Daniel Del Giglio Guida

Análise de Requisitos para um Sistema de Informações de uma Empresa Gestora de Ativos Alternativos

Trabalho de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Diploma de Engenheiro de Produção – Área Mecânica

Orientador:
Prof. Mauro de Mesquita Spínola

São Paulo
2002

"Não desanimes. Geralmente é a última chave do molho a que abre a fechadura".

Autor desconhecido

Agradecimentos

À minha família, por todo incentivo dado em todas as fases de minha vida.

A todos os profissionais da Stratus, pela oportunidade, pelo ambiente da empresa e indubitavelmente pelo apoio sem igual na realização deste trabalho, sem o qual ele se tornaria apenas uma árdua tarefa.

Ao professor Mauro de Mesquita Spínola, pela orientação valiosa, atenção, paciência e esforço continuado.

A todos aqueles que de alguma forma participaram e ajudaram no desenvolvimento deste trabalho.

E em especial a Rosvita, por todo apoio nos momentos difíceis.

Sumário

Sumário

Este trabalho versa sobre a análise de requisitos de um sistema de informações, cujo resultado é um modelo que visa atender às necessidades de uma empresa gestora de ativos alternativos. A instituição onde o trabalho foi realizado vem evoluindo e ganhando espaço no mercado nos últimos anos, fazendo com que a maior complexidade e quantidade de informações advindas do seu crescimento tenham que ser classificadas e organizadas de forma lógica, o que a estrutura informática atual não permite. O trabalho apresenta inicialmente o problema e a descrição da instituição, buscando fazer um mapeamento da situação atual. Na seqüência é feita uma discussão acerca dos fundamentos nos quais a análise e o modelo estão baseados. A análise de requisitos do sistema é feita através da Análise Orientada a Objetos e linguagem UML. Após a análise, são feitas considerações a respeito dos resultados e comentários sobre as fases seguintes de desenvolvimento do sistema.

Índice

Capítulo 1

Introdução

1.1 Introdução ao trabalho

Este trabalho versa sobre a análise de requisitos de um sistema de informações para a Stratus, empresa gestora de ativos alternativos.

Inicialmente é introduzido o problema para o qual se propõe a apresentação de uma solução, buscando-se fazer um mapeamento da situação atual. A instituição é descrita para que seja dado não só o pano de fundo, mas também a direção tomada para o desenvolvimento deste trabalho e também o seu contexto relacionado ao estágio.

Na seqüência é feita uma discussão acerca dos fundamentos teóricos nos quais a análise e o modelo são baseados, abordando-se as metodologias mais comumente utilizadas e considerações a respeito de uma linguagem para modelagem.

A análise de requisitos do sistema é feita através da Análise Orientada a Objetos e da linguagem UML para modelagem. As três visões da Análise Orientada a Objetos — o modelo de objetos, o modelo funcional e o modelo dinâmico — são representados, respectivamente, através da linguagem UML com o uso de diagramas de classes, *use cases* e diagrama de seqüências, provendo as partes ortogonais da descrição de um sistema completo.

Após a análise são feitas considerações a respeito dos resultados e comentários sobre as fases seguintes de desenvolvimento do sistema.

1.2 Objetivo do Trabalho

Este trabalho se propõe a promover a análise de requisitos de um sistema de informações, que possibilite o registro, armazenamento e acompanhamento da geração e de trocas de informação da empresa com o ambiente externo, sendo que essas informações devem estar dentro do contexto dos negócios da empresa.

O principal produto do trabalho, o resultado da análise, é um modelo, o qual será a base e prestará auxílio à fase de desenvolvimento final, ou seja, o projeto propriamente dito, e sua implantação.

Esse modelo deverá ser composto por três visões distintas que se fazem sobre as necessidades existentes e que descrevem as necessidades de um sistema completo. Essas necessidades envolvem objetos do domínio do problema e da solução, interações entre objetos e funções a serem executadas pelos usuários.

Espera-se que o fim da análise possa ser seguido pela fase de desenvolvimento da arquitetura do sistema (*design*), utilizando-se dos resultados obtidos. O resultados devem servir de referência para o desenvolvimento e, assim, as idéias desenvolvidas neste trabalho repercutirão nas fases posteriores. Este trabalho dá o primeiro passo para que a fase de *design* possa ser iniciada com rapidez, além de prover uma base sólida sobre a qual todo o desenvolvimento será feito. Evoluções futuras no sistema devem ser bem aceitas, para que o investimento no seu desenvolvimento não seja perdido caso mudanças sejam necessárias ao longo do tempo.

Além disso, há outros fatores que motivam o desenvolvimento deste trabalho. Exemplos desses fatores são o aumento da variedade de canais de comunicação, da freqüência com que informações são trocadas e a necessidade de registro e armazenamento das informações transmitidas para efeito de controle. Além disso, um sistema útil, confiável, seguro, de manutenção facilitada que provê ganhos de produtividade pode trazer à empresa vantagem competitiva no mercado em que atua, melhorando a percepção de clientes e, conseqüentemente, a sua lucratividade.

1.3 O Problema da Organização e Registro

A questão com a qual este trabalho se defronta é a organização das informações de uma empresa gestora de ativos alternativos (termo definido no item 1.4), cuja origem se dá internamente e na troca de informações com o meio externo. Independentemente do meio utilizado para transmissão das informações, o conteúdo deve ser registrado de modo que posteriormente seja possível acesso via computador, obedecendo-se o

contexto dos negócios, recursos, processos e pessoas, visando controle e acompanhamento dessas mesmas trocas e geração de informações.

A organização e o registro coerente de informações advindas da comunicação de uma empresa com o meio externo ou mesmo internamente é um desafio mesmo para instituições de pequeno porte. Diariamente, diversos documentos são criados por funcionários como, por exemplo, estudos setoriais, modelos financeiros, apresentações diversas, etc., e não necessariamente esses documentos têm o conteúdo baseado na mídia aberta como livros, jornais, revistas e canais de televisão. Diversos meios de comunicação privados disponíveis como telefone, fax, correio convencional e e-mail (correio eletrônico), além de outras opções notadamente digitais como, por exemplo, uma teleconferência via Internet, possibilitam transmissão de informações (cada um com suas vantagens e desvantagens a respeito de facilidade de uso, rapidez, confiabilidade e custos envolvidos), tanto com o meio externo como no ambiente interno de uma empresa. Além disso, há a comunicação que se dá de forma presencial, como no caso de reuniões entre pessoas que, de imediato, não faz uso de dispositivos eletroeletrônicos ou de outros serviços para registro. Assim, essa variedade de canais e a possibilidade de se registrar o conteúdo dessas informações de maneira desestruturada (sem uma base centralizada de dados) pode causar problemas futuros, como perda de informações, menor produtividade, agilidade, lucratividade e insatisfação de clientes.

Dentro de muitas empresas a geração de documentos faz uso do computador. Programas da família Office, da Microsoft, como o Word, Excel, PowerPoint e Access são utilizados, respectivamente, na produção de documentos texto, planilhas de cálculo, apresentações diversas e bancos de dados. Esses documentos são disponibilizados de várias formas para outras pessoas, e não ficam armazenados apenas no microcomputador da pessoa que os criou. Pode-se ter que disponibilizá-los para conhecimento de terceiros (inclusive pessoas que não trabalham na empresa), tendo que enviá-los através de e-mail, anexá-los ao restante da documentação de um projeto em desenvolvimento ou mesmo associá-los ao cliente que os solicitou, por exemplo, com a finalidade de gerar um histórico para consulta futura e assim obter um melhor entendimento de suas necessidades. Deste modo, existem diversas finalidades para

esses documentos e é necessário que uma empresa que faça uso desses programas não se restrinja apenas aos limites de um computador pessoal ou à rede interna.

Para a maioria dos canais de comunicação de uma empresa há certa facilidade de se armazenar as informações que foram trocadas entre duas pessoas. Faxes, cartas, brochuras, etc. (entes físicos) ou e-mails (entes digitais), por exemplo, são documentos que contêm a mensagem destinada ao receptor e são a própria entidade a ser armazenada, ou seja, os próprios faxes e cartas são colocados em pastas de arquivos físicos e e-mails em diretórios de servidores de e-mails. É possível ainda digitalizar cartas e faxes para criar arquivos digitais e colocá-los em diretórios para armazenagem. No entanto, o que freqüentemente acontece é a pouca prioridade dada à organização de documentos e diretórios ou, até mesmo quando há preocupação com a organização, pode haver inviabilidade em associá-los a mais de uma entidade relacionada à empresa, como, por exemplo, associar uma apresentação simultaneamente a um cliente, a um projeto e a um funcionário, com uso da estrutura de arquivos e diretórios. Além dessa situação, pode haver na empresa falta de espaço físico para arquivamento (em especial no caso de correio convencional ou faxes físicos), escassez de tempo para localização do local adequado, correndo-se o risco de arquivamento incorreto e somado à dificuldade de monitorização adequada dos arquivamentos realizados por diversos indivíduos envolvidos.

Além dos meios em que o registro é relativamente fácil, como cartas, faxes e e-mails, resta ainda a transmissão de informações trocadas por comunicação oral como em reuniões, telefonemas, etc. É preciso que informações trocadas dessa forma sejam registradas para que o conteúdo não se perca.

A comunicação feita por intermédio de mídias (telefone, rádio, televisão, etc.) pode ser registrada e armazenada com relativa facilidade quando se dispõe de aparelhos apropriados, mas também podem enfrentar problemas, tanto em tecnologias analógicas como digitais. Os aparelhos que fazem registros de forma analógica, no entanto, não possuem diversas vantagens encontradas na forma digital de registro atualmente. Uma delas é a possibilidade de integração facilitada a outras mídias também digitais, como celulares, agendas eletrônicas e mesmo microcomputadores, ainda que haja a princípio

padrões de comunicação diferentes, o que pode ser contornado com softwares tradutores. Além disso, a tecnologia digital é vantajosa quando se trata de espaço físico de armazenamento e também pelo fato de ser a tendência mais aceita para comunicação e transmissão de dados para o futuro. Ainda assim, mesmo que se considerem apenas aparelhos digitais, o registro de comunicações orais pode sofrer dos mesmos problemas citados para o caso de cartas, faxes e e-mails, quando não há uma estrutura que permita estruturação das informações de forma apropriada.

1.4 A Empresa

A empresa na qual o autor realizou o período de estágio e onde este trabalho se desenvolveu é uma gestora de ativos alternativos chamada Stratus Investimentos Ltda., denominada "Stratus" doravante neste trabalho. A Stratus é uma empresa do ramo de *Venture Capital/Private Equity* com atuação focada em projetos de investimento em médias empresas brasileiras e novos negócios, com alto potencial de crescimento. A principal atividade da empresa é adquirir participações em empresas de capital fechado, ajudá-las a crescer para depois vender a participação acionária e realizar os lucros. Essa atuação é viabilizada através da gestão de instrumentos de investimento de longo prazo (3 a 5 anos) no modelo dos fundos tradicionais de *Venture Capital/Private Equity*. Além disso, pode haver outros tipos de investimento, como em fundos de mezanino (nos quais o tipo de capital está a associado a dívidas e o patrimônio líquido). Pelo fato de a empresa não gerir ativos tradicionais, como ações de empresas abertas ou títulos de renda fixa, a Stratus é denominada gestora de ativos alternativos.

Adicionalmente, a Stratus atua no mercado assessorando empresas em processos de fusões e aquisições, além de prestar consultoria estratégica. Tal frente de negócios é denominada internamente por *Advisory*.

A equipe da empresa possui sócios-diretores, analistas de negócios e estagiários, num total de 9 pessoas. Conta também com o suporte de secretárias.

Desde a fundação da empresa em 1999 até os dias de hoje é significativo o aumento das trocas de informações da empresa, tanto internamente como com o meio exterior. A

mudança de porte da instituição, com aumento do número de pessoas em sua equipe, que antigamente possuía 2 pessoas para as atuais 11 pessoas, a recente captação de recursos para o primeiro fundo de investimentos, o contato intenso com seus investidores e o maior número de negócios originados e em processo de análise caracterizam a mudança ocorrida. Dentro desse contexto prevê-se um crescimento ainda maior da atuação da empresa dentro do mercado, com um fluxo maior de informações, fazendo-se necessário o desenvolvimento de um sistema adequado e que compreenda as frentes de negócios da empresa.

1.4.1 Venture Capital e Private Equity

LEVIN (1998) assinala que *Venture Capital* e *Private Equity* são expressões de utilização intercambiável, porém *Private Equity* é um termo mais relacionado ao tipo de capital empregado enquanto que o termo *Venture Capital* relaciona-se antes de tudo à atividade.

Venture Capital é uma modalidade de investimento no qual grandes investidores, através dos profissionais de *Venture Capital*, aplicam capital em empresas promissoras por meio de compra de participação acionária. Os profissionais de *Venture Capital* planejam e executam investimento do tipo *Private Equity*, incluindo projetos de investimentos em empresas novas (*start-ups*), crescimento e desenvolvimento de patrimônio, aquisições baseadas na gestão (chamados de *management buy-out*) e reestruturações.

Os fundos de *Private Equity* em sua maioria são constituídos em acordos contratuais privados entre investidores e gestores, não sendo oferecidos abertamente no mercado e sim através de colocação privada. Além disso, empresas tipicamente receptoras desse tipo de investimento ainda não estão no estágio de acesso ao mercado público de capitais, ou seja, não são de capital aberto, tendo composição acionária normalmente em estrutura fechada.

O termo '*Venture Capital*' é também comumente utilizado para definir o financiamento de novos negócios (*start-ups*) em transações que envolvem diretamente um grupo empreendedor, detentor de um projeto ou idéia, e um grupo provedor de capital.

1.4.2 Evolução Histórica

LEVIN (1998) indica que a atividade de *Venture Capital* tomou impulso com a Revolução Industrial, tornando-se a opção de financiamento para os negócios não cobertos pela atuação dos bancos, porém sendo inicialmente praticada de forma amadora por famílias muito ricas ou capitalistas que financiavam empreendedores amigos.

Sua institucionalização ocorreu com o aumento do número de empresas e de suas necessidades de financiamento como, por exemplo, para a abertura de ferrovias ou para carregamentos marítimos de grãos para áreas em colonização. Foram os *merchant banks* ingleses que passaram a desempenhar a atividade de forma mais profissionalizada, financiando a revolução industrial americana e exportando para os Estados Unidos esse modelo de investimento.

Na década de 40 iniciou-se a profissionalização do *Venture Capital* também nos Estados Unidos, e a aprovação da "Lei dos Pequenos Negócios" em 1958 trouxe o reconhecimento da indústria de *Venture Capital* como atividade financiadora independente e regulamentou a participação dos bancos no setor.

Além de bancos e de indivíduos com alto poder de investimento, iniciou-se em 1970 nos Estados Unidos uma crescente alocação para *Private Equity* a partir de fundos de pensão e de reservas de capital doado a universidades. Com o grande impulso da década de 70, a atividade de *Venture Capital* atingiu seu estágio atual de independência e profissionalização. Desde então o modelo difundiu-se pelo mundo e se expandiu, com a estruturação em fundos.

1.5 O Estágio

O estágio desenvolvido pelo autor na empresa descrita teve início em 9 de abril de 2001. Durante o período de trabalho o autor pôde desenvolver diversas habilidades como a modelagem financeira de empresas, análise de Plano de Negócios (*Business Plans*), pesquisa de mercados, além de se envolver com o relacionamento de investidores, decisões estratégicas, discussão e análise de projetos de investimento.

No entanto, uma outra atividade do estágio tomou corpo com a identificação das necessidades relacionadas ao registro e organização de informações resultantes de trocas de informações internamente e com o meio externo, como explicitado no item 1.3. Nesse sentido, a empresa apoiou e incentivou o desenvolvimento deste trabalho de formatura para que as necessidades fossem mais bem estudadas.

1.6 Engenheiro de Produção e Tecnologia de Informação

O Engenheiro de Produção formado pela Escola Politécnica da Universidade de São Paulo é um profissional demandado pelo mercado de trabalho nos mais distintos setores. Essa abrangência inclui o setor de tecnologia de informação, por sua visão sistêmica, fundamentada e estruturada.

Durante o curso, o aluno adquire conhecimento e desenvolve capacidade raciocínio para analisar, modelar e projetar diferentes sistemas, sejam eles de caráter mecânico (devido à própria orientação do curso no caso da Escola Politécnica) ou financeiro, entre outros. Disciplinas como Sistemas de Informação I auxiliam o aluno a adquirir habilidades específicas no desenvolvimento de sistemas de informações e, além disso, a vasta gama de disciplinas com conteúdo matemático fornecem uma base que viabiliza estudos complexos, que podem envolver cálculo, lógica, estatística e simulações matemáticas.

Deve-se ainda reforçar a importância da visão sistêmica, habilidade que o aluno desenvolve durante o curso. O Engenheiro de Produção é um profissional que se por um lado entende problemas e situações pontuais às quais é exposto, por outro é capaz

de olhar para ele como um todo, avaliando e estimando em que medida os eventos pontuais estão inseridos dentro de um contexto mais amplo. Considera também possíveis interferências no todo ao solucionar os problemas individuais. Para o desenvolvimento de sistemas, é imprescindível que o analista não só saiba posicionar o sistema dentro da empresa, analisando requisitos e fatores operacionais isolados, mas que ele avalie as necessidades do sistema frente elementos externos como, por exemplo, a diferenciação da empresa frente à concorrência de outras empresas e conseqüente melhora da percepção do mercado sobre a qualidade dos serviços prestados.

Capítulo 2

Situação Atual

Para um melhor entendimento da organização atual da Stratus, este capítulo descreve a estrutura de negócios da empresa, bem como sua estrutura de informática.

2.1 Estrutura de Negócios da Empresa

Retomando a descrição contida no item 1.3, pode-se ter uma visão macro das duas principais frentes de negócios da empresa, como mostra a figura 2.1.

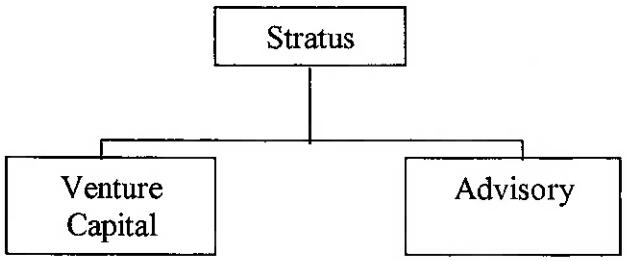

Fig. 2.1 - Estrutura de negócios (elaborado pelo autor)

Cada frente de negócios engloba elementos importantes que norteiam o trabalho desenvolvido pela empresa. Na frente de *Venture Capital* existem os *investidores*, que são os verdadeiros clientes da empresa, em torno dos quais se desenvolvem *projetos de investimento*, cada um dos quais representando todas as ações ou atividades relacionadas a um investidor. Os investidores disponibilizam recursos necessários aos investimentos e sua satisfação se baseia, em princípio, no tratamento dispensado a eles por parte da Stratus, mas principalmente no retorno obtido em cima do capital investido. Além disso, também são desenvolvidos internamente *projetos de Venture Capital*. Em outras palavras, esses projetos de *Venture Capital* representam todas as ações ou atividades relacionadas às empresas que receberão ou receberam aporte de capital, caracterizando o fechamento de um negócio de *Venture Capital*. Essas empresas são, na prática, o meio que a Stratus utiliza para fazer com que o capital investido adquira valor maior e, ao fim do período de investimento, remunere o capital aplicado pelo investidor.

Um conjunto de projetos de *Venture Capital* (VC) e os respectivos investidores (podendo haver um único investidor ou projeto de *Venture Capital*) inserem-se conjuntamente dentro do que se denomina *processo*. Processo é uma denominação que, na prática agrupa determinados projetos que possuem características semelhantes e que podem receber investimento de um mesmo fundo pelo fato de, por exemplo, pertencer a um determinado setor da indústria, além de investidores dispostos a fornecer capital a esses projetos. A Figura 2.2 ilustra essa forma de organização.

