3 – Trabalhos Relacionados

Este capítulo apresenta um estudo comparativo sobre alguns modelos de organização de sistemas multi-agentes já existentes, destacando suas principais características e fazendo uma correlação com o modelo contratual que estamos propondo.

3.1 Organização de Sistemas Multi-Agentes

A definição do que é uma organização ainda não é um consenso entre os pesquisadores. Há diferentes abordagens de organização, algumas das quais serão descritas nas próximas seções. (Lameitre et al, 1998) propõem dois tipos básicos de organizações: uma que possui a visão centrada no agente e a outra na organização:

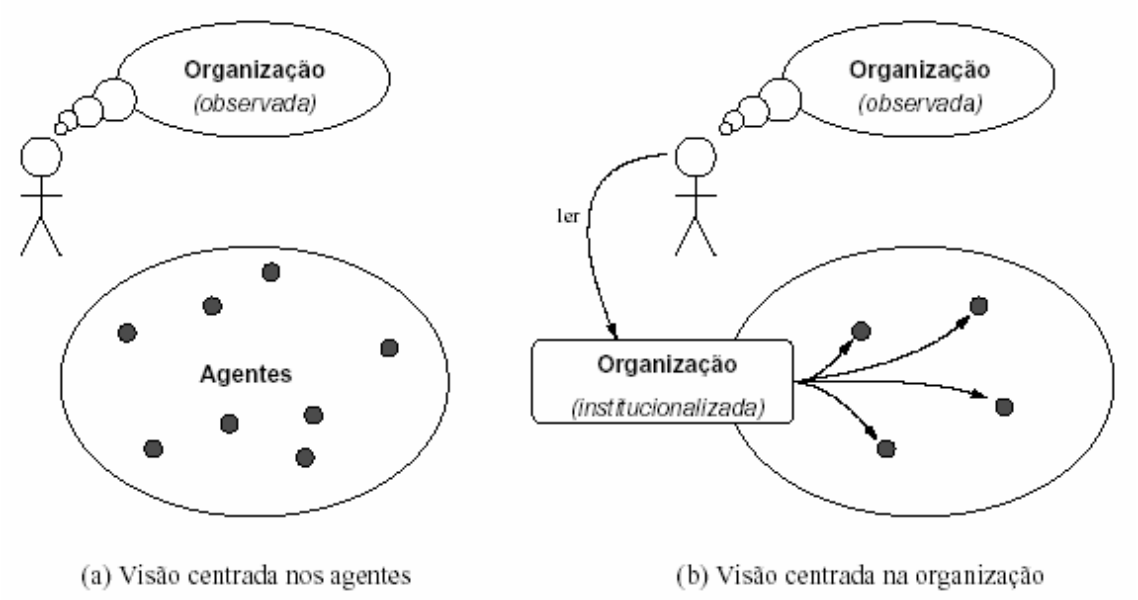


Figura 4: Organização de sistemas multi-agentes (Lameitre et al, 1998).

Na primeira abordagem (visão centrada nos agentes), o SMA não possui uma representação explícita de sua organização, tal representação está distribuída nos seus agentes. Um observador ou um agente da sociedade somente pode inferir uma descrição subjetiva da organização, isto é, uma

descrição construída por ele mesmo a partir da observação do comportamento dos agentes de tal sociedade. Esta descrição subjetiva da organização pode ser chamada de *organização observada* (Hubner et al, 2002).

No segundo ponto de vista (visão centrada na organização), a organização existe objetivamente, isto é, o observador pode obter uma descrição da organização que a sociedade está adotando sem precisar observar seu comportamento ou mesmo considerar os agentes que a compõem, Esta descrição é chamada de *organização institucionalizada*. Por exemplo, em uma escola a descrição de sua organização existe fora dos agentes na forma de manuais de procedimentos, organogramas, etc (Hubner et al, 2002).

(Hubner et al, 2002) destaca as organizações com visão centrada na organização como mais adequadas em vários domínios de aplicação, baseando-se nas seguintes propriedades das organizações centradas na organização:

- Uma organização existe por mais tempo que os agentes, e sua memória deve ser preservada independentemente dos agentes;

- Não é tarefa simples obter uma descrição para a organização a partir dos seus agentes, principalmente no caso de agentes reativos;

- Separando-se conceitualmente a organização da arquitetura interna do agente, torna-se possível raciocinar sobre a organização sem considerar a estrutura e o funcionamento dos agentes, isto é, pode-se projetar organizações em um nível de maior abstração. Por esta razão, esta visão é mais adequada em sistemas multi-agentes abertos, onde novos agentes podem entrar no sistema e não se conhece a arquitetura de tais agentes *a priori.*

A descrição explícita da organização de um SMA, necessária na visão centrada na organização, é feita segundo um modelo organizacional (uma “forma de ver” a organização), que determina, por exemplo, o que constitui a organização. Nas próximas seções são apresentados alguns modelos desta abordagem organizacional.

3.1.1 Modelo LGI – *Law-Governed Interation* (Minsky et al, 2000)

(Minsky et al, 2000) propõe um mecanismo de coordenação e controle para sistemas heterogêneos e distribuídos. Este mecanismo é baseado em quatro princípios, descritos a seguir:

1. As políticas de coordenação precisam ter o seu cumprimento garantido - regras de coordenação possuem um conjunto de regras de engajamento (políticas de coordenação) que precisam ser obedecidas por todos os participantes de uma atividade. A implementação desta política precisa garantir a aplicação destas regras. Quando isto é feito por um grupo de pessoas que trabalham de forma cooperativa, os membros podem desenvolver os componentes de forma a cumprir as especificações. Entretanto, esta hipótese não pode ser utilizada em sistemas abertos, onde as pessoas possuem razões para não confiarem uma nas outras. Então um mecanismo que imponha o cumprimento das regras é desejável.

2. Este cumprimento precisa ser descentralizado - se a imposição for feita de forma centralizada, introduz-se dois problemas: torna-se o sistema não escalável e introduz-se um perigoso ponto de falha no sistema. A idéia da imposição descentralizada é fortemente baseada na crença que uma política de coordenação pode ser realizada sem o conhecimento dos outros agentes.

3. As políticas de coordenação precisam ser formuladas explicitamente - torna-se difícil fazer com que grupos de agentes operem sob a mesma política se elas estão implícitas no código dos envolvidos ou se são expressas por diferentes formalismos.

4. Deve ser possível realizar o *deployment* da lei de forma incremental – em sistemas de larga escala, se o *deployment* não puder ser feito de forma incremental, sem impactar as partes do sistema que não operam sob a lei, então dificilmente o mecanismo irá ser utilizado.

Os agentes que interagem através deste mecanismo são governados por uma lei L que especifica um conjunto de regras de engajamento. Estas regras são ativadas por vários tipos de eventos, entre eles, mensagens enviadas e recebidas, obrigações, entre outros.

