RUBENS SILVA

DEFINIÇÃO DE ARQUITETURAS DE SOFTWARE COM O USO DE UM FRAMEWORK

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para conclusão do curso de MBA em Tecnologia de Software.

São Paulo
2013

RUBENS SILVA

DEFINIÇÃO DE ARQUITETURAS DE SOFTWARE COM O USO DE UM FRAMEWORK

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Tecnologia de Software.

Área de Concentração: Tecnologia de Software

Orientador:
Profa. Dra. Jussara Pimenta Matos

São Paulo
2013

FICHA CATALOGRÁFICA

Silva, Rubens
Definição de arquiteturas de software com o uso de um framework / R. Silva. – São Paulo, 2013. 120 p.

Monografia (MBA em Tecnologia de Software) – Escola Politécnica da Universidade de São Paulo. Programa de Educação Continuada em Engenharia.

1. Framework de arquitetura 2. Arquitetura de software. Universidade de São Paulo. Escola Politécnica. Programa de Educação Continuada em Engenharia II.

DEDICATÓRIA

Dedico este trabalho aos meus pais Maria Lúcia de Fátima Silva e Sebastião Neto da Silva, pois sem o apoio deles, este trabalho jamais seria possível.

AGRADECIMENTOS

Primeiramente à Deus por permitir a conclusão deste curso de pós graduação.

À professora Dra. Jussara Pimenta Matos pelo comprometimento, apoio e orientação com sua perícia e experiência.

À Escola Politécnica da Universidade de São Paulo – EPUSP que concede seu espaço e professores altamente qualificados para os cursos de especialização.

Ao PECE – Programa de Educação Continuada em Engenharia que com sua iniciativa de qualificar profissionais de mercado, contribuiu para meu aperfeiçoamento e capacitação profissional para um mercado altamente competitivo.

Aos meus pais e meus irmãos que acreditaram e me apoiaram desde o início.

RESUMO

Este trabalho consiste em desenvolver um framework de arquitetura com base em princípios, padrões, normas e diretrizes para apoiar as atividades de especificação, análise, concepção e descrição de arquiteturas de software. Manter a continuidade dos negócios com qualidade e produtividade através do apoio de sistemas corporativos torna-se um desafio quando não são adotados métodos e padrões para documentar apropriadamente as decisões pelas quais o sistema é projetado. Para restringir os inúmeros fatores que influenciam um projeto de arquitetura e manter o enfoque nas principais considerações à respeito dos objetivos do negócio, faz-se necessário a adoção de métodos, padrões e diretrizes para apoiar o desenvolvimento de sistemas corporativos.

Palavras-chave: Arquitetura, Framework de arquitetura, Pontos de vista.

ABSTRACT

This work consist in specify an architecture framework based on principles, standards, norms and guidelines to support the activities of specification, analysis, design and description of software architectures. Maintaining business continuity with quality and productivity through the support of enterprise systems becomes a challenge when they are not adopted methods and standards for suitably documenting the decisions by which the system is designed. To narrow the number of influence factors about architectural design, keeping the focus on the key concerns regarding the business objectives, it is necessary to adopt methods, standards and guidelines to support the design of enterprise systems.

Keywords: Architecture, Architecture Framework, Viewpoint.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADD	Attribute Driven Design
ASR	Architecturally Significant Requirement
DDD	Domain Driven Design
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronic Engineers
ISO	International Organization for Standardization
RM-ODP	Reference Model of Open Distributed Processing
SOA	Software Oriented Architecture
UML	Unified Modeling Language

SUMÁRIO

1. INTRODUÇÃO

1.1 Motivações

Dentre as motivações para a realização deste trabalho está a dificuldade para adoção de uma metodologia que aborde as características e cuidados inerentes aos diferentes tipos de domínios, para a definição de arquiteturas de software sustentáveis que efetivamente agregam valor ao negócio.

No que diz respeito à continuidade e evolução do negócio, a falta de padrões para documentação das diferentes hipóteses que os interessados fazem sobre os sistemas produzidos implicam em soluções sem maturidade que não contribuem eficazmente para confiabilidade e segurança das manutenções.

Em relação aos requisitos, a ausência de padrões para especificar os requisitos arquiteturalmente significativos de modo claro, preciso e não ambíguo, dificulta o raciocínio sobre o que agrega valor para os interessados e sobre como uma modelagem estratégica implica em redução de custos e esforços com manutenções e alterações em larga escala, para resolver questões não associadas com as funções do sistema.

Quando a modelagem e/ou a descrição da arquitetura do sistema não revelam as intenções originais pelas quais o software foi projetado, as manutenções posteriores, e a evolução do sistema como um todo se torna restrita, o entendimento do código é vulnerável, as intenções e as suposições implícitas levam a introdução de código não confiável e o corrompimento da arquitetura do software como um todo.

1.2 Objetivo

O objetivo deste trabalho é desenvolver um framework de arquitetura que contemple as considerações[1] dos interessados em um domínio de uma empresa de recuperação de ativos financeiros e a identificação de um conjunto de atributos de qualidade associados com as questões significativas para o sucesso do sistema. Além de apoiar o projetista no raciocínio sobre as questões significativas que contribuem para a automação eficaz dos processos de negócio.

Ao designar um conjunto de atividades e modelos de forma organizada e estruturada espera-se conceber os esboços para um documento de descrição da arquitetura constituído por diferentes visões arquitetônicas que possam ser interpretadas consistentemente e avaliadas apropriadamente.

Embora as distinções entre modelos de diferentes visões arquitetônicas sejam fundamentais para esclarecer aspectos específicos dos sistemas para diferentes grupos de interessados, tais modelos possuem relacionamentos de dependência entre si que quando não são abordados apropriadamente podem levar à inconsistência e interpretações equivocadas.

A contribuição para evitar que essas distinções impliquem em inconsistências sobre os modelos concebidos será obtida através de especificações de regras de correspondência[2] para governar os principais relacionamentos de dependência entre os modelos.

---

[1] Uma consideração pode se manifestar de muitas formas, tal como a relação a um ou mais necessidades dos interessados, objetivos, expectativas, requisitos, restrições de projeto, suposições, dependências, atributos de qualidade, decisões de arquitetura, riscos ou outras questões pertinentes ao sistema [ISO/IEC/IEEE 42010:2011].

[2] Correspondências e regras de correspondências são usadas para expressar e impor relações arquitetônicas tal como composição, refinamento, consistência, rastreabilidade, dependência, restrição e obrigação. [ISO/IEC/IEEE 42010:2011]

1.3 Justificativas

Dentre as principais contribuições que o desenvolvimento do framework de arquitetura visa alcançar estão orientações para identificar os níveis de qualidade esperados pelos interessados em um domínio particular, documentação descrita implicando em curvas de aprendizado menores, padronização de modelos e regras designadas para o público alvo contribuindo para que questões significativas para o domínio deixem de ser de livre interpretação pelos interessados.

A dispersão do conhecimento entre diversos interessados e artefatos implica na fragmentação da modelagem e duplicidade de código, elevando as margens para interpretações ambíguas sobre o código e aumentando os custos com manutenções.

Usualmente os especialistas técnicos mesmo quando concebem soluções apropriadas não documentam seus raciocínios e suas próprias decisões, permitindo que a falta de conhecimento sobre o negócio e as intenções iniciais do projeto sejam rapidamente descaracterizadas quando surgem as mudanças.

A descrição de arquiteturas por meio de um conjunto de pontos de vista bem organizado e estruturado deve contribuir para a assimilação do conhecimento sobre o negócio com mais eficácia, no que diz respeito às diferentes considerações que os interessados fazem sobre o sistema.

1.4 Estrutura do Trabalho

Este trabalho está organizado por capítulos da seguinte forma:

O Capítulo 1, INTRODUÇÃO apresenta as motivações para a realização desse trabalho, o objetivo, as justificativas e a estrutura.

O Capítulo 2, REVISÃO BIBLIOGRÁFICA apresenta definições fundamentais sobre arquitetura de software e a terminologia básica utilizada no trabalho com o objetivo de facilitar a compreensão de seus fundamentos, os diferentes contextos nos quais

ela reside, os métodos e técnicas adotados para apoiar a proposta principal do trabalho.

O Capítulo 3, PROPOSTA apresenta o desenvolvimento do framework de arquitetura para definição de arquiteturas de software no domínio de uma empresa de recuperação de ativos financeiros.

O Capítulo 4, ESTUDO DE CASO tem como finalidade aplicar o conjunto de pontos de vista do framework de acordo com as atividades propostas, concebendo os modelos com respeito às regras de correspondência que governam as relações de dependência entre eles.

O Capítulo 5, CONSIDERAÇÕES FINAIS, apresenta as considerações finais e propostas de trabalhos futuros.

Finalmente, as referências bibliográficas utilizadas ao longo do trabalho.

2. REVISÃO BIBLIOGRÁFICA

2.1 Considerações iniciais

Este capítulo apresenta os conceitos relacionados à arquitetura de software, incluindo a terminologia básica utilizada no trabalho com o objetivo de facilitar a compreensão de seus fundamentos, os diferentes contextos nos quais ela reside, os métodos e técnicas adotados para apoiar a proposta do trabalho.

2.2 Arquitetura de software

Arquitetura diz respeito aos conceitos fundamentais ou propriedades de um sistema em seu ambiente incorporado nos seus elementos, relacionamentos e nos princípios de sua concepção e evolução [ISO/IEC/IEEE 42010:2011].

Com o advento das linguagens de descrição de arquitetura e a correspondência entre categorias de problemas e soluções arquitetônicas apropriadas, as empresas começaram a considerar explicitamente as arquiteturas e os especialistas técnicos são frequentemente reconhecidos como arquitetos [SHAW-09].

Os problemas enfrentados pelos engenheiros de software estão cada vez mais situados em contextos sociais complexos e a contextualização dos problemas está cada vez mais difícil [SHAW-09]. Sensibilidade e técnicas para reconhecer e resolver tanto classes de problemas existentes quanto novas classes de problemas que surgem são significantes para medir o progresso do desenvolvimento de software rumo à uma disciplina de engenharia.

Para SHAW (2009), a democratização da internet foi um fator significante que contribuiu com o aumento do número de pessoas desenvolvendo sofware, inclusive pessoas que não são profissionais de computação e que agora exercem o controle direto sobre suas atividades computacionais. Assim, o âmbito de uma disciplina de engenharia de software se expandiu para além do software sob medida para incluir suporte à software largamente utilizado e criado pelo público.

No que diz respeito à codificação, a programação orientada a objetos certamente é uma melhoria sobre a programação estruturada tradicional [CLEMENTS-09]. Considerando que a programação estruturada é a arquitetura agnóstica, os frameworks que apoiam a programação orientada a objetos incorporam as decisões de arquitetura dificultando o pensamento arquitetônico.

Arquitetos anciosos para se comunicar com os programadores, sem uma linguagem voltada para a arquitetura, submetem-se à terminologia dos programadores [CLEMENTS-09]. A restrição do pensamento imposta pela falta de linguagem prejudica o arquiteto na escolha da arquitetura mais adequada para seu projeto.

Na medida em que o aprimoramento contínuo de processos e a modernização tecnológica traz benefícios para as empresas, e com o aumento da quantidade de informações e do público cada vez mais exigente, estes fatores introduzem novos desafios aos projetos de arquitetura. Em nível macro, a gestão estratégica de uma empresa com enorme volume de informação, sem o apoio de ferramentas para projeção dos cenários atual e futuro, pode levar à tomadas de decisão não confiáveis.

Além da complexidade para gerir as empresas com volumes de informação cada vez maior, a formulação de novos modelos de negócio pela internet requer infraestruturas mais robustas, para suportarem a execução de sistemas altamente escaláveis. Tais infraestruturas podem compreender um poder computacional de magnitude tão ampla que os custos para mantê-las tornariam-se injustificáveis.

Avanços tecnológicos contribuiram para o desenvolvimento de linguagens de programação adjacentes à linguagem humana e a incorporação de camadas de software sobre o hardware que permitiu alavancar o desenvolvimento de software, integrado entre diferentes plataformas, para automatizar e apoiar processos de negócio de modo interoperável[3].

[3] Sistemas que precisam ser intrinsecamente interoperáveis podem requerer adoção a um estilo de arquitetura orientada a serviços - SOA, onde um de seus principais objetivos é estabelecer a interoperabilidade nativa entre serviços, eliminando a necessidade de integração [ERL-07].

Embora os praticantes da programação orientada a objetos tenham se beneficiado pelo apoio de frameworks e ferramentas, onde muitas decisões de arquitetura estejam embutidas em frameworks, a complexidade na arquitetura dos sistemas corporativos apenas mudou de perspectiva.

2.3 Relação entre a arquitetura e os requisitos

Profissionalmente, o arquiteto estará sempre fazendo escolhas[4] com base em requisitos funcionais, restrições do negócio, restrições técnicas e atributos de qualidade [EELES-10]. De fato, não é economicamente viável resolver todas as considerações dos interessados [EELES-10]. O arquiteto deverá tomar decisões para habilitar medidas de respostas apropriadas em alguns cenários e inibir medidas de respostas em cenários conflitantes [BASS-12].

Para facilitar a tomada de decisões em um projeto arquitetural, [BASS-12] aborda uma série de táticas[5] e métodos para definir arquiteturas com base em requisitos arquiteturalmente significativos (do inglês, architecturally significant requirement - ASR). Segundo [BASS-12], estes requisitos tem profundo efeito na arquitetura, isto é, sem eles, a arquitetura seria diferente.

2.4 Requisitos arquiteturalmente significativos

Requisitos arquiteturalmente significativos (ASR's) determinam e moldam uma arquitetura de software. Caso esses tipos de requisitos apresentem erros, estejam incompletos, imprecisos ou não detalhados suficientemente, então a arquitetura de software concebida com base neles provavelmente possuirá erros [LIANPING-13].

Em seu estudo empírico realizado com entrevistas de profissionais de noventa empresas em diversos domínios, LIANPING (2013) apresenta suas considerações em forma de um framework baseado em evidências, que distingue sistematicamente

---

4 Quando existem questões onde não há uma solução que satisfaz a todos os requisitos igualmente bem, os arquitetos devem escolher habilitar alguns requisitos e abrir mão dos requisitos conflitantes [LIANPING-13].
5 Táticas são decisões de projeto que influenciam na resposta a um cenário de atributo de qualidade. Elas afetam a resposta do sistema ou parte dele a um estímulo [BASS-12].

os ASR’s dos outros tipos de requisitos. O framework consiste de quatro conjuntos de características:

- Definição: requisitos arquiteturalmente significativos são aqueles que tem um impacto mensurável sobre a arquitetura de software, onde a medida é determinada pelo custo, monetário ou não, das mudanças.
- Descrições: frequentemente os requisitos arquiteturalmente significativos são difíceis de definir e articular, tendendo a ser vagamente expressos, inicialmente negligenciados, permanecendo ocultos em outros requisitos, são subjetivos, variáveis e situacionais. Determinar uma lista definitiva de ASR’s é inviável. No entanto, durante a análise de requisitos, é possível dizer que certos requisitos provavelmente serão arquiteturalmente significativos.
- Indicadores: embora o custo das mudanças seja uma medida significativa, obter o custo preciso de um requisito é desafiador. Apesar das estimativas de custo serem realizadas após a coleta de requisitos, uma estimativa pode ser importante para designar um requisito arquiteturalmente significativo. É possível distinguir os ASR’s dos demais requisitos por meio de indícios como: impacto grande, implicações que levam à escolha entre um requisito ou outro, rigidez (restritivo, limitador, inegociável), descarte de suposições existentes, dificuldade de alcançar.
- Heurísticas: julgar se um requisito possui grande impacto, se possui implicações que levam à escolha entre um requisito ou outro, se requer rigorosamente uma abordagem de projeto, se descarta as suposições existentes ou se é difícil de alcançar pode requerer conhecimento substancial da solução. Muitos engenheiros de requisitos não estão aptos a utilizar tais indicadores para identificar os ASR’s. Para orientá-los foram descobertas um conjunto de características definidas como heurísticas [LIANPING-13]. As heurísticas podem orientar os engenheiros de requisitos a questionar pro-ativamente os usuários no que diz respeito as considerações que são prováveis de serem arquiteturalmente significativas. Tais heurísticas são estabelecidas pela associação do requisito com: atributos de qualidade, principais características, restrições e o ambiente de aplicação.

Na prática, não existe uma propriedade para cada requisito que ajude a identificar se é arquiteturalmente significativo ou não. Para distinguir os ASR's dos demais requisitos é preciso utilizar características pragmáticas que habilitam a descoberta dos ASR's [LIANPING-13].

As heurísticas identificadas por LIANPING (2013) destacam requisitos que tendem a ser arquiteturalmente significativos através de uma terminologia familiar para os interessados, sem a necessidade de um conhecimento profundo do espaço da solução.

2.5 Arquitetura de software em um contexto técnico

Em meados de 1980, os pesquisadores de software começaram a reconhecer a importância de se tornar explícitas as decisões sobre a organização geral de um sistema [SHAW-09]. O reconhecimento sobre as variações das arquiteturas abstratas que guiam a modelagem de sistemas de software levou a classificação de diferentes tipos de componentes, protocolos abstratos (conectores) que conectam esses componentes e os estilos de arquitetura que guiam o uso consistente de componentes e conectores compatíveis.

De acordo com SHAW (1996), um estilo de arquitetura define uma familia de sistemas em termos de padrão de organização estrutural. Mais especificamente, um estilo de arquitetura define o vocabulário de componentes e tipos de conectores e um conjunto de restrições sobre a forma como eles podem ser combinados.

2.5.1 Padrões de projetos

Padrões de projetos documentam soluções comprovadas para problemas recorrentes de elementos de modelagem, que resolvem problemas em contextos particulares [ROZANSKI-11].

Para BUSCHMANN (2007), padrões de projetos fornecem um vocabulário para expressar visões arquiteturais com clareza, modelagem representativa e implementação detalhada, permitindo que desenvolvedores se comuniquem com eficiência e menos ambiguidade.

O uso de padrões de projetos tem sido uma das práticas mais amplamente adotadas pelos profissionais de arquitetura de software [BASS-12], embora depois de implementados não sejam mais detectáveis, sendo um risco durante o desenvolvimento inicial [CLEMENTS-09].

2.5.2 Interfaces econômicas

De uma forma geral, arquiteturas de software sofrem de complexidades desnecessárias através da incorporação de flexibilidade excessiva, características desnecessárias, onde o enfoque é dado ao reuso ao invés da usabilidade[6] [BUSCHMANN-10b]. Para que os sistemas sejam fáceis de evoluir e manter é necessário projetar interfaces com clareza. Um projeto econômico equilibra clareza, compreensão e abstração. Algumas abordagens de desenvolvimento como desenvolvimento dirigido por testes e modelagem dirigida por domínios auxiliam o arquiteto a dar enfoque no que projetar identificando naturalmente os conceitos usuais.

As interfaces devem conter operações com um propósito claro e significativo a partir da perspectiva de uso. Também devem dar enfoque a um conjunto mínimo de operações necessárias para cumprir com as responsabilidades do componente. Para apoiar a modelagem de interfaces econômicas, existem alguns padrões como: método combinado (do inglês, combined method), interface de extensão (do inglês, extension interface), objeto de contexto (do inglês, context object) e objeto de transferencia de dados (do inglês, data transfer object) [BUSCHMANN-11b].

2.5.3 Visibilidade

A visibilidade diz respeito à objetividade e expressividade dos artefatos que descrevem e constituem arquiteturas de software [BUSCHMANN-10b]. Muitas vezes as interfaces tem pouca expressividade e seu significado é obscuro, a documentação é inexistente ou excessiva e mal comunicada. As consequências

[6] Nesse contexto, usabilidade refere-se ao público de desenvolvimento, não aos usuários finais.

incluem acentuadas curvas de aprendizado, manutenibilidade e manutenção não confiáveis. Para tornar visíveis os conceitos no código, é preciso representar esses conceitos do domínio no código de modo que não seja de livre interpretação. A introdução de um tipo explícito para um conceito reduz código duplicado e suposições implícitas. A modelagem dirigida por domínios está enraizada no princípio da visibilidade.

Padrões são outro meio de tornar os conceitos visíveis. Por exemplo, o padrão estratégia (do inglês, strategy) materializa os algoritmos a utilizar, objetos de valor (do inglês, value objects) materializam conceitos simples do domínio e o padrão camadas (do inglês, layers) materializa diferentes níveis de abstração do sistema.

Uma das técnicas que ajuda a aprimorar a visibilidade é separar claramente conceitos distintos. O espaçamento apropriado conduz naturalmente a projetos modulares com conceitos claramente distinguíveis e visíveis [BUSCHMANN-10b]. Por exemplo, ao separar os módulos em camadas, o enfoque no domínio do problema deve ser primário em relação ao domínio de infraestrutura. A delegação através de raízes conceituais expressadas como interfaces puras pode levar a uma separação ainda mais rigorosa.

2.5.4 Adequação para o negócio

Projetos de software tendem a manter o mínimo possível de diálogo com seus clientes. Os projetos de software que não mantém o diálogo contínuo com os interessados no negócio perdem uma grande oportunidade de aprender com com esses interessados até para garantir que seus sistemas funcionem apropriadamente [BUSCHMANN-11a].

Para compreender a perspectiva geral que o interessado tem sobre o sistema, um bom ponto de partida é entender e capturar um modelo de domínio concreto, compreender os principais fluxos de trabalho, as tarefas e as atividades que ocorrem naquele domínio [BUSCHMANN-11a].

Modelos de domínio, em conjunto com especificações de cenários, também são apropriados para elicitar e esclarecer os principais requisitos. Cenários ajudam a especificar os estímulos relevantes e as respostas esperadas para todas as tarefas relevantes para o sistema. Mapeando cenários para o modelo de domínio extende essa visão de como o sistema deverá desempenhar tarefas relevantes como fluxos de trabalho no domínio de aplicação específico, incluindo requisitos de qualidades relevantes tais como desempenho [BUSCHMANN-11a].

Utilizando modelos de domínio e cenários como uma orientação, a especificação e a clareza do escopo do sistema e dos requisitos serão baseados em menos suposições e mais em necessidades reais e valor agregado [BUSCHMANN-11a].

2.5.5 Atributos de qualidade

Tecnicamente o objetivo das arquiteturas é definir soluções apropriadas para responder aos eventos que chegam ao sistema, sejam esses eventos estimulados em tempo de execução ou em tempo de desenvolvimento [BASS-12]. Normalmente cada um desses eventos está associado à um atributo de qualidade [BASS-12].

Atributo de qualidade[7] é uma propriedade do sistema mensurável ou testável utilizada para indicar o quão bem o sistema atende as necessidades de seus interessados [BASS-12]. As estruturas criadas pelos projetos de arquitetura inibem ou habilitam alguns desses atributos [BASS-12]. A norma [ISO-25010:2011] padroniza alguns dos principais atributos de qualidade do produto de software, conforme mostra a figura 2-1.

[7] O estabelecimento da qualidade de uma arquitetura sendo descrita ou de um documento de descrição de arquitetura são fatores de avaliação arquitetural [ISO/IEC/IEEE 42010:2011]. A norma [ISO/IEC/IEEE 42010:2011] não impõe condições requeridas para considerações sobre a qualidade e não requer que os resultados das avaliações sejam registrados.

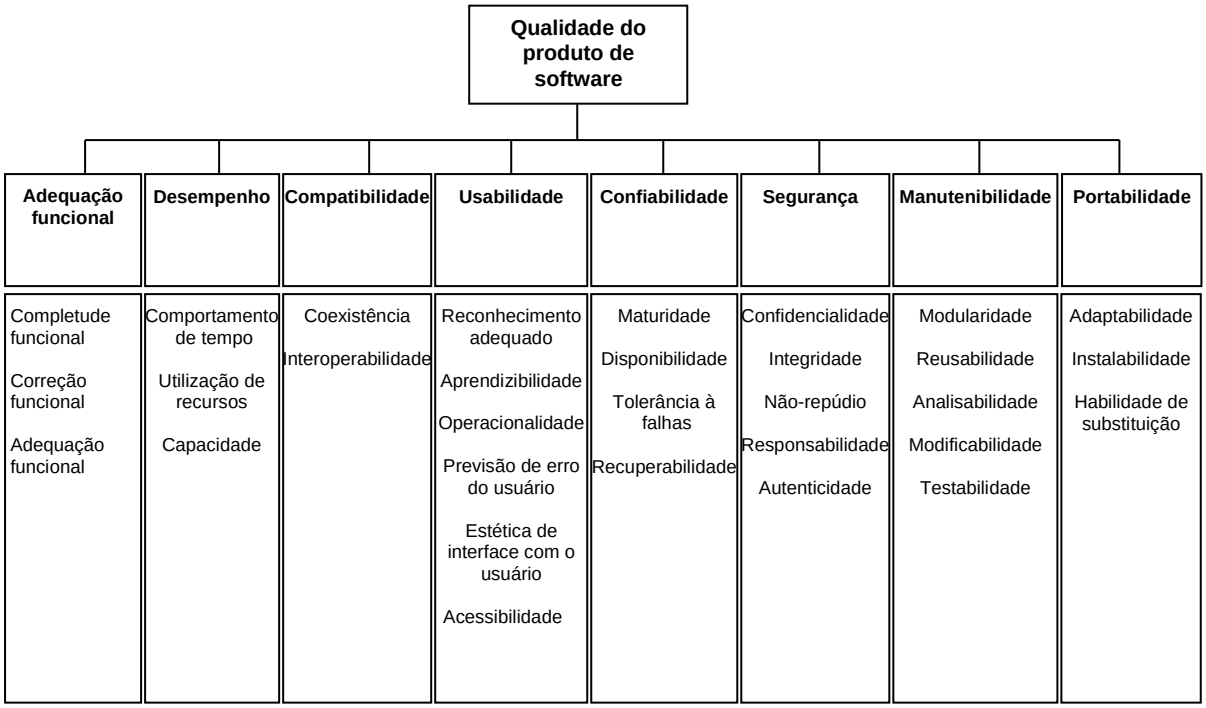


Figura 2-1 Qualidade do produto de software [ISO-25010:2011].

Para LIANPING (2013), os atributos de qualidade se caracterizam por heurísticas que orientam o engenheiro de requisitos à questionar os usuários proativamente sobre considerações que, usualmente, são significantes para a arquitetura, mas que os usuários frequentemente não mencionam.

No entanto, nem sempre alta qualidade é requerida pelo sistema e nem sempre as arquiteturas de baixa qualidade são ruins [BUSCHMANN-10c]. Qualquer atributo de qualidade que não seja necessário, aumenta a complexidade arquitetural do sistema, os custos para mantê-lo e diminui a aceitação do usuário.

2.6 Aquitetura de software para um domínio específico

Uma das técnicas que contribuem com o reuso efetivo e ajudam a diminuir suposições conflitantes entre diferentes tipos de interessados é trabalhar em um domínio especializado da arquitetura [GARLAN-09]. Com isso, é possível restringir os tipos de componentes permitidos, as interações entre eles, os tipos de interfaces, eliminando assim uma parte da variabilidade que pode conduzir o projeto à incompatibilidade arquitetural [GARLAN-09].

Para tratar aspectos específicos do domínio, BUSCHMANN (2010a) define quadro do problema[8] como um padrão que descreve uma classe recorrente de problema designando a nomenclatura e o limite do problema. O resultado de uma combinação de quadros de problema é a descrição do problema e seus fatores influenciantes que proporcionam orientações concretas[9] para definição das soluções de arquitetura.

Em geral, os sistemas seguem mais de um quadro de problema. Consequentemente suas arquiteturas se apresentam como combinações de abordagens e estilos de arquitetura correspondentes [BUSCHMANN-10a].

Complementando a escolha dos estilos de arquitetura, duas considerações com significante impacto para a arquitetura e a escolha da tecnologia subjacente são destacadas [BUSCHMANN-10a]: as propriedades essenciais dos objetos de domínio e o potencial do fluxo de trabalho do domínio para distribuição e concorrência. Tais considerações caracterizam decisões de arquitetura como questões de latência, vazão e a escolha entre processar informações específicas em memória ou em um banco de dados.

Segundo BUSCHMANN (2010a), a orientação que precisamos para tomar essas decisões é obtida através das propriedades fundamentais dos objetos em um domínio de aplicação específico. É preciso distinguir entre objetos de domínio que representam coisas físicas do mundo real de coisas lógicas como fluxos de trabalho.

Para evidenciar tais percepções, BUSCHMANN (2010a) relata sua experiência com sistemas de processamento concorrente e sistemas distribuídos. Modelagens de sucesso para sistemas distribuídos e altamente concorrentes minimizam o uso da rede e a sincronização de processos [BUSCHMANN-10a]. É possível alcançar esse objetivo retratando a distribuição e o paralelismo diretamente na arquitetura, dentro do domínio da aplicação, utilizando alguns princípios [BUSCHMANN-10a]:

[8] do inglês, problem frames.
[9] A descrição de métodos, técnicas e estilos de arquitetura que contribuem efetivamente para a solução de uma classe generalizada de problemas em um domínio particular. [BUSCHMANN-10a].

- Localidade: os dados e o comportamento são fornecidos e mantidos no mesmo local físico onde eles são necessários.
- Independência: tarefas que executam em paralelo no mundo real deveriam executar em paralelo também no software.
- Divisão do trabalho: dados cuja manipulação pode ser dividida em diversos passos devem ser manipulados apropriadamente no software, por exemplo, em manipulação de imagens.

Segundo BUSCHMANN (2010a) é possível obter orientações para as escolhas em um projeto através da natureza e das propriedades dos tipos de tarefas que os sistemas devem executar.

Arquiteturas de software sustentáveis retratam as propriedades fundamentais do domínio de aplicação de forma explícita assegurando que o mundo virtual possa simular apropriadamente o mundo real [BUSCHMANN-10a]. Essa introspecção é independente dos requisitos, do negócio e de aspectos organizacionais de desenvolvimento de software que o sistema deve atender.

Utilizando padrões como quadro de problema e a modelagem dirigida por domínios, arquitetos pragmáticos obtêm orientações para designar corretamente uma abordagem arquitetônica e a tecnologia subjacente [BUSCHMANN-10a]. As decisões mais fundamentais da arquitetura são aquelas que custam para modificar e baseiam-se em uma fundamentação adequada ao invés da intuição e experiência pessoal única.

2.7 Especificação de cenários de qualidade

De acordo com [BUSCHMANN-10c] um conjunto de cenários arquiteturalmente significantes é uma base ideal para orientar de forma concreta o projeto de arquitetura. A especificação de cenários para as tarefas relevantes do sistema e o respectivo mapeamento para um modelo de domínio aborda como o sistema desempenhará os fluxos de trabalho no domínio de aplicação [BUSCHMANN-11a].

No que diz respeito ao processo, KRUTCHEN (1995) defende que o desenvolvimento de software deve seguir um processo iterativo, onde a arquitetura é

construída sobre um protótipo evolutivo. Tal protótipo é testado, medido e analisado, permitindo que os requisitos sejam refinados, amadurecidos e melhor compreendidos durante as iterações subsequentes, contribuindo para mitigar riscos associados com a arquitetura.

Para expressar requisitos de qualidade com precisão e significado, BASS (2012) define um método que consiste de seis partes:

- Estímulo: condição que requer uma resposta quando chega ao sistema.
- Origem do estímulo: uma entidade (humano, sistema ou qualquer outro atuador) que gera o estímulo.
- Resposta: atividade tomada como resultado da chegada do estímulo.
- Medida de resposta: uma forma de medida em que o requisito pode ser testado.
- Ambiente: o estímulo ocorre abaixo de certas condições. O sistema pode estar sobrecarregado, sobre condições normais de operação, ou outro estado relevante.
- Artefato: algum artefato é estimulado. Pode ser uma coleção de sistemas, o sistema inteiro ou apenas parte dele. Com frequência é o sistema inteiro.

Cada cenário é especificado como um evento que chega à uma parte do sistema ou no sistema como um todo, em um instante específico que corresponde ao estado em que o sistema se encontra. Esses eventos precisam ser controlados apropriadamente para que o retorno seja enviado de acordo com a medida de resposta esperada [BASS-12].

2.8 Modelagem dirigida por atributos de qualidade

O método de modelagem dirigido por atributos de qualidade (ADD) baseia-se na abordagem de “geração e teste” no qual os elementos são modelados através de um processo iterativo de decomposições sucessivas [BASS-12]. De acordo com BASS (2012), esse método compreende as seguintes atividades:

- Inicialmente são selecionados um conjunto de hipóteses. As hipóteses podem ser sistemas existentes, frameworks, padrões e táticas, decomposição do domínio ou verificações[10].
- Uma modelagem candidata é formulada utilizando as hipóteses selecionadas.
- A modelagem candidata é testada.
- Com base nos resultados dos testes realizados, o próximo conjunto de hipóteses é gerado.

Os artefatos resultantes não descrevem completamente a arquitetura, eles esboçam soluções nas quais as principais abordagens do projeto foram selecionadas e avaliadas. As cinco etapas do ADD são:

1. Escolher um elemento para modelagem. Na primeira iteração do ADD o elemento de modelagem é o sistema inteiro. A partir da segunda iteração o elemento será um dos elementos resultantes da iteração anterior.
2. Identificar os ASR's para o elemento selecionado.
3. Definir uma solução para o elemento escolhido utilizando como apoio uma ou mais hipóteses.
4. Verificar e refinar os requisitos e gerar as entradas para a próxima iteração.
5. Repetir as etapas de 1 a 4 até que todos os ASR's tenham sido resolvidos.

A cada iteração do método, os elementos são refinados. É importante que os modelos utilizados como entrada a partir da segunda iteração sejam provenientes da saída da iteração anterior.

A escolha dos elementos para modelagem pode ser feita por meio de uma das seguintes estratégias de refinamento: em largura ou em profundidade. Escolher a estratégia em largura significa que todos os elementos de segundo nível serão modelados antes de qualquer elemento de terceito nível e assim por diante. Na estratégia em produndidade uma cadeia inteira de elementos é modelada antes que a segunda cadeia de elementos seja modelada.

---

10 Do inglês checklists.

2.9 Framework de arquitetura

Atualmente muitos arquitetos devem trabalhar com um ou mais frameworks de arquitetura conforme as diretrizes da organização, seus clientes, o método empregado ou devido a outras forças [EMERY-09].

Embora a essência da ontologia da IEEE-1471 (2000) seja estabelecer um modelo de referência para descrever arquiteturas, é conveniente estender a ontologia para abordar frameworks de arquitetura definindo as convenções e as práticas comuns para descrição de arquiteturas estabelecida em um domínio específico ou para uma comunidade de interessados [EMERY-09].

De acordo com [EMERY-09] um framework de arquitetura precisa identificar um grupo de interessados, um conjunto de considerações, um conjunto de pontos de vista que abordam tais considerações e quaisquer correspondências a serem aplicadas entre visões resultantes da aplicação dos pontos de vista. A figura 2-2 apresenta de forma macro os conceitos de um framework e seus relacionamentos.

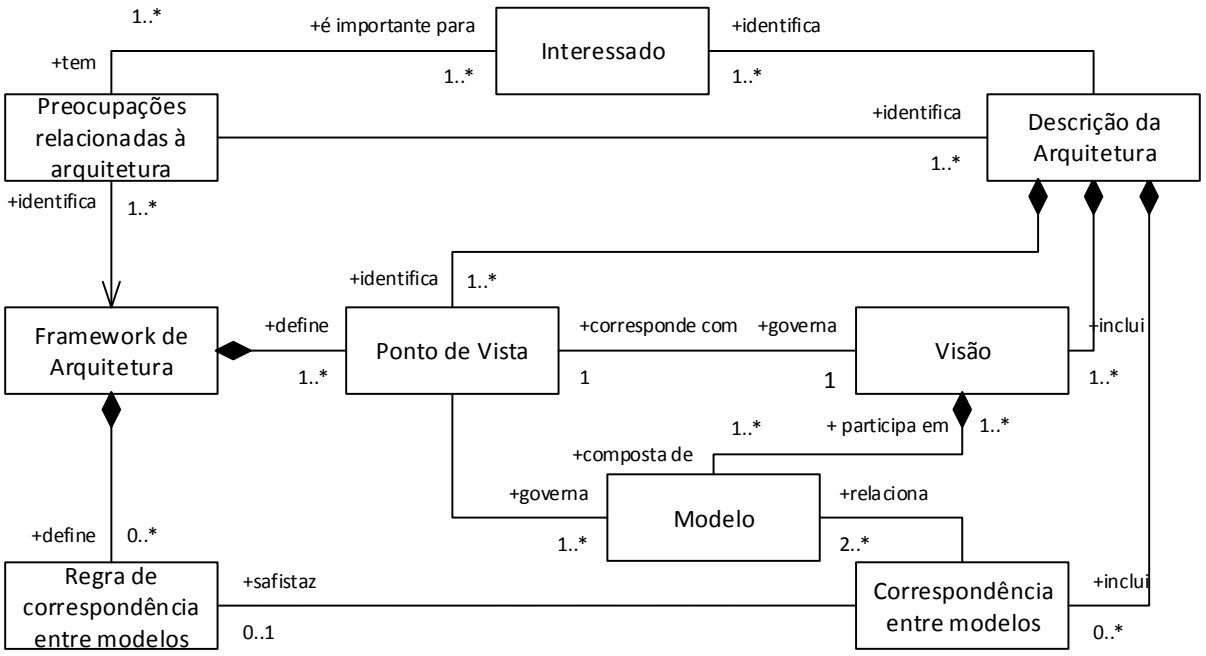


Figura 2-2 Metamodelo de um framework de arquitetura [EMERY-09]

Uma das implicações de se usar múltiplas visões nas descrições de arquitetura é a necessidade de expressar dependências, em particular as correspondências, entre as visões [EMERY-09].

2.9.1 Pontos de vista

Ponto de vista é uma especificação das convenções para construir e usar uma visão, um padrão ou modelo para desenvolver visões individuais estabelecendo as finalidades e o público alvo da visão e as técnicas para a sua criação e análise. [IEEE-1471-2000].

A ideia principal da norma IEEE-1471 (2000) foi introduzir pontos de vista como um mecanismo para codificar melhores práticas na descrição de arquiteturas. Cada ponto de vista especifica os interesses que devem ser abordados, os interessados, as notações, os modelos e os métodos que devem ser usados para criar, interpretar e analisar uma visão resultante da aplicação de um ponto de vista [EMERY-09].

A [ISO/IEC/IEEE 42010:2011] estende a ontologia da [IEEE-1471-2000] para padronizar o conceito de framework de arquitetura com a adição de regras para correspondência entre modelos que visam capturar explicitamente as relações entre os modelos definidos pelos pontos de vista em uma descrição arquitetural [EMERY-09].

2.9.2 Biblioteca de pontos de vista

Bibliotecas de pontos de vista permitem aos usuários especificarem as definições dos pontos de vista que poderiam ser reusados de um documento de descrição de arquitetura para outro para capturar abordagens tal como os modelos 4+1 do Kruchten's e o RM-ODP[11] [EMERY-09].

A definição das bibliotecas de pontos de vista tinha como expectativa a padronização de um conjunto de pontos de vista bem definido e codificado dentro de

[11] Nos termos da [ISO/IEC/IEEE 42010:2011] tanto o modelo de visões 4+1 quanto o RM-ODP são designados como frameworks de arquitetura.

organizações como conhecimento corporativo ou como parte de métodos ou normas [EMERY-09].

Desde a publicação da IEEE-1471 (2000), a prática de arquitetura de software continuou a evoluir, arquiteturas corporativas construídas com base no Zachman framework também emergiram como, por exemplo, DoDAF, MoDAF, ToGAF [EMERY-09].

2.9.3 Regras de correspondência entre modelos

As correspondências entre modelos expressam relações entre um ou mais modelos, podendo ser capturadas graficamente ou em tabelas. Tais correspondências podem ser utilizadas para associar visões para expressar consistência, rastreabilidade, refinamento e outras dependências [EMERY-09].

Com frequência os arquitetos procuram impor restrições entre tipos de modelos e então demonstram que a restrição é satisfeita pela arquitetura [EMERY-09]. Para expressar tais restrições a norma [ISO/IEC/IEEE 42010:2011] introduz o conceito de regras de correspondência entre modelos [EMERY-09].

As regras de correspondência entre modelos podem ser expressas em um documento de descrição de arquitetura ou como parte da especificação de um framework de arquitetura [EMERY-09].

2.9.4 Descrição de arquiteturas

Embora não especificado pela norma [ISO/IEC/IEEE 42010:2011], implica-se que a identificação dos pontos de vista deve preceder a formulação das visões associadas e as regras de correspondências devem ser definidas antes das correspondências, assim um documento de descrição de arquitetura utilizará os pontos de vista e as regras de correspondência selecionadas para documentar a arquitetura de maneira eficaz [EMERY-09].

2.9.5 Documento de descrição de arquitetura

Para EMERY-(09), um documento de descrição de arquitetura consiste em:

- Framework de arquitetura: Identifica os interessados, as considerações, os pontos de vista e as regras de correspondência.
- Conjunto de visões e correspondências que satisfazem ao framework.
- Razões e informações administrativas necessárias.

A concepção de documentos de arquitetura com base em múltiplos frameworks pode causar implicações nas atividades de revisão da documentação, onde os revisores poderão identificar considerações abordadas sem conformidade com o framework adotado [EMERY-09].

2.9.6 Especificação de frameworks de arquitetura

No âmbito da norma IEEE-1471 (2000), uma descrição arquitetural aborda as considerações dos interessados conhecidos para o sistema de interesse [EMERY-09].

Contudo, no momento em que a especificação do framework é realizada, os interessados e suas considerações talvez não sejam conhecidos [EMERY-09]. Para tratar dessas questões, os frameworks de arquitetura podem proporcionar orientações adicionais para definir grupos típicos de interessados ou taxonomias de considerações [EMERY-09].

A prática tem mostrado que os desenvolvedores de frameworks frequentemente conhecem os grupos de interessados no domínio do framework [EMERY-09]. Os interessados motivam o conjunto de considerações das quais o framework deverá abordar [EMERY-09].

Para determinar os pontos de vista a serem incluídos no framework, é necessário identificar as considerações relacionadas com a arquitetura. Cada ponto de vista explica as convenções simbólicas utilizadas na visão e estabelece a base para interpretá-la [EMERY-09].

Além dos requisitos mínimos, os desenvolvedores de frameworks são livres para adicionarem não somente modelos e ferramentas, mas também diretrizes, princípios, padrões, estilos, formato dos entregáveis, tipos de raciocínio, captura de decisões, processos para construir as visões [EMERY-09].

Para estar em conformidade com o padrão, um framework de arquitetura deverá ser construído com base no núcleo da ontologia do padrão, opcionalmente documentando explicitamente a extensão da ontologia como um metamodelo [EMERY-09].

2.9.7 Conformidade com o padrão

Para estar em conformidade com o padrão, um framework deverá possuir [EMERY-09]:

- Conformidade de uma descrição de arquitetura (de acordo com a [IEEE-1471-2000]);
- Conformidade de um framework de arquitetura (de acordo com a [ISO/IEC/IEEE 42010:2011]);
- Conformidade de uma descrição de arquitetura para um framework de arquitetura;

Tornando os frameworks de arquitetura um ponto de conformidade, abre novas possiblidades para interoperabilidade e compartilhamento de conhecimento na comunidade de arquitetura [EMERY-09].

Possuindo uma definição padronizada do framework de arquitetura e os requisitos para especificá-los em conformidade permite compartilhar o conhecimento entre comunidades que anteriormente possuíam diferentes linguagens [EMERY-09].

2.10 Modelo de visões 4+1 de Kruchten

Usualmente as arquiteturas de software têm sido descritas sem abordar as considerações de todos os interessados pelo sistema. Além disso, a concepção de modelos onde são representados mais detalhes do que eles podem expressar de fato, implica em falta de clareza para interpretá-los [KRUCHTEN-95].

Como solução para tratar de questões como ambiguidade introduzida