Tópicos de Engenharia de Software Orientada a Agentes

Lisiane Maria Bannwart Ambiel

***Abstract*— A análise de requisitos e teste são fases importantes do processo de desenvolvimento de sistemas. O interesse pelo paradigma de orientação a agentes para construção de sistemas complexos motiva a melhoria das metodologias existentes para garantir a implementação de sistemas com qualidade. O trabalho apresenta algumas propostas de melhoria das metodologias Gaia e OPM na fase de requisitos e para a metodologia Tropos na fase de testes.**

***Index Terms*— Metodologia Orientada a Agentes, Requisitos, Testes, Sistemas multi-agentes**

I. Introdução

O uso de aplicações distribuídas, concorrentes e com amplo uso de internet tornam os sistemas cada vez mais complexos. O grande interesse pelo paradigma de orientação a agentes para construção de sistemas complexos de software tem motivado propostas de melhoria e extensão das metodologias existentes com objetivo de tratar a fase de especificação e análise de requisitos com vistas à melhor qualidade dos sistemas.

O desenvolvimento de sistemas complexos envolve variados usuários e plataformas heterogêneas e necessita tratar engenharia de requisitos e técnicas de teste para garantir qualidade no processo de desenvolvimento e nos produtos, considerando teste como processo que dá suporte ao desenvolvimento e manutenção de sistemas.

O objetivo do presente trabalho é complementar o tópico "Engenharia de Software Orientada a Agentes" abordado durante o curso IA_009 apresentando propostas encontradas na literatura para extensão da metodologia Gaia na modelagem de requisitos, complementando a metodologia Tropos para tratar a fase de testes e da OPM/MAS para modelagem de sistema.

II. Metodologias - Melhorias na Modelagem e Tratamento de Requisitos

Observa-se que as linguagens de modelagem estão sendo melhoradas para incorporar características importantes para sistemas orientados a agentes assim como novas tentativas em criar linguagens mais simples e flexíveis.

A. Modelagem de Requisitos – extensão da metodologia Gaia[3]

O paradigma de orientação a agentes tem crescido nos últimos anos e muitas metodologias têm sido propostas para o projeto e desenvolvimento de sistemas multi-agentes (MAS - *multiagents systems*). Um dos pontos em aberto para este paradigma é o desafio de selecionar e modelar requisitos em termos de abstrações de agentes, tópico da área de engenharia de requisitos considerado com chave para desenvolvimento de sistemas de qualidade. A especificação de requisitos deve ser clara para qualquer usuário, a notação usada para representação deve ser útil para as próximas fases do desenvolvimento: análise, projeto e implementação.

A metodologia Gaia, que se inspira na metáfora organizacional, considera a complexidade de interações entre agentes, o projeto de estruturas organizacionais e os sistemas de controle para gerenciar o comportamentos dos agentes na organização [5]. No entanto, a metodologia Gaia não considera a fase de requisitos.

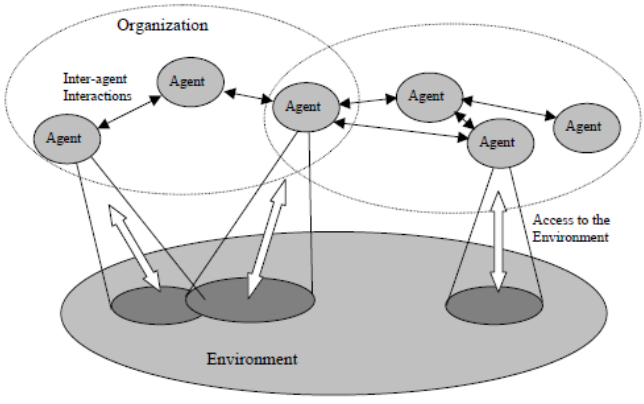

Figura 1 Visão Organizacional de um sistema multi-agentes - Metodologia Gaia

A figura mostra que a perspectiva organizacional da metodologia leva a uma caracterização genérica de arquitetura de sistemas multi-agentes. A medida que a complexidade aumenta o sistema é dividido em sub-organizações, sendo que um subconjunto de agentes pode ser envolvido em múltiplas

Enviado em 23 de junho de 2010.