3.1.2 Modelo de Regulação de Leis proposto por (Paes et al, 2005a)

O modelo proposto por (Paes et al, 2005a) baseia-se no trabalho de governança de leis proposto por (Minsky et al, 2000). (Paes et al, 2005a) trata conceitos como cenas, normas e restrições de forma integrada. Esse modelo propõe uma linguagem declarativa, denominada XMLaw, para a especificação da interação de acordo com elementos do modelo conceitual e uma infra-estrutura de software que age como mediador das interações garantindo que elas estejam de acordo com as leis de interação especificadas.

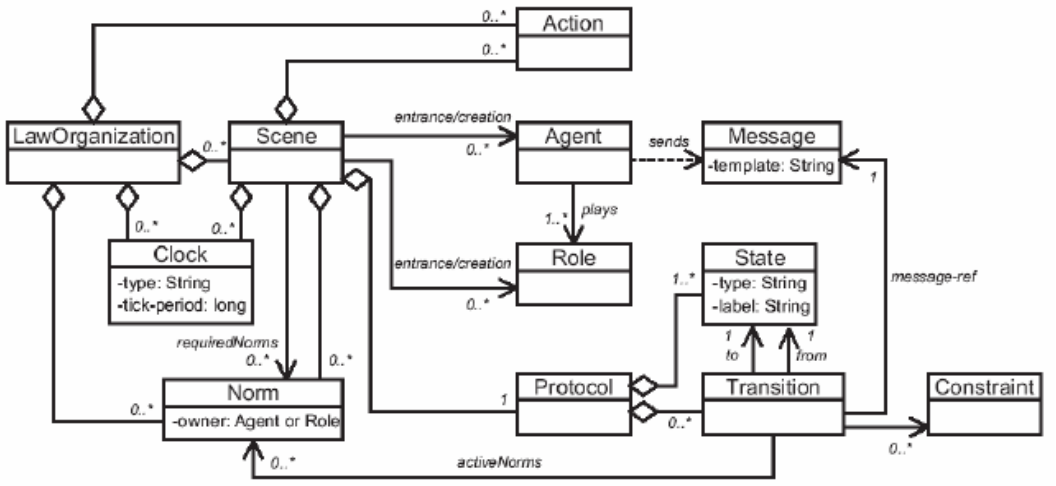


Figura 5: XMLaw: Modelo Conceitual proposto por (Paes et al, 2005a).

O modelo conceitual fornece um elemento denominado Relógio (Clock) que provê os meios para especificar leis sensíveis ao tempo. Um relógio é capaz de gerar eventos em intervalos de tempo específicos. Por exemplo, um relógio permite a ativação e desativação de normas após um determinado período de tempo ter se esgotado. Além disso, um relógio pode ser ativado por transições ou mesmo por normas.

Normas (Norms) descrevem quais comportamentos dos agentes são permitidos, quais são obrigados e quais são proibidos. As normas são geralmente adquiridas pelos agentes durante o decorrer das interações, e conseguem representar noções de compromissos adquiridos e cumpridos. Por exemplo, em um processo de negociação, um agente pode assumir o

compromisso (obrigação) de pagar por uma mercadoria negociada e, enquanto essa obrigação não for cumprida, o agente fica impedido de participar de novas negociações.

O modelo conceitual utiliza a abstração de cenas para auxiliar na organização e modularização das interações, e são representadas pelo elemento do modelo conceitual *Scene*. Este elemento especifica quais agentes e quais os papéis de agentes podem interagir em uma cena, ou mesmo dar início a sua execução. Além disso, uma cena é composta por um protocolo de interação e por um conjunto de normas, ações e relógios.

Estes elementos compartilham um contexto comum de interação definido pela cena. Isto significa que uma norma definida no contexto de uma cena é somente visível naquela cena. Para que um agente inicie ou participe de um protocolo de interação de uma cena, é necessário que ele desempenhe algum papel (*role*) previamente definido na organização e especificado no contexto da cena.

Além do modelo conceitual apresentado, existe um *framework*, chamado M-Law, que garante a regulação das interações dos agentes através dos elementos do modelo conceitual e do modelo de interação baseado em eventos. Este mecanismo intercepta as mensagens enviadas por agentes e verifica sua conformidade com as leis (Figura 6).

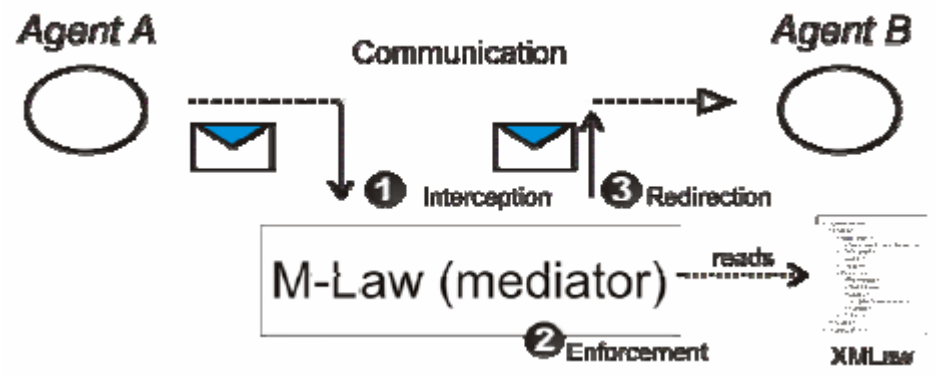


Figura 6: Arquitetura M-Law (Paes et al, 2005a).

Para que o *framework* M-Law pudesse prover os meios para efetivamente suportar a linguagem XMLaw e sua evolução, ele foi desenvolvido com quatro módulos (Figura 6). O módulo *Agent* contem as classes que os desenvolvedores de agentes devem utilizar para implementá-los. Este módulo por sua vez, utiliza o módulo de Comunicação para enviar e receber mensagens.

O Mediador engloba os outros dois módulos, Evento e Componente, e também utiliza o módulo de Comunicação. Os módulos de Evento e Componente não são visíveis para os desenvolvedores de agentes, mas podem ser estendidos para evoluir as funcionalidades do Mediador.

O modulo de Evento implementa a notificação e propagação de eventos. E o módulo de Componente define um conjunto de classes abstratas e concretas, além de interfaces, que permitem que novas funcionalidades sejam inseridas. De uma forma geral, os componentes implementam o comportamento dos elementos da linguagem XMLaw.

Para especificar as Leis em XMLaw é necessário descrever elementos de leis como os papéis, normas, relógios, cenas, protocolos e mensagens. Cada cena especifica o protocolo de interação que os agentes devem seguir e, conseqüentemente, quais são as mensagens válidas. Os passos para a especificação de leis são:

1. Criação da organização (*LawOrganization*) e estabelecimento das normas (*Norms*), ações (*Actions*), restrições (*Constraints*) e relógios (*Clocks*) que podem existir nela.
2. Criação das possíveis cenas (*Scenes*) nas quais os agentes desempenhando os papéis especificados podem interagir dentro da organização.
3. Definição do identificador de cada cena e, caso necessário, do seu tempo máximo de execução.
4. Definição dos protocolos com as suas mensagens, transições e estados possíveis de acordo com uma máquina de estados.

