Pré-Projeto e Modelagem de Negócios

Ricardo R. Gudwin
DCA-FEEC-UNICAMP
06/10/2011

Introdução

Desde sua proposição original, em (Jacobson et.al. 1999), o Processo Unificado sofreu diversas modificações e aperfeiçoamentos. Nessas modificações, algumas fases foram acrescentadas, outras foram modificadas e aperfeiçoadas para tornar o processo mais eficiente e menos ambíguo. Uma das modificações proposta neste ínterim foi a preconização de uma fase anterior ao ciclo de desenvolvimento envolvendo múltiplas iterações, que neste documento será chamada de pré-projeto. Normalmente, as atividades desenvolvidas no pré-projeto são executadas uma única vez durante o ciclo de desenvolvimento de software, ao contrário das fases de especificação de requisitos. As atividades do pré-projeto envolvem normalmente a assim chamada "Modelagem do Negócio". Como na maioria das situações, o software sendo desenvolvido visa aumentar a produtividade ou melhorar a eficiência de um negócio ou empresa, presume-se que seria de alguma utilidade para o desenvolvimento do software conhecer em detalhes os aspectos do negócio da empresa. Essa presunção origina-se da interpretação de que os requisitos de um software devem expressar as necessidades reais do contratante, e não seus desejos. Muitas vezes o contratante de um desenvolvimento de software não tem claras suas reais necessidades, sendo que o puro atendimento de seus desejos poderia levar à construção de um software que não atendesse as reais necessidades do cliente, levando a seu descontentamento com o resultado. A lógica por trás de se levantar um modelo de negócios é tentar entender o negócio do cliente para propor uma ferramenta computacional capaz de dar suporte ao negócio do mesmo. Esse modelo pode ser bastante rudimentar, envolvendo normalmente a elaboração de um modelo conceitual de domínio do negócio, e algumas vezes uma elaboração dos processos de negócios envolvidos, organizados a partir dos casos de uso de negócios.

Em nossa customização do processo Unificado que iremos estudar para fins didáticos, utilizaremos 2 diferentes artefatos que envolvem a modelagem de negócios, e que por conseguinte serão desenvolvidas nesta fase de pré-projeto:

- A Elaboração do Documento de Visão do Problema
- O Modelo Conceitual de Domínio

Além disso, faremos uma breve descrição sobre casos de uso de negócios e sua comparação com casos de uso do sistema.

O Documento de Visão do Problema

Durante a fase do pré-projeto, ou seja, antes de começarmos as iterações que correspondem ao projeto de um software, segundo o processo Unificado, é necessário fazermos um planejamento prévio de como esse desenvolvimento se dará. Esse planejamento se consolida na criação de um documento que é chamado de **Documento de Visão do Problema**. Esse nome se justifica, pois seu conteúdo dará uma visão geral de como se imagina o sistema que pretendemos construir. Não existe uma **norma** regulando como esse documento de visão deve ser desenvolvido. Entretanto, utilizaremos como guia uma sugestão dada no livro de Larman (Larman 1998), de um conjunto de capítulos que de um modo geral constituem uma boa sequência de elementos para definir a visão de um problema. Em seu livro, Larman propõe a criação de 5 capítulos, entitulados:

- Compreensão do Problema
- Proposta de Solução de Software
- Visão Geral dos Pré-Requisitos
- Planejamento Logístico
- Glossário

