3
Trabalhos Relacionados

Neste capítulo, são apresentados trabalhos relacionados à linha de pesquisa desta dissertação, que se resume em verificar se determinados comportamentos dos agentes de um sistema multi-agente estão realmente acontecendo. Estes trabalhos auxiliam a formar uma visão geral do estado da arte da linha de pesquisa, e servem como fonte de inspiração para trabalhos futuros.

3.1
Abordagem *Interagent* para o desenvolvimento de agentes

Neste trabalho [38], argumenta-se que os desenvolvedores de sistemas multi-agentes investem muito tempo no gerenciamento de questões referentes às interações entre os agentes e que muitas destas questões são independentes do domínio da aplicação. Desta forma, é proposta uma infra-estrutura que visa resolver as questões de interação comuns à qualquer sistema multi-agente, deixando o desenvolvedor focado nas funcionalidades internas do agente (aprendizado, representação do conhecimento, raciocínio . . .).

A interação é definida como sendo composta por vários níveis: nível de conteúdo, relacionado com o conteúdo das informações trocadas entre os agentes; nível intencional, relacionado em expressar as intenções dos agentes, geralmente através de uma ACL; nível de conversação, relacionado às convenções compartilhadas entre as trocas de expressões (*utterance*); nível de transporte, fornece mecanismos para auxiliar no transporte das expressões; e nível de conexão que contempla protocolos tais como TCP/IP, IIOP etc.

A proposta é fundamentada em dois elementos: *interagents* e protocolos de conversação. Em poucas palavras, os protocolos de conversação podem ser vistos como padrões de coordenação que restringem a seqüência de expressões (*utterances*) trocadas durante uma conversação (o que pode ser dito, por quem e quando). Os *interagents* são agentes de software que mediam as comunicações entre os agentes de uma organização e sua principal função é o gerenciamento

dos protocolos de conversação, similar ao agente mediador proposto nesta dissertação.

Os protocolos são especificados do ponto de vista de cada um dos participantes, em oposição à uma especificação global, como no caso desta dissertação. Desta forma, cada participante do protocolo possui uma visão parcial da execução global do protocolo. Para a especificação dos protocolos é utilizado um *Pushdown Transducer* [38]. Desta forma, o protocolo pode ser visto como tendo um conjunto finito de estados, arcos que representam transições entre estes estados, uma memória que armazena as mensagens recebidas pelos agentes e uma outra memória que é usada para armazenar as informações do contexto da execução do protocolo. O argumento utilizado para justificar a utilização deste modelo de especificação diante de outros já apresentados na literatura (FSMs, Dooley Graphs, Petri Nets...) é que este modelo provê um mapeamento direto da especificação para a implementação. A proposta de *interagents* foi estendida em [9] e é apresentada na Seção 3.2.

3.2 Instituições Eletrônicas

Neste trabalho [9], advoga-se que o objetivo das metodologias para sistemas abertos não deve ser obter um conjunto de agentes que interagem seguindo um protocolo pré-definido para alcançar um objetivo em comum. Ao invés disso, um ambiente normativo que irá determinar os possíveis comportamentos dos agentes. Além disso, como os sistemas abertos irão ser povoados por agentes que podem apresentar objetivos diferentes, o sistema deve ser composto de mecanismos computacionais que apliquem as regras da sociedade de agentes.

Para isso, é apresentado um modelo formal de instituições eletrônicas que define uma instituição como sendo composta dos seguintes componentes:

– *Dialogic Framework*: em linhas gerais, este componente define todas as mensagens que podem ser trocadas entre os agentes e quais são os papéis válidos. Mais especificamente, é definida uma ontologia que estabelece quais são os possíveis valores para os conceitos em um determinado domínio; uma linguagem de conteúdo que permite expressar as informações trocadas entre os agentes; um conjunto de *illocutionary particles* que expressam as intenções do agente locutor; um conjunto de papéis internos; um conjunto de papéis externos; e um conjunto de relacionamento entre os papéis.

Um papel interno é aquele que é desempenhado por um agente cuja função é manter a instituição funcionando. Fazendo uma analogia ao mundo real, eles são os empregados de uma instituição. Os papéis externos são aqueles que são desempenhados por aqueles que interagem com a instituição. Por exemplo, em um leilão existe o agente leiloeiro que é responsável por recolher os lances e anunciar o vendedor. Além dele, existem os agentes vendedores e os compradores. Neste caso, o leiloeiro é um papel interno e os vendedores e compradores são papéis externos.