3.1.3 Modelo Organizacional AALAADIN

No modelo AALAADIN, proposto por (Ferber et al, 1998), a organização é definida como um conjunto de grupos que possuem uma determinada estrutura. Cada grupo contém um conjunto de papéis necessários ao seu funcionamento e um conjunto de agentes membros (Figura 7). Os papéis são representações abstratas para as funções que os agentes disponibilizam. Nenhuma restrição é feita quanto à arquitetura interna dos agentes: um agente é simplesmente considerado como uma entidade ativa e comunicativa que assume papéis nos grupos onde é membro.

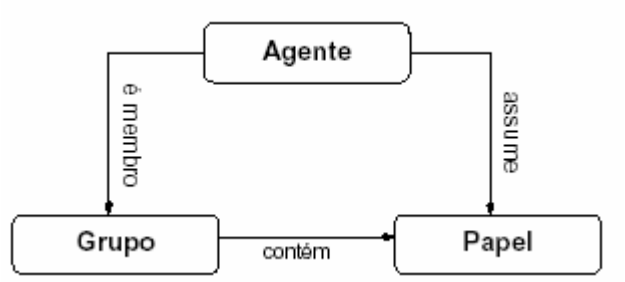


Figura 7: Modelo AALAADIN de organização, segundo (Ferber et al, 1998).

Esta abordagem também sugere que exista uma entidade capaz de controlar a entrada dos agentes aos grupos. Esta entidade pode usar qualquer estratégia para decidir sobre a aceitação ou rejeição de um dado agente em um grupo, por exemplo, ela pode perguntar se ele tem uma determinada competência , e em caso de resposta afirmativa ela deixa o agente entrar.

Os relacionamentos entre os papéis dentro dos grupos são definidos indicando-se a linguagem de interação utilizada entre os papéis, e qual o papel que vai iniciar a interação. Os relacionamentos entre os grupos são definidos, quando se define a estrutura da organização. Isso se dá quando existe um agente que desempenha um papel *R1* em um grupo e ao mesmo tempo desempenha um papel *R2* em outro grupo.

Esta abordagem não é adequada para sistemas abertos, pois precisa que os agentes sejam cooperativos. Além disso, a noção de grupos é vista como uma entidade atômica, ou seja, não existem subgrupos. Desta forma, não é possível o uso de várias camadas de abstração para a modelagem e projeto do sistema.

3.1.4 Modelo Organizacional GAIA

A metodologia Gaia (Wooldridge et al, 2000) é aplicável a um grande conjunto de sistemas multi-agentes, além de lidar com aspectos de nível macro (sociedade) e de nível micro (agente) dos sistemas. Gaia é baseada na visão de que um sistema multi-agentes se comporta como uma organização computacional que consiste de vários papéis interagindo. Ela permite que um analista vá sistematicamente do estabelecimento de requisitos até um projeto que seja suficientemente detalhado a ponto de ser implementado diretamente. Além disso, Gaia toma emprestada parte da terminologia e notação da análise e projeto orientados a objetos.

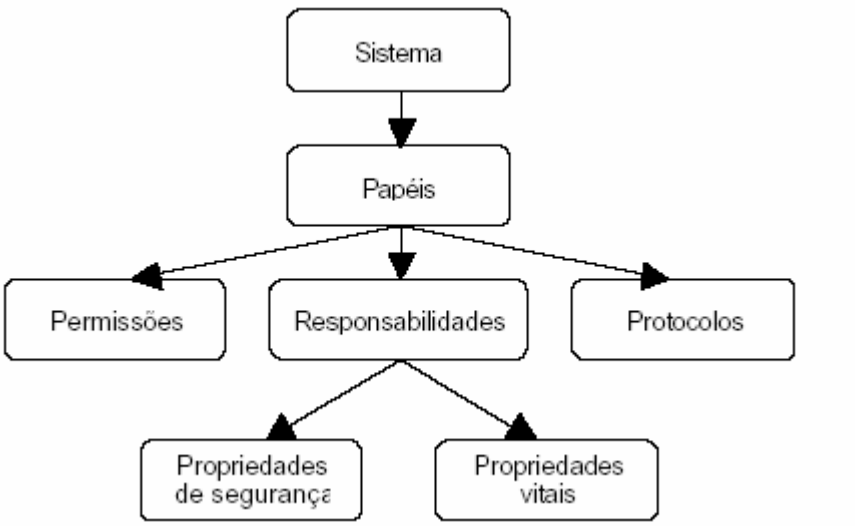


Figura 8: Modelo GAIA de organização, segundo (Wooldridge et al, 2000).

Em particular, Gaia encoraja um projetista a pensar na construção de sistemas baseados em agentes como um processo de projeto de uma estrutura organizacional. Seus principais conceitos podem ser divididos em duas categorias: abstratas e concretas. Entidades abstratas são aquelas usadas durante a análise para conceituar o sistema, mas que não têm necessariamente qualquer realização direta no sistema. Os conceitos abstratos (Figura 8) incluem Papéis, Permissões, Responsabilidades, Protocolos, Atividades, Propriedades Vitais e Propriedades de Segurança. As entidades concretas, por outro lado, são usadas no processo de projeto e tipicamente terão realização direta no sistema executável. Os conceitos concretos incluem Tipos de Agente, Serviços e Conhecimentos.

O objetivo do estágio de análise é o de desenvolver um entendimento do sistema e a sua estrutura (sem referência a nenhum detalhe de implementação). No caso de Gaia, este entendimento é capturado na organização do sistema. Uma organização é vista como uma coleção de papéis, que se mantêm em certos relacionamentos uns com os outros. Além disso, estes papéis participam de padrões sistemáticos e institucionalizados de interação com outros papéis. Assim, o modelo de organização Gaia é compreendido de dois outros modelos (Figura 9): o modelo de papéis, que identifica os papéis chave no sistema; e o modelo de interações, que identifica as dependências e os relacionamentos entre os vários papéis numa organização multi-agentes.

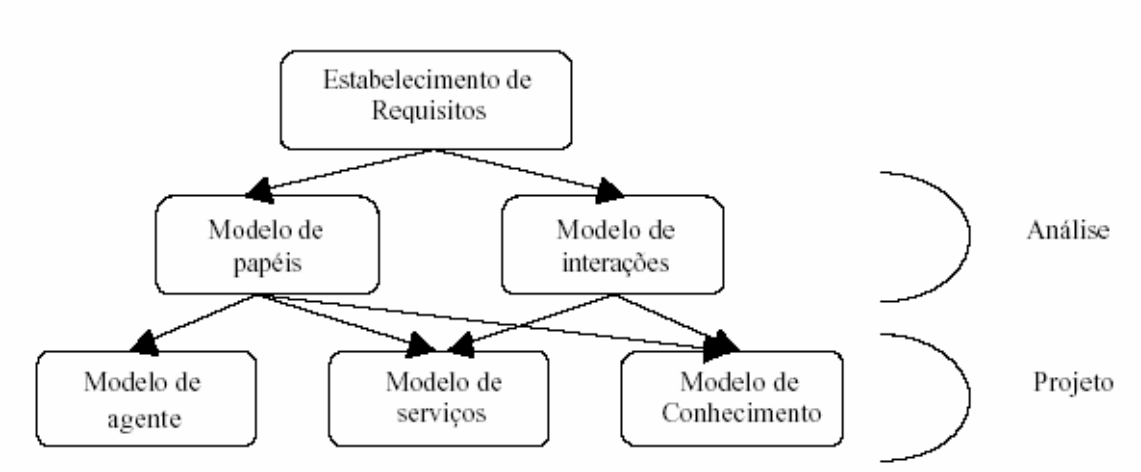


Figura 9: Divisão em camadas do modelo GAIA.