No capítulo referente à **Compreensão do Problema**, devemos incluir um texto, geralmente 2 ou 3 parágrafos, com uma descrição verbal do domínio do problema. Essa descrição tentará abstrair o problema que se tenta resolver, sem envolver conjuntamente a proposta de software que (seguramente) já temos para sua resolução. Podemos entender esse capítulo como sendo uma espécie de elicitação das necessidades dos investidores, ou seja, aqueles que em princípio estão nos pagando para que façamos o software. Por exemplo … vamos supor que em nossa proposta de tema, sugerimos a criação de um banco de dados para uma loja de peças de automóvel, sugerindo que nossos investidores possuem uma loja de peças de automóveis e resolveram gastar dinheiro desenvolvimento um sistema de software para resolver algum problema que lá apareceu (pois se não existe um problema, um comerciante não se iria pensar em gastar dinheiro sem mais nem menos !). Na compreensão do problema, devemos tentar abstrair o real problema que se coloca e que motivou essa proposta. Ou seja, na verdade o problema que se tem é o fato de que uma loja de peças de automóveis possui um número muito grande de dados e informações que precisam ser organizados, acessadas e atualizadas de maneira eficiente. Tais informações envolvem clientes, fornecedores, produtos, peças, etc. que necessitam ser organizadas e controladas em seu cotidiano de funcionamento. Devemos portanto descrever aqui quais seriam as particularidades desse problema em questão, descrevendo características relacionadas ao domínio em questão (no caso, uma loja de peças de automóveis), e não a solução previamente escolhida para combatê-la. Muitas vezes, o cliente já vem com o que ele pensa ser uma solução. Por exemplo … o cliente chega e já vai dizendo … "quero um banco de dados para cadastrar as auto-peças". Um bom engenheiro de computação deve nessa hora saber questionar a si mesmo se é exatamente essa a solução que o cliente precisa. Caso contrário, após o programa ficar pronto, você terá um cliente insatisfeito, pois apesar de você ter construído o que ele pediu, isso que ele pediu não resolve o problema dele. Por exemplo, se o negócio do cliente envolve também um estoque rotativo, utilizado por vários funcionários, talvez o que ele precise realmente seja um controle de estoque mais eficiente, que informe também qual funcionário fez uso de qual peça. Um simples cadastro de peças não resolveria por exemplo um problema de "sumiço" de peças do estoque, que é o real problema do cliente e que ele implicitamente deseja solucionar. Precisamos pois ser bem críticos, para garantir que conhecemos o problema de nosso cliente, e que a solucão que estaremos propondo realmente é o que ele precisa. Nem sempre um banco de dados é a solução para os problemas de um cliente. Veja por exemplo outros exemplos de problemas:

- Deseja-se desenvolver um novo produto eletrônico (e.g. um telefone celular), com um conjunto rico de funções (e.g. discagem rápida, memória, discagem de emergência, etc.)
- Têm-se um processo físico que se deseja modelar e analisar, de tal forma a viabilizar a predição de seu comportamento e o teste do impacto de diferentes decisões nos resultados
- Têm-se uma organização, com toda sua logística e métodos, e se deseja melhorar sua eficiência, confiabilidade e custo
- Deseja-se produzir algum tipo de produto de maneira eficiente e confiável, utilizando máquinas automátizadas

Para cada um dos domínios acima, existe claramente um problema que necessita ser resolvido. No presente capítulo, nossa missão é descrever sucintamente esse problema.

No capítulo referente à **Proposta de Solução via Software**, devemos, aí sim, descrever sumariamente como estamos visualizando um sistema de software para resolver o problema. Aqui devem ser estabelecidos metas e objetivos que desejamos atingir com o software que será desenvolvido. Esse capítulo, a semelhança do anterior, será um texto enxuto que descreva de maneira geral qual é o tipo de sistema de softare que se preconiza para o problema do cliente. Por exemplo, para o problema da loja de peças, a solução óbvia é a criação de um banco de dados que gerencie não somente as peças mas também os funcionários e o estoque do estabelecimento. Para os outros exemplos apontados, teríamos (na ordem em que foram apresentados):

- Usar microprocessadores ou micro-controladores como parte do produto e um sistema embutido que proveja as funcionalidades desejadas
- Desenvolver um simulador para simular o processo e prover estatísticas para a análise de decisões
- Uso de ferramentas de produtividade (e.g. ferramentas do tipo office, e-mail, web-sites, etc), e aplicações de software de propósito específico, de tal forma a automatizar parte do processo organizacional
- Desenvolvimento de um software de controle de sistemas para controlar as máquinas envolvidas na produção dos produtos desejados

