# Uma Abordagem para a Transformação Automática do Modelo de Negócio em Modelo de Requisitos

Felipe Dias, Gisele Morgado, Pedro Oscar, Denis Silveira,
Antonio Juarez Alencar, Priscila Lima, Eber Schmitz
*NCE – IM – Universidade Federal do Rio de Janeiro*
geti@dcc.ufrj.br

## Abstract

*Este artigo apresenta uma abordagem que visa facilitar a especificação de requisitos, melhorando e acelerando o desenvolvimento de sistemas de informação. Tal abordagem apresenta uma técnica para o mapeamento de processos em casos de uso e para o mapeamento de termos em classes de domínio. Esta técnica é acompanhada de uma ferramenta que implementa este mapeamento. Assim, é possível a transformação do Modelo de Negócio em Modelo de Requisitos de forma automática, diminuindo o tempo gasto na especificação do sistema de informação e diminuindo também os erros cometidos na fase especificação de requisitos.*

## 1. Introdução

Todas as organizações fazem uso de algum tipo de sistema de informação. Podemos dizer que a tecnologia da informação é parte integrante de uma organização e que os seus negócios definem os requisitos de um sistema de informação. Mas, apesar de todos os recentes avanços tecnológicos, modelar e implementar sistemas de informação ainda é uma atividade árdua e nem sempre bem sucedida.

Assim, freqüentemente as organizações se encontram insatisfeitas com a qualidade dos seus sistemas. Neste sentido, os problemas mais comuns são: a incapacidade destes sistemas de oferecer um suporte eficiente e efetivo às operações do negócio, a dificuldade de manutenção dos sistemas, a deficiência na integração com os outros sistemas da organização e, conseqüentemente, a falta de confiança das pessoas da organização nestes sistemas.

Visando solucionar estes problemas, o OMG[1] desenvolveu o MDA (*Model Driven Architeture*) [10], cujo objetivo é separar o que o sistema necessita fazer de como o sistema pode fazer. Para atingir este objetivo, o MDA divide o desenvolvimento em três modelos diferentes: o modelo independente de computação CIM (*Computation Independent Model*), o modelo independente de plataforma PIM (*Plataform Independent Model*) e o modelo específico de plataforma PSM (*Plataform Specific Model*).

Entre cada um destes modelos há uma série de transformações que são aplicadas para que se possa chegar até uma plataforma específica. Estas transformações constituem um dos pontos chaves do MDA e muitos trabalhos têm sido realizados no sentido de aperfeiçoá-las e automatizá-las [10]. A grande maioria destes trabalhos, porém, trata apenas das transformações entre o modelo independente de plataforma PIM e o modelo específico de plataforma PSM, deixando em segundo plano as transformações que envolvem o modelo independente de computação CIM. Consideramos, entretanto, que as transformações que envolvem o modelo CIM são de fundamental importância, pois é através deste modelo que se obtém um entendimento adequado do negócio que utilizará o sistema de informação que será implementado [6]. Portanto, é a partir da especificação do negócio feita no CIM que devem ser determinados os requisitos dos sistemas de informação que constituem parte do PIM.

Apresentamos neste artigo uma abordagem, devidamente acompanhada de técnicas e ferramentas, que torna mais natural e confiável a transformação do Modelo de Negócio em Modelo de Requisitos de um sistema de informação. O restante deste texto encontra-se organizado da seguinte forma. Na segunda seção apresentamos os principais conceitos relacionados a este trabalho: MDA, Modelo de Negócio e Modelo de Requisitos. Na terceira seção introduzimos a técnica utilizada na transformação do Modelo de Negócios em Modelo de Requisitos. Na quarta seção apresentamos uma ferramenta que permite automatizar a aplicação desta técnica. E, finalmente, na quinta seção relatamos as conclusões deste texto e as perspectivas de trabalho futuro.

## 2. Contextualização

Antes de descrever detalhes mais específicos do trabalho, alguns conceitos envolvidos necessitam ser discutidos. As seções 2.1, 2.2 e 2.3 apresentam resumidamente estes conceitos.

[1] *Object Management Group*. Grupo para produção e manutenção de especificações para a indústria da computação. Disponível em www.omg.org .

## 2.1. MDA (*Model Driven Architecture*)

O MDA é um arcabouço para desenvolvimento de sistemas. Ele visa separar as decisões orientadas ao negócio das decisões de plataforma permitindo, assim, maior flexibilidade durante as fases de especificação e de desenvolvimento de sistemas [10]. No MDA são utilizados diferentes modelos de diferentes níveis de abstração: o CIM, o PIM e o PSM.

O CIM é o modelo de mais alto nível do sistema, representando as possibilidades e funcionalidades deste que não dependem de computação. Este modelo é importante para o entendimento do problema e serve como fonte do vocabulário a ser usado nos demais modelos. O PIM descreve o sistema, porém não apresenta os detalhes da tecnologia que será usada na implementação. Este modelo oferece especificações formais da estrutura e funcionalidade do sistema, abstraindo-se de detalhes técnicos. O PSM combina a especificação do modelo PIM com detalhes de uma determinada plataforma. Este modelo é o de mais baixo nível de abstração e os seus elementos estão prontos para a geração de código [1].