Lisiane Maria Bannwart Ambiel é aluna do curso IA 009 – Introdução à Teoria de Agentes, da Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Brasil
(e-mail: lisiane_ambiel@hotmail.com)

organizações. Em cada organização, um agente pode assumir um ou mais papéis e interage com outros agentes para troca de conhecimento e coordenar suas atividades. Um sistema multi-agentes está tipicamente imerso em um ambiente com o qual os agentes interagem para atingir seus objetivos.

A metodologia Gaia trata das fases de análise, projeto arquitetural e projeto detalhado de sistemas multi-agentes, definindo seus próprios modelos e sua própria linguagem de modelagem. As fases são dispostas como um processo de desenvolvimento que parte de modelos conceituais em direção a modelos mais concretos, que possam ser implementados usando objetos.

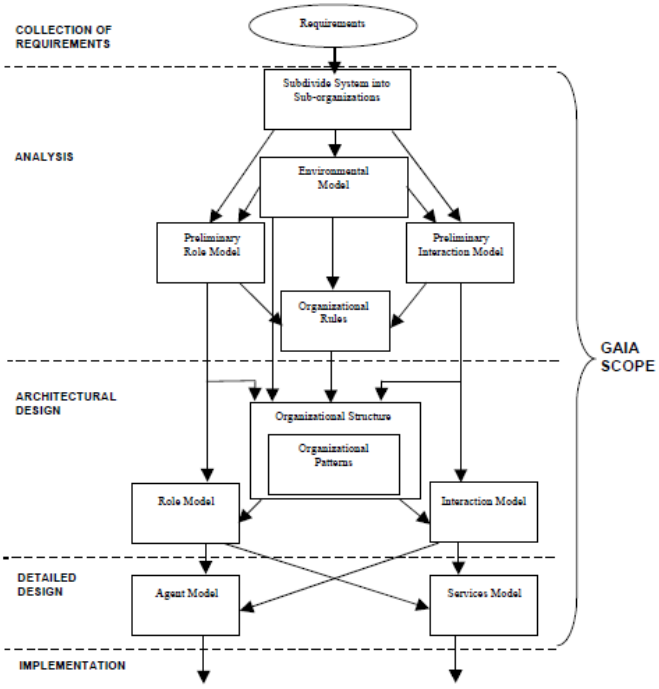

Figura 2 Modelos e fases do processo de desenvolvimento na Metodologia Gaia

A proposta de um método de análise e modelagem de requisitos apresenta quatro modelos, resultado da aplicação de outras duas metodologias (RETO e GBRAM) que não serão detalhadas neste trabalho. Os seguintes modelos são utilizados para a fase de requisitos:

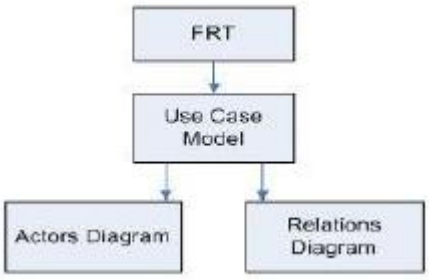

Figura 3 Os quatro modelos da fase de requisitos

Os diagramas dos diversos modelos são apresentados posteriormente usando um estudo de caso - Sistema de Gerenciamento de Conferência.

Árvore de Refinamento Funcional (FRT – *Functional Refinement Tree*): os objetivos do sistema são identificados e organizados em forma hierárquica, onde a raiz representa o objetivo global do sistema. Para facilitar a tarefa de identificar objetivos usa-se como estratégia a identificação de palavras chaves, onde as palavras que denotam a construção de algum tipo de tarefa, ação ou operação são consideradas candidatos a se tornar objetivos. Para cada grupo de objetivos é possível analisar obstáculos, cenários, restrições, relações de dependência.