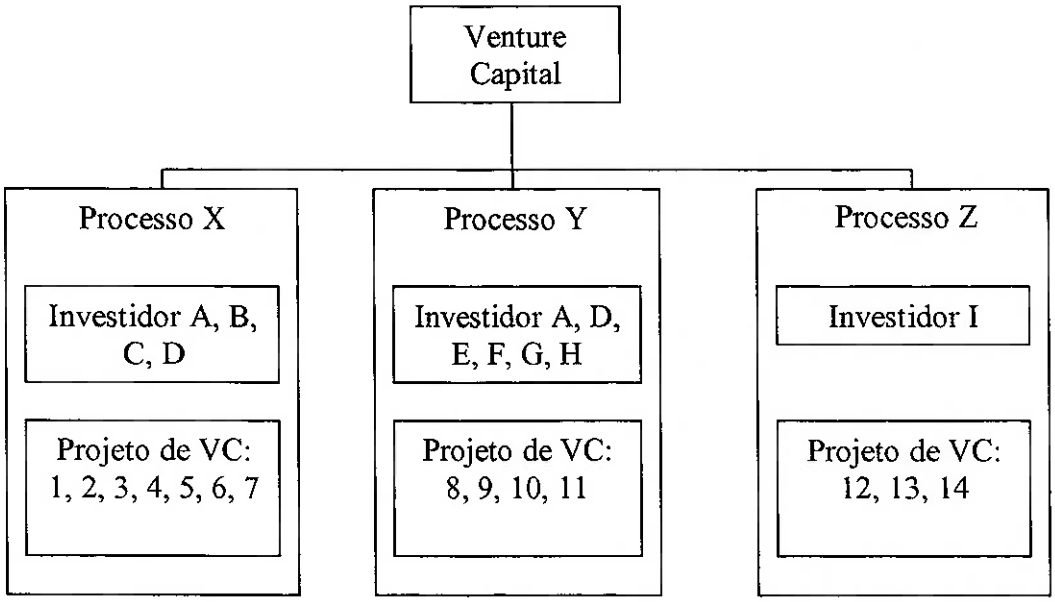

Fig. 2.2 - Frente de *Venture Capital* e seus processos (elaborado pelo autor)

Para a frente de Advisory há a figura da *empresa cliente* que solicita serviços da Stratus de consultoria estratégica ou intermediação para fusão ou aquisição. Em torno da empresa cliente estabelece-se um *projeto de Advisory*, que representa todas as ações ou atividades relacionadas a um único cliente. Há ainda a figura das empresas em prospecção, que podem ser adquiridas por parte da empresa cliente quando esta se situa do lado comprador, ou empresas que podem adquirir a empresa cliente, quando esta se situa do lado vendedor, como ilustra a figura 2.3.

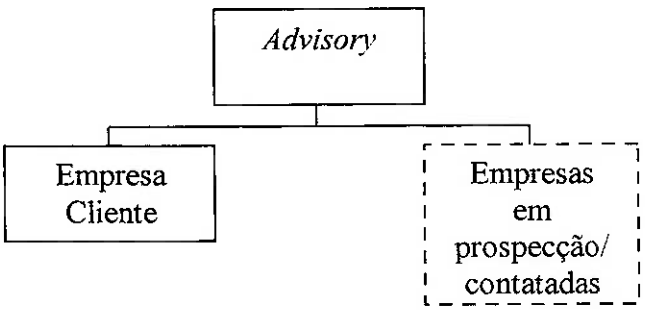

Fig. 2.3 - Frente de *Advisory* (elaborado pelo autor)

2.2 Estrutura de informática

2.2.1 Hardware

A Stratus possui 12 microcomputadores pessoais, conectados em uma LAN (*Local Area Network)*, com o auxílio de um servidor de rede. Além dele, a Stratus possui um servidor para acesso à Internet, no qual está instalado um *firewall*, que protege a rede interna contra a entrada de intrusos.

Os microcomputadores, dependendo do usuário, têm velocidade do *clock* variando entre 233 MHz e 800 MHz e tamanho de memória RAM variando entre 128 MB e 256 MB. Esses microcomputadores possuem um disco rígido com acesso exclusivo por parte do usuário, e tem tamanho entre 3 e 10 GB. O servidor de rede possui também um disco rígido, com tamanho de 20 GB, sendo uma partição destinada ao uso compartilhado, com acesso liberado a todos usuários, que é usada como repositório de documentos, e também uma segunda partição com acesso exclusivo e individual.

O servidor da LAN tem *clock* de 450 MHz e disco rígido de capacidade de 25 GB e memória RAM de 256 MB. O servidor para acesso à Internet possui *clock* interno de 800 MHz, disco rígido de 10 GB e memória RAM de 128 MB.

Além disso, há outros periféricos disponíveis, como um scanner e um aparelho de fax, que só tem conexão com linha telefônica e não com a LAN.

A figura 2.4 ilustra os elementos mais importantes da estrutura.

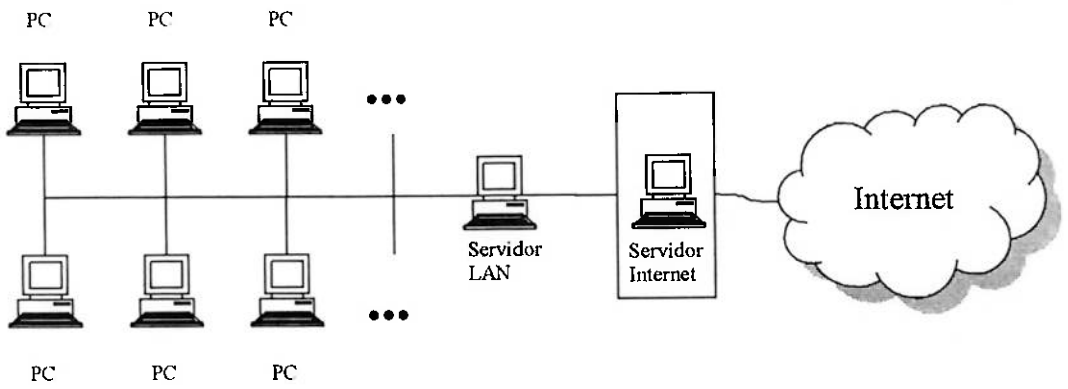

Fig. 2.4 – Estrutura de Hardware (elaborado pelo autor)

2.2.2 Software

Cada microcomputador usa como sistema operacional o Microsoft Windows 98 nos quais estão instalados softwares da Microsoft, como o Office XP 2002, que inclui o Microsoft Word, Microsoft Excel, Microsoft PowerPoint. Alguns microcomputadores possuem também o Microsoft Access 2000.

O servidor da rede roda com Windows NT Server versão 4.0 e possui instalado o Exchange Server 5.5 para troca de e-mails, além de outros aplicativos. O servidor para Internet roda o software Linux versão 7.1 e softwares de segurança, como um *firewall*.

2.3 Organização

No contexto das duas principais frentes de negócios, *Advisory* e *Venture Capital*, e para os processos desta última, são realizadas com freqüência diferentes trocas de informações. Faz-se contato com investidores de diferentes processos, com

empreendedores dos projetos de *Venture Capital* e com empresas clientes de *Advisory* para obtenção de informações a respeito das respectivas empresas. São feitos telefonemas, e-mails, faxes e correspondência de correio tradicional são enviados e recebidos, além de reuniões realizadas com investidores, empreendedores e funcionários, e ainda outras que são contatadas para obtenção de informações no contexto de um processo.

A comunicação interna também se dá utilizando-se os mesmos canais de comunicação, à exceção do envio de faxes e da correspondência por correio. Esta última é substituída pela circulação manual de documentos.

Considerando-se a comunicação realizada com investidores, empreendedores e empresas clientes com auxílio de meios cuja informação é contida em meio material, o armazenamento é feito utilizando-se arquivos físicos que com uma primeira macro-divisão de assuntos. Em cada arquivo faz-se uso de separadores que dividem as pastas presentes num mesmo arquivo. Assim, percebe-se uma divisão genérica em três níveis diferentes (arquivos, separadores e pastas).

No caso da informação digital (basicamente constitui-se de documentos gerados por computador), os e-mails são deixados no servidor de e-mails e em diretórios no disco rígido do computador pessoal, ficando a organização a critério do usuário, podendo-se criar diretórios digitais que separam a correspondência eletrônica em diversos níveis.

Tratando-se de documentos produzidos pelo usuário através de seu computador pessoal, como planilhas do Microsoft Excel ou documentos do Microsoft Word, o armazenamento é feito no disco rígido do microcomputador do usuário ou no disco rígido do servidor da rede na partição dedicada a servir como repositório de documentos. Ao dispor o documento no disco rígido do servidor da rede, o usuário deve ter em mente a estrutura de diretórios existente e a possibilidade de que o documento seja visto e alterado por terceiros. O usuário tem liberdade de escolher o diretório ao qual pode destinar o arquivo digital, podendo eventualmente criar novos diretórios se assim julgar conveniente. Ao fazer uso do disco rígido do seu microcomputador, a organização fica a seu critério.

Em geral, a escolha de onde colocar o arquivo digital leva em conta principalmente se ele já se encontra em estágio final e se pode ser considerado uma versão oficial para dispô-lo na partição do disco rígido do servidor aberta aos outros usuários, pois se o arquivo digital for colocado no disco rígido do computador pessoal, não é permitido a terceiros acesso aos arquivos do usuário pela rede. Os arquivos existentes no disco rígido dos usuários são em geral versões inacabadas de arquivos digitais, usadas para se chegar a uma versão final, que é colocada no disco rígido do servidor.

O capítulo 3 fará considerações sobre os fundamentos teóricos deste trabalho, sendo que o problema a ser resolvido será novamente abordado nos capítulos seguintes.

Capítulo 3

Fundamentos Teóricos

3.1 Metodologias de análise

A análise do sistema, segundo Rumbaugh (1995), deve incluir informações significativas sob a perspectiva do mundo real e apresentar a visão externa do sistema. O modelo resultante da análise deve ser compreensível para o cliente do sistema e para os desenvolvedores finais, devendo fornecer uma base útil para evidenciar os requisitos realmente necessários do sistema, conforme as necessidades apresentadas. Neste ponto, é diferente de um projeto de sistema que considera a implementação em computador, o que obriga que o projeto seja razoavelmente eficiente e de codificação prática.

As metodologias de análise a seguir são as mais bem difundidas na literatura da área:

- Decomposição funcional
- Análise estruturada
- Desenvolvimento Estruturado de Jackson (DEJ)
- Análise Orientada a Objetos

Este capítulo discute o uso dessas metodologias para o sistema em análise e as razões para a escolha de uma delas são apresentadas ao longo do capítulo.

3.1.1 Decomposição Funcional

A decomposição funcional busca entender um sistema e interagir com ele através da identificação das diversas etapas e níveis de processamento necessários, de acordo com suas funções.

Segundo Coad e Yourdon (1992), a decomposição funcional pode ser entendida como a soma de funções do sistema, subfunções e interfaces funcionais. É necessário que seja feito um mapeamento completo do sistema e seu ambiente, desde o domínio do problema até as funções e subfunções. Com este método é necessário um entendimento profundo por parte dos analistas para saber exatamente quais os processamentos necessários para atender aos requisitos do sistema.

Assim, utilizando-se este enfoque a compreensão do domínio do problema não é representada tão especificamente e detalhes sutis não podem ser vistos sem um aprofundamento maior. A decomposição funcional encara todo o problema como uma unidade só, buscando como resultado apresentar os níveis de sistema, subsistema, função e subfunção. A dificuldade é que a escolha das funções e subfunções é extremamente difícil, devido ao seu mapeamento indireto e muito instável devido a constantes alterações na parte funcional do sistema, causadas pela variação das necessidades e da capacidade funcional que pode ser oferecida dentro do orçamento e cronograma.

3.1.2 Análise Estruturada

A análise estruturada é uma forma mais elaborada de decomposição funcional, mas Rumbaugh (1994) afirma que ainda é orientada à decomposição de funções. Ela consiste em diversas notações que são apresentadas ao longo da análise e *design* do sistema, mas o vê como fornecendo uma ou mais funções ao usuário e a partir daí as documenta em uma série de etapas que formam os elementos presentes na análise.

Durante a análise, o sistema é descrito logicamente com o auxílio de:

- Diagrama de fluxo de dados
- Especificação de processos
- Dicionário de dados
- Diagramas de transição de estados
- Diagramas de entidade-relacionamento

Os diagramas de fluxo de dados são o foco da análise estruturada. Eles modelam a transformação das informações na medida em que elas fluem por um sistema. Estes diagramas consistem em processos, fluxo de dados, atores e armazenamento de dados. Começando do diagrama de nível mais alto, a análise estruturada divide sucessivamente processos complexos em subdiagramas, até que haja diversos processos simples de implementar. Quando os processos descritos são suficientemente simples, é

feita a especificação dos processos, que podem ser expressas através de tabelas de decisão, pseudocódigos ou outras técnicas.

O dicionário de dados apresenta detalhes que podem não ser abrangidos pelo diagrama de fluxo de dados. No dicionário, características de fluxos, depósitos de dados e o vocabulário são definidos. Diagramas de transição de estado mostram diversos elementos do sistema que podem ter suas características variadas conforme o instante de tempo em que se observa. A maioria dos diagramas de transição de estado mostra processos de controle ou tempo de execução de funções e acesso de dados acionado por eventos.

Os diagramas de entidade-relacionamento ressaltam relacionamentos entre depósitos de dados que normalmente só seriam descritos na especificação de processos.

3.1.3 Desenvolvimento Estruturado de Jackson (DEJ)

O Desenvolvimento Estruturado de Jackson (DEJ) não distingue análise e projeto. Ela reúne ambas as etapas como especificações. É primeiro determinado "o que" e depois "como". O DEJ é especialmente indicado para sistemas onde nos quais o tempo é uma variável importante.

O DEJ, ainda segundo Rumbaugh (1994), inicia-se com uma consideração do mundo real, pois Jackson defende que, apesar do intuito de um sistema ser o de prover funcionalidade, é preciso que seja estudado como se encaixa no mundo real. A descrição do mundo real é feita com o auxílio de entidades, ações e o ordenamento destas ações. A análise de sistemas através da DEJ consiste em seis etapas seqüenciais:

1. Ação das entidades
2. Estrutura das entidades
3. Modelo inicial
4. Função
5. Sincronização do sistema
6. Implementação

Na ação das entidades o analista de sistemas lista as entidades e ações de parte do mundo real, determinadas através do propósito geral do sistema. Nesta etapa, a entrada de dados provém da declaração dos requisitos do sistema.

A estrutura das entidades ordena parcialmente as ações de cada entidade no tempo, enquanto que o modelo inicial descreve como o mundo real se conecta com o modelo abstrato. A função usa pseudocódigos para ilustrar a saída de ações.

A sincronização do sistema determina quanto é permitido ao modelo atrasar em relação ao mundo real. A implementação foca no agendamento de processos e determina processadores para processos.

3.1.4 Análise Orientada a Objetos (AOO)

A Análise Orientada a Objetos (AOO) é, segundo Rumbaugh (1994), uma técnica que utiliza modelos fundamentados em conceitos do mundo real. A estrutura básica é o objeto, que combina a estrutura e o comportamento dos dados em uma única entidade, ao contrário de metodologias convencionais, onde se tem a estrutura de dados e seu comportamento só superficialmente relacionados. Estes objetos são abstrações do mundo real, que podem existir fisicamente ou não, e dividem o problema em partes independentes, cada uma com sua devida função ou grupo de funções dentro do sistema.

A AOO faz uso de três modelos para descrever um sistema:

- Modelo de objetos
- Modelo dinâmico
- Modelo funcional

O modelo de objetos descreve a estrutura estática dos objetos de um sistema e seus relacionamentos, contendo diagramas de objetos, que são gráficos cujos nós são classes de objetos e cujos arcos são os relacionamentos entre as classes.

O modelo dinâmico descreve os aspectos de um sistema que se modificam com o tempo e é usado para especificar e implementar os aspectos de controle do sistema. O modelo dinâmico contém diagramas de estado, mostrando estados e transições entre estados causadas por eventos.

O modelo funcional descreve as transformações dos valores dos dados de um sistema. Esse modelo mostra o fluxo de dados, que representando os cálculos efetuados, os processos e o fluxo de dados.

Os três modelos são partes ortogonais da descrição de um sistema completo e são interligados. O modelo de objetos é o mais importante por ser necessário descrever o elemento que sofre uma mudança, por exemplo, ou se transformando, antes de se descrever quando ou como a mudança ocorre.

3.2 Comparação das Metodologias

Busca-se aqui uma análise comparativa das metodologias descritas com o intuito de se determinar vantagens e desvantagens de cada uma, assim como as metodologias adequadas a cada tipo de sistema.

A decomposição funcional apresenta um menor nível de detalhamento de alguns dos aspectos sutis do problema, mas nem por isso ela pode ser considerada uma metodologia ruim. Em último caso, é apenas necessário fazer um mapeamento mais detalhado dos dados e de seu processamento. Outras metodologias como a análise estruturada e a orientada a objetos também utilizam a decomposição funcional. A diferença é que elas vão mais além, possibilitando uma visão de maior abrangência do sistema.

Na prática, a AOO faz uso de um tipo de decomposição funcional, só que dividida em partes. Ao invés de encarar o sistema como uma entidade única, ela procura dividi-lo em partes menores para uma maior compreensão.

Já a análise estruturada pode até ser vista como uma evolução da decomposição funcional, uma vez que ela incorpora muitas de suas características, mas com uma construção de modelos mais complexa e elaborada. A análise estruturada, juntamente com a análise orientada a objetos, utiliza as três visões ortogonais de um sistema, que são os modelos de objeto, dinâmico e funcional.

A diferença entre elas é que na análise estruturada o modelo funcional é dominante, o modelo dinâmico é o segundo mais importante e o modelo de objetos o menos importante. Na AOO, o modelo de objetos é o mais importante, seguido pelo dinâmico e pelo funcional.

A análise estruturada organiza um sistema em cima de procedimentos, e a AOO organiza um sistema voltado para objetos do mundo real, ou objetos conceituais que existem na visão de mundo real do analista. Isto faz com que, na análise estruturada, a maioria das mudanças nos requisitos sejam mudanças relacionadas a funções, fazendo com que as mudanças sejam grandes tarefas, com alto teor de complexidade. Já na AOO as mudanças de função podem ser facilmente incorporadas na estrutura ao se mudar ou adicionar operações, o que deixa a estrutura de objetos intacta.

A análise estruturada cria uma fronteira bem definida do sistema com o mundo real, pois a estrutura da análise vem em parte da sua fronteira de sistema. Esse tipo de análise, por ser voltada a procedimentos, mostra-se inadequada para lidar com bancos de dados, uma vez que é difícil de se integrar uma linguagem de programação organizada por funções com bases de dados organizadas de acordo com o tipo de informação.

A análise estruturada apresenta, no entanto, algumas vantagens, como o fato de os programadores terem a tendência de pensar em termos de funções, tornando as metodologias baseadas em fluxo de dados mais fáceis de se aprender. A análise estruturada é mais útil para sistemas onde as funções são mais importantes e complexas que os dados com que lidam em si, pois a análise supõe que isto ocorre com freqüência. No mais, a análise estruturada foi o primeiro método formal usado para o

desenvolvimento de sistemas e softwares, fazendo com que seu uso já esteja difundido e as mostras de sua utilidade disponíveis.

O desenvolvimento estruturado de Jackson (DEJ) pode ser defendido por alguns autores como sendo orientada a objeto. Apesar de também começar com considerações do mundo real, a DEJ identifica poucas entidades (objetos) e não se aprofunda no desenvolvimento de sua estrutura.

A DEJ é complexa e difícil de ser totalmente entendida, sendo didaticamente mais obscura que as demais metodologias discutidas. Isto se deve ao fato de a DEJ depender fortemente de pseudocódigos, que são muito menos ilustrativos que modelos gráficos. Não se pode esquecer que parte da complexidade da DEJ também está relacionada ao fato de ter sido desenvolvida especialmente para se tratar de problemas difíceis de tempo real, pois sua complexidade se mostra desnecessária para a solução de problemas comuns ou simples.

A DEJ dá grande ênfase a ações, enquanto que na AOO esta ênfase está voltada para os atributos. As ações na DEJ se assimilam muito com os relacionamentos da análise orientada a objetos, porém Jackson considera "atributo" uma definição confusa e prefere evitá-la.

A DEJ é muito útil para ser utilizada em aplicações onde o tempo e a sincronização são importantes, como aplicações com muitos processos simultâneos que devem ser sincronizados, softwares de tempo real e programação paralela de computadores. Ela não é apropriada para aplicações com análise de alto nível e não é eficiente para abstração e simplificação de características de sistema. Embora trate detalhes meticulosamente, não ajuda o analista a entender a essência de um problema.

A DEJ também não é a metodologia mais apropriada para lidar com bases de dados, uma vez que o projeto de bases de dados é mais complexo do que o assumido por Jackson. O fraco foco em entidades e atributos faz com que a DEJ seja uma técnica pobre para o tratamento de bancos de dados. As diversas abstrações de processos

também tornam a DEJ confusa para lidar com softwares convencionais que rodam em um sistema operacional.

Já a AOO foca sua atenção na estrutura de dados ao invés das funções a serem exercidas. Esta mudança no enfoque com relação a muitas das outras metodologias dá ao processo de desenvolvimento uma base sólida através da criação de um único conceito ao longo de todo processo de criação de um sistema: o conceito de objeto. Todos os outros conceitos apresentados na metodologia são organizados ao redor dos objetos, como atributos, relacionamentos, operações, fazendo com que as informações adquiridas durante a análise não sejam alteradas ou perdidas durante as fases seguintes de *design* e implementação.

A fronteira com o mundo real no caso da AOO é menos definida, sendo mais fácil de se entender o sistema do que com metodologias orientadas a função. Para isto, é necessário apenas se adicionar objetos e seus relacionamentos próximos à fronteira para representar algo que anteriormente existia somente "do lado de fora". É importante ressaltar que ao se organizar um sistema por objetos, a estrutura de dados e seus relacionamentos tornam-se muito menos vulnerável a mudanças nos requisitos que as metodologias orientadas a função. A AOO recebe uma mudança de forma mais amigável e extensível.