O documento do OMG que explica o MDA, encontrado em [10], não explicita como construir os modelos CIM, PIM e PSM. No entanto, devido à descrição presente neste documento, podemos dizer que o CIM corresponde ao modelo de negócio e que parte do PIM corresponde ao Modelo de Requisitos de um sistema de informação. Nas seções a seguir apresentaremos estes dois modelos e a forma utilizada neste trabalho para representá-los.

## 2.2. Modelo de Negócio

Um modelo do negócio é uma abstração do funcionamento do próprio negócio. De acordo com [5], um negócio possui os seguintes componentes: objetivos, recursos, processos e regras. Os objetivos são os propósitos do negócio, ou simplesmente, os resultados que toda a organização deseja atingir. Os recursos constituem os objetos utilizados em um negócio, tais como pessoa, material, informação ou produto. Já os processos constituem um conjunto de atividades estruturadas para que um produto (bem ou serviço) seja gerado. E, finalmente, as regras são declarações que restringem, derivam e fornecem condições de existência, representando o conhecimento do negócio.

Embora todos estes objetos sejam fundamentais para o entendimento do negócio que está sendo modelado, sob o ponto de vista do desenvolvimento de sistemas de informação, podemos dizer que os objetos mais importantes são os processos e as regras de negócio. É a partir dos processos e das regras que são definidos os requisitos dos sistemas de informação que apoiarão as operações de um determinado negócio [7], como será apresentado em mais detalhes na terceira seção.

Para representar os processos utilizaremos neste trabalho o diagrama de atividades da UML (Unified Modeling Language) [11]. O diagrama de atividades constitui um elemento de modelagem simples, porém eficaz para descrever fluxo de trabalho numa organização. Entre as suas características, destaca-se a possibilidade de representação de atividades concorrentes, com execuções em paralelo, diferenciando-o de um mero fluxograma. Neste diagrama, as atividades são situadas em raias de responsabilidade, representativas dos locais onde elas são executadas. Nestes locais existem recursos que interagem com as atividades durante sua execução. São representados, também, os insumos consumidos, recursos utilizados e produtos gerados na forma de objetos relacionados às atividades [2].

Para descrever as regras de negócio utilizaremos, seguindo a linha adotada por [3], um subconjunto da língua portuguesa, ao qual chamamos de Português Estruturado. Esta forma de representação estruturada tem, por um lado, a aparência da linguagem natural e, por outro lado, é uma representação consistente e não ambígua do conhecimento.

A representação do Português Estruturado baseia-se no uso de modelos de sentença como forma de viabilizar a representação das regras de negócio. Um modelo de sentença é uma seqüência padronizada de termos usados para construir regras de negócio. Cada categoria de regra deve ter um modelo de sentença particular característico. Segundo o modelo em [3], as regras podem ser agrupadas em seis categorias: termos, fatos, cálculos, derivações, restrições e habilitações de ação.

Neste trabalho utilizaremos apenas os termos e os fatos. Os termos constituem os elementos básicos da linguagem utilizada para expressar as regras de negócio, onde a própria definição de um termo é considerada como uma regra. Para definir os termos utilizamos regras do tipo fato. Os fatos descrevem a natureza ou estrutura operacional de um negócio, relacionando os termos uns aos outros. Na tabela 1 apresentamos os modelos de sentença relacionados à definição dos termos.

**Tabela 1. Modelos de Sentença Utilizados na Definição dos Termos**

| Modelo de Sentença |
|---|
| <termo1> É SUBTIPO DE <termo2> |
| <termo1> TEM COMO ATRIBUTO <termo2> |
| <termo> POSSUI COMO DOMÍNIO <domínio> |
| <termo1> TEM COMO PARTE <termo2> |
| <termo1> ESTÁ RELACIONADO A <termo2> POR <verbo_associação> [COM GRAU <M>..<N>] |

### 2.3. Modelo de Requisitos

Assim como na modelagem de negócio, utilizamos um modelo para representar os componentes de um sistema de informação. A primeira fase na modelagem de sistemas de informação é a definição do Modelo de Requisitos, que tem por objetivo definir o domínio do problema e quais são as funcionalidades esperadas pelo sistema [14]. Neste trabalho utilizaremos o diagrama de classes e o diagrama de casos de uso da UML para construção deste modelo.

O diagrama de classes descreve as classes que formam a estrutura do sistema e as suas relações [2]. Uma classe define os atributos e as operações de um conjunto de objetos. Todos os objetos desta classe (instâncias da classe) compartilham o mesmo comportamento e possuem os mesmos atributos. As relações entre as classes podem ser associações, composições ou heranças. No caso do Modelo de Requisitos, utilizaremos o diagrama de classes apenas para representar as classes do domínio do problema, especificando quais são as entidades e suas definições.

