Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Programa de Pós-Graduação em Ciência da Computação

Thiago Locatelli da Silva

ESTENDENDO O MASUP PARA A ESPECIFICAÇÃO DE PAPÉIS DE AGENTES A PARTIR DA MODELAGEM DE NEGÓCIO

Porto Alegre, Março de 2009

Thiago Locatelli da Silva

ESTENDENDO O MASUP PARA A ESPECIFICAÇÃO DE PAPÉIS DE AGENTES A PARTIR DA MODELAGEM DE NEGÓCIO

Dissertação apresentada como requisito para obtenção do grau de Mestre, pelo Programa de Pós-Graduação da Faculdade de Informática da Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador:
Prof. Dr. Ricardo Melo Bastos

Porto Alegre, Março de 2009

Dados Internacionais de Catalogação na Publicação (CIP)

S586e Silva, Thiago Locatelli da
Estendendo o MASUP para a especificação de papéis de agentes a partir da modelagem de negócio / Thiago Locatelli da Silva. – Porto Alegre, 2009.
135 f.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientadora: Prof. Dr. Ricardo Melo Bastos.

1. Informática. 2. Sistemas Multiagentes. 3. Sistemas de Informação. 4. Processo Decisório. I. Bastos, Ricardo Melo. II. Título.

CDD 006.39

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada **"Estendendo o MASUP para a Especificação de Papéis de Agentes a Partir da Modelagem de Negócio"**, apresentada por Thiago Locatelli da Silva, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Sistemas de Informação, aprovada em 26/03/09 pela Comissão Examinadora:

Prof. Dr. Ricardo Melo Bastos – Orientador — PPGCC/PUCRS

Profa. Dra. Edimara Mezzomo Luciano - — PPGCC/PUCRS

Prof. Dr. Marcelo Blois Ribeiro – — PPGCC/PUCRS

Homologada em 01/08/09, conforme Ata No. 013/09. pela Comissão Coordenadora.

Prof. Dr. Fernando Gehm Moraes
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P32 – sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320–3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

Agradecimentos

Gostaria de agradecer a todos aqueles que fizeram parte deste trabalho: família, colegas, amigas, professores e financiadores.

Resumo

Sistemas Multiagentes ganharam muita atenção nos últimos anos através do surgimento de novas propostas para metodologias de desenvolvimento, linguagens de modelagem e plataformas para implementação e execução de tais sistemas, entre outras. Tais propostas visam o amadurecimento o processo de desenvolvimento de aplicações orientadas a agentes. Entretanto, observamos a necessidade de um método menos subjetivo para a identificação de papéis de agentes, que simultaneamente não dependesse do conhecimento técnico dos analistas de sistemas e que não envolvesse a modelagem de sistemas em si, uma vez que nesta etapa do processo de desenvolvimento, o sistema já está em construção. Devido as vantagens fornecidas pela Modelagem de Negócio, muitas empresas começaram a fazer o uso desta disciplina para reestruturação seus processos, buscando desta forma, identificar possíveis melhorias dentro da organização. Dentro desse contexto, neste trabalho está sendo proposta a introdução da Modelagem de Negócio à uma metodologia voltada a construção de sistemas multiagentes para a realização da identificação de papéis de agentes. Junto com a introdução dessa disciplina, utilizamos conceitos provenientes do Processo Decisório com o intuito de tornar mais objetivo a etapa de identificação de papéis de agentes.

Palavras-chave - modelagem de negócio, sistema multiagentes, metodologia de desenvolvimento, sistema de informação, papel de agente, processo decisório.

Abstract

Multiagent Systems gained a lot of attention in the last years through the appearance of new proposals for development methodologies, modelling languages, implementation and execution environments of these systems, among others. Such proposals aim the maturing of the agent oriented development process. However, we have noticed the need for a less subjective method for agent role identification that does not depend simultaneously on the technical knowledge of systems analysts and that does not involve the system modelling, once in this phase the system is already in construction. Due to the advantages brought by the Business Modelling, a great number of companies have started using this discipline in order to redefine their processes, aiming to find possible improvements on such process and in the organization by itself. In this context, it is being proposed in this dissertation a extension in an existing agent oriented methodology by adding the Business Modelling for agent role identification. We also bring to bear the concepts from the Decision Process to make the role identification a process more objetive.

Palavras-chave - business modelling, multiagent systems, development methodology, information systems, agent role and decision process.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

SMA	Sistemas Multiagentes	19
MaSE	Multiagent Systems Engineering	45
RUP	Rational Unified Process	55
MASUP	Multi-agent Systems Unified Process	55
ACL	Agent Communication Language	59
SI	Sistemas de Informação	61
SADE	Sistema de Apoio à Decisão	72
SSD	Sistemas de Suporte a Decisão	72
GPS	Global Positioning System	97
Case	Computer Aided Software Engineering	113

Sumário

Capítulo 1

Introdução

"A imaginação é mais importante que o conhecimento"
— Albert Einstein

O crescimento das tecnologias de informação nas organizações leva ao aparecimento de novas siglas, novos fabricantes e produtos que vêm a substituir outros. Dentro destas novas mudanças podemos destacar o surgimento de novos procedimentos na área de desenvolvimento de software. Ao longo dos últimos 20 anos, o software tem conquistado um papel essencial e crítico em nossa sociedade. Neste sentido, a indústria do software vem experimentando um grande crescimento, tendo como conseqüências o aumento da complexidade do software e as exigências cada vez maiores do mercado.

Na busca por soluções mais complexas e confiáveis, nos últimos anos as pesquisas relacionadas com agentes inteligentes têm proposto alternativas para o desenvolvimento de sistemas introduzindo a idéia de agentes inteligentes dentro dos sistemas de informação [14]. Desses estudos, muitas propostas emergem com o objetivo de facilitar e aprimorar o desenvolvimento de sistemas utilizando o conceito de agente. Podemos entender como propostas as metodologias voltadas ao desenvolvimento de sistemas cujo paradigma é orientado a agentes, as linguagens de modelagem, as plataformas para simulação, execução e testes e os *frameworks* para construção de sistemas multiagentes. Por definição, uma metodologia de desenvolvimento de software é um conjunto de procedimentos sistemáticos para a geração de modelos, que descrevem aspectos de um sistema sob desenvolvimento [21]. Ainda, pode ser definida como um conjunto de processos e técnicas de modelagem bem como uma seqüência de passos para gerar uma descrição (formal ou técnica) de um sistema.

Sistemas Multiagentes é o nome dado à subárea da Inteligência Artificial Distribuída que estuda o comportamento de um conjunto de agentes autônomos objetivando a solução de um problema que está além das capacidades individuais

[32]. Segundo Wooldridge [73], um agente é um sistema computacional que está situado em algum ambiente, e que é capaz de ações autônomas neste ambiente objetivando alcançar seus objetivos.

O advento dos sistemas multiagentes trouxe consigo muitas disciplinas num esforço conjunto para construir aplicações distribuídas, inteligentes e robustas [16]. Estas disciplinas têm nos dado uma nova maneira de olhar para os sistemas distribuídos e nos deram um caminho para construir sistemas mais robustos. Entretanto, muitos dos atuais métodos existentes de análise e projeto de aplicações não se aplicam ao paradigma multiagentes. Nesse contexto, muitas metodologias, linguagens e ferramentas de suporte para o desenvolvimento de sistemas multiagentes têm sido propostas nos últimos anos com o objetivo de suprir a necessidade de um método eficaz para modelagem de um sistema multiagente. Dentre estas metodologias podemos destacar MASUP [6] e MaSE [69, 16, 15], Troppos [22, 51], Gaia [77, 72].

Por se tratar de uma área de pesquisa relativamente nova quando comparada com a história da computação, as metodologias voltadas para o desenvolvimento de SMAs ainda possuem algumas lacunas, fazendo com que estas se tornem incompletas ou possuam procedimentos subjetivos durante a modelagem e construção dos sistemas. Uma destas lacunas refere-se ao processo de identificação de papéis de agentes durante a etapa de modelagem. Tal processo, muitas vezes, tem como base o conhecimento do analista que usa sua experiência ou a conveniência para a identificação de tais artefatos. Dentre as metodologias estudadas, não foi identificado nenhum método detalhado e objetivo para a identificação de papéis de agentes, direcionando assim, o estudo desta pesquisa, cujo intuito é preencher tal lacuna.

Neste contexto de sistemas multiagentes e de metodologias de desenvolvimento, este trabalho tem como objetivo fazer um estudo das principais metodologias de desenvolvimento de sistemas multiagentes, focando-se nas etapas de identificação e na modelagem inter e intra-agentes. Após o estudo das metodologias, será proposto um método alternativo para identificação e modelagem de agentes dentro de um sistema, procurando suprir as carências dos principais métodos já propostos.

1.1 Questão de Pesquisa

Existem inúmeras metodologias para modelagem e construção de sistemas orientados a agentes, porém estas metodologias apresentam uma lacuna em comum, que diz respeito à identificação e especificação de papéis de agentes. Neste sentido, emerge a questão de pesquisa deste estudo: "*Como realizar a identificação e a especificação de papéis de agente durante o desenvolvimento de um sistema multiagentes?*".

1.2 Objetivos

Uma vez definida a questão de pesquisa, definiu-se o objetivo geral e os objetivos específicos deste trabalho, os quais são apresentados a seguir.

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é realizar um estudo sobre as metodologias para sistemas multiagentes e propor um método para identificação e especificação de papéis de agentes e integrá-lo ao já existente processo de desenvolvimento de sistemas multiagentes MASUP.

1.2.2 Objetivos Específicos

- Estudar e identificar os métodos para identificação de papéis de agentes utilizados pelas metodologias de desenvolvimento de sistemas multiagentes.
- Propor um método para a identificação de papéis de agentes tendo por base a Modelagem de Negócios.
- Especificar uma ferramenta de software e desenvolver um protótipo que possibilite o uso do método proposto.
- Aplicar o método proposto utilizando o protótipo da ferramenta desenvolvida e analisar os resultados.

1.3 Organização da Dissertação

A dissertação está dividida em seis capítulos. No capítulo 1 encontram-se a introdução ao tema, os objetivos propostos, as principais contribuições, o escopo do trabalho e por fim, a organização deste volume.

No Capítulo 2 é apresentado ao leitor os conceitos básicos acerca de sistemas multiagentes. Neste capítulo são abordados os conceitos mais importantes sobre agentes de *software* bem como as suas características. Também são introduzidas as características dos sistemas multiagentes, que tratam desde a organização interna, comunicação, colaboração e outras formas de interação entre os agentes que constituem um sistema de software agentificado.

O Capítulo 3 segue a linha de sistemas multiagentes, porém, este trata de metodologias voltadas para o paradigmas orientado a agentes. Neste capítulo, um breve histórico sobre metodologias para sistemas multiagentes é apresentado e logo em seguida, a metodologia MASUP é apresentada de forma mais detalhada.

No Capítulo 4 discorre sobre modelagem de negócio, disciplina ausente no MASUP e utilizada na proposta deste trabalho para a extensão dessa metodologia com um método para identificação de papéis de agentes tendo como base elementos do negócio, como por exemplo, o trabalhador do negócio. No Capítulo 5 são apresentados ao leitor, os conceitos provenientes dos estudos relacionados a processo decisórios são apresentados bem como um *framework* que surgiu da junção destes conceitos e que representa o processo decisório como um todo.

O Capítulo 6 referencia a proposta deste trabalho, cuja apresentação decorre em uma introdução descritiva (fazendo-se uso de um exemplo real) do método que está sendo proposto. Por fim, no Capítulo 7 encontram-se as conclusão deste trabalho bem como sugestões de possíveis trabalhos futuros relacionados a proposta desta dissertação.

Capítulo 2

Sistemas Multiagentes

"O fracasso é a oportunidade de se começar de novo inteligentemente"
— Henry Ford

Comparados com a história da Ciência da Computação, os Sistemas Multiagentes (SMA) são relativamente novos, tendo ganho grande atenção de pesquisadores nos últimos anos. Com o crescimento da complexidade e do tamanho das aplicações, fez-se necessário um ambiente que suportasse a heterogeneidade e a distributividade das aplicações em tempo real [37]. Assim surgiu a proposta dos SMAs, que representam uma solução a esse tipo de aplicação. Bastos em [5], também afirma que os SMAs representam uma alternativa para resolver problemas relacionados a processos de negócio, em que se fazem necessárias a distribuição e a descentralização da tomada de decisão e execução de processos.

Nesse contexto, o principal objetivo do presente capítulo é apresentar um embasamento teórico acerca do que são os SMAs e quais as suas características. Motivações, áreas de aplicação e conceitos serão apresentados nos tópicos a seguir. Este capítulo serve como base para o próximo capítulo, onde serão estudadas algumas metodologias voltadas ao paradigma orientado à agente.

2.1 Agentes Inteligentes

Para Oliveira [46], o conceito de agência deixou de ser utilizado apenas para especificar unidades computadorizadas com certos tipos de características (agentes), e passou a ser empregado também de uma maneira mais geral e abstrata, como uma nova metáfora para análise, especificação e construção de aplicações complexas. Wooldridge [73] afirma que um agente é um sistema computacional inserido em um ambiente, e que é capaz de, através de ações autônomas nesse ambiente, atingir os seus objetivos propostos. Assim, de acordo com muitos autores, um agente deve ter as seguintes características:

- Observar e entender o ambiente em que está inserido;
- Ter a capacidade de interagir com outros agentes;
- Ser proativo, ou seja, ter iniciativa para tomada de ações a fim de atingir os seus objetivos.

Hoje, pode-se encontrar na literatura diversas definições acerca do que vem a ser um agente e suas características. Corrêa [12], por exemplo, define um agente como uma entidade que funciona contínua e autonomamente em um ambiente onde existem outros processos e agentes.

Para Amandí [2], um agente é uma entidade computacional que possui um comportamento autônomo, o qual lhe permite tomar decisões para agir, levando em consideração as mudanças ocorridas no ambiente em que está inserido e o desejo de alcançar seus objetivos.

Shoham [63] afirma que um agente é uma entidade à qual se atribuem estados, denominados *estados mentais*. Os estados mentais são: crenças, decisões, capacidades, objetivos, intenções, compromissos e expectativas; conceitos análogos ou similares aos humanos. Sob este ponto de vista explicitado, o que faz qualquer componente de *hardware* ou *software* ser considerado um agente é precisamente o fato de que este pode ser analisado e controlado nos termos dos estados mentais.

D'Amico [13] considera que, do ponto de vista prático, são agentes: (a) robôs que atuam em um ambiente interagindo com outros robôs ou com humanos em língua natural e utilizando sensores para captar informações do ambiente, tais como controle de motor e restrições de tempo; (b) sistemas que interagem com outros sistemas ou com o ser humano.

Segundo Wooldridge [73], não existe uma definição universalmente aceita do termo agente, mas há um consenso geral de que autonomia é a idéia central da noção de agência. Para o autor, a dificuldade em se definir o termo agente surge do fato de que para diferentes domínios de aplicação os atributos associados ao conceito de agência assumem diferentes níveis de importância. Assim, da mesma forma que para alguns domínios de aplicação a capacidade de aprendizado a partir das experiências é de grande importância; para outros, a capacidade de aprender pode constituir um fato não só pouco importante, mas também indesejável. A Figura 2.1 ilustra o relacionamento do agente com o ambiente em que está inserido. Percebe-se a ação de saída gerada pelo agente, visando à interação com o ambiente. Normalmente, o agente não possui o controle total do ambiente em que participa, mas sim, uma influência sobre este. Sendo assim, ações aparentemente idênticas podem apresentar efeitos completamente diferentes. Isto confirma a importância da preparação do agente para possíveis falhas.

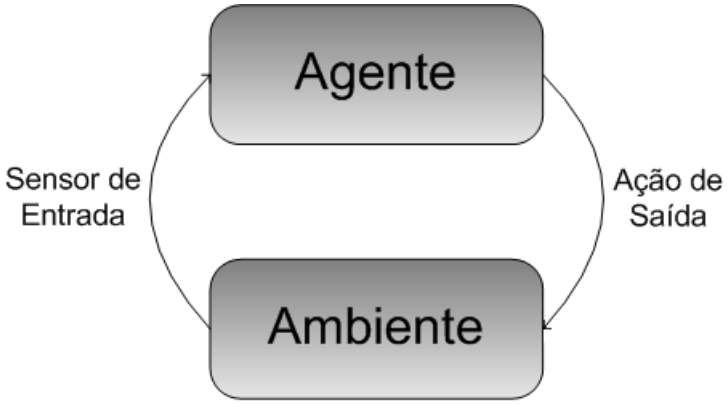


Figura 2.1: Agente e seu Ambiente - extraído de [73]

O agente tem, geralmente, um repertório de ações disponíveis capazes de modificar o seu ambiente. Essas ações não são executadas em todas as situações. Além disso, por ter em si pré-condições associadas, apenas as situações possíveis ocorrem. O problema surge da decisão sobre ações precisam atuar para satisfazer, da melhor forma, os objetivos buscados pelo agente. Disso, são introduzidas as arquiteturas de agentes, confirmando o seu uso como sistemas de tomada de decisão embutidas em um ambiente.

O agente é considerado como uma entidade encapsulada com capacidade de resolução de problemas e, desta forma, tem as seguintes características [71]:

- **Autonomia -** capacidade de executar a maior parte de suas ações sem interferência direta de agentes humanos ou de outros agentes computacionais, possuindo controle total sobre suas ações e estado interno;
- **Habilidade social** - capacidade que possibilita a interação com outros agentes (computacionais ou humanos) para solucionarem problemas que de certa maneira não podem ser resolvidos por apenas um agente;
- **Reatividade -** capacidade de perceber seu ambiente e reagir em um tempo satisfatório às mudanças, satisfazendo seus objetivos;
- **Proatividade -** capacidade de decisões ou principiar ações dentro do ambiente por iniciativa própria.

2.1.1 Tipos de Agentes

A estrutura básica de um agente tem uma forma bem simples: ela possui uma estrutura de dados interna que irá se atualizar com a chegada de novas percepções. Essa estrutura é utilizada nos procedimentos de tomada de decisão, os quais irão gerar ações para serem executadas (Figura 2.2).

```
function SKELETON-AGENT(percept) returns action
  static: memory, the agent's memory of the world
  memory  ← UPDATE-MEMORY(memory, percept)
  action  ← CHOOSE-BEST_ACTION(memory)
  memory  ← UPDATE-MEMORY(memory, action)
return action
```

Figura 2.2: Estrutura genérica de um agente - adaptado de [58]

Na construção de programas que utilizam agentes, deve-se decidir como construir o mapeamento de percepções a ações. Segundo Russel em [58], existem quatro tipos diferente de agentes, cada um possuindo aspectos diferenciados na busca da solução de problemas. São eles:

1. Agentes reflexivos;
2. Agentes reflexivos que mantêm registro do mundo;
3. Agentes baseados em metas ("*Goal-based*");
4. Agentes baseados na utilidade ("*Utility-based*").

Os agentes reflexivos têm como característica básica que cada condição ou percepção dispara alguma ação pré-estabelecida no programa, sendo essa relação conhecida como regra de condição-ação, por exemplo, "se o carro da frente freou então começar a frear" (Figura 2.3). Este agente faz uma simulação dos reflexos natos e das respostas aprendidas dos humanos.

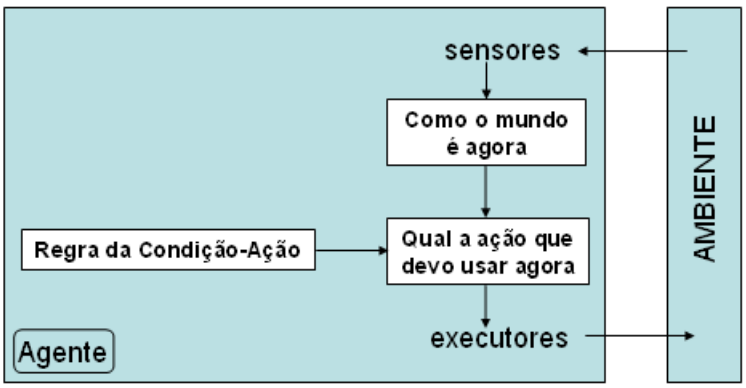


Figura 2.3: O agente reflexivo

A seguir (Figura 2.4) tem-me a estrutura de um agente reflexivo, onde se percebe o relacionamento da percepção com a ação.

O segundo modelo de agentes é uma versão melhorada do anterior, onde o estado interno do agente (por exemplo, o conhecimento interno do agente e relações entre percepção e ação) é atualizado, ou seja, ocorre um registro do estado do mundo (Figura 2.5).

```
function SIMPLE-REFLEX-AGENT(percept) returns action
  static: rules, a set of condition-action rules
  state  ← INTERPRET-INPUT(percept)
  rule   ← RULE-MATCH(state, rules)
  action ← RULE-ACTION(rule)
return action
```

Figura 2.4: Estrutura de um agente reflexivo - adaptado de [58]

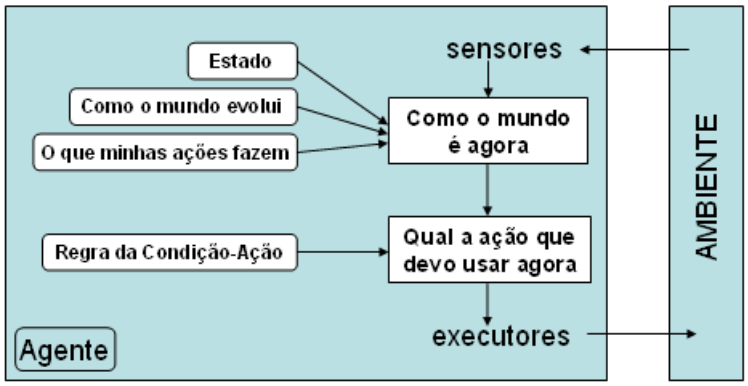


Figura 2.5: O agente reflexivo com estado

A estrutura deste tipo de agente mostra como a percepção corrente combinada com o estado interno antigo gera a atualização do estado corrente. A parte mais interessante é a função *update-state* (Figura 2.6), que é responsável pela criação do novo estado interno.

```
function REFLEX-AGENT-WITH-STATE(percept) returns action
  static: state, a description of the current world state
          rules, a set of condition-action rules
  state  ← UPDATE-STATE(state, percept)
  rule   ← RULE-MATCH(state, rules)
  action ← RULE-ACTION[rule]
  state  ← UPDATE-STATE(state, action)
return action
```

Figura 2.6: Estrutura de um agente reflexivo com estado - adaptado de [58]

O terceiro modelo é o agente baseado em metas (Figura 2.7). Internamente, os agentes possuem uma idéia do estado atual do ambiente, sendo isso importante, porém, o agente também precisa de uma meta, na qual terá a descrição de um estado desejável a ser atingido. Assim, o agente pode fazer uma combinação do desejável com os resultados de possíveis ações para tomar decisões na busca da meta. Em algumas das vezes isso pode ser feito de forma simples, nas quais a meta é alcançada com apenas uma ação; em outras vezes, pode ser mais complicado, pois serão necessárias longas seqüências de ações para alcançá-la.

Note que esse tipo de agente não apenas cumpre uma ação pré-determinada: há uma preocupação com o futuro, "o que acontecerá se eu fizer isso e aquilo?" e

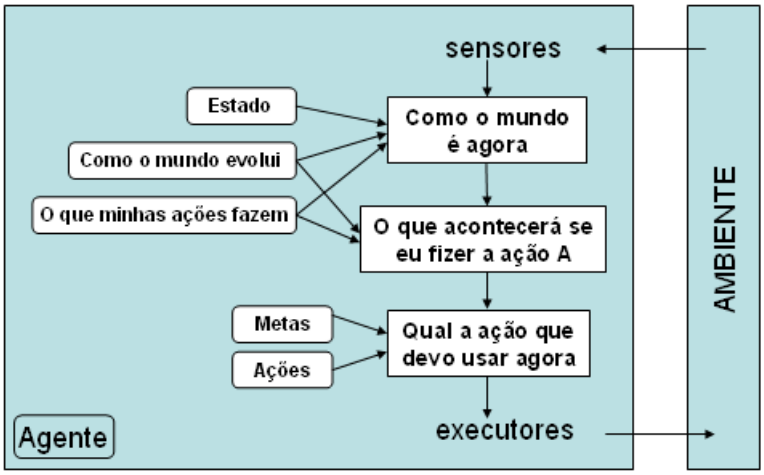


Figura 2.7: O agente baseado em metas

"isto irá me satisfazer?". Na Figura 2.8 te-se a estrutura deste tipo de agente.

```
function GOAL-BASED-AGENT (percept, goal) returns action
  static: state, a description of the current world state
          rules, a set of condition-action rules
  state  ← UPDATE-STATE(state, percept)
  rule   ← RULE-MATCH(state, rules)
  action ← RULE-ACTION[rule]
  state  ← UPDATE-STATE(state, action)
  if (state in goal) then
      return action
  else
      percept ← OBTAIN_PERCEPT(state, goal)
      return GOAL-BASED-AGENT(percept, goal)
```

Figura 2.8: Estrutura de um agente baseado em metas - extraído de [58]

O último tipo de agente é baseado na utilidade (Figura 2.9). Apesar da importância das metas, elas sozinhas não representam o bastante para se ter um comportamento de qualidade [58]. No agente baseado em metas só há preocupação entre os estados "satisfeito" e "insatisfeito"; enquanto que uma medida de performance mais geral permite uma comparação entre vários diferentes estados, de acordo com o *quão satisfeito se busca estar*. Devido ao termino "satisfeito"não soar de forma científica, utiliza-se o termo "utilidade". Isso quer dizer que, se há preferência por um estado, então este é mais útil para o agente.

No agente baseado na utilidade, existe uma preocupação com o grau de satisfação alcançado. Isso facilita a tomada de decisões em duas situações: primeira, quando há metas conflitantes e somente uma delas pode ser alcançada; segunda, quando há várias metas, porém, nenhuma pode ser alcançada de forma correta. Neste caso, o fator utilidade auxiliará na busca do estado que consegue um melhor grau de satisfação para o agente. A estrutura global do agente baseado na utilidade é ilustrada na Figura 2.10.

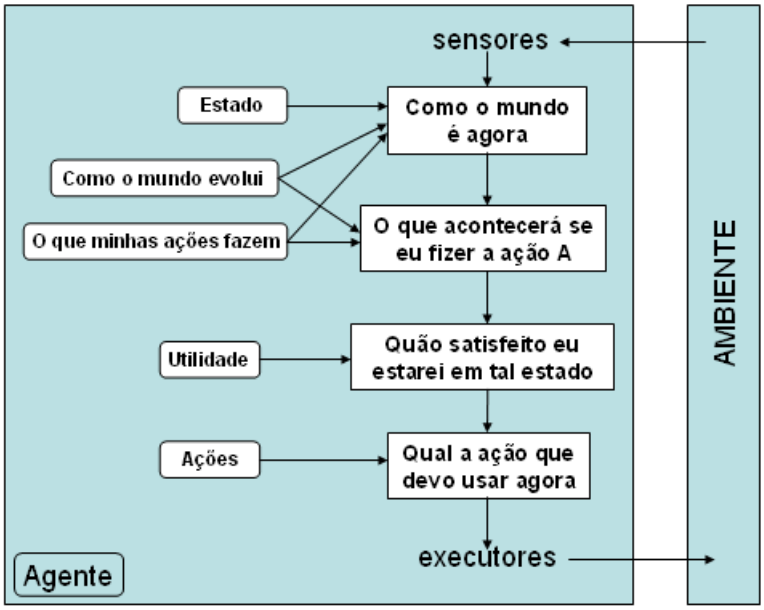


Figura 2.9: O agente baseado na utilidade

```
function UTILITY-BASED-AGENT (percept, goal) returns action
  static: state, a description of the current world state
          rules, a set of condition-action rules
  state   ← UPDATE-STATE(state, percept)
  rule    ← RULE-MATCH(state, rules)
  action  ← RULE-ACTION[rule]
  state   ← UPDATE-STATE(state, action)
  score   ← OBTAIN-SCORE(state)
  if (state in goal) and BEST_SCORE(score) then
      return action
  else
      percept ← OBTAIN_PERCEPT(state, goal)
      return UTILITY-BASED-AGENT(percept, goal)
```

Figura 2.10: Estrutura de um agente baseado na utilidade - extraído de [58]

2.1.2 Categorias de Agentes

Segundo Jennings [32], para que possam agir de maneira autônoma, os agentes podem ter várias habilidades: percepção e interpretação de mensagens, raciocínio baseado em crenças, tomada de decisão, planejamento e habilidade para executar planos incluindo passagem de mensagens. Jennings categoriza os agentes quanto ao nível de capacidade de resolução de problemas:

- **Reativos -** reagem a alterações no ambiente ou a mensagens de outros agentes. Não têm capacidade de raciocínio sobre suas intenções, reagindo tão somente sobre regras e planos estereotipados. Suas ações podem ser: atualizar a base de fatos e enviar mensagens para outros agentes ou para o ambiente.

- **Intencionais -** têm a habilidade de raciocínio sobre suas intenções e crenças, e sobre a criação e execução de planos de ações. São considerados como sistemas

de planejamento que selecionam objetivos - de acordo com suas motivações, e raciocinam sobre eles - detecção e resolução de conflitos e coincidências de objetivos, seleção e criação de planos (agendamento de ações), detecção de conflitos entre planos (alocação de recursos) e, se necessário, execução e revisão planos.

- **Sociais -** possuem modelos de outros agentes, sobre os quais raciocinam para tomar decisões e criar planos.

2.1.3 Classificação das arquiteturas de agentes

Em computação, o termo arquitetura pode compreender uma faixa razoável de possibilidades, principalmente no aspecto complexidade. As arquiteturas de agentes não são uma exceção e podem ser classificadas de acordo com as necessidades da aplicação, dos usuários e do grau de sofisticação ou nível de inteligência dos agentes.

Os autores Wooldridge e Jennings em [71], se baseiam na forma de construção dos agentes envolvidos para dividir as arquiteturas em três áreas:

- **Arquiteturas deliberativas -** seque a abordagem clássica da Inteligência Artificial, onde os agentes contêm um modelo simbólico do mundo, explicitamente representado, e cujas decisões são tomadas via raciocínio lógico, baseado em casamento de padrões e manipulações simbólicas. Esse tipo de arquitetura é utilizado nos agentes baseados em metas e agentes baseados na utilidade segundo a classificação de Russel, em [58].

- **Arquiteturas reativas -** as arquiteturas reativas são aquelas que não incluem nenhum tipo de modelo central e simbólico do mundo e não utilizam raciocínio complexo e simbólico. Baseiam-se na proposta de que um agente pode desenvolver inteligência a partir de interações com seu ambiente, não necessitando de um modelo pré-estabelecido. Esse tipo de arquitetura é utilizado nos agentes reflexivos e nos agentes reflexivos com estado na classificação de Russel, [58].

- **Arquiteturas híbridas -** misturam componentes das arquiteturas deliberativas e reativas, o que as tornam mais adequadas e funcionais para a construção de agentes. Elas propõem um subsistema deliberativo, que planeja e toma decisões da maneira proposta pela Inteligência Artificial Simbólica; e um reativo, capaz de reagir a eventos que ocorrem no ambiente sem ocupar-se de raciocínios complexos [58].

Weiss [Weiss,1999], apresenta quatro tipos de arquiteturas para agentes, que se aproximam muito da classificação proposta por Wooldridge e Jennings em [71].

Tabela 2.1: Classificação das arquiteturas de agentes

Arquiteturas de Agentes	Variações	Descrição
Deliberativa	Baseada em Lógica Arquitetura BDI Intencional Baseada em Metas	Possui um modelo simbólico do mundo. As decisões são tomadas via raciocínio lógico. Possui um conjunto de metas e intenções, onde é elaborado um plano baseando-se nessas metas. Possui certas restrições sobre a construção do modelo simbólico. Agentes complexos.
Reativa	Reflexiva	Sem modelo simbólico interno. As decisões tomadas são implementadas em alguma forma de mapeamento direto da situação para a ação, usando regras de condição/ação.
Híbrida		Mistura componentes das arquiteturas Deliberativa e Reativa.
Camadas		Apresentam a tomada de decisão por meio de camadas de software.

- **Agentes baseados em lógica -** nesses agentes a decisão é tomada através de dedução lógica (agentes baseados em metas e agentes baseados na utilidade segundo a classificação de Russel [58]).

- **Agentes reativos** - nesses agentes a decisão a tomar é implementada em alguma forma de mapeamento direto da situação para a ação (agentes reflexivos e agentes reflexivos com estado, de acordo com a classificação de Russel [58]).

- **Agentes híbridos -** nesses agentes a decisão é tomada via várias camadas de software, onde cada uma está mais ou menos explicitamente raciocinando sobre o ambiente em diferentes níveis de abstração.

- **Agentes de crença-desejo-intenção -** nesses agentes a decisão é tomada dependendo da manipulação de estruturas de dados representando as crenças, desejos e intenções do agente.

2.2 Agentes *versus* Objetos

Embora a construção de agentes nos leva a pensar e desenvolver usando o paradigma orientado a objetos, e de que existem algumas similaridades entre agentes e objetos, estas duas entidades não são semelhantes, tanto do ponto de vista de projeto, quanto de implementação [70].

Em [34], Kendall apresenta uma definição de agentes em comparação com objetos: "Agentes são extensões de objetos, eles apresentam todas as características

que os objetos possuem, porém, incorporam novas características como comportamento autônomo, pró-atividade, habilidade social, reatividade e comportamento inteligente".

Objetos são definidos como entidades computacionais que encapsulam algum estado, são capazes de executar ações, ou métodos sobre o seu estado, e de se comunicarem entre si através de troca de mensagens, tradicionalmente implementadas como chamada de métodos [73]. Esta é considerada a primeira diferença entre objetos e agentes. Uma maneira de entender como os objetos têm controle sobre o seu estado interno é pensar nas técnicas de programação orientadas a objetos, onde variáveis e métodos podem ser declarados como privados, ou seja, são acessados internamente apenas pelo próprio objeto. Desta forma, o objeto tem controle sobre o seu estado interno, porém, não tem controle sobre o seu comportamento; pois, uma vez que um método é declarado como público e invocado por outro objeto, esse deve ser executado sem que haja interferências. Não existe nenhum controle em relação à decisão de executar ou não o método, já que a ação é invocada por outro objeto através da troca de mensagens. Por isso, objetos são considerados passivos porque seus métodos são executados apenas quando alguma entidade externa faz uma requisição através de uma mensagem. Por outro lado, agentes são considerados ativos, já que podem iniciar uma ação sem a intervenção humana ou de outro agente.

Agentes são considerados entidades autônomas, uma vez que observam o ambiente em busca de informações necessárias para executar suas ações e atingir seus objetivos [45]. Além disso, os agentes podem ser interativos, ou seja, eles são capazes de usar diversas formas de mensagens, as quais podem suportar chamadas de métodos e pedidos de informações, entre outras propriedades. Devido ao fato de agentes serem autônomos, eles podem iniciar interações e responder a pedidos da maneira mais conveniente, já que, conforme James Odell em [45], agentes são objetos que podem dizer "Sim" ou "Não". Esta é outra diferença entre objetos e agentes: a não execução de um método em um objeto pode ser considerada um erro; já a não execução de uma ação em um agente pode ser considerada como uma decisão interna do mesmo.

Agentes não apenas respondem à chamadas de métodos, mas também a eventos que ocorrem dentro do ambiente em que estão inseridos. Agentes proativos irão procurar no ambiente alguns eventos ou mensagens que possam determinar o tipo de ação que deverá ser executada. Objetos são passivos, respondendo apenas a chamadas de métodos. Entretanto, algumas técnicas foram introduzidas na programação orientada a objetos como, por exemplo, o *event listener*, que permite objetos a executar métodos quando algum evento ocorre.

Outra característica importante quando se discute a diferença entre objetos e agentes é que os agentes possuem a sua própria *thread* de controle - no padrão

de objetos existe apenas uma *thread* em todo o sistema. Muitas linguagens de programação adicionaram propriedades que permitem o desenvolvimento de objetos concorrentes utilizando *threads*, porém isso é o mais próximo que um objeto pode chegar ao conceito de autonomia.