– *Performative structure*: este elemento define quais são as cenas, e como os agentes dependendo de seus papéis e de suas ações passadas podem se mover entre elas. As cenas representam uma conversação entre agentes.

– *Norms*: as normas representam quais são as conseqüências das ações dos agentes. Ou seja, uma ação do agente pode acarretar na aquisição de uma obrigação. Por exemplo, um agente que venceu o leilão se torna obrigado a efetuar o pagamento em algum momento no futuro. Um agente externo pode não cumprir suas obrigações, pois eles são autônomos e podem ter sido desenvolvidos por outras pessoas. Entretanto, a instituição sabe as obrigações que cada agente adquiriu e portanto tem como detectar quando o agente não cumpre com as suas obrigações e com isso pode restringir as ações deste agente.

Limitações nas interações dos agentes se dão através de restrições (*constraints*), que limitam um conjunto de valores válidos para determinadas informações trocadas entre os agentes, e através das normas que restringem as ações dos agentes na instituição. As restrições são representadas em uma linguagem própria e, portanto, algumas conseqüências desta decisão são que a expressividade é limitada à linguagem proposta, e o desenvolvedor precisa aprender mais uma linguagem. Isto difere das restrições propostas mais adiante nesta dissertação, que utilizam componentes Java para implementá-las.

Para dar suporte à especificação de uma instituição seguindo este modelo, foi desenvolvida uma linguagem chamada ISLANDER e um editor gráfico que permite a especificação através da combinação de elementos gráficos e textuais. Além da especificação, este editor também permite a verificação de algumas propriedades, mais especificamente: integridade, que permite checar as auto-referências entre os elementos da linguagem estão corretas; *liveness*, que permite verificar se algum agente estará bloqueado na *performative structure* e que o estado final das cenas sempre é alcançado, corretude do protocolo; e finalmente a verificação se os agentes podem cumprir as normas especificadas.

Ao término da especificação da instituição, foi proposto um mecanismo de software que verifica se a especificação da instituição eletrônica está sendo cumprida durante a interação dos agentes. Este mecanismo utiliza o JADE [44] como camada de comunicação e é composto de 3 elementos principais:

– *Institution Manager* é o elemento responsável por autorizar os agentes a entrar na instituição, iniciar o sistema e controlar a participação dos agentes entre as cenas. Uma vez autorizados, é associado a cada agente um agente *Governor*.
– O *Scene Manager* é responsável por controlar a execução de uma cena, identificando quais os agentes podem participar da cena.
– Cada *Governor* é responsável por mediar a comunicação o agente e o resto da instituição. Eles dão aos agentes toda a informação de que eles precisam para participar de uma instituição.

Um dos pontos importantes que vale ser ressaltado nesta abordagem é que o protocolo é especificado sob um ponto de vista global. Desta forma, é possível saber qual o exato estado de execução deste protocolo. Na abordagem, os protocolos são especificados através de uma máquina de estados finita, onde não existe a idéia de concorrência. Desta forma, os *governors* (responsáveis por atualizar o estado da execução do protocolo ou cena) precisam se coordenar para que somente um deles atualize o estado do protocolo por vez. Além disso, uma vez que a atualização do estado de execução do protocolo é feita, todos os governors precisam atualizar a sua visão do estado do protocolo para poder executar alguma mudança de estado. Uma transição que era válida em um estado anterior pode não ser mais válida no novo estado.

Em uma possível solução para esta questão, poderia existir um elemento centralizador que mantém o estado atual da execução de uma cena e, portanto, quando cada *governor* precisar atualizar o estado, é preciso "travar" o estado atual, solicitar uma cópia deste estado, realizar a interpretação da transição de estados, possivelmente modificar o estado e enviar de volta o novo estado destravando-o. Desta forma, a cada mensagem que o agente envia, existem mais duas, uma para a recuperação e outra para a atualização do estado. Somente este fato não ocasiona muitos problemas em termos de ordem de complexidade do algoritmo. Entretanto, introduz-se um único ponto de falha no sistema. Ou seja, caso este elemento que detém as informações sobre o estado da cena se torne indisponível, todos os agentes que participam da cena passam a não poder se comunicar.