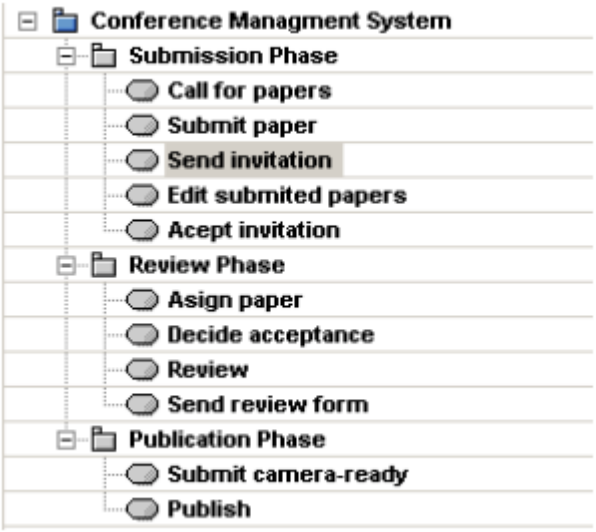

Figura 4 Exemplo do CMS – Conference Management System – Modelo FRT

Modelo de Caso de Uso (*Use Case Model*) – permitem a identificação e representação de atores – uma abstração importante para MAS, relacionando cada ator com o caso de uso (objetivo) o qual ele está responsável. O modelo é útil para representação de requisitos funcionais e comunicação com outros envolvidos (*stakeholders*). Um caso de uso equivale a um objetivo de um ator. Cada elemento funcional da FRT (folha) está representado por um caso de uso e respectivos cenários (básico e alternativos). Os recursos do ambiente que os agentes necessitam para atingir seus objetivos são identificados nos passos/ações de um cenário, que será detalhado para cada elemento da FRT.

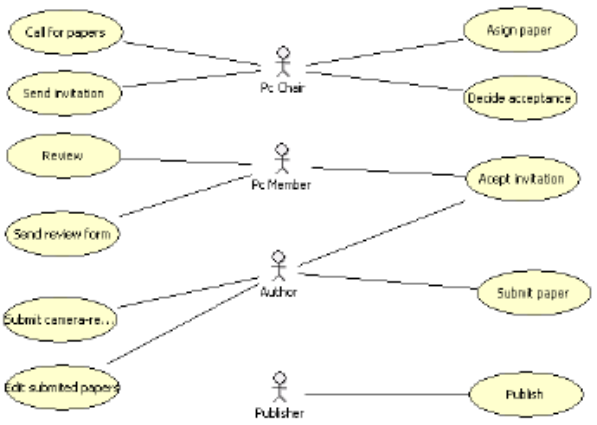

Figura 5 Exemplo do CMS – Use case diagram

N°	Step	Type	Resource
1	Request papers	(A/S C)	
2	Show papers	(S R)	Paper[i]
3	Select paper	(A/S C)	
4	Save selection	(S R)	
5	Request reviewers	(A/S C)	
6	Show participants reviewers	(S R)	
7	Select reviewer	(A/S C)	
8	Save the number of reviewer that are responsible for the paper	(S R)	
9	Verify if there are more papers to be assigned	(S R)	

Figura 6 Exemplo do CMS – *basic course scenario (Assign Paper use case)*

Diagrama de Atores (*Actors Diagram*) – atores externos (ou clientes) normalmente têm um parceiro dentro do sistema, que pode ser considerado um agente a ser implementado. Na proposta apresentada, um ator é uma entidade que demanda interesse por um objetivo ou é afetado pelo acontecimento de um objetivo. Um ator pode estar associado a vários objetivos e um objetivo pode estar associado a vários atores.

Diagrama de Relacionamentos (*Relations Diagram*) – representam restrições do sistema, o comportamento desejado e a seqüência do comportamento, identificados durante a fase de requisitos. Duas relações de dependência são definidas entre atores e objetivos: precedência ou restrição, onde a identificação de pré-requisitos se faz necessária.

Na metodologia Gaia, o primeiro passo – na fase de análise - é determinar se o sistema pode ser dividido em múltiplas organizações, que representam grupos de objetivos.