A AOO também integra de maneira mais natural bancos de dados e linguagens de programação. O objeto pode modelar tanto o banco de dados quanto a estrutura de programação.

O processo de desenvolvimento usando-se a AOO é mais uniforme que nas outras metodologias, uma vez que o modelo de objetos desenvolvido durante a análise é utilizado para o *design* e implementação, e o trabalho se resume em refinar o modelo em níveis progressivamente mais detalhados, pois não há mudança nas notações utilizadas.

O método de desenvolvimento da AOO também é iterativo, ao invés de seqüencial, pois ocorrem retornos constantes ao que já foi desenvolvido, para que partes do trabalho sejam refinadas até se atingir o nível de detalhamento desejado.

Pode-se observar um resumo das principais características das metodologias mencionadas na tabela 3.1.

Metodologias Caracterist.	**Decomposição Funcional**	**Análise Estruturada**	**Desenv. Estrut. de Jackson (DEJ)**	**Análise Orientada a Objetos (AOO)**
Enfoque	Função	Função	Tempo real	Objetos
Visões Ortogonais de Sistema	Não	Sim 1. Modelo Funcional 2. Modelo Dinâmico 3. Modelo de Objeto	Não	Sim 1. Modelo de Objeto 2. Modelo Dinâmico 3. Modelo Funcional
Vantagens	▪ Fácil entendimento da metodologia de análise ▪ Ideal para sistemas simples com poucas responsabilidades ▪ Uso difundido	▪ Programadores pensam em termos de função ▪ Ideal para sistemas onde as funções são mais importantes e complexas que os dados ▪ Uso difundido	▪ Aplicações em atividades onde o tempo e a sincronização de processos sejam importantes	▪ Conceito de objeto permanece durante todo o desenvolvimento ▪ Fronteira do sistema flexível ▪ Estrutura menos abalada em mudanças ▪ Integra bancos de dados de maneira natural
Desvantagens	▪ Baixo nível de detalhamento e entendimento das sutilezas do sistema	▪ Mudanças nos requisitos são atividades complexas ▪ Fronteira do sistema é fixa	▪ Não prega alto nível de detalhamento e entendimento de um problema ▪ Ruim para simplificação de um sistema ▪ Ruim para bancos de dados	▪ Complexa demais para sistemas onde as funções sejam mais importantes que os dados ▪ Não é útil para sistemas com responsabilidades muito limitadas

Tabela 3.1 – Características das Metodologias (elaborado pelo autor)

3.2.1 Escolha da Metodologia para a Análise

Para verificar a metodologia mais adequada, considere uma matriz de decisão para especificar através de uma pontuação a conveniência de cada metodologia para a análise e a construção de um modelo.

A matriz de decisão considerada é baseada em (1) enfoque, (2) vantagens e (3) desvantagens de cada metodologia para análise, descritas na tabela 3.2, à luz de critérios (requisitos para o sistema) necessários para o modelo no problema apresentado. Para cada metodologia será atribuída uma pontuação com valores 0, 1, 3 ou 5, sendo que os valores maiores demonstram maior aderência e maior adequação de enfoque e de vantagens da metodologia à análise, e com respeito a desvantagens, a pontuação será de 0, -1, -3, -5, sendo que quanto mais negativo o valor, mais a análise será comprometida com o uso da metodologia.

Os critérios levados em conta para a avaliação de enfoque, vantagens e desvantagens são:

- Interface única que permita fluxo de informações sem redundância;
- Centralização das informações arquivadas, com fácil acesso e localização;
- Estrutura que permita modificações futuras sem que as informações ou operações sejam afetadas;
- Sistema que abranja banco de dados e tratamento estruturado das informações.

Quanto ao enfoque, a decomposição funcional e a análise estruturada possuem foco na função, limitando a abrangência da análise e do modelo resultante dela, se comparadas à análise orientada a objetos. Pelo fato de nesta última poder-se incluir conceitos do mundo real, o modelo resultante torna-se mais compreensíveis e de entendimento mais fácil, ainda que o nível de detalhe seja maior.

Com relação às vantagens, o modelo não extrairia do desenvolvimento estruturado de Jackson a sua mais importante vantagem, que é o seu uso para aplicações de tempo real, já que o sistema em questão não será voltado para sincronização entre suas entidades. A

decomposição funcional e a análise estruturada têm uso difundido e um maior número de pessoas possuem conhecimento sobre elas, o que poderia baratear ou facilitar o desenvolvimento em termos de mão-de-obra. Análise orientada a objetos, entretanto, possui a vantagem de ter o seu resultado menos abalado em mudanças, o que significa que evoluções e novas necessidades para o modelo ou sistema (na fase de desenvolvimento ou até quando já implementado) podem ser identificadas e mais facilmente incorporadas, além de permitir uma integração mais fácil com banco de dados (graças ao foco dado à estrutura).

Já a respeito das desvantagens, ambos o desenvolvimento estruturado de Jackson e a decomposição funcional não permitem um detalhamento maior do sistema, com o agravante de esta última ser considerada ruim quando bancos de dados precisam ser adicionados. A análise estruturada também tem a limitação de ser uma metodologia "rígida", pois novos resultados advindos da análise ou novos requisitos não podem ser facilmente incorporados ao modelo ou sistema. Com a análise orientada a objetos, o modelo poderia se tornar um pouco complexa devido à existência de algumas funções.

Deste modo, atribuindo-se as notas de acordo as questões acima levantadas, pode-se construir a seguinte matriz, representada pela tabela 3.2:

Metodologias / Critérios	**Decomposição Funcional**	**Análise Estruturada**	**DEJ**	**AOO**
Enfoque	1	1	0	3
Vantagens	3	3	0	5
Desvantagens	-5	-5	-3	-1
Soma	-1	-1	-3	7

Tabela 3.2 – Matriz de decisão (elaborado pelo autor)

Assim, a metodologia mais aderente frente aos critérios apresentados e escolhida para o desenvolvimento deste trabalho é a análise orientada a objetos.

3.3 O Contexto Recente da Modelagem Orientada a Objetos

Na área de software há várias maneiras de se abordar um modelo. Basicamente, as duas maneiras mais comuns são o uso de uma perspectiva algorítmica ou orientada a objetos. Considerando-se os casos descritos, a decomposição funcional, a análise estruturada e o desenvolvimento estruturado de Jackson são exemplos do primeiro caso e a análise orientada a objetos do segundo.

Booch; Rumbaugh; Jacobson (1998) sugerem que a visão tradicional de análise e desenvolvimento de software leva a perspectiva algorítmica. Nesta abordagem, o cerne de todo software é o procedimento ou função. Esta visão leva os analistas e desenvolvedores a focar em questões de controle e decomposição de algoritmos complexos em outros menores e mais simples. Isso tem suas aplicações, mas essa abordagem tende a produzir sistemas frágeis. Como os requisitos mudam (e eles irão) e o próprio sistema cresce (e ele irá), sistemas construídos com foco algorítmico são difíceis de manter.

A visão contemporânea de desenvolvimento e análise de software tem a perspectiva da orientação a objetos. Nesta abordagem, o cerne de qualquer software é o objeto ou classe. Grosso modo, objeto é uma coisa, um ente geralmente provindo do vocabulário do espaço do problema ou da solução. Uma classe é a descrição de um conjunto de objetos comuns. Todo objeto tem uma identidade (pode-se nomeá-lo ou então distingui-lo de outros objetos), estado (geralmente há dados associados a ele), e comportamento (pode-se fazer coisas com o objeto e ele pode fazer coisas com outros objetos).

Ainda segundo Booch; Rumbaugh; Jacobson (1998), a abordagem da orientação a objetos é decididamente uma prática prevalecente simplesmente porque ela provou seu valor na construção de sistemas na maioria dos tipos de domínios de problemas e abrangendo praticamente qualquer tamanho ou complexidade. Além disso, as linguagens, os sistemas operacionais e ferramentas mais recentes são em alguma medida orientadas a objetos, corroborando ainda mais com a idéia de enxergar o mundo através de objetos.

3.4 Processo e Linguagem

Fowler; Scott (1997) assinalam que, independentemente do método de desenvolvimento ou processo (etapas de desenvolvimento de um software), a linguagem é a parte mais importante e a chave para comunicação. Se deseja-se discutir o *design* com alguém, é a linguagem que o interlocutor precisa entender, não o processo usado para se obter esse *design*.

Não há um único processo para o desenvolvimento de software. Vários fatores associados ao desenvolvimento podem levar a diferentes tipos de processos. Esses fatores incluem o tipo de software a ser desenvolvido (tempo real, sistema de informações, software para computador pessoal), da escala (desenvolvedor único, equipe de desenvolvedores pequena ou equipe com centenas de desenvolvedores) e assim por diante.

Booch (1996) sugere o *Macro Development Process*, cujas características são a flexibilidade de aplicação (tanto no que diz respeito a tipos de software como a não linearidade das etapas do processo), a iteratividade e a possibilidade de incrementação. As diferentes fases desse processo são a conceituação (estabelecimento das necessidades principais), análise (desenvolvimento de um modelo com comportamento desejado), *design* (criação da arquitetura), implementação propriamente dita e manutenção (pós-desenvolvimento).

De forma semelhante, Jacobson; Booch; Rumbaugh (1999) indicam o *Unified Software Development Process*, também iterativo e configurável, com ênfase nos modelos e arquitetura, suportando também técnicas orientadas a objeto. As fases desse processo consistem na concepção (delimitação do escopo do projeto), elaboração (estabelecimento de um plano de projeto e de arquitetura), construção e transição (fornecimento do sistema para os usuários finais).

Tanto as três primeiras fases do Macro Development Process, como as duas primeiras do Unified Software Development Process, englobam as atividades de engenharia do

ciclo de desenvolvimento, enquanto as fases subseqüentes consistem na produção do software propriamente dito.

3.5 A UML

A *Unified Modeling Language* (UML) é uma linguagem padrão para se desenhar modelos de software e independe do processo usado.

A UML é para Booch; Rumbaugh; Jacobson (1998) a linguagem apropriada para a modelagem de sistemas abrangendo desde sistemas de informação empresariais a aplicações *web-based* distribuídas ou até mesmo para sistemas complexos de tempo real. É uma linguagem que inclui todas as visões necessárias para desenvolver e dispor de tais sistemas.

Uma linguagem provê vocabulário e regras de combinação das palavras desse vocabulário com a finalidade da comunicação. Uma linguagem de modelagem é uma linguagem cujo vocabulário e regras focam na representação física e conceitual de um sistema.

A UML é usada para visualizar, especificar, construir e documentar os resultados de um sistema.

3.5.1 Visualização

Há diversos problemas que ocorrem quando não se faz uso de uma linguagem única. Primeiramente a comunicação de modelos conceituais a terceiros que não estão envolvidos diretamente com o projeto tende a ser difícil e, além disso, há elementos de um software que não são facilmente compreensíveis a menos que se construam modelos que extrapolam a linguagem textual de programação (por exemplo, não se consegue verificar facilmente todas as hierarquias presentes em determinada classe através do código do programa). Finalmente, se a pessoa que escreveu o código de determinado software não tiver registrado os modelos em que pensou, esse

conhecimento poderá facilmente ser perdido, sem que a organização possa fazer uso dele no futuro.

A UML resolve esses problemas, pois permite se fazer modelos explícitos que facilitam a comunicação e a compreensão por ser essencialmente gráfica. Por trás de cada símbolo na UML há uma semântica bem definida, de modo que o modelo desenvolvido possa ser interpretado sem ambigüidade também por outros desenvolvedores de software.

3.5.2 Especificação

Especificar significa construir modelos que são precisos, completos e não ambíguos. A UML atende esses requisitos nas decisões que devem ser feitas nas fases de análise, *design* e implementação.

3.5.3 Construção

Os modelos baseados em UML podem ser diretamente usados por uma grande variedade de linguagens de programação. Ela é suficientemente expressiva e não ambígua para permitir a execução direta de modelos, simulação e controle de sistemas.

3.5.4 Documentação

Com a UML é possível documentar a arquitetura de um sistema e todos os seus detalhes, provendo também linguagem para expressar requisitos, testes e modelagem de atividades.

3.6 Finalidade da UML para o trabalho

Para que a modelagem do sistema contemple os três modelos da análise orientada a objetos são usados três diagramas da linguagem UML: diagrama de classes, *use cases*, e diagrama de seqüências.

O diagrama de classes é o tipo de diagrama mais comum e importante na modelagem de sistemas orientados a objeto para prover a visão do modelo de objetos. Os *use cases* especificam o comportamento do sistema através de funções ao nível do usuário. O diagrama de seqüências é um tipo de diagrama que mostra as interações e relações entre objetos, contemplando a visão dinâmica da modelagem de objetos, fechando assim as três visões ortogonais (modelos de objetos, funcional e dinâmico) necessárias para a análise orientada a objetos de um sistema.

A figura 3.3 ilustra a estrutura da análise para a concepção do sistema.

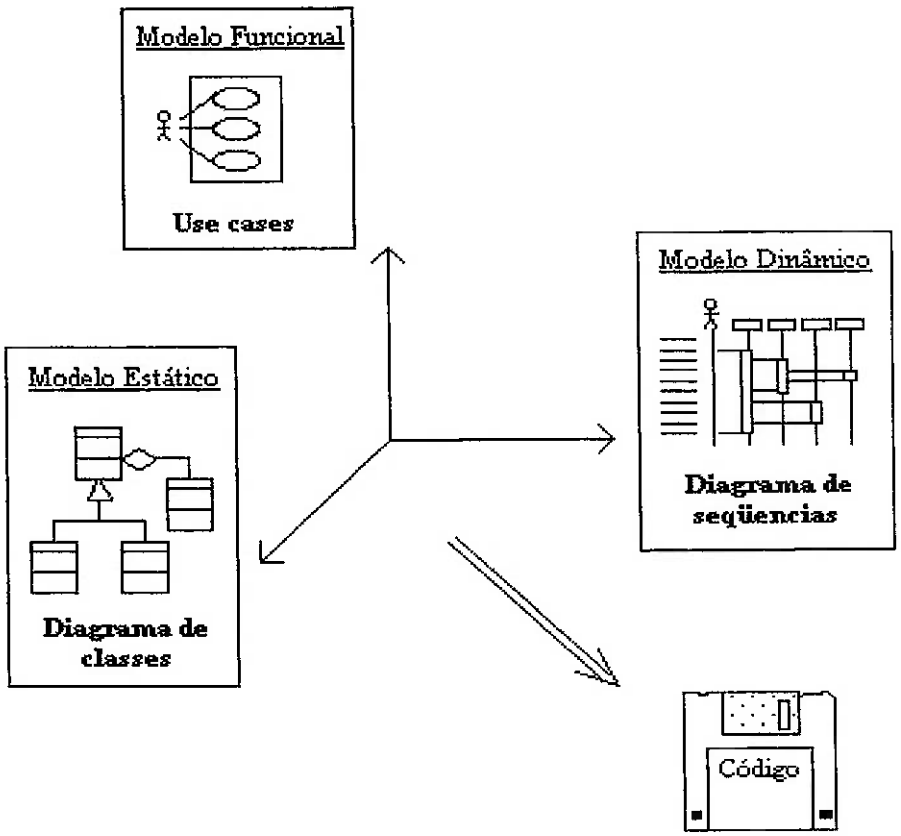

Fig. 3.3 – Estrutura da análise para codificação (elaborado pelo autor)

3.6.1 Use cases

Os *use cases* são diagramas chave para a modelagem do comportamento de um sistema, subsistema ou classe. Um *use case* especifica as funções de um sistema ou parte de um

sistema; é uma descrição de um conjunto de seqüências de ações executadas pelo sistema para permitir um resultado com valor para o usuário.

Use cases tentam capturar o comportamento almejado do sistema sem que se tenha que especificar como esse comportamento é implementado. Eles fornecem aos desenvolvedores de sistemas uma ferramenta que os auxilia a chegar em um entendimento comum com os usuários finais e na validação da arquitetura antes e durante o desenvolvimento do sistema.

A UML fornece uma representação gráfica de um ator e de um *use case*, possibilitando mostrá-lo no contexto com outros *use cases*.

3.6.2 Diagrama de classes

Um diagrama de classes mostra um conjunto de classes, interfaces, além de colaborações e seus relacionamentos. É destinado para se modelar uma visão estática do *design* de um sistema. Esta visão dá suporte primordialmente aos requisitos funcionais de um sistema, ou seja, os serviços que o sistema deve prover aos usuários finais.

Classes são os elementos mais importantes de um sistema orientado ao objeto. Uma classe é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica. Classes são usadas para se constituir o vocabulário do sistema que se está desenvolvendo. Essas classes podem incluir, além do domínio do problema, classes para implementação, elementos de software, hardware ou mesmo elementos puramente conceituais.

A UML provê uma representação gráfica de classes que permite visualizar uma abstração independentemente de qualquer linguagem de programação. Ela também permite enfatizar as partes mais importantes da abstração, que são o nome, atributos e operações.

3.6.3 Diagramas de seqüências

O diagrama de seqüências é um tipo de diagrama de interações. Um diagrama de interações envolve a modelagem de exemplos concretos ou protótipos de classes, interfaces, componentes e nós, entre os quais mensagens são enviadas e recebidas.

O diagrama de seqüências em particular enfatiza a seqüência de mensagens no tempo, mostrando um grupo de objetos que trabalham conjuntamente com envio e recebimento de mensagens, tendo como objetivo a execução de determinadas ações. Essas mensagens podem invocar uma operação, envio de sinal ou mesmo a criação ou destruição de objetos.

Na prática, há uma fusão do diagrama de classes com os *use cases*, onde elementos desses dois diagramas estão presentes para a construção da parte dinâmica do sistema.

A UML provê uma notação para mensagens, permitindo visualizá-las de modo a enfatizar seus nomes, parâmetros (quando houver) e a seqüência em que ocorrem.

Capítulo 4

Use Cases e Requisitos Não Funcionais

4.1 Introdução

O sistema, num contexto abrangente, deve contemplar, além de todas as trocas de informação com investidores, clientes de *Advisory* e empreendedores (cada uma dessas trocas inseridas em diferentes frentes de negócios da empresa e processos internos), as trocas efetuadas do pessoal da Stratus com outros agentes do meio externo à empresa.

Há uma variedade desses agentes que atuam externamente à empresa e que podem ser alvo da análise do sistema. No entanto, dado o estágio inicial do relacionamento da Stratus com esses agentes, as informações intercambiadas com esses agentes não ocorrem com grande intensidade atualmente, fazendo com que essas trocas de informação não sejam prioridade no momento para a análise do sistema.

Assim mesmo, uma visão mais abrangente é mostrada através de *use cases*, os quais permitem ao leitor a identificação de funções a serem executadas por esses diversos agentes no sistema. Para a comunicação associada a investidores, empreendedores e clientes de *Advisory*, no qual este trabalho terá seu foco, *use cases* são executados, com o intuito de destacar os elementos mais importantes do sistema.

Além dos requisitos caracterizados pelos *use cases*, considerados requisitos funcionais, devem ser considerados outros requisitos que não estão diretamente ligados às funções de uso corrente do sistema por parte do usuário. Esses são os requisitos não-funcionais ou de qualidade do sistema, discutidos no contexto do sistema ao fim do capítulo corrente.

4.2 Use cases – casos amplos do sistema

Nesta modelagem pode-se observar os macro-elementos que caracterizam o sistema, que são os atores e as funções desejadas para o sistema, na visão de cada ator. A notação utilizada é mostrada pela figura 4.1.

Figura 4.1 – Notação para *use cases*

Observando-se o diagrama da figura 4.2 percebe-se a separação entre o ambiente em que se situam os funcionários da Stratus ("Pessoal Stratus") e o ambiente externo, e as funções a serem executadas por eles. O Pessoal Stratus é responsável pelo registro das comunicações realizadas com outros atores (e também entre eles mesmos) que geram informações relevantes para armazenamento no sistema e pelo controle dos processos internos através de relatórios de produtividade, enquanto os outros atores, de modo geral, fornecem informações específicas ao Pessoal Stratus, também fazendo uso do sistema.

Para investidores potenciais, as informações a serem registradas são relacionadas à comunicação de relacionamento. Com esses investidores desenvolve-se uma relação de longo prazo com os quais há troca de informações passíveis de registro. Quando há a decisão de investimento por parte dos investidores potenciais, estes se tornam investidores efetivos, com os quais se estabelece uma relação mais próxima e com maior volume de informações trocadas como, por exemplo, a respeito do fundo no qual o investidor aplicou seu capital, valor de cotas, reuniões específicas, etc. No entanto, investidores efetivos não deixam de ser potenciais na medida em que podem investir em outros fundos. Entre os investidores efetivos, no entanto, os investidores pertencentes ao Comitê de Investimentos terão disponíveis no sistema informações mais detalhadas que os outros cotistas como, por exemplo, faturamento das empresas que receberam investimento e andamento das análises de outras empresas na Stratus. Previamente deverá ter havido um cadastro do investidor por parte do Pessoal Stratus, com a inclusão de informações relevantes, para que os registros efetuados das trocas de informações com o investidor potencial possam ser armazenados nos locais corretos.