Quanto à comunicação, esta é considerada assíncrona entre agentes, uma vez que não existe um fluxo de controle pré-definido. Um agente pode realizar uma solicitação para outro e continuar com a execução do seu fluxo de ações sem que haja uma resposta. Já no modelo orientado a objetos, quando um objeto invoca um método de outro, ele interrompe o seu fluxo de execução até que receba uma resposta (ou que o método invocado termine sua execução).

Para Wooldridge [73], existem três distinções básicas entre a visão de objetos e a visão de agentes, que são resumidas da seguinte maneira:

- Agentes expressam uma idéia de autonomia muito mais forte do que objetos; em particular, agentes decidem por eles mesmos quando executar uma ação requisitada por outro agente;
- Agentes possuem um comportamento flexível (reativo, proativo e social); e objetos possuem um comportamento restrito definido pelo estado interno dos seus métodos e variáveis;
- Sistemas Multiagentes são considerados *multi-threaded*, uma vez que cada agente possui pelo menos uma *thread* em seu controle de fluxo de execução de ações.

2.3 Sistemas Mutiagentes

Devido ao grande crescimento da utilização de computadores em redes - e pode-se tomar a Internet como um importante exemplo prático deste fenômeno -, onde a informação está distribuída através dos diversos nodos que a compõe, situações onde uma entidade computacional possui todo o conhecimento necessário para resolver problemas sem o auxílio de outras estão tornando-se cada vez mais raras. Problemas como aumento da complexidade dos sistemas e a necessidade de tratamento de grandes massas de dados para a resolução de problemas têm levado pesquisadores a buscar métodos de resolução baseados em arquiteturas distribuídas. Isso tem acontecido porque as arquiteturas distribuídas vêm se mostrado muito úteis para resolução de problemas onde a própria natureza é distribuída. Pode-se pensar na aplicação de um modelo monolítico para a resolução de problemas em uma realidade distribuída, onde os eventos ocorrem concorrentemente. No entanto, isto requer uma complexa etapa intermediária de mapeamento que, muitas vezes, resulta em

um algoritmo não-computável. Da mesma forma, a distribuição pode levar à descoberta de algoritmos computacionais que talvez não teriam sido descobertos com uma abordagem centralizada. Em outros casos, como negociação entre empresas independentes, uma abordagem centralizada é inviável, já que cada empresa deseja manter suas informações em um âmbito privado por razões mercadológicas [29].

Sendo um agente uma entidade que encapsula conhecimento sobre algum domínio, nada mais natural do que agrupar agentes que possuam parte do conhecimento envolvido na estratégia de resolução de um problema e que, a partir disso, interajam com o objetivo de complementarem suas habilidades. Assim, da mesma forma que no mundo real existem empresas com funcionários possuidores de diferentes habilidades e que, utilizando essas habilidades, desenvolvem parte das atividades necessárias ao processo produtivo, pode-se compor uma sociedade de agentes onde para cada agente seja alocada um subconjunto das habilidades requeridas pela estratégia de solução onde a cada um seja designado parte das tarefas a serem cumpridas, de acordo com sua disponibilidade de recursos (computacionais, materiais, tempo, etc.). Pode-se distinguir duas principais classes de sistemas com múltiplos agentes [76]:

- **Sistemas de Resolução Distribuída de Problemas**, nos quais os agentes envolvidos são explicitamente projetados para, de maneira cooperativa, atingirem um dado objetivo, considerando-se que todos os eles são conhecidos *a priori* e supondo que todos são benevolentes, existindo desta forma confiança mútua em relação às suas interações; e

- **Sistemas Abertos**, nos quais os agentes não são necessariamente projetados para atingiram um objetivo comum, podendo ingressar e sair do sistema de maneira dinâmica. Neste caso, a chegada dinâmica de agentes desconhecidos precisa ser levada em consideração, bem como a possível existência de comportamento não benevolente no curso das interações.

Dentro desta segunda classificação, estão inseridos os Sistemas Multiagentes. Neste tipo de sistema, investiga-se o comportamento de um conjunto de agentes autônomos, possivelmente pré-existentes, que interagem objetivando a resolução de um problema que está além das capacidades de um único indivíduo [32]. Desta forma, o comportamento global do sistema deriva da interação entre os agentes que fazem parte do sistema [76]. A partir disso, está envolvida a busca por uma funcionalidade neste sistema que permita que estes agentes possam coordenar seus conhecimentos, objetivos, habilidades e planos individuais de uma forma conjunta, em favor da execução de uma ação ou da resolução de algum problema onde se faça necessária a cooperação entre os agentes [9]. Nestes casos, diz-se que o agente exibe um comportamento social [76].

Moulin e Chaib-Draa [44] evidenciam as características que constituem vantagens significativas dos Sistemas Multiagentes sobre um solucionador de problemas monolítico, dentre elas:

- maior rapidez na resolução de problemas através do aproveitamento do paralelismo;
- diminuição da comunicação por transmitir somente soluções parciais em alto nível para outros agentes ao invés de dados brutos para um lugar central;
- mais flexibilidade por ter agentes de diferentes habilidades que são dinamicamente agrupados para resolver problemas; e
- aumento da segurança pela possibilidade de agentes assumirem responsabilidades de agentes que falham.

A justificativa de aplicação da tecnologia de agentes na concepção de Sistemas de Informação é justificada quando o problema possui as seguintes características [32]:

- o domínio envolve distribuição intrínseca dos dados, capacidades de resolução de problemas e responsabilidades;
- necessidade de manter a autonomia de subpartes, sem a perda da estrutura organizacional;
- complexidade nas interações, incluindo negociação, compartilhamento de informação e coordenação;
- impossibilidade de descrição da solução do problema *a priori*, devido à possibilidade de perturbações em tempo real no ambiente (p.ex.: falhas de equipamento) e processos de negócio de natureza dinâmica.

2.3.1 Aspectos a considerar

As características da abordagem multiagentes impõem necessidades que devem ser viabilizadas para que o sistema possa ser considerado eficaz. Considerando o exemplo de uma organização empresarial tradicional como sendo uma sociedade de pessoas que combinam esforços para a resolução de um problema comum, pode-se dizer que essa organização constitui um exemplo de sistema de resolução de problemas de natureza distribuída. Nesse caso, para se obter resultados favoráveis nesta empresa devem ser criadas regras para definir uma estrutura que viabilize o alcance dos seus objetivos e implementação de suas estratégias de funcionamento. Em SMA, deve-se

considerar critérios que viabilizem e garantam a coerência das ações dos agentes, visando atingir de maneira efetiva os objetivos propostos.

Moulin e Chaib-Dras [44] propõem um *framework* que fornece uma estrutura de análise e classificação da maior parte das atividades de pesquisa em Sistemas Multiagentes, do qual pode-se citar duas perspectivas:

- ***Perspectiva do agente*** enfoca elementos que caracterizam o agente envolvido em SMA. São eles: categorias de agente, estrutura e manutenção do conhecimento, habilidades de raciocínio, habilidades de adaptação e aprendizado e arquiteturas de agente.

- ***Perspectiva de grupo*** reúne aspectos de grupo, tais como: organização, coordenação, cooperação, negociação, comportamento coerente, planejamento, comunicação e interação.

Dentro da perspectiva de grupo, pode-se definir três grandes grupos de aspectos a serem considerados no projeto de SMA:

- ***aspectos fundamentais*** definem as características que devem ser viabilizadas para a garantia da compatibilidade entre as ações dos agentes que constituem o sistema.

- ***aspectos arquiteturais*** definem as características que devem ser providas pela arquitetura a ser adotada para a viabilização dos aspectos fundamentais dentro do sistema.

- ***aspectos ambientais*** definem as características do ambiente no qual os agentes do sistema estão inseridos, para que se possa determinar os tipos de técnicas de percepção que devem ser utilizadas por estes agentes.

2.3.2 Aspectos fundamentais

A coerência de um Sistema Multiagentes é viabilizada pela garantia de um comportamento coerente de seus agentes, ou seja, as ações dos agentes fazem sentido em relação aos objetivos comuns do grupo [44]. Dentro da perspectiva de grupo, serão examinadora, a seguir, os aspectos fundamentais a serem considerados para o desenvolvimento de sistemas multiagentes para a garantia da compatibilidade das ações dos agentes.

2.3.2.1 Estrutura

Atualmente, adotar a idéia de **organização** e de **mudanças organizacionais** é importante para criar Sistemas Multiagentes mais adaptáveis. No entanto, é necessário notar a distinção entre os termos **estrutura** e **organização**, como indica [44] .

Entende-se **estrutura** como sendo o padrão de relações de informação e controle entre agentes, bem como a distribuição das habilidades entre eles. Desta forma, a estrutura provê uma visualização de como os problemas são tratados e solucionados pelo grupo e o papel que cada agente desempenha dentro do mesmo, com isso, papéis e relacionamentos são especificados para atender as condições abaixo descritas [44] :

- **Cobertura -** qualquer habilidade necessária para a resolução do problema deve estar inserida no rol de habilidades de ao menos um agente;
- **Conectividade** - agentes devem interagir de maneira que suas habilidades sejam integradas e desempenhadas em contribuição a uma solução global;
- **Potencialidade -** cobertura e conectividade devem ser atingíveis dentro de limitações computacionais e de comunicação, assim como as especificações de confiabilidade do grupo.

Desta forma, cada agente desempenha um ou mais papéis específicos no sistema. Define-se como papel aquilo que é esperado que o agente faça dentro da organização, ou seja, um conjunto de responsabilidades bem definidas dentro do contexto global do sistema que o agente pode cumprir com certo grau de autonomia [76].

Esta perspectiva pode tornar o projeto de sistemas paralelos e distribuídos menos complexo e de gerenciamento mais simples em relação à maioria das metáforas tradicionais de sistemas concorrentes, que, de acordo com Zambonelli em [76], devem considerar:

- Cada agente tem controle próprio sobre o seu processamento, sendo totalmente responsável em cumprir o seu papel;
- As interdependências entre os componentes do sistema são reduzidas, uma vez que cada agente têm embutida a maior parte da funcionalidade para cumprir o seu papel. Essa independência facilita o projeto por fornecer uma clara separação entre os níveis de componente (micronível) e do sistema (macronível).
- Em muitos casos, SMAs pretendem suportar ou controlar alguma organização existente na realidade, pois sua adoção reduz a distância entre os sistemas de

software e os sistemas de mundo real, o que simplifica o desenvolvimento do sistema.

Para Zambonelli [76], em SMA o comportamento autônomo e proativo exibido pelos agentes constituintes sugere que aplicações podem ser projetadas tomando-se como exemplo o comportamento e a estrutura de sociedades humanas.

2.3.2.2 Organização

O conceito de **organização** refere-se ao conjunto de compromissos globais, crenças mútuas e intenções comuns aos agentes, quando agem juntos para atingir um dado objetivo [44]. Estes elementos definem um conjunto de diretrizes a serem seguidas por cada um dos agentes do SMA, descrevendo uma política de interação entre eles.

Estas diretrizes podem impedir a ocorrência de situações caóticas no SMA, criando padrões comportamentais que evitam que o SMA se torne um sistema desorganizado, facilitando ou mesmo viabilizando a realização dos objetivos globais da sociedade.

2.3.2.3 Coordenação

Para Bond e Gasser [9], um sistema multiagentes pressupõe coordenação entre um conjunto existente de agentes autônomos e inteligentes. Fundamentalmente, está envolvida a busca por uma funcionalidade neste sistema, a qual permita que estes agentes possam coordenar seus conhecimentos, objetivos, habilidades e planos individuais de uma forma conjunta, em favor da execução de uma ação ou da resolução de algum problema. Para Jennings [32], coordenação é o processo pelo qual um agente raciocina sobre suas ações locais e ações de outros agentes com o objetivo de garantir que funcione de maneira coerente[1].

A definição de estratégias que conciliem os interesses individuais de cada agente para que as atividades relacionadas desenvolvam-se de modo coordenado é um dos aspectos fundamentais a serem considerados no projeto de sistemas multiagentes [61].

Jennings, em [32], afirma que a necessidade de coordenação entre múltiplos agentes surge do fato de que:

- existem dependências entre as ações dos agentes, ou seja, a ação de um agente pode ser pré-requisito da ação de outro agente; e
- nenhum indivíduo tem competência, recursos ou informação suficiente para resolver um problema completo de forma independente; onde

[1] Coerência significa quão bem um sistema comporta-se como uma unidade [29].

- deve ser garantido o respeito às restrições globais do problema; e
- devem ser viabilizados procedimentos que garantam a harmonia na execução de uma tarefa de forma conjunta por mais de um agente.

A coordenação é um fator vital para o funcionamento de um SMA, pois sem coordenação os benefícios advindos da resolução distribuída de problemas desaparecem e a comunidade pode degenerar em uma caótica coleção de indivíduos que agem de forma incoerente em relação ao sistema como um todo [44, 32]. Para isso, basta que um único agente tenha uma visão parcial ou imprecisa do sistema e que suas ações possam interferir nas ações de outros agentes ao invés de suportá-las [44].

Segundo Lesser e Corkill apud [32], os objetivos do processo de coordenação visam garantir que:

- todas as partes necessárias ao sistema estejam inseridas nas capacidades funcionais de ao menos um agente;
- os agentes interajam de maneira a permitir que suas atividades sejam desenvolvidas e integradas em uma solução global;
- os membros da sociedade atuem com propósitos e consistentemente;
- todos esses objetivos sejam atingíveis dentro das limitações computacionais impostas e dos recursos disponíveis.

Durfee em [18] identifica três fatores básicos que devem estar presentes para uma coordenação bem sucedida:

- existência de uma estrutura que permita aos agentes interagirem de forma preditiva;
- flexibilidade nas interações de tal forma que os agentes possam operar em ambientes dinâmicos e agir satisfatoriamente com uma visão parcial e imprecisa da sociedade;
- os agentes devem possuir conhecimento e capacidade de raciocínio suficiente para explorar esta estrutura e flexibilidade.

A maneira mais fácil de garantir um comportamento coerente dentro de um SMA é implantar um agente que tenha uma perspectiva mais ampla do sistema, atuando como um coordenador que, reunindo informações sobre toda a sociedade, seria responsável por criar planos e atribuir tarefas aos membros da sociedade [32].

No entanto, esta não é uma abordagem prática em sistemas reais por ser muito difícil criar um agente que se mantenha informado sobre todas as intenções e crenças de todos os agentes da sociedade [44]. Além disso, um coordenador centralizado se tornaria um gargalo de comunicação, que degradaria a performance do sistema.

Também deve ser consideraro o fato de que, nesta abordagem, uma falha do agente coordenador comprometeria o funcionamento de todo o sistema [32], apesar de que, neste caso, poderiam ser adotados mecanismos de tolerância à falhas onde outro agente poderia assumir o papel de coordenador.

A partir disso, uma problemática apresentada para o problema de sistemas multiagentes passa a ser a manutenção da coerência global sem um controle global explícito [29]. Para o autor, "os agentes devem raciocinar a respeito das ações, mas também sobre o processo de coordenação em si".

Uma desvantagem advinda da distribuição do controle e dos dados é a dificuldade de se ter conhecimento sobre o estado global do sistema, que está disperso através da comunidade, sendo que cada indivíduo possui uma visão parcial e imprecisa desta perspectiva. Seghrouchni, em [61], elicita os principais requisitos para a coordenação, abaixo listados:

- comunicação entre os agentes;
- reconhecimento das interações potenciais entre planos; e
- negociação entre os agentes.

O conceito de coordenação define aspectos gerais de interação entre agentes de forma a viabilizar a coesão entre seus comportamentos e ações em relação aos objetivos globais do sistema. A partir disso, apresenta-se uma taxonomia para os processos de coordenação (Figura 2.11) que possui duas abordagens principais [29]:

- **Cooperação -** é a coordenação entre agentes não-antagônicos[2].
- **Negociação -** é a coordenação entre agentes competitivos ou que agem em interesse próprio.

2.3.2.3.1 Cooperação Lux e Steiner [42] consideram que a cooperação acontece quando vários agentes planejam e executam suas ações de uma forma coordenada, sendo que é requerida quando:

- o agente não consegue encontrar um plano local que contemple o objetivo;

[2]O conceito de não-antagônico define que os interesses dos agentes não são conflitantes.

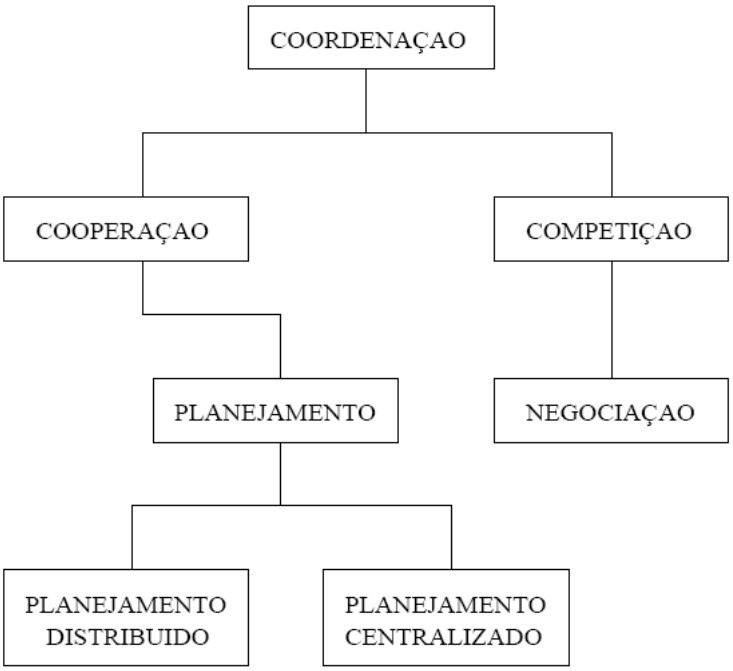


Figura 2.11: Taxonomia da coordenação - extraído de [29]

- o plano adequado ao objetivo envolve ações de outros agentes;
- o agente considera que um plano pode ser melhor (de menor custo ou mais eficiente) do que um plano local;
- durante a fase de planejamento:
  - o agente encontra planos incompletos, que podem ser completados em cooperação com outros agentes; ou
  - o agente encontra eventos para os quais não está habilitado a responder, mas sabe que outros agentes estão.

Os objetivos genéricos para a cooperação entre agentes de acordo com [44] são:

- diminuição do tempo de execução de uma tarefa através do paralelismo;
- aumento do escopo de tarefas executáveis através do compartilhamento de recursos;
- maior probabilidade de finalização de uma tarefa em função de sua dupla incumbência, a ser realizada possivelmente através de distintos métodos de execução; e

- diminuição da interferência entre tarefas evitando interações prejudiciais.

O processo de planejamento constitui uma forma especializada do processo de cooperação, o qual tem como produto um conjunto de atividades organizadas com um curso de ação definido, em que estas atividades são distribuídas entre agentes capacitados a executá-las. Esse planejamento pode ocorrer de duas formas:

- **centralizada -** um único agente constrói o plano; ou
- **distribuída** - pressupõe que o plano é construído por mais de um agente, sendo considerada quando um único agente não possui uma visão global das atividades do grupo.

Segundo Agre e Chapman, em [1], a utilização de planos pode ser vista de duas maneiras:

- **Plano como programa -** O uso de plano é a execução de determinado procedimento, sendo que esta visão implica na execução de planos de forma independente do domínio, e a construção de planos pode ser independente ou dependente do domínio, algorítmica ou baseada em casos, formalmente correta ou heurística. Nesta visão, o plano pode ser decomposto em ações primitivas que podem ser simplesmente executadas através de um dispositivo independente do domínio. As razões para dúvidas da visão de plano como programa são:
  - possui problemas intratáveis computacionalmente;
  - é inadequada para um mundo caracterizado por eventos imprevisíveis, tais como as ações de outros agentes;
  - requer que planos sejam muito detalhados;
  - falha ao tratar problemas de relacionamento do texto do plano para a situação concreta.
- **Planos como comunicação -** O uso de plano é seguir instruções em linguagem natural, sendo que um plano não determina diretamente as ações de um agente. Ao invés disso, um plano é um recurso que o agente utiliza para decidir o que fazer. Nesta visão, resolver qual atividade um plano sugere requer um esforço interpretativo contínuo. A única situação completa conhecida é a situação inicial passada ao planejador. Durante a execução, as circunstâncias que aparecem podem somente determinar ramificações condicionais ou causar um retorno do controle ao planejador se algo está obviamente errado.

2.3.2.3.2 Negociação A negociação representa um papel fundamental em atividades cooperativas dentro de sociedades humanas, permitindo que pessoas resolvam conflitos que interferem no comportamento cooperativo [44].

O processo de negociação atua sobre o melhoramento de concordância acerta de pontos de vista comuns ou planos através de compartilhamento de estruturas de informações relevantes, ocorrendo entre agentes com objetivos diferentes no qual decisão conjunta é alcançada por dois ou mais agentes, cada um tentando alcançar seus objetivos individuais [29]. Para o autor, as seguintes abordagens de negociação podem ser utilizadas:

- **centradas no ambiente** - foca o problema de como as regras do ambiente podem ser desenvolvidas para que os agentes nele envolvidos, independente de suas origens, capacidades, ou intenções, interajam produtivamente e razoavelmente. Nesse caso, o mecanismo de negociação resultante deve possuir as seguintes propriedades:
  - ***eficiência*** - os agentes não devem desperdiçar recursos para chegar a um acordo;
  - ***estabilidade*** - nenhum agente deve ter um incentivo para desviar das estratégias acordadas;
  - ***simplicidade*** - deve impor baixas demandas computacionais e de largura de banda[3] sobre os agentes;
  - ***distribuição*** - não deve requerer um tomador de decisões centralizado;
  - ***simetria*** - não deve haver diferenciação no tratamento dos agentes por razões arbitrárias ou inapropriadas.
- **centradas no agente** - assume que os agentes são economicamente racionais. O conjunto de agentes deve ser pequeno e necessita ter uma linguagem e abstração do problema comuns, e têm que alcançar uma solução comum. A partir desses pressupostos, foi desenvolvido um protocolo de negociação unificado ([57]), no qual agentes criam um acordo que constitui um plano conjunto que satisfará todos os seus objetivos, cuja *utilidade* para um agente é a quantidade que ele pagará subtraindo o custo do acordo. Agentes discutem um *conjunto de negociações*, que é o conjunto de todos os acordos que têm uma utilidade positiva para todo agente. Do processo de negociação, três situações que podem surgir são elicitadas abaixo:
  - ***conflito*** - o conjunto de negociações está vazio;

[3]Define-se largura de banda como sendo a capacidade de transmissão de um canal de comunicação

- ***compromisso*** - agentes preferem trabalhar de maneira isolada, mas se isto não for possível, chegarão a um acordo negociado;
- ***cooperativo*** - todos os acordos, no conjunto de negociações, são preferidos por ambos agentes prioritariamente à realização dos objetivos de forma isolada.

2.3.3 A cooperação entre agentes

Dentro de um Sistema Multiagentes, uma política de cooperação se faz necessária, uma vez que é através desse mecanismo que os agentes expressam suas necessidades a outros agentes a fim de realizar uma determinada tarefa. O mecanismo de cooperação de agentes visa determinar a maneira como os agentes expõem suas necessidades a outros agentes para atingirem determinados objetivos.

O processo de cooperação pode acontecer de duas maneiras: Partilha de Tarefas (*Task Sharing*) e Partilha de Resultados (*Result Sharing*). A primeira se caracteriza pela necessidade de agentes auxiliares durante a execução de uma tarefa por um determinado agente, enquanto que na segunda, os agentes disponibilizam informações para a sociedade, prevendo que algum outro agente tenha necessidade delas em determinado momento.

Desta forma, na grande maioria das vezes, durante a realização de tarefas, os agentes necessitam da ajuda dos outros agentes da sociedade ou, de forma contrária, utilizam outros agentes para que o objetivo geral do sistema possa ser atingido. Conseqüentemente, a localização de um determinado agente dentro da sociedade se faz necessária pra que a cooperação possa ser efetuada. As principais abordagens de arquiteturas de sociedades de agentes, onde a cooperação está em destaque são: a arquitetura quadro-negro, a arquitetura de troca de mensagens e a arquitetura federativa, descritas a seguir.

2.3.3.1 Arquitetura quadro-negro

Em uma sociedade de agentes baseada na arquitetura quadro-negro (*Blackboard*) os agentes não se comunicam entre si de uma maneira direta, mas sempre através de um quadro-negro. Este tipo de arquitetura não surgiu com o aparecimento dos sistemas multiagentes, sendo utilizada anteriormente como solução para outros paradigmas [oliveira, 1996].

O quadro-negro é uma estrutura de dados persistente onde existe uma divisão em regiões ou níveis, visando facilitar a busca de informações. Tal estrutura é um meio de interação entre os agentes (uma espécie de repositório), no qual estes escrevem e lêem mensagens que serão usadas para atingir o objetivo do sistema. Pode-se assim dizer que um quadro-negro é uma memória de compartilhamento global

onde existe uma quantidade de informações e conhecimento usados para leitura e escrita por parte dos agentes.

Em Sistemas Multiagentes, os quadros-negros são utilizados como um repositório de perguntas e respostas. Os agentes que necessitam de alguma informação escrevem seu pedido no quadro à espera que outros agentes respondam, à medida que acessem o mesmo. O processo de gravação e recuperação das mensagens no quadro-negro pelos agentes pode se tornar demorado demais para um sistema em tempo real. Essa particularidade faz o uso da arquitetura quadro-negro ser inviável para sistemas dessa natureza; em que o tempo de resposta adequado é fundamental para satisfazer sua funcionalidade.

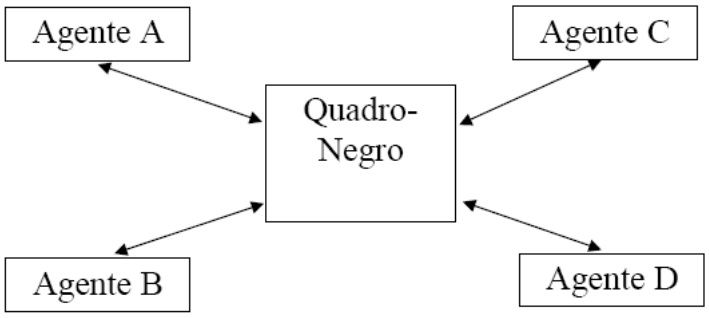


Figura 2.12: Arquitetura quadro-negro - extraído de [48]

2.3.3.2 Arquitetura de troca de mensagens

Nesta arquitetura (*Message Passing*), os agentes se comunicam diretamente uns com os outros diretamente, através de mensagem assíncronas. Não existe o papel do quadro-negro como intermediário na interação, mas pode existir um agente facilitador de comunicação.

Desta forma, é necessário que cada agente saiba os nomes e endereços de todos os agentes que formam o sistema para que as mensagens possam ser trocadas. Este método é mais eficiente no sentido de obter as mensagens em tempo hábil, mas por outro lado, sua implementação é dificultada pelo crescimento das diferentes mensagens trocadas pelos agentes que compõem a sociedade.

Para que as trocas de mensagens ocorram de maneira adequada entre os agentes é necessário estabelecer um protocolo de conversação. O protocolo é quem dita as regras e impõe o formalismo necessário para que os as mensagens sejam encaminhadas e compreendidas pelos agentes.

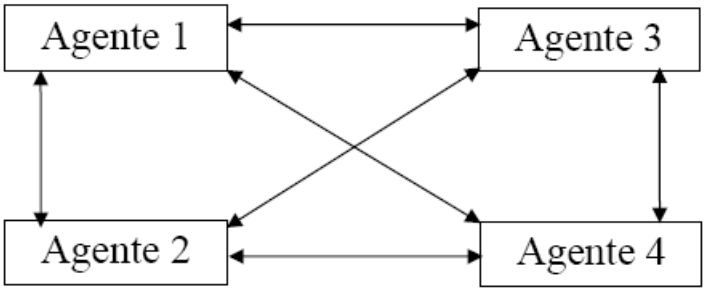


Figura 2.13: Arquitetura troca de mensagens - extraído de [48]

2.3.3.3 Arquitetura federativa

Considerando uma arquitetura de troca de mensagens, onde o número de agentes é muito grande, a emissão de uma mensagem do tipo *broadcast* levará um tempo que pode inviabilizar todo o processo de comunicação do sistema.

Diante deste problema, surgiu a arquitetura federativa (*Federal system*), Figura 2.14,onde os agentes da sociedade são divididos em grupos ou federações segundo um critério de agrupamento escolhido. Junto a cada grupo de agentes encontram-se os agentes facilitadores responsáveis por receber a mensagem que chega a cada grupo e por encaminhá-la para o agente destinatário presente naquele grupo. A vantagem deste tipo de arquitetura é que o agente facilitador tem a propriedade de identificar se a mensagem recebida que chega é destinada a algum agente de seu grupo e, sendo este o caso, fazer a devida entrega.

A arquitetura federativa propõe a diminuição do fluxo de mensagens, principalmente as desnecessárias, entre os agentes que formam a sociedade, pois os facilitadores têm a capacidade de remetê-las ao respectivo destinatário sem a necessidade de enviá-las a todos os agentes.

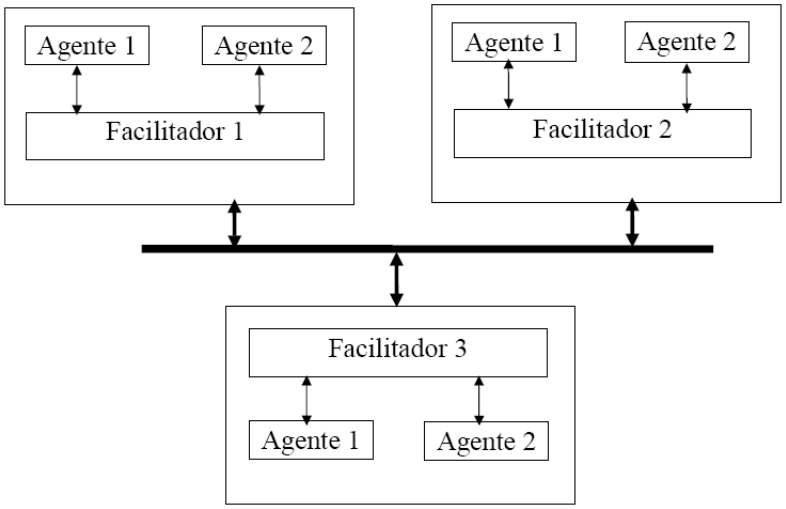


Figura 2.14: Arquitetura federativa - extraído de [48]

2.4 Considerações

O presente capítulo surge como a primeira referência do embasamento teórico deste trabalho. Nele foram apresentadas informações acerca do assunto de Sistemas Multiagentes bem como todas as informações que servirão como um dos pilares para a proposta de identificação de papéis de agentes inteligentes. Desde as características mais básicas como autonomia, pro-atividade e reatividade até conceitos relacionados ao comportamento dos agentes dentro de uma sociedade e aos processos de cooperação entre os mesmos, todos terão um papel relevante durante o processo de pesquisa que dará origem ao resultado final deste estudo. No capítulo seguinte, será discutido o assunto referente a metodologias voltadas ao paradigma orientado a agentes.

Capítulo 3

Metodologias para SMAs

"A liberação da energia atômica mudou tudo,
menos nossa maneira de pensar"
— Albert Einstein

Sistemas Multiagentes são cada vez mais usados em desenvolvimento de software. Segundo [5], estes representam uma alternativa para se resolver problemas em processos de negócio. Uma grande vantagem deste tipo de sistema é a possibilidade de descentralização e distribuição na tomada de decisão.

A construção de tais sistemas necessita seguir uma metodologia de desenvolvimento. Desta forma, muitas metodologias vêm sendo propostas ultimamente com o objetivo de facilitar a construção de tais sistemas. De acordo com [73], metodologias geralmente consistem em uma coleção de modelos, e associados a estes modelos, um conjunto de *guidelines*. O intuito destes modelos é formalizar o sistema de maneira compreensível. Estes modelos, no início do processo de especificação, representam o sistema de maneira abstrata, e durante o andamento das fases de análise e projeto se tornam mais detalhados, passando a ficar mais perto da implementação. Nas subseções seguintes, serão apresentadas as principais metodologias para desenvolvimento de SMA e um estudo sobre o método de identificação de agentes será realizado em cada uma delas.

Neste capítulo serão apresentadas três metodologias para a modelagem de sistemas multiagentes que foram introduzidas no cenário acadêmico. Para cada uma destas metodologias uma descrição sobre o seu processo é apresentada e posteriormente é realizada uma análise sobre a forma como a metodologia trata a identificação de papéis de agentes durante o processo de modelagem de um SMA.

3.1 MaSE

A metodologia Multiagent Systems Engineering (MaSE) surgiu de esforços realizados por pesquisas do Instituto de Tecnologia da Força Aérea Americana. Em [16], o autor destaca que o foco principal desta metodologia é auxiliar o projetista a ter um conjunto inicial de requisitos e analisar, projetar e implementar sistemas multiagentes.

A MaSE é semelhante às metodologias tradicionais de engenharia de software, porém é orientada para a construção de sistemas multiagentes. A metodologia divide-se em duas fases: Análise e Projeto. A primeira é dividida nos seguintes passos: Capturar Objetivos, Aplicar Casos de Uso e Refinar Papéis. Os passos de Criar Classes de Agente, Construir Conversas, Montar Classes de Agente e Projeto do Sistema são parte integrante da fase de Projeto. A Figura 3.1 mostra uma visão geral das fases da metodologia.

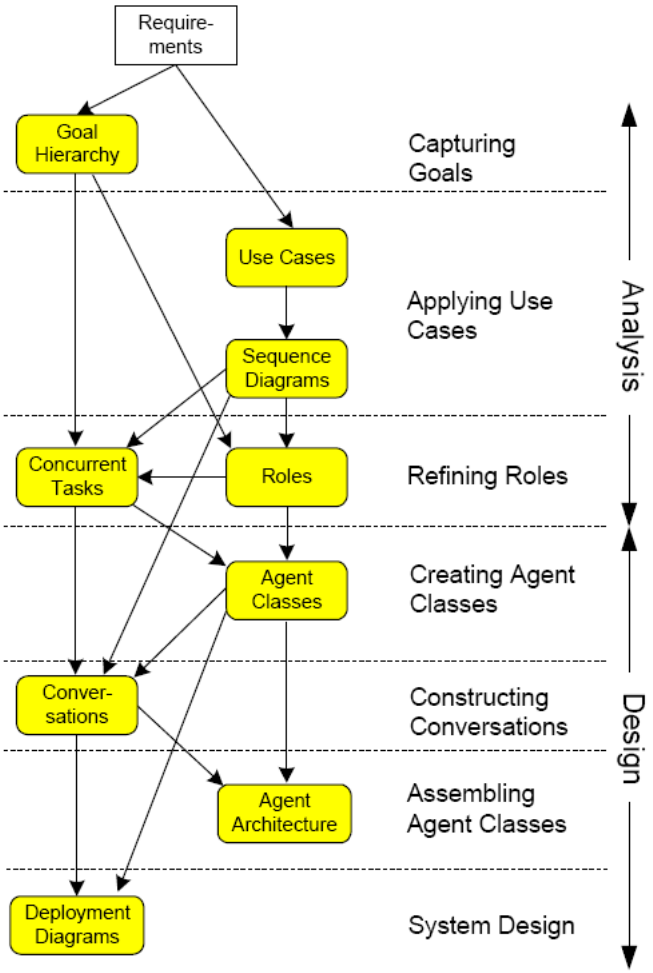


Figura 3.1: Visão geral das fases da metodologia MaSE - extraído de [69]

Na etapa denominada Capturar Objetivos, ocorre a identificação dos objetivos do sistema bem como a estruturação dos mesmos [15]. A identificação dos

objetivos deve ser extraída dos requisitos definidos na especificação inicial do sistema, ou seja, devem ser capturados os objetivos de alto-nível. Esta etapa deve ser realizada com muito cuidado, pois uma alteração tardia pode gerar um impacto indesejável no projeto. Os objetivos são organizados por sua importância, em que cada nível superior contém os objetivos mais gerais do sistema e os níveis inferiores contém os objetivos mais específicos, ou sub-objetivos.