Na integração dos modelos de requisitos com os modelos da fase de análise da metodologia Gaia, temos as seguintes recomendações para mapeamento:

- criar uma sub-organização na fase de análise para cada grupo funcional do FRT (grupo de objetivos);
- o *Environment Model* em Gaia é derivado dos recursos identificados nos diversos cenários dos casos de uso;
- o *Preliminary Roles Model* em Gaia identifica os perfis básicos necessários para a organização atingir seus objetivos. Um papel define o comportamento do agente e tem associado objetivos e tarefas específicas, o que está relacionado ao modelo de atores. A recomendação é iniciar a fase de análise considerando como papéis os objetivos de um ator;
- o *Preliminary Interaction Model* em Gaia mostra atividades que envolve mais que um papel, o que necessita a definição de protocolos. A recomendação é verificar os cenários identificando as interações entre atores.
- o *Organization Rules* em Gaia é derivado do Diagrama de Relacionamentos, onde se observam as restrições, limites do sistema.

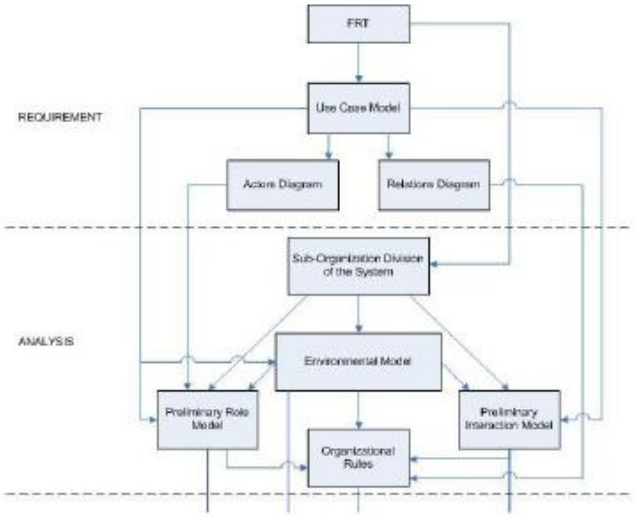

Figura 7 Integração Modelos de Requisitos com Metodologia Gaia

Os autores consideram como trabalho futuro: explorar a possível adequação da fase de modelagem de requisitos a outras metodologias orientadas a agentes.

B. Modelagem de sistemas multi-agentes – linguagem com base em objetos e processos[4]

Muitos pesquisadores e desenvolvedores reconhecem as vantagens na aplicação do paradigma de agentes no desenvolvimento de sistemas. Apesar de considerarem o uso de uma linguagem de modelagem como ponto chave para especificação do sistema desejado, ainda não existe um

consenso no uso de linguagem de modelagem para tais aplicações. Muitas linguagens de modelagem para sistemas multi-agentes (*Multiagent Systems* - MAS) tem sido desenvolvidas. Entre elas: ADELFE, ADEPT, AUML, CAMLE, DESIRE, FAF, Gaia, INGENIAS, MAS-CommonKADS, MaSE, MESSAGE, Prometheus e TROPOS. No entanto, nenhuma delas tem sido amplamente adotada.

O estudo identifica as seguintes características como sendo as possíveis falhas: acessibilidade – facilidade de codificação ou simplicidade de entender e usar a linguagem de modelagem; expressividade – capacidade da linguagem de modelagem em apresentar conceitos do sistema em diversos níveis de abstração e com vários aspectos: estrutural, comportamental e interações; e flexibilidade – capacidade de mudar ou ajustar a linguagem de modelagem de acordo com as necessidades do projetista ou da organização.

Na intenção de resolver tais limitações, os autores optaram por adaptar uma linguagem de modelagem existente – *Object-process methodology* (OPM) – uma abordagem integrada para o estudo e desenvolvimento de sistemas em geral e sistemas de software, mas também indicada para desenvolvimento de aplicações MAS. OPM usa um modelo gráfico e uma especificação textual gerada automaticamente; trata processos e objetos de forma equivalente, como classes de coisas, o que é considerado um requisito essencial na área de sistemas orientados a agentes, pois agentes são entidades com comportamentos. OPM usa uma ferramenta de diagramação com conjunto diagramas (OPD – *object-process diagram*), linguagem (OPL – *object-process language*) e descrição semântica que são entendidas por técnicos e não técnicos.