A idéia de um sistema como uma sociedade é útil quando pensamos sobre o próximo conceito: papéis. Um papel é definido por quatro atributos: responsabilidades, permissões, atividades e protocolos. As responsabilidades determinam funcionalidades e, conseqüentemente, são os atributos chaves associados a um papel. Elas são divididas em dois tipos: propriedades vitais (liveness) e propriedades de segurança (safety). As propriedades vitais intuitivamente estabelecem que “alguma coisa boa aconteça”. Elas descrevem aquelas situações que um agente pode provocar em determinadas condições ambientais. Em contraste, propriedades de segurança são invariáveis. Intuitivamente, uma propriedade de segurança estabelece que “nada de ruim aconteça” (isto é, que uma situação aceitável seja mantida durante todos os estados da execução). Um papel tem um conjunto de permissões necessárias para realizar suas responsabilidades. Permissões são “direitos” associados a um papel. Elas identificam os recursos que estão disponíveis para aquele papel de modo que ele possa realizar suas responsabilidades. Permissões tendem a ser recursos de informação. As atividades de um papel são computações associadas a ele. Elas podem ser realizadas por um agente sem que ele precise interagir com outros agentes. Finalmente, um papel também é identificado por vários protocolos que definem a maneira com que ele interage com outros papéis.

O processo de projeto Gaia envolve a geração de três modelos (Figura 9). O modelo de agente identifica os tipos de agente que vão formar o sistema e as instâncias de agente geradas a partir destes tipos. O modelo de serviços identifica os principais serviços que são requisitados para realizar o papel do

agente. Por fim, o modelo de conhecimento documenta as linhas de comunicação entre os diferentes agentes.

A metodologia Gaia está preocupada em como uma sociedade de agentes coopera para realizar as metas do sistema. Na verdade, como um agente realiza seus serviços está além do escopo de Gaia e irá depender do domínio da aplicação.

3.1.5 Modelo Organizacional MESSAGE/UML

No modelo MESSAGE, Metodology for Engineering Systems of Software Agents, proposto por (Caire et al, 2001) , uma organização é vista como um conjunto de agentes trabalhando de forma cooperativa para alcançar um objetivo comum. MESSAGE cobre a análise e o projeto de sistemas multi-agentes usando conceitos bem definidos e uma notação baseada em UML.

Os conceitos da UML são usados para modelar as entidades MESSAGE em um nivel detalhado (ou micro). Isto é, de um ponto de vista estrutural eles são objetos com atributos e operações realizadas por métodos. Já de um ponto de vista comportamental eles são máquinas de estado. A maioria dos conceitos de nível de conhecimento, definidos em MESSAGE, se divide nas seguintes categorias: Entidade Concreta, Atividade e Entidade de Estado Mental. Os principais tipos de Entidade Concreta são: Agente, Organização, Papel e Recurso. Um Agente é definido como uma entidade autônoma atômica que é capaz de executar alguma função útil. A capacidade funcional é capturada pelos serviços do Agente. Uma organização é um grupo de Agentes que trabalham juntos para algum propósito comum. Um papel descreve as características externas de um Agente em um contexto particular. A distinção entre Papel e Agente é análoga a que existe entre Interface e Classe na orientação a objetos. Um recurso é usado para representar entidades não-autônomas como bancos de dado ou programas externos usados por Agentes.

Os conceitos de Interação e Protocolo de Interação são emprestados da Metodologia Gaia. Uma Interação possui mais de um participante e um propósito que os participantes visam atingir. Um Protocolo de Interação define um padrão de troca de mensagens associado a uma Interação.

Meta é definida como um tipo de Entidade de Estado Mental. A Meta associa um Agente a uma Situação. Uma Situação é um nível de conhecimento análogo ao de um estado do mundo.

Conceitos comportamentais da UML como Ação, Evento e Estado são usados para definir as propriedades, interações e processos físicos da sua visão do mundo. A figura 10 proporciona uma visão geral informal centrada em agente de como estes conceitos são inter-relacionados.

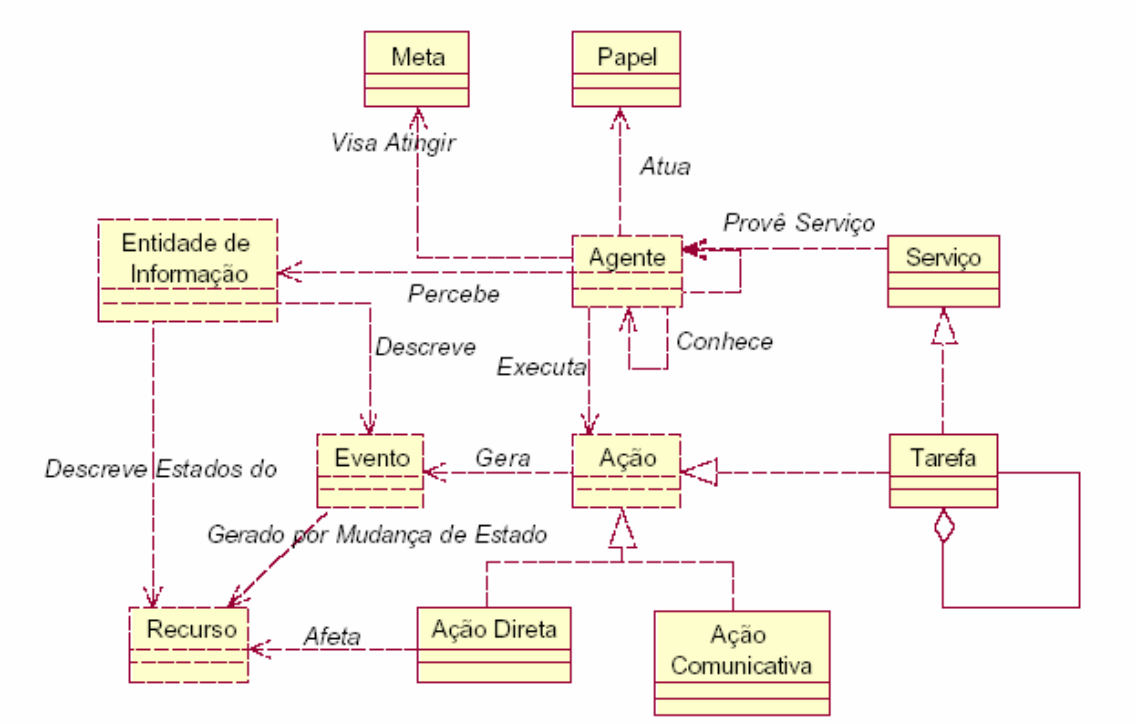


Figura 10: Modelo de organização MESSAGE, segundo (Caire et al, 2001).

