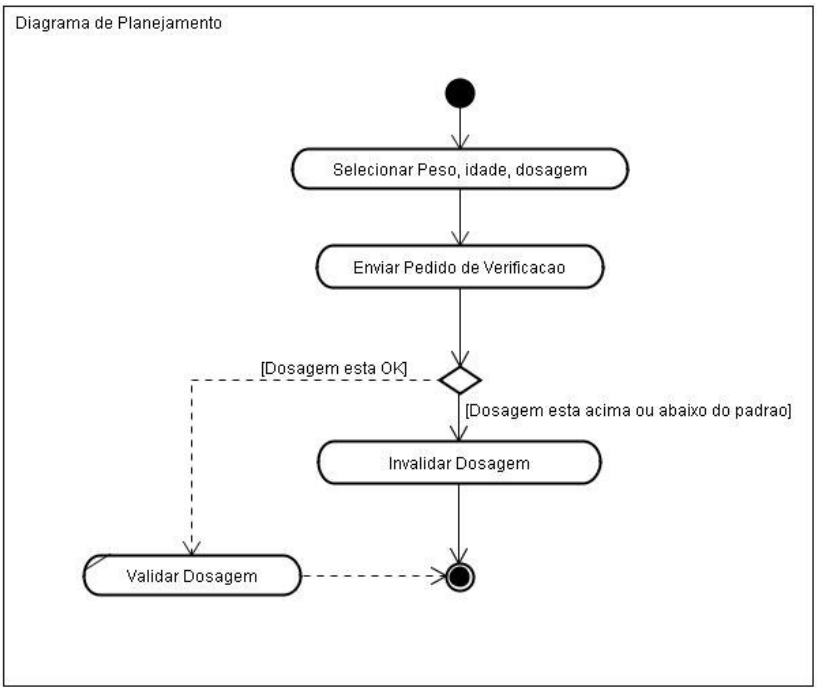

Figura 36. Modelo *planning view* do agente dosador com a linguagem ANote.

Tabela 27. Modelo *scenario view* do agente dosador com a linguagem ANote.

Invalidar/Validar Medicamento por Dosagem	
Lead Agent:	Dosador.
Precondition:	Existir um Medicamento e/ou um Princípio Ativo para validar.
Main Action Plan:	1. Selecionar Prescrição. 2. Calcular a dosagem e ver se está de acordo com a dosagem Padrão. 3. Invalidar Dosagem.
Interactions:	Medicamentoso.
Variant Plan:	Variant PreCondition: O Agente não encontra erros na dosagem. Plan Description:

	Se a dosagem está OK 1.1 Valida a Dosagem.

Anexo C -
Um Exemplo de Flexibilização de Ação

Um exemplo que pode ser estudado para o caso (de abstração de ações) é o de um sistema de compra. Dentro desse sistema pode ser observado um agente com o objetivo de realizar a compra de um produto. Agora suponha que a compra será feita em um supermercado e o produto a ser adquirido é um refrigerante.

O agente que tem como seu objetivo a compra de um refrigerante e tem a possibilidade de abstração de planos (refrigerante normal, com limão, *diet* e *light*) suponha que aescolha do agente seja comprar um refrigerante normal, ou seja, o plano dos refrigerantes normais. Após a escolha do refrigerante a ser adquirido faz-se necessário o pagamento do valor correspondente ao refrigerante. Neste ponto é fácil observar que a ação de pagar pode ser executada de maneiras distintas.

A possibilidade de formas distintas na execução de uma ação traz a idéia de abstração de ações, o que fica mais claro com a seguinte exemplificação.

A ação de pagar pode ser feita de diversas maneiras distintas, uma seria o pagamento em dinheiro, outra seria o pagamento em cheque e uma última seria o pagamento em cartão de crédito. Aqui fica fácil observar a variabilidade que ocorre neste agente, onde a ação de pagar será a ação abstrata e as suas instanciações serão feitas de acordo com as formas de pagamentos que devem ser aceitas no estabelecimento. Uma aplicação pode conter todas as formas de pagamento ou apenas algumas.

Por isto, o diagrama *planning view* do ANote deverá receber na ação “pagar valor devido” a restrição *{abstract}* como pode ser visto a seguir.