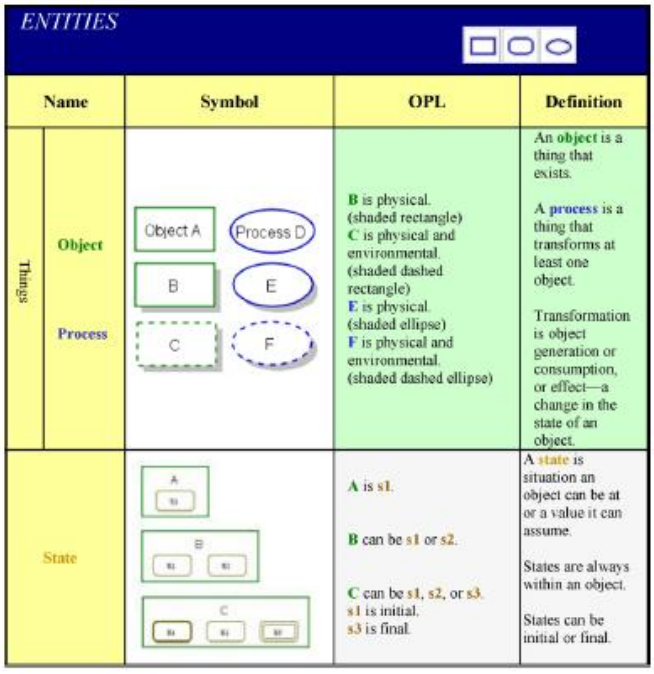

Figura 8 Entidades OPM (*things: object, process; state*) / Diagramas e texto OPL

A extensão da OPM, que ocorre pela adição de níveis intermediários, recebe o nome *Object Process Methodology/Multiagent Systems* (OPM/MAS), e compreende um conjunto de *building blocks* que podem ser definidos conforme a necessidade do usuário, o que aumenta a flexibilidade da linguagem. Os seguintes *building blocks* foram definidos como possibilidades para uso em MAS, considerados como elementos padrões: *Object, Agent, Environment, Message, Task, Role, Protocol, Mobilizing, Platform, Organization* e *User*.

OPM/MAS é definida considerando os seguintes níveis: M2 – metamodelo, forma genérica da OPM; M1.5 – metamodelo intermediário e opcional, que descreve os *building blocks* específicos no domínio de MAS (ex. onde estão definidas as noções de agente, protocolo); M1 – o modelo, que segue as regras especificadas nos níveis M1.5 e M2, que também vão servir para validação do modelo em M1; M0 – nível da aplicação, que é uma implementação do modelo M1.