A etapa chamada Aplicar Casos de Uso é de grande importância para o mapeamento de papéis e para a associação de tarefas no sistema [69]. Neste, devem ser desenhados os comportamentos esperados para o alcance dos objetivos, por meio de diagramas de caso de uso. Elaborados os casos de uso, estes devem ser reestruturados em diagramas de seqüência, auxiliando na determinação das comunicações requeridas pelo sistema multiagentes. Estes diagramas mostram a seqüência de eventos entre múltiplos papéis, e por conseqüência definem a comunicação mínima do sistema. Papéis identificados nesta etapa formam um conjunto inicial que será detalhado no próximo passo com o objetivo de gerar a completa identificação dos papéis do sistema.

Os objetivos da etapa chamada Refinar Papéis são a identificação completa dos papéis necessários ao sistema e o desenvolvimento de tarefas, definindo o comportamento dos papéis e os padrões de comunicação. Papéis são capturados no Modelo de Papéis, onde normalmente, cada papel é associado a um único objetivo. Todavia, segundo o autor, em [16], existem situações em que é útil combinar múltiplos objetivos em um único papel por conveniência ou eficiência. Definidos os papéis do sistema, tarefas devem ser criadas e associadas a cada um destes. Deloach, em [15], destaca que as maiores dificuldades na metodologia MaSE são a transformação de papéis em classes de agente, a definição de conversas entre agentes e a definição dos comportamentos internos dos agentes. Devido a isto, devem ser definidas tarefas de alto-nível que podem ser transformadas em papéis específicos de agentes, auxiliando na definição dos componentes internos dos agentes assim como no detalhamento das conversas entre agentes. O envio de mensagens entre papéis e tarefas é especificado por tarefas concorrentes. Estas possibilitam uma definição de alto-nível dos protocolos de interação que requerem coordenação entre múltiplos agentes.

Na etapa seguinte, denominada Criar Classes de Agentes, devem ser identificadas as classes de agente que o sistema deverá conter. Para tal, deve ser criado um Diagrama de Classes de Agente. Além da identificação dos agentes do sistema, este diagrama deve conter as conversas entre os mesmos. A comunicação entre os papéis do sistema é herdada pelas classes de agente geradas, ou seja, conversas entre papéis tornam-se conversas entre classes de agente. Atribuídos todos os papéis, a organização geral do sistema está definida. Para uma organização mais eficiente, segundo o autor que afirma em [16], afirma que é desejável combinar dois papéis que

compartilham um alto volume de tráfego de mensagens.

Na etapa Construir Conversas, devem ser definidos os protocolos de comunicação entre os agentes. Para modelar uma conversa, devem ser criados dois Diagramas de Classe de Comunicação, um para a classe de agente responsável pelo início da conversa, denominada *conversation initiator*, e outro para a classe de agente responsável pela resposta, conhecida como *conversation responder*, estabelecendo uma comunicação ponto a ponto. O *initiator* sempre inicia uma conversa pelo envio da primeira mensagem. Quando o responder recebe uma mensagem, ele compara esta com suas conversas ativas. Se achar a conversa correspondente, o responder transfere a conversa para um estado novo e executa quaisquer atividades ou ações requeridas. Em caso contrário, o responder assume que a mensagem é uma requisição para iniciar uma nova conversa e compara isso com todas as conversas possíveis em que pode participar com o *initiator*.

Já a etapa denominada Criar Agentes é responsável pela criação da parte interna do agente. Enquanto o projetista pode usar arquiteturas existentes, como o BDI, ou projetá-las do início, investigações atuais derivam a arquitetura de agentes diretamente dos papéis e tarefas definidas na fase de análise. Esta visão tem a vantagem de um mapeamento mais direto da análise para o projeto, embora perca alguma flexibilidade e potencial reuso. Basicamente, cada tarefa de cada papel executado por um agente define um componente em uma classe de agente. A tarefa concorrente, citada na etapa de refinamento de papéis, é transformada em uma combinação de componentes internos ao diagrama de estados e em um conjunto de conversas. Atividades identificadas na tarefa concorrente tornam-se métodos do componente.

Por fim, a etapa de Projeto do Sistema é considerada a mais simples da metodologia, onde a configuração do sistema a ser implementado é especificada. Um diagrama de implantação deve ser criado, onde serão mostrados números, tipos e locais dos agentes no sistema. Possíveis detalhes de implementação ainda não definidos devem ser decididos nesta etapa, como por exemplo, a linguagem de programação e o framework de comunicação.

Uma maneira de criar um agente proposta nesta metodologia, é a criação de tarefas de alto-nível, e que podem ser atribuídas a papéis específicos para agentes. Os agentes também podem ser definidos através dos papéis que foram encontrados na fase de análise. Mesmo com essas propostas, ainda não se identifica um método de identificação ou criação de agentes que seja automático ou baseado em algum artefato de alto-nível, como por exemplo, diagrama de caso de uso.

3.2 Gaia

A metodologia Gaia surgiu como uma resposta que foi proposta por Wooldridge, Jennings e Kinn em [72] para o problema introduzido no início deste capítulo: a necessidade por uma metodologia que forneça suporte para a análise, para o projeto e desenvolvimento de aplicações sob o paradigma orientado a agentes. Os autores afirmam que se agentes se tornaram um novo paradigma para desenvolvimento de sistema, então se faz necessária a criação de técnicas de desenvolvimento de software direcionadas a este novo conceito de componente de *software*.

Os autores consideram Gaia uma metodologia apropriada para o desenvolvimento de aplicações reais de larga-escala orientadas a agentes [77]. Diante dessa afirmação, um conjunto de ítens são identificados como características principais seguidas pela metodologia:

- Agentes são sistemas computacionais complexos, que fazem um significando uso de recursos computacionais (imagina cada agente como possuidores de recursos de um processo UNIX);
- Assume-se que o objetivo é obter um sistema que maximiza alguma medida de qualidade global, mas que pode ser sub-otimizada sob o ponto de vista de componentes de sistema;
- Agentes são heterogêneos, onde agentes diferentes podem ser implementados usando diferentes linguagens de programação, arquiteturas ou técnicas;
- A estrutura organizacional dos agentes no sistema é estática, não sofrendo alterações em tempo de execução;
- As habilidades bem como os serviços oferecidos pelos agentes são estáticos, ou seja, não sofrem alterações em tempo de execução;
- O sistema como um todo contém um número relativamente pequeno de tipos diferentes de agentes (menor do que 100).

Gaia trabalha tanto com o nível intra-agente como o macro-agente, onde aspectos relacionados a organização dos agentes são trabalhados. Essa característica representou um avanço em relação as outras metodologias quando de sua apresentação no artigo "The Gaia Methodolofy for Agent-Oriented Analysis and Design", uma vez que essas apenas referenciavam as estruturas intra-agentes.

A intenção principal desta metodologia é permitir que o analista vá sistematicamente dos requisitos até um projeto de sistema que seja suficientemente detalhado de forma que este possa ser implementado diretamente. Os autores tratam o

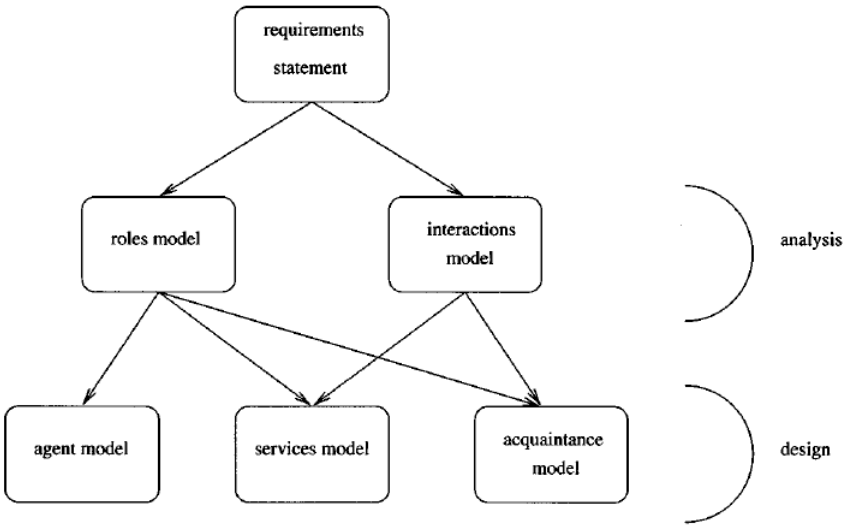


Figura 3.2: Relacionamentos entre os modelos da metodologia Gaia - adaptado de [72]

processo de elicitação dos requisitos como uma fase independente da fase de projeto. Aplicando a metodologia, o analista vai de conceitos abstratos até conceitos altamente concretos. Cada movimento sucessivo dentro da metodologia introduz uma maior inclinação para a implementação, e diminui o espaço de possíveis sistemas que possam ser implementados para satisfazer os requisitos iniciais. Análise e projeto podem ser vistos como um processo crescente de desenvolvimento de modelos detalhados do sistema a ser construído.

A metodologia Gaia usa algumas terminologias e notações provenientes das fases de análise e projeto da modelagem orientada a objetos. Entretanto, isto não é apenas uma simples tentativa para aplicar tais métodos no desenvolvimento orientado a agentes. Ao contrário, Gaia provê um conjunto de conceitos específicos para agentes com o qual o analista ou engenheiro de sistemas poderá entender e modelar um sistema complexo. Particularmente, essa metodologia encoraja o desenvolvedor a pensar na construção de sistemas multiagentes como um processo de projeto organizacional.

Os principais conceitos dessa metodologia podem ser divididos em duas categorias: abstrato e concreto; Entidades abstratas são aquelas utilizadas durante a fase de análise para conceitualizar o sistema, mas que não necessariamente possuem alguma relação direta com o sistema. Entidades concretas, ao contrário, são utilizadas dentro do processo de projeto, e serão reproduzidas diretamente no sistema. A Tabela 3.1 sumariza os conceitos abstratos e concretos explicados acima.

O papel de um agente define o que é esperado que esse realize dentro de uma organização, tanto interagindo com outros agentes bem como a própria organização

Tabela 3.1: Conceitos abstratos e concretos da metodologia Gaia - adaptado de [72]

Abstract concepts	Concrete concepts
Roles	Agent Types
Permissions	Services
Responsibilities	Acquaintances
Protocols	
Activities	
Liveness properties	
Safety properties	

em si. Freqüentemente, um papel de agente é simplesmente definidos em termos de uma tarefa específica que esse deverá efetuar dentro do contexto da organização como um todo. Entretanto, para os autores, a noção de papel de agente é muito mais precisa; essa noção dá ao agente uma posição bem definida dentro da organização, sendo associada também à um conjunto de comportamentos esperados.

Os modelos organizacionais dos papéis definem precisamente todos os papéis que constituem a organização computacional. Tais modelos fazem isto em termos de funcionalidades, atividades, e responsabilidades, bem como em termos de padrões e protocolos de comunicação. Os modelos de interação organizacional descrevem os protocolos que governam as interações entre os papéis. Além disso, estes modelos descrevem as características e a dinâmica de cara protocolo.

3.3 Prometheus

De acordo com os autores Lin Padgham e Michael Winikoff [47], a metodologia Prometheus tende a ser uma metodologia prática. Os autores buscam com esta proposta uma metodologia que seja completa, por isso Prometheus provê tudo o que é necessário para especificar e projetar sistemas multiagentes. O objetivo dos autores é desenvolver um processo com produtos "entregáveis", o qual pudesse ser introduzido na indústria e ensinado para estudantes que não tivessem conhecimento na área de agentes para que então, desenvolvessem sistemas inteligentes. Prometheus foi criado para ser usado por pessoas sem experiência na área de sistemas multiagentes [68]. Algumas de suas outras características são listadas abaixo [68]:

- Prometheus é uma metodologia detalhada, fornecendo guias detalhados de como realizar os inúmeros passos que a compreendem.
- Prometheus provê suporta (embora não esteja limitado apenas para) o projeto de agentes que são baseados em objetivos e planos. Os autores acreditam que

a parte mais significante dos benefícios do uso da metodologia é proveniente da engenharia de *software* orientada a agentes que trata do uso de objetivos e planos para modelar os agentes.

- Prometheus provê um conjunto de atividades que vão desde as especificações de requisitos até a detalhamentos do projeto.

A metodologia Prometheus compreende um conjunto de três fases, descritas abaixo e ilustradas pela Figura 3.3:

- **System Specification** (especificação de sistema) - onde o sistema é especificado por meio de objetivos e cenários; a interação entre o sistema e o ambiente que este estará inserido é descrito através de ações, percepções e dados externos. Nesta fase as funcionalidades do sistema também são definidas.

- **Architectural design** (projeto arquitetural) - nesta fase os tipos de agentes são identificados. Também tem-se que a estrutura do sistema como um todo é capturada por meio de um diagrama de visão geral do sistema. Os cenários descritos na fase anterior são então traduzidos para protocolos de interações.

- **Detailed design** (projeto detalhado) - a terceira e última fase compreende no desenvolvimento e na definição dos detalhamentos dos agentes em capacidades, dados, eventos e planos. Além disso, os diagramas de processos são considerados como pontes entre protocolos de interação e os planos.

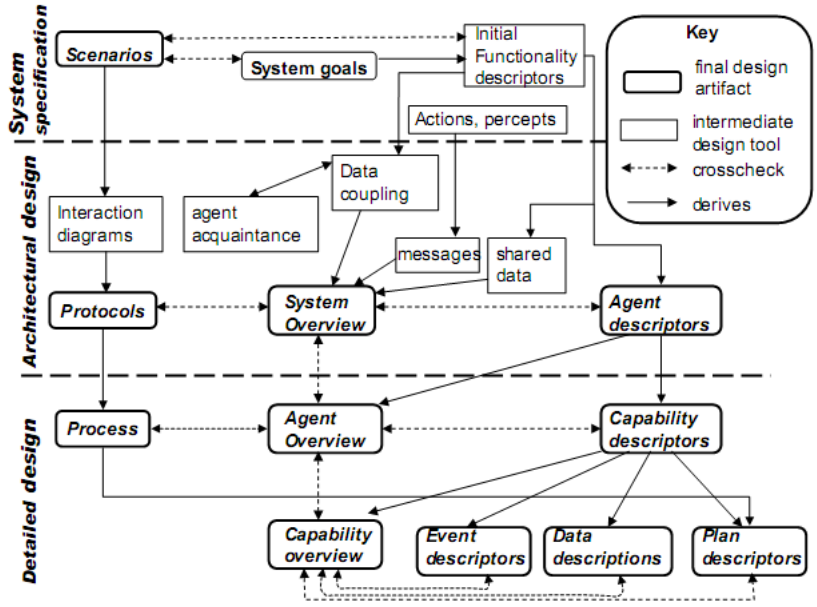


Figura 3.3: Fases da metodologia Prometheus - extraído de [68]

3.3.1 Especificação do Sistema

A primeira fase, relativa a especificação do sistema, começa com uma simples descrição do sistema, que podem conter alguns parágrafos. A partir desta descrição, os requisitos do sistemas são definidos em termos de objetivos, cenários de casos de uso, funcionalidades, ações e percepções. Estas duas últimas são relativas a interação do sistema com o ambiente em que este estará inserido. Como identificado em [68], não há uma seqüência explícita para a definição dos requisitos, porém, o trabalho realizado com uma característica pode levar idéias posteriores em outra. Por exemplo, o autor exemplifica que modelar os objetivos do sistema é um passo inicial para a definição de casos de uso. Entretanto, a criação dos detalhamentos dos casos de uso geralmente podem indicar sub-objetivos (criando assim uma hierarquia de objetivos) que devem ser considerados quando da construção do sistema. As funcionalidades do sistemas são criadas a partir do agrupamento de objetivos. Esse agrupamento deve levar em conta a organização e as relações entre os objetivos, para que não sejam criadas funcionalidades com objetivos diferentes. Outro aspecto da especificação do sistema são os cenários de casos de uso. Esses são uma descrição detalhada de uma seqüência de eventos associada a um objetivo ou a alguma resposta para um evento em particular.

3.3.2 Projeto Arquitetural

A segunda etapa, que compreende o projeto arquitetural do sistema a ser construído, envolve a identificação dos tipos de agentes a serem construídos dentro do sistema, a definição dos protocolos de interação entre os agentes e a definição da arquitetura do sistema como um todo. A construção dos tipos de agentes é a etapa mais importante da segunda fase. Nessa metodologia os tipos de agentes são construídos por meio da combinação de duas ou mais funcionalidades. Diferentes agrupamentos entre funcionalidades fornecem alternativas de projeto que são avaliadas de acordo com a coesão entre os tipos de agentes e com o grau de acoplamento entre estes. Algumas características devem ser consideradas quando da criação de tipos de agentes: (a) se duas funcionalidades se relacionam entre si, faz sentido agrupá-las no mesmo tipo de agente, do contrário não devem ser agrupadas; (b) se duas funcionalidades utilizam as mesmas informações, estas devem ser agrupadas no mesmo tipo de agente.

3.3.3 Projeto Detalhado

Por fim, na terceira fase, cujo detalhamento do projeto é definido por meio da definição da estrutura interna dos agentes em termos de habilidades[1] e dos protocolos de interação entre estes. As habilidades são definidas em termos de eventos, planos e dados ou podendo ser definidas em função de outras habilidades, criando desta forma, uma hierarquia de relacionamento entre elas. Os diagramas e os protocolos de interação definidos na fase anterior são transformados em diagramas de processos que representam a comunicação entre os agentes sob o ponto de vista dos mesmos. A notação utilizada para representar os diagramas de processos é uma extensão ao diagrama de atividades da UML [68].

3.4 Tropos

Para os autores [23], Tropos surge como uma metodologia que visa cobrir todas as fases do desenvolvimento de sistemas. Esta metodologia é baseada em conceitos para realizar a modelagem e especificação de requisitos. Tropos utiliza um *framework* que provê noções como atores, objetivos e dependências entre atores que são utilizados durante todo o ciclo de desenvolvimento. Além disso, a metodologia enfatiza aspectos relacionados às fases iniciais da análise de requisitos, permitindo um melhor entendimento do ambiente onde o software irá operar. De acordo com a afirmação dos autores, a Figura 3.4 ilustra um comparativo entre o nível de cobertura entre a metodologia Tropos e algumas outras.

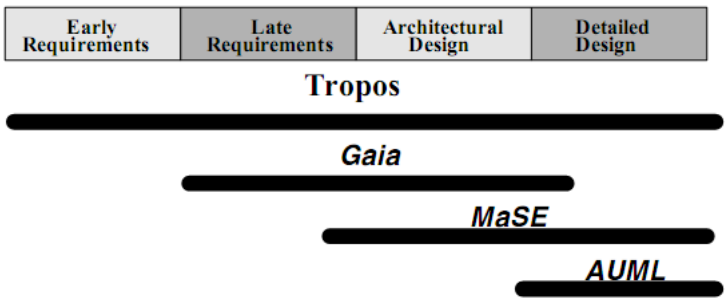


Figura 3.4: Comparativo entre Tropos e outras metodologias - extraído de [23]

A metodologia Tropos é dividida nas fases de **Requisitos Iniciais**, **Requisitos Finais**, **Projeto Arquitetural**, **Projeto Detalhado** e **Implementação**. Estas fases serão detalhadas a seguir para um melhor entendimento do funcionamento da metodologia.

[1] O termo utilizado pelos autores da metodologia Prometheus é *Capabilities*, porém, nesta dissertação foi traduzido pela palavra "habilidades"

3.4.1 Requisitos Iniciais

Durante esta fase os principais *stakeholders*[2] são identificados, juntamente com seus respectivos interesses. Os *stakeholders* são representados como atores, enquanto os seus interesses são representados como objetivos.

3.4.2 Requisitos Finais

Na análise dos requisitos finais, deve ser feita uma modelagem sistema, que passa a ser representado por um ator. Nesta, o modelo conceitual é estendido incluindo um novo ator, representando o sistema, e um número de dependências com outros atores do ambiente. Estas novas dependências assim como o refinamento dos requisitos devem ser realizados por meio de uma modelagem de metas. Ao final desta fase, as dependências representam os requisitos funcionais e não-funcionais do sistema.

3.4.3 Projeto Arquitetural

Para a escolha do estilo arquitetural, Tropos faz uso do *framework NFR* proposto por [10] e de um catálogo de correlação entre estilos arquiteturais organizacionais e requisitos não-funcionais, ou atributos de qualidade [31]. Definido o estilo arquitetural, este deve ser aplicado aos modelos gerados na fase de Requisitos Finais. Normalmente, nesta etapa, ocorre a inclusão de novos atores e dependências, assim como a decomposição de atores e dependências já definidos na fase anterior.

3.4.4 Projeto Detalhado

A fase de projeto detalhado serve para introduzir detalhes adicionais para cada componente da arquitetura do sistema, ou seja, esta fase tem como foco o desenvolvimento interno dos agentes. Metas, crenças e capacidades, assim como a comunicação dos agentes são especificados com detalhe nesta fase, logo os aspectos identificados nesta etapa irão depender diretamente de características da plataforma de implementação que será adotada para a construção do sistema.

3.4.5 Implementação

Esta fase deve estabelecer um mapeamento para a plataforma de implementação. Nesta etapa, a modelagem de atores corresponde à codificação dos agentes no sistema.

[2]Stakeholder, em português, é definido como parte interessada ou interveniente, refere-se a todos os envolvidos num processo, por exemplo, clientes, colaboradores, investidores, fornecedores, comunidade, etc.

3.5 MASUP

A construção de sistemas multiagentes também necessita seguir uma metodologia de desenvolvimento. O RUP atualmente é conhecido como um dos processos de desenvolvimento mais aceitos na indústria de software. Porém, o mesmo não pode ser utilizado para Sistemas Multiagentes, visto que é voltado totalmente para a orientação a objetos. Com base nesta necessidade, foi criado o Multi-agent Systems Unified Process (MASUP) [6].

O MASUP nada mais é do que uma extensão do RUP focada no desenvolvimento de Sistemas Multiagentes. Bastos e Ribeiro, em [6], salientam que o principal objetivo de tal metodologia é identificar sistematicamente a aplicabilidade de uma solução orientada a agentes durante a modelagem. As fases de Levantamento de Requisitos, Análise e Projeto, são usadas na metodologia MASUP, pois a especificação do sistema tem seu início no Levantamento de Requisitos e, além disso, as fases de Implementação, Teste e Implantação tratariam da especificação interna dos agentes, fugindo do escopo inicial desta metodologia.

As fases de Levantamento de Requisitos, Análise e Projetos são utilizadas no MASUP, uma vez que o início da especificação do sistema ocorre na fase de requisitos. Cabe também salientar, que as fases de implementação, teste e implantação não são abordadas pela metodologia porque estas tratariam da organização interna dos agentes, fugindo assim, do escopo inicial. A Figura 3.5 apresenta os modelos e artefatos do MASUP.

3.5.1 Levantamento de Requisitos

Os requisitos no MASUP são, assim como no RUP, capturados através dos casos de uso. Nesta fase, além dos casos de uso, também são produzidos outros artefatos, como a descrição dos casos de uso (formato textual), e diagramas de atividades de cada caso de uso. Para a etapa de levantamento de requisitos, encontramos as seguintes atividades: buscar de atores e casos de uso, priorizar casos de uso, detalhar os casos de uso e estruturar o diagrama de casos de uso. Apesar de o MASUP ser uma extensão ao RUP, a fase de levantamento de requisitos ocorre da mesma maneira nas duas metodologias.

3.5.2 Análise

De acordo com RUP, o propósito da fase de Análise é alcançar uma especificação mais precisa dos requisitos, produzindo assim, um modelo que representa a primeira visão do projeto. Enquanto o modelo de casos de uso representa a visão externa do sistema, o modelo de análise descreve a visão interna do mesmo. A fase de análise

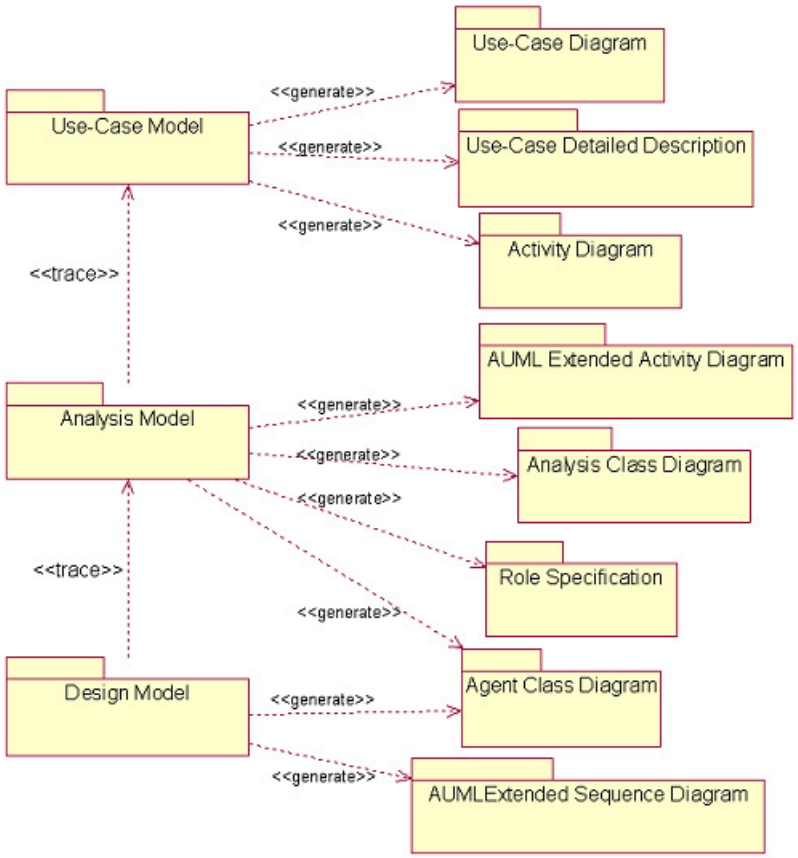


Figura 3.5: Modelos e Artefatos pertencentes ao MASUP - extraído de [73]

envolve a definição da estrutura do sistema, identificando as classes de análise que realizam os casos de uso.

Ao contrário da fase de levantamento de requisitos, nesta fase, o MASUP apresenta algumas diferenças relevantes em relação ao RUP. Segundo a metodologia MASUP, deve-se, por exemplo, definir os agentes que compõem uma sociedade juntamente com suas funções e, além disso, devem ser representadas as interações entre os agentes necessários para a resolução das atividades definidas nos casos de uso. As seguintes atividades fazem parte da etapa de análise do MASUP.

- Re-projeto do diagrama de atividades para modelar a solução orientada a agentes para os casos de uso selecionados;
- Identificação dos papéis necessários para a solução multiagentes baseada no diagrama de atividades gerado no passo anterior;
- Especificação de cada papel de agente definindo suas respectivas atribuições;
- Identificação dos agentes que devem executar papéis específicos;

- Definição dos relacionamentos entre os agentes que compõem a arquitetura da sociedade de agentes.

Para o reprojeto dos diagramas de atividades é conveniente construir os diagramas de classes de análises, seguindo as regras do RUP. Após esta construção, segue-se o fluxo de atividades e é feita uma análise verificando se uma solução orientada a agentes é aplicável. Não sendo aplicada a solução orientada a agentes, continua-se o processo normal do RUP. Caso contrário, segue-se o fluxo de atividades do modelo MASUP.

Os diagramas de atividades são remodelados após serem identificadas as atividades que são executadas por agentes, ou seja, caracterizadas por responsabilidades de tomada de decisão. Em seguida, são identificados os papéis que caracterizam o comportamento de cada agente. Os seguintes aspectos, segundo [6], são considerados na associação de um objeto de análise a um papel:

- Responsabilidade: a associação ocorre quando são necessárias capacidades de tomada de decisão e autonomia.

- Informação: os atributos mantidos pelo objeto definem o conhecimento do sistema em termos de informação. Sendo assim, a associação ocorre quando o objeto mantém atributos requisitados pela solução multiagentes.

- Comportamento: como no conhecimento da informação, a associação ocorre quando o comportamento do objeto pode ser total ou parcialmente relevante para a solução multiagentes.

É importante salientar que quando mais de um agente é necessário para executar uma atividade por meio de cooperação, uma coalizão é criada. Nos Diagramas de Atividades Estendidos *AUML*, os retângulos representando os objetos participantes de cada atividade que se transformam em papéis executados por agentes, recebem o estereótipo *«role»*.

Terminados os novos diagramas de atividades, é realizada a especificação dos papéis. Esta fase fornece o conhecimento necessário sobre os papéis usados na execução dos casos de uso especificados para o sistema multiagentes. Identifica-se a atribuição para cada papel. As tabelas geradas nesta fase apresentam as atribuições derivadas das atividades do caso de uso que solicitam a participação do papel, o caso de uso e a atividade em questão, e as restrições a serem observadas pelo papel para a sua execução.

Especificados os papéis, são identificados os agentes. No MASUP, [6] define um agente como uma agregação de papéis cujas atribuições são complementares. Por

atribuição complementar entende-se que o agente deve mudar seu papel quando assumir outra atribuição requisitada por uma atividade do caso de uso. Os agentes são representados por classes de agentes que especificam as atribuições, comportamento e arquitetura compartilhados por um conjunto deles. As informações necessárias para se definir uma classe de agente são: seu nome, seu número máximo de instâncias na sociedade, seus atributos, as interfaces de interação, seus papéis e suas atribuições. As interfaces de interação fornecem os atos de comunicação que o agente é capaz de reconhecer como válido para atender a requisição de outro agente. Elas são definidas apenas após a especificação dos cenários de interação nos diagramas de seqüência. A classe de agente é representada por um retângulo que contém, no mínimo, o nome da classe e o número de instâncias.

Especificados os agentes, é definida a sociedade de agentes. Nesta especificação, é necessário definir a relação hierárquica entre os agentes por meio do Diagrama de Classes de Agentes. As relações são identificadas dos Diagramas de Atividades Estendidos *AUML*, nos pontos em que os papéis são responsáveis em executar conjuntamente uma ou mais atividades, e representam canais de comunicação em que mensagens são trocadas. As relações são representadas por setas que conectam os emissores aos receptores, contendo, opcionalmente, os seus nomes. Elas também contêm, em cada extremidade, a multiplicidade e o papel do agente.

A relação hierárquica entre os agentes pode ser definida como: de autoridade e de comunicação. A primeira é dividida em duas outras, a permanente e a dependente de papel. Na permanente, o receptor deve necessariamente atender uma requisição do emissor. É representada por uma linha sólida com a ponta da seta em negrito. Na dependente de papel, o receptor responde a requisição apenas quando o emissor está executando um papel específico. É representada por uma linha sólida com uma metade de ponta de seta em negrito. Na relação de comunicação, o receptor decide atender ou não a requisição. É representada por uma linha sólida com uma seta.

3.5.3 Projeto

O propósito da fase de Projeto é adaptar os resultados encontrados na fase anterior com as limitações impostas pela implementação. Nesta fase o MASUP possui atividades que definem as interações entre os agentes e como eles vão interagir com o ambiente de implementação. Por não forçar o uso de uma plataforma específica de implementação, o MASUP usa a noção de serviços para mapear a plataforma de implementação, demonstrando assim, como os agentes irão interagir com a mesma. As seguintes atividades fazem parte da fase de Projeto:

- Especificação dos cenários de interação dos agentes;

- Complementação da especificação das classes de agentes com os atos de comunicação para a implementação das interações modeladas;
- Identificação da infra-estrutura de serviços envolvida nos cenários especificados pelas interações modeladas.

Na especificação dos cenários de interação dos agentes, é descrita a interação entre os agentes que executam, conjuntamente, uma atividade de um caso de uso. O Diagrama de Seqüência Estendido *AUML* descreve estas interações, acrescentando algumas propriedades novas, que são: o nome do agente, o número de instâncias dos agentes, os nomes das coalizões, os papéis dos agentes e as relações hierárquicas entre as instâncias dos mesmos. Este novo diagrama deve estar em conformidade com o Diagrama de Classes de Agentes.

O Diagrama de Seqüência Estendido *AUML* possui setas que representam os tipos de mensagens trocadas entre os agentes. As mensagens, neste trabalho, são expressas pelos atos de comunicação da ACL . Segundo Perotoni, em [52], uma mensagem enviada a um agente específico (*unicast*) é representada por uma linha cheia. Uma mensagem enviada a um conjunto específico de agentes (*multicast*) é representada por uma linha sólida em negrito. Quando todos os agentes de uma classe recebem uma mensagem (*broadcast*), esta é representada por uma linha tracejada em negrito. O tempo de vida de um agente é representado por uma linha tracejada e uma linha dupla define sua participação na interação. A definição de fluxos alternativos de mensagens é representada por um losango contendo uma letra "x".

Especificados os Diagramas de Seqüência Estendidos *AUML*, a especificação das classes de agentes precisa ser atualizada. Cada mensagem ACL envolvida na interação deve ser corretamente inserida na especificação das classes de agentes.

Atualizada a especificação das classes de agentes, a metodologia MASUP encerra o seu ciclo de execução. É importante ressaltar que a mesma não obriga o uso particular de qualquer plataforma de implementação.

3.5.4 Considerações sobre o MASUP

No que tange o assunto discutido neste trabalho, a identificação dos agentes no MASUP é baseada na escolha dos papéis que foram identificados na fase de análise ao serem encontradas atividades de tomada de decisão dentro dos diagramas de atividades derivados dos casos de uso, assim, não há um processo automatizado de identificação dos agentes a partir dos casos de usos. Uma decisão do analista baseada em sua experiência é a principal característica da identificação dos agentes nessa metodologia.

3.6 Considerações

Embora o conceito de papel de agente esteja presente em algumas das metodologias apresentadas neste capítulo, o processo de identificação de tais elementos bem como da criação dos agentes inteligentes ainda passam pela experiência do analista, sendo resultados da aplicação de seus conhecimentos técnicos sobre um conjunto inicial de requisitos. Esse processo de identificação de papéis se torna subjetivo uma vez que a experiência e o conhecimento das pessoas que estão modelagem o sistema orientado a agentes pode ser muito diversificado, resultando em diferentes análises para o mesmo cenários.

Diante do fato supra-citado, surge a motivação desta dissertação que se baseia na busca por um método capaz de realizar a identificação de papéis de agentes e que seja independente do conhecimento ou experiência das pessoas envolvidas na construção de um SMA. O capítulo a seguir traz os conceitos inerentes a Modelagem de Negócio a qual compõe um dos pilares desta pesquisa.

Capítulo 4

Modelagem de Negócios

"A ciência fez de nós Deuses antes mesmo de merecermos ser homens"
— Jean Rostand

Os Sistemas de Informação (SI) estão se tornando cada vez mais essenciais e mais presentes no cotidiano das pessoas, nos mais diferenciados contextos. Eles podem envolver desde sistemas triviais até sistemas que implementam algoritmos muito complexos. Uma questão de grande relevância no desenvolvimento de um software é garantir que ele represente, de forma real, uma solução para o problema específico.

Expressar requisitos é uma das partes críticas no desenvolvimento de um sistema e qualquer erro pode trazer como conseqüência vários problemas futuros. Sendo assim, diferentes tipos de projetos exigem diferentes técnicas para a organização dos requisitos [30].

Com o aumento da competitividade entre as empresas, torna-se cada vez mais fundamental a escolha por sistemas de informação que vão ao encontro das necessidades do negócio. Aspectos relacionados ao ambiente organizacional, no qual o software está inserido, são comumente desconsiderados ou avaliados de forma incompleta [60]. É extremamente importante estabelecer uma correspondência entre o ambiente organizacional e o sistema de software desenvolvido para ser executado nele. Isso implica na avaliação de objetivos e metas organizacionais e no apontamento das formas com as quais os sistemas de software podem satisfazê-los.