No capítulo referente à **Visão Geral dos Pré-Requisitos**, devemos definir basicamente um conjunto de características que desejamos para nosso sistema, na forma de funções e atributos do sistema. As funções do sistema devem descrever basicamente o que se supõe que o sistema deve **fazer**. Os atributos do sistema devem descrever o que se supõe que o sistema deve **ser** ou **ter**. Por exemplo, funções de um sistema editor de plantas residenciais incluiriam desenhar plantas, alterá-las, imprimí-las, exportá-las em formatos de outros editores, projetar a planta tridimensionalmente, calcular a quantidade materiais necessários para a obra, etc. Para esse mesmo exemplo, atributos do sistema seriam por exemplo, dizer que o sistema opera em múltiplos sistemas operacionais, permite a edição de paredes, janelas, portas e pisos, opera em preto e branco ou colorido, permite a inserção de móveis para a visualização da projeção 3D, etc. Observe que as funções denominam ações que o sistema deve executar, ao passo que os atributos modelam características de capacidade ou de variações que o sistema apresenta. Outro ponto é que os atributos sempre podem ser colocados na forma atributo-valor:

Sistemas operacionais	-	múltiplos
Caracteres editáveis	-	paredes, portas, janelas, pisos
Tipos de Impressão	-	preto e branco e colorida
Tipos de Visualização	-	com móveis ou sem móveis

Para determinarmos as funções e atributos do sistema, devemos fazer um brainstorm, normalmente junto com o cliente, propondo "features" que passariam a caracterizar o sistema. Tome muito cuidado para não confundir funções com atributos. Tanto funções como atributos devem ser colocados na forma de tabelas, indicando, dentre outras coisas os seguintes quesitos:

Status da Função/Atributo	-	(proposto, aprovado, incorporado, validado, etc..)
Custo Estimado de Implementação	-	(em termos de tipos de recursos e homens-hora)
Prioridade	-	(crítica, importante, desejável, dispensável)
Nível de Risco na Implementação	-	(crítico, significativo, ordinário)

O Status da Função/Atributo indicará qual o status da função ou atributo em questão. Assim, pode-se planejar com o cliente quais as funções/atributos que se desejam implementar. O custo estimado de implementação permite ao cliente fazer escolhas quanto às funções atributos que deseja custear (lembrando que tudo tem um custo). A prioridade define se a omissão da função ou atributo pode comprometer o projeto como um todo ou se a função atributo poderia ser suprimida sem problemas, caso o cliente deseje. Por fim o nível de risco na implementação caracteriza as dificuldades previstas na incorporação da função/atributo, caso o cliente opte por incluí-lo.

O capítulo **Planejamento Logístico** visa determinar a equipe de trabalho que será envolvida no projeto e detalhes do planejamento logístico para o desenvolvimento em questão. Esse capítulo deve apresentar os seguintes ítens:

- Equipes de Trabaho
- Grupos Afetados
- Fatos Presumidos
- Riscos
- Dependências

A *equipe de trabalho* deve indicar claramente as pessoas que estarão envolvidas no projeto. No caso de vocês, deve conter o nome, RA e e-mail de cada um dos elementos participantes do grupo. Em um caso geral, poderia ainda conter endereços e telefones dos elementos participantes, bem como qualquer outra informação relevante, tal como a formação acadêmica, anos de experiência, etc.

O ítem *grupos afetados* deve indicar quais os grupos afetados pelo projeto em questão. Em uma grande empresa, muitas vezes o projeto de um software envolve diversos grupos dentro da empresa. De um modo geral é necessário saber-se quais serão os grupos afetados, ou seja, que estão esperando a conclusão de nossas atividades para dar continuidade em seus próprios projetos.

O ítem *fatos presumidos* deve incluir os fatos que se presumem verdadeiros antes de se iniciar o projeto. Por exemplo, presume-se que determinado equipamente estará a disposição, ou que determinados elementos estarão a disposição para colaborar com o projeto. Muitas vezes é importante se documentar os fatos presumidos, para que o cliente fique ciente das condições em que o planejamento logístico está sendo efetuado, de tal forma que, caso haja alguma modificacão nas condições operacionais, ele esteja ciente de que isso poderá afetar o projeto. Da mesma forma, os fatos presumidos servem para estabelecer um acordo mútuo entre o cliente e a empresa de desenvolvimento, onde cada um está ciente das condições em que se dará o projeto e o que a equipe de desenvolvimento espera do cliente em relação a colaboração, recursos, etc.