Na abordagem, existe um elemento centralizador que mantém o estado atual da execução de uma cena. Porém, várias cópias do estado atual estão

distribuídas entre os *governors*. Esta solução elimina o problema de se ter apenas um único ponto de falha, porém, introduz um custo de comunicação relacionado a manter os estados dos *governors* atualizados. Cada mensagem enviada pelos agentes que muda o estado da execução do protocolo tem como conseqüência a atualização do estado deste protocolo em todos os governors. Ou seja, a cada mensagem existe o custo de um *broadcast* para atualização dos estados. Logo, se x mensagens forem enviadas entre os agentes, o número total de mensagens do sistema é de aproximadamente: $x \times n$, onde n é o número total de agentes em uma cena. Este alto custo de coordenação torna esta solução imprópria para sistemas onde existam um grande número de agentes interagindo em uma mesma cena.

A principal diferença deste trabalho para o proposto nesta dissertação está na natureza dos elementos utilizados para especificar as leis, que se reflete no modelo conceitual. Ou seja, são utilizados diferentes conceitos, com diferentes comportamentos, para a especificação das interações. Além disso, a comunicação baseada em eventos entre os elementos do modelo conceitual proposta nesta dissertação, faz com que seja possível adicionar expressividade combinando elementos. Por exemplo, é possível conectar um relógio a uma norma, fazendo com que a norma se torne sensível ao tempo.

3.3
Normas abstratas, estatutos e instituições eletrônicas

Neste trabalho [12], argumenta-se que as normas são descritas em estatutos ou regulamentações de uma instituição. Porém, em geral estas descrições são muito vagas, o que torna muito difícil verificar se os protocolos que são seguidos durante a interação estão em conformidade com as regulamentações. O objetivo principal do trabalho é apresentar como as normas especificadas em regulamentos podem ser incorporadas em uma instituição de forma que os agentes que fazem parte desta instituição possam operar de acordo com estas normas e, caso não o façam, que sejam punidos.

Argumenta-se a necessidade de trazer as normas do alto nível de abstração inerentes de sua natureza, para um nível onde seus impactos na instituição possam ser descritos diretamente.

Também defende-se a idéia de que deve-se ter normas representadas explicitamente, possibilitando que agentes deliberativos usem esta especificação das normas para alcançar comportamentos inteligentes, decidindo quando vale a pena descumprir uma norma e arcar com as conseqüências.

Um estatuto de uma instituição descreve o objetivo da instituição, os valores que ela segue e as normas que devem ser seguidas. O objetivo é a razão da

existência de uma instituição. Por exemplo, o objetivo da *National Organization for Transplants (ONT)* é aumentar o número de doações de órgãos. Os valores são crenças sobre o que acredita-se ser importante tanto para a instituição quanto para a sociedade, por exemplo, não pode haver discriminação de cor na doação de órgãos. E as normas nos dizem como devemos nos comportar em uma determinada situação, por exemplo, não negando uma doação de um órgão por causa da cor de um doador. Note-se que os valores restringem as formas de alcançar o objetivo dizendo que os valores devem ser seguidos na sua busca. E cada valor tem associado a ele uma lista de normas que contribuem para ele.

Este trabalho defende que tentar controlar todos os possíveis comportamentos para garantir um comportamento normativo resulta em uma grande ineficiência da instituição. Então em alguns casos a instituição pode decidir, por questões de flexibilidade e eficiência, não fazer o "*enforcement*" de uma norma restringindo o comportamento, mas ao invés disso reagindo às violações.

O relacionamento deste trabalho com esta dissertação se dá no sentido que uma série de requisitos para o estabelecimento de leis devem ser levantados. Estes requisitos podem, inclusive servir como indícios de um método de engenharia de software onde a partir da descrição de um problema, guie os desenvolvedores para a especificação das leis.

3.4 Agentes deliberativos: princípios e arquitetura

Em [45], assume-se que normas são úteis em sociedades de agentes. Além disso, a violação inteligente destas normas também é útil. Com base nestas hipóteses, são identificados os princípios que agentes capazes de agir deliberativamente com base na descrição das normas precisam possuir. Finalmente, com base nestes princípios, é proposta uma arquitetura de um agente.

Além disso, um dos problemas de se ter protocolos fixos é que como uma característica importante dos agentes é reagir às mudanças em um ambiente dinâmico, quando os protocolos usados para reagir a estas mudanças são fixos, então eles não têm como reagir a mudanças não previstas. Por exemplo, se um agente nota que o outro agente está trapaceando, ele não pode mudar para outro protocolo que lhe protegeria. Agentes deliberativos não são abordados nesta dissertação e, portanto, podem ser pesquisados em trabalhos futuros.