Ter uma boa gestão e domínio do negócio tornaram-se itens indispensáveis para as organizações se manterem competitivas [43]. Para que esses sistemas auxiliem uma organização é essencial que sejam bem especificados, de forma que seus requisitos estejam alinhados ao negócio, e que permitam aos funcionários executar as tarefas de forma adequada e eficiente [60]. Entretanto, especificar esses sistemas não é uma tarefa trivial. Muitas vezes, a falta de entendimento sobre o negócio impede que as necessidades existentes sejam percebidas durante a fase da elicitação

dos requisitos.

Nesse contexto, a Modelagem de Negócio surge como uma abstração da empresa e de suas atividades, com a finalidade de representar quais destas são executadas e como são articuladas para chegar aos objetivos daquela. Busca-se auxiliar as organizações, através de seus modelos, no processo de compreenderem melhor o seu próprio negócio.

4.1 Vantagens da Modelagem de Negócio

É de comum acordo entre todos os estudiosos que o sistema que resulta de um processo de desenvolvimento de software deve atender as necessidades do negócio, ou seja, que o sistema represente aquilo que o cliente está esperando. Porém, nem sempre essa idéia demonstra uma realidade sendo que, muitas vezes, o sistema resultante não está alinhado ao negócio da organização que o solicitou.

Um dos maiores desafios está em compreender o verdadeiro problema a ser resolvido e às vezes isso impede que as necessidades a serem atendidas sejam identificadas adequadamente. A dificuldade na comunicação é um dos principais fatores que acarretam em problemas, tanto por parte dos engenheiros como por parte da organização, que nem sempre possui uma visão clara das suas necessidades.

Em uma realidade na qual os negócios estão cada vez mais automatizados e o computador faz realmente parte das organizações, compreender o negócio e como ele funciona passa a ser algo essencial e, talvez, a chave para o sucesso [28]. Quando uma organização decide desenvolver o seu modelo de negócio, com o objetivo de aumentar a eficiência de seus serviços, são descobertas as principais informações que possibilitam a tomada de decisões.

A modelagem de negócio emerge com esta finalidade: garantir que o sistema resultante é realmente o sistema solicitado pelo cliente, no caso a organização, sendo que as informações presentes neste modelo ajudarão a considerar o contexto organizacional, de maneira a possibilitar a compreensão das necessidades de negócio que demandam esse sistema. Com isso, pode-se partir, então, para a definição dos requisitos que atenderão a essas necessidades.

4.2 Conceitos Chave da Modelagem de Negócios

São divididos em três partes e são descritos individualmente como Recurso do Negócio, Processos do Negócio e Regras, que são detalhas a seguir.

4.2.1 Recursos do Negócio

São os objetos utilizados ou produzidos pelo negócio, tais como materiais, informações e produtos. Os recursos são organizados em estruturas e têm relação uns com os outros. São manipulados (usados, consumidos, transformados, produzidos) através dos processos [20]. Usando como exemplo uma biblioteca, podem-se citar como recursos: obras (livros, teses, revistas) e fichas de identificações dos títulos, entre outros.

4.2.2 Processos do Negócio

Atividades conduzidas no negócio, durante as quais o estado dos recursos do negócio muda. Os processos descrevem como o trabalho é feito no negócio; mostram como o negócio é conduzido [20]. Eles são, desta forma, um conjunto de atividades de trabalho ordenado ao longo do tempo, que possui começo e fim bem definidos com entradas e saídas. São governado pelas regras. Exemplo (Biblioteca): Emprestar Obras, Reservar Títulos, Devolver Obras.

4.2.3 Regras de Negócio

Sentenças que definem ou restringem algum aspecto do negócio. Representam um conhecimento a respeito do negócio. As regras definem como o negócio deve ser conduzido (como os processos devem ser executados). Regras podem definir como os recursos devem ser estruturados e relacionados uns com os outros [20].

4.3 Abordagem da Rational - RUP

Na abordagem da Rational (RUP - Rational Unified Process) os autores acreditam que a notação padrão de modelagem visual, proposta pela UML, para a análise e projeto dos sistemas de software pode ser usada efetivamente para a criação de modelos. Os analistas de negócios podem usar, para documentar o processo de negócios, a mesma notação e ferramentas que os arquitetos e projetistas usam para documentar o sistema de software. Os autores acreditam, ainda, que falando a mesma língua os dois grupos poderão se comunicar melhor, permitindo que os sistemas de software realmente representem as necessidades do negócio.

Como, atualmente, os projetos estão sendo mais focados nos negócios, é essencial que o time de Tecnologia da Informação (TI) tenha como base descrições não ambíguas do funcionamento do negócio [28]. Sendo assim, foi definido um modelo que consiste em duas partes: o modelo de caso de uso do negócio e o modelo de objeto de negócio. Ambos podem ser criados usando UML.

Para auxiliar na manipulação desses modelos, a Rational propôs a utilização de uma de suas ferramentas, o Rational Rose, que possui mecanismos para diagramação tanto de modelos de sistemas quanto do modelo de negócio. Cada diagrama, proposto em sua abordagem, fornece uma visão diferente do negócio [28, 38].

Um **modelo de casos de uso de negócio** consiste em atores e casos de uso de negócio, onde os atores representam papéis externos para o negócio e os casos de uso representam os processos [28, 39].

- **Diagrama de Caso de Uso de Negócio -** os casos de uso de negócio descrevem o processo do negócio, e são documentados como uma seqüência de ações que fornecem valores observáveis para um ator de negócio. Os detalhes associados com este modelo são documentados em uma especificação de casos de uso de negócio, que inclui um texto e um ou mais diagramas de atividades UML e, possivelmente, outros diagramas de casos de uso de sistema. Descrevem o que fazer e não como o negócio pode ser resolvido.

- **Diagrama de Atividades -** representa a estrutura descrita textualmente na especificação dos casos de uso de negócio, ou seja, descreve o fluxo de trabalho executado em cada um deles.

Um **modelo de objetos do negócio** descreve como resolver o problema em questão. Serve como uma abstração de como os casos de uso de negócio serão executados em termos de interações de trabalhadores e entidades do negócio [28, 39].

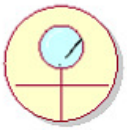
- **Diagrama de Classes de Negócio** - mostra as relações entre os trabalhadores e as entidades do negócio.

- **Diagrama de Seqüência do Negócio -** apresenta detalhes sobre a interação entre os trabalhadores e os atores do negócio. Também mostra como as entidades do negócio são acessadas durante a realização do caso de uso. Este diagrama pode fornecer, ainda, a interação de um ator do negócio com o negócio em si.

Da mesma maneira que existem diversas abordagens para a modelagem de negócio, também existem diversas notações para representar os modelos dos processos de negócio. A Tabela 4.1 apresenta os elementos do modelo de processos de negócio bem como as respectivas representações UML de cada um destes elementos (representações referentes à ferramenta de modelagem de sistema Rational Rose).

Para documentar um processo de negócio o modelo apresenta duas visões, uma externa e outra interna. A primeira é feita através da descrição dos casos de uso, que consiste no detalhamento passo a passo da interação entre os atores

Tabela 4.1: Elementos UML que compõem o modelo de processos de negócio

Elemento	Descrição	Representação no perfil de Modelagem de Negócios	Notação
Caso de Uso do Negócio	Representação de um processo de negócio que gera valor para algum cliente externo à organização.	Caso de uso estereotipado como <<business use case>>	
Ator do Negócio	Papel desempenhado em relação ao negócio por alguém ou alguma coisa no ambiente de negócio	Ator estereotipado como <<business actor>>	
Trabalhador do Negócio	Agente humano ou informático que atua internamente no negócio durante a realização de casos de uso, interagindo com outros agentes de negócio e/ou manipulando entidades de negócio.	Classe estereotipada como <<business worker>>	
Entidade do Negócio	Recurso físico, abstrato ou de informação, utilizado ou produzido nos processos de negócio.	Classe estereotipada como <<business entity>>	
Regra do Negócio	Regra que define ou restringe aspectos do negócio, com o objetivo de satisfazer a requisitos externos (leis, restrições impostas por outros negócios, etc) e de garantir que objetivos sejam alcançados de forma segura.	Não há nenhuma representação direta na UML para regras de negócio. As mesmas vêm descritas como restrições, anotações ou descrições textuais.	-

e a organização, durante a execução do processo em questão. Esse detalhamento pode ser feito por meio de descrições textuais, para processos mais simples, ou por diagramas de atividades, em notação UML A visão interna é modelada através da realização dos casos de uso, que detalha como os trabalhadores do negócio interagem entre si e como as entidades do negócio são manipuladas durante a realização do processo. A realização dos casos de uso é representada através dos diagramas de interação da UML.

4.4 Do modelo de negócio ao modelo de sistema

O processo de desenvolvimento proposto pelo RUP possui uma maneira concisa e direta para gerar requisitos que darão suporte as ferramentas de negócios e sistemas. Um bom entendimento acerca do negócio se faz importante para a construção dos sistemas corretos. Usando a modelagem de negócio, mais valor passa a ser agregado ao desenvolvimento de sistemas, uma vez que passam-se a utilizar de papéis de pessoas e responsabilidades, bem como definições das entidades manipuladas pelo negócio, formando assim uma base para a construção de sistemas. Segundo a Rational em [11], é deste ponto de vista mais interno sobre o negócio, capturado no modelo de análise do negócio, que encontramos uma ligação sobre como os modelos do sistema devem ser construídos.

Do ponto de vista de arquitetura de sistemas, modelar o negócio é um processo útil se a intenção é construir algum dos sistemas abaixo descritos:

- Sistemas customizados para uma ou mais companhias de um ramo específico, tais como bancos, companhias de seguro, lojas, etc;
- Famílias de aplicações para o mercado, tais como sistema de estoque, cobranças, controle de tráfego aéreo, etc.

4.4.1 Modelo do Negócio e Atores para o Sistema

Para identificar os atores do sistema de informação, deve-se iniciar a transformação do modelo de negócio para o modelo de sistemas utilizando os trabalhadores do negócio. Com esse objetivo, os passos abaixo devem ser seguidos:

- Decidir se o trabalhador do negócio será uma pessoa que irá utilizará o sistema de informação;
- Caso a resposta para o passo acima seja sim, deve-se identificar um ator do sistema para o trabalhador do negócio em questão, e criá-lo no modelo de sistema, aplicando o mesmo nome;
- Repetir os passos anteriores para todos os trabalhadores do negócio.

Para a identificação dos casos de uso do sistema, deve-se analisar cada realização dos casos de uso do negócio. Com esse objetivo, os passos a seguir devem ser seguidos:

- Identificar as seqüências de passos que são iniciadas pelos atores do sistema (identificados anteriormente).

- Criar um caso de uso do sistema para cada passo identificado na seqüência. O nome do caso de uso passa a ser o nome da operação, ou seja, o nome descrito no passo.

- Garantir que o diagrama de casos de uso atenda todos os critérios dos requisitos do sistema (provendo valores significativos para os atores e assim por diante).

Esses são os primeiros passos para partir da modelagem de negócio e chegar aos modelos do sistema. Desta forma, os atores e casos de uso do sistema identificados podem vir a ser modificados caso se façam necessárias mudanças nos requisitos do sistema. Um caso de uso do negócio pode ser suportado por apenas um caso de uso do sistema bem como dois ou mais, de acordo com os modelos de análise do negócio.

Para exemplificar o que foi apresentado anteriormente, um cenário do domínio bancário é apresentado, onde um cliente solicita um empréstimo para a instituição financeira, situação ilustrada pelo diagrama de caso de uso do negócio na Figura 4.1.

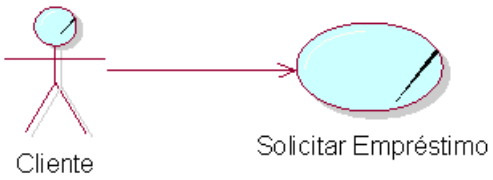


Figura 4.1: Diagrama de Caso de Uso do Negócio

Temos como um fluxo básico para a solicitação de empréstimo os seguintes passos: o cliente solicita ao atendente um pedido de empréstimo. O atendente, por sua vez, submete o pedido de empréstimo ao Sistema de Empréstimo. O pedido de empréstimo fica disponível no sistema para que o especialista de empréstimo faça uma análise do mesmo e tome a ação de aceitá-lo ou rejeitá-lo. Nesse contexto, têm-se dois passos: "Submeter Empréstimo"e "Aprovar Empréstimo"que são iniciados respectivamente pelo Atendente e Especialista de Empréstimo (conforme Figura 4.2).

Uma vez identificados os passos descritos acima, casos de uso do sistema são criados para cada passo que é iniciado por um trabalhador do negócio. Logo, temos os casos de uso "Aprovar Empréstimo"e "Submeter Pedido de Empréstimo", associados aos atores do sistema, respectivamente, Especialista de Empréstimo e Atendente. O diagrama de casos de uso do sistema para este cenário é ilustrado pela Figura 4.3.

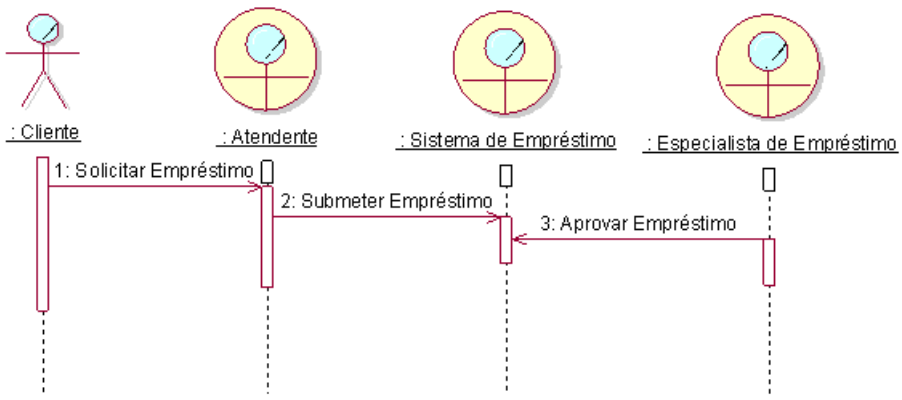


Figura 4.2: Diagrama de Sequência do Negócio

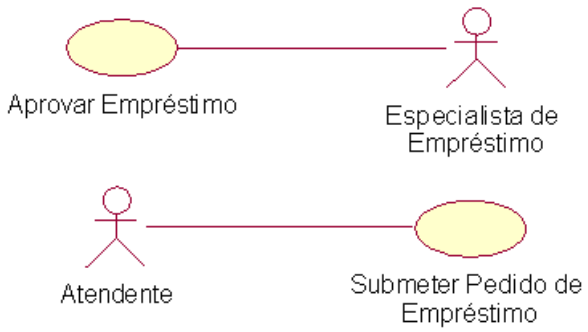


Figura 4.3: Diagrama de Caso de Uso do Sistema

4.4.2 Trabalhadores do Negócio automatizados

Se a intenção é construir um sistema que automatize completamente um conjunto de processos de negócio (como, por exemplo, um sistema de e-commerce), não será mais o trabalhador do negócio que irá interagir diretamente com ele. Ao contrário, é o ator do negócio que irá interagir diretamente com o sistema, agindo como ator de tal. Desta forma, a maneira como o negócio é executado passa a ser diferente quando um sistema com estas características está sendo desenvolvido, uma vez que as responsabilidades outrora atribuídas ao trabalhador do negócio passam a ser parte do contexto do ator do negócio (vide Figura 4.4).

Para exemplificar, supõe-se, do exemplo anterior, que a instituição financeira deseja desenvolver uma aplicação de comércio eletrônico para os seus clientes para facilitar a interação através da internet. Assim tem-se:

- As responsabilidades do atendente são transferidas para o cliente;
- Um ator do sistema que corresponde ao ator do negócio deverá ser criado;
- Os trabalhadores do negócio Atendente e Sistema de Empréstimo serão unidos em um só, dando origem ao Sistema de Empréstimo Aprimorado.

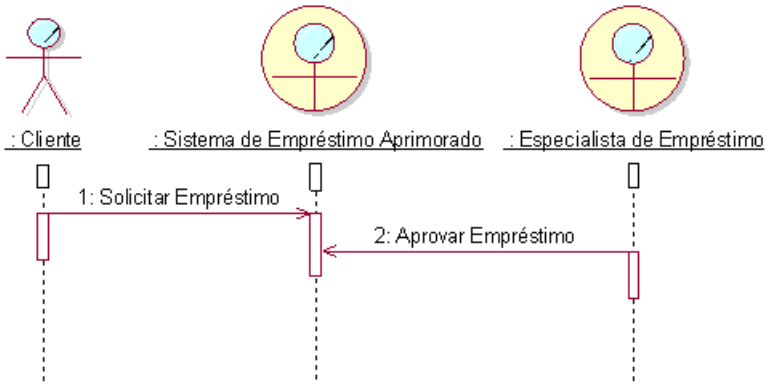


Figura 4.4: Diagrama de Seqüência do Negócio com Trabalhador Automatizado

- Modificar a realização do caso de uso de acordo com as mudanças realizadas nos passos anteriores;
- Identificar os novos casos de uso do sistema ou adaptar os já existentes, baseando-se nas modificações da realização do caso de uso do negócio.

A Figura 4.5 apresenta o diagrama de casos de uso para a nova abordagem.

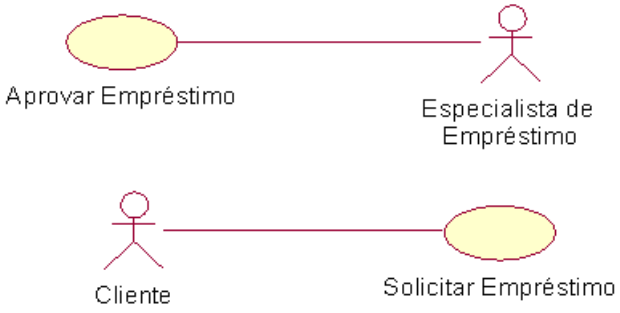


Figura 4.5: Diagrama de Caso de Uso do Sistema com Trabalhador Automatizado

4.4.3 Entidades do Negócio para Classes de Análise

Uma entidade do negócio a ser controlada por um sistema de informação terá uma entidade correspondente no modelo de análise do sistema. Em alguns casos, pode ser mais adequado criar entidades do sistema correspondendo aos atributos das entidades de negócio. Como muitos trabalhadores do negócio podem vir a acessar as entidades do negócio, conseqüentemente as entidades do sistema correspondentes podem participar de vários casos de uso do sistema de informação.

No exemplo em questão, no modelo de negócio são identificadas três entidades, são elas: Perfil do Cliente, Conta e Empréstimo (Figura 4.6).

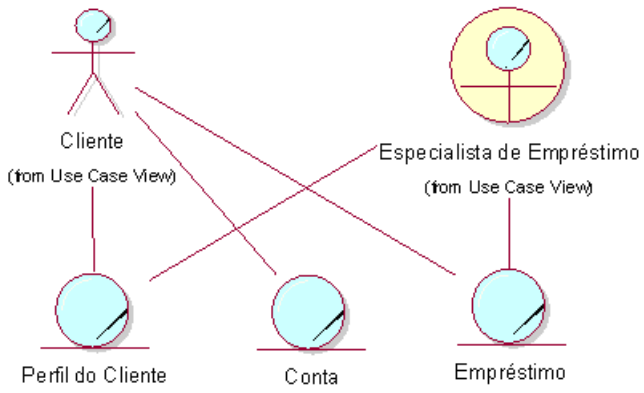


Figura 4.6: Diagrama de Entidades do Negócio

4.5 Considerações

Como visto neste capítulo, a modelagem de negócio assume um papel importante durante a processo de desenvolvimento de *software*. Essa importância se dá pelo fato de que nesta disciplina são identificados os processos de negócios de uma organização, os quais podem endereçar melhorias dentro da empresa quando estes vierem a ser traduzidos por um sistema de informação que os implementa.

É de comum conhecimento que a modelagem de negócio não introduz nenhum artefato que descreva aspectos de sistema, uma vez que é através dela que são identificados os pontos de melhoria que possam vir a ser automatizados por um ou mais sistemas (não existe uma relação em que um caso de uso do negócio é implementado por um caso de uso do sistema, essa relação é muito variável de acordo com a complexidade do processo de negócio a ser implementado).

A proposta deste trabalho visa introduzir o conceito de papel de agente dentro da modelagem de negócio através da aplicação dos conceitos do processo decisório (apresentados no capítulo à seguir) sobre as atividades do negócio, indicando assim, possíveis situações que possam ser automazidas com agentes caso o processo de negócio venha a ser implementado.

Capítulo 5

Processo Decisório

"Já que você tem que pensar de qualquer forma, pense grande"
— Donald Trump

O termo DECISÃO é uma das palavras mais pronunciadas e ouvidas; e a sua correta aplicação é uma das coisas mais almejadas. Segundo Pereira e Fonseca, em [50] a palavra decisão é formada pelo prefixo *de* (prefixo latino que tem aqui o significado de parar, extrair, interromper) que se antepõe à palavra *caedere* (que significa cindir, cortar). Tomada ao pé da letra, a palavra decisão significa "parar de cortar" ou "deixar fluir". Uma decisão precisa ser tomada sempre que existe um problema com mais de uma alternativa para a sua solução.

No dia-a-dia, praticamente em todos os instantes, as pessoas necessitam tomar decisões. Decisões que variam da simplicidade extrema até o mais alto nível de complexidade, mas que precisam, de qualquer forma, ser tomadas. Elas abrangem, ainda, diferentes níveis de responsabilidade, podendo afetar apenas o DECISOR, ou os seus subordinados, ou a organização como um todo, ou talvez a própria nação. Para qualquer situação, é perguntado: "Então, o que deve ser feito? Qual a sua decisão? Já identificou o problema? E quanto a possíveis soluções alternativas?" Estas questões norteiam a vida de todas as pessoas e, em particular, daquelas que intervêm em processos decisórios profissionalmente. Por ser algo tão quotidiano, supõe-se que a tomada de uma decisão seja algo totalmente compreendido e conhecido. Entretanto, tal não acontece. O que se observa é uma quase ausência de metodologia para orientar e ou apoiar o PROCESSO DECISÓRIO, no sentido de torná-lo uma atividade estruturada. Na primordial questão sobre o que significa o termo decisão, uma possível resposta seria considerá-lo como um processo complexo e abrangente que se inicia com a percepção da necessidade de uma mudança; tem sua continuação com a escolha de um curso de ação entre os vários viáveis e que se finaliza com a implantação deste.

Como em todas as situações nas quais a tomada de uma decisão é necessária,

esta necessidade nem sempre se apresenta de forma explícita e normalmente envolve PROBLEMAS específicos para cada situação. Não existem conhecimentos teóricos disponíveis nem informações suficientes para sua solução. Isto obriga o tomador de decisão a ser criativo, original e racional, valendo-se, para sua análise, dos acontecimentos passados e do conhecimento presente, a fim de prever eventuais ocorrências negativas e precaver-se para o futuro.

Neste capítulo serão apresentados conceitos referentes ao processo decisório, iniciando por uma introdução aos sistemas de apoio à decisão e suas características, e chegando até o processo de tomada de decisão propriamente dito. Neste final constarão os estudos pertinentes para a formulação da proposta do presente trabalho.

5.1 Sistemas de Apoio à Decisão

A maior utilidade de Sistemas de Apoio à Decisão (SADE) (ou também conhecidos como Sistemas de Suporte à Decisão, SSD) está no suporte a problemas não-estruturados e semi-estruturados. Se por um lado soluções estruturadas praticamente não necessitam de meios de suporte à decisão, pois o caminho elas já é conhecido antecipadamente, problemas geográficos costumam exigir tecnologias de suporte, porquanto apresentam pouco ou nenhum grau de estruturação. A estrutura de um problema se refere ao nível de conhecimento que existe sobre as causas, as conseqüências e o processo de solução.

Tipicamente, um problema é não-estruturado quando ocorre pela primeira vez. Por exemplo: chuva torrencial provoca alagamento num ponto da cidade nunca atingido antes, ou barranco desliza sobre uma estrada até o momento intacta. Problemas semi-estruturados são os que já sofreram alguma análise capaz de conduzir a conclusões parciais. Por exemplo: modelos de previsão de chuva. Sua existência decorre de décadas de pesquisa e desenvolvimento tecnológico, na tentativa de se compreender a estrutura do fenômeno e predizer seu comportamento.

Já os problemas estruturados normalmente decorrem de processos criados pelo Homem, em oposição aos problemas não-estruturados, que decorrem de processos do meio ambiente natural.

Segundo Sprague e Carlson, em [56], um sistema de suporte a decisão deve compreender um conjunto de seis objetivos, tais quais estão listados abaixo:

1. Apoiar os decisores em decisões difíceis, pouco definidas, tanto não-estruturadas como estruturadas;

2. Auxiliar a tomada de decisão em todos os níveis da organização e integrar os mesmos quando for apropriado. De acordo com o já bem conhecido *framework* proposto por Anthony em [3], Sprague classifica as decisões como:

- Planejamento estratégico - decisões relacionadas à aplicação de políticas, escolha de objetivos e seleção de recursos,
- Controle gerencial - decisões relacionadas à garantia da efetividade na aquisição e uso dos recursos,
- Controle operacional - decisões relacionadas à garantia da efetividade na execução das operações;

3. Auxiliar a comunicação entre os decisores e as tomadas de decisões interdependentes. Sprague identifica a classificação dos tipos de tomada de decisão sob o ponto de vista da comunicação dos decisores apresentada pelos autores Hackatorn e Keen, [26]: independente, seqüencial interdependente e agregado interdependente. Tais classificações serão apresentadas mais adiante no texto.

4. Auxiliar todas as fases do processo de tomada de decisão e facilitar a interação entre as fases. De acordo com o Simon *apud* Sprague ([56]), os estágios de uma tomada de decisão (também considerada abordagem racional do processo decisório) são:

   - Análise e Identificação da situação - a situação do ambiente onde o problema ou a oportunidade estão inseridos deve ser claramente identificada através do levantamento de informações para que se possa chegar a uma decisão segura e precisa;
   - Desenvolvimento das alternativas - em função dos levantamentos das informações, ou seja, da coleta de dados, pode-se chegar a possíveis alternativas para a resolução do problema proposto;
   - Comparação entre as alternativas - levantamento das vantagens e desvantagens de cada alternativa;
   - Classificação dos riscos de cada alternativa - as decisões sempre envolvem riscos, seja em grau quase nulo, em alto grau ou em um grau intermediário;
   - Escolha da melhor alternativa - com o conhecimento das vantagens, desvantagens e riscos, o decisor é capaz de identificar a melhor alternativa para solucionar o seu problema;
   - Execução e avaliação - a alternativa escolhida fornecerá resultados que deverão ser comparados e avaliados com as previsões anteriores.

5. Suportar certa variedade de processos de tomada de decisão sem, entretanto, estar dependente de nenhum deles. Em outras palavras, trabalhar com processos independentes e que estejam sob total controle do usuário.

6. Ser de fácil manipulação e dar suporte a modificações em tempo de mudanças no ambiente, nas tarefas e no próprio usuário.

Hampton [27] coloca que existe uma grande diferença entre a abordagem racional e idealizada da decisão, e a realidade organizacional.Porém, o autor afirma que fazer uma abordagem idealizada e totalmente racional da decisão e, a seguir, uma comparação entre o ideal e o que realmente ocorre é uma boa maneira para entender o processo decisório. A Figura 5.1 mostra o processo decisório ideal, de acordo com o autor:

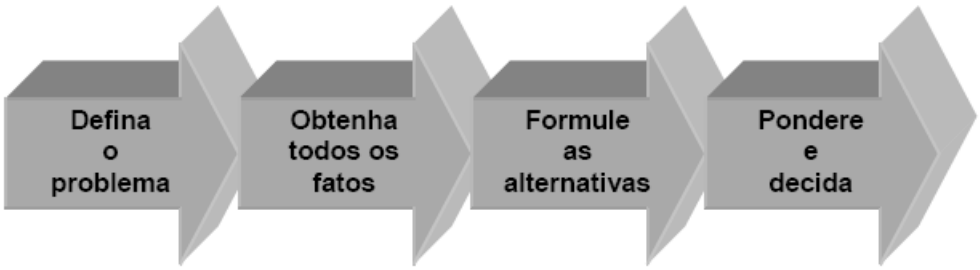


Figura 5.1: O processo decisório ideal - adaptado de [27]

As etapas, segundo Hampton, estão descritas a seguir:

- **Definir o problema -** o administrador deve investigar o problema a fundo, a fim de defini-lo corretamente. O autor exemplifica essa etapa com o trabalho de um médico: o doutor não pode aceitar passivamente as queixas do paciente e considerar apenas as manifestações superficiais da doença. O médico deve usar os sintomas apresentados para desencadear uma busca mais minuciosa e diagnosticar o problema. Quando os sintomas são erroneamente definidos como problema, ou quando se define o problema de maneira que o pensamento leve a uma solução preconcebida, as chances são poucas para se visualizar uma ampla gama de possíveis soluções. O autor afirma que existe uma tendência em se definir os problemas em termos prejudiciais, o que torna as possibilidades de solução bem menores.

- **Obter todos os fatos -** a primeira coisa a dizer sobre esta fase da tomada de decisão é que nem sempre o administrador obterá todos os fatos. Para ser um administrador eficaz, é necessário aprender a tolerar a ambigüidade, a informação incompleta, a incerteza e ainda decidir. Nessa tarefa, o que pode ajudar é a obtenção seletiva de dados críticos que conduzem ao centro do problema. Esta busca por novos fatos pode ser acelerada ou encurtada, de acordo com a urgência da questão. A busca por informações chega a um ponto em que não se consegue mais esclarecimentos para o problema, momento em que a informação pode tomar mais tempo e custar mais do que o valor em si.

- **Formular alternativas -** muitas pessoas aceitam a primeira impressão de um problema como sendo a definição adequada para ele. Partindo disso, formulam apenas uma ou poucas soluções alternativas, muitas vezes forçadas pela pressão da urgência.

- **Ponderar e escolher -** de acordo com a versão ideal da tomada de decisão racional na organização, os objetivos, estratégias, políticas, procedimentos, orçamentos e cronogramas fornecem o contexto para a mesma. Porém, a realidade é outra, e a escolha pode ser feita sob condições incertas. Outro problema é a possibilidade de ocorrerem conseqüências adversas e, neste caso, o processo de tomada de decisão deve esforçar-se para antecipá-las e calculá-las.

5.1.1 A evolução dos Sistemas de Apoio à Decisão

Sprague e Watson, em [55] afirmam que, no início da década de 70, várias empresas e vários grupos de pesquisas começaram a estudar e desenvolver Sistemas de Apoio à Decisão, os quais passaram a ser caracterizados como sistemas computacionais interativos que auxiliavam no processo decisório de problemas considerados não-estruturados. Porém, segundo os autores, na década seguinte, vários pesquisadores e desenvolvedores de sistemas ampliaram essa definição dos SADE, tratando de incluir quaisquer sistemas capazes de dar alguma contribuição ao processo decisório, desde que possuam as seguintes características:

- serem voltados para problemas menos estruturados e menos especificados com os quais os gerentes deparam;

- combinem o uso de modelos ou técnicas analíticas a funções tradicionais de acesso e recuperação de informações;

- concentrem-se especificamente em recursos que facilitem seu uso para pessoal não-especializado em computação e

- enfatizem a flexibilidade e a adaptabilidade de acomodar mudanças no ambiente e na abordagem ao processo decisório.

Segundo Power [53], o conceito de suporte computacional à decisão surgiu com a evolução de duas áreas de pesquisa: os estudos teóricos sobre o Processo de Tomada de Decisão Organizacional, feitos no Carnegie Institute of Technology, durante as décadas de 50 e 60, e os trabalhos realizados com Sistemas Computacionais Interativos no Massachusetts Institute of Technology, nos anos 60.

5.2 Processo de Tomada de Decisão

Na década de 70, com a ascensão dos computadores, muitas pesquisas foram realizadas acerca de Sistemas de Apoio a Decisão e das características que estes abrangem. Dentre as características que envolvem os SSDs, grande destaque tiveram os estudos referentes aos tipos de decisões. Simon, em [64], deu início à discussão referente aos processos de tomada de decisão ao apresentar sua classificação baseada na estrutura do processo. Anos mais tarde, outros autores também apresentaram suas propostas e classificações acerca desse assunto. Das propostas aceitas na época, podemos destacar três tipos: estrutura da tomada de decisão, interdependências das atividades do processo de tomada de decisão e atividades de administração. Na Figura 5.2 encontramos um arcabouço para SADEs de três dimensões contendo as três classificações apresentadas.

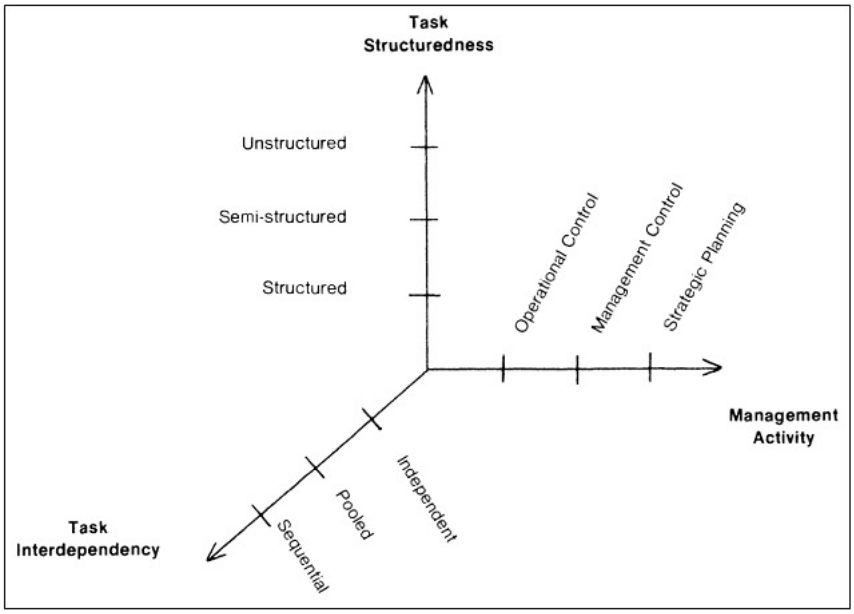


Figura 5.2: Framework SSD - adapatado [26]

5.2.1 Tipos de problema

- **Problemas Estruturados -** de acordo com Shimizu [62], um problema é considerado estruturado ou bem definido quando sua definição e fases de operação para chegar aos resultados desejados estão claras e sua execução repetida é sempre possível. Também se entende por problema bem-estruturado aquele em que o decisor consegue facilmente identificar uma estratégia de ação. Temos como exemplo deste tipo: folha de pagamento, lançamento contábil e operação de processamento de dados em geral;

- **Problemas Semi-Estruturados -** os problemas semi-estruturados são aqueles com operações bem conhecidas, mas que contêm algum fator ou critério variável que pode influir no resultado, tais como: problema de previsão de vendas ou problema de compras. Outra definição encontrada para problemas semi-estruturados argumenta ser este o tipo onde existe uma estratégia de ação que pode ser encontrada [62];

- **Problemas não-estruturados -** aqui, tanto os cenários como o critério de decisão não estão fixados ou conhecidos *a priori*. Um exemplo de problema não-estruturado é a operação de escolha da capa de uma revista semanal, para a qual diversas alternativas estão previstas, mas todas podem ser substituídas, na última hora, se algum fato importante ocorrer. Segundo Simon [64], nos problemas não-estruturados, embora haja uma percepção difusa de que algo está errado, não se consegue num primeiro momento identificar e definir a natureza deste diferencial. A estratégia de ação só será encontrada após trabalho de identificação, definição e percepção da inter-relação entre os fatores que possam vir a influenciar o problema.

Por sua vez, autores como Barros [4] apresentam uma visão mais abrangente para este nível de classificação de problemas ou situações que demandem ação. Para Barros, um problema é tão mais estruturado quanto mais intimamente o processo de sua representação puder ser repetido para outras situações semelhantes. Ao contrário, quanto maior o nível de incerteza e subjetividade envolvida na situação que demande ação, menos estruturado ele será. Vale lembrar também que a complexidade de um problema não está relacionada com a sua estruturação, uma vez que problemas complexos podem ser decompostos.