A abordagem MESSAGE não adota a noção de leis globais para o bom funcionamento do sistema. Nesta metodologia, as leis precisariam ser implementadas pelos desenvolvedores dos agentes, o que dificulta seu uso em sistemas abertos baseados em leis.

3.1.6 Modelo Organizacional Tove

No modelo organizacional Tove, proposto por (Fox et al, 1998), uma organização consiste de várias divisões e subdivisões, um conjunto de agentes alocados nestas divisões, um conjunto de papéis que os agentes assumem e um conjunto de metas.

Papéis são definidos como protótipos de funções a serem desempenhadas pelos agentes na organização. As seguintes propriedades são associadas aos papéis:

- Um conjunto de metas a serem atingidas pelos agentes que assumiram o papel;
- Um conjunto de processos que definem como as metas podem ser alcançadas. Estes processos são descritos como uma estrutura de atividades;
- Um conjunto de permissões que o agente necessitará para alcançar suas metas, tais como permissões de alocação de recursos, entre outras;
- Um conjunto de habilidades que o agente deve possuir ao assumir o papel;
- Um conjunto de restrições na execução dos processos;
- Por fim, um conjunto de recursos necessários para o correto desempenho do papel assumido pelo agente.

O modelo ainda define uma relação entre papéis, obedecendo às seguintes regras:

- hierarquia: um papel pode ser subordinado a outro;
- especialização: um papel pode especializar outro papel e herdar os direitos, obrigações, permissões, etc, deste ultimo.

Um agente pode assumir um ou mais papéis e se comunicar com outros agentes, caso haja uma ligação de comunicação entre eles. Os agentes também realizam atividades e podem assumir determinados recursos na utilização de atividades, porém sujeitos a determinadas restrições de comportamentos.

Eventualmente, agentes podem formar times para executar tarefas especificas. Um time tem um tempo de vida menor que uma divisão, sua existência termina quando a tarefa para o qual foi criado for concluída.

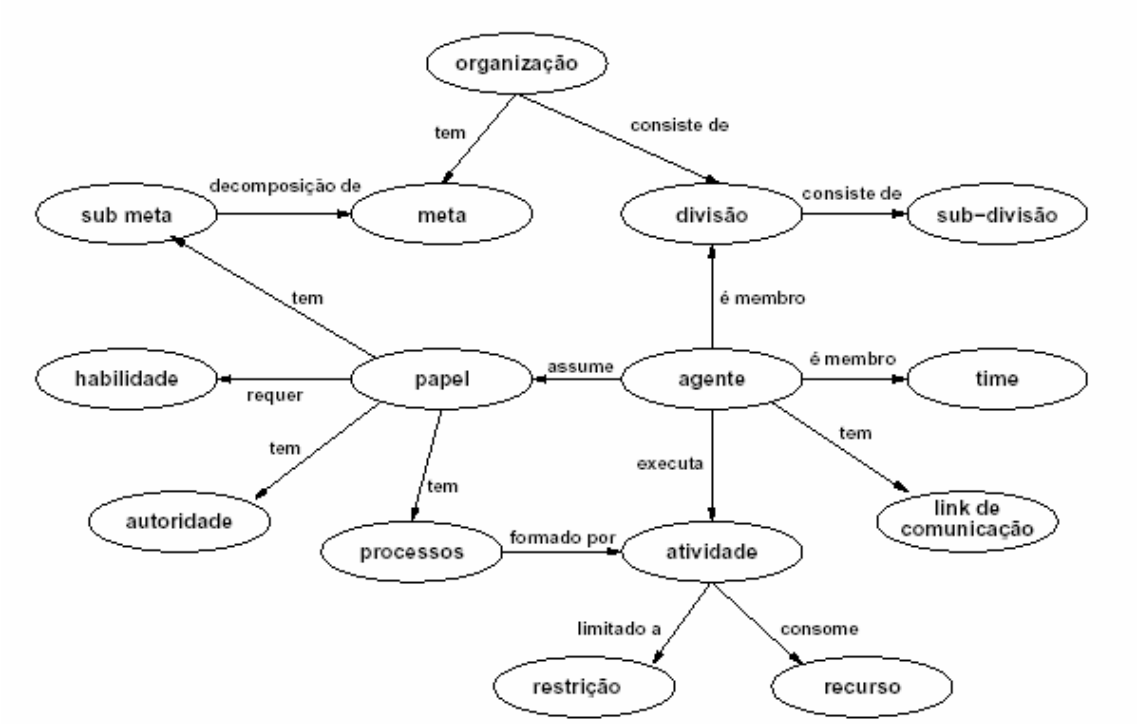
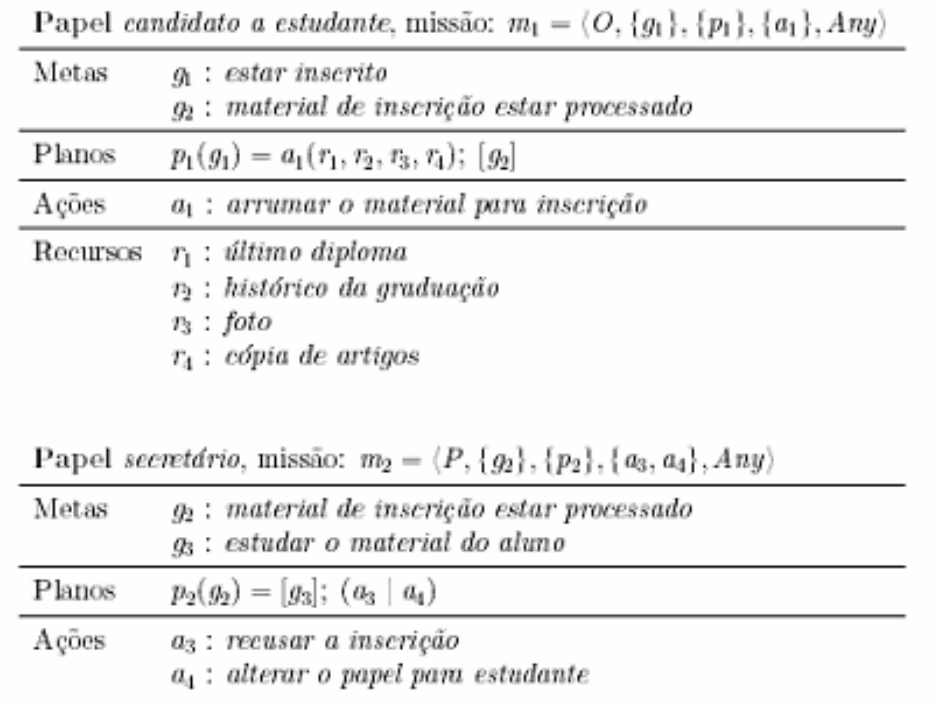


Figura 11: Modelo de organização Tove, proposto por (Fox et al, 1998).