Do mesmo modo que para investidores potenciais, informações registradas sobre empresas potenciais também estão ligadas ao relacionamento inicial que se mantém com a Stratus. No caso das empresas investidas, o fluxo de informações, assim como no caso dos investidores efetivos, também tende a ser maior, na medida em que a solicitação de informações se torna freqüente e se deseja um maior nível de controle sobre as operações da empresa. Igualmente também deve ter havido um cadastro prévio com informações sobre a empresa.

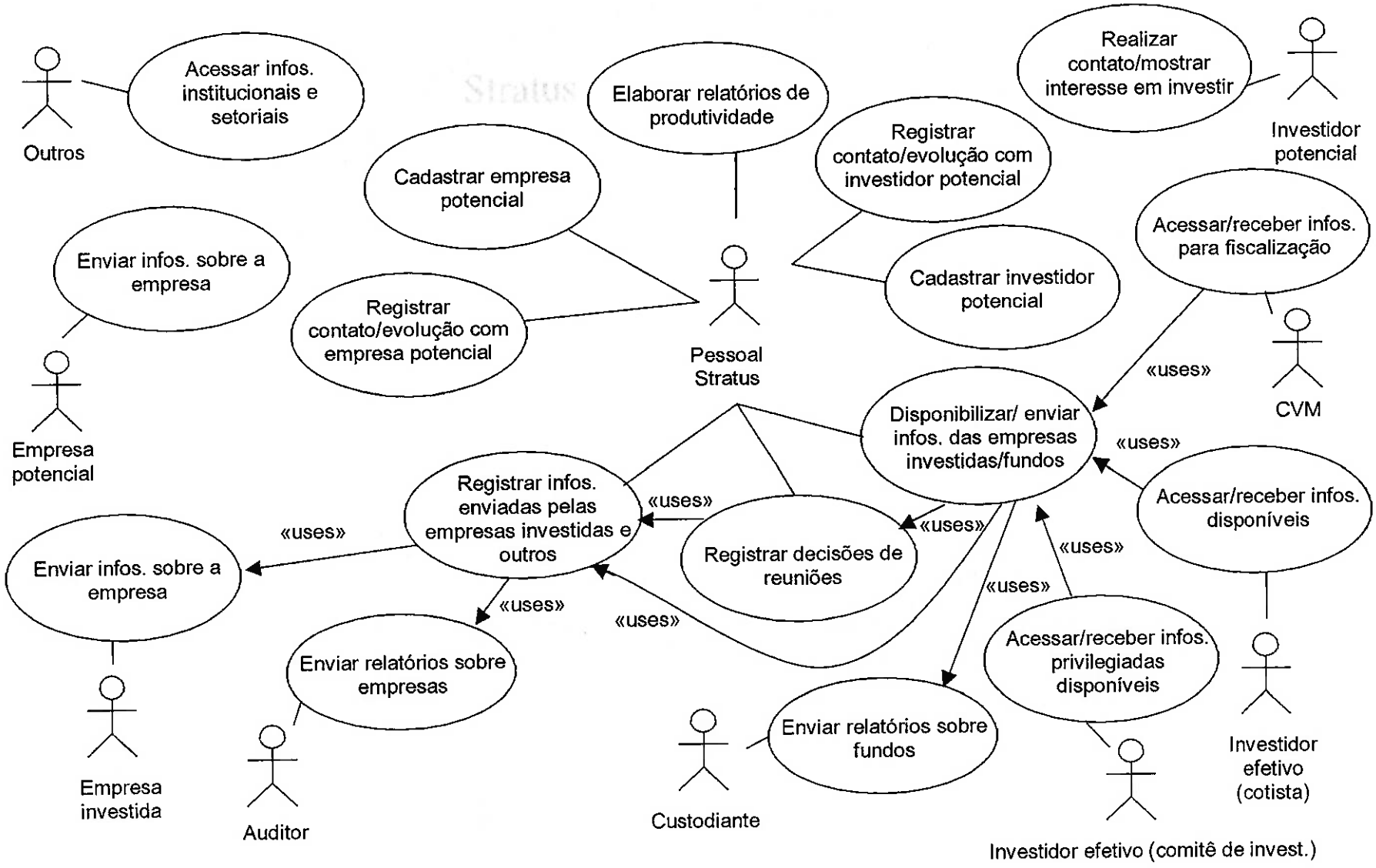

Fig. 4.2 – Use cases – caso amplo

Pode-se notar ainda no diagrama um fluxo de informações que em princípio parte do "Investidor Efetivo" e "CVM" e que termina em "Empresa Investida" e "Auditor". O fluxo, entretanto, origina-se destes dois últimos, pois as setas na notação UML são a representação gráfica do acesso à informação por parte do *use case*, indicando uma ordem aparentemente invertida. Com este fluxo pode-se observar as entradas (por meio da "Empresa Investida", "Custodiante" e "Auditor"), e as saídas relativas ao sistema ("Investidor efetivo" e "CVM").

A empresa auditora ("Auditor") examina as informações contábeis das empresas investidas e envia seu parecer a Stratus, de forma a se obter uma versão legal das informações recebidas anteriormente com a empresa investida. A empresa custodiante ("Custodiante"), responsável pela guarda de títulos e valores do fundo também é responsável pelo envio de informações a respeito de movimentações relacionadas ao fundo de investimento.

A Comissão de Valores Mobiliários ("CVM"), com o papel de fiscalizar administradores de valores mobiliários, deverá receber relatórios relativos aos fundos de investimento, como demonstrativos financeiros, valor patrimonial de cotas, composição de carteira, etc.

O "Outros" presente no diagrama está relacionado a pessoas que desejam adquirir informações a respeito da Stratus, do setor ou obter alguma publicação feita pela empresa. Atualmente essas informações são supridas pela *homepage* da empresa, mas uma integração com o sistema em análise neste trabalho facilitaria a disponibilização dessas informações.

4.3 Use cases – casos em estudo

Nesta modelagem, dos *use cases* em estudo, estão presentes os elementos focados para o desenvolvimento deste trabalho. Esses elementos são representados por atores e *use cases*, os quais mostram as principais funções desejadas para o sistema, na visão de

cada ator, utilizando-se a mesma notação que para o diagrama do caso amplo do item 4.2.

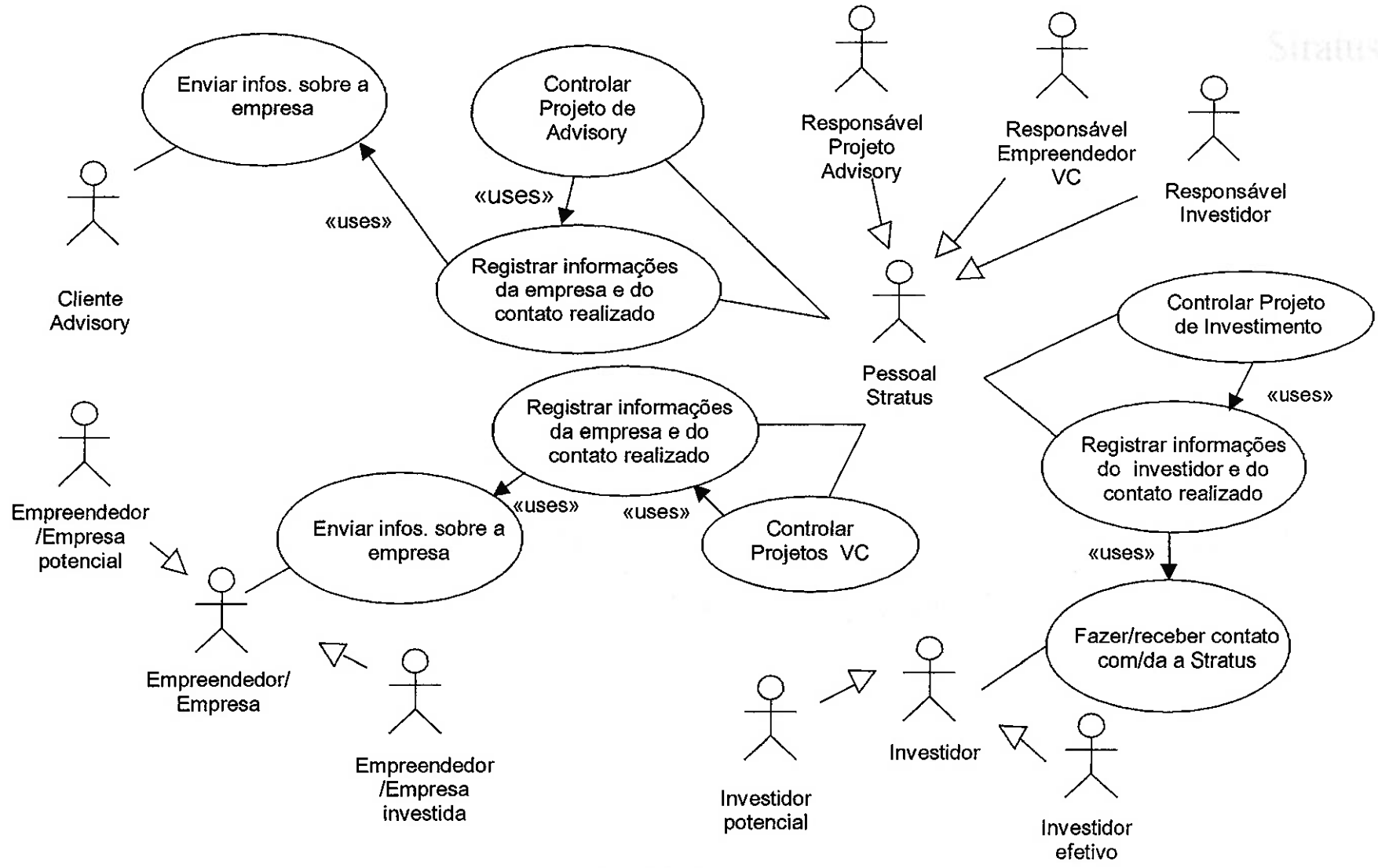

Fig. 4.3: Use cases – casos em estudo

Na figura 4.3 pode-se observar algumas diferenças em relação aos *use cases* dos casos amplos. Nos casos em estudo há maior destaque para as trocas de informação entre a Stratus e investidores, empreendedores e clientes de *Advisory* (estes últimos não estavam presentes no diagrama anterior), além do surgimento de novos tipos de "Pessoal Stratus" e diferenciação das frentes de atuação.

O "Pessoal Stratus" é na verdade uma generalização dos três tipos de responsáveis que existem de acordo com a frente de atuação. Há responsáveis por negócios na frente *Advisory* ("Responsável projeto *Advisory*"), negócios de *Venture Capital* ("Responsável Empreendedor VC") e pelo relacionamento com investidores ("Responsável Investidor"). Na prática pode haver acúmulo de funções como, por exemplo, uma mesma pessoa pode ser responsável por um projeto de investimento. No entanto, um projeto de investimento não pode ter todos os funcionários da Stratus como responsáveis, mas sim uma equipe fixa ou mesmo um só indivíduo, sendo que o sistema deve ser capaz associar corretamente o agente externo (investidores, empresa investida ou negócio de *Advisory*) ao seu(s) responsável(eis).

No diagrama amplo do sistema, a figura do cliente de *Advisory* podia ser considerada como uma empresa potencial. No entanto, há necessidade de se diferenciar a frente de *Advisory* da frente de *Venture Capital.* É fundamental a separação desses dois elementos pois, ainda que sejam as duas atividades mais importantes da empresa, elas são distintas e não há sinergia entre elas, como detalhado no item 2.1.

Pode-se perceber ainda na figura 4.3 que não está mais presente o fluxo do diagrama da figura 4.2. Esse fluxo será estabelecido somente num desenvolvimento posterior, ficando agora em segundo plano.

Tanto para investidores como para empreendedores/empresas, potenciais ou não, ou clientes de *Advisory*, pode-se perceber a semelhança dos *use cases*. Nos três casos, há registro das informações trocadas entre o "Pessoal Stratus" e investidores, empreendedores/empresas e clientes de *Advisory.* Essas informações trocadas deverão ser associadas a uma entidade denominada genericamente de "projeto".

O "projeto" centraliza todas as informações relevantes referentes a um agente externo, tanto sobre a comunicação efetuada diretamente com o agente, bem como sobre atividades internas de manutenção do relacionamento, análises ou decisões a respeito do projeto. Para cada cliente de *Advisory* ou empresa/empreendedor deve haver um único projeto associado, mas investidores poderão pertencer a mais de um projeto.

Esta última definição e outras são explicadas no capítulo seguinte, no qual a identificação de classes, os diagramas de classes e seus relacionamentos permitirão uma leitura estruturada de todos elementos relevantes ao sistema, com, a modelagem da estrutura do sistema, seu vocabulário e o relacionamento entre esses elementos.

4.4 Requisitos não funcionais ou de qualidade

PRESSMAN (1994) indica quatro principais requisitos não funcionais ou fatores de qualidade que devem ser levados em conta na análise e *design* de um sistema. São elas: correção, manutenção integridade e usabilidade.

4.4.1 Correção

Requisitos de correção devem assegurar a operação correta de um sistema, executando sem erros as suas funções. Isso significa que na análise e no *design* do sistema deve estar assegurada uma preocupação com relação à ocorrência de defeitos, ou seja, a ocorrência de não-conformidades em relação ao que foi requerido. Defeitos ocorrerão; deve-se buscar então a minimização dessas ocorrências e uma métrica de referência, como número defeitos ocorridos em determinado período.

No caso do sistema, cuidado especial deve ser dedicado ao modelo de objetos e ao modelo dinâmico. As mensagens a serem enviadas entre objetos devem ter o seus destinos garantidos, ou seja, mensagens de um objeto a outro não devem ser direcionadas a outros objeto, tanto por erro de programa como pela parte do usuário. Deve haver mecanismos que possibilitem a entrega da mensagem ao seu destinatário correto.

4.4.2 Manutenção

Requisitos de manutenção atestam a facilidade com a qual o sistema pode ser corrigido se um erro é encontrado, adaptado a mudanças de ambiente ou quando há mudança nos requisitos funcionais. Uma métrica de referência é o *mean time to change*: tempo que se leva para analisar uma mudança potencial, fazer o seu *design* e implementá-la.

No sistema em questão, a manutenção, a possibilidade de se executar mudanças é facilitada pelo uso da metodologia orientada a objetos. A estrutura desenhada através da metodologia é menos abalada em mudanças, grande parte devido ao fato de o conceito de objetos permanecer presente em todo o desenvolvimento. Assim, desenvolvimentos posteriores utilizam a estrutura já existente, diminuindo tempo e custos para atualizações.

4.4.3 Integridade

Requisitos de integridade relacionam-se a questões relativas a segurança das informações e possibilidade de conter possíveis ameaças, como ataque de *hackers*, conforme o caso do sistema.

Quanto à segurança dos dados, um registro (*"log"*) de operações efetuadas através do sistema e de alterações feitas nas bases de dados possibilitaria um controle eficaz se contivesse uma assinatura do autor da alteração. Além disso, atribuições de perfis aos usuários limitariam o acesso a determinadas partes da base de dados, impedindo que pessoas não autorizadas efetuem operações ilegais. Já a prevenção contra ataques de *hackers* tem se mostrado eficaz com a estrutura atual de informática que, no entanto, deve ser revista quando se desejar utilizar o sistema através da internet, o que, a princípio, poderia tornar o sistema mais vulnerável. Porém, o uso de softwares e hardwares específicos existentes no mercado podem prover uma barreira adequada para conter esses tipos de evento.

4.4.4 Usabilidade

Usabilidade refere-se à interface e à dinâmica (amigável) de operações do sistema. Deve-se então buscar o projeto de um sistema com interface e dinâmica de operações amigável (*user-friendly*), pois mesmo que suas funções sejam executadas corretamente, ele pode estar condenado ao fracasso se não for bem aceito pelos usuários.

Métricas conhecidas para se medir a usabilidade para lidar com o sistema são a habilidade intelectual do usuário necessária para executar as funções programadas (se há necessidade de conhecimentos específicos, como de matemática, da empresa, de pessoas, etc.), tempo para se tornar um usuário moderadamente eficiente, o aumento de produtividade (em relação ao sistema usado previamente) e avaliação subjetiva por parte dos usuários por meio de um questionário.

No caso do sistema, novamente um cuidado grande com o modelo de objetos deve ser tomado, a fim de que a lógica das operações entre objetos corresponda à realidade dos usuários, de maneira que as funções programadas tenham, após treinamento e tempo de adaptação adequados, uso instintivo, ou seja, que o sistema seja uma ferramenta facilitadora no registro e obtenção de informações.

Capítulo 5

Diagrama de Classes

5.1 Identificação de classes

A modelagem do sistema envolve a identificação dos elementos mais importantes para o sistema. Esses elementos formam o vocabulário do sistema e cada elemento pode ser distinguido um do outro por possuir um conjunto de propriedades, algumas delas exclusivas. No entanto, esses elementos não existem isoladamente e por isso na análise deve-se considerar também o relacionamento existente entre eles.

Retomando o capítulo 3, um diagrama de classes mostra um conjunto de classes, interfaces, além de colaborações e seus relacionamentos. É destinado para se modelar uma visão estática do *design* de um sistema. Esta visão dá suporte primordialmente aos requisitos funcionais de um sistema, ou seja, os serviços que o sistema deve prover aos usuários finais.

Classes são os elementos mais importantes de um sistema orientado ao objeto. Uma classe é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica.

Neste capítulo são apresentados os componentes das classes (nome, atributos, operações e responsabilidades) e, por fim, as classes finais e seus relacionamentos nos diagramas de classes.

5.1.1 Representação na UML

Na UML, classes podem ser representadas com retângulos, possuindo no seu espaço interno um nome, como a figura 5.1 a seguir indica.

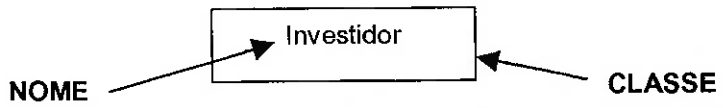

Fig. 5.1 – Notação de classes na UML (adaptado pelo autor de Booch; Rumbaugh; Jacobson, 1998)

5.1.2 Classes candidatas

Para o sistema, foram identificadas inicialmente as seguintes classes tentativas, mostradas na figura 5.2, de forma a abranger os elementos mais importantes para a modelagem, tendo como base entidades citadas até então neste trabalho.

Fig. 5.2 – Primeira identificação das classes candidatas (elaborado pelo autor)

Nesta tentativa de compor as classes mais importantes (classes candidatas), procurou-se evitar classes redundantes, classes vagas, classes atributo e elementos de implementação.

Classes redundantes representam o mesmo conjunto de objetos os quais possuem, em geral, as mesmas propriedades. Um exemplo disso é a não consideração, a princípio, da classe "Pessoal Stratus", como já mencionado nas figuras e 4.2 4.3 para efeito da construção dos *use cases*. O sistema tem como premissa o fato de que os responsáveis pelos agentes externos são os próprios funcionários da Stratus (que são os usuários do sistema) e, assim, essa denominação é substituída por "Responsável" pois é mais interessante que o nome da classe tenha um vínculo maior com a realidade. Outro exemplo de uma classe redundante seria a inclusão da classe "Contato Empresa" ou "Contato Investidor", significando o pessoal com o qual se estabelece contato do lado de empresas ou investidores, respectivamente (o significado de cada classe é explicado

mais adiante). O ideal é ter-se somente uma classe "Contato", pois suas propriedades (ainda que ligadas ou a empresas, investidores ou clientes de *Advisory*) são as mesmas, independentemente se relacionada a empresas ou investidores.

Classes vagas não possuem uma definição clara e são uma generalização abrangente de outras classes, de modo que sua utilidade é perdida no sistema. Um exemplo de uma classe vaga seria "Agente Externo" como mencionado ao longo do trabalho. A necessidade é justamente em estruturar esses agentes, diferenciado-os conforme seus atributos e papel para o sistema.

Uma classe que é representada como um atributo é uma classe atributo. Um atributo deve ser considerado uma classe se esse atributo possui propriedades (em outras palavras, outros atributos) que podem ser agrupados em uma classe. Se não existem essas propriedades, ele deve preferencialmente permanecer como atributo da classe original. Se houvesse uma classe chamada "Modelo Financeiro" com diversos atributos, para designar um só tipo de documento, esta seria uma classe atributo, possibilitando detalhar o conceito por trás de "Modelo Financeiro". No entanto, no caso do sistema esta classe não teria função, pois são os documentos com diversas finalidades ou usos, como planilhas financeiras, apresentações para investidores, etc. que se deseja conceituar numa única classe e, assim, o mais adequado seria apenas um atributo "Tipo" para qualificar a classe "Documento", já que o atributo "Tipo", no sistema, não teria propriedades ou atributos adicionais.