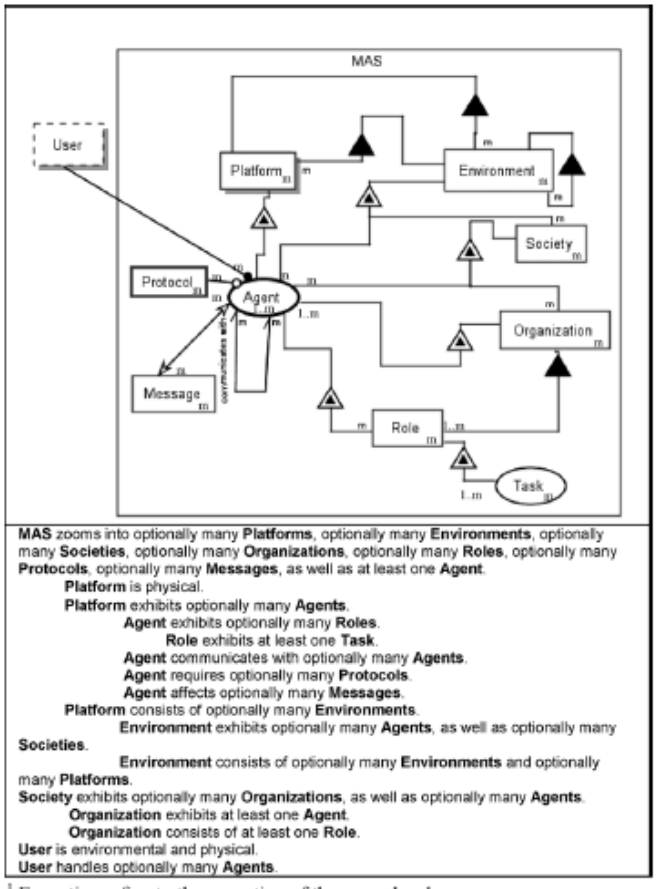

Figura 9 Diagrama do sistema e correspondente texto em OPL

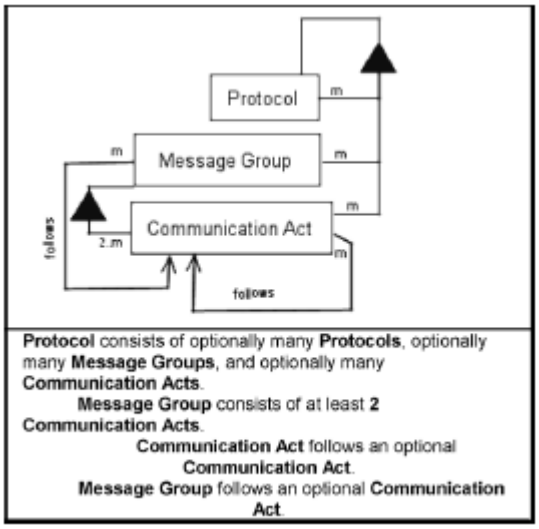

Figura 10 Diagrama do Protocol (*building block*) e correspondente texto em OPL

Na avaliação conduzida pelos autores, OPM/MAS apresentou deficiência no tratamento das fases de desenvolvimento, não faz referência, não distingue as mesmas e a documentação é insuficiente; a flexibilidade de incorporar novos *building blocks* é apontada como uma vantagem apesar de outras linguagens de modelagem usarem o conceito de metamodelo (ex. INGENIAS); também foi considerada de fácil uso e aprendizagem devido às características de expressividade e acessibilidade.

III. FASE DE TESTES

Aumenta o número de metodologias para desenvolvimento de sistemas complexos e distribuídos com base no paradigma de agentes, na forma de sistemas multi-agentes (*multiagent systems* - MAS). A maioria das metodologias cobrem as fases de análise e projeto, mas falham na descrição detalhada da implementação e fases de teste. Poucas metodologias oferecem verificação formal com base na especificação para permitir a correção de possíveis erros nas fases iniciais do processo de desenvolvimento.

A. A fase de testes nas metodologias orientadas a agentes[1]

A clássica definição de que "teste é um processo de executar um programa com a intenção de encontrar erros" evoluiu nos últimos tempos. O teste de software é visto como um processo completo que dá suporte ao processo de desenvolvimento e manutenção. Testes podem ser derivados de requisitos, especificações, artefatos de projeto ou do código. O modelo em V mostra que dependendo das atividades no ciclo de vida do projeto, diferentes tipos de testes podem ser definidos embora a execução dos mesmos só ocorra após a fase de implementação.

As metodologias orientadas a agentes tem foco maior nas fases de análise, projeto e implementação de sistemas multi-agentes (*multiagent systems* - MAS). Pouca atenção tem sido dada a testes e ferramentas para depuração (*debug*). Entretanto muitas das ferramentas que dão suporte a cada metodologia incluem algumas características que são relevantes aos testes.

Como exemplo, a Prometheus *Design Tool* (PDT) suporta debug de interações através de um agente de depuração que monitora as trocas de mensagens entre agentes e verifica os mesmos com relação aos protocolos. As violações de protocolos como falhas ao receber uma mensagem esperada ou a recepção de uma mensagem inesperada podem ser detectadas e explicadas. Também uma infra estrutura de teste unitário foi incorporada ao PDT que dá uma visão do processo de teste e mecanismos para identificar a ordem em que as unidades são testadas, geração de dados de entrada que formam os casos de teste.

A metodologia MaSE propõe depuração de interações baseada em verificação de modelo para suportar automaticamente a verificação de conversação de multi-agentes.

A metodologia Tropos tem uma infra estrutura de teste de agente chamada eCAT. Esta ferramenta suporta a derivação de casos de teste semi automático através de técnicas de geração de casos de teste (exemplo: geração de esqueleto de casos de teste orientados a objetivos a partir do diagrama de análise de objetivos).