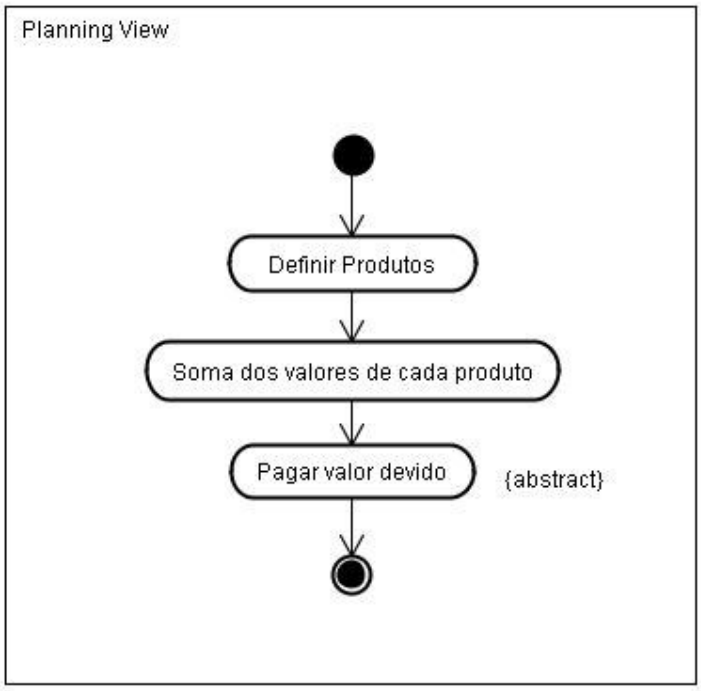

Figura 37. Diagrama *planning view* estendido para flexibilizar ações.

Como dito nas seções anteriores, os diagramas *planning view* e *scenario view* do ANote estão atrelados, por isto seguir pode-se ver o diagrama *scenario view*.

Tabela 28. Diagrama *scenario view* estendido para flexibilizar ações.

Comprar um Produto (refrigerante)	
Lead Agent:	Efetivador.
Precondition:	Existir um produto para ser adquirido.
Main Action Plan:	1. Obter o valor de cada Produto. 2. Somar o valor de cada produto. 3. Pagar valor devido. → **Flexibility (Action Creation)** 4. Validar pagamento.
Interactions:	Vendedor.
Variant Plan:	Variant PreCondition: O Agente encontra erros no modo de pagamento (cheque sem fundo, cartão sem limite, dinheiro abaixo do total da conta). Ou

	o a problema com o produto. Action Description: → **Flexibility (Action Creation)** Se teve problemas com o pagamento 1.1 Invalidar o pagamento da conta. Se teve problemas com o produto 2. Efetuar a troca do produto.

Todas as formas de pagamento exercem a mesma ação, que é a de pagar, porém de maneiras distintas e com a abstração da ação será possível dar uma maior coesão e clareza para a modelagem com relação ao código e possibilitando também a definição de uma linha de produto para o sistema multi-agente exemplificado.

A aplicação de compra de produtos traz uma ação abstrata, a ação de pagar e como foi observado existem diversas maneiras distintas de se pagar uma conta, uma seria o pagamento em dinheiro, outra seria o pagamento em cheque e uma última seria o pagamento em cartão de credito.

Nesta ação foi observado que há uma grande variabilidade na maneira de se executar esta ação, por isto essa ação pode ser definida como abstrata e as suas instanciações serão feitas de acordo com as formas de pagamentos que podem ser aceitas no estabelecimento que deseja adquirir a aplicação. Aqui fica claramente definida uma linha de produto, já que, é possível tem distintas aplicações onde uma aplicação pode conter todas as formas de pagamento ou apenas algumas e assim por diante.

A seguir vê-se o diagrama *instantiation view* sendo utilizado para guiar a instanciação da ação através de estruturas RDL, com a abstração da ação será possível dar uma maior coesão e clareza para a modelagem com relação ao código.

Tabela 29. Diagrama *instantiation view* guiando a instanciação de uma ação.

Instantiation View

Description:	ADD_CODE(Plano, ação, código) indica que determina ação num dado plano receberá a implementação descrita.

Code:	**COOKBOOK** Pagar Valor
	RECIPE main
	//adding a new action
	ADD_CODE(Refrigerante, pagarValorDevido, // código para implementar a compra com cheque);
	// …
	END_RECIPE;
	END_COOKBOOK