3.1.7 Modelo Organizacional Moise

O modelo MOISE (Model of Organization for Mult-Agent System), proposto por (Hannoun, 2002), considera características organizacionais (chamadas de nível organizacional), tarefas e responsabilidades dos agentes (nível individual). Para a descrição do nível individual, o conceito de papel social é utilizado. Segundo o autor, um papel é formado por um conjunto de missões e cada missão pelos seguintes atributos:

- Marcação deontica que indica se a missão é obrigatória (denotado por O) ou facultativa (denotado por P) do papel,
- Um conjunto de metas a serem alcançadas,
- Um conjunto de planos que podem ser utilizados,
- Um conjunto de ações que podem ser utilizadas, e
- Um conjunto de recursos que podem ser usados.



Papel *candidato a estudante*, missão: $m_1 = \langle O, \{g_1\}, \{p_1\}, \{a_1\}, Any \rangle$	
Metas	g_1 : *estar inscrito* g_2 : *material de inscrição estar processado*
Planos	$p_1(g_1) = a_1(r_1, r_2, r_3, r_4); [g_2]$
Ações	a_1 : *arrumar o material para inscrição*
Recursos	r_1 : *último diploma* r_2 : *histórico da graduação* r_3 : *foto* r_4 : *cópia de artigos*

Papel *secretário*, missão: $m_2 = \langle P, \{g_2\}, \{p_2\}, \{a_3, a_4\}, Any \rangle$	
Metas	g_2 : *material de inscrição estar processado* g_3 : *estudar o material do aluno*
Planos	$p_2(g_2) = [g_3]; (a_3 \mid a_4)$
Ações	a_3 : *recusar a inscrição* a_4 : *alterar o papel para estudante*

Figura 12: Descrição de uma missão no modelo MOISE, proposto por (Hannoun, 2002).

No exemplo da figura 12, dois papéis são definidos, cada um com uma única missão. O agente que assume um papel de candidato a estudante é obrigado a realizar a missão m_1 devendo alcançar a meta g_1 e pode utilizar para este fim somente o plano p_1, a ação a_1 e qualquer recurso (Any). No caso, o agente pode realizar a ação a_1 com os recursos necessários (r_1 , r_2, r_3, r_4) para iniciar a execução do plano p_1 que atinge a meta de sua missão. Para que o plano seja completamente realizado, a meta g_2 deve ser alcançada, porém o agente não tem permissão para tentar realizar esta segunda meta. Neste caso, o agente precisa da ajuda de outro agente (um secretário, por exemplo) para completar seu plano e atingir a meta.

No nível social, há três tipos de ligações que restringem a interação entre papéis. São elas:

- comunicação: dois papéis que possuem uma ligação de comunicação podem trocar mensagens. Este tipo de ligação também estabelece o protocolo de comunicação que deve ser seguido caso os dois papéis venham a interagir;

- autoridade: indica que um papel tem o direito, por exemplo, de requisitar um serviço a outro;

- conhecimento: um agente somente pode conhecer (representar mentalmente) um outro agente se um dos seus papéis possuir uma ligação de conhecimento com um dos papéis de outro agente.

As ligações e os papéis sociais formam a estrutura organizacional da sociedade. Esta estrutura organizacional, mais um conjunto de agentes nela alocados formam uma Entidade Organizacional, como ilustra a figura 13:

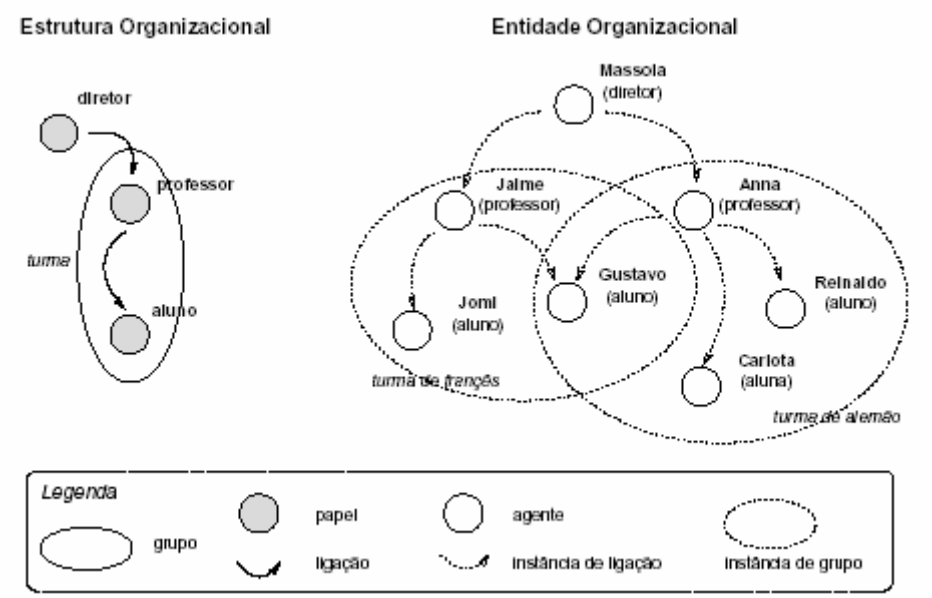


Figura 13: Entidade organizacional no modelo MOISE.

A estrutura organizacional do lado esquerdo da figura é instanciada criando a entidade representada do lado direito. Neste exemplo, os quatro agentes que possuem o papel de aluno aceitam também uma ligação com o agente de papel professor a partir do momento em que passam a fazer parte de um grupo (turma de francês, por exemplo).

3.1.8 Comparação entre modelos organizacionais

Atualmente existem diversos modelos de organizações para sistemas multi-agentes. Tais modelos podem ter sua visão centrada no agente, na organização ou até mesmo apresentar uma visão mista.

O modelo de regulação de leis através da linguagem declarativa XMLaw, (Paes et al, 2005a), utiliza cenas, normas e restrições de forma integrada. O foco do modelo está em regular as interações entre agentes através da interceptação das mensagens trocadas e sua verificação com as normas estabelecidas. Um

mediador das interações entre agentes garante que as mensagens trocadas estejam de acordo com as especificações.

O modelo AALAADIN, (Ferber et al, 1998), tem como elemento central os papéis assumidos pelos agentes na organização. Um papel é visto como função social, enquanto que uma organização é um conjunto de grupos. O funcionamento da sociedade é definido internamente nos agentes, por isso sua dificuldade em utilização em sistemas abertos.

Na metodologia Gaia (Wooldridge et al, 2000) é feita uma importante distinção entre as fases de análise e projeto. Ela fornece um conjunto de modelos para ser utilizado em cada uma destas fases. Com ela, é possível construir uma sociedade de agentes, definindo os papéis e as capacidades de cada agente individual, e como esta sociedade será estruturada.