O ítem *riscos* apontam fatos, eventos ou situações que podem levar ao fracasso ou atrasos no projeto, bem como as potenciais consequências do fracasso ou do atraso. É importante o cliente estar ciente dos riscos envolvidos e ao mesmo tempo saber que os desenvolvedores também estão cientes da possibilidade de que ocorram.

O ítem *dependências* devem identificar outros parceiros, sistemas ou produtos dos quais o projeto depende para sua implementação, e que podem eventualmente levar ao aparecimento de riscos.

Assim, o planejamento logístico deve documentar as condições em que o desenvolvimento será efetuado e qual a logística que será implementada.

O capítulo **glossário**, por final, deve conter um levantamento do vocabulário relativo ao domínio, ou seja, contendo os principais termos utilizados para descrever as características do problema..

O Modelo Conceitual de Domínio

O modelo conceitual de domínio visa a aquisição e modelagem de informações sobre o domínio organizacional sob o qual o contexto do projeto se insere. Em outras palavras, ele visa estruturar o conhecimento que se tem sobre o problema que se deseja resolver. Em alguns tipos de software, esse modelo do domínio tem uma importância somente marginal, mas para outros, tais como bancos de dados, por exemplo, o modelo conceitual de domínio tem uma importância muito grande.

De uma maneira geral, o modelo de domínio envolve o contexto onde o software será aplicado, e sua compreensão, antes da introducão do sistema de software que se pretende desenvolver. Novamente, em outras palavras: antes de nos colocarmos a pensar no software que iremos desenvolver, gastaremos um certo tempo conhecendo o problema que queremos resolver: seu contexto, sua terminologia, seu jargão, o conhecimento e descoberta de relações existentes entre seus elementos, etc. Assim, vemos que o modelo conceitual ilustra conceitos significativos de um domínio - aquilo que devemos estar cientes a respeito do domínio, representando sempre coisas do mundo real **não** componentes de software.

O modelo conceitual é obtido por meio de um procedimento chamado de Análise de Domínio. Nesta análise, os conceitos apurados são expressos por meio de um diagrama de classes UML. Os possíveis relacionamentos entre os conceitos são expressos por meio de associações entre os conceitos, e as particularidades envolvendo os conceitos representados são expressas por meio dos atributos nos diagramas de classe.

Nesse ponto, é necessário fazermos uma introdução, comparando a metodologia de análise orientada a objetos com a metodologia de análise estruturada. A análise estruturada enfoca nossos esforços na descoberta de processos ou funções, ao passo que a análise orientada a objetos focaliza a descoberta de conceitos e seus inter-relacionamentos. É exatamente isso que fazemos aqui na análise do domínio.

Um dos principais métodos na análise do domínio consiste na busca por entidades participantes da Lista de Categorias de Conceitos. Essa lista abrange as seguintes categorias:

- objetos físicos ou tangíveis
- especificações ou descrições de coisas
- lugares
- transações
- ítens sendo transacionados
- papéis de pessoas
- coisas que contém outras coisas
- coisas que são contidas em outras coisas
- regras e políticas
- eventos
- catálogos de coisas

Assim, devemos procurar em nosso entendimento do domínio, todos os conceitos que possam ser categorizados de objetos físicos ou tangíveis, especificações ou descrição de coisas, lugares, transações, … , etc..

O modelo conceitual passa então a ser assim elaborado: inicialmente fazemos um levantamento dos conceitos candidatos utilizando a lista acima. Em seguida, fazemos a inserção dos conceitos levantados no diagrama de classes. Em seguida, passamos a procurar por possíveis associações existentes entre os conceitos, representando as mesmas também no diagrama de classes. Por fim fazemos a inserção dos atributos necessários.