B. Testes – infra estrutura para a metodologia Tropos [2]

É reconhecida a forte ligação entre a engenharia de requisitos e teste. O projeto de casos de testes no início do projeto, em paralelo com especificação de requisitos, ajuda a descobrir problemas nas fases iniciais evitando a implementação de especificação com erros. Além disso, requisitos bem especificados produzem melhores testes e especificação de teste prévia produz bons requisitos porque ajuda a esclarecer ambigüidades nos requisitos.

Esta ligação entre requisitos e testes é também chamada desenvolvimento direcionado a testes. Neste tipo de abordagem os casos de testes são produzidos a partir dos requisitos antes da implementação dos mesmos.

Os autores propõe uma infra estrutura de teste considerando a ligação entre requisitos e casos de teste, seguindo o modelo em V (V-Model). A proposta considera a metodologia Tropos que adota uma abordagem dirigida a requisitos. Tropos usa a noção de agente e conceitos relacionados (objetivos, planos, tarefas, etc) durante as fases de desenvolvimento. Objetivos são classificados em fortes (hardgoals) – funcionais, e leves (softgoals), não funcionais.

O modelo em V é uma representação do processo de desenvolvimento de sistemas que estende o tradicional modelo em cascata.

No modelo em V, o lado esquerdo representa as fases de desenvolvimento e o lado direito representa as fases de teste, momento em que o sistema é testado com base em

especificação definida no lado esquerdo.

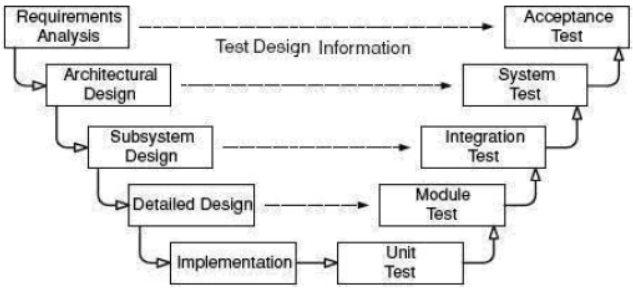

Figura 11 Atividades de desenvolvimento e níveis de testes no Modelo em V

A metodologia Tropos guia os desenvolvedores a construir um modelo conceitual que é refinado de forma incremental desde a fase de requisitos iniciais até a implementação.

A integração da fase de testes na metodologia Tropos com base no modelo em V, propõe uma visão das fases de desenvolvimento num ramo superior e as fases de testes num ramo inferior. Também propões uma maneira sistemática de derivar casos de teste a partir dos artefatos (modelos, diagramas) resultantes em cada fase de desenvolvimento.

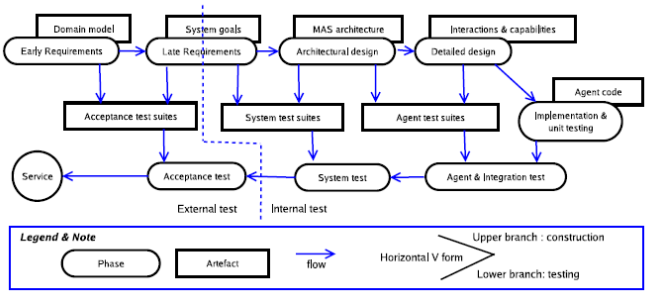

Figura 12 Modelo em V proposto para Tropos

Os artefatos produzidos durante o processo de desenvolvimento são: modelo de domínio/fase de requisitos iniciais, gráfico de objetivos do sistema/fase de requisitos finais, modelo de arquitetura do sistema (interações entre agentes)/fase de projeto arquitetural, especificação de agentes, papéis, capacidades, interações /fase de projeto detalhado e código do agente/fase de implementação.

Duas fases de testes são definidas: teste externo, executado após a entrega, os casos de teste são especificados pelos analistas juntamente com os interessados/envolvidos (*stakeholders*); teste interno, executados antes da entrega, os casos de teste são especificados com base nos produtos/artefatos de cada fase. A derivação dos casos de teste é conduzida ao mesmo tempo em que desenvolvedores especificam ou projetam o sistema o que ajuda os mesmos a refinar seu projeto e identificar defeitos.