5.2.2 Níveis de Decisão

Os níveis de decisão organizacional obedecem à hierarquia padrão existente na maioria das organizações. Esta divisão em níveis pode ser chamada de pirâmide organizacional, onde os níveis são conhecidos como estratégico, tático ou gerencial e operacional (como ilustrado na Figura 5.3). O tipo de decisão que é tomada em cada nível requer diferente grau de agregação da informação. E os diferentes níveis de decisão requerem diferentes informações em seus diversos tipos de produtos, tais como diagramas, gráficos, relatórios etc.

No nível estratégico, as decisões ocorrem no alto escalão (ou alta administração) da organização e geram atos cujo efeito é duradouro e mais difícil de inverter. Emanam do planejamento em longo prazo da organização, conhecido como planejamento estratégico, tais como construção de uma nova fábrica, nova linha

Figura 5.3: Níveis de Decisão

de produção, novos mercados, novos produtos, novos serviços privados ou públicos. Esse nível de influência considera a estrutura organizacional de toda a organização e a melhor interação desta com o ambiente. Nesse caso, o nível da informação é macro, contemplando a organização em sua totalidade, ou seja, relacionando-a com meio ambiente interno e externo.

No nível tático ou gerencial, as decisões táticas ocorrem nos escalões intermediários (ou corpo gestor) e geram atos de efeito com prazo mais curto, porém de menos impacto no funcionamento estratégico da organização. Essas decisões emanam do controle tático e do planejamento gerencial da organização.

O nível tático de influência considera determinado conjunto de aspectos homogêneos da estrutura organizacional da organização. Nesse caso, o nível da informação é em grupos (agrupada ou sintetizada), contemplando a junção de determinadas informações de uma unidade departamental, de um negócio ou atividade da organização.

No nível operacional, as decisões operacionais estão ligadas ao controle e às atividades operacionais da organização. Essas decisões visam alcançar os padrões de funcionamento preestabelecidos, com controles do detalhe ou do planejamento operacional. Criam condições para a adequada realização de trabalhos diários da organização, onde o nível operacional de influência considera uma parte bem específica da estrutura organizacional da organização. Nesse caso, o nível da informação está detalhado (informação analítica, singular), contemplando pormenores específicos de um dado, de uma tarefa ou atividade da organização.

5.2.3 Interdependência entre Tarefas

Hackthorn e Keen, em [26], adicionaram a dimensão Interdependência das Tarefas ao framework ilustrado na Figura 5.2, baseados na teoria proposta por Thompson em 1977 sobre a relação das tarefas dentro do ambiente administrativo, mais precisamente relacionadas à tomada de decisão. Esta dimensão refere-se à interdependência das tarefas dentro do processo decisório, caracterizando assim a comunicação entre os responsáveis pela tomada de decisão.

- **Independente -** o responsável pela tomada de decisão tem completa responsabilidade e autoridade para realizar a tomada de decisão.
- **Seqüencial interdependente -** um tomador de decisão faz parte do processo decisório onde a decisão é tomada e passada adiante para outra pessoa.
- **Agregado interdependente -** a decisão deve resultar de uma negociação e da interação entre os responsáveis pela tomada de decisão.

5.3 Considerações

Este capítulo constitui o último pilar da base teórica desta dissertação. Nele encontramos conceitos pertinentes que serão utilizados para a formalização do método de identificação de papéis de agentes.

Dentro do processo decisório, encontramos as três definições inerentes ao processo de tomada de decisão que compõem o *framework* que teve origem nos estudos de Simon. Deste framework, serão utilizadas características referentes aos tipos de problemas e interdependência entre tarefas, uma vez que, para a proposta deste trabalho, o nível em que a decisão está sendo tomada não interfere no processo de identificação de um papel de agente.

Capítulo 6

Incluindo o Modelo do Negócio ao MASUP para a especificação de papéis de agentes

"Mestre não é quem sempre ensina, mas quem de repente aprende"
— João Guimarães Rosa

Pesquisadores e engenheiros de software estão se esforçando numa busca contínua por ferramentas e técnicas para gerenciar a complexidade inerente aos sistemas computacionais, especialmente quando estes tratam dos conceitos relacionados à sistemas multiagentes [70]. Deste esforço surgiram várias técnicas para construção de SMAs, que vão desde propostas de linguagens para especificação e construção de tais sistemas até metodologias para análise e projeto dos mesmos.

De acordo com Wooldridge [71], tais metodologias podem ser divididas em dois grupos:

- Aquelas que têm sua base no desenvolvimento orientado a objetos, nas metodologias para desenvolvimento orientado a objetos ou na adaptação destas metodologias existentes para suprir as necessidades da engenharia de software para sistemas multiagentes;
- Aquelas que derivam da adaptação da engenharia do conhecimento ou outras técnicas existentes.

Traçando uma relação de algumas das metodologias existentes com o processo unificado de software, percebe-se que, nas metodologias orientadas a agentes, o processo de desenvolvimento dos sistemas está compreendido basicamente entre as etapas de requisitos e implementação. Assim sendo, uma lacuna encontrada nas

metodologias orientadas a agentes se refere à identificação de papéis de agentes durante as etapas de análise e projeto. Nenhuma delas propõe um método sistemático para a derivação de papéis de agentes por meio dos artefatos de entrada das mesmas, como por exemplo, as especificações de requisitos. Tal processo de identificação acaba sendo uma análise dos diagramas e modelos do sistema e a decisão do que irá se tornar um agente, por parte do analista de sistemas, acaba sendo tomada com base no seu conhecimento ou muitas vezes apenas por conveniência.

Dentro do contexto acima explicitado, as seções subseqüentes deste capítulo introduzem o modelo proposto nesta dissertação que busca solucionar o problema descrito.

6.1 MASUP Estendido

Neste trabalho está sendo proposto uma extensão a metodologia para sistemas multiagentes entitulada MASUP, resultado da inclusão da disciplina de modelagem de negócios no processo de desenvolvimento de sistemas multiagentes. Tal extensão tem como objetivo preencher a lacuna existente hoje nas metodologias voltadas para o paradigma orientado a agentes que diz respeito à identificação de papéis de agentes.

Nas metodologias existentes, por não tratarem diretamente o assunto, durante o processo de modelagem e desenvolvimento de um sistema multiagentes, a identificação de papéis de agentes (ou até mesmo de agentes) que ocorre de maneira subjetiva. Essa atividade envolve muito mais a experiência por parte do analista, muitas vezes se baseando no seu conhecimento técnico e/ou na conveniência, demonstrando que o entendimento do problema e o resultado final esperado pode variar consideravelmente.

Pode-se destacar como motivações deste trabalho o fato de que, com a inclusão da Modelagem de Negócio no MASUP, pretende-se fazer com que essa metodologia passe a trabalhar com soluções que outras não tratam. Como conseqüência desta inclusão e com o processo de identificação de papéis de agentes a modelagem de negócio, almeja-se encontrar papéis de agentes para todo o negócio da organização, ou seja, quando o desenvolvimento de sistemas multiagentes se inicia na etapa de requisitos, o papéis de agentes identificados são específicos para o sistema em questão. Ao se modelar o negócio e realizar a identificação dos papéis sobre os modelos gerados, os papéis de agentes encontrados são referentes a todos os processos de negócio que foram modelados, podendo estes papéis darem suporte a um ou mais sistemas a serem implementados.

Outra motivação deste trabalho é a inexistência de uma metodologia para sistemas multiagentes que faça uso da modelagem de negócios dentro do seu processo.

Desta forma, pretende-se com este trabalho, incorporar a modelagem de negócios ao MASUP para ser possível sua extensão para a identificação de papéis. Vale lembrar que o escopo do que está sendo proposto nesta dissertação trata apenas do processo de identificação de papéis de agentes por meio da modelagem de negócios, e que as demais atividades inerentes a especificação, construção e testes de sistemas multiagentes não são aqui debatidas.

Na Figura 6.1 é apresentada a proposta desta dissertação na forma de um diagrama de pacotes UML. A configuração atual do MASUP compreende os pacotes Modelo de Requisitos, de Análise e de Projeto, conforme apresentado na Figura 3.5. O pacote Modelo de Negócio e os diagramas gerados por este, representam a extensão proposta neste trabalho. Na seqüência, serão apresentadas as características de cada novo pacote adicionado à metodologia.

6.2 *Workflow* de Negócio

Um dos grandes problemas encontrados com a introdução da modelagem de negócio no processo de desenvolvimento de software é que os engenheiros de sistemas e analistas de negócios não se comunicam de maneira correta e eficiente uns com os outros. Isso faz com que os artefatos de saída da modelagem de negócios não sejam utilizados adequadamente como entrada para o desenvolvimento de sistemas, e vice-versa. O *Rational Unified Process*, RUP, provê uma linguagem comum a ambas partes, bem como processos que descrevem a criação e a manutenção direta da rastreabilidade entre o modelo de negócio e os modelos do sistema [39].

Essa disciplina permite que os processos da organização sejam documentados inicialmente na forma de casos de uso, propiciando um entendimento comum dos usuários, clientes e desenvolvedores sobre os seus processos, bem como os responsáveis por cada um destes. Além disto, permite um melhor entendimento dos problemas da organização, permitindo identificar oportunidades de melhorias do processo da organização. De acordo com o RUP, as finalidades da modelagem de negócios são:

- Entender a estrutura e a dinâmica da organização na qual um sistema deve ser implantado (a organização-alvo).
- Entender os problemas atuais da organização-alvo e identificar as possibilidades de melhoria.
- Assegurar que os clientes, usuários e desenvolvedores tenham um entendimento comum da organização-alvo.
- Derivar os requisitos de sistema necessários para sustentar a organização-alvo.

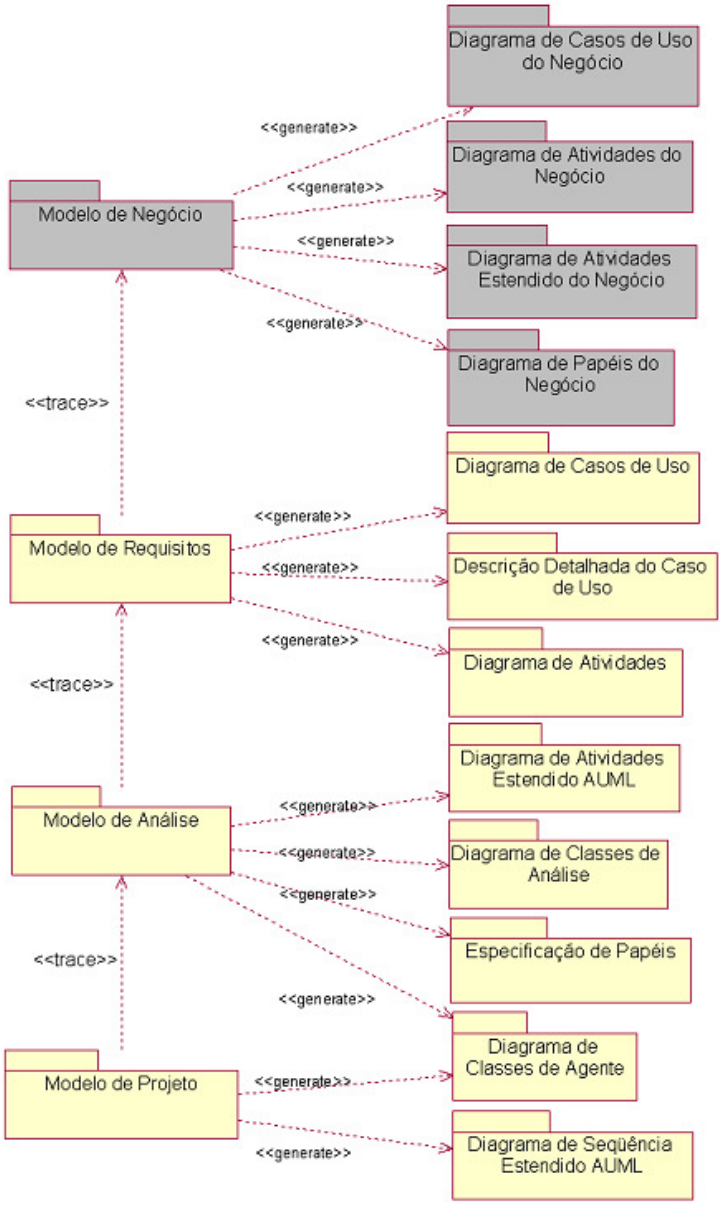


Figura 6.1: Modelos e Artefatos pertencentes ao MASUP estendido

Para atingir essas metas, a disciplina de modelagem de negócio descreve como desenvolver uma visão da nova organização-alvo e, com base nesta visão, definir os processos, os papéis e as responsabilidades dessa organização em um modelo de casos de uso de negócios e em um modelo de objetos de negócios. Complementando esses modelos, são desenvolvidos os seguintes artefatos: especificação suplementar de negócios e o glossário.

Trançando um relacionamento entre a disciplina de modelagem de negócio com outras disciplinas do RUP, temos os seguintes levantamentos:

- A disciplina Requisitos utiliza modelos de negócios como um importante subsídio para entender os requisitos do sistema.

- A disciplina Análise e Design utiliza entidades de negócios como subsídio para identificar classes de entidade no modelo de design.
- A disciplina Ambiente desenvolve e mantém artefatos de suporte, como o Guia de Modelagem de Negócios.

No decorrer deste capítulo, segue-se o mesmo padrão utilizado para introduzir o MASUP. Portanto, nas seções subseqüentes serão explicadas as características dos elementos introduzidos pela modelagem de negócio, bem como a maneira como são identificados os papéis e a(s) mudança(s) sofrida(s) pela atual versão da metodologia em questão.

6.2.1 Exemplo Ilustrativo

Esta seção descreve o exemplo utilizado para ilustrar a extensão que está sendo proposta neste capítulo. O exemplo mostra como se dá um processo de solicitação de crédito rural em uma instituição financeira nacional. O funcionamento do processo de solicitação de crédito será explicado a seguir.

O processo tem início quando o Cliente solicita ao seu Gerente de Conta um pedido de crédito rural. O Gerente por sua vez aceita o pedido do Cliente e percebendo que os dados fornecidos são insuficientes, pode solicitar uma análise técnica para um Analista Financeiro. O Analista Financeiro complementa o pedido do cliente com informações necessárias para a decisão do Gerente de Conta. Ao realizar uma análise sobre os dados fornecidos pelo Cliente (ou dados complementares fornecidos pelo Analista Financeiro), o Gerente da Conta pode reprovar o pedido, encerrando assim o processo de solicitação de crédito; ou pode aprovar diretamente o pedido realizado pelo Cliente, desde que este possua alçada suficiente para tal ação. Caso este Gerente de Conta não possua alçada, a responsabilidade pela aprovação do crédito é repassada para o Gerente da Agência, que precisa realizar a homologação do resultado da solicitação.

O Gerente da Agência deve ou não homologar a decisão tomada pelo Gerente de Conta em relação ao pedido, fazendo com que o fluxo do processo de solicitação tenha seu término nesta etapa, independente da decisão tomada pelo Gerente da Agência.

6.2.2 Diagrama de Casos de Uso do Negócio

Este é o principal diagrama para representar de forma gráfica os processos de negócio. No diagrama de casos de uso são representados somente os atores do negócio. Devemos pensar no diagrama de casos de uso como a representação das entidades

externas ao negócio e as iterações com este. Assim, como trabalhadores do negócio são internos ao negócio, eles não são representados no modelo de casos de uso.

Embora possuam definições similares, de acordo com Arthur English [19], existem três pontos importantes que diferenciam um caso de uso do negócio de um caso de uso de sistema:

- **Escopo -** Casos de uso do negócio são focados nas operações. Esses representam um fluxo de trabalho específico do negócio que busca alcançar um objetivo. Processos de negócio podem envolver tanto operações manuais ou automatizadas e podem durar um período de tempo extenso. Casos de uso de sistema são focados no sistema propriamente dito a ser implementado. O fluxo de eventos que descrevem uma especificação de caso de uso do sistema deve ser detalhado suficientemente para ser usado como ponto de partida para a construção de cenários de teste.

- **Caixa-branca *versus* Caixa-preta -** Casos de uso do negócio são freqüentemente escritos na forma de caixa-branca (*whitebox*). Eles descrevem as interações entre as pessoas e departamentos na organização que está sendo modelada. Casos de uso de negócio descrevem a forma como a organização trabalha atualmente e estes podem ser modificados para que as necessidades da organização seja satisfeita com o novo modelo de negócio. Algumas perguntas do tipo podem ser feitas durante esse processo de amadurecimento do modelo de negócio: Quais novos papéis e departamentos podem ser criados para agregar maior valor ou eliminar problemas nos processos? Que papeis e departamentos podem ser eliminados? Já os casos de uso de sistemas são escritos na forma de caixa-preta (*blackbox*). Estes por sua vez descrevem como atores externos ao sistema interagem com o mesmo. Os casos de uso do sistema elaboram os requisitos e possuem como principal propósito a documentação dos requisitos sob a perspectiva do cliente e não como especificar como satisfazer tais requisitos.

- **Trabalhadores do Negócio -** Num diagrama de casos de uso de sistema, apenas atores interagem com os casos de uso, enquanto no diagrama de casos de uso de negócio, podemos encontrar tanto atores do negócio como trabalhadores do negócio interagindo com os casos de uso de negócio. Um ator do negócio é alguém fora do negócio e pode ser um papel ou outra entidade organizacional, por exemplo. Já o trabalhador do negócio é alguém ou algum departamento interno ao negócio. Quando a realização de um caso de uso do negócio é criada e diagramas de seqüencia e/ou comunicação são desenvolvidos para ilustrar como os atores, trabalhadores e entidades do negócio colaboram

entre si para a execução do caso de uso, os trabalhadores do negócio são transportados do modelo de casos de uso para o modelo de análise do negócio, bem como os trabalhadores adicionais que se fazem necessários para prover as funcionalidades do caso de uso do negócio.

A Figura 6.2 apresenta um diagrama de casos de uso do negócio onde pode-se notar a presença de um ator do negócio, o Cliente, interagindo com dois processos de negócio definidos como: Emitir Pedido de Crédito Rural e Solicitar Empréstimo Financeiro.

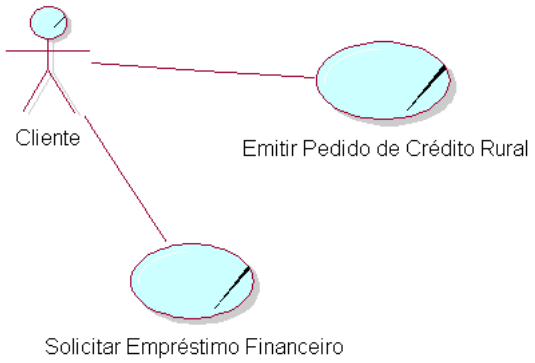


Figura 6.2: Diagrama de Casos de Uso do Negócio

Assim como o diagrama de casos de uso do sistema é utilizado na etapa de requisitos do MASUP, o diagrama de caso de usos do negócio também é utilizado na etapa de modelagem de negócios, entretanto, esse diagrama não sofre alterações nessa proposta, sendo assim, segue as regras e especificações da linguagem UML. Para a continuidade deste capítulo, as seções subseqüentes utilizarão como base para a explicação da proposta o caso de uso do negócio "Emitir Pedido de Crédito Rural", cuja descrição é apresentada na Figura 6.3.

6.2.3 Diagrama de Atividades do Negócio

O Diagrama de Atividades do Negócio é um diagrama comportamental que mostra um fluxo seqüencial de atividades. Esse diagrama é tipicamente usado para descrever as atividades realizadas em uma operação, tendo o objetivo de mostrar como um determinado processo é executado, mostrando o seu fluxo de trabalho. A representação das atividades também pode ser vista como um relacionamento entre os conceitos do modelo de negócio, o que faz com que informações importantes para a organização estejam representadas junto com as atividades. A Figura 6.4 ilustra na forma de diagrama de atividades do negócio o caso de uso "Emitir Pedido de Crédito Rural", apresentado na Figura 6.2.

Identificação: UCN1
Caso de uso: Emitir Pedido de Crédito Rural
Atores: Cliente
Stakeholders e Interesses:
Cliente: ter seu pedido de empréstimo aprovado
Gerentes de Conta e da Agência: manter o nível de satisfação do seu cliente
Pré-Condições: Cliente deve ser um correntista da instituição financeira.
Pós-Condições: Cliente obtém o resultado referente ao seu pedido de crédito.

Seqüência Típica de Eventos:

Seção Principal
1. Este caso de uso inicia quando o Cliente emite um pedido de crédito rural entregando a documentação necessária ao seu Gerente de Conta.
2. O Gerente de Conta aceita o pedido do Cliente.
3. O Gerente de Conta realiza uma análise sobre o pedido de crédito
4. O Gerente de Conta informa o Cliente do Resultado do seu pedido.
5. O caso de uso é finalizado.

Seqüências Alternativas:
2a. O Gerente de conta necessita de informações complementares
2a1. Este subfluxo inicia quando o Gerente de Conta solicita ao Analista Financeiro uma análise técnica sobre o pedido de crédito e como resultado da análise forneça informações complementares referente ao pedido.
2a2. O Analista Financeiro analisa o pedido de crédito e complementa com informações pertinentes e inerentes ao pedido de crédito.
2a3. O fluxo retorna ao item 3 da seção principal.

4a. O Gerente de conta não possui alçada para aprovar o pedido
4a1. Este subfluxo inicia quando o Gerente de Conta aprova o pedido de crédito, porém, não possui alçada suficiente para finalizar o fluxo.
4a2. O Gerente da Agência analisa o pedido de crédito, homologando ou não a decisão do Gerente de Conta.
4a3. O caso de uso é finalizado.

Figura 6.3: Descrição do caso de uso do negócio "Emitir Pedido de Crédito Rural"

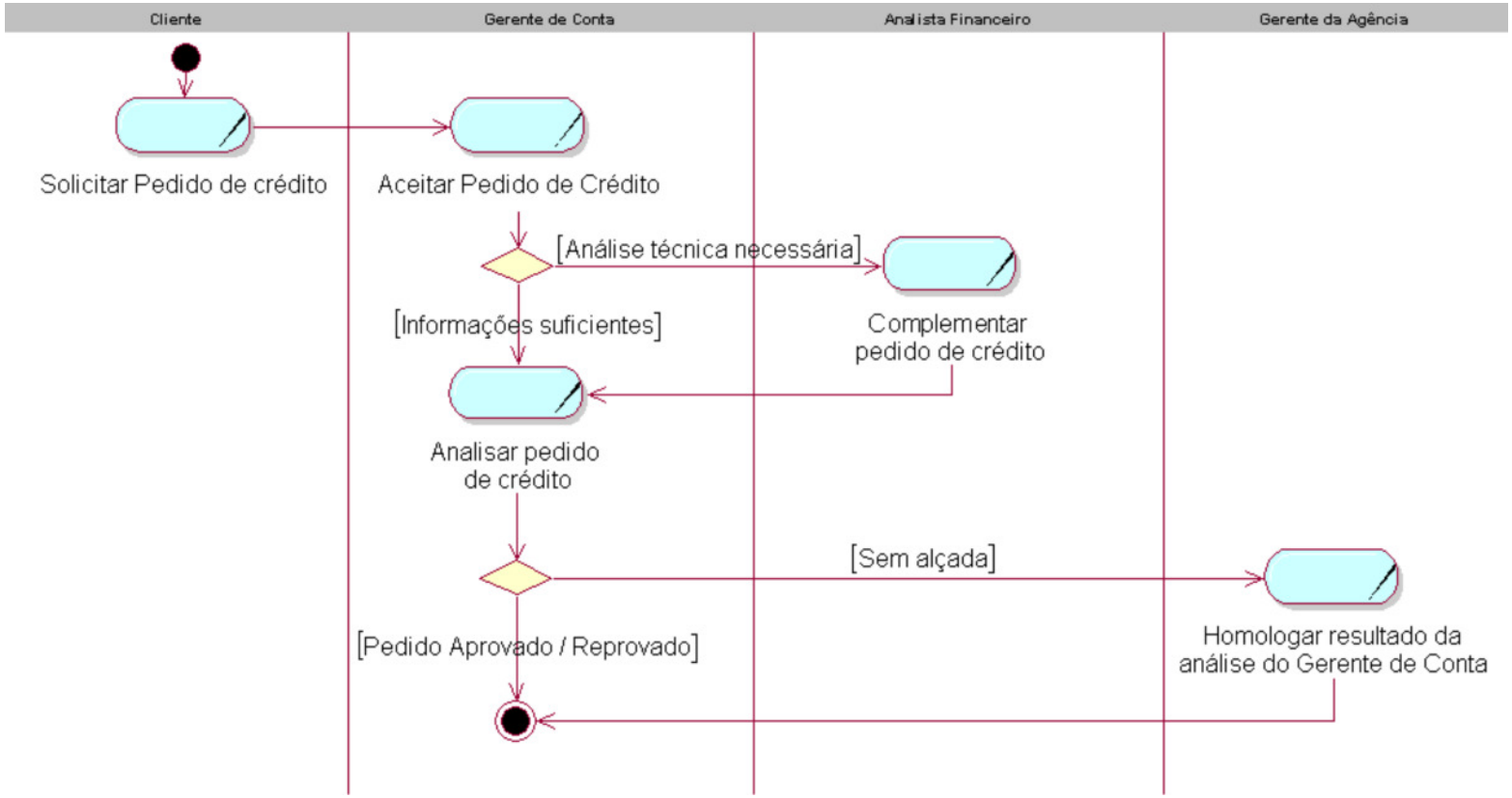


Figura 6.4: Diagrama de Atividades do Negócio para o caso de uso Emitir Pedido de Crédito Rural

O diagrama de atividades do negócio é o artefato mais importante desta extensão, uma vez que é através dele que será realizada a identificação de papéis de agentes. Esta etapa representa a maior alteração proposta na versão atual do MASUP, já que o processo de identificação originalmente transcorrido no *workflow* de análise e, com esta nova proposta, o referido processo é transferido para o *workflow* de negócio. A motivação para o uso do diagrama de atividades do negocio para identificação de agentes está relacionada ao fato deste ser o principal diagrama a expressar os comportamentos e características do negócio de uma organização. Embora o mesmo tipo de diagrama seja utilizado na fase de requisitos e análise na modelagem de sistemas nestas fases o objetivo é a busca por um modelo de sistema e uma solução computacional que dê suporte ao que é especificado na modelagem de negócio. Os comportamentos e características do negócio tornam-se bastante importantes para a identificação de agentes uma vez que os próprios agentes são dotados de tais comportamentos e características, conforme já discutido nas seções 2.1 e 2.3.

Como descrito no Capítulo 4, a modelagem de negócio é utilizada para criar uma abstração de uma empresa e de suas atividades, com a finalidade de representar quais destas atividades são executadas e como são articuladas para atingir os objetivos da organização. Diante desse fato, nessa etapa do desenvolvimento de software é que os processos da organização são modelados em um nível de abstração mais alto em relação aos modelos criados pela disciplina de requisitos.

Por envolver conceitos de alto nível dentro da organização (descritos na seção 4.2), com a modelagem de negócio somos capazes de identificar certos comportamentos e/ou características nos processos de negócio que são muito difíceis de serem identificados na etapa de requisitos, uma vez que nesta etapa, o objetivo é a busca por um modelo de sistema e uma solução computacional que dê suporte ao que é especificado na modelagem de negócio. Quando utilizamos os termos "comportamentos" e "características" vamos ao encontro de assuntos abordados nos capítulos anteriores.

Em referência aos comportamentos, cita-se o assunto discorrido na seção 2.1, em que apresenta-se aquelas que são, de acordo com a literatura, as principais características de um agente, tais como pró-atividade, reatividade, autonomia e por fim a habilidade social. Já em relação as características, fazemos referência aos conceitos estudados na seção 5.2, onde podemos definir o grau de estruturação das atividades do negócio bem como o grau de interdependência entre as mesmas. Na modelagem de negócio também pode-se encontrar atividades que exigem tomada de decisão por parte dos trabalhadores do negócio envolvidos em cada atividade do negócio, motivo pelo qual optou-se por trabalhar com esta disciplina para a identificação de agentes.

Na seção 4.4 é apresentada uma abordagem introduzida pelo RUP que serve como guia na transformação de um modelo de negócio para um modelo de sistema, ou seja, partindo de um conceito mais abstrato e de generalização mais alta, para um modelo mais concreto e mais específico.

Ao se trabalhar diretamente com a disciplina de requisitos, sem a realização da modelagem de negócio, muito se perde em relação as abstrações que são criadas por meio dos modelos do negócio em relação a organização, uma vez que, quando se parte para o desenvolvimento de software com a modelagem de requisitos, modelos para uma solução computacional para um determinado do processo de negócio já estão sendo desenvolvidos.

O diagrama ilustrado pela Figura 6.4, que representa as atividades de um processo de negócio, provê meios para a identificação de papéis de agentes. Esses meios passam por uma análise das atividades sob o ponto de vista das características referentes ao nível de estruturação do problema bem como do relacionamento destas atividades com outras dentro do processo de negócio, para que em uma etapa posterior, seja feita a interpretação do resultado dessa análise para avaliar a necessidade ou não de papéis de agentes. Para uma melhor compreensão dessa proposta, é introduzida a Tabela 6.1.

Tabela 6.1: Tipos de Decisão x Relação entre Atividades

Tipo de Decisão	Relação entre atividades		
	Independente	Seqüencial Interdependente	Agregado Interdependente
Estruturada	O.O. O.A.	O.A.	O.A.
Semi Estruturada	O.O. O.A.	O.A.	O.A.
Não Estruturada	R.H.	R.H.	R.H.

Na Tabela 6.1 cada célula da tabela identifica a solução computacional a ser utilizada como sendo a mais adequada de acordo com as características do tipo de decisão em função do relacionamento das tarefas do processo decisório. Temos então por O.O. uma solução utilizando o paradigma orientado a objetos, O.A. uma solução utilizando o paradigma orientado a agentes e por fim temos R.H., onde nenhum paradigma computacional se adequa uma vez que surge o raciocínio humano como base para a tomada de decisão. Cada cenário apresentado na Tabela 6.1 será discutido na seqüencia, onde os motivos pelos quais as soluções computacionais foram propostas são explicados e exemplificados através de referências bibliográficas.

- **Estruturada x Independente:** de acordo com a natureza das características dos problemas estruturados, a definição e as fases de operação para atingir o

objetivo proposto são bem definidas, criando assim a possibilidade de execução repetidamente sem alterar o resultado esperado. Desta forma pode-se afirmar que problemas desta ordem associados a tarefas independentes, onde apenas um tomador de decisão tem completa autoridade e responsabilidade sobre o processo decisório, requerem uma implementação sob o paradigma procedural, neste caso a orientação à objetos. Entretanto também podem ser implementados, não obrigatoriamente, utilizando a orientação a agentes. Assim tem-se uma seqüência de passos já definidos a serem executados por objetos dentro do sistema, sem a necessidade de comunicação com outras entidades de cunho decisório (outros objetos ou agentes inteligentes) dentro do sistema.

- **Exemplo 1 -** Este exemplo, introduzido pelo artigo [40], apresenta um exemplo de aplicação multiagentes que trabalha lado a lado com o decisor oferecendo soluções estratégicas para o usuário. Tal aplicação utiliza um *framework* do tipo *task-sharing*, uma vez que o problema principal é dividido em pequenos sub-problemas que são então atribuídos para agentes, onde estes, utilizando-se da tecnologia de sistemas inteligentes, realizam suas tarefas produzindo recomendações estratégicas de acordo com o problema fornecido. Os agentes dessa aplicação estão classificados nesta categoria uma vez que estes utilizam algoritmos já conhecidos e que são aplicados sobre uma quantidade de informações previamente fornecida.
- **Exemplo 2 -** A aplicação apresentada no artigo [24] introduz um sistema para suporte ao usuário no que se refere a leilões na internet. A aplicação é dividida em três agentes: o comprador, o vendedor e o agente que provê informações referente aos leilões de interesse do usuário. O nosso alvo nessa aplicação para este exemplo é o agente responsável pelo fornecimento de informações, uma vez que este age de forma autônoma no sistema buscando e recomendando os leilões que mais se aproximam das necessidades do usuário. Tais leilões são recomendados de acordo com uma base de dados que contem regras de recomendações, classificando assim o problema a ser resolvido por este agente como estruturado, uma vez que a recomendação é criada utilizando regras já definidas no sistema, sem a necessidade de um resolução mais complexa de problemas por parte do agente.
- **Exemplo 3 -** No artigo [74] é apresentado um sistema musical interativo para auxiliar músicos em geral. Tal aplicação é relativamente simples, uma vez que os agentes implementados neste sistema são responsáveis por emitirem notas musicais de acordo com as notas previamente disponibi-

lizadas no ambiente por outros agentes. A sucessão de notas fornecidas pelos agentes tende a formar uma melodia. Tais agentes estão classificados nesta parte uma vez que deles apenas é exigida a autonomia, uma vez que os estes ficam observando o ambiente procurando por notas musicais para então, aplicarem um algoritmo para gerarem a próxima nota de acordo com um mecanismo de inferência que utiliza uma base de dados com notas musicais previamente cadastras e que ao longo do funcionamento da aplicação é atualizado pelos próprios agentes.

- **Estruturada x Seqüencial Interdependente:** no processo de tomada de decisão onde as tarefas para a tomada da decisão são interdependentes, surge a presença de outros tomadores de decisão, e entre estes, a necessidade de comunicação, uma vez que o resultado de uma tarefa decisória é passado adiante dentro do processo como um todo. Problemas estruturados cujas tarefas de decisão são do tipo interdependentes podem sem implementados utilizando o paradigma orientado a agentes, já que surge como resultado das relações entre as tarefas a comunicação entre os tomadores de decisão.

  - **Exemplo 1 -** A aplicação apresentada no artigo [67] apresenta uma solução orientada a agentes para sub-estações de energia elétrica. Esta solução está estruturada da seguinte forma: cada subestação possui um agente coordenador, responsável pela coordenação dos agentes que nela estão ligados. Este agente coordenador também age como uma interface provendo serviços, informações e outras funcionalidades a outras sub-estações que com essa mantém um canal de comunicação. Cada dispositivo (*hardware*) da sub-estação possui um agente trabalhando a seu favor, buscando informações no ambiente, realizando *backups*, entre outras atividades. O agente possui uma funcionalidade de decidir desligar o dispositivo dependendo do estado atual do mesmo e das alterações no ambiente com o intuito de não prejudicar o funcionamento. Porém, tal decisão é repassada ao agente coordenador, que realiza uma avaliação da decisão do agente buscando encontrar possíveis mudanças negativas no ambiente, podendo este reivindicar a decisão do agente ou acatá-la. Esta troca de informações e decisões entre os agentes se caracteriza como um seqüenciamento de tarefas decisórias, onde ambos os agentes procuram, através da intenção conjunta, manter a estabilidade dos dispositivos bem como da própria sub-estação.
  - **Exemplo 2 -** O artigo entitulado *Designing multi-agent systems: a framework and application* [49], apresenta uma solução multiagentes para um sistema inteligente de transporte e de informações pré-viagens. Na

arquitetura deste sistema, foi criado um agente responsável pelo recebimento do pedido do cliente e um agente responsável por cada área relacionada ao sistema de transporte: informações do tempo, vôos, estradas e origem e destino do trajeto a ser realizado pelo passageiro. Após capturar o pedido, o agente repassa este pedido para cada um dos agentes especialistas, estes por sua vez, buscam os melhores resultados que estejam de acordo com as necessidades inclusas no pedido do cliente, e retornam os resultados para o agente que unificará todas as informações e indicará para o passageiro os melhores trajetos e datas para a sua viagem. Percebe-se nesta arquitetura um seqüenciamento das atividades, onde as decisões são passadas do nível mais inferior (agentes especialistas) para o agente principal, caracterizando o problema tratado nesta aplicação como seqüencial-interdependente. Em relação ao tipo de problema, este problema é caracterizado como um problema estruturado uma vez que a captura de informações sobre o transporte e a criação de itinerários que melhor se adaptam a estas informações não exigem capacidade de inteligente, e sim um algoritmo estruturado para a realização desta tarefa.