Algumas "dicas" são importantes durante essas atividades. Por exemplo, … , devemos evitar a representação de conceitos irrelevantes. Ou seja, tratemos de usar nomes de coisas que realmente existem, excluindo detalhes que possam ser irrelevantes. Uma boa política é utilizarmos o princípio do cartógrafo (desenhista de mapas). O que devemos colocar em um mapa quando o elaboramos ? Vemos que os ítens que aparecem nos mapas são os ítens relevantes para a descrição geral da região coberta pelo mapa. Por exemplo, um mapa pode apresentar a localização de parques e jardins, igrejas e outros edifícios relevantes, mas dificilmente mostrará onde ficam os postos de gasolina e os números das casas na rua. Assim, durante a elaboração do modelo conceitual, devemos nos preocupar em não poluir o modelo, e introduzir somente os elementos importantes para a compreensão do domínio, evitando conceitos que não sejam relevantes. Por exemplo: imaginemos que queremos elaborar o modelo conceitual de uma aula em sala de aula. Assim, diversos conceitos importantes surgirão, tais como *lousa*, *carteira*, *matéria*, *professor*, *aluno*, *caderno*, *lápis*, *giz*. Outros conceitos, que existem de fato em uma sala de aula, tais como *cortina*, *vidro*, *interruptor*, *piso*, *teto*, etc. não são relevantes para a descrição do domínio, não sendo portanto incluídos no modelo conceitual.

Uma outra dúvida frequente que aparece durante o levantamento do modelo conceitual é saber quando um determinado conceito deve ser representado como um conceito e quando deve ser apresentado como um atributo. Essa dúvida é muito pertinente. Imagine durante o levantamento de um modelo conceitual que nos deparamos com o conceito *cliente*. Bem, analisando esse conceito, descobrimos que todo *cliente* deve ter um *endereço*. O que é o *endereço* ? Um novo conceito ou um atributo de *cliente* ? Para resolver esse conflito, utilizaremos uma regra prática muito utilizada: se um candidato a conceito X não pode ser pensado como um número ou um texto no mundo real, então X é realmente um conceito. Caso contrário é um atributo. Na dúvida, devemos sempre torná-lo um conceito separado. Com relação ao nosso *endereço*: se podemos imaginar que o *endereço* é somente um texto (algo como "Rua dos Alecrins, 32 -13083-560 Campinas SP", então devemos tornar *endereço* um atributo. Se, ao contrário, quisermos desmembrar *endereço* em *rua*, *número*, *CEP*, *cidade*, *estado*, então *endereço* passa a ser um conceito independente, e *rua*, *número*, *CEP*, *cidade* e *estado* passam a ser atributos de *endereço*.

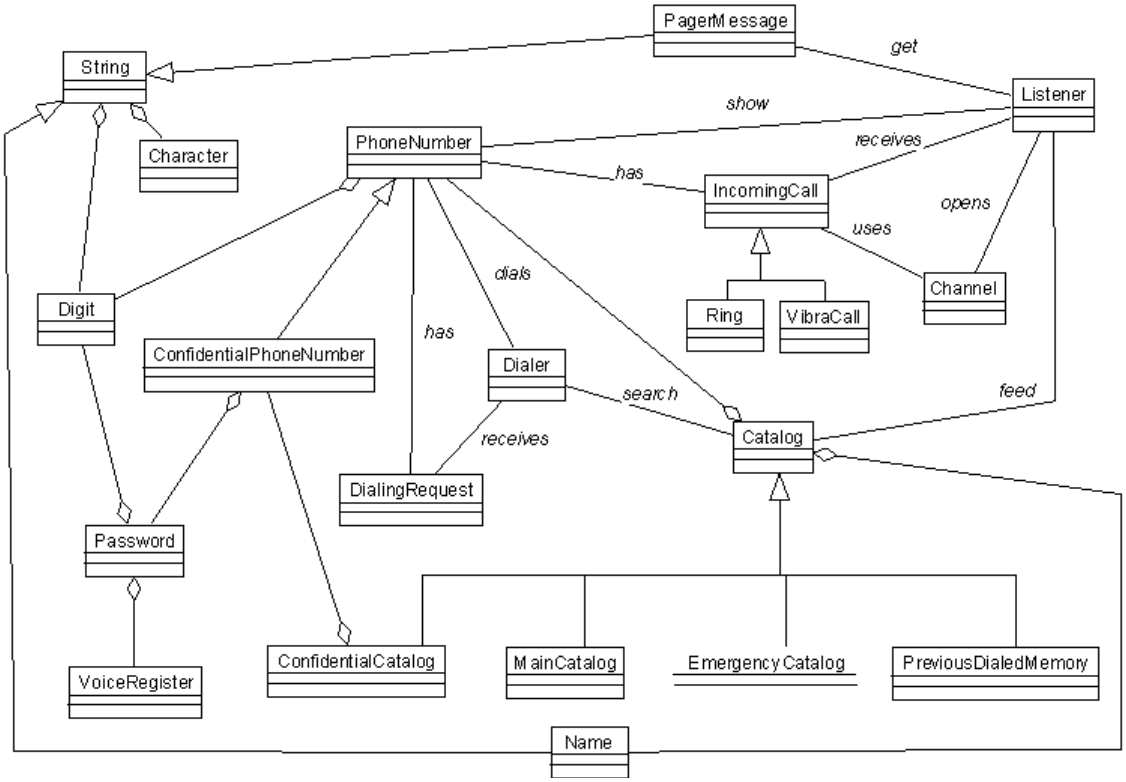
Com relação à inserção de associações: associações representam um relacionamento entre conceitos que indicam uma conexão significativa ou interessante entre conceitos. Assim, um bom critério para a elicitação de associações úteis é verificar se um possível relacionamento entre conceitos é um conhecimento que deva ser preservado durante certo tempo. Nesse caso, essa associação é útil e deve ser inserida no modelo. Uma outra maneira de descobrir associações úteis é verificar se o relacionamento corresponde a um dos relacionamentos presentes na Lista de Associações Comuns. Assim, devemos incluir uma associação no diagrama de classes entre os conceitos A e B se:

- A é uma parte física de B
- A é uma parte lógica de B
- A está fisicamente contido em B
- A está logicamente contido em B
- A é uma descrição para B
- A é um ítem transacionado por B
- A é armazenado em B
- A é membro de B
- A utiliza ou gerencia B
- A se comunica com B
- A possui ou é possuído por B
- A está próximo de B

Entretanto, devemos tomar cuidado durante o levantamento de associações. Uma armadilha comum é gastar muito tempo tentando descobrir associações. Normalmente, encontrar conceitos é muito

mais importante que encontrar associações. Deve-se gastar mais tempo na descoberta de conceitos do que de associações. Muitas vezes, a inserção de muitas associações tendem a confundir, mais do que iluminar nosso modelo conceitual.

Um exemplo de um modelo conceitual de domínio referente ao domínio de telefones celulares é mostrado a seguir. Observe-o e tente interpretá-lo, para verificar se já compreende a notação UML.



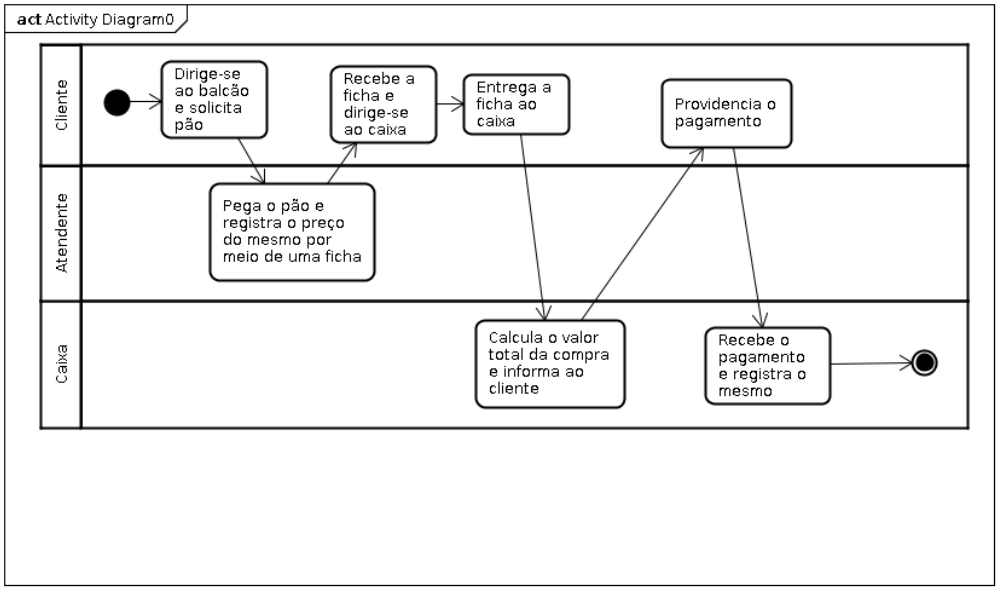
Uma última dúvida frequentemente persegue aqueles que se iniciam na arte do levantamento de modelos conceituais: saber quando parar. Até onde devemos continuar perscrutando nossas mentes atrás de mais conceitos, associações e atributos ? Normalmente não existe uma resposta fácil para esta questão. Entretanto, novamente, usaremos uma regra prática: devemos olhar para o modelo como um todo e nos perguntarmos: será que isso representa o domínio como um todo ? Se concordarmos que sim, paramos. Se não, continuamos mais um pouco. Entretanto, devemos sempre impor uma condição de limite absoluto de tempo. O que é isso ? Dizemos para nós mesmos: temos mais X minutos (ou horas, dependendo). Assim que o tempo acabar, terminamos a análise do domínio. Com qualquer modelo que tenhamos !