O diagrama de casos de uso ilustra um conjunto de casos de uso, atores e suas relações. Sua aplicação visa à representação do contexto de um sistema em estudo, com ênfase na identificação das fronteiras (atores), e dos requisitos funcionais através do significado das suas funções [2].

Os atores representam as fronteiras de interação do sistema em estudo. Eles podem representar papéis executados por pessoas, sistemas ou organizações, que em algum momento interagem com as funcionalidades do sistema. Em geral, um ator pode interagir com várias funcionalidades, sendo que, por outro lado, uma funcionalidade pode interagir com vários atores.

Um caso de uso é uma seqüência de ações que um ou mais atores realizam num sistema de modo a obterem um resultado particular. Os casos de uso permitem capturar os requisitos funcionais de um sistema através do detalhe de todos os cenários que os usuários podem realizar [2].

## 3. Transformando o Modelo Negócio em Modelo de Requisitos

A transformação do Modelo de Negócio em Modelo de Requisitos consiste em duas etapas: a transformação da definição dos termos em classes e a transformação dos processos em casos de uso. As seções a seguir descrevem e apresentam exemplos destas duas etapas de transformação.

### 3.1. Transformação da Definição dos Termos em Classes

Uma vez que os termos de um determinado negócio tenham sido definidos, é possível mapear alguns destes termos em classes de domínio (diagrama de classes). O critério de seleção de termos a serem representados como classes foi o relacionamento de atributo. Segundo [9], um objeto deve possuir atributos para ser considerado um objeto. Desta forma, o mapeamento dos termos para uma classe de modelo conceitual considera apenas os termos que possuem atributos.

Assim, cada termo que possui relacionamento de atributo com outro termo é mapeado em uma classe. Os termos atributos são mapeados em atributos desta classe. Caso um termo atributo também possua atributos, ele será representado através de uma classe e o relacionamento de atributo será representado através de uma associação com rótulo “possui” entre os termos.

Os relacionamentos de herança, parte e associação são representados, respectivamente, por relacionamentos de generalização, composição e associação da UML. A tabela 2 mostra o mapeamento utilizado para gerar as classes de domínio.

**Tabela 2. Mapeamento da Definição dos Termos para Classes de Domínio.**

| Sigla | Descrição |
|---|---|
| MPT-1 | Cada termo que possui pelo menos um relacionamento de atributo com outro termo (modelo de sentença "TEM COMO ATRIBUTO") é mapeado como uma classe. |
| MPT-2 | Os termos atributos (modelo de sentença "TEM COMO ATRIBUTO") são mapeados em atributos da classe, onde o domínio utilizado na especificação do relacionamento será mapeado como o tipo do atributo. Caso um termo atributo também possua atributos, ele será representado através de uma classe e o relacionamento de atributo será representado através de uma associação com rótulo "possui" entre os termos. |
| MPT-3 | Os relacionamentos de herança (modelo de sentença "É SUBTIPO DE") será mapeado como um relacionamento de generalização entre as classes. |
| MPT-4 | Os relacionamentos de parte (modelo de sentença "TEM COMO PARTE") será mapeado como um relacionamento de composição entre as classes. |
| MPT-5 | Os relacionamentos de associação (modelo de sentença "ESTÁ RELACIONADO A") será mapeado como um relacionamento de associação entre as classes, onde o verbo e o grau utilizados na definição do relacionamento entre os termos serão mapeados, respectivamente, como o rótulo e o grau da associação entre as classes. |

Como exemplo desta transformação, apresentaremos o mapeamento dos termos de uma locadora de carros. Este exemplo é baseado no estudo de caso disponível em [3]. A tabela 3 apresenta a definição dos termos através dos modelos de sentença apresentados na seção 2.2. A figura 1 mostra o diagrama de classes obtido a partir desta definição.

**Tabela 3. Definição dos Principais Termos de uma Locadora de Carros. A definição foi construída utilizando os modelos de sentença definidos para representação das regras de negócio**

| Tipo | Definição |
|---|---|
| Atributos | Cliente TEM COMO ATRIBUTO CNH, nome, endereço, idade e tempo de habilitação.<br>Aluguel TEM COMO ATRIBUTO data início, data fim, valor da diária, CNH e placa.<br>Carro TEM COMO ATRIBUTO placa, estacionado no pátio, limpo e tanque cheio. |
| Domínios dos atributos | Nome, endereço e placa POSSUEM COMO DOMÍNIO texto.<br>CNH, idade e tempo de habilitação POSSUEM COMO DOMÍNIO número inteiro.<br>Data início e data fim POSSUEM COMO DOMÍNIO data.<br>Valor da diária POSSUI COMO DOMÍNIO número.<br>Estacionado no pátio, limpo e tanque cheio POSSUEM COMO DOMÍNIO sim/não. |
| Associações | Cliente ESTÁ RELACIONADO A aluguel POR realiza COM GRAU um PARA muitos.<br>Carro ESTÁ RELACIONADO A aluguel POR utiliza COM GRAU um PARA muitos. |