- **Estruturada x Agregado Independente:** neste cenário os problemas ainda são bem definidos bem como as fases de operação para obter os resultados esperados permanecem claras. A comunicação introduzida pela forma seqüencial de execução das tarefas do processo decisório ainda persiste, o que varia deste cenário para o anterior é que a decisão deve resultar da uma negociação e da interação entre os responsáveis pela tomada de decisão. No cenário anterior, parte da decisão era tomada e passada adiante, já neste cenário, a decisão é tomada por todos os responsáveis envolvidos no processo decisório.

  - **Exemplo 1** - O primeiro exemplo desta categoria é bem conhecido no meio acadêmio: *Trading Agent Community*, ou apenas TAC. TAC Classic é um ambiente orientado a agentes na internet para negociação de pacotes turísticos, que compreende compra e venda de passagens aéreas, reservas de hotéis, pacotes de diversão, etc. Neste jogo, como assim é chamado por seus criadores, encontramos dois tipos de agentes: o agente responsável pela compra dos produtos escolhidos pelo cliente de uma agência de turismo e os agentes que representam companhias aéreas, hotéis, etc. A negociação para compra dos produtos ocorre através de leilões que ocorrem durante o jogo. Esta aplicação está classificada nesta categoria uma vez que as tarefas dos agentes pertencem ao tipo agredado-independente por causa da negociação envolvida (uma das características de SMAs) e por possuir um problema já conhecido e que não possui fatores possam

alterar o funcionamento do algoritmo que dá suporte ao mesmo.

- **Exemplo 2 -** Em *Multi-agent resource allocation for road passenger transportation* [41], foi introduzida uma solução orientada a agentes para a alocação de recursos em um sistema de transporte público. O principal objetivo dessa aplicação é maximizar o tempo de trabalho de motoristas e o tempo de funcionamento dos veículos de transporte, nesse caso: ônibus. A aplicação trabalha basicamente com dois agentes: um agente é responsável pelo serviço prestado (uma linha de ônibus, por exemplo) pela empresa pública de transporte e o outro agente representa o motorista. Cada linha de transporta e cada motorista possuem um agente dentro da aplicação que contém as informações necessárias para o funcionamento do algoritmo, tais como: hora de início e fim do expediente, hora do intervalo, carga horária mensal, etc. A aplicação não trabalha com falhas de veículos, mas em caso de necessidade, os autores afirmam que a alternativa é realizar a locação dos mesmos. A alocação de motoristas é resultado de uma negociação direta entre o agente da linha e o agente do motorista. O problema trabalhado neste artigo exige a comunicação e a negociação entre os dois agentes, buscando um resultado positivo para o sistema como um todo, dessa forma, podemos classificar tal negociação uma tarefa do tipo agregado-independente.
- **Exemplo 3 -** No artigo *An integrated approach for developing e-commerce applications* [59], é apresentada uma plataforma orientada a agentes cuja funcionalidade é fornecer suporte para negociações de compra e venda através da internet. A estrutura desta plataforma é composta por dois tipos de agentes: o agente comprador e o agente vendedor. O agente comprador é o responsável por capturar as necessidades do usuário, realizar buscas de produtos dentro do ambiente multiagentes que se qualifiquem dentro das necessidades e então iniciar um processo de negociação com o agente vendedor afim de maximizar a satisfação do usuário. Após as negociações serem feitas, o agente comprador repassa os resultados para o usuário escolher o que vai comprar. As necessidades do usuário podem ser variáveis do tipo preço, quantidade, disponibilidade para entrega, marca, garantia, entre outros. Este processo realizado entre os agentes da plataforma exige uma característica importante referente a SMAs, que é a negociação. Porém, esta negociação é do tipo estruturada, uma vez que o problema é conhecido: maximizar a satisfação do cliente dentro das necessidades do mesmo; desta forma o problema tratado por este artigo foi classificado como estruturado e as tarefas realizadas entre os agentes

definidas como agregadas e independentes, uma vez que o resultado esperado deriva da negociação entre os agentes.

- **Semi-Estruturada x Independente:** neste cenário a solução para os problemas deixam de estar bem definidas e passam a ter parte da definição do resultado esperado variável, tornando um resultado que antes já era esperado em algo que agora passar a ser indefinido. Problemas semi-estruturados são aqueles que algumas partes do problema podem ser resolvidas por métodos de decisão formais e/ou por meio da automatização, assim uma parte do problema exige um grau de inteligência para a tomada da decisão. Dessa necessidade por inteligência durante a tomada de decisão, tem-se que problemas classificados neste cenário tendem a requerer a orientação a agentes como solução computacional adequada.

  - **Exemplo 1 -** O artigo *A Multi-Agent System for Building Control* [54] apresenta uma solução chamada MASBO. Esta aplicação realiza o controle interno de ambientes buscando uma melhor economia de energia sem afetar as preferências dos ocupantes. Segundo os autores, MASBO surge como uma melhoria a sistemas de controle de ambientes já existentes adicionando características de SMAs, tais como: aprendizagem, raciocínio e autonomia. O principal objetivo do sistema é manter o conforto das pessoas que se fazem presente no ambiente, e para isso, o sistema deve utilizar as preferências de cada um para tomar as suas decisões, como por exemplo: fechar janelas, diminuir intensidade da luz, aumentar a temperatura do ambiente, entre outras. Porém, o ambiente pode estar em constante mudança, uma vez que podem surgir novos ocupantes, sistemas de ar condicionado e iluminação podem falhar. Podemos determinar essas mudanças no ambiente como variáveis do problema no qual a aplicação busca solucionar, por isso MASBO está classificado nesta categoria, pois seus agentes são autônomos e estão lidando com um problema não estruturado.
  - **Exemplo 2 -** No artigo *Operation of a Multiagent System for Microgrid Control* [17] encontramos uma proposta orientada a agentes que busca controlar de forma eficiente plantas de energia elétrica, nesse caso, estação de armazenamento e distribuição de energia. Diferente do artigo já apresentado que também propõe um sistema para controle de estações de energia elétrica, nesse em questão, a aplicação apresentada faz referência ao fato de que dispositivos podem falhar sem previsão e com isso o sistema deve buscar alternativas dentro do ambiente para que o mesmo continue estável. Esta característica demonstra que o problema a ser solucionado

é do tipo semi-estruturado. Dentro da arquitetura da solução proposta, damos destaque à um agente que é responsável pelo gerenciamento da estação de energia, cuja alçada abrange decisões a serem tomadas referente ao estado da estação de forma autônoma toda vez que alguma mudança no ambiente é realizada. Desta forma, temos neste sistema duas características que o classificam nesta categoria: um problema semi-estruturado e a autonomia do agente que gerencia a estação de energia.

  - **Exemplo 3 -** Uma solução multiagentes para o controle de recursos em uma linha de produção é apresentada em *Survivability of a Distributed Multi-Agent Application - A Performance Control Perspective*. Esta aplicação é composta por um agente que é responsável pelo controle da carga de trabalho e pela alocação de recursos de acordo com a demanda em uma linha de produção. Este agente trabalha de forma autônoma procurando por modificações no ambiente que podem variar desde novas solicitações até falhas de componentes. Tais modificações podem acontecer de maneira inesperada e sem algum aviso, fazendo com que o agente tenha que buscar em tempo de execução alternativas que mantenham o estado atual da linha de produção estável. Este é um tipo de problema que se caracteriza como semi-estruturado, uma vez que problemas e/ou variações dentro da linha de produção podem ocorrer de forma inesperada, exigindo do agente um grau de solução mais complexo em relação à um problema estruturado.

- **Semi-Estruturada x Seqüencial Interdependente:** diferente do cenário anterior onde apenas um decisor lidava com um problema semi-estruturado, neste cenário a característica do problema requer o relacionamento entre as tarefas do processo de decisão que passa a ser seqüencial. Assim, tem-se dois ou mais decisores interagindo de forma seqüencial para solucionar um problema que exige um grau de inteligência em relação aos problemas estruturados. Diante destas características, podemos afirmar que o paradigma orientado a agentes surge como única opção para a implementação computacional.

  - **Exemplo 1 e 2 -** Nos artigos *Towards Adaptive Workflow Enactment Using Multiagent Systems* e *Multiagent Systems with Workflows*, respectivamente [8, 65], encontramos duas soluções propostas para ambientes de *workflow* (fluxo de trabalho). Um *workflow* pode ser definido como a seqüência de passos necessários para que se possa atingir a automação de processos de um negócio, de acordo com um conjunto de regras definidas, envolvendo a noção de processos, permitindo que estes possam ser transmitidos de uma pessoa para outra de acordo com algumas regras. Pode-

mos considerar as atividades nesse tipo de ambiente como seqüenciais e interdependentes, uma vez que as decisões são tomadas em cada etapa e passadas adiante até que o fluxo de trabalho chegue ao seu final. Estes artigos estão classificados nesta categoria de acordo com o tipo de relação entre as tarefas bem como pelo tipo de problema envolvido em ambas aplicações. Como o ambiente organizacional onde uma aplicação deste tipo está empregada geralmente sofre mudanças, o problema envolvido possui muitas variáveis (mudança de recursos, informações em constante atualização, etc) e muitas vezes tais mudanças se tornam difíceis de serem mapeadas, fazendo com que o processo de negócio implementado apresente a característica de um problema não-estruturado.

  - **Exemplo 3 -** Uma Sistema de Apoio a Decisão orientado a agentes é apresentado no artigo *Agent Coordination for Bus Fleet Management* [7]. Tal solução foi criada para dar suporte à uma empresa espanhola de transporte urbano com o intuito de manter a qualidade dos serviços. Nessa solução os ônibus são controlados por uma central, onde um agente realiza o papel de coordenador. Cada ônibus é representado por um agente e possui um GPS que mantém a central atualizada sobre a sua posição geográfica. O agente responsável pelo ônibus verifica constantemente a situação do serviço e havendo anormalidades poderá tomar uma decisão propondo alternativas para melhorar o serviço. Tal decisão é repassada para o agente central que verifica se essa poderá afetar o funcionamento de outras linhas diminuindo assim a qualidade do serviço e, não encontrando alterações que possam prejudicar outras linhas, pode aceitar a decisão e colocá-la em prática. As anormalidades que podem afetar uma linha de ônibus são inúmeras, tais como congestionamento, acidentes, obras de reparação em estradas, capacidade total do ônibus atingida (necessitando de mais veículos), etc. Lidar com estas mudanças não é uma tarefa trivial e exige uma complexidade maior de processamento por parte dos agentes uma vez que os problemas que podem ocorrer são imprevisíveis, que podem assim ser classificados como problemas não-estruturados, de acordo com a definição apresentada.

- **Semi-Estruturada x Agregado Independente:** neste cenário apenas uma solução computacional é possível: paradigma orientado a agentes. Problemas semi-estruturados exigem um grau de inteligência por parte do decisor e tarefas definidas como agregadas independentes exigem uma comunicação entre os decisores envolvidos no processo de tomada de decisão. Sendo assim, a inteligência para solucionar o problema somada a comunicação introduzida

pelas relações entre as tarefas nos remete à um cenário próprio de um sistema multiagentes.

- **Exemplo 1 -** No artigo *Application of multiagent systems in project management* [75] encontramos uma proposta de uma aplicação orientada a agentes para uma disciplina bem conhecida dentro da engenharia de software: gerência de projetos. O gerenciamento de projetos é por si só um problema semi-estruturado uma vez que dentro desse, existem muitas disciplinas e workflows a serem controlados pelo gerente de projetos. Esse, por sua vez, precisa lidar com inúmeras variáveis durante seu trabalho, necessitando muitas vezes utilizar a negociação entre as pessoas que estão envolvidas para encontrar o melhor caminho para o desenvolvimento do projeto. Na aplicação multiagentes apresentada no artigo, cuja arquitetura é distribuída, encontramos três tipos de agentes que representam os recursos (pessoas, computadores, etc), atividades e serviços. O principal problema dentro do gerenciamento do projeto, que é a alocação de recursos, é tratado via negociação (motivo pelo qual está enquadrado dentro desta categoria) entre os agentes responsáveis pelas atividades e recursos, enquanto o agente responsável pelos serviços realiza tarefas mais alto nível, tais como cálculo do caminho crítico, estimativas de custo, tempo, etc.
- **Exemplo 2 -** O artigo *Resource Allocation in Continuous Production using Market-Based Multi-Agent Systems* [66] apresenta uma proposta multiagentes similar ao exemplo anterior, alocação de recursos em ambientes distribuídos: Contudo, não é uma proposta para um domínio específico de aplicação, sendo assim genérico. Um exemplo de uma refinaria de açúcar foi citado como forma de validação do trabalho proposto pelos autores. Como a solução apresentada possui as mesmas características do exemplo anterior, o problema abordado no artigo foi classificado nesta categoria.

- **Problemas não estruturados:** a natureza desde tipo de problema nos remete a cenários completamente imprevisíveis, onde a solução para o problema é completamente indefinida, muito menos o processo através do qual se deseja passar para conseguir algum resultado desejado é conhecido. Problemas desta ordem são complexos demais para serem projetados para uma solução computacional utilizando os paradigmas em discussão, já que o raciocínio humano se faz necessário durante o processo decisório. Como já era esperado no decorrer deste estudo, não foram encontrados na literatura artigos que fizessem

referência a soluções orientadas a agentes que dêem suporte a problemas não-estruturados, motivo pelo qual utilizamos o raciocínio humano como principal requisito para a busca de soluções de tais problemas.

Apresentado o embasamento teórico referente a Tabela 6.1 e listadas aplicações como exemplo para cada cenário existente na tabela, podemos realizar um levantamento das soluções computacionais para cada tipo de problema em relação aos tipos de tarefas, bem como fazer uma análise sob a perspectiva de sistemas multiagentes identificando as características deste tipo de sistema, como mostra a Tabela 6.2. A construção dessa tabela se deu por meio de um estudo realizado sobre as aplicações de sistemas multiagentes descritas anteriormente, buscando extrair de cada aplicação os requisitos necessários para a construção de um SMA. Esta tabela apresenta o levantamento das características dos SMAs então definidos como requisitos para cada cenário formado por tipos de decisão *versus* relação entre atividades.

Tabela 6.2: Requisitos para Tipos de Decisão x Relação entre Atividades

Tipo de Decisão	Relação entre Atividades		
	Independente	Seqüencial Interdependente	Agregado Interdependente
Estruturada	Autonomia	Comunicação Cooperação	Comunicação Negociação
Semi Estruturada	Autonomia	Comunicação Cooperação	Comunicação Negociação

Com base no que foi discorrido na sub-seção 2.3.2, verifica-se na Tabela 6.2 que as características oriundas dos sistemas multiagentes variam conforme os cenários. Quando identifica-se uma atividade do tipo independente, não importando o problema em questão, tem-se a autonomia como requisito, uma vez que nesses tipos de atividade não se faz necessária a interação do decisor com outros. Entretanto, quando trabalha-se com as atividades dos tipos Seqüencial-Interdependente e Agregado-Interdependente verifica-se a necessidade de comunicação entre os decisores, já que para estas atividades, o processo de negócio requer mais do que um decisor. Além disso, os requisitos não ficam restritos apenas à comunicação, requer-se uma coordenação por parte do sistema multiagentes. As atividades seqüenciais e interdependentes, por sua natureza (conforme explicitado na sub-seção 5.2.3), requerem uma cooperação entre os decisores, já que a decisão é tomada em diferentes níveis do processo de negócio. No caso das atividades agregadas e interdependentes, a negociação é necessária, uma vez que a decisão resultante do processo de negócio é resultado de uma negociação entre os decisores.

O processo de identificação de papéis de agentes tem como base as Tabelas 6.1 e 6.2. Esse processo decorre de uma análise sobre as atividades realizadas pe-

los trabalhadores do negócio para que então estas sejam mapeadas em algum dos cenários apresentados nas tabelas. A análise a ser feita leva em conta a existência de uma tomada de decisão por parte do trabalhador do negócio; não havendo uma decisão envolvida na atividade, a mesma não será mapeada. Quando da necessidade de uma tomada de decisão, a atividade então passa a ser uma candidata a ser associada à um papel de agente, dependendo do cenário em que esta for mapeada. O mapeamento das atividades que requerem tomada de decisão deve levar em conta o grau de estruturação do problema, onde identifica-se se o problema envolvido na tomada de decisão está inserido num contexto de problema estruturado, semi-estruturado e não estruturado. Além do grau de estruturação do problema, para que a atividade seja caracterizada dentro de um cenário, o relacionamento das atividades também deve ser considerado. Deve-se identificar o tipo de relacionamento entre as atividades que exigem tomada de decisão, sendo elas independentes, seqüencial-interdependentes ou agregado-interdependentes.

Na subseção seguinte, é apresentado o diagrama de atividades do negócio estendido, onde são representadas as alterações realizadas com a identificação de papéis de agentes.

6.2.4 Diagrama de Atividades do Negócio Estendido

Após a utilização dos conceitos apresentados para a identificação de papéis de agentes através do diagrama de atividades do negócio, um segundo diagrama de atividades é produzido. Na verdade, este diagrama surge como parte da transformação do diagrama de atividades anterior, abrangendo as alterações registradas com a identificação de papéis de agentes. Vale lembrar que a necessidade deste diagrama é diretamente relacionada com as alterações realizadas na etapa de identificação de papéis de agentes, uma vez que não há motivos para criar um diagrama de extensão sem que hajam mudanças a serem documentadas, no caso, papéis de agentes identificados.

A representação gráfica dos papéis identificados é similar à que representa atores e trabalhadores, feita através de *swimlanes* que utilizam o estereótipo *«agent role»* como prefixo no título. *Swimlanes* (raias) são usadas para definir quais são as classes (ou papéis) responsáveis pela realização de cada atividade e são especialmente úteis para a modelagem de processos empresariais. A Figura 6.5 apresenta o diagrama de atividades estendido para o caso de uso "Emitir Pedido de Crédito Rural", ilustrando os novos papéis identificados bem como o novo fluxo de atividades para este novo cenário.

Na Figura 6.4, tem-se que a atividade "Solicitar Pedido de Crédito" é executada pelo ator do negócio Cliente, entretanto, esta atividade não necessita nenhuma

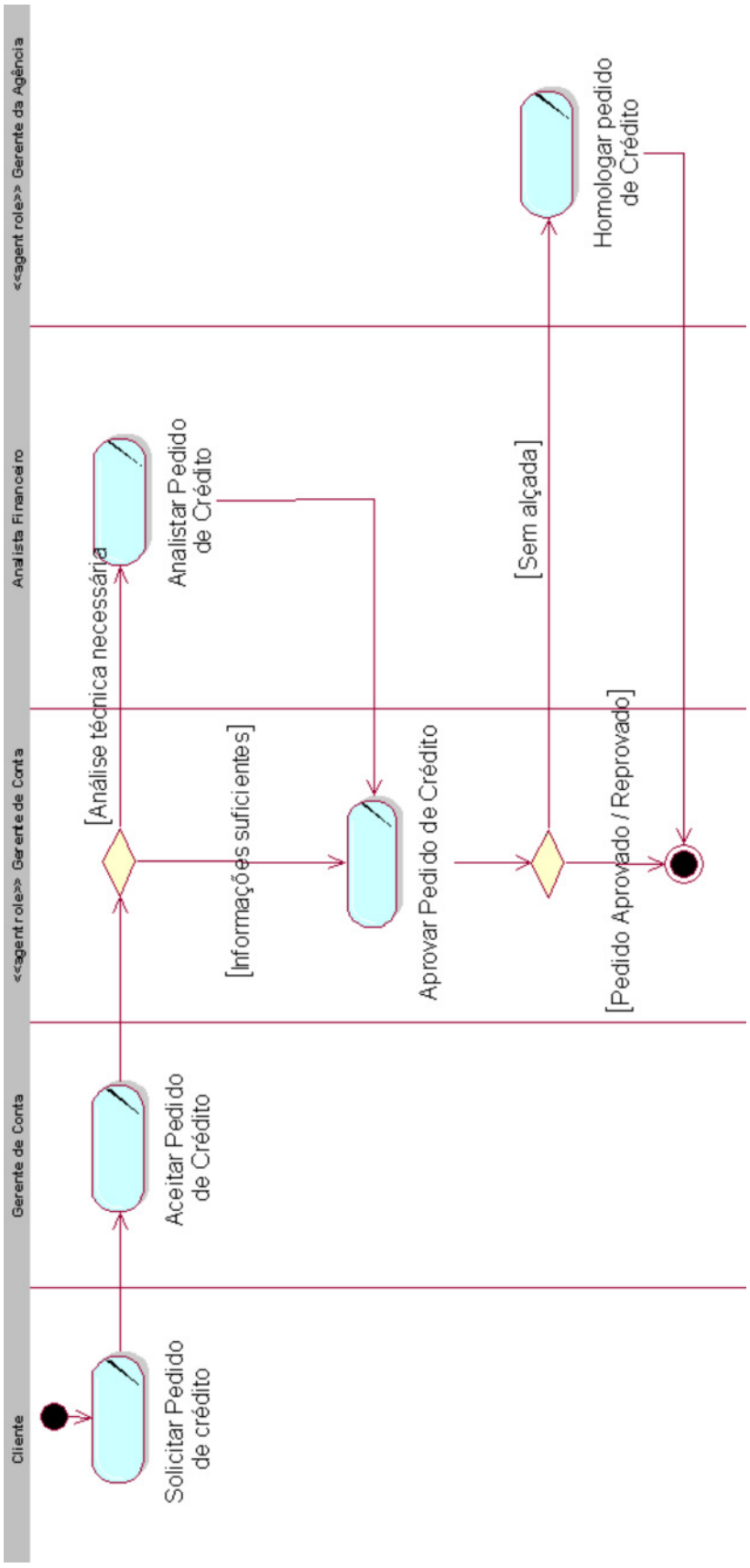


Figura 6.5: Diagrama Estendido de Atividades do Negócio para o caso de uso Emitir Pedido de Crédito Rural

análise para a identificação de papéis de agentes pois o objetivo desta proposta é buscar por atividades executadas apenas pelo trabalhador do negócio. Já a atividade "Aceitar Pedido de Crédito" é executada pelo trabalhador do negócio Gerente de Conta; esta atividade é considerada de simples execução uma vez que não requer nenhum grau de decisão por parte do trabalhador do negócio, além de representar uma interação com o ator do negócio, excluindo a necessidade de um papel de agente para executá-la. Por outro lado, a atividade "Analisar Pedido de Crédito", também executada pelo trabalhador do negócio Gerente de Conta requer uma tomada de decisão: aprovar ou não o pedido de crédito. O problema envolvido nessa tomada de decisão é considerado estruturado, pois a decisão é tomada com base em uma análise do Gerente de Conta sobre as informações fornecidas pelo Cliente (ou das informações complementares fornecidas pelo Analista Financeiro). Porém, a decisão tomada pelo Gerente de Conta, devido a alguma característica inerente ao negócio, como por exemplo o valor solicitado no empréstimo, pode necessitar de uma segunda aprovação de um nível mais superior, o que classifica esta atividade como seqüencial-interdependente. De acordo com a tabela 6.1, atividades mapeadas dentro do cenário problema estruturado com atividades seqüenciais e interdependentes, requerem a orientação a agentes como solução computacional adequada, desta forma, deriva-se do trabalhador de negócio Gerente de Conta, um papel de agente, para executar esta atividade, o *«agent role» Gerente de Conta*, conforme ilustrado na Figura 6.5.

A atividade "Complementar Pedido de Crédito" é executada pelo trabalhador do negócio Analista Financeiro e por ser uma atividade que requer apenas uma análise, sem nenhum grau de decisão, não necessita de nenhum papel de agente para executá-la. O trabalhador do negócio Gerente da Agência realiza a atividade "Homologar Resultado da Análise do Gerente de Conta", que só é necessária quando da falta de alçada por parte do Gerente de Conta. Essa atividade envolve um problema menos estruturado que a atividade "Analisar Pedido de Crédito" pois as variáveis a serem a analisadas pelo Gerente da Agência para homologar ou não a aprovação do Gerente de Conta são desconhecidas, pois podem variar consideravelmente. Diante desse fato, esta atividade é mapeada para um problema considerado semi-estruturado. Também classifica-se essa atividade como seqüencial-interdependente já que a decisão tomada nesta atividade depende de outra previamente realizada. Assim tem-se que a atividade "Homologar Resultado da Análise do Gerente de Conta" passe a ser mapeada, conforme a tabela 6.1 no cenário de problema semi-estruturado com atividades seqüenciais e interdependentes, indicando também, a orientação a agentes como solução computacional mais apropriada para a implementação dessa atividade, dando origem ao papel de agente *«agent role» Gerente da Agência*. O trabalhador do Negócio Gerente da Agência, responsável apenas por uma atividade

dentro do processo de negócio em estudo, deixa de existir com o surgimento de um papel de agente pois a atividade outrora executada por esse trabalhador, passa a ser responsabilidade de um papel de agente.

6.2.5 Diagrama de Papéis

Dentro do processo de identificação de papéis se faz necessária também a representação gráfica dos mesmos. Esta representação segue o padrão do RUP e é introduzida nesta proposta em forma de diagrama, o Diagrama de Papéis. Nesse diagrama, cada entidade (ilustrada pelo estereótipo mostrado na Figura 6.6) representa um papel de agente, e nela são especificadas em forma de lista as atividades que o papel estaria executando. O relacionamento entre as entidades do diagrama representam a idéia de comunicação entre os papéis, sem detalhar o fluxo de comunicação ou mesmo a multiplicidade de cada entidade. O estereótipo é um componente composto por duas seções: na superior, encontra-se o nome do papel e, na inferior, a lista de atividades que podem ser executadas pelo mesmo.

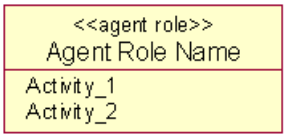


Figura 6.6: Estereótipo proposto para Papel de Agente

Uma vez que os papéis forem identificados, pode-se ilustrá-los visualmente através desse diagrama, artefato incluído por meio desta proposta, uma vez que este não faz parte do conjunto de diagramas oferecidos pelo MASUP. Nesse diagrama são apresentados os papéis identificados na etapa anterior bem como os relacionamentos que definem a interação entre estes papéis. A Figura 6.7 ilustra o diagrama com os papéis que foram identificados anteriormente.

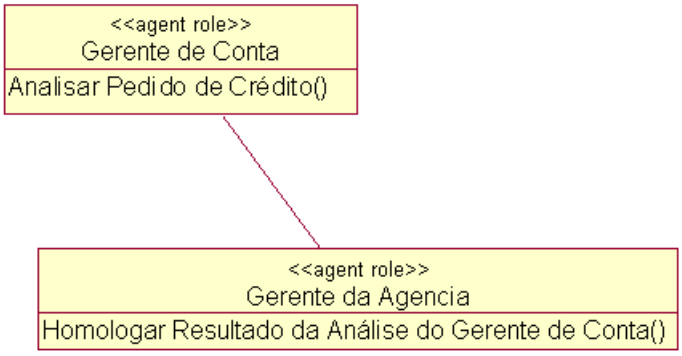


Figura 6.7: Diagrama de Papéis

6.3 *Workflow* de Requisitos

O *workflow* de Negócios é antecessor ao *workflow* de Requisitos no processo de desenvolvimento de software proposto pelo RUP. Motivo pelo qual a sua aplicação implica diretamente no que é produzido no *workflow* de Requisitos, mantendo a coerência e a rastreabilidade entre os artefatos manipulados por ambos. Diante desse fato, a inclusão da Modelagem de Negócios no processo de desenvolvimento de sistemas multiagentes, mais especificamente no MASUP, implica em uma mudança sobre alguns artefatos já definidos anteriormente.

Neste trabalho, o sistema a ser construído é tratado de acordo com o seguinte aspecto: o sistema é composto por dois pacotes para a modelagem do mesmo, o pacote referente a modelagem orientada a objetos e o pacote referente a modelagem orientada a agentes. Não identificando a necessidade de uma solução multiagentes para o sistema, este passa a ser modelado apenas por meio do pacote da modelagem orientada a objetos. Como o nome dos pacote já diz, o pacote da modelagem orientada a objetos abrange todos os casos de uso do sistema que não são executados ou iniciados por papéis de agentes, por outro lado, no pacote da modelagem orientada a agentes, são especificados os casos de uso onde existem interações com os papéis de agentes. Entretanto, pode-se ter os casos de uso da orientação a objetos referenciando os casos de uso especificados na modelagem orientada a agentes, criando desta forma, uma dependência entre os dois pacotes. A Figura 6.8 ilustra um diagrama de pacotes demonstrando a relação de dependência entre os pacotes. Esta relação de dependência entre os pacotes não está restrita apenas ao cenário em que casos de uso de um pacote referenciam outro, mas também quando um ator do sistema interage com um caso de uso que se encontra no pacote da modelagem orientada a agentes.

O diagrama de casos de uso do sistema é um artefato que sofre uma extensão de acordo com o propósito deste trabalho. Na subseção 4.4, é apresentada a proposta do processo unificado para a transição do modelo de negócio para o modelo de sistema. Na Figura 6.9 encontra-se o diagrama de casos de uso do sistema que derivou do diagrama estendido de atividades de negócio referente ao caso de uso do negócio Solicitar Empréstimo Financeiro.

De acordo com o método de transformação do modelo de negócio para o modelo de sistemas apresentado na subseção 4.4, cada trabalhador do negócio torna-se um ator do sistema e cada atividade do negócio passa a ser um caso de uso do sistema. No diagrama estendido de atividades do negócio (Figura 6.5), os dois trabalhadores do negócio Gerente de Conta e Analista Financeiro deram origem aos atores do sistema de mesmo nome.A novidade neste diagrama está na presença de um novo esteriótipo *«invoke»* para representar o relacionamento de casos de uso de

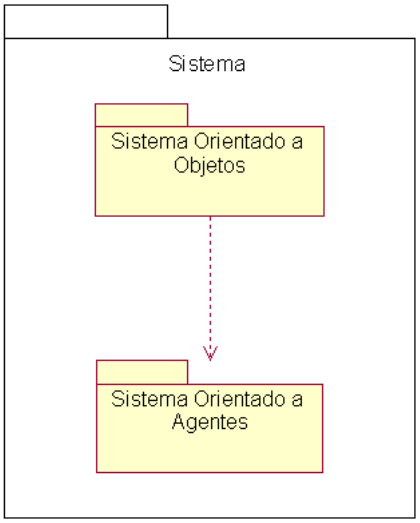


Figura 6.8: Diagrama de Pacotes do Sistema

sistema com casos de uso executados por papéis de agentes.

Na linguagem de modelagem UML, existe um esteriótipo de relacionamento que é denominando *«include»*. Essa notação é usada para representar sub-fluxos complexos e comuns a vários casos de uso, sempre usados, isto é, necessários. Na prática, o caso de uso "incluído"é referenciado no fluxo do caso de uso que o inclui. O caso de uso A inclui o caso de uso B quando B representa uma atividade complexa, comum a vários casos de uso. Entretanto, este esteriótipo de inclusão passa a ser inadequado quando um caso de uso do pacote da modelagem orientada a objetos se relaciona com um caso de uso do pacote da modelagem orientada a agentes, uma vez que, ao se relacionar com um caso de uso executado por um papel de agente, o caso de uso da modelagem orientada a objeto estaria invocando o agente, motivo pelo qual optou-se pela criação de um novo esteriótipo *«invoke»* para suportar a idéia utilizada neste trabalho.

6.4 *Workflows* de Análise e Projeto

Com a introdução da Modelagem de Negócio no MASUP com o intuito de antecipar o processo de identificação de papéis de agentes, algumas modificações no restante das atividades se fazem necessárias para entrarem em conformidade com a proposta do trabalho.

O MASUP mapeia a disciplina de Análise e Projeto do RUP em dois modelos: *workflow* de Análise e *workflow* de Projeto, respectivamente, como ilustrado na Figura 3.5. Na atual estrutura do MASUP, o *workflow* de Requisitos, se mantém inalterada em relação ao RUP, entretanto, no *workflow* de Análise e Projeto é

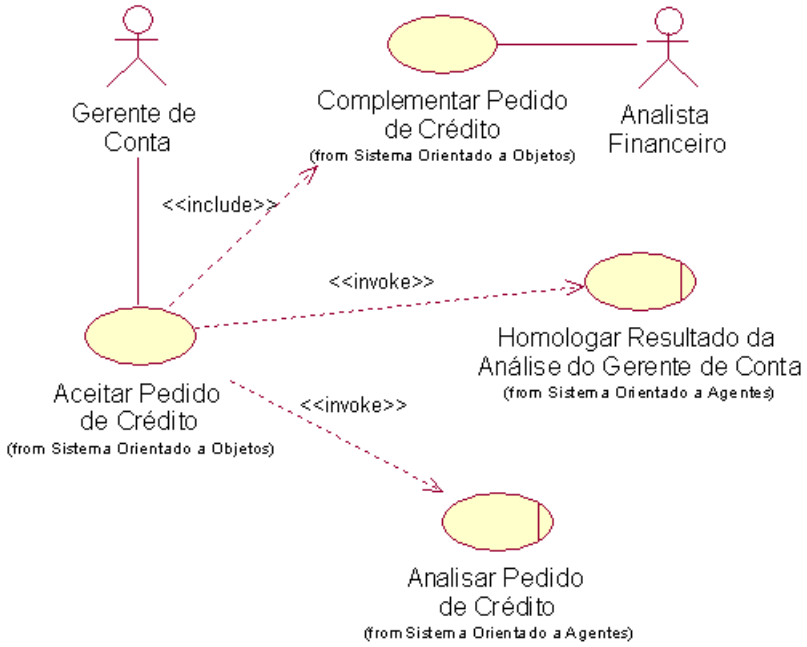


Figura 6.9: Diagrama de Casos de Uso do Sistema

onde ocorre mudanças nas atividades quando identificada a necessidade de modelar um sistema multiagentes, caso contrário, segue-se com a proposta de modelagem definida pelo RUP.