Negócios e Processos de Negócios

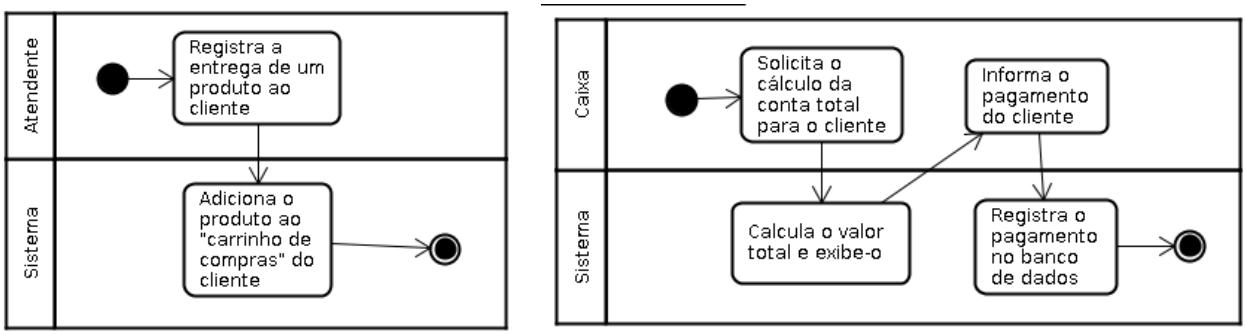
Um **negócio** é uma atividade econômica exercida por uma pessoa ou grupo de pessoas visando a obtenção de recursos econômicos. Desta forma, podemos atribuir o termo "negócio" desde às atividades desenvolvidas por profissionais autônomos como médicos, dentistas, advogados, passando por estabelecimentos de pequeno porte como comerciantes, e donos de pequenos estabelecimentos comerciais, oficinas mecânicas, restaurantes, postos de gasolina, até grandes conglomerados de pessoas organizadas em indústrias, corporações, joint-ventures e empresas multi-nacionais.

Para gerir o negócio, usualmente utiliza-se do trabalho de diferentes pessoas, possivelmente organizadas em departamentos ou seções especializadas, onde cada participante do negócio possui uma função ou atribuição, para que a dinâmica do negócio se consolide de forma a resultar na obtenção dos recursos econômicos que são o objetivo do negócio.