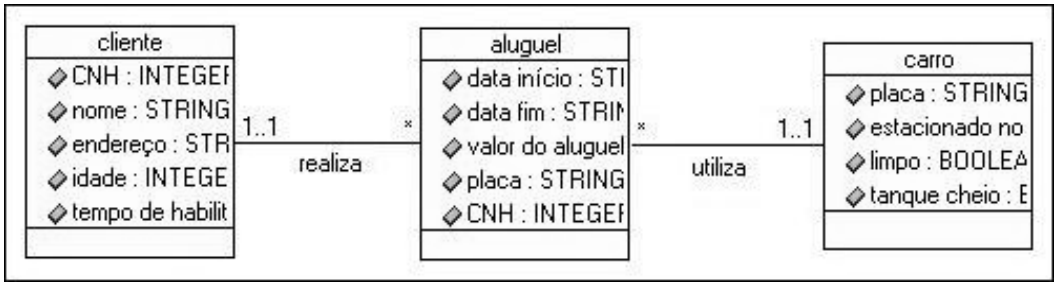


**Figura 1. Diagrama de Classes Gerado Automaticamente a Partir da Definição dos Termos. Os termos cliente, aluguel e carro foram mapeados como classes. Os demais foram mapeados como atributos.**

### 3.2. Transformação dos Processos em Casos de Uso

A transformação dos processos em casos de uso consiste na aplicação de quatro conjuntos de heurísticas a um determinado modelo de processo (diagrama de atividades), para a obtenção do respectivo modelo de requisitos funcionais (diagrama de casos de uso) [4]. Os diagramas utilizados são modelos gráficos que possuem a vantagem da facilidade de comunicação, porém não possuem um rigor matemático. Assim, a identificação das heurísticas foi realizada através de experimentos e em conformidade com a literatura especializada de modelagem de processos e requisitos [8, 12, 13]. A partir dos experimentos, foi possível observar a existência de algumas situações padronizadas de atividades que implicavam em padrões correspondentes de casos de usos. Estes padrões foram analisados e deles foram extraídas as heurísticas.

Em um processo de negócio, uma atividade é um conjunto de ações detalhadas para o cumprimento do objetivo do processo. Por outro lado, um caso de uso, é um conjunto de passos a serem executados com a finalidade de atingir um objetivo computacional (do sistema). Assim, existe uma correlação entre estes elementos (atividade e caso de uso), sendo esta a base para as heurísticas.

Algumas das heurísticas foram definidas baseando-se na topologia, disposição e interação entre os elementos utilizados nos diagramas. Esta topologia acarreta na adoção de determinados padrões de diagramação do modelo de requisitos funcionais. Uma restrição de topologia imposta para uso deste método é a de representação de apenas uma transição de entrada e uma de saída para as atividades. Esta restrição não caracteriza uma limitação, tendo em vista que o diagrama de atividades possui recursos para a junção e separação de transições.

A seguir, apresentamos as heurísticas desenvolvidas, divididas em quatro grupos: HAG (Heurísticas para Ajuste Geral), HIA (Heurísticas para Identificação de Atores), HIC (Heurísticas para Identificação de Casos de Uso) e HIR (Heurísticas para Identificação de Relações). Cada um dos grupos possui uma finalidade própria, sendo que a seqüência apresentada deve ser respeitada para a obtenção dos resultados.

**Tabela 4. Heurísticas para Ajuste Geral (HAG). A finalidade deste grupo de heurísticas é direcionar a aplicação das demais heurísticas, expressando ordenamento e limitações.**

| Sigla | Descrição |
|---|---|
| HAG-1 | As heurísticas serão aplicadas seguindo-se o fluxo de trabalho. |
| HAG-2 | As heurísticas serão aplicadas para elementos contidos em cada raia de responsabilidade isoladamente. Ou seja, elementos de raias diferentes não serão analisados em conjunto para aplicação das heurísticas. |

**Tabela 5. Heurísticas para Identificação dos Atores (HIA). A finalidade deste grupo de heurísticas é identificar atores a serem representados no diagrama de casos de uso, a partir dos elementos existentes no diagrama de atividades.**

| Sigla | Descrição |
|---|---|
| HIA-1 | Um processo, independente de sua natureza, é concebido para o atendimento de algum cliente externo ao processo. Este cliente deve ser identificado como um ator em potencial. |
| HIA-2 | As raias de responsabilidades caracterizam postos de trabalho e/ou seus responsáveis, representando efetivamente quem executa as atividades nelas contidas. Estes executores devem ser identificados como atores em potencial. |
| HIA-3 | Os artefatos caracterizam recursos, insumos ou produtos. Particularmente os recursos podem representar os executores de uma determinada atividade do processo. Recursos representados poderão ser identificados como responsáveis pela execução das atividades. Estes executores devem ser identificados como atores em potencial, interagindo com o caso de uso derivado da atividade. |

| | |
|---|---|
| HIA-4 | Os sinais representados para envio de notificações de eventos para fora da fronteira da aplicação possuem intrinsecamente um receptor. Este receptor deve ser identificado como um ator em potencial, interagindo com o caso de uso derivado da atividade que enviou o sinal. Analogamente, os sinais representados para recepção de eventos externos possuem um emissor que também será um ator em potencial. |
| HIA-5 | Atores identificados que apresentam interação irrelevante dentro do contexto em análise deverão ser desprezados pelo projetista. |
| HIA-6 | Atores identificados que apresentam interação irrelevante dentro do contexto em análise deverão ser desprezados pelo projetista. |