6.4.1 *Workflow* de Análise

O *workflow* de análise requer mudanças com a identificação de papéis de agentes sendo realizada no *workflow* de Negócio, uma vez que neste *workflow* é que se encontra as atividades referentes ao reprojeto dos diagramas de atividades e a identificação de papéis de agentes. Atualmente, o MASUP compreende as seguintes atividades no *workflow* de análise [6]:

- Reprojeto dos diagramas de atividades para modelar a solução orientada a agentes;
- Identificação dos papéis necessários para a solução orientada a agentes baseado nos diagramas de atividades gerados na atividade anterior;
- Especificação de cada papel de agente definindo seus atributos;
- Identificação dos agentes que implementarão os papéis de agentes especificados;
- Definição dos relacionamentos entre os agentes compondo a arquitetura sociedade de agentes;

Uma vez que os papéis já foram identificados no *workflow* de Negócio, as atividades acima necessitam ser redefinidas, de forma a entrar em conformidade com a metodologia como um todo. Com esta reorganização, as atividades realizadas no *workflow* etapa de análise passam a ser definidas de acordo com a listagem abaixo:

- Refinamento dos papéis de agentes identificados na Modelagem de Negócio;
- Identificação dos agentes que implementarão os papéis de agentes especificados;
- Definição dos relacionamentos entre os agentes compondo a arquitetura sociedade de agentes;

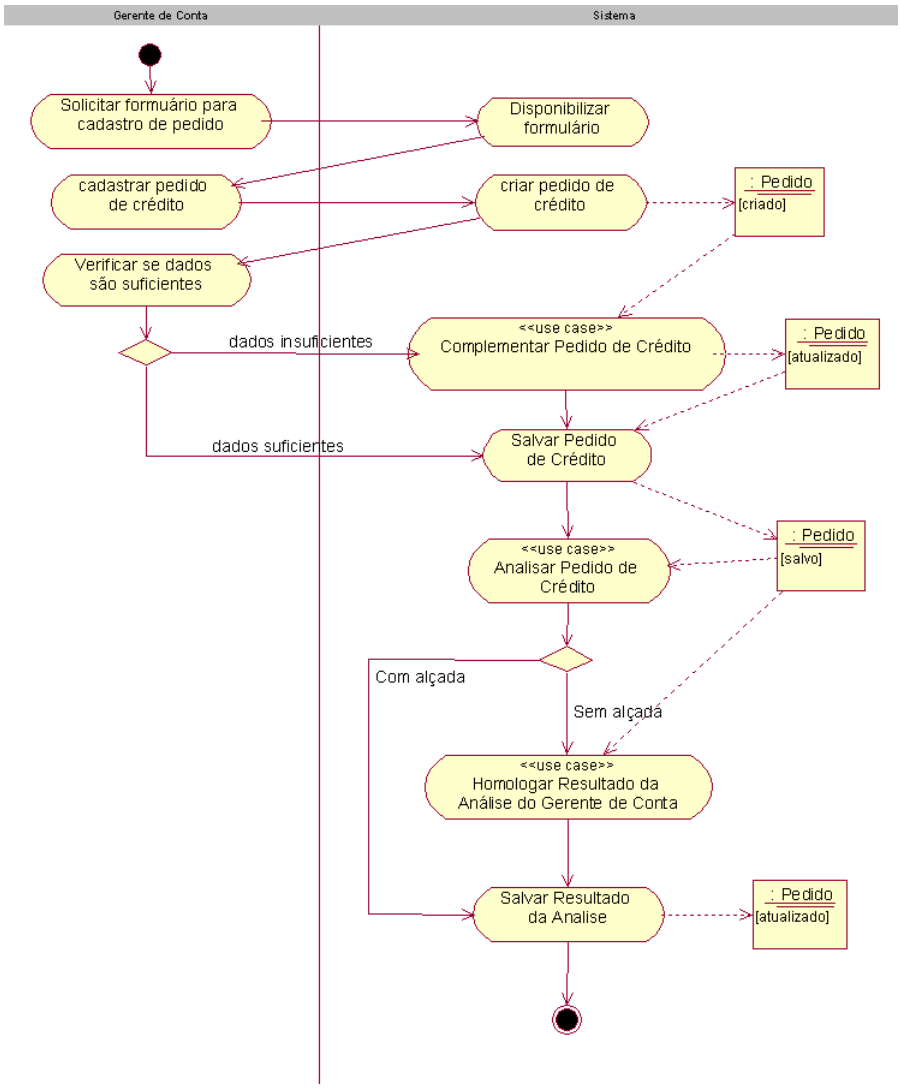


Figura 6.10: Diagrama de Atividades para o caso de uso Aceitar Pedido de Crédito

A Figura 6.10 ilustra o diagrama de atividades UML para o caso de uso "Aceitar Pedido de Crédito", executado pelo ator do sistema Gerente de Conta. Percebe-se nessa ilustração a presença das atividades dos casos de uso executados

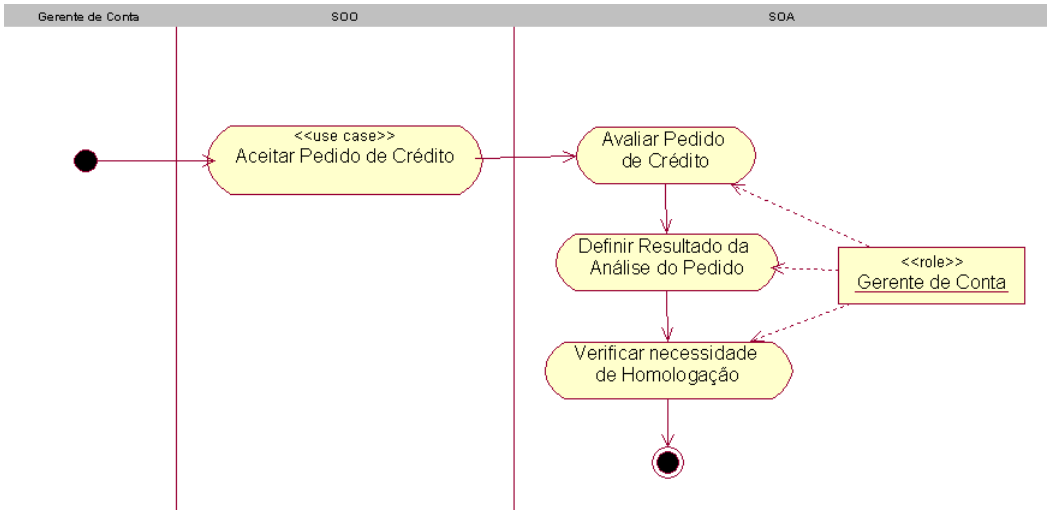


Figura 6.11: Diagrama de Atividades para o caso de uso Analisar Pedido de Crédito

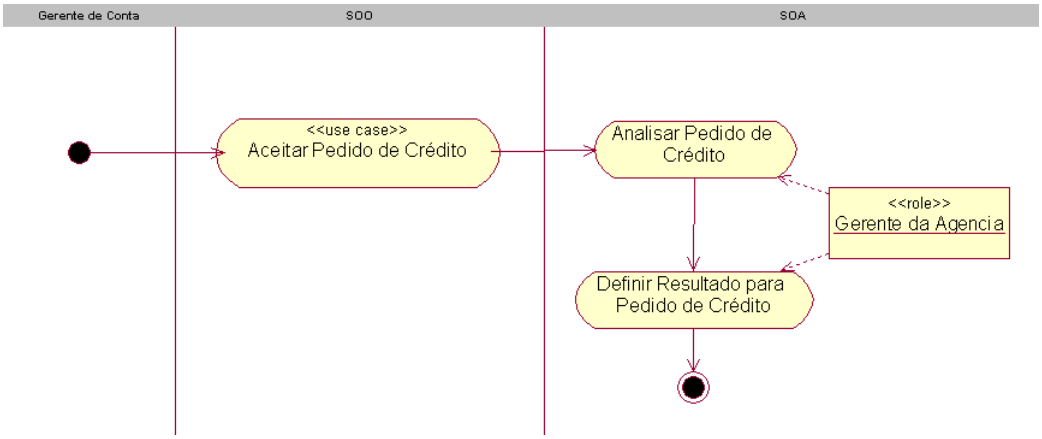


Figura 6.12: Diagrama de Atividades para o caso de uso Homologar Resultado da Análise do Gerente de Conta

pelos papéis de agentes, "Analisar Pedido de Crédito" e "Homologar Resultado da Análise do Gerente de Conta", que pertencem aos casos de uso que são invocados pelo caso de uso "Aceitar Pedido de Crédito". Tem-se as Figuras 6.11 e 6.12 que apresentam o diagrama de atividades dos casos de uso que são executados pelos papéis de agentes. Nestes diagramas de atividade, tem-se que, as atividades "Avaliar Pedido de Crédito", "Definir Resultado da Análise" e "Verificar Necessidade de Homologação" são executadas pelo papel de agente Gerente de Conta, e que, as atividades "Analisar Resultado do Pedido" e "Definir Resultado do Pedido" são executadas pelo papel de agente Gerente de Conta.

Com os diagramas de atividade dos casos de uso de sistema especificados, deve-se realizar o refinamento dos papéis de agentes identificados anteriormente (ou especificar novos papéis que foram identificados durante a etapa de análise). O refinamento dos papéis de agentes passa pela identificação das atividades, atribuições e restrições associadas aos papéis nos diagramas de atividades do sistema que os ref-

erenciam. Segundo Bastos [5], as atribuições são derivadas das atividades dos casos de uso que requerem a participação de papéis. As Figuras 6.13 e 6.14 apresentam as definições dos papéis de agentes Gerente de Conta e Gerente da Agência de acordo com o diagrama de atividades ilustrado pela Figura 6.10.

Role: Gerente de Conta			
Caso de Uso	**Atividade**	**Atribuições**	**Restrições**
Analisar Pedido de Crédito	Avaliar Pedido de Crédito	- Avalia as informações contidas no pedido de crédito	- haver um pedido de crédito a ser analisado
Analisar Pedido de Crédito	Definir Resultado da Análise do Pedido	- Aceita ou rejeita o pedido de crédito	- análise do pedido realizada
Analisar Pedido de Crédito	Verificar necessidade de Homologação	- Verifica se há a necessidade da aprovação de alçada superior	

Figura 6.13: Definição do papel de agente Gerente de Conta

Role: Gerente da Agencia			
Caso de Uso	**Atividade**	**Atribuições**	**Restrições**
Homologar Resultado da Análise do Gerente de Conta	Analisar Pedido de Crédito	- Analisa o pedido de Crédito	- haver um pedido de crédito que necessite de homologação
Homologar Resultado da Análise do Gerente de Conta	Definir Resultado do Pedido	- Define o resultado da análise do pedido de crédito	- O pedido de crédito deve ser aceito ou rejeitado

Figura 6.14: Definição do papel de agente Gerente da agência

De acordo com Bastos [5], um agente é uma agregação de papéis cujas atribuições são complementares. Por atribuições complementares entende-se que os agentes devem mudar seu papel para assumir outra atribuição necessária para uma atividade do caso de uso. Desta forma, no exemplo utilizado para demonstrar a proposta deste trabalho, optou-se por não agregar os papéis Gerente de Conta e Gerente da Agência em um mesmo agente. Apesar desses papéis terem acesso ao pedido de crédito, as extras manipuladas por ambos são diferentes a nível de negócio. As Figuras 6.15 e 6.16 ilustram as especificações dos agentes Gerente de Conta e Gerente da Agência, respectivamente. A especificação da classe de agente segue o padrão utilizado pelo MASUP.

Após a especificação das classes de agentes, se faz necessária a definição dos relacionamentos entre os agentes, considerando os papéis que esses abrangem para a realização dos casos de uso do sistema, formando assim, a chamada sociedade de agentes. Os relacionamento entre as classes de agentes representam canais de comunicação onde mensagens são trocadas [5]. Setas conectando tanto o agente receptor como o agente emissor representam relacionamento entre eles. Nesse diagrama, o nome do relacionamento é opcional e a multiplicidade, representada no limite de

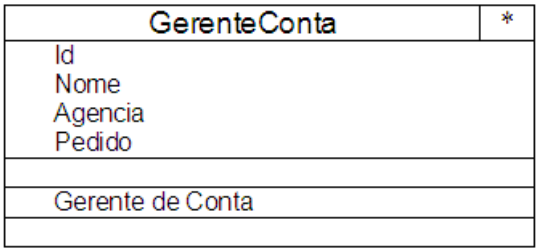


Figura 6.15: Especificação do agente Gerente de Conta

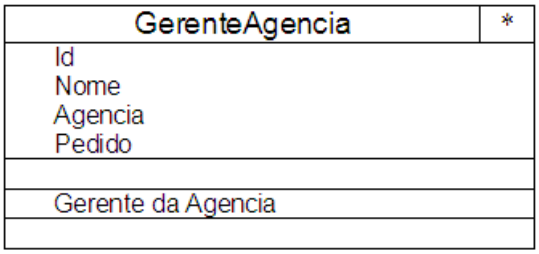


Figura 6.16: Especificação do agente Gerente da Agência

cada seta, pode ser definida da seguinte forma: exata (1), um intervalo (1..5), uma ou mais (1..*) ou várias (0..*).

No MASUP, relações hierárquicas entre os agentes são definidas de duas diferenças formas: relação de autoridade (permanente ou dependente de papel) e relação de comunicação. No relacionamento de autoridade permanente, o agente receptor da mensagem deve necessariamente responder ao pedido solicitado, enquanto no relacionamento dependente de papel, o receptor só responde ao pedido solicitado se o agente emissor estiver executando um determinado papel. No relacionamento de comunicação, o agente receptor tem o poder de decidir se irá ou não atender ao pedido solicitado. A Figura 6.17 ilustra o diagrama contendo a sociedade de agentes para o caso de uso Aceitar Pedido de Crédito. Nesse exemplo, optou-se pelo relacionamento dependente de papel, uma vez que o agente Gerente da Agencia só irá realizar uma análise de um pedido de crédito se esta atividade foi solicitada por um agente executando o papel Gerente de Conta.

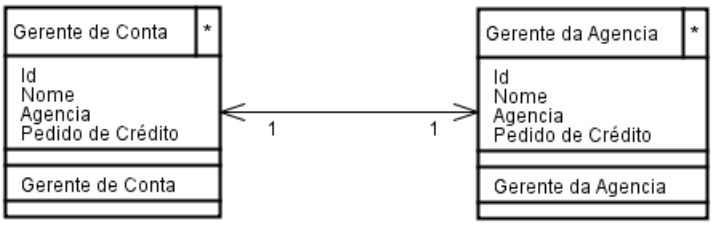


Figura 6.17: Especificação da sociedade de agentes

6.4.2 *Workflow* de Projeto

Como *workflow* de Projeto é subseqüente ao *workflow* de Análise, esse faz uso dos artefatos gerados pela etapa anterior. Tendo que a proposta deste trabalho envolve a identificação de papéis de agentes, outrora realizada no *workflow* de Análise, o *workflow* de Projeto não requer nenhuma mudança em suas atividades, já que estas atividades são realizadas sobre os papéis de agentes bem como a sociedade de agentes já definidos. Isso faz com que o *workflow* de Projeto mantenha a sua estrutura original proposta pelo MASUP.

Nesta etapa devem ser definidos os cenários de interação entre os agentes. Essas interações são descritas por meio Diagramas de Seqüencia Estendidos AUML, que incluem algumas novas propriedades que permitem a caracterização de cenários de comunicação para a solução orientada a agentes. Mais detalhes sobre estas características podem ser encontrados em [5]. A Figura 6.18 ilustra o cenário de interação entre os papéis de agentes Gerente de Conta e Gerente da Agência para o processo de aprovação de pedido de crédito. Nesta figura, uma mensagem do tipo *request* é enviada do agente GerenteConta para o agente GerenteAgencia solicitando a avaliação do pedido de crédito. O agente GerenteAgencia realiza a avaliação do pedido, salva as informações no banco de dados e comunica o agente GerenteConta do término da sua avaliação. Nesta etapa de projeto também devem ser identificados os serviços de infra-estrutura envolvidos no sistema multiagentes, desta forma, na figura é introduzido um objeto *«database-handler»* que provê serviços de acesso para o acesso de informações à um banco de dados.

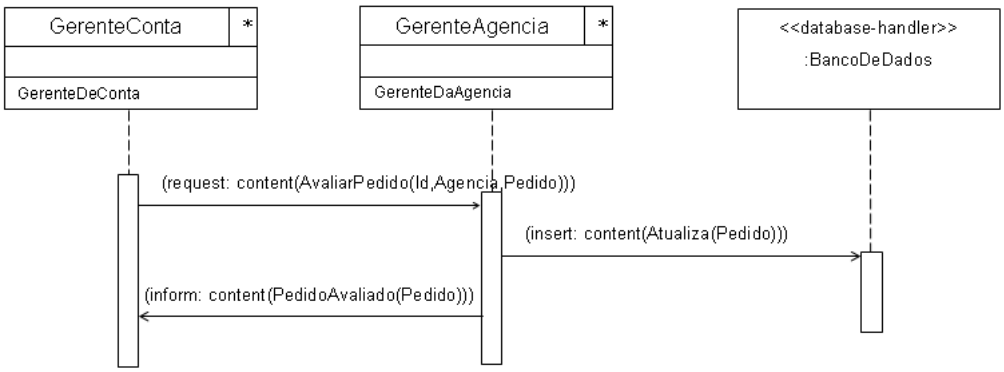


Figura 6.18: Diagrama de Seqüencia Estendido AUML

Com os cenários de comunicação identificados através dos diagramas estendidos de seqüencia AUML, todas as mensagens encontradas devem ser inseridas apropriadamente nas classes de especificações dos agentes. As Figuras 6.19 e 6.20 ilustram as especificações dos agentes GerenteConta e GerenteAgencia com suas respectivas mensagens identificadas anteriormente e ilustradas pela Figura 6.18.

GerenteConta	*
Id Nome Agencia Pedido	
(inform: content(PedidoAvaliado(Pedido)))	
Gerente de Conta	
- Avalia as informações contidas no pedido de crédito - Aceita ou rejeita o pedido de crédito - Verifica se há a necessidade da aprovação de alçada superior	

Figura 6.19: Refinamento da especificação do agente Gerente de Conta

GerenteAgencia	*
Id Nome Agencia Pedido	
(request: content(AvaliarPedido(Id,Agencia,Pedido)))	
Gerente da Agencia	
- Analisar o pedido de Crédito - Definir o resultado da análise do pedido de crédito	

Figura 6.20: Refinamento da especificação do agente Gerente da Agência

6.5 Considerações

Este capítulo apresentou uma proposta de extensão para a metodologia voltada a sistemas multiagentes chamada MASUP. Esta extensão tem sua importância nas suas duas principais características: a introdução da disciplina Modelagem de Negócio proposta pelo RUP no processo de desenvolvimento de SMAs, tornando o MASUP um processo mais completo em relação as outras metodologias estudadas; e um modelo para a identificação de papéis de agentes através da aplicação das características provenientes dos estudos referentes ao processo decisório sobre os elementos da nova disciplina inserida.

Capítulo 7

Protótipo para Identificação de Papéis

"Write to be understood, speak to be heard, read to grow"
— Lawrence Clark Powell

Visando demonstrar a aplicabilidade de forma automatizada da extensão proposta nesta dissertação, foi construído um protótipo, resultado da adaptação de um *software* já existente cujo objetivo é dar suporte a modelagem dos artefatos do MASUP.

7.1 MASUP Modeler

O MASUP Modeler é uma ferramenta CASE desenvolvida utilizando a linguagem de programação java e tem como finalidade dar suporte ao processo de desenvolvimento introduzido pelo MASUP. Essa ferramenta permite a manutenção dos artefatos referentes a modelagem de sistemas orientados a agentes, através da construção de diagramas de casos de uso e de atividades, especificação de papéis de agentes, classes de agentes, entre outros.

A construção da ferramenta de apoio foi realizada sobre o modelo conceitual ilustrado pela Figura 7.1. Esse modelo representa a base da ferramenta, quaisquer implementações extras que visam o aperfeiçoamento da mesma, devem seguir este modelo. Uma breve descrição de cada classe é apresentada abaixo:

- Classe *Model* - classe abstrata que representa os pacotes que podem ser implementados no protótipo.
- Classe *Diagram* - classe abstrata que representa qualquer diagrama disponível no protótipo. Para a criação de um novo diagrama e para que este seja fun-

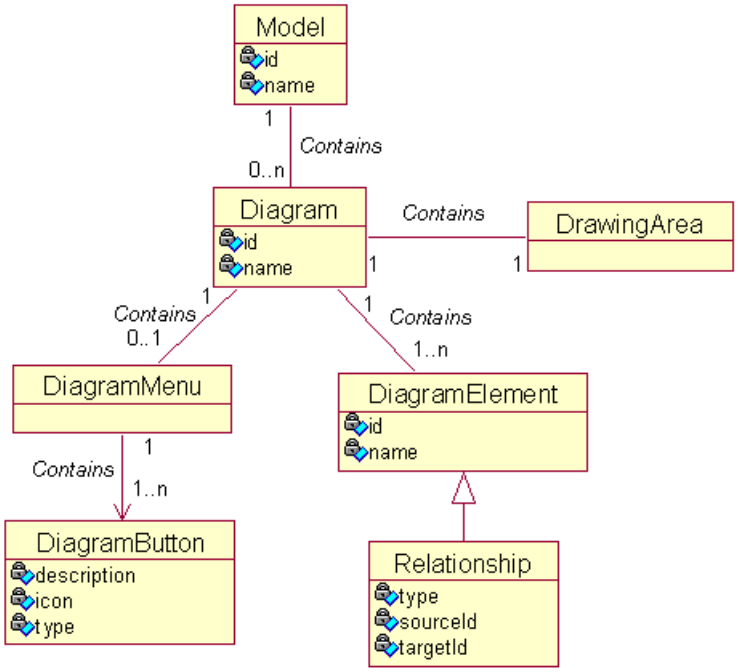


Figura 7.1: Modelo Conceitual utilizado no Masup Modeler

cional, deve-se estender esta classe para que a aplicação reconheça o objeto como um diagrama e possa adicionar as funcionalidades necessárias.

- Classe *DiagramMenu* - classe abstrata que representa um menu para o diagrama.

- Classe *DiagramButton* - classe abstrata que representa um botão a ser adicionado a classe textitDiagramaMenu.

- Classe *DiagramElement* - classe abstrata que representa qualquer elemento a ser inserido em um diagrama. Para que a aplicação consiga adicionar as funcionalidades de um elemento de um diagrama, a classe do objeto precisa estender esta classe.

- Classe *RelationShip* - classe abstrata que representa qualquer tipo de relacionamento entre dois objetos do tipo *DiagramElement*.

- Classe *DrawingArea* - classe abstrata que representa a área disponível de um diagrama para a inserção e manipulação de objetos do tipo *DiagramElement* e *RelationShip*.

A Figura 7.2 apresenta a interface de interação com o usuário do MASUP Modeler. Junto nesta figura, são ilustradas as classes acima descritas, de forma a permitir que o leitor visualize a implementação de cada uma.

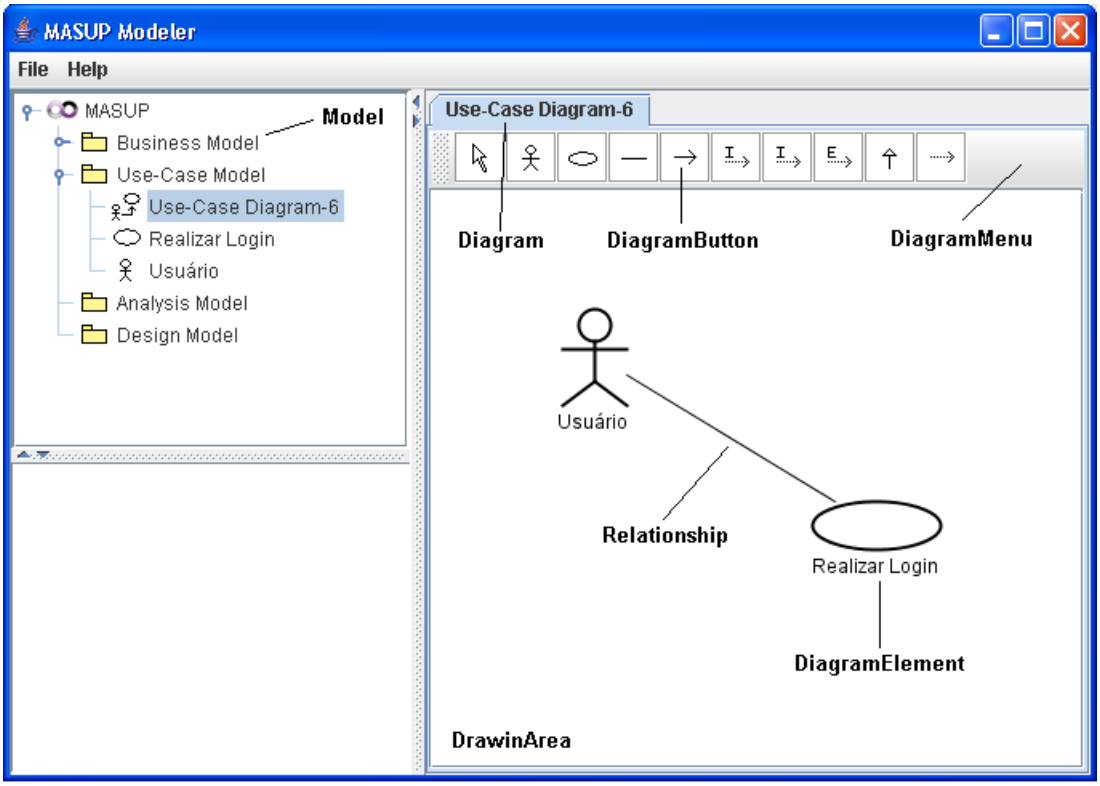


Figura 7.2: Interface com o usuário do MASUP Modeler

Como o MASUP Modeler provê suporte a modelagem de sistemas multi-agentes ao longo do processo proposto pelo MASUP, são disponibilizados para os seus usuários os seguintes artefatos:

- *Diagrama de Caso de Uso* - O diagrama de casos de uso é um diagrama cujo objetivo é representar um conjunto de requisitos, na forma de casos de uso, do sistema que será automatizado.

- *Diagrama de Atividades* - este diagrama apresenta o conjunto de atividades que compreendem o fluxo da execução de um determinado requisito do sistema.

- *Diagrama de Atividades Estendido* - este diagrama surge como uma extensão ao diagrama anterior, uma vez que possui uma *swinlane* para representar o sistema inteligente.

- *Diagrama de Classes de Análise* - este diagrama descreve o conjunto de objetos de um sistema bem como o relacionamento existente entre eles;

- *Especificação de Papel* - este diagrama apresenta a especificação de cada papel de agente, compreendendo suas atividades, restrições, atribuições e casos de uso em que o papel está presente.

- *Diagrama de Classe de Agentes* - este diagrama apresenta as classes de agentes bem como os relacionamento entre as mesmas, o comportamento dos agentes em relação aos mesmos.

7.1.1 Extensões realizadas

Uma vez que o MASUP Modeler não fornecia suporte a extensão proposto neste trabalho, foram necessárias modificações na versão atual da ferramenta, para que esta pudesse servir também aos propósitos deste trabalho, ou seja, apresentar um protótipo na forma de ferramenta que realizasse de forma automatizada a identificação de papéis de agentes. Sendo assim, alguns objetos se fazem necessários em relação ao diagrama apresentado pela Figura 7.1, desta forma, os principais objetos desta extensão realizada são descritos abaixo e ilustrados pela Figura 7.3 (representados com a cor cinza).

- Classe *BusinessModel* - classe que implementa o pacote referente a modelagem de negócio;
- Classe *BusinessUseCaseDiagram* - classe que implementa o diagrama de casos de uso do negócio;
- Classe *BusinessActivityDiagram* - classe que implementa o diagrama de atividades do negócio;
- Classe *BusinessExtActivityDiagram* - classe que implementa o diagrama estendido de atividades do negócio;
- Classe *BusinessRoleDiagram* - classe que implementa o diagrama de papéis do negócio.

Com a extensão do MASUP Modeler adicionando as funcionalidades que dão suporte a identificação de papéis de agentes, os seguintes diagramas passaram a ser disponibilizados no protótipo:

- *Diagrama de Caso de Uso do Negócio* - este diagrama envolve a apresentação dos casos de uso do negócio e a interação destes com os atores do negócio;
- *Diagrama de Atividades do Negócio* - este diagrama representa o fluxo de atividades necessárias para a realização de um caso de uso do negócio;
- *Diagrama de Atividades Estendido do Negócio* - este diagrama é uma extensão ao diagrama anterior. Sua diferença em relação ao Diagrama de Atividades do Negócio constitui na presença de *swinlanes* que ao invés de repre-

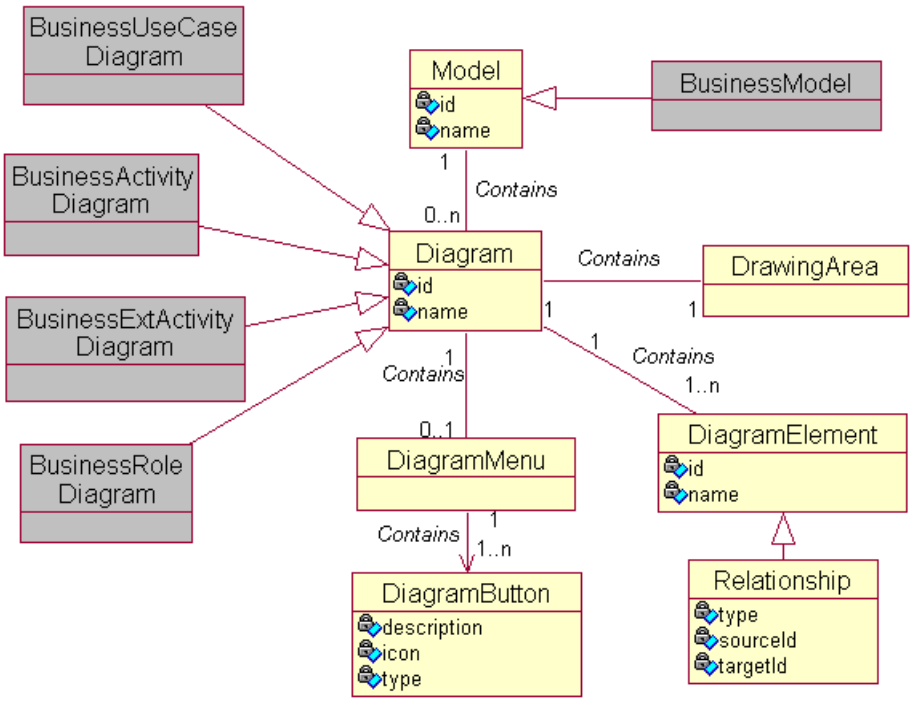


Figura 7.3: Modelo Conceitual utilizado no Masup Modeler com novos objetos

sentarem atores do negócio, representam papéis de agentes que foram identificados através da aplicação do método proposto neste trabalho sobre os diagramas de atividades do negócio;

- *Diagrama de Papéis do Negócio* - este diagrama apresenta graficamente os papéis que foram identificados através do diagrama de atividades do negócio. Neste diagrama são apresentados os papéis bem como as atividades do negócio que cada papel executa dentro de um fluxo de atividades. Junto com o Diagrama de Atividades Estendido do Negócio, este diagrama foi introduzido com a proposta discutida neste trabalho;

A Figura 7.4 apresenta a interface gráfica do MASUP Modeler com os artefatos que foram adicionados como resultado da extensão do protótipo com o método para a identificação de papéis de agentes.

7.2 Exemplo de Uso

Esta seção tem como objetivo principal ilustrar e exemplificar o uso do protótipo no que se refere ao processo de modelagem de negócio e a identificação de papéis de agentes. A demonstração de utilização do protótipo se inicia com a construção dos artefatos para a modelagem de negócio (bem como com a classificação dos elementos pertinentes ao método proposto de acordo com a tabela 6.1) e posteriormente ilustra

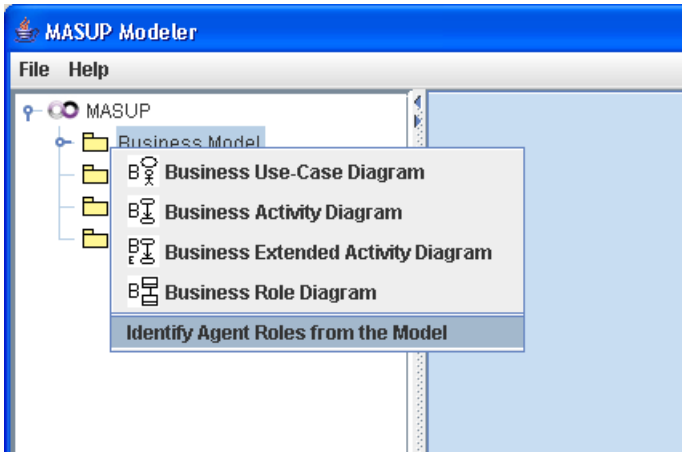


Figura 7.4: Novos elementos adicionados ao protótipo

como realizar de forma automatizada a identificação dos papéis de agentes utilizando os artefatos que foram gerados no passo anterior.

7.2.1 Criando os artefatos básicos

De acordo com a proposta apresentada no capítulo 6, o processo para a construção de um sistema, sendo ele ou não um sistema multiagentes, começa com a modelagem de negócio demonstrando os processos do negócio que poderão ser suportados com o desenvolvimento do novo sistema. Como pode ser visto na ilustração apresentada pela Figura 7.4, o protótipo permite a criação do diagrama de caso de uso do negócio bem como do diagrama de atividades do negócio. A Figura 7.5 apresenta o diagrama de caso de uso do negócio para o mesmo cenário utilizado para a discussão da proposta no Capítulo 6.

Uma vez tendo o diagrama de casos de uso do negócio definido e todos os processos de negócio envolvidos já identificados, devem-se ser criados os diagramas de atividades para cada um dos casos de negócio previamente identificados. A construção do diagrama de atividade do negócio segue as mesmas diretrizes proposta pelo RUP. Com o diagrama de atividades do negócio construído, deve-ser identificar as características cada atividade presente de acordo com o tipo de problema envolvido na atividade e com o tipo de dependência desta atividade com as demais. A Figura 7.6 ilustra o diagrama de atividades para o caso de uso do negócio "Solicitar Empréstimo Financeiro" e como pode ser visto na mesma, ao lado esquerdo inferior da tela com o usuário, é disponibilizado dois menus (*Problem* e *Task*) para que o usuário possa classificar cada atividade. O exemplo ilustrado pela Figura nos mostra a atividade do negócio "Analisar Pedido de Crédito" sendo classificada como sendo um problema semi-estruturado e do tipo seqüencial-interdependente. Todas as atividades dos diagramas de atividade do negócio devem ser classificadas, caso

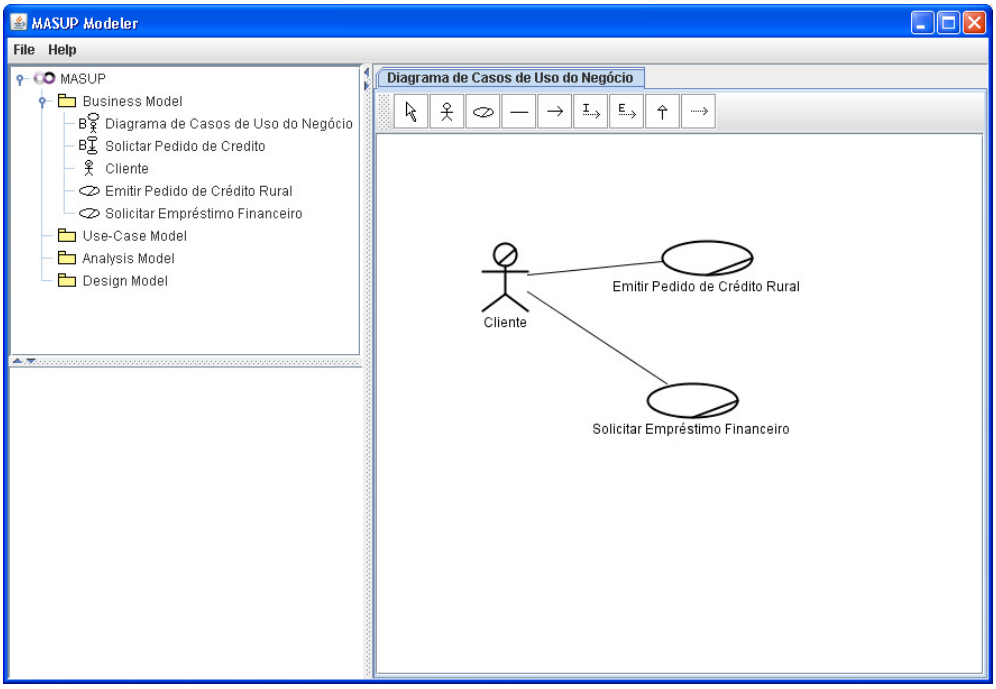


Figura 7.5: Ilustração de um diagrama de casos de uso do negócio

contrário, o protótipo não irá considerá-las no momento da identificação dos papéis de agentes.

7.2.2 Identificando os papéis de agentes

Com a criação dos artefatos apresentados na subseção anterior, pode-se dar início ao processo de identificação dos papéis de agentes. Basicamente, o protótipo realiza uma verificação em todas as atividades do negócio que foram classificadas na etapa anterior, e de acordo com a tabela 6.1 irá proceder com a criação dos papéis de agentes quando necessário.