Um **modelo de negócios** descreve a lógica por meio da qual uma organização captura, cria, gerencia e disponibiliza valores. Além de um Modelo Conceitual de Domínio, que normalmente descreve os principais recursos (recursos humanos, materiais ou econômicos) envolvidos em um negócio, um modelo de negócio envolve a descrição de um ou mais processos de negócios. Um **processo de negócio** corresponde a uma sequência de interações entre os diferentes participantes de um negócio, (incluindo aí os clientes do negócio), cada um com sua função, de tal forma que o resultado seja a geração de valores para o negócio, bem como para os clientes do negócio. Desta forma, para a caracterização de um processo de negócio, deve-se apontar quais são os personagens envolvidos e quais as atividades desempenhadas por cada personagem. Diversas linguagens de modelagem para processos de negócios foram desenvolvidas (e.g. BPML, BPEL, BPMN). Uma alternativa, utilizando UML é o desenvolvimento de casos de uso de negócios, onde os processos de negócios são modelados de maneira análoga aos casos de uso em desenvolvimento de sistemas. A diferença entre uma caso de uso de negócios e um caso de uso do sistema pode ser bastante sutil, à medida em que sistemas computacionais são utilizados para dar apoio a processos de negócios. De uma maneira geral, quando se desenvolve casos de uso do sistema, descreve-se a interação entre diferentes usuários e o sistema, de forma a agregar algum valor ao usuário. Já nos casos de uso de negócios, a interação não é entre os usuários e o sistema, mas entre os diferentes participantes de um negócio. Vamos ilustrar essa diferença por meio de um exemplo. Imaginemos como negócio uma padaria, e um processo de negócio que seja a venda de pães a um cliente. Poderíamos detalhar esse processo de negócios por meio de um caso de uso de negócios descrito por meio de um diagrama de atividades:

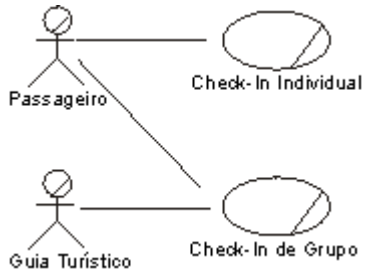


Esse mesmo caso de uso de negócios daria margem a pelo menos dois diferentes casos de uso do sistema: “Registro de Pedido” e “Pagamento de Compra”.



Observe que, apesar do cliente participar do caso de uso de negócios, nos casos de uso do sistema, o cliente não aparece em nenhum momento, pois o mesmo nunca interage diretamente com o sistema. Desta forma, é importante distinguir entre **casos de uso de negócio** e **casos de uso de sistema**, e compreendermos a diferença que existe entre os dois. No caso de uso de negócio, todas as interações referentes ao negócio devem ser detalhadas. No caso de uso do sistema, somente as ações que envolvam a interação com o sistema são detalhadas, e somente os usuários que interagem com o sistema devem ser representados e ter suas ações descritas.

Casos de uso de negócios são representados em diagramas de casos de uso, da mesma maneira que os casos de uso do sistema. Alternativamente, pode ser usada uma versão estereotipada de atores e casos de uso, para indicar que se trata de casos de uso de negócios. Um exemplo deste caso é apresentado na figura a seguir:



O levantamento (por meio de um diagrama de casos de uso) e a modelagem (por meio de diagramas de atividades ou texto) de casos de uso de negócios são uma etapa potencialmente importante do desenvolvimento de um sistema. Antigamente, havia um tipo de profissional explicitamente encarregado de desenvolver a modelagem de negócios, o Analista de Organização e Métodos, ou analista de O&M. Atualmente, existem vários tipos de profissionais envolvidos com a modelagem de negócios, o analista de processos de negócios, o designer de processos de negócios, o designer de negócios, etc. Existem muitas variações em como proceder a uma modelagem de negócios. Pode ser que o negócio já envolva algum tipo de sistema computacional, ou pode ser que o negócio ainda não envolva um sistema computacional. De qualquer forma, é necessário conhecer os processos de negócio para que um sistema computacional possa ser proposto para dar suporte a esse processo de negócios. Muitas vezes, caso o negócio já envolva um sistema computacional, pode ser necessário repensar os processos de negócio de forma que o uso dos recursos computacionais possam ser utilizados de modo mais produtivo para o negócio.

Em casos em que o sistema sendo desenvolvido não se destina a dar suporte a um processo de negócios, como por exemplo um sistema embutido de controle, um sistema geral de produtividade ou um jogo de computador, essa etapa de modelagem de negócios pode ser dispensada, entretanto.

Bibliografia

(Jacobson et.al. 1999) Ivar Jacobson, Grady Booch, James Rumbaugh - **The Unified Software Development Process** - Addison Wesley, 1999.

(Larman 1998) Craig Larman - **Applying UML and Patterns - An Introduction to Object Oriented Analysis and Design** - Prentice Hall Inc., New Jersey 1998.