Em [46], é proposta uma arquitetura de um agente que permite o desenvolvimento de agentes motivados por normas. Esta arquitetura é determinada por dois elementos: uma linguagem que permite a especificação de planos e normas e um interpretador capaz de executar as especificações desta linguagem. Nesta

linguagem são especificados três elementos principais, (a) um conjunto de crenças, (b) um conjunto de planos e (c) um conjunto de normas. O comportamento de um agente que segue esta arquitetura é influenciado por estes três elementos. Um plano possui a informação de quando ele deve ser selecionado para a execução, quais os efeitos da execução deste plano e, finalmente, qual é o comportamento do plano. Uma norma contém condições de ativação e de expiração que, obviamente, determinam quando a norma está ativa e quando ela expirou, respectivamente. As normas tornam-se ativas quando as crenças do agente refletem as condições de ativação da norma. Desta forma, o comportamento do agente é determinado pelas normas e pelos planos. Mais especificamente, as normas influenciam o comportamento do agente de duas maneiras, (a) motivando a geração de um objetivo que precisa ser alcançado ou uma ação que precisa ser realizada e (b) sendo usado como filtro restringindo as ações do agente.

3.5
Adoção de normas em um agente deliberativo

Em [47], discute-se o assunto da consistência de normas do ponto de vista de um agente normativo, mais especificamente, de um agente normativo que utiliza a arquitetura proposta em [46]. Ou seja, discute-se se um conjunto de normas declaradas em um contrato está consistente com o conjunto de normas que o agente atualmente segue.

Desta forma, para um agente adotar uma nova obrigação, proibição ou permissão, é preciso que elas estejam consistentes com as normas externas. Em outras palavras, a introdução de uma nova obrigação não deve se contrapor às obrigações existentes ou motivar a adoção de um estado que é permitido pela norma.

São identificados três níveis de consistência que ocorrem quando o agente adota uma nova norma.

- Forte consistência - uma norma é fortemente consistente quando não existe um conflito. Por conflito entende-se que a adoção de uma determinada norma motiva que um estado *S* seja alcançado e este estado é um estado proibido de ser alcançado (pela norma externa).
- Forte inconsistência - uma norma é fortemente inconsistente quando existe um conflito.
- Fraca consistência - uma situação onde a adição de uma obrigação possivelmente leva a uma inconsistência.

Através destas classificações, um agente pode decidir se vai adotar ou não uma determinada norma (interna).

3.6 A abordagem LGI - Law-Governed Interaction

Em [8], propõe-se um mecanismo de coordenação e controle[1] 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

[1] Coordenação e controle na verdade são termos com significados muito próximos, uma vez que pode-se definir coordenação como o controle das dependências entre as atividades [48]

ativadas por vários tipos de eventos, entre eles, mensagens enviadas e recebidas, obrigações etc.

Além disso, dois outros elementos importantes são os grupos e os *control-states*. Os grupos, por sua vez, são divididos em dois tipos: os implícitos e os explícitos. A diferença entre os grupos explícitos e implícitos é que o grupo explícito possui controle sobre quem faz parte dos grupos e este controle é realizado através de uma entidade chamada *secretary*. Os *control-states* funcionam como uma memória que somente pode ser acessada pelo mecanismo de LGI, mais especificamente, pelos controladores. Os controladores são agentes confiáveis que mediam a comunicação entre todos os agentes de uma sociedade. São eles que interpretam as leis e se comunicam com os controladores dos outros agentes.

Em contraste, esta dissertação fornece um modelo conceitual explícito e foca em conceitos diferentes, tais como, cenas, normas e relógios. Além disso, em relação ao suporte de software, a interação é especificada de forma centralizada.

3.7 Outros trabalhos

Nesta seção, apresentam-se trabalhos que não estão fortemente relacionados a esta dissertação mas que, no entanto, apresentam idéias que serviram de inspiração para este trabalho e, também, de inspiração para trabalhos futuros.

3.7.1 Perspectiva normativa para sistemas de software

O principal assunto tratado em [18] é a importância de se adotar uma perspectiva de sistemas normativos no desenvolvimento de software e ilustrar os papéis da lógica deôntica e da lógica de ação para a representação de leis, na formulação de modelos de sistemas de computação e na especificação de regulamentações que governam a interação humano-computador.

A lógica deôntica torna explícito o raciocínio sobre a distinção entre situação ideal da situação atual. Ou seja, o que deveria acontecer em um sistema e o que está acontecendo realmente.