**Tabela 6. Heurísticas para Identificação dos Casos de Uso (HIC). A finalidade deste grupo de heurísticas é identificar as funcionalidades a serem representadas no diagrama de casos de uso, a partir dos elementos existentes no diagrama de atividades.**

| Sigla | Descrição |
|---|---|
| HIC-1 | Atividades de execução “passiva”, sem realização de trabalho e representadas na forma de estado, serão retiradas do diagrama sem a interrupção do fluxo de trabalho. Os elementos relacionados ao estado deverão ser transferidos para a próxima atividade de execução “ativa”. |
| HIC-2 | Atividades devem ser representadas na forma de casos de uso distintos, na relação de uma atividade para um caso de uso. |
| HIC-3 | Objetos interagindo com atividades darão origem a novos casos de uso que permitam a consulta de suas informações. |
| HIC-4 | Objetos não interagindo com atividades para atualização de informações darão origem a novos casos de uso que permitam a manutenção de suas informações. |

**Tabela 7. Heurísticas para Identificação de Relações (HIR). A finalidade deste grupo de heurísticas é identificar as relações a serem representadas no diagrama de casos de uso, a partir dos elementos existentes no diagrama de atividades. Estas relações ocorrem tanto entre os atores e casos de uso, como também entre os casos de uso.**

| Sigla | Descrição |
|---|---|
| HIR-1 | Os casos de uso originados de atividades seqüenciais em uma mesma raia de responsabilidade serão relacionados na forma “<<include>>” com um novo caso de uso que agrupe suas funcionalidades. |
| HIR-2 | Os casos de uso originados de atividades concorrentes em uma mesma raia de responsabilidade representadas a partir de um ponto de separação e sincronizadas em um outro ponto de junção dando continuidade ao fluxo de trabalho ou finalizando o processo serão representados isoladamente. |
| HIR-3 | Os casos de uso originados de atividades alternativas em uma mesma raia de responsabilidade representadas a partir de um ponto de desvio e unificadas em um outro ponto de fusão dando continuidade ao fluxo de trabalho serão relacionados na forma “<<extend>>” com um novo caso de uso que agrupe suas funcionalidades. |
| HIR-4 | Os casos de uso originados de atividades alternativas em uma mesma raia de responsabilidade representadas a partir de um ponto de desvio e finalizando o processo serão relacionados na forma “<<extend>>” com um novo caso de uso que agrupe suas funcionalidades. Os casos de uso originados da atividade alternativa, cujo caminho dá continuidade ao fluxo de trabalho, deverão ser representados isoladamente. |

| | |
|---|---|
| HIR-5 | Os casos de uso criados pelas heurísticas HIR-1, HIR-3 ou HIR-4 para agrupamento de funcionalidades, que se relacionarem apenas a um caso de uso, deverão ser desprezados. |
| HIR-6 | Os atores identificados pela heurística HIA-1 serão relacionados com todos os casos de uso principais, exceto os identificados nas heurísticas HIR-3 e HIC-4, através de relações do tipo interação. |
| HIR-7 | Os atores identificados pela heurística HIA-2 serão relacionados através de relações do tipo interação com todos os casos de uso principais originados de atividades da raia de responsabilidade em referência. |
| HIR-8 | Os atores identificados pela heurística HIA-3 serão relacionados através de relações do tipo interação com todos os casos de uso originados das atividades, que se relacionam os objetos em referência. |
| HIR-9 | Os atores identificados pela heurística HIA-4 serão relacionados, através de relações do tipo interação, com todos os casos de uso originados das atividades, que se relacionam os sinais em referência. |
| HIR-10 | Quando da existência de relação entre casos de uso do tipo “<<include>>” ou “<<extend>>”, os atores identificados pela heurística HIA-2 terão sua relação do tipo interação com o caso de uso principal. |

Dando continuidade ao exemplo da locadora de carros apresentado na seção anterior, mostraremos a aplicação destas heurísticas em um processo de aluguel de carros. Este processo é representado através do diagrama de atividades da figura 2. Após a aplicação das heurísticas o processo é transformado automaticamente no diagrama de casos de uso da figura 3.