Classes que são não estão ligadas diretamente ao domínio do problema, como elementos de implementação, não devem ser consideradas, ainda que possam aparecer durante a fase de projeto. Como exemplo, classe como "Usuários" ou entes de software ou hardware que podem ser agrupados, não pertencem ao domínio do problema da análise.

5.1.3 Refinamento das classes

As classes apresentadas anteriormente não constituem uma versão final. Há classes que devem ser eliminadas por redundância ou por serem classes atributos, enquanto outras

devem ser mais bem detalhadas ou mantidas É conveniente detalhar as classes existentes, fazendo com que surjam classes base (generalização de um conjunto de classes), sub-classes e novas classes.

Assim, a figura 5.3 mostra uma tabela com o refinamento das classes para o sistema. Pode-se observar uma primeira separação entre as classes, que será utilizada com finalidade didática para compor os diagramas finais de relacionamento. Nas duas primeiras linhas da tabela nota-se a presença de classes relacionadas a investidores, na terceira linha classes relacionadas a empresas para investimento, na quarta classes relacionadas a *Advisory* e nas últimas linhas as classes associadas às trocas de informação e estruturação do armazenamento das informações.

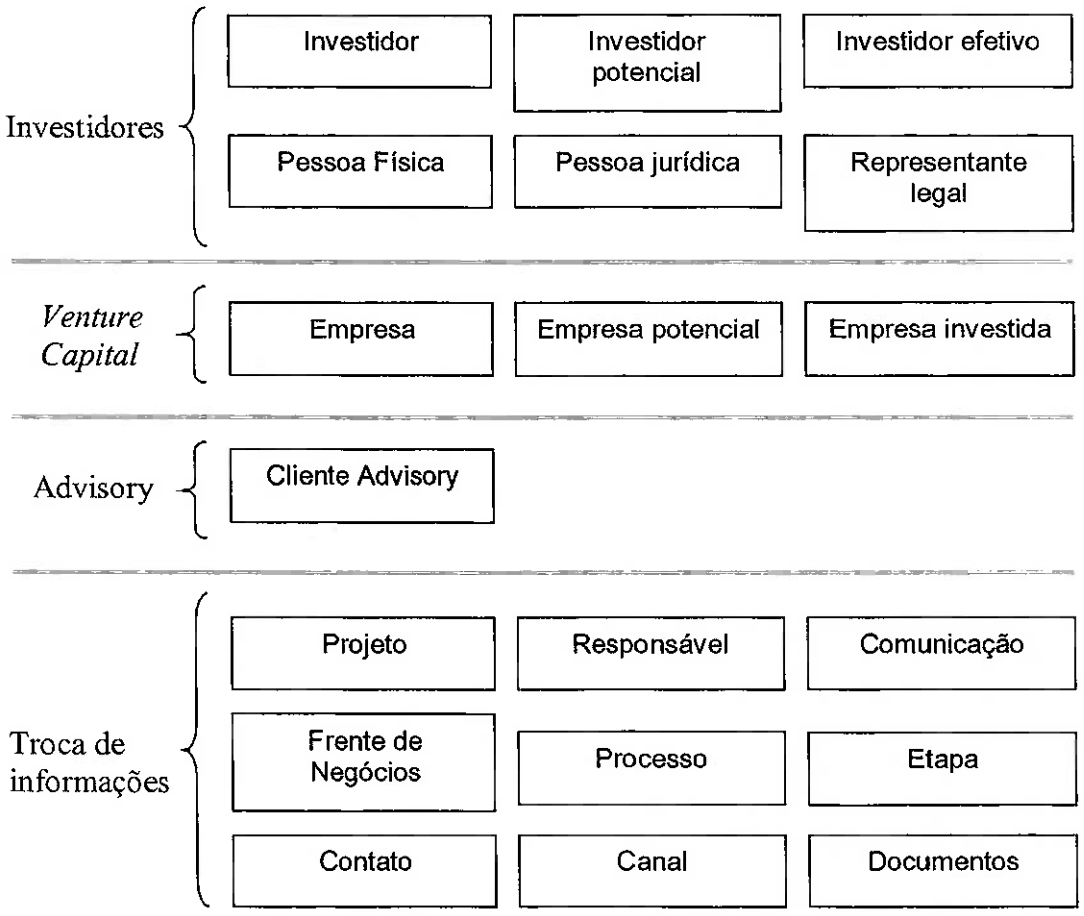

Fig. 5.3 – Classes Refinadas (elaborado pelo autor)

Pode-se observar, em relação à figura 5.2, o surgimento de cinco novas sub-classes na parte referente a investidores e mais duas classes para empresas de investimento de *Venture Capital*, além da supressão da classe "Empreendedor" e o aparecimento da classe "Canal". Essas classes são complementos às classes já apresentadas e definem mais precisamente os objetos a serem tratados pelo sistema. O significado de cada classe para o sistema é apresentado no dicionário de dados a seguir.

5.1.4 Dicionário de dados - Classes

Para se evitar interpretações errôneas de cada classe, o dicionário de dados descreve o significado das classes escolhidas à luz da realidade da empresa e do sistema. Nesse dicionário, separado como na figura 5.3, em quatro partes para melhor compreensão, é descrito o significado da classe dentro do problema atual, incluindo qualquer medida ou restrição à sua posse ou uso.

- Parte de investidores:

Investidor: pessoa ou instituição que disponibiliza a Stratus recursos para investimento com a expectativa de retorno no longo prazo. Com o investidor estabelece-se uma relação que em estágios mais avançados é caracterizada pela proximidade e troca contínua de informações.

Investidor potencial: investidor alvo para contato por parte da Stratus com o intuito de convencê-lo a confiar o seu capital a um investimento a ser gerido pela Stratus. É ainda um candidato a investir que, no entanto, pode já ter investido em um fundo de investimentos da Stratus, mas continua candidato potencial a outros fundos, cada um com um projeto de investimento específico (vide classe Projeto). Investidor potencial é uma sub-classe da classe Investidor.

Investidor efetivo: investidor propriamente dito, que já efetuou investimento em algum fundo ou *holding* de investimentos da Stratus. Com ele estabelece-se um relacionamento diferenciado, com os quais deseja-se cuidado (há preocupação com

relação a boas práticas que levem à satisfação e lealdade do cliente) nas trocas de informação. Investidor efetivo é uma sub-classe da classe Investidor

Investidor CI: investidor efetivo pertencente ao Comitê de Investimentos, que possui um tratamento diferenciado na Stratus por ter feito um aporte de recursos em geral maior do que outros investidores. Recebe informações mais detalhadas sobre o fundo de investimentos, sobre a Stratus e empresas investidas. É sub-classe da classe Investidor efetivo.

Investidor comum: investidor que não pertence ao Comitê de Investimentos. Recebe informações com um menor nível de detalhamento sobre o fundo de investimentos ou *holding*. É sub-classe da classe Investidor efetivo.

Pessoa física: associada à classe Investidor, é um indivíduo que responde como pessoa física pelo investimento. É, para efeito da análise, uma sub-classe que se contrapõe à classe Pessoa jurídica.

Pessoa jurídica: associada à classe Investidor, é uma instituição que em geral dispõe de um grande volume de recursos para investimento (comparada à pessoa física) e possui necessidade de garantir certa rentabilidade ou retorno atuarial, para renda patrimonial, reserva de risco ou pagamento de pensões.

Representante legal: associada à classe Pessoa Jurídica, é o indivíduo que legalmente representa a instituição perante a Stratus, ou seja, é o interlocutor da instituição. É também associado à classe Contato, explicada mais adiante.

- Parte de empresas para investimento de *Venture Capital*:

Empresa: é a instituição que combina recursos físicos e humanos para produzir bens ou serviços. Em estágios mais avançados de relacionamento com a Stratus recebem aporte de capital e atua em parceria desde etapas preliminares de estudo e preparação, até a monitorização do processo de utilização do capital. O empreendedor, em geral responsável pela administração da empresa, é uma pessoa inovadora que tenta

introduzir novos produtos, serviços, técnicas de produção e até mesmo novas formas de organização, tomando as decisões que irão nortear o futuro do negócio, assumindo não só riscos pessoais, mas também dos investidores e de todos os envolvidos em seu negócio. O empreendedor terá o seu lugar no sistema na classe Contato.

Empresa potencial: é a empresa alvo para investimento por parte da Stratus. É ainda uma empresa candidata a investir e, para chegar ao estágio de investimento, deverá passar por análises de mercado, financeira e legal, em diferentes etapas de relacionamento (vide classe Etapa). Empresa potencial é uma sub-classe da classe Empresa.

Empresa investida: empresa que já recebeu investimento da Stratus. Com essa empresa estabelece-se um relacionamento estreito, desde etapas preliminares de estudo e preparação, passando pela monitorização do processo de utilização do capital até a fase de venda da participação acionária da Stratus na empresa. Empresa investida é uma sub-classe da classe Empresa

- Parte de *Advisory*

Cliente *Advisory*: é a empresa que procura auxílio da Stratus para processos de fusões, aquisições, ou de consultoria estratégica. A empresa, em estágios mais avançados de relacionamento, passa por análises de mercado, financeira e legal. Esse relacionamento é dividido em etapas de acordo com o grau de amadurecimento do serviço (vide classe Etapa).

- Parte de troca de informações

Frente de negócios: é a espinha dorsal da estrutura dos negócios desenvolvidos pela Stratus, representando uma macro-classificação. Dois exemplos disso são as frentes de *Venture Capital* e *Advisory*.

Processo: é uma classificação intermediária entre Frente de negócios e projetos, sendo atualmente exclusiva da frente de *Venture Capital*.

Projeto: conceito de realizações ou execuções de tarefas ou atividades que se estabelecem nas frentes de negócios. Assim que, por exemplo, um investidor é contatado pela primeira vez, ou uma empresa procura a Stratus para obter investimento, ou um cliente de *Advisory* procura os serviços de assessoria da Stratus, diversas atividades são desenvolvidas até que se alcancem os objetivos almejados de cada frente. No sistema, o registro dessas atividades possui como característica trocas de informações que devem registradas.

Responsável: é pessoa integrante da equipe da Stratus que tem como responsabilidade o acompanhamento de um projeto e de ser o interlocutor da Stratus frente às empresas de investimento de *Venture Capital*, investidores e clientes de *Advisory* e a própria Stratus.

Comunicação: é a troca de informação a respeito de ou com investidores, empresas, clientes de *Advisory* ou internamente a Stratus.

Etapa: denota o grau de amadurecimento da relação da Stratus com investidores, empresas para investimento de *Venture Capital* e clientes de *Advisory*, sendo que para cada uma dessas áreas há diferenças no conceito de cada etapa.

Contato: é a pessoa com a qual o Responsável entra em contato para troca de informações, seja através de e-mail, ligação telefônica, reunião, etc. Podem pertencer a esta classe todas as pessoas com as quais o responsável entrou em contato.

Canal: é a interface utilizada para comunicação com o Contato (e-mail, ligação telefônica, etc.).

Documento: é um registro feito na forma de arquivo digital, contendo uma base de conhecimento para consulta futura. São associados às comunicações efetuadas com investidores, empreendedores, clientes de *Advisory* ou outras pessoas correlacionadas.

5.2 Atributos

Atributos representam propriedades dos objetos que estão sendo modelados. Em uma classe, todos os objetos compartilham os mesmos atributos. Eles são uma abstração do tipo de dado ou do estado em que um objeto se encontra.

5.2.1 Representação na UML

Na UML, atributos são listados em um compartimento exatamente abaixo do nome da classe e podem ser desenhados apenas com seus nomes, como a figura 5.4 a seguir indica.

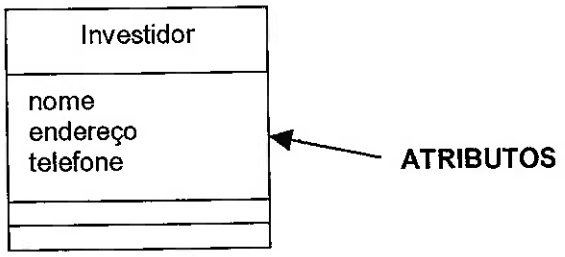

Fig. 5.4 – Notação de atributos na UML (adaptado pelo autor de Booch; Rumbaugh; Jacobson, 1998).

5.2.2 Atributos no sistema

Com a mesma finalidade do item 5.1.4, que era de se evitar interpretações errôneas de cada classe, adiante são explicitados os atributos das classes, separados quatro partes para melhor compreensão. Entretanto, aqui são levados em conta somente os atributos mais importantes para o relacionamento entre as classes e à criação de classes de apoio. Assim, há atributos no item 5.1.4 já foram considerados como classes, o que faz que os mesmos conceitos se repitam como atributos. Outros atributos estão incluídos no modelo e presentes no diagrama de classes no item 5.5.2.

- Parte de investidores:

A classe Investidor é uma generalização da realidade e por esse motivo ela não leva consigo atributos. Na prática há diferentes tipos de investidores, cada um dos quais separados em classes diferentes e com atributos distintos.

Para as sub-classes da classe Investidor, os atributos mais importantes são:

Tipo: uma classe-atributo, que distingue pessoas físicas de jurídicas.

Contato: já apresentado como classe no item 5.1.4, Contato é também um atributo das sub-classes da classe Investidor.

Endereço: corresponde ao endereço do investidor.

Endereço para correspondência: atributo usado quando a correspondência de um investidor tiver que ser enviada a um endereço diferente do especificado no atributo Endereço.

Representante legal: já apresentado como classe no item 5.1.4, Representante legal é também um atributo das sub-classes de Investidor.

Delegação de poder: atributo que permite ao usuário do sistema elaborar uma descrição da hierarquia ou organograma de cargos existentes em Pessoa jurídica.

- Parte de empresas para investimento de *Venture Capital*:

A classe Empresa é, do mesmo modo que a classe Investidor o é, uma generalização da realidade e, por esse motivo, não leva consigo atributos. Na prática há diferentes tipos de empresas, cada uma das quais separadas em classes diferentes e com atributos distintos.

Para as sub-classes da classe Empresa, os atributos mais importantes são:

Contato: já apresentado como classe no item 5.1.4, Contato é também um atributo das sub-classes da classe Empresa.

Endereço: corresponde ao endereço do investidor.

- Parte *Advisory*

A classe Cliente *Advisory* diferentemente da classe Investidor e Empresa, reflete diretamente o ponto de vista da Stratus, que não enxerga as empresas cliente diferenciadamente e, portanto, não há sub-classes.

- Parte de troca de informações

A classe Projeto possui os atributos mais importantes referentes à troca de informações. Os atributos relevantes para esta classe são:

Processo, Etapas, Responsável, Comunicação: já apresentados como classe no item 5.1.4., possuem o mesmo significado para o sistema, mas aqui como atributo da classe Projeto.

Como classe, o atributo comunicação possui os seguintes atributos:

Frente de negócios, Processo, Etapa, Projeto, Responsável, Canal, Documentos: já apresentados como classe no item 5.1.4., possuem o mesmo significado para o sistema como atributo da classe Projeto.

5.3 Operações

Uma operação é uma implementação de um serviço que pode ser requerido por qualquer objeto da classe e que afeta seu comportamento. Em outras palavras, uma operação é uma abstração de algo que se pode fazer com um objeto e que pode compartilhado por todos os objetos da classe. Freqüentemente, invocar uma operação em um objeto altera os dados do objeto ou seu estado.

5.3.1 Representação na UML

Na UML, operações são listadas imediatamente abaixo dos atributos das classes e podem ser desenhadas apenas com seus nomes, como a figura 5.5 a seguir indica.

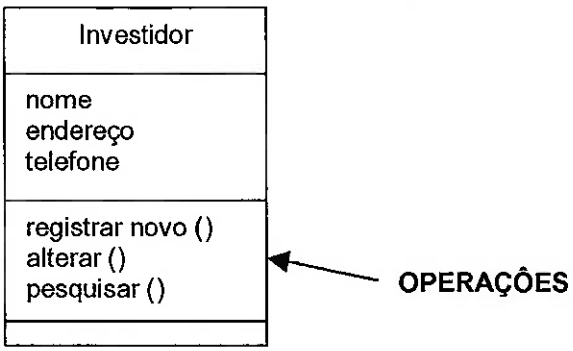

Fig. 5.5 – Notação de operações na UML (adaptado pelo autor de Booch; Rumbaugh; Jacobson, 1998).

5.3.2 Operações no sistema

Do mesmo modo que nos itens anteriores, convém explicitar as operações mais importantes das classes.

Em princípio, o sistema apresenta uma baixa complexidade no que tange às possíveis operações a serem executadas pelos objetos das classes. Como não há processamento de valores, nem complicadas operações matemáticas entre os objetos das classes, as operações consistem em modificar, acrescentar ou alterar informações, sem que a estrutura das classes seja abalada.

As principais operações que se aplicam a todas as classes são:

- Registrar: inclusão de novos objetos em cada classe.

- Alterar: modificação de atributos de objetos como, por exemplo, endereço, número de telefone, de investidor, etc.
- Apagar: exclusão de objetos desnecessários.
- Pesquisar: procura por objetos específicos que leva em consideração os atributos das classes às quais os objetos pertencem.

5.4 Responsabilidades

Responsabilidade é a função ou papel que a classe deve executar. Quando uma classe é criada, estabelece-se que todos os objetos daquela classe têm o mesmo tipo de estado e o mesmo tipo de comportamento. Seus atributos e operações são os meios através dos quais a classe assume suas responsabilidades.

Algumas classes, em especial as classes mais genéricas, não possuem um papel específico no sistema. Este papel é mais bem compreendido se forem designadas responsabilidades às suas sub-classes. A classe Investidor, por exemplo, é uma generalização das sub-classes Investidor potencial e Investidor efetivo, mas só existe para melhor compreensão do universo ao qual suas sub-classes pertencem e, por isso, não são designadas responsabilidades a ela.

5.4.1 Representação na UML

Na UML, responsabilidades são listadas imediatamente abaixo das operações das classes, como mostra a figura 5.6 a seguir.

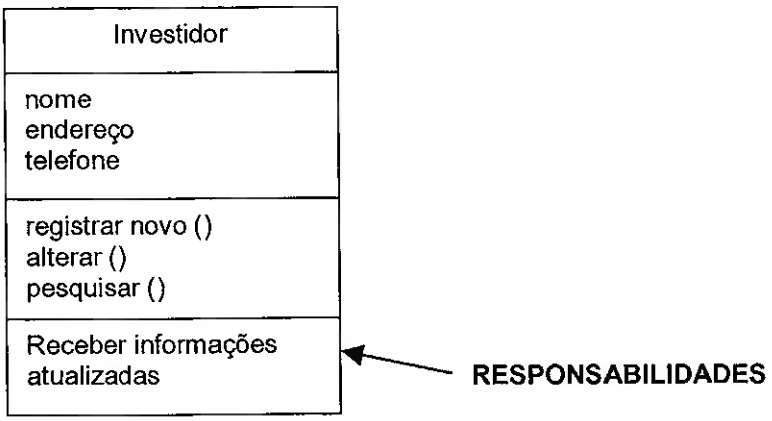

Fig. 5.6 – Notação de responsabilidades na UML (adaptado pelo autor de Booch; Rumbaugh; Jacobson, 1998).

5.4.2 Responsabilidades no sistema

As responsabilidades de cada classe são:

- Parte de Investidores

Investidor potencial: Ser contatado pela Stratus com o intuito de convencê-lo a investir.

O investidor potencial e a Stratus estabelecerão um relacionamento de longo prazo, em que a comunicação deve ser controlada e registrada. Parte-se do princípio que informações atualizadas a respeito do investidor potencial e de outros assuntos de interesse do investidor facilitem o convencimento desse investidor a dispor seus recursos para gestão da Stratus.

Investidor efetivo: Receber informações sobre fundos e empresas investidas.

O relacionamento que se estabelece com o investidor efetivo (que não o exime de ser também um investidor potencial) é também de longo prazo e visa mantê-lo constantemente atualizado com respeito ao seu investimento, na esperança de que isto o satisfaça e o torne leal para mais investimentos em outras oportunidades.

- Parte de empresas para investimento de *Venture Capital*:

Empresa potencial: enviar informações sobre a empresa

Em estágios iniciais de contato com empresas candidatas a receber investimentos estabelece-se uma relação em que a empresa fornece uma variedade de informações necessárias às primeiras análises por parte da Stratus.

Empresa investida: enviar informações a respeito do andamento da empresa

Empresas que receberam investimento da Stratus e têm que informá-la periodicamente a respeito do andamento dos negócios da empresa.

- Parte *Advisory*

Cliente *Advisory*: enviar informações sobre a empresa

Analogamente à Empresa potencial, estabelece-se com a empresa cliente, em estágios iniciais de contato, uma relação em que a empresa fornece uma variedade de informações necessárias às primeiras análises por parte da Stratus.

- Parte de troca de informações:

Projeto: ser base das informações relacionadas aos agentes externos, provendo o histórico de todo o projeto com relação às trocas de informação.