Na Figura 7.4, que ilustra os novos artefatos adicionados ao protótipo, existe uma opção no menu *Identify Agent Roles from Model* para dar início ao processo de identificação dos papéis de agentes de forma automatizada, o que representa a grande vantagem em ter um protótipo que realize tal tarefa sem a necessidade de interação humana. Entretanto, durante o processo de identificação de papéis, o usuário pode ser questionado quando escolha se deseja que uma atividade seja realizada por um papel de agente ou não. Isso ocorre quando o protótipo identifica que a atividade é do tipo independente, que de acordo com a Tabela 6.1 pode ser implementada pelo paradigma orientado a agentes ou pelo paradigma orientado a objetos, não dependendo do tipo de problema envolvido na atividade. Tem-se na Figura 7.7 um exemplo da janela de diálogo[1] que pede para o usuário escolher se

[1]Janelas de diálogo aparecem sob demanda de uma janela de aplicação. A janela de diálogo

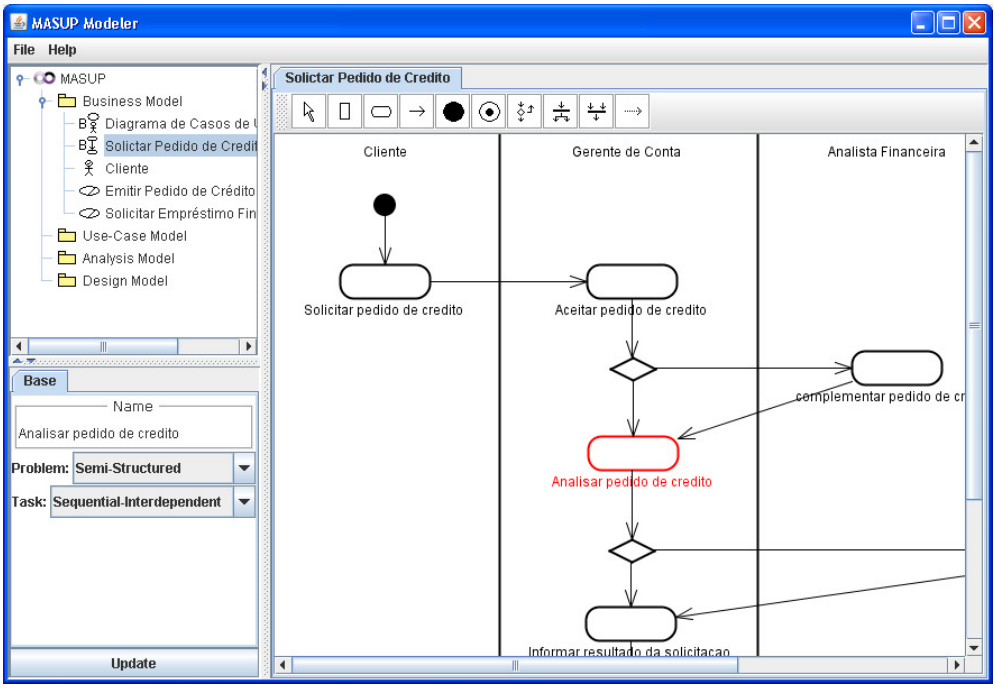


Figura 7.6: Ilustração de um diagrama de atividades do negócio

a atividade "Aceitar Pedido de Crédito" será executada por um papel de agente ou não. Caso o usuário escolha *yes*, o protótipo irá criar um papel de agente para executar tal atividade, caso contrário, o protótipo segue para a próxima atividade no modelo de negócio.

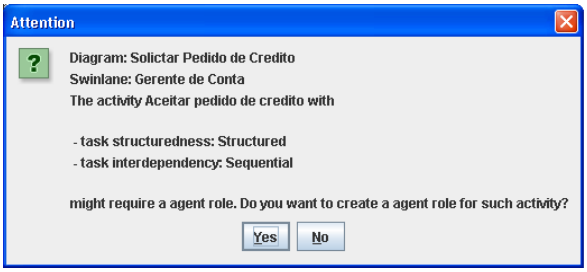


Figura 7.7: Janela de diálogo solicitando a criação do papel de agente

Uma vez que o processo de identificação de papéis chega ao fim, o protótipo gera para o usuário alguns diagramas de acordo com o resultado obtido no processo. O diagrama de papéis do negócio é um diagrama novo introduzido neste trabalho, e tem por objetivo apresentar os papéis que foram identificados bem como as atividades do negócio que cada um se torna responsável, além de representar também os relacionamentos entre cada papel, que passam a representar um processo de comunicação e/ou troca de informações entre os mesmos. A Figura 7.8 ilustra o diagrama gerado pelo protótipo para o cenário em discussão com os dois papéis que foram

pode alertar o usuário sobre um problema, pedir confirmação sobre uma ação, ou solicitar que dados sejam fornecidos.

identificados de acordo com as atividades do negócio.

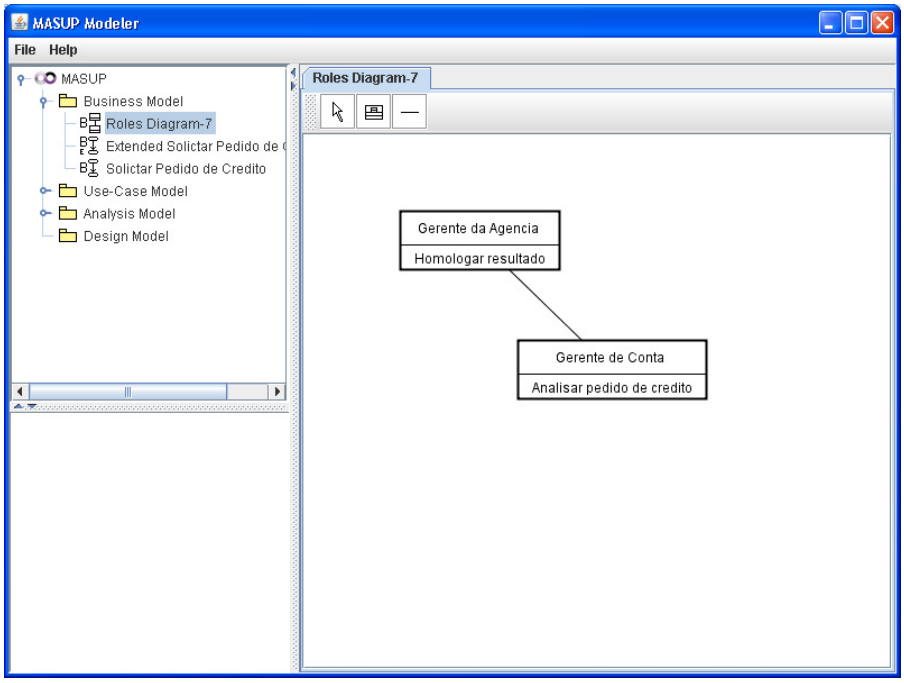


Figura 7.8: Papéis de Agentes identificados

Além do diagrama apresentado anteriormente, o protótipo apresenta ainda uma extensão ao diagrama de atividades do negócio, que pode apresentar uma *swinlane* representando o papel de agentes quando da necessidade de existir uma atividade que tenha sido identificada como passível de ser executada sob o paradigma de orientação a agentes. O protótipo de baseia no diagrama original para a criação do diagrama estendido, realizando apenas uma re-alocação das atividades nas *swinlanes* que foram adicionadas durante a identificação dos papéis. Quando o protótipo identifica que não houve nenhum papel de agente necessário no diagrama de atividades, este não irá realizar a extensão do diagrama original, apenas dos diagramas em que papéis de agentes cuja presença se faz necessária. Tem-se então a Figura 7.9 que ilustra o diagrama de atividades estendido do negócio que foi gerado a partir do diagrama de atividades para o caso de uso "Solicitar Empréstimo Financeiro". Ao analisar o diagrama, percebe-se que foram adicionadas duas novas *swimlanes* utilizando o estereótipo *«role»* para identificar os papéis de agentes envolvidos no fluxo do negócio. Além dessas alterações, percebe-se também que as atividades que deram origem aos papéis foram movidas para as novas *swimlanes*.

Até este momento, o protótipo trabalhou apenas com artefatos referente ao modelo de negócio, entretanto, com os diagramas de atividades do negócio bem como o diagrama dos papéis do negócio, o protótipo vai mais além e já provê outros diagramas para auxiliar o usuário na modelagem e especificação do sistema, sejam eles: diagrama de casos de uso do sistema e especificação de papéis.

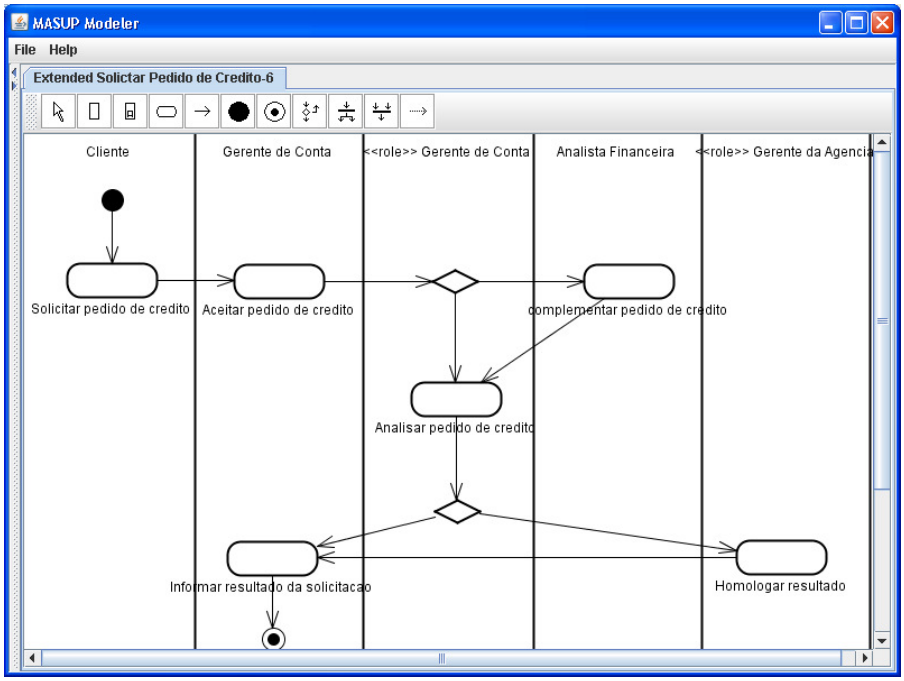


Figura 7.9: Diagrama de atividades estendido do Negócio para o caso de uso "Solicitar Empréstimo Financeiro"

Como apresentado na seção 4.4, o RUP segure um modelo para a transição de requisitos do negócio para requisitos do sistema de uma maneira concisa. Tendo todos os artefatos do modelo de negócio já definidos após o processo de identificação dos papéis de agentes, o protótipo faz uso dessa sugestão proposta pelo RUP e implementa um algoritmo que extrai os casos de uso do sistema através dos diagramas de atividades do negócio, onde temos basicamente a transformação de atividades do negócio em casos de uso do sistema. As atividades do negócio que devem ser executadas por papéis de agentes são apresentadas para o usuário em um novo estereótipo, para diferenciá-lo dos casos de uso de sistema. Na Figura 7.10 temos o diagrama de casos de uso do sistema que foi gerado a partir dos artefatos do negócio apresentados anteriormente, e em detalhes na cor azul, temos os dois casos de uso que serão realizados por agentes inteligentes.

Para finalizar o processo de automatização do método proposto nesse trabalho, o protótipo ainda faz uso do diagrama de papéis do negócio para criar as especificações de papéis de agentes, que pertencem ao modelo de análise. Entretanto vale lembrar que a especificação dos papéis criada é um esboço a partir do modelo de negócio e deve ser posteriormente refinado pelo responsável pela análise ou implementação do sistema. Um exemplo da especificação dos papéis pode ser visto na Figura 7.11, onde os dois papéis que foram encontrados no modelo de negócio são ilustrados.

A existência de um protótipo responsável pela automatização do método

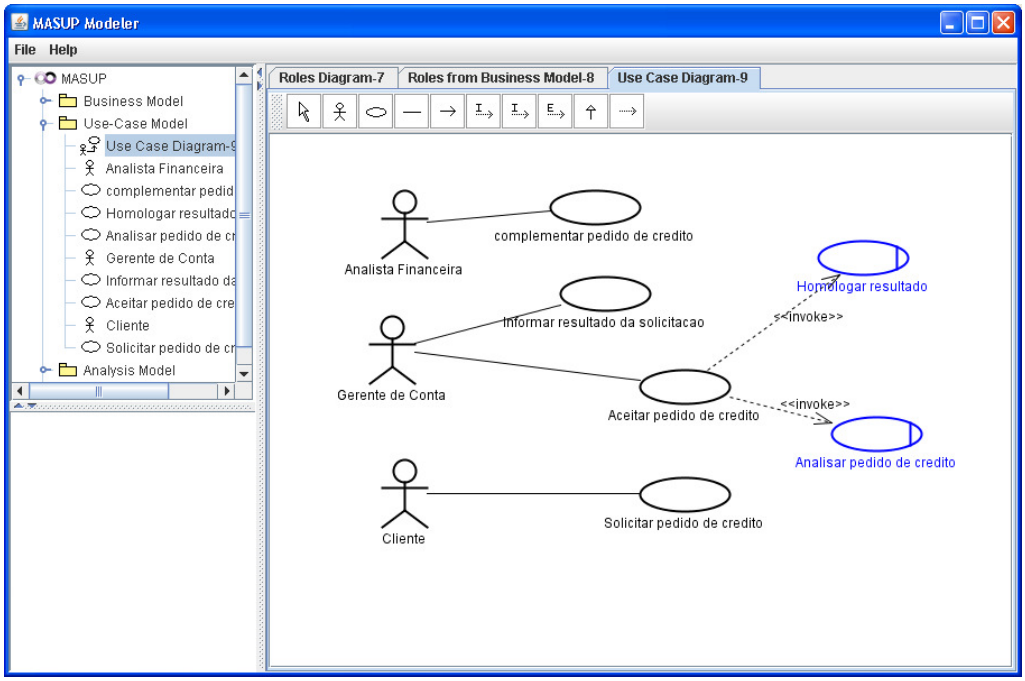


Figura 7.10: Diagrama de Casos de uso do sistema gerado método

de identificação de papéis de agentes é de grande relevância para este trabalho, pois prova que o método é suscetível a automatização na forma de *software* computacional, além de tornar a construção de sistemas multiagentes uma tarefa menos árdua quando da existência de uma modelagem de negócio extremamente extensa, o que tornaria a modelagem de papéis de agentes uma atividade muito cansativa.

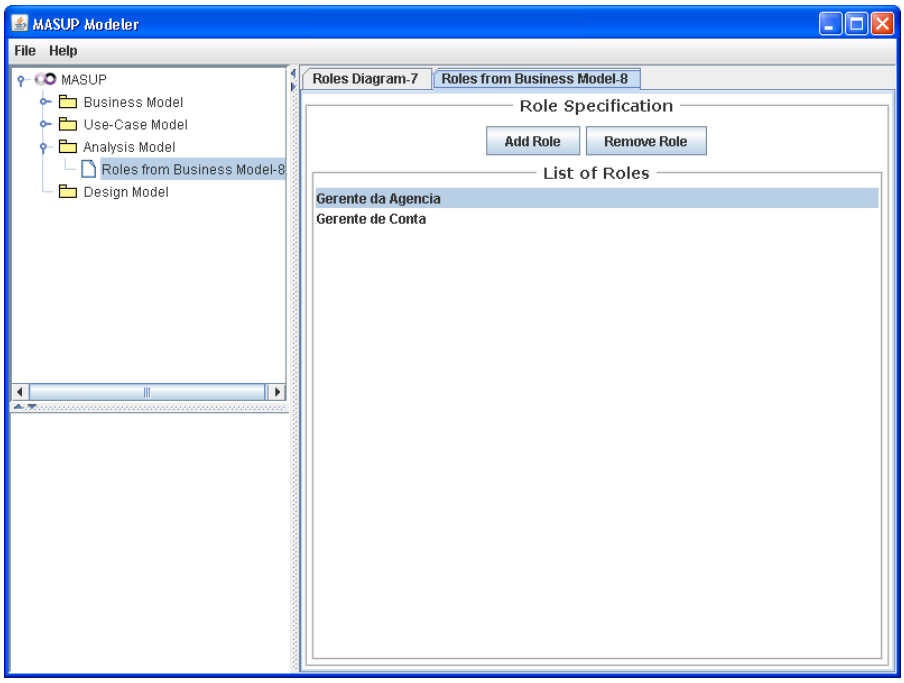


Figura 7.11: Especificação de papéis básica gerada pelo método

Capítulo 8

Considerações Finais

"Não devemos ter medo dos confrontos.
Até os planetas se chocam e do caos nascem as estrelas"
— Charles Chaplin

8.1 Conclusão

Resultado dos esforços da área de Inteligência Artificial e Sistemas de Informação, o campo de estudo referente a Sistemas Multiagentes é uma abordagem recente e tem despertado grande interesse na comunidade acadêmica. A construção de tais sistemas passa a ser uma realidade na medida em que eleva-se o interesse dos pesquisadores por este assunto e começam a surgir propostas para metodologias, linguagens e plataformas que vêm a dar suporte ao desenvolvimento de sistemas orientados a agentes. Contudo, nas propostas existentes podemos identificar lacunas que fazem com que determinadas atividades inerentes ao desenvolvimento de sistemas se tornem uma tarefa subjetiva. Dentre as atividades que se enquadram neste cenário, destacamos o processo de identificação de papéis de agentes.

Também temos que a Modelagem de Negócio passou a ser muito bem vista por organizações devido aos benefícios que esta oferece quando utilizada de forma adequada. Um dos benefícios mais destacados por pesquisadores e profissionais que utilizaram a modelagem de negócio é a qualidade apresentada pelos sistemas que são desenvolvidos partindo de um modelo de negócio, alinhando desta forma, os objetivos da organização com a tecnologia de informação. Além disso, a realização ou utilização da Modelagem de Negócio (não utilizada por nenhuma das metodologias para sistemas multiagentes) é considerada atualmente uma atividade essencial para aquelas empresas que desejam ter seus processos de negócio bem delimitados e com escopo bem definido.

Dentro desse contexto, este trabalho apresentou uma proposta de um método

para a identificação de papéis de agentes integrada à já existente metodologia para sistemas multiagentes chamada MASUP. O processo de identificação de papéis proposto compreende uma extensão realizada sobre o MASUP, com a inclusão da disciplina Modelagem de Negócios proposta pelo RUP, e com características provenientes da área de Processo Decisório; fazendo com que a atividade de identificação de papéis passe a ser executada nessa disciplina, uma vez que, anteriormente a este trabalho, essa atividade de identificação de papéis era realizada na disciplina de Análise do MASUP.

Através do estudo realizado neste trabalho e da proposta introduzida por este, a identificação de papéis de agentes passa a decorrer de uma análise dos diagramas de atividades da modelagem de negócio, buscando mapear as atividades dentro das características apresentadas no estudo referente ao Processo Decisório. Tais características são sob dois pontos de vista: sob o ponto de vista referente ao grau de estruturação do problema envolvido na atividade, podendo ser descrito como não-estruturado, semi-estruturado e estruturado; sob o ponto de vista relativo ao grau de dependência da atividade com outras dentro do fluxo de execução do processo do negócio, onde encontramos atividades que são independentes, seqüenciais-interdependentes e agregado-interdependentes.

Tendo as atividades mapeadas conforme suas características, é indicada a melhor solução computacional que venha a dar suporte ao processo de negócio que esteja sendo trabalhado. Caso a solução computacional resultante seja a orientação a agentes, o fluxo do desenvolvimento se dá com o MASUP, seguindo todas as suas atividades subseqüentes a disciplina de Modelagem de Negócio. Caso a solução computacional identificada para o problema seja a orientação a objeto, ou seja, nenhum papel de agentes foi identificado, o fluxo do desenvolvimento do sistema segue as atividades do RUP.

8.2 Contribuições

A proposta introduzida neste trabalho para a identificação de papéis de agentes através da modelagem de negócio visa contribuir para os estudos realizados na área de engenharia de software para sistemas multiagentes, propondo uma extensão que busca preencher uma lacuna existente no que diz respeito a identificação de papéis de agentes.

Com o estudo realizado durante este trabalhado, destacam-se as seguintes contribuições:

- a criação de uma extensão mais objetiva para a identificação de papéis de agentes durante o processo de desenvolvimento de sistemas multiagentes;

- a utilização da modelagem de negócio como ponto de partida para o desenvolvimento de sistemas utilizando o MASUP, tendo nela, uma forma de definir o paradigma de desenvolvimento a ser aplicado na construção dos aplicativos que darão suporte aos processos do negócio;

- a criação de novos artefatos no modelo de negócio que buscam tornar a modelagem dos sistemas multiagentes mais clara.

8.3 Trabalhos Futuros

A continuidade deste trabalho indica uma contribuição para a área de desenvolvimento de sistemas, com o intuito de avançar as pesquisas referentes a sistemas multiagentes. A área que abrange o estudo de sistemas multiagentes está em constante crescimento, e referente ao que foi estudado neste trabalho, são identificados como trabalhos futuros os seguintes ítens abaixo:

- A identificação dos papéis de agentes é realizada sobre as atividades executadas pelo trabalhador do negócio, buscando encontrar tomadas de decisão que estão inerentes aos processos de negócios modelados. Entretanto, a modelagem de negócio pode ser estendida afim de apoiar também as decisões dos atores do negócio, desta forma, papéis de agentes também podem ser identificados através desta extensão.

- A aplicação da extensão proposta deste trabalho bem como do protótipo em casos reais com o intuito de encontrar pontos de melhorias em ambos, uma vez que os exemplos utilizados para a construção da proposta e validação do protótipo são poucos.

- A construção da extensão foi especificamente realizada sobre o MASUP, entretanto, esta mesma pode ser adaptada para que seja mais generalizada e possa permitir sua utilização em outras metodologias para sistemas multiagentes. Logo, um estudo direcionado neste sentido, traria grande valor para esta proposta, visando buscar uma maior aplicabilidade deste trabalho dentro da área de sistemas multiagentes.

- O processo decisório pode envolver inúmeras variáveis de acordo com o ambiente no qual o processo de decisão está inserido. Avaliar e identificar outras características que podem afetar o processo de decisão torna-se um tópico interessante a ser estudado, uma vez que, o ambiente no qual a decisão a ser tomada está inserida pode afetar o resultado esperando.

Referências Bibliográficas

[1] Agre, P. E., and Chapman, D. What are plans for? Tech. rep., Cambridge, MA, USA, 1988.

[2] Amandí, A. *Programação de Agentes Orientada a Objetos*. Tese de Doutorado, Universidade Federal do Rio Grande do Sul, Porto Alegre, 1997.

[3] Anthony, R. N. *Planning and Control Systems: A Framework Analysis*, vol. 1. Harvard University Press, Boston, 1965.

[4] Barros, O. *Sistemas de Información Administrativos*. Universidade de Chile, 1976.

[5] Bastos, R. M., and Ribeiro, M. B. Modeling agent-oriented information systems for business processes. In *Third International Workshop on Software Engineering for Large-Scale Multi-Agent Systems in the 26th International Conference on Software Engineering* (Edimburgo, Escócia, Fevereiro 2004), pp. 90–97.

[6] Bastos, R. M., and Ribeiro, M. B. MASUP: An agent-oriented modeling process for information systems. In *Software Engineering for Multi-Agent Systems III: Research Issues and Practical Applications* (Berlim, Alemanha, 2005), pp. 19–35.

[7] Belmonte, M.-V., de-la Cruz, J. L. P., Triguero, F., and Fernández, A. Agent coordination for bus fleet management. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing* (New York, NY, USA, 2005), ACM, pp. 462–466.

[8] Buhler, P. A., and Vidal, J. M. Towards adaptive workflow enactment using multiagent systems. *Inf. Technol. and Management 6*, 1 (2005), 61–87.

[9] Cammarata, S., McArthur, D., and Steeb, R. Strategies of cooperation in distributed problem solving. In *Readings in Distributed Artificial Intelligence*, A. H. Bond and L. Gasser, Eds. Kaufmann, San Mateo, CA, 1988, pp. 102–105.

[10] Chung, L., Nixon, B., Yu, E., and Mylopoulos, J. *Non-Functional Requirements in Software Engineering*, vol. 5 of *International Series in Software Engineering*. Kluwer Publishing, 2000.

[11] Corporation, R. S. Rational unified process, 2003.

[12] Correa, M. *A Arquitetura de Diálogo entre Agentes Cognitivos Distribuídos*. Tese de Doutorado, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 1994.

[13] D'Amico, C. B. Inteligência artificial: uma abordagem de agentes. Tech. rep., CPGCC da UFRGS, 1995.

[14] DeLoach, S. A. Multiagent systems engineering: a methodology and language for designing agent systems. In *Proceedings of Agent Oriented Information Systems '99 (AOIS'99)* (Seattle WA, May 1999), pp. 45–57.

[15] DeLoach, S. A. The MaSE methodology. In *Methodologies and Software Engineering for Agent Systems. The Agent-Oriented Software Engineering Handbook Series : Multiagent Systems, Artificial Societies, and Simulated Organizations* (2004).

[16] DeLoach, S. A., Wood, M. F., and Sparkman, C. H. Multiagent systems engineering. *The International Journal of Software Engineering and Knowledge Engineering 11*, 3 (Junho 2001), 231–258.

[17] Dimeas, A., and Hatziargyriou, N. D. Operation of a multiagent system for microgrid control. *IEEE Transactions on Power Systems 20*, 3 (August 2006), 1447–1455.

[18] Durfee, E. H. Planning in distributed artificial intelligence. In *Foundations of distributed artificial intelligence* (New York, NY, USA, 1996), John Wiley & Sons, Inc., pp. 231–245.

[19] English, A. V. Business modeling with uml: Understanding the similarities and differences between business use cases and system use cases. http://www.ibm.com/developerworks/rational/library/apr07/english, Abril 2007. Último acesso 10/06/2008.

[20] Eriksson, H., and Penker, M. *Business Modeling With UML: Business Patterns at Work*. John Wiley & Sons, Inc., New York, NY, USA, 1998.

[21] Fowler, M. Describing and comparing object-oriented analysis and design methods. *Object development methods* (1994), 79–109.

[22] Giorgini, P., Perini, A., Mylopoulos, J., Giunchiglia, F., and Bresciani, P. Agent-oriented software development: A case study. In *Proceendings of the 13th International Conference on Software Engineering and Knowledge Engineering (SEKE01)* (Buenos Aires, Argentina, Junho 2001), Springer, pp. 13–15.

[23] Giunchiglia, F., Mylopoulos, J., and Perini, A. The tropos software development methodology: processes, models and diagrams. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems* (New York, NY, USA, 2002), ACM, pp. 35–36.

[24] Gregg, D. G., and Walczak, S. Auction advisor: an agent-based online-auction decision support system. *Decision Support Systems 41*, 2 (2006), 449–471.

[25] Hackathorn, R. D. Modeling unstructured decision making. *SIGMIS Database 8*, 3 (1977), 41–42.

[26] Hackathorn, R. D., and Keen, P. G. W. Organizational strategies for personal computing in decision support systems. *MIS Quarterly 5*, 3 (September 1981), 21–27.

[27] Hampton, D. R. *Administração: processos administrativos.* McGraw-Hill, São Paulo, 1990.

[28] Heumann, J. Introduction to business modeling using the unified modeling language (uml). *The Rational Edge* (Novembro 2003).

[29] Huhns, M. N., and Stephens, L. M. Multiagent systems and societies of agents. 79–120.

[30] IEEE. Ieee recommended practice for software requirements specifications, ieee std 830-1998, new york, 1998.

[31] Jaelson Castro, Manuel Kolp, J. M. A social organization perspective on software architectures. In *roceedings of theFirst International Workshop From Software Requirements to Architectures (STRAW 01) at ICSE 2001* (Toronto, Canada, 14 May 2001).

[32] Jennings, N. R. Coordination techniques for distributed artificial intelligence. In *Foundations of distributed artificial intelligence* (New York, NY, USA, 1996), John Wiley & Sons, Inc., pp. 187–210.

[33] Jennings, N. R., Sycara, K., and Wooldridge, M. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems 1*, 1 (1998), 7–38.

[34] Kendall, E. A. Agent roles and role models: New abstractions for multiagent system analysis and design. International Workshop on Intelligent Agents in Information and Process Management, September 1998. Germany.

[35] Kendall, E. A. Role model designs and implementations with aspect-oriented programming. *SIGPLAN Not. 34*, 10 (1999), 353–369.

[36] Kendall, E. A. Role models - patterns of agent system analysis and design. *BT Technology Journal 17*, 4 (1999), 46–57.

[37] Kim, M., Lee, S., Park, I., Kim, J., and Park, S. Agent-oriented software modeling. In *APSEC '99: Proceedings of the Sixth Asia Pacific Software Engineering Conference* (Washington, DC, USA, 1999), IEEE Computer Society, p. 318.

[38] Kroll, P., and Kruchten, P. *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[39] Kruchten, P. *The Rational Unified Process: an introduction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[40] Li, S. Agentstra: an internet-based multi-agent intelligent system for strategic decision-making. *Expert Systems Applicatons 33*, 3 (2007), 565–571.

[41] Lopez, B., Urra, P., and Abelló, X. Multi-agent resource allocation for road passenger transportation. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems* (New York, NY, USA, 2005), ACM, pp. 52–59.

[42] Lux, A., and Steiner, D. Understanding cooperation: an agent's perspective. In *Readings in agents* (San Francisco, CA, USA, 1998), Morgan Kaufmann Publishers Inc., pp. 471–478.

[43] Marshall, C. *Enterprise modeling with UML: designing successful software through business analysis*. Addison-Wesley Longman Ltd., Essex, UK, 2000.

[44] Moulin, B., and Chaib-draa, B. An overview of distributed artificial intelligence. In *Foundations of distributed artificial intelligence* (New York, NY, USA, 1996), John Wiley & Sons, Inc., pp. 3–55.

[45] Odell, J. Objects and agents compared. *Journal of Object Technology 1* (May-June 2002), 41–53.

[46] Oliveira, E., Fischer, K., and Stepankova, O. Multi-agent systems: which research for which applications. *Robotics and Autonomous Systems 27* (1999), 91–106.

[47] Padgham, L., and Winikoff, M. Prometheus: a methodology for developing intelligent agents. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems* (New York, NY, USA, 2002), ACM, pp. 37–38.

[48] Paraiso, E. C. Concepção e implementação de um sistema multi-agentes para monitoração e controle de processos industriais. Dissertação de Mestrado, Centro Federal de Educação Tecnologócia do Paraná - CEFET-PR, Janeiro 1997.

[49] Park, S., and Sugumaran, V. Designing multi-agent systems: a framework and application. *Expert Syst. Appl. 28*, 2 (2005), 259–271.

[50] Pereira, M. J. L. B., and Fonseca, J. G. M. *Faces da decisão: as mudanças de paradigmas e o poder da decisão*. Makron Books, São Paulo, 1997.

[51] Perini, A., Bresciani, P., Giorgini, P., Giunchiglia, F., and Mylopoulos, J. Towards an agent oriented approach to software engineering. In *Workshop "From Objects to Agents", Evolutive Trends of Software Systems* (Modena, Italia, Setembro 2001), pp. 74–79.

[52] Perotoni, R. Alocação dinâmica baseada em sistemas multiagentes. Dissertação de Mestrado, Programa de Pós Graduação em Ciência da Computação, Faculdade Informática, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2003.

[53] Power, D. J. A brief history of decision support systems. http://dssresources.com/history/dsshistory.html, 1997. Último acesso 14/02/2008.

[54] Qiao, B., Liu, K., and Guy, C. A multi-agent system for building control. In *IAT '06: Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology* (Washington, DC, USA, 2006), IEEE Computer Society, pp. 653–659.

[55] Ralph, H. Sprague Jr. e Watson, H. J. *Sistemas de Apoio a Decisão; Colocando a Teoria em Prática*, 2ª edição ed., vol. 1. Editora Campus, Rio de Janeiro, 1991.

[56] Ralph H. Sprague, J., and Carlson, E. D. *Building Effective Decision Support Systems*. Prentice Hall Professional Technical Reference, 1982.

[57] Rosenschein, J. S., and Zlotkin, G. Designing conventions for automated negotiation. *AI Mag. 15*, 3 (1994), 29–46.

[58] Russell, S., and Norvig, P. *Artificial Intelligence: A Modern Approach*, second ed. Prentice Hall, New Jersey, US, 2003.

[59] Sánchez, F. G., Valencia-García, R., and Martínez-Béjar, R. An integrated approach for developing e-commerce applications. *Expert Systems Applications 28*, 2 (2005), 223–235.

[60] Santander, V., and Castro, J. Deriving use cases from organizational modeling. In *RE '02: Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering* (Washington, DC, USA, 2002), IEEE Computer Society, pp. 32–42.

[61] Seghrouchni, A. E. F. Rational agent cooperation through concurrent plan coordination. In *Proceedings of the Iberoamerican Workshop on Artificial Intelligence and Multiagent Systems* (Lania, Mexico, 1996), Mia Universidad Veracruzana, pp. 162–171.

[62] Shimizu, T. *Decisão nas Organizações: Introdução aos problemas de decisão encontrados nas organizações e nos sistemas de apoio a decisão*, vol. 1. Editora Atlas, São Paulo, 2001.

[63] Shoham, Y. Agent oriented programming. In *International Conference Logic at Work on Knowledge Representation and Reasoning Under Uncertainty, Logic at Work* (London, UK, 1994), Springer-Verlag, pp. 123–129.

[64] Simon, H. A. *The New Science of Management Decision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1977.

[65] Vidal, J. M., Buhler, P., and Stahl, C. Multiagent systems with workflows. *IEEE Internet Computing 8*, 1 (2004), 76–82.

[66] Voos, H. Resource allocation in continuous production using market-based multi-agent systems. *Industrial Informatics, 2007 5th IEEE International Conference on 2* (2007), 68–75.

[67] Wan, H., Li, K., and Wong, K. Multi-agent application of substation protection coordination with distributed generators. *European Transactions on Electrical Power 16*, 5 (2006), 495–506.

[68] Winikoff, M., and Padgham, L. The prometheus methodology. In *Methodologies and Software Engineering for Agent Systems* (April 2004), Springer US, pp. 217–234.

[69] Wood, M., and DeLoach, S. A. An overview of the multiagent systems engineering methodology. In *Agent-Oriented Software Engineering: First International Workshop, AOSE 2000* (Limerick, Irlanda, Junho 2000), M. W. P. Ciancarini, Ed., Springer, pp. 1–53.

[70] Wooldridge, M., and Ciancarini, P. Agent-oriented software engineering: The state of the art. In *First Int. Workshop on Agent-Oriented Software Engineering (AOSE'00)* (Limerick, Ireland, 2000), P. Ciancarini and M. Wooldridge, Eds., Springer-Verlag, Berlin, pp. 1–28.

[71] Wooldridge, M., and Jennings, N. R. Intelligent agents: Theory and practice. *Knowledge Engineering Review 10*, 2 (1995), 115–152.

[72] Wooldridge, M., Jennings, N. R., and Kinny, D. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems 3*, 3 (2000), 285–312.

[73] Wooldridge, M. J. *Introduction to Multiagent Systems*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[74] Wulfhorst, R. D., Nakayama, L., and Vicari, R. M. A multiagent approach for musical interactive systems. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems* (New York, NY, USA, 2003), ACM, pp. 584–591.

[75] Yan, Y., Kuphal, T., and rgen Bode, J. Application of multiagent systems in project management. *International Journal of Production Economics 68* (2000), 60–68.

[76] Zambonelli, F., Jennings, N. R., and Wooldridge, M. Organizational abstractions for the analysis and design of multi-agent system. In *First international workshop, AOSE 2000 on Agent-oriented software engineering* (Secaucus, NJ, USA, 2001), Springer-Verlag New York, Inc., pp. 235–251.

[77] Zambonelli, F., Jennings, N. R., and Wooldridge, M. Developing multiagent systems: The gaia methodology. *ACM Transactions on Software Engineering and Methodology 12*, 3 (2003), 317–370.