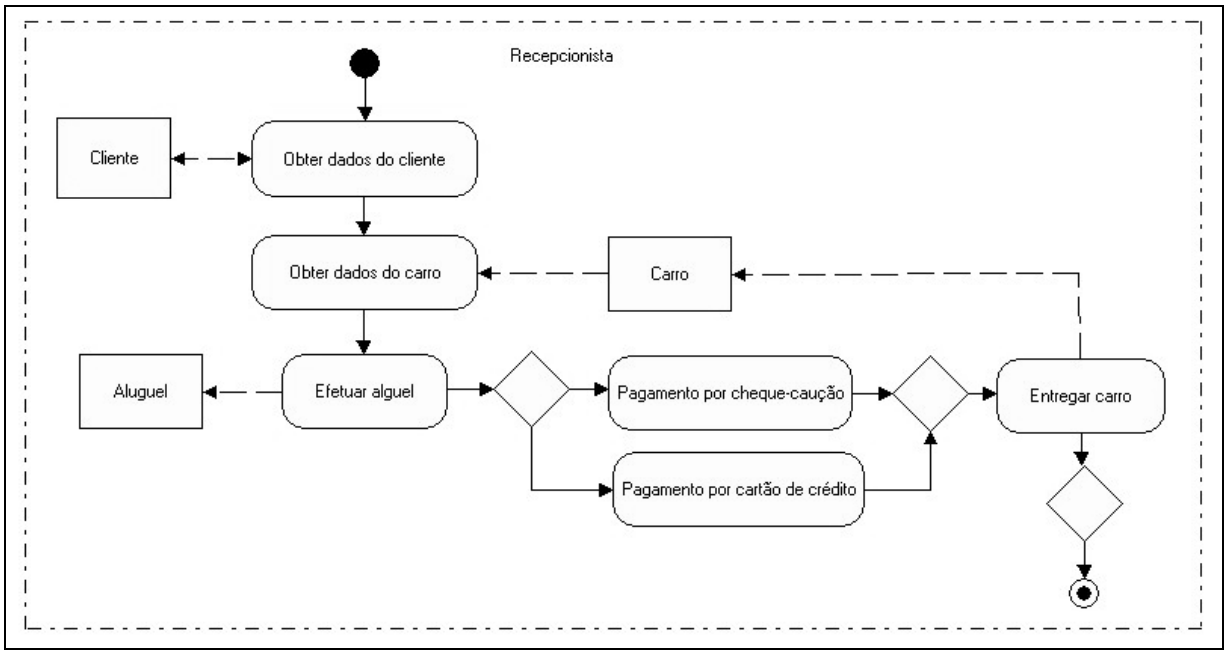


**Figura 2. Processo de Aluguel de Carros. A representação do processo foi feita através de um diagrama de atividades**

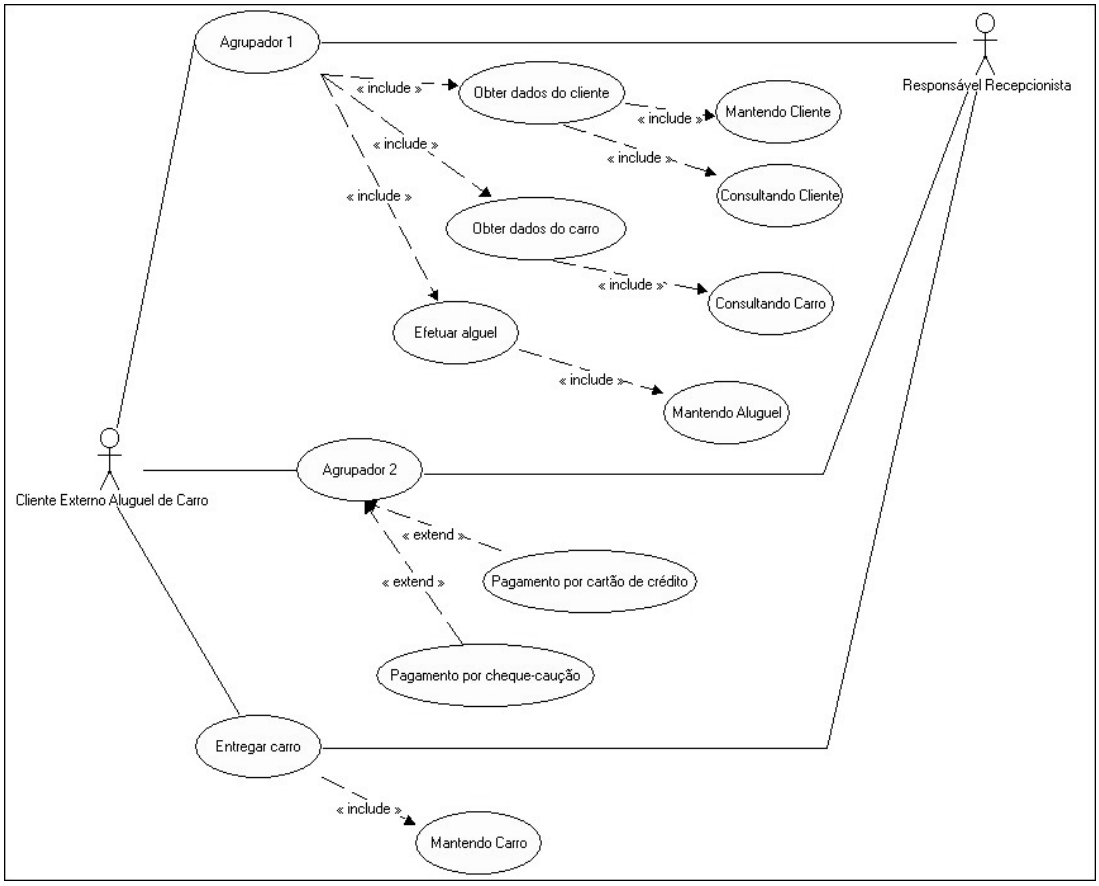


**Figura 3. Diagrama de Casos de Uso Gerado Automaticamente a Partir do Processo. As atividades do processo foram transformadas em casos de uso.**