Responsável: Efetuar controle das informações registradas e dos projetos sob sua responsabilidade

Comunicação: Servir de base para o registro da uma troca de informações.

5.5 Relacionamentos entre classes

Na modelagem de um sistema deve-se não somente identificar os elementos que compõem o sistema, mas também convém modelar como esses elementos interagem uns com os outros.

Na modelagem orientada a objetos existem três tipos de relacionamentos que são especialmente importantes, nomeados a seguir: *dependências*, que representam relações de dependência; *generalizações* que ligam classes gerais a classes específicas; e *associações*, que representam relacionamento estrutural entre classes. Cada um desses relacionamentos provê um modo diferente de combinar as abstrações existentes no domínio do problema.

A tentação de construir uma vasta rede que abranja todas as classes em um só diagrama pode tornar a compreensão muito difícil. Por isso é usada a separação entre a parte de investimentos, empresas de investimento de *Venture Capital*, clientes de *Advisory* e comunicações, ainda que algumas classes possuam relacionamentos que ultrapassem os limites dessa divisão.

5.5.1 Representação na UML

Na UML, os relacionamentos são representados graficamente através de linhas entre classes, como mostra a figura 5.7 a seguir. Dependências são representadas com uma linha pontilhada e uma seta ao extremo da linha, generalizações com linhas cheias e uma seta triangular ao extremo e associações com uma linha simples.

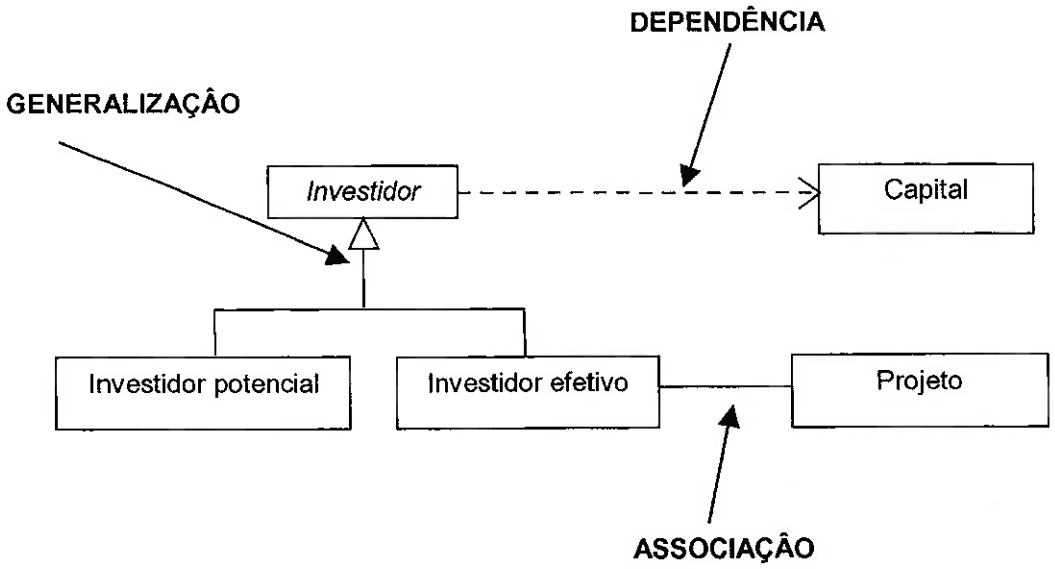

Fig. 5.7 – Notação de relacionamentos na UML – Dependência, generalização e associação (adaptado pelo autor de Booch; Rumbaugh; Jacobson, 1998).

Dependência é um relacionamento entre classes que denota que a mudança na especificação de um objeto em uma classe afeta outra classe, mas não necessariamente o reverso. Em outras palavras, é usada quando uma classe faz uso de uma outra, e não o contrário. Nesta análise, dependências não foram identificadas para a modelagem do sistema.

Generalização é um relacionamento entre uma classe base e uma sub-classe, ou classe derivada. Ela denota uma relação em que a sub-classe deve ser lida como sendo um *tipo* da classe mais genérica, da qual foi derivada. Objetos na sub-classe herdam os atributos e operações especificadas na classe genérica.

Associação é um relacionamento estrutural que especifica que um objeto de uma classe está conectado a objetos de outras classes. Nas associações, as maiores ocorrências no sistema estão relacionadas a associações entre atributos de uma classe e uma classe criada a partir de atributos. Um tipo de comum associação que se estabelece é do tipo *multiplicidade*, em que a quantidade de objetos referente às classes é específico ou, se não há um valor explícito, um intervalo de fechado de valores pode ser indicado.

Assim, para se especificar a multiplicidade, usa-se a notação mostrada na figura 5.8.

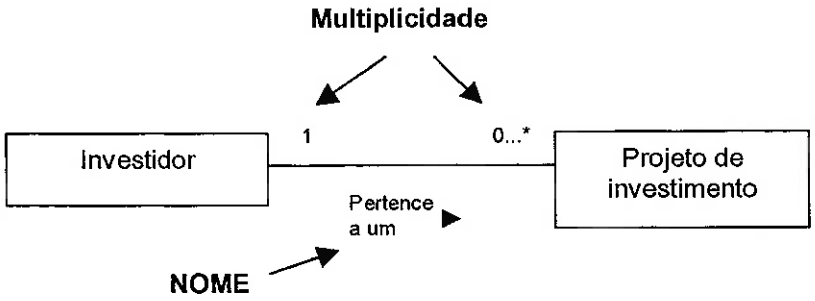

Fig. 5.8 – Notação de multiplicidade para associações na UML (adaptado pelo autor de Booch; Rumbaugh; Jacobson, 1998).

As indicações explícitas podem ser feitas através de números e faixas de valores com a notação X...Y, com X indicando o início do intervalo e Y o fim (X e Y inclusive). O símbolo " * " indica que limite no infinito.

Para cada objeto da classe no extremo oposto da indicação deve haver aquela determinada quantidade de objetos na classe adjacente. Assim, no exemplo da figura 5.8, para cada investidor (classe no extremo oposto da associação), deve haver 0 (zero) ou mais projetos de investimento associados e, da mesma forma, cada projeto de investimento deve ter somente um investidor associado.

A associação pode conter ainda um nome, que descreve a natureza da associação, evitando que haja ambigüidades relacionadas ao seu significado. O triângulo preenchido dá o sentido em que o nome da relação deve ser lido. No exemplo, deve-se ler que o "investidor efetivo" "pertence a um" "projeto de investimento".

5.5.2 Diagramas de classes e relacionamentos

Assim, esclarecida a complexidade dos elementos integrantes das classes na UML, são considerados agora o relacionamento entre elas, incluindo todos seus elementos. Os diagramas apresentam classes que estão separadas para melhor compreensão mas, no entanto, deve-se estar atento para considerá-las como um todo.

- Parte de investidores:

A classe Investidor é uma generalização da realidade e por esse motivo não leva consigo atributos, como se pode observar na figura 5.9. Na prática há dois tipos diferentes de investidores, cada um dos quais com atributos diferentes e relacionamentos com outras classes. São esses dois tipos de investidores que no modelo tornam-se sub-classes da classe investidor e que carregam os primeiros atributos. Essas duas classes distintas são Investidor potencial e Investidor efetivo, que são as classes mais significativas no trato com investidores, pois permitem distinguir entre os investidores àqueles que são os mais importantes e que efetivamente são clientes da Stratus, ou seja, investidores que pertencem à classe Investidor efetivo.

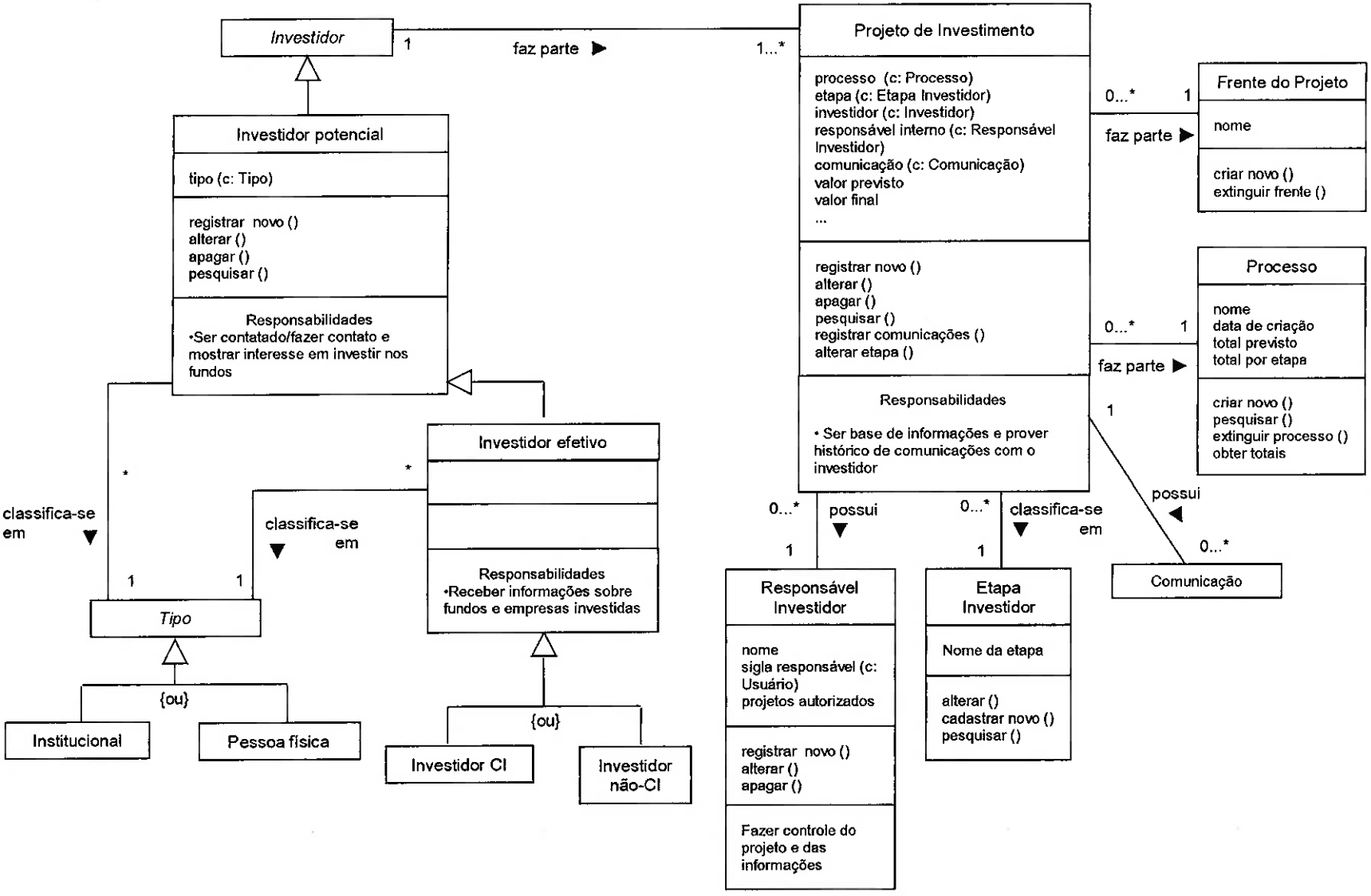

Fig. 5.9 – Diagrama de classes e relacionamentos – investidor (elaborado pelo autor)

Como sub-classe de Investidor efetivo nota-se no diagrama duas sub-classes derivadas relacionadas com uma generalização, que são Investidor CI e Investidor não CI, cujo papel foi explicado no item 5.1.4.

Com um relacionamento do tipo associação, a classe Investidor (e as suas sub-classes) liga-se à classe Projeto. Nesta última classe e nas classes relacionadas concentram-se as informações importantes para a Stratus, que estão contidas em seus atributos. Relacionadas à classe Projeto com uma associação estão as classes Processo e Etapa, que são classes de auxílio à estruturação das informações (vide item 2.1), e a classe Responsável.

Para as classes Investidor potencial e Investidor efetivo há um atributo comum que leva à criação da classe Tipo, pelo fato de que dele derivam-se outros atributos e classes. A classe Tipo é uma segunda classificação para investidores, sejam eles potenciais ou não. Com as sub-classes da classe genérica Tipo pode-se discernir entre investidores que são pessoa física ou jurídica através de uma relação de generalização. A importância disso se deve ao fato de que o trato com uma pessoa jurídica é mais complexo do que com pessoa física por geralmente envolver maior número de pessoas, entraves legais e outros aspectos relacionados à localização da empresa e ao representante legal, em especial para o envio de correspondência. Esses aspectos constam nos atributos das respectivas classes, como se pode observar no diagrama da figura 5.10.

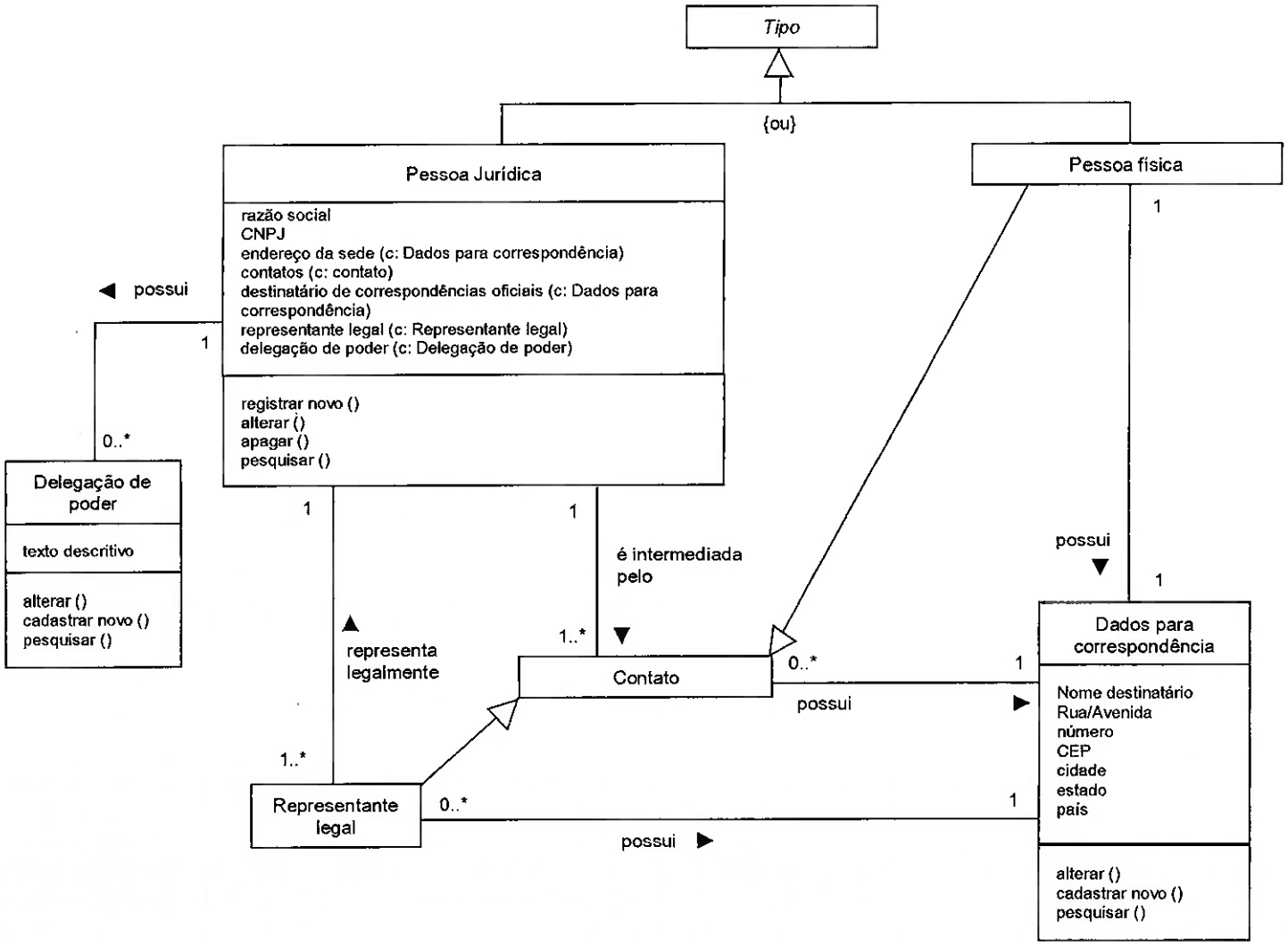

Fig. 5.10 – Diagrama de classes e relacionamentos – tipo (elaborado pelo autor)

Associadas à classe Pessoa jurídica está a classe Contato. Ela permite visualizar as pessoas com as quais a Stratus pode se comunicar já que, relacionadas à Pessoa Jurídica, pode haver mais de uma. A classe Representante legal é mostrada como uma classe derivada da classe Contato, pois o representante legal de uma pessoa jurídica é também um tipo de contato. Do mesmo modo, uma pessoa física é considerada uma pessoa a ser contatada e, por isso, também é uma sub-classe da classe Contato.

As classes Contato, Representante legal e Pessoa física estão associadas à classe Dados para correspondência. Cada um deles pode possuir individualmente dados para envio de correspondência. No caso de contatos ligados a pessoas jurídicas, pode haver um endereço diferente daquele da sede da empresa, por exemplo, quando houver uma filial.

- Parte de empresas para investimento de *Venture Capital*:

A classe Empresa, analogamente à classe Investidor, é uma generalização da realidade e por esse motivo também não leva consigo atributos, como se pode observar na figura 5.11. Na prática há dois tipos diferentes de Empresa, cada uma das quais com atributos diferentes e relacionamentos com outras classes. E são esses dois tipos de empresa que no modelo tornam-se sub-classes da classe Empresa e que carregam os primeiros atributos. Essas duas classes distintas são Empresa potencial e Empresa investida, que são as classes significativas no trato com empresas relacionadas a *Venture Capital*, pois permitem distinguir entre elas aquelas que são as mais importantes para a Stratus, ou seja, as empresas pertencentes à classe Empresa investida. Resumidamente, isso se pelo fato de o capital aplicado estar em jogo e se buscar o retorno daquilo que foi investido ao final de um determinado período, enquanto que no trato com empresas potenciais a responsabilidade do relacionamento é menor.

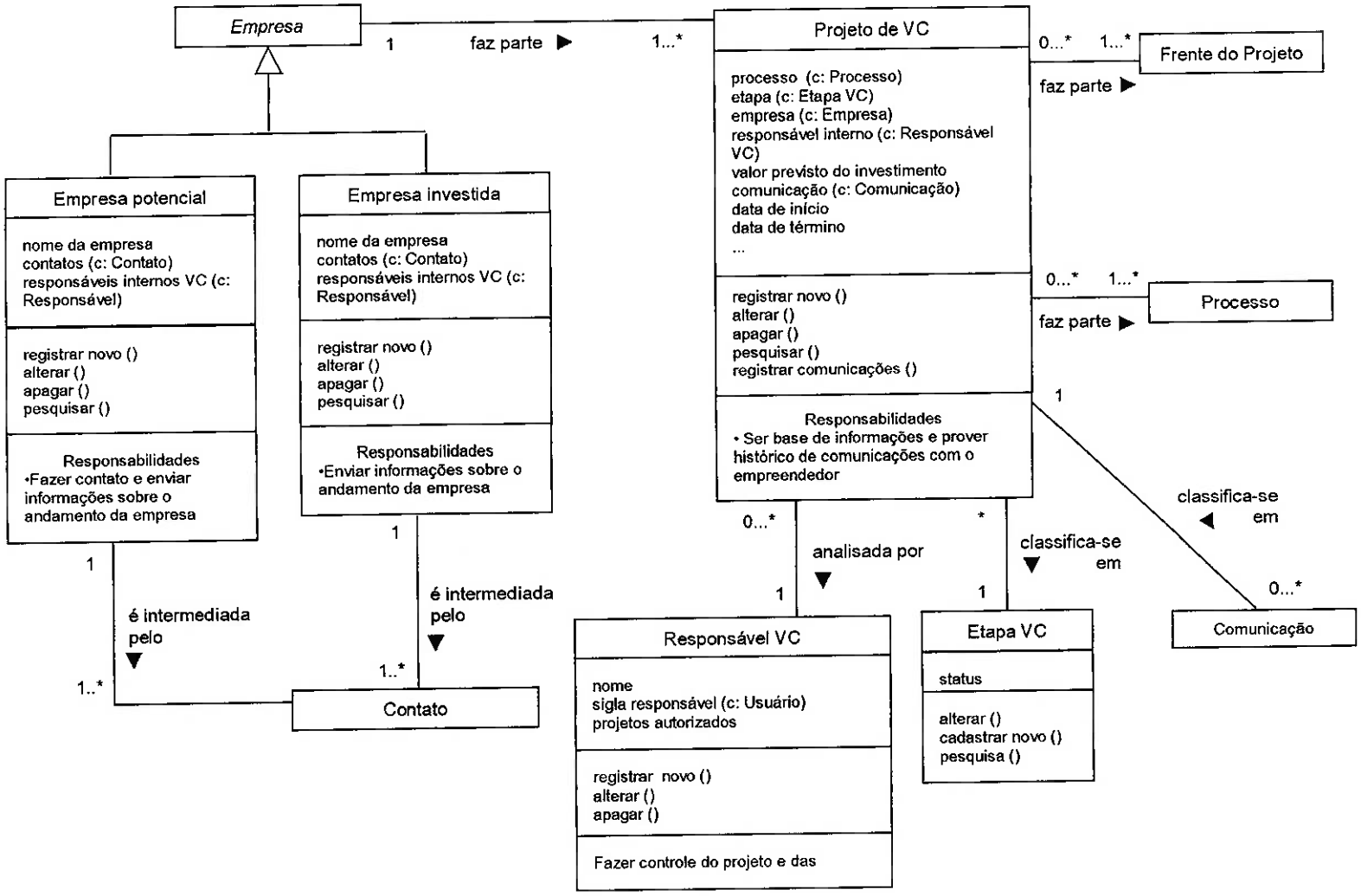

Fig. 5.11 – Diagrama de classes e relacionamentos – Empresa (elaborado pelo autor)

Analogamente à figura 5.9, pode-se ver na figura 5.11 um relacionamento do tipo associação entre a classe empresa (e as suas sub-classes) e à classe Projeto, no caso, da frente de *Venture Capital*. Nas classes relacionadas à classe Projeto concentram-se as informações importantes para a Stratus, que são as informações contidas em seus atributos. Relacionadas à classe Projeto com uma associação estão as classes Processo e Etapa, que são classes de auxílio à estruturação das informações (vide item 2.1), e a classe Responsável.