É apresentado um exemplo de regulamentação no contexto de uma biblioteca de livros que ao mesmo tempo é simples e expressivo. Expressivo no sentido que ele possui muitas das características presentes em leis. Entre elas, (a) uma regulamentação faz parte de uma teia de regulamentações, ou seja, ela se relaciona com outras regulamentações, (b) respeitam valores (morais, corporativos

etc), (c) podem ocorrer mudanças na regulamentação, (d) podem existir casos que precisam ser tratados de forma especial etc.

Apresenta-se a idéia de que a regulamentação é uma especificação da maneira como o sistema deve operar. Então, a partir desta especificação, o sistema pode ser projetado de forma que a garantia da conformidade com a regulamentação esteja no código. Este processo é conhecido como regimentação, ou seja, a conformidade com a regulamentação é garantida pela implementação.

Entretanto, para garantir que a implementação corresponde realmente a especificação, precisa-se ter uma especificação dos requisitos do sistema e uma outra especificação para descrever como o sistema de computação deve funcionar. Nesta dissertação, apresenta-se uma contribuição neste sentindo, uma vez que as leis são uma especificação de como o sistema deve funcionar.

3.7.2
Relacionando agentes, mercados, instituições e protocolos de interação

Em [11], discute-se quais são os principais tipos de transações eletrônicas e quais os papéis dos agentes neste contexto. Mais especificamente, apresenta-se um pouco da teoria sobre mercados e instituições, além de relacionar agentes de software com ambos os assuntos. Finalmente, o trabalho esboça uma arquitetura de um agente comerciante (*trading agent*) contendo uma descrição dos principais módulos deste agente.

As transações econômicas podem ser organizadas através de dois mecanismos: hierárquico ou mercado (*market*). No hierárquico as partes que participam da transação possuem um relacionamento especial, em geral por fazer parte da mesma empresa ou porque possuem um contrato relacionado à transação. Por outro lado, a principal idéia no mecanismo de mercado é que existem muitos fornecedores de um produto e muitos clientes que desejam comprar o produto. Em geral, os custos de coordenação no mecanismo de mercado são maiores que os custos do mecanismo hierárquico.

Porém, existem vários produtos e situações onde o uso de um dos dois mecanismos não é adequado. Para estes casos, é preciso descobrir que tipo de suporte deve ser oferecido e qual o papel dos agentes se não for possível utilizar unicamente um mecanismo de mercado ou hierárquico.

Instituições que empregam protocolos rígidos para as interações possuem como principais vantagens (a) o aumento da confiança dos usuários nos agentes, uma vez que os procedimentos estão bem definidos e rígidos; (b) os agentes que participam da instituição podem ser mais simples em termos de comunicação; (c) o modelo do ambiente pode ser mais simples uma vez que não é preciso

se adaptar a eventos inesperados; (d) e, finalmente, toda a inteligência pode ser colocada em módulos que determinam a melhor estratégia sobre um protocolo fixo.

Porém, em alguns casos, é possível que as instituições não fixem todo os procedimentos da transação mas, ao invés disso, forneçam somente algumas fronteiras ou suporte. Por exemplo, a instituição pode armazenar todos os compromissos assumidos durante a fase de negociação para verificar se eles não estão sendo quebrados. Este tipo de fronteira pode ser alcançado através do uso de normas.

A situação ideal é aquela onde todos os tipos de agentes podem negociar seus produtos. Os agentes mais simples funcionam através de um protocolo simples e os mais complexos funcionam desde protocolos mais simples até protocolos mais complicados (não fixados *a priori*) para conduzir negociações com outros agentes complexos.

3.7.3
Interação em foco: abstraindo padrões de interação

Em [49], argumenta-se que é preciso uma especificação modular de padrões de interação para lidar com sistemas distribuídos. É apresentado o modelo de ator (Seção 3.7.4) e algumas abstrações em nível de programação, tais como, *activators*, *protocols*, *synchronizers* e *actorspaces*.

Foi apresentada uma linguagem de programação que permite que os protocolos sejam especificados e associados aos atores escritos em uma linguagem própria.

Os sincronizadores, apresentados na Seção 3.7.4, são responsáveis pelo que os autores chamam de restrição de coordenação (*coordination constraints*). Estas restrições determinam quando as ações acontecem. Tais restrições generalizam as sincronizações locais para sincronizações multi-atores.