Como pode ser observado pelos diagramas gerados (figura 1 e figura 3), além do mapeamento e das heurísticas, foram utilizadas algumas regras de diagramação. Estas regras têm como objetivo organizar a topologia dos diagramas gerados, facilitando a visualização dos elementos. Por considerações de espaço e de escopo, estas regras não serão apresentadas.

## 4. Automatizando a Transformação do Modelo de Negócio em Requisitos

Com o objetivo de automatizar as transformações apresentadas anteriormente, desenvolvemos o ambiente RAPDIS[2]. Através deste ambiente é possível construir modelos com alto nível de abstração e transformá-los automaticamente em modelos com baixa abstração, facilitando e tornando mais rápido o processo de desenvolvimento sistemas de informação. A figura 4 mostra a tela principal deste ambiente.

Para facilitar a apresentação das suas funcionalidades, dividimos este ambiente em duas partes: o Modelo de Negócio e o Modelo de Sistema de Informação. Cada uma destas partes possui um editor e ferramentas para realizar transformação entre os modelos.

Na parte relativa ao Modelo de Negócio é possível especificar os termos, objetivos, recursos, processos e regras do negócio. Além disso, esta parte do RAPDIS dispõe de dois geradores: o gerador de classes de domínio e o gerador de casos de uso. O gerador de classes de domínio realiza a transformação da definição dos termos do negócio em um diagrama de classes, implementando o mapeamento apresentado na seção 3.1. O gerador de casos de uso transforma um processo de negócio em um diagrama de casos de uso, implementando as heurísticas descritas na seção 3.2.

A parte relativa ao Modelo de Sistema de Informação permite a edição dos principais diagramas da UML: o diagrama de casos de uso, o diagrama de classes, o diagrama de estados e o diagrama de seqüência. Esta parte apresenta também um gerador de código que transforma estes diagramas no código-fonte do esqueleto da arquitetura do sistema de informação modelado. O mapeamento implementado por este gerador não faz parte do escopo deste artigo. Entretanto, vale notar que os requisitos obtidos a partir do modelo de negócio são transformados automaticamente em código através deste gerador.

[2] O RAPDIS é um software gratuito e pode ser baixado no seguinte endereço eletrônico: www.geti.dcc.ufrj.br/projetos.php.