Associada à classe Empresa potencial e Empresa investida está a classe Contato. Esta classe permite visualizar as pessoas com as quais a Stratus pode se comunicar, já que, numericamente, pode haver mais de uma.

- Parte *Advisory*

Na figura 5.12 pode-se observar relacionamentos semelhantes ao já visto com a parte de investidores e empresas investimento de *Venture Capital*. Ali, nota-se a associação entre a classe empresa cliente e a classe Projeto de *Advisory*. Nesta classe e nas classes relacionadas à classe Projeto concentram-se as informações importantes para a Stratus, que são as informações fornecidas pelos seus atributos. Relacionadas à classe Projeto com uma associação estão as classes Etapa, que é uma classes de auxílio à estruturação das informações (vide item 2.1), e a classe Responsável.

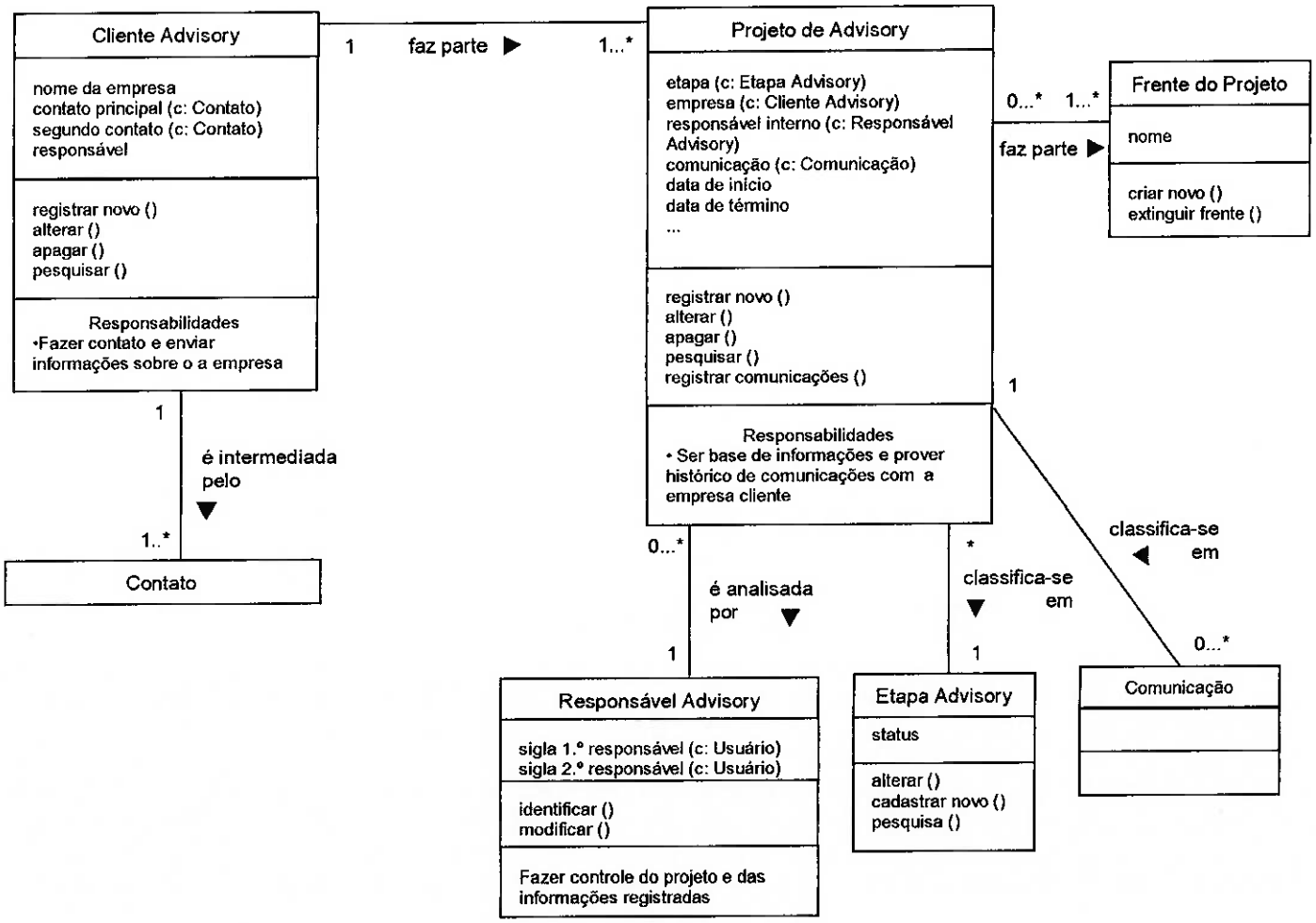

Fig. 5.12 – Diagrama de classes e relacionamentos – Empresa (elaborado pelo autor)

Também neste caso está associada à classe Empresa potencial a classe Contato, sempre com a mesma função já descrita, que é a de visualizar as pessoas com as quais a Stratus pode se comunicar, já que, numericamente, pode haver mais de uma.

- Parte de troca de informações

A classe Comunicação é mais importante no diagrama da figura 5.13. Quando houver troca de informações entre uma pessoa responsável por um projeto e um contato associado, o registro dessa troca de informações será feito através desta classe.

No diagrama pode-se observar que a estrutura da troca de informações e o registro desta troca. A troca de informações se dá através de um canal (classe Canal, cujos objetos podem ser e-mail, telefonema, reunião, por exemplo) com um contato (classe Contato) de uma empresa ou com um investidor. A comunicação (classe Comunicação) é estabelecida com o responsável do projeto (classe Responsável), mas essa comunicação dever ser localizada entre as frentes de negócios, processos, etapas, e projetos existentes para que o registro se dê de forma correta. À comunicação, cujo registro pode ser feito através de um texto descritivo, podem ser associados documentos digitais (classe Documento).

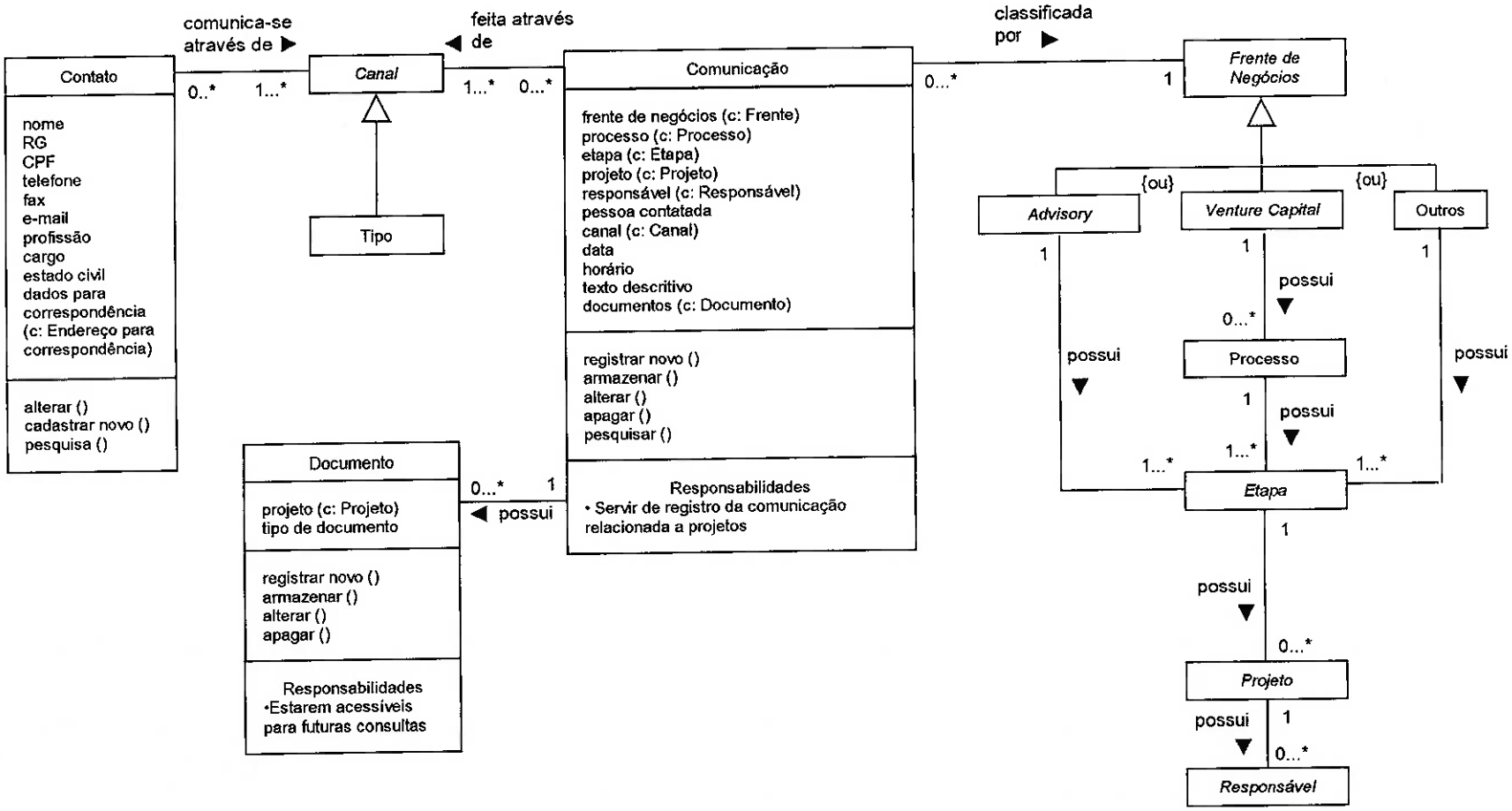

Fig. 5.13 – Diagrama de classes e relacionamentos – Comunicações (elaborado pelo autor)

Capítulo 6

Diagrama de Seqüências

6.1 Interações

A última parte da modelagem do sistema envolve a identificação das interações entre elementos importantes do sistema. Em qualquer sistema os objetos não ficam ociosos; eles interagem uns com os outros trocando mensagens. Uma interação é um comportamento que engloba um grupo de mensagens trocadas entre um conjunto objetos com vistas a um objetivo determinado.

Retomando o capítulo 3, o diagrama de seqüências é um tipo de diagrama de interações. Um diagrama de interações envolve a modelagem de exemplos concretos ou protótipos de classes, interfaces e componentes entre os quais mensagens são enviadas e recebidas. O diagrama de seqüências tem o seu foco na seqüência de mensagens ao longo do tempo, mostrando um grupo de objetos que trabalham conjuntamente com envio e recebimento de mensagens, tendo como objetivo a execução de determinadas ações.

Na prática, há uma fusão do diagrama de classes com os *use cases*, onde elementos desses dois diagramas estão presentes para a construção das interações (parte dinâmica do sistema), que são representadas neste trabalho por diagramas de seqüências. As seqüências exemplificam *use cases* mais típicos do sistema, utilizando-se de objetos pertencentes às classes descritas no capítulo 5.

Neste capítulo é apresentado inicialmente o conceito de mensagem, a notação para o diagrama de seqüências e por fim as seqüências mais importantes do sistema.

6.2 Mensagens

Uma foto de um conjunto de objetos e as ligações existentes entre eles daria uma idéia de uma situação estática num momento em particular. Imaginando que haja mudanças nos estados desse conjunto de objetos ao longo do tempo, é necessário outro tipo de abordagem que permita que as mudanças ocorridas com esses objetos sejam visualizadas. Imaginando-se fotos tiradas desse conjunto de objetos, cada uma num instante de tempo sucessivo, essas fotos passam a retratar as mudanças nos objetos ao longo do tempo. Se os objetos não estão totalmente ociosos, pode-se ver objetos

passando mensagens a outros objetos, enviando eventos e invocando operações. Além disso, em cada quadro pode-se visualizar explicitamente o estado atual e o papel de cada objeto em questão.

Uma mensagem é a especificação de uma comunicação entre objetos que transmite informações com a expectativa de que uma haverá uma atividade como resultado. A ação resultante é uma instrução executável em uma linguagem de programação.

6.3 Seqüências

Quando um objeto envia uma mensagem a outro objeto (em geral delegando uma ação ao receptor), o objeto receptor envia por sua vez uma mensagem para outro objeto, o qual envia outra mensagem para um terceiro e assim por diante. Essa corrente de mensagens forma uma seqüência.

O diagrama de seqüências em particular enfatiza a seqüência de mensagens no tempo, mostrando um grupo de objetos que trabalham conjuntamente com envio e recebimento de mensagens, tendo como objetivo a execução de determinadas ações. Essas mensagens podem invocar uma operação, envio de sinal ou mesmo a criação ou destruição de objetos.

6.3.1 Representação na UML

No diagrama de seqüências estão modelados os objetos que participam das trocas de mensagens e as ações que ocorrem entre eles.

Os objetos participantes das iterações estão no topo do diagrama, ao longo do eixo X, e as mensagens que os objetos enviam e recebem estão ao longo do eixo Y, ordenadas em ordem crescente de tempo do topo até a base do diagrama, dando uma referência do fluxo de controle ao longo do tempo.

Na figura 6.1 pode-se observar esses elementos.

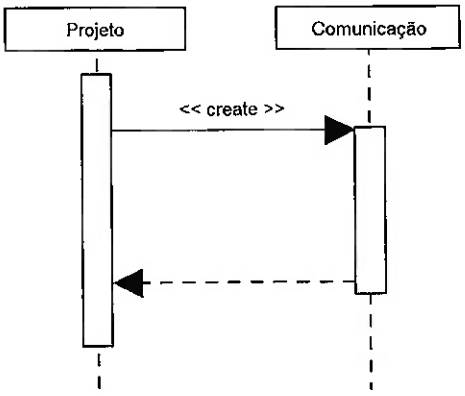

Fig. 6.1 – Notação para ações e seqüências na UML (adaptado pelo autor de Booch; Rumbaugh; Jacobson, 1998).

Além dos objetos participantes, 5 tipos básicos de ações entre objetos podem ser considerados:

- *Call*: invoca uma operação em um objeto
- *Return*: retorna um valor ao executor do *call*
- *Send*: envia um sinal a um objeto
- *Create*: cria um objeto
- *Destroy*: destrói um objeto

Essas ações têm representação distinta na UML, como mostra a figura 6.1.

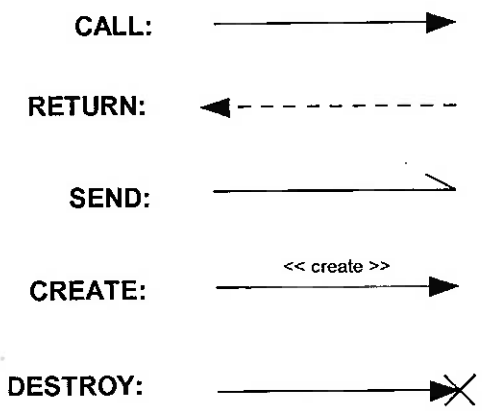

Fig. 6.2 – Notação para ações na UML (adaptado pelo autor de Booch; Rumbaugh; Jacobson, 1998).

6.3.2 Diagrama de seqüências

Assim, apresentados os elementos integrantes dos diagramas de seqüências da UML, é considerada agora sua aplicação ao sistema. Os diagramas a seguir mostram condições primárias de uso do sistema, ou seja, o uso mais comum por parte dos usuários, não levando em conta o uso excepcional decorrente de erros ou exceções, que obrigaria a consideração de seqüências distintas e alternativas. Além disso, os diagramas são de alto nível, ou seja, o nível de detalhes e especificações apresentado é pequeno, visando-se obter uma visão abrangente das trocas de mensagens que ocorrem entre os objetos.

Os dois primeiros diagramas referem-se ao cenário de registro de comunicação relacionada a um projeto de investimento de por parte de um investidor e, o segundo, ao cadastro de um projeto de *Venture Capital*. Esses exemplos foram escolhidos por representarem os usos típicos do sistema, que são o cadastramento e o registro de comunicação que, no caso, podem ser estendidos a projetos de investimento por parte de investidores ou projetos de *Advisory,* e também ao cadastro de novos projetos, independentemente da frente.

- Registro da comunicação de um projeto de investimento:

No diagrama da figura 6.3 pode-se observar as interações e o envio de mensagens entre os objetos da parte superior ao longo do tempo. Como primeiro passo para se efetuar o registro, o Responsável João seleciona a frente de negócios de *Venture Capital* (VC), o processo (Fundo A), a Etapa (2) em que o projeto se encontra e o investidor Contato (Vicente) com o qual realizará a comunicação. Antes de a comunicação se concretizar, ele poderá consultar o histórico das comunicações previamente realizadas, com o intuito de recuperar as últimas informações e assim poder dar continuidade ao que foi conversado anteriormente, tendo o responsável João participado da conversa ou não, isto é, uma terceira pessoa pode ter se comunicado com o contato Vicente.

Essas trocas de mensagens corroboram com os diagramas de classes apresentados no capítulo 5 e suas multiplicidades. No diagrama 5.7 pode-se ver a multiplicidade da classe projeto em relação às demais classes, como Frente de negócios, Processo e Etapa. Para cada projeto há somente um elemento associado de cada uma dessas classes e essa associação é obrigatória. O diagrama de seqüências deve mostrar os objetos (e suas respectivas classes) mais importantes e os objetos com os quais estes se relacionam para a concretização de uma operação completa e, por isso, estas classes estão novamente presentes neste diagrama, além de prover objetos de exemplo.

Nota-se ainda no diagrama a chamada das mensagens que são todas originadas pelo Responsável (João) e o retorno das mensagens se dá ao Projeto. Somente depois de selecionadas a Frente, o Processo e a Etapa é que o usuário obtém resposta de um projeto que deseja acionar. Por último, existe a recuperação de informações de comunicações passadas, cuja troca de mensagens se dá no âmbito de Responsável e Projeto.

O diagrama da figura 6.4 mostra as seqüências para o registro da comunicação. Para que uma nova interação entre o Responsável (João) e o Contato (Vicente) se dê, é criado um novo objeto da classe Comunicação, que fica encarregado do registro dessa interação. Por já terem sido selecionados no diagrama da figura 6.3 os objetos da Frente de Negócios, Processo, Etapa e Contato (que para efeito da Comunicação devem ser considerados como atributos), eles encarregam-se de informar o objeto em Comunicação para que a comunicação tenha os atributos corretos.

O próximo passo do usuário é registrar o conteúdo da interação e da troca de informações feita com o Contato (Vicente). O usuário pode também incluir não só os dados inseridos por meio de digitação, mas também arquivos digitais. Para isso, um novo objeto da classe Documento é criado para que a inserção do arquivo possa ser feita, mas esse arquivo digital é anexado à Comunicação e ele estará em última instância associado a esse elemento. Assim, terminada a inserção do arquivo digital, o objeto em Documento desaparece.

Do mesmo modo, terminado o registro da interação com o Contato, as informações da comunicação, que incluem o conteúdo, atributos e documentos digitais, devem ficar, por fim, associadas ao projeto.