Na metodologia MESSAGE (Caire et al, 2001) a arquitetura do SMA é definida através de um modelo organizacional focado na estrutura da organização e no relacionamento entre os agentes que ela contém. O modelo relacional descreve mecanismos de resolução de conflitos e normas que habilitam o grupo de agentes a funcionar como uma unidade servindo a um propósito comum. Os agentes são identificados a partir de um modelo orientado a metas, onde as metas da organização são decompostas e associadas a tarefas. A decomposição de metas é feita de forma recursiva, até que a tarefa associada a metas possa ser executada integralmente por um agente isolado ou em colaboração com outros agentes. Esta modelagem utiliza uma abordagem de BDI (*Believes, Desires, Intentions*) para a modelagem da estrutura de aprendizado e conhecimento.

O modelo Tove (Fox et al, 1998) define uma estrutura organizacional formada por divisões e subdivisões, agentes, papéis e metas sociais. Seu elemento central são os papéis assumidos pelos agentes, que são representados por um conjunto de metas, processos, autoridades (permissões), habilidades e recursos. Papéis são relacionados hierarquicamente. O funcionamento da sociedade é determinado por metas sociais e pelos processos associados aos papéis.

O modelo MOISE (Hannoun, 2002) utiliza uma estrutura organizacional formada por grupos, papéis e ligações entre papéis. Papéis são definidos por um conjunto de missões e podem ser relacionados entre si (autoridade, comunicação e conhecimento). As missões (obrigações ou permissões para metas, planos, ações e recursos) determinam o funcionamento da sociedade.

Na literatura encontramos diversas abordagens para organizações SMA, como as apresentadas neste estudo. Porém, uma dificuldade comum encontrada por estas abordagens está na garantia de que agentes, entidades autônomas e independentes, estão de fato cumprindo o papel assumido na sociedade.

Este trabalho apresenta uma proposta para modelagem SMA que adota o conceito de contratos para a criação de sistemas fidedignos, cujo foco seja prover uma organização na qual agentes que não respeitem seus papéis na sociedade sejam facilmente identificados.

3.2 Organização Contratual

3.2.1 Teoria Design By Contract

Podemos considerar contratos individuais (Silva, 2005), entre um agente e o sistema, ou contratos envolvendo múltiplos agentes, neste caso o contrato é tido como uma forma de negociação entre agentes. Em ambos casos, cada parte espera alguns benefícios do contrato e está preparada para cumprir algumas obrigações para obtê-los. Geralmente, o que é uma obrigação para uma das partes é um benefício para a outra (Meyer, 1992). Um contrato protege ambas partes, o cliente, especificando o quanto deverá ser realizado, e o fornecedor, delimitando o escopo do serviço a ser oferecido (Meyer, 1992).

	Obligations	Benefits
Client	must ensure *Q*	entitled to expect *R*
Supplier	must ensure *R*	entitled to expect *Q*

Figura 14: Contrato proposto por Meyer, 1992.

A teoria Design by Contract (Meyer, 1992) propõe o uso de asserções para representar contratos para componentes em paradigma orientado a objetos. Algumas asserções, denominadas precondições e pós-condições, são aplicadas a rotinas individuais, através da declaração da rotina. Outras, as invariantes de classe, restringem todas as rotinas de uma dada classe.

3.2.2 Contratos para Componentes – Beugnard et Al

(Beugnard et al, 1999) propõe uma divisão em quatro classes de contratos para componentes de softwares: contratos básicos ou sintáticos,

comportamental, sincronização e quantitativos. No primeiro nível, contratos básicos (ou sintáticos) são requeridos apenas para o funcionamento do sistema. No segundo nível, contratos comportamentais melhoram o nível de confiabilidade em um contexto seqüencial. No terceiro nível, contratos de sincronização melhoram a confiabilidade em um contexto distribuído e processamento concorrente. No quarto nível, contratos de qualidade quantificam a qualidade do componente de software.

3.2.2.1 Contratos Básicos

Contratos básicos permitem que o projetista de componentes especifique:

- as operações que um componente pode desenvolver;
- os parâmetros de entrada e saída que cada componente requer;
- as possíveis exceções que talvez sejam arremessadas durante a execução do componente.

Em tempo de compilação, verificações estáticas garantem que os clientes estão utilizando corretamente a interface do componente.

```
Interface BankAccount {
  void deposit(in Money amount);
  void withdraw(in Money amount);
  Money balance();
  Money overdraftLimit();
  void setOverdraftLimit(in Money
    newLimit);
}
```

Figura 15: Contrato básico, proposto por (Beugnard et al, 1999).

No exemplo da figura 15, podemos realizar as operações *deposit, withdraw* e *setOverdraftLimit* em um banco, bem como obter algumas informações de uma conta, como *balance e overdraftLimit.*

3.2.2.2 Contratos Comportamentais

Contratos comportamentais fazem o uso de assertivas lógicas, denominadas pré e pós-condições por (Meyer, 1992), para especificar o comportamento das operações:

```
Interface BankAccount {
  void deposit(in Money amount) {
    Require amount > 0;
    Ensure balance() = balance()@pre + amount;
  }
  void withdraw(in Money amount){
    Require amount > 0 and
      amount <= balance() + overdraftLimit();
    Ensure balance() = balance()@pre - amount;
  }
  Money balance();
  Money overdraftLimit();
  void setOverdraftLimit(in Money newLimit){
    Require balance() >= - newLimit
  }
  Invariant balance() >= - overdraftLimit()
}
```

Figura 16: Contrato comportamental, proposto por (Beugnard et al, 1999).

3.2.2.3 Contratos de Sincronização

Contratos de sincronização especificam o comportamento global dos objetos levando em consideração a sincronização entre as chamadas dos métodos. O foco deste contrato é descrever as dependências entre os serviços providos por um componente, como seqüência, paralelismo ou concorrência.

Políticas de sincronização podem ser usadas para descrever este tipo de contrato. O contrato de sincronização é importante para ambientes com um servidor e vários clientes. Sua principal finalidade é garantir que um cliente tenha sua requisição atendida, mesmo que vários outros clientes realizem requisições ao servidor, o que pode ser interpretado como evitar que atualizações concorrentes de um mesmo espaço de memória ocorram.

3.2.2.4 Contratos Quantitativos

Contratos quantitativos estão relacionado a qualidade do serviço oferecido por um componente. Alguns parâmetros podem ser levados em consideração para mensurar a qualidade de um serviço:

- Tempo máximo de resposta;
- Tempo médio de resposta;
- Qualidade do resultado, expresso com precisão;
- Número de requisições de clientes atendidas simultaneamente por um servidor.

3.2.3 Plataforma TAMAGO para Componentes

Uma outra técnica, definida por (Peschanski et al, 2006) adota o conceito de serviços para descrever contratos. A técnica descreve a plataforma Tamago, que assimila modelos de componentes de software, mais precisamente o modelo FRACTAL, proposto por (Collet et al, 2004a) e a Teoria de Design By Contract (Meyer, 1992). Fundamentado no conceito de interfaces de componentes, o modelo Fractal aborda contratos sobre a semântica de serviços requeridos e providos:

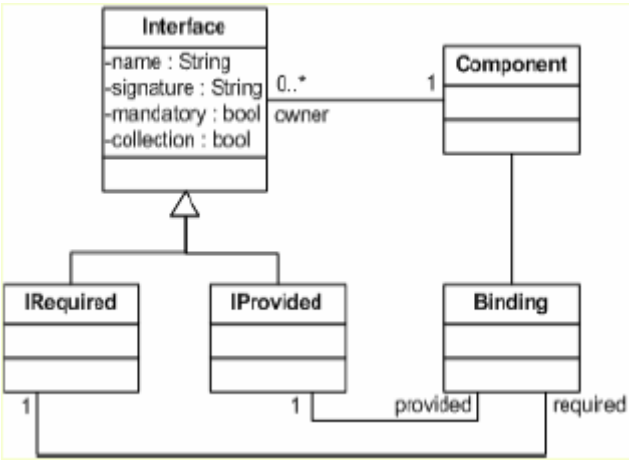


Figura 17: Modelo Fractal, extraído de (Collet et al, 2004a).

A plataforma Tamago é composta por uma API (Tamago-API) e módulos associados, conforme mostrado na figura 17. Tamago utiliza a tecnologia de *web services* no ambiente java, via plataforma Axis, com WSDL (Web Service Definition Language). TamagoCL é uma linguagem adaptada de *Object Constraint Logic* (OCL) (Warmer et al, 2003) que permite separar a especificação estática (via WSDL) e a especificação dinâmica (contratos OCL), enquanto que Tamago-CDL é a linguagem de formalização dos contratos e adota uma estrutura XML. O diferencial desta técnica está relacionado à possibilidade de especificar o comportamento de serviços utilizando transições de autômatos juntamente com condições lógicas.

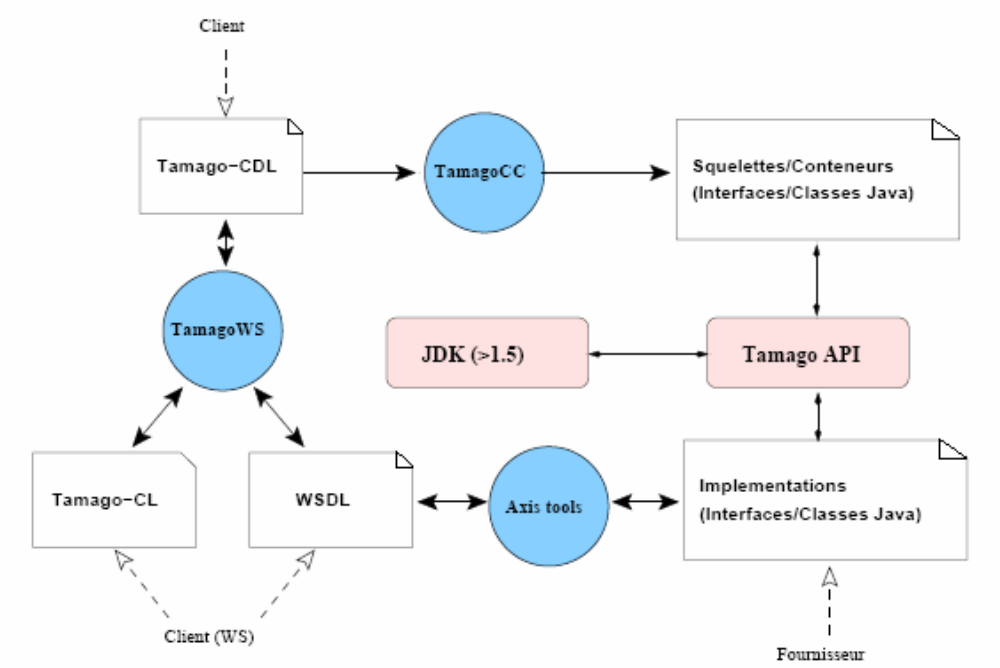


Figura 18: Plataforma Tamago, extraída de (Peschanski et al, 2006).

Tamago sugere a classificação de contratos para componentes em 4 categorias: (1) contratos de serviços, (2) contratos de componentes, (3) contratos de ligação e (4) contratos de composição:

- Contratos de serviços: correspondem a uma interface descrevendo as propriedades (estados observáveis) e as funcionalidades (assinatura dos métodos) dos serviços. Assertivas lógicas são utilizadas na definição de pré e pós-condições e invariantes, conforme a teoria Design By Contract. Exemplo extraído de (Peschanski et al, 2006):

```
service ByteInputStreamService {
   property read int size ;
   property read boolean isOpen ;
   property read boolean canRead ;
   invariant size>=0 ;
   method void open() {
      pre ¬isOpen fail "Stream already opened" ;
      post isOpen fail "Stream should be opened" ;
   }
   method void close() {
      pre isOpen fail "Stream not opened" ;
      post ¬isOpen fail "Stream should be closed" ;
   }
   method int read(byte[] buffer, int buflength, int offset) {
      pre isOpen fail "Stream not opened, cannot read" ;
      post size=size@pre-return fail "Wrong value" ;}}
```

- Contratos de componentes: em Tamago, cada componente é definido como uma implementação de um ou mais serviços. O contrato de componente descreve não somente os serviços oferecidos, como também os serviços requeridos pelo componente, e relaciona, através de inserções lógicas, a interdependência entre serviços:

```
component SoapBinaryDecoder {

   provide SoapDecoderService ;
   require ByteInputStreamService as stream;
   property read boolean completed ;
   property read boolean fetched ;
   invariant completed=¬fetched ;
   method void pump() {
      pre ¬completed^stream.isOpen^stream.canRead ;
      post completed ;
   }
   method SoapEnvelope fetch() {
      pre ¬fetched ;
      post fetched ;
   }
}
```

- Contratos de ligação: especifica uma arquitetura de disponibilização e componentes, conectando serviços que são requeridos e providos pelas instâncias de componentes:

```
assembly MailFrontend {
   use SoapBinaryDecoder as soapdec ;
   use MailBackend as backend ;
   bind service SoapDecodedService client backend.soapdecoder
                                          provider soapdec ;
}
```

- Contratos de composição: uma composição encapsula um grupo de sub-componentes e, similarmente a um componente, pode prover e requerer serviços. A composição define internamente um contrato de ligação entre seus sub-componentes:

```
composite MailBackend {

   use AuthComponent as auth ;
   property read boolean isActivate ;
   use MailParserComponent as parser ;
   use DatabaseComponent as db ;
```

```
    bind service MainParserService client db.parser
                                                provider parser ;
    bind service AuthService client parser.auth provider auth ;

    export DatabaseService by db ;

    require SoapDecoderService as soapdec ;

    invariant parser.sucessParse)(db.count@pre+1=db.count)

    method void online() {
       pre ¬isActivate ; post isActivate ;
    }

    method void offline() {
       pre isActivate ; post ¬isActivate ;
    }
}
```