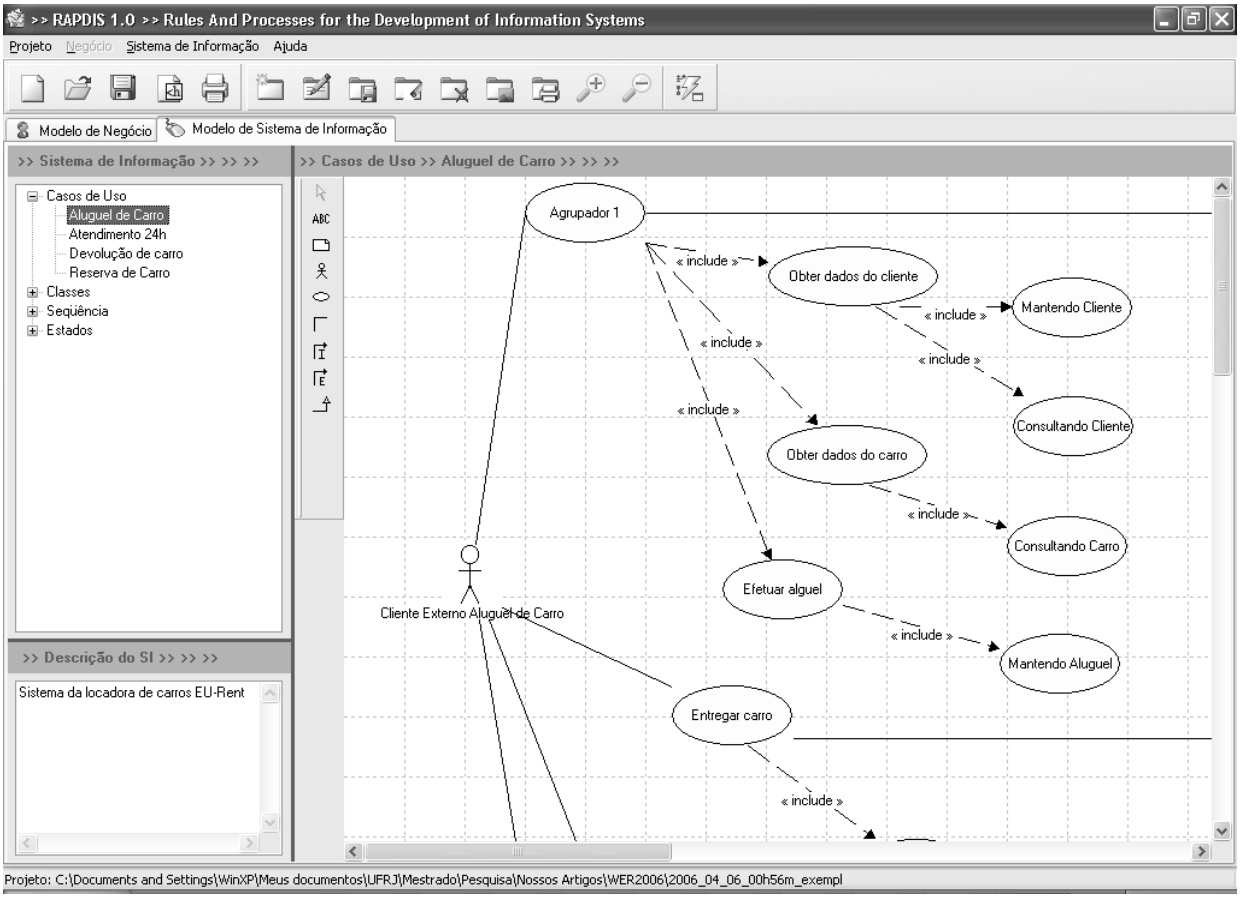


**Figura 4: Tela Principal do Ambiente RAPDIS**

## 5. Conclusão

Este artigo apresentou uma abordagem que permite realizar a transformação automática do Modelo de Negócio para o Modelo de Requisitos do sistema de informação. Portanto, ao modelar o negócio para qual será desenvolvido o sistema de informação, estão sendo também definidos os requisitos do sistema, cujos modelos de caso de uso e de classes de domínio podem ser gerados automaticamente através do ambiente RAPDIS.

Essa abordagem oferece como uma de suas contribuições o desenvolvimento de sistemas de informação cujas funcionalidades estejam mais aderentes ao comportamento do negócio. Adicionalmente, a abordagem promove a uniformização da nomenclatura utilizada nos modelos e uma maior consistência e padronização dos modelos de requisitos gerados. Uma outra contribuição se deve ao fato de que com esta abordagem torna possível a construção do Modelo de Requisitos mais rapidamente, já que parte deste modelo pode ser obtida automaticamente a partir da definição dos termos e dos processos do Modelo de Negócio.

Neste trabalho apresentamos apenas a transformação das regras de negócio relacionadas à definição dos termos. A transformação das demais regras para o Modelo de Requisitos é um trabalho que se encontra em andamento. Outro trabalho que está sendo desenvolvido para aprimorar esta abordagem consiste na ampliação das heurísticas da transformação dos processos, visando contemplar não só a parte gráfica, mas também a parte textual de apoio aos modelos.

## 12. Referências

[1] Blanc, X., Gervais, M., Sriplakinch, P.: Model Bus: Towards the interoperability of modeling tools. Proceedings of Model-Driven Architecture: Fundations and Applications. (2004)

[2] Booch, G., Rumbaugh, J. Jacobson, I.: Unified Modeling Language User Guide. Addison-Wesley. (1999)

[3] BRG - Business Rules Group: GUIDE Business Rules Project Final Report. (1997)

[4] Cruz, P. O. S., Silveira, D. S., Schmitz, E. A.: Heurísticas para Extração dos Casos de Uso de Negócio a Partir da Modelagem de Processos. III CAPSI. Coimbra. Portugal. (2002)

[5] Eriksson, H., Penker, M.: Business Modeling with UML: Business Patterns at Work. John Wiley & Sons. (2000)

[6] Kleppe, A., Warmer, J., Bast W.: MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley. (2003)

[7] Larman, C.: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Hardcover. (2004)
[8] Marshall, C.: Enterprise Modeling with UML. Addison-Wesley. (2000)
[9] Mellor, S., Shlaer, S.: Análise de Sistemas Orientada para Objetos. São Paulo. MacGraw-Hill. (1990)
[10] Mukerji, J., Miller, J.: MDA Guide. http://www.omg.org/docs/omg/03-06-01.pdf. (2003)
[11] OMG: Unified Modeling Language: Superstructure. Versão 2.0. http://www.omg.org/docs/formal/05-07-04.pdf. (2005)
[12] Santander, V. F. A., Castro, J. F. B.: Desenvolvendo Use Case a partir de Modelagem Organizacional. WER. (2000)
[13] Shnieder, G., Winters, J. P.: Applying Use-Case, A practical Guide. Second Edition. Addison-Wesley. (2001)
[14] Silveira, D. S., Schmitz, E. A.: Desenvolvimento de Software Orientado a Objetos. Brasport. (2000)