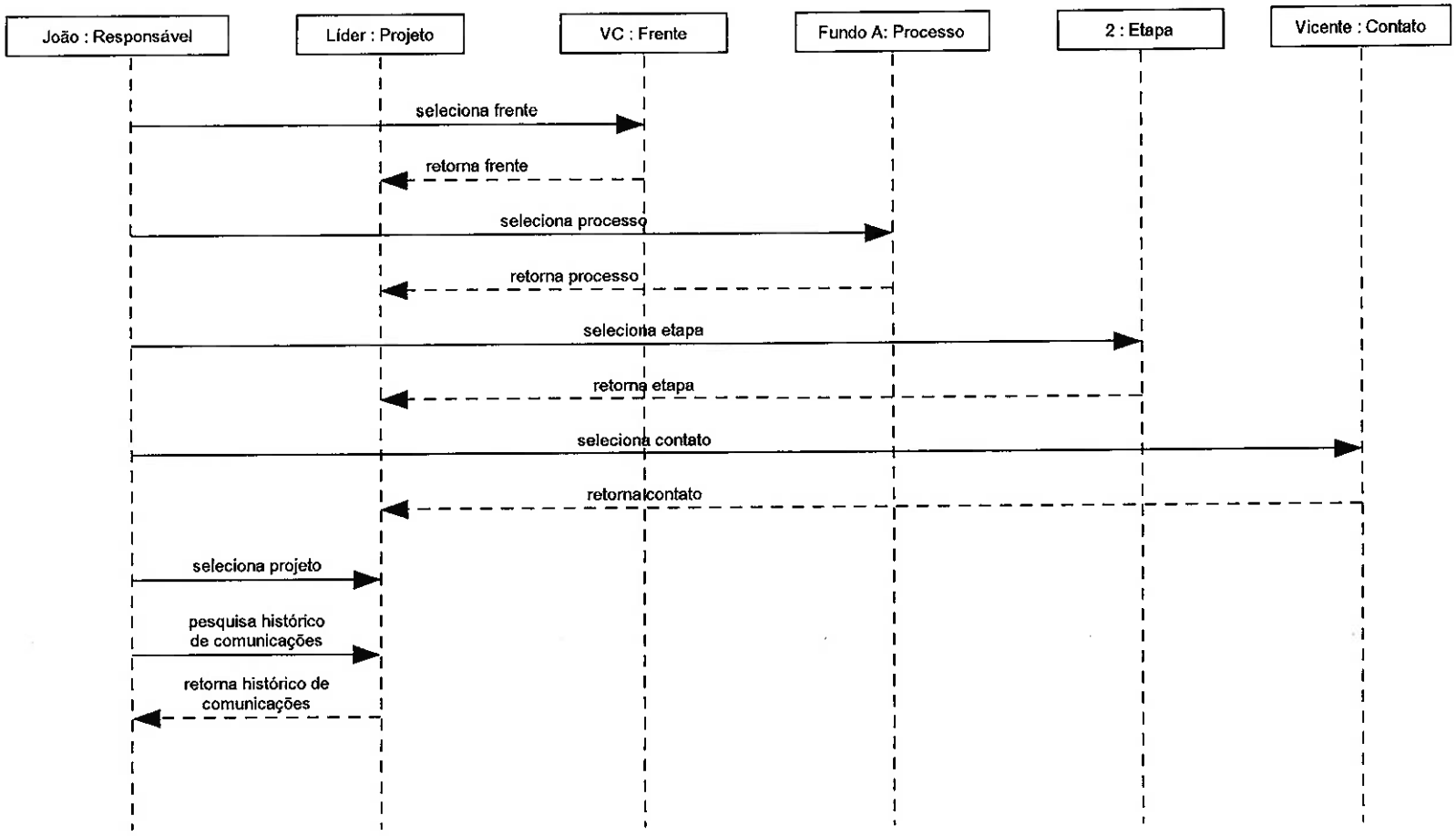

Fig. 6.3 – Diagrama de seqüências para comunicação (elaborado pelo autor)

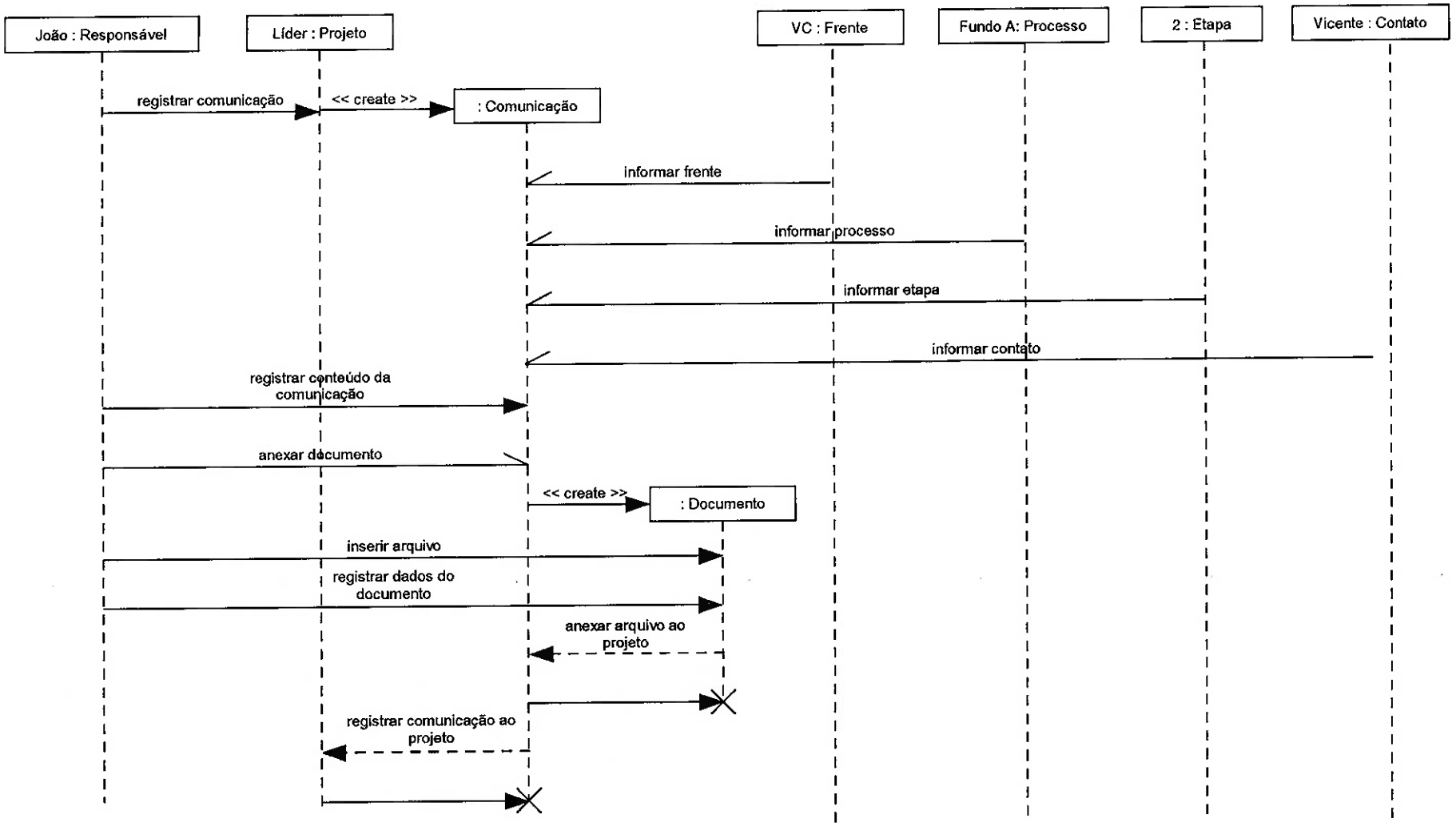

Fig. 6.4 – Diagrama de seqüências para comunicação - Continuação (elaborado pelo autor)

- Cadastro de um novo projeto para investimento de *Venture Capital*:

O diagrama da figura 6.5 mostra o envio de mensagens entre objetos para a realização de cadastramento de um novo projeto para investimento de *Venture Capital*. Para se efetuar o cadastro, o usuário cria um novo objeto da classe Projeto (Delta, no exemplo) para que sejam incluídas informações pertinentes.

Criado o projeto, o usuário deverá, assim como no exemplo do registro da comunicação, selecionar a frente de negócios e o processo. A etapa atribuída ao projeto é designada automaticamente pelo sistema, pois para projetos de investimento de *Venture Capital* o projeto deverá necessariamente iniciar pela etapa 1. Isso acontece porque o projeto de investimento de *Venture Capital* deverá obrigatoriamente passar por várias análises, cada uma delas em etapas distintas e não é permitido ao projeto "pular" etapas. Isso, no entanto, seria diferente para projetos de investimento por parte de investidores, já que as etapas neste caso denotam a "temperatura" do relacionamento da Stratus com o investidor e, sendo assim, ela pode já começar em níveis mais altos.

Em seguida o usuário deve incluir outros dados a respeito do projeto como o valor previsto do empreendimento, data de início das análises e outros, que constam na classe Projeto da figura 5.11. Além de dados do projeto, deve-se incluir as pessoas da empresa candidata com as quais a Stratus trocará informações. Essas pessoas pertencem à classe Contato (no exemplo, Geraldo e Maria) para ss quais deverão também ser inseridos dados específicos.

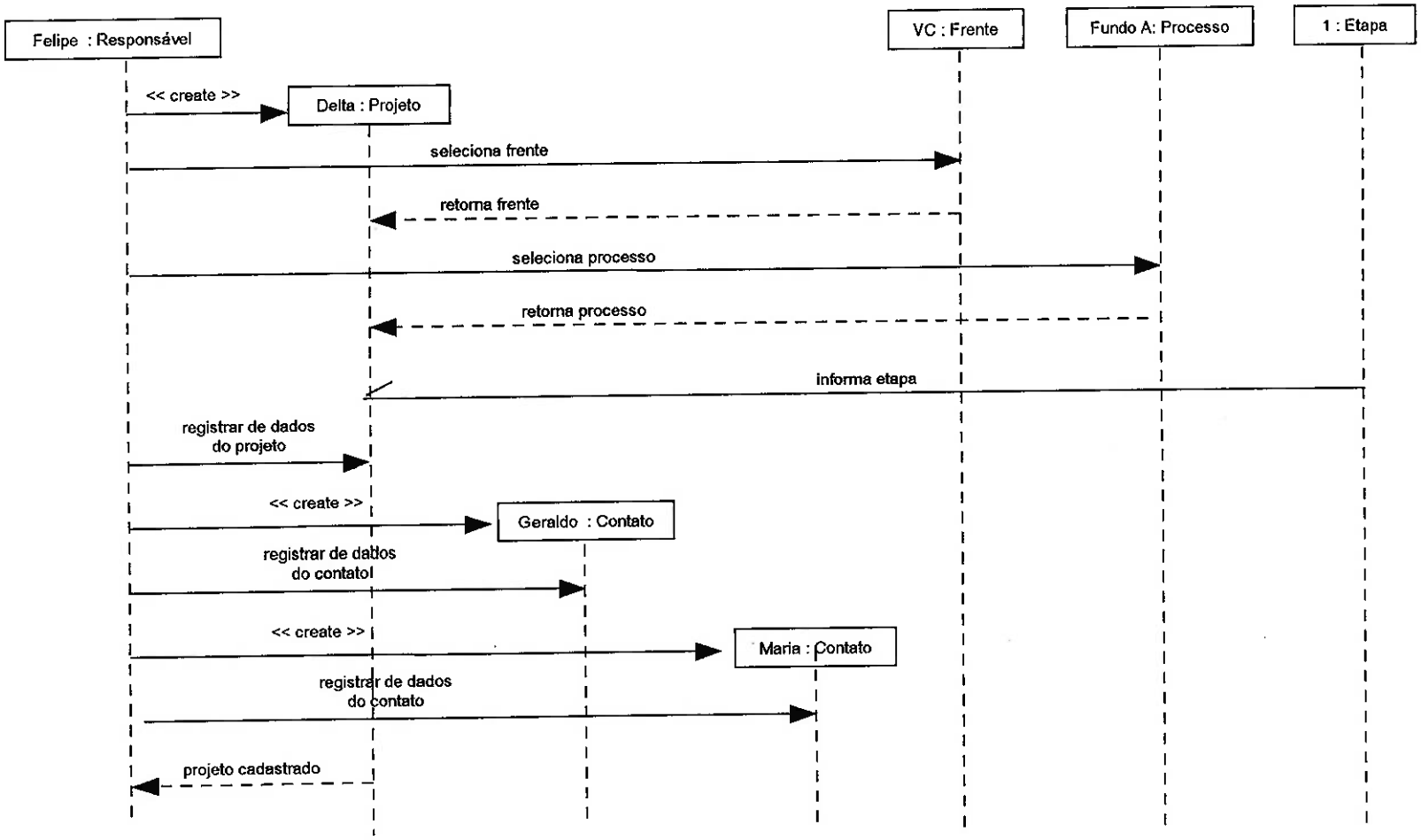

Fig. 6.5 – Diagrama de seqüências para comunicação Cadastro de contatos (elaborado pelo autor)

Capítulo 7

Resultado da análise

7.1 Introdução

Neste capítulo são feitas considerações a respeito das características do sistema obtidas a partir do modelo resultante da análise deste trabalho, assim como sobre considerações sobre o seu desenvolvimento.

7.2 Características do novo sistema

O novo sistema modelado elimina deficiências presentes no sistema atualmente em uso na Stratus e possui singularidades em sua estrutura. As principais características são:

- Arquitetura racional e funcional

A estrutura do sistema centraliza as informações levando em conta os diferentes negócios da Stratus, sem que haja sobreposição de classificações, característica que não é suportada pelo sistema existente atualmente na empresa, além de que apenas uma parte das informações nos sistema atual está centralizada. A presença de classes que dividem as informações de forma racional e armazenam dados compartilhados pelo sistema torna mais fácil a busca por informações.

- Base única de dados e melhor gerenciamento da comunicação

Com a centralização das informações, em especial na classe Projeto, a base de dados oferecida pelo sistema torna-se um elemento de importância fundamental. Através dessa base pode-se ter controle de todas as informações trocadas com pessoas do meio externo a Stratus relacionadas a esses projetos e também das informações provindas do ambiente interno da empresa.

- Sistema com atualização facilitada

O novo sistema está apto a receber modificações futuras, seja na sua estrutura ou no fluxo de informações. A estrutura obtida com auxílio da AOO permite que novas

informações e funcionalidades sejam adicionadas ao sistema sem que haja maior impacto na arquitetura desenhada.

- Maior segurança

Em relação à organização antiga, em que não havia controle de acesso por parte dos usuários, podia-se incluir, alterar ou excluir documentos livremente no antigo repositório de documentos comuns. O novo sistema restringe essas operações a pessoas previamente designadas.

- Ganhos de produtividade e maior lucratividade

Ainda que a mensuração seja difícil, prevê-se economia de tempo na localização de informações em razão da estrutura racional do sistema e devido também à centralização dessas informações, liberando tempo para tarefas mais importantes. Assim, maior rapidez e maior qualidade dispensada às atividades centrais tende a melhorar a percepção dos clientes, e por conseqüência a lucratividade da empresa.

7.3 Considerações sobre o desenvolvimento

Segundo Booch (1996), o processo de desenvolvimento integral de um sistema orientado ao objeto passa por 5 etapas, com maior ou menor ênfase em cada uma delas, dependendo da equipe que o desenvolve.

As principais etapas são:

- Conceituação: estabelecimento de requisitos mínimos
- Análise: desenvolvimento de um modelo com comportamento desejado
- *Design*: criação de uma arquitetura
- Evolução: implementação
- Manutenção: evolução após o desenvolvimento inicial.

Neste trabalho, avançou-se até a fase de análise, com a proposta de um modelo com um conceito definido e requisitos mínimos necessários ao desenvolvimento posterior.

No entanto, a Stratus não fará o *design*, implementação e manutenção do sistema de informações internamente. Ela pretende selecionar uma empresa para esse fim, de acordo com critérios como custo, conceito de desenvolvimento proposto, confiabilidade, tempo de execução, suporte, entre outros.

Capítulo 8

Conclusões

8.1 Metodologia

A utilização da AOO correspondeu à expectativa de promover uma análise de requisitos para um sistema de informações. Esse sistema deve possibilitar o registro, armazenamento e acompanhamento da geração e de trocas de informação da empresa com o ambiente externo, sendo que essas informações devem estar dentro do contexto dos negócios da empresa. O resultado da análise é um primeiro conceito, o qual será a base e prestará auxílio à fase de desenvolvimento final.

A funcionalidade e a flexibilidade da AOO fazem com que o modelo de sistema desenhado não só funcione com as características atuais, mas que também aceite evoluções futuras. A capacidade do novo sistema em aceitar novas informações e funcionalidades sem alterar sua estrutura geral é importante, pois implica no aproveitamento do investimento em sistemas de informações desde o início, sem que haja descarte do que foi desenvolvido. O sistema a ser implementado pode ser aperfeiçoado e complementado, obtendo-se um sistema sempre atualizado e capaz de satisfazer as necessidades dos usuários.

O melhor conhecimento das peculiaridades do problema durante a análise se mostrou valioso, uma vez que foi possível perceber detalhes que não poderiam ser facilmente percebidos sem o uso de uma metodologia adequada. O desenvolvimento dos três modelos, com *use cases*, diagrama de classes e diagrama de seqüências para a análise permitiu a obtenção do sistema sob três óticas distintas, de tal forma que diferentes nuances pudessem ser percebidas e modeladas corretamente. Nesse sentido, o uso da linguagem UML para o trabalho foi fundamental por fornecer as ferramentas mais importantes para a visualização, especificação, construção e documentação do modelo.

Além disso, a estruturação da análise feita no decorrer do trabalho serve de roteiro para o desenvolvimento de um sistema não só na empresa em questão como também em outras empresas com outros ramos de atuação, pois o ferramental utilizado é flexível e pode ser repetido com sucesso em outras aplicações.

8.2 Solução

O produto deste trabalho é um modelo obtido como resultado da análise orientada a objetos. Esse modelo fornece três visões distintas, apresentadas pelos:

- *Use cases*
- Diagramas de classes
- Diagrama de seqüências

Os use cases e os diagramas de classes e seqüências descrevem as necessidades de um sistema completo, que envolvem objetos do domínio do problema e da solução, interações entre objetos e funções a serem executadas pelos usuários.

O modelo proposto é uma alternativa simples, eficiente e estruturada para cobrir as necessidades apresentadas pela empresa. No ambiente de trabalho espera-se que o sistema seja útil, uma vez que seu comportamento foi caracterizado de acordo com funções ao nível dos usuários (como modelado nos *use cases*), incluindo rotinas de registros de comunicação e cadastros (modelados nos diagramas de seqüências). Esta rotina é feita de forma distribuída entre os usuários, mas os responsáveis por projetos têm maior controle sobre eles. Com isso, diminui-se a probabilidade da não observância de atividades acordadas, além de se conseguir uma maior agilidade na busca por informações pelo fato de a estrutura do sistema refletir a estrutura de negócios da empresa (modelada no diagrama de classes), obtendo-se assim ganhos de produtividade.

Deve-se considerar também uma melhora no aspecto de segurança das informações com a inclusão de responsáveis específicos para os projetos e permitindo alteração ou exclusão de conteúdo, que só é permitido a pessoas autorizadas.

Os modelos provenientes da análise serão seguidos na fase seguinte pelo desenvolvimento de sua arquitetura (*design*). Os resultados obtidos nesta análise servirão de referência para o desenvolvimento e, assim, as idéias desenvolvidas neste trabalho repercutirão nas fases posteriores. Assim, dá-se o primeiro passo para que a

fase de *design* possa ser iniciada com rapidez, além de prover uma base sólida sobre a qual todo o desenvolvimento será feito.

Deste modo, há ainda todo um trabalho a ser desenvolvido, com várias etapas a serem percorridas, que poderiam motivar outros trabalhos de formatura para cada uma dessas etapas. No caso, como se deseja contratar uma empresa para a realização do restante do desenvolvimento, a atuação da empresa deve-se se dar de modo a facilitar o processo de desenvolvimento, fornecendo informações e avaliando de tempos em tempos o *design* em desenvolvimento, a execução da implantação, fornecimento treinamento e a manutenção periódica oferecida. Assim, a seleção das empresas prestadoras desses tipos de serviço devem passar por critérios escolhidos pela Stratus, de modo que a escolha recaia sobre empresas capacitadas, com serviço de qualidade e custos viáveis.

Bibliografia

BOOCH, G. **Object Solutions – Managing the object-oriented project**. Addison-Wesley Object Technology Series, 1996.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **Unified Modeling Language User Guide**. Addison-Wesley Object Technology Series, 1998.

COAD, P.; YOURDON, E. **Análise baseada em objetos.** Editora Campus, 1992.

FOWLER, M.; SCOTT, K. **UML Distilled - Applying the standard object modeling language**. Addison-Wesley Object Technology Series, 1997.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. **The Unified Software Development Process**. Addison-Wesley Object Technology Series, 1999.

LEVIN, J. **Structuring Venture Capital, Private Equity and Entrepreneurial Transactions**. Aspen Law & Business, 1998.

PRESSMAN, R. **Software Engineering - A practitioner's approach**. Mc Graw-Hill Book Company Europe, 3rd edition, 1994.

RUMBAUGH, J.; *et al*. **Modelagem e projetos baseados em objetos**. Editora Campus, 1994.