O tratamento da coordenação apresentado neste trabalho assume que os atores estão todos escritos na mesma linguagem de atores. De certo modo, o mecanismo de coordenação é uma extensão da linguagem de atores.

3.7.4
Interação em foco: o modelo de atores

Neste trabalho [50], apresenta-se o modelo de *Actor* como uma entidade computacional que se comunica através de mensagens assíncronas. Cada ator estende o conceito de objeto no sentido que eles encapsulam um estado, um

conjunto de operações que manipulam este estado, uma *thread* de controle e, finalmente, uma localização conceitual - um único endereço de email associado ao ator. São propostas duas classes de abstrações que fazem parte do modelo de *actors* - comunicação e modularidade. O objetivo destas abstrações é, principalmente, facilitar a programação dos sistemas.

Três abstrações são propostas para a comunicação. A primeira delas é a *Call/Return*, onde o objetivo é permitir aos programadores expressar um padrão de comunicação onde o remetente invoca uma operação remota e usa o resultado para continuar a sua computação, em outras palavras, o programador tem a impressão que a chamada foi realizada de forma síncrona. A segunda abstração é a *Pattern-Directed*. Esta abstração permite a referência a grupos de agentes através de padrões dos atributos específicos dos atores. Desta forma, pode-se comunicar com um grupo específico ou com um membro em particular sem saber exatamente qual o seu endereço. Finalmente, a terceira abstração é um mecanismo interno para sincronização (*Local Synchronization*), que pode ser utilizado, por exemplo, para controlar o fluxo de produção do produtor em um esquema do tipo produtor-consumidor.

Outras três abstrações também são propostas para tratar modularidade. O *Synchronizer* é um tipo especial de ator que, através da mediação da troca de mensagens entre os atores, permite a manipulação das mensagens trocadas. Desta forma, pode-se ora implementar a sincronização de mensagens, ora funciona-se como um filtro, escolhendo quais as mensagens serão enviadas. A segunda abstração é de modularidade é uma extensão do *Synchronizer* para suportar restrições de real time, onde basicamente o *Synchronizer* passa a lidar com o atributo tempo. Finalmente, a última abstração é a de protocolo. Através dos protocolos a interação é separada do código do componente.

Os autores dizem também que os *actors*, são equivalentes a definição de agentes. Porém, não utiliza os argumentos necessários para sustentar esta hipótese.

3.7.5
Contratos entre objetos

Neste trabalho [51], apresenta-se um cenário de desenvolvimento de software onde com a evolução dos requerimentos do mercado, os sistemas vem se tornando cada vez maiores e mais complexos. Os sistemas de informação devem ser concebidos para lidar com esta evolução das regras de negócio. Regras de negócio determinam a maneira que o comportamento e interação de um objeto precisam ser coordenadas.

As regras de negócio devem ser descritas explicitamente, através de entidades de primeira ordem. Uma maneira de se fazer isso na orientação a objetos é através do uso de classes de associação para representar as interações. Desta forma, ela passa a ser tratada como entidade de primeira ordem (classe). O principal problema desta abordagem é que os mecanismos usualmente empregados para implementar as associações não fornecem o grau de flexibilidade necessário para criar, apagar ou modificar estas classes sem mudar a implementação dos papéis que participam da associação.

Diante deste problema, é proposta a idéia de contratos como uma extensão das classes de associação. O principal objetivo dos contratos é prover flexibilidade para a definição de novas interações ou modificação das existentes sem modificar os parceiros do contrato.

Em um contrato, os parceiros[2] não têm conhecimento de que estão sendo coordenados pelo contrato, ou seja, o contrato age interceptando as chamadas entre os parceiros. Os Contratos também podem ser superpostos cumulativamente, ou seja, um contrato pode agir sobre o resultado da atuação de outro contrato sobre um mesmo objeto.

Também é apresentada a notação, linguagem e semântica dos contratos, além de um padrão de projeto para implementar contratos em linguagem de programação orientadas a objetos.

A diferença entre a abordagem de contratos e a abordagem desta dissertação é que enquanto na dissertação preocupa-se com a representação e a implementação de leis que se aplicam ao sistema como um todo, a abordagem de contratos possui um contexto mais local, preocupando-se em coordenar as interações entre objetos. Porém, as abordagens são complementares e são necessários estudos para a construção de linguagens e métodos que tratem desta complementaridade.

[2]Parceiros são os objetos que interagem através de um contrato.