IV. Conclusão

As fases de especificação de requisitos e testes, de grande importância na Engenharia de Software, estão sendo introduzidas ou incorporadas nas metodologias de orientação a agentes. Os trabalhos na área deixam claro que existe muito a ser feito no sentido de complementar o que é proposto, ou mesmo de verificar a utilização de técnicas em conjunto como o exemplo do estudo de OPM/MAS com outras metodologias como Gaia, Prometheus ou Tropos, a adequação da modelagem de requisitos a outras metodologias que ainda não tratam esta fase no ciclo de desenvolvimento. Outros propostas são apresentadas na literatura como definição de requisitos de qualidade e extensão de metodologia para suportar cooperação entre agentes e que não foram abordadas neste trabalho. Na área de testes, observa-se a necessidade de desenvolver métodos ou técnicas que sejam direcionados a características próprias dos agentes como pro atividade e autonomia.

Referencias

[1] MORENO, M.; PAVÓN, J.; ROSETE, A. Testing in Agent Oriented Methodologies. Book Series Lecture Notes in Computer Science, vol. 5518 LNCS, nPART 2, p 138-145, 2009, Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, Ambient Assisted Living - 10th Int. Work-Conf. Artificial Neural Networks, IWANN 2009 Workshops, Proceedings Disponível em: < http://www.springerlink.com/content/g617408558411t45/fulltext.pdf> Acesso em: 18 jun.2010.

[2] NGUYEN, D.C.; PERINI, A.; TONELLA, P. A Goal-Oriented Software Testing Methodology. Lecture Notes in Computer Science, vol. 4951 LNCS, p 58-72, 2008, Agent-Oriented Software Engineering VIII - 8th International Workshop, AOSE 2007, Revised Selected Papers Disponível em: < http://www.springerlink.com /content / vj9k7855378827w4/ fulltext.pdf> Acesso em: 17 jun.2010.

[3] RODRIGUES, L.; HUME, A.; CERNUZZI, L.; INSFRAN, E. Improving the Quality of Agent-Based Systems: Integration of Requirements Modeling into Gaia. Quality Software, 2009. QSIC '09. 9th International Conference on , vol., no., pp.278-283, 24-25 Aug. 2009 Disponível em: < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp= &arnumber= 5381442&isnumber=5381320 > Acesso em: 17 jun.2010.

[4] STURM, A.; DORI, D.; SHEHORY, O. An Object-Process-Based Modeling Language for Multiagent Systems Systems. Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on , vol.40, no.2, pp.227-241, March 2010 Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber= 5398876&isnumber=5415839> Acesso em: 16 jun.2010.

[5] ZAMBONELLI, F; JENNINGS, N.R.; WOOLDRIDGDE M. Developing multiagent systems: The Gaia methodology. ACM Transactions on Software Engineering and Methodology (TOSEM) archive, vol.12 , no. 3, pp.317-370, July 2003 Disponível em: < http://delivery.acm.org/10.1145/960000/958963/p317-zambonelli.pdf?key1 =958963&key2=1181527721&coll=GUIDE&dl=GUIDE&CFID=92596 426&CFTOKEN=57151885> Acesso em: 22 jun. 2010.

BIBLIOGRAFIA

[6] ALHASHEL, E.; BALACHANDRAN, B.M.; SHARMA, D. Extending Prometheus with Agent Cooperation. Computational Intelligence for Modelling Control & Automation, 2008 International Conference on , vol., no., pp.912-918, 10-12 Dec. 2008 Disponível em: < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp = &arnumber =5172747 &isnumber=5172580 > Acesso em: 17 jun. 2010

[7] ABDELAZIZ T.; ELAMMARI, M.; UNLAND, R. A Framework for the Evaluation of Agent-Oriented Methodologies. Innovations in Information Technology, 2007. IIT '07. 4th International Conference on , vol., no., pp.491-495, 18-20 Nov. 2007 Disponível em: <http://ieeexplore.ieee.org/ stamp/stamp.jsp?tp =&arnumber=4430391& isnumber =4430362> Acesso em: 17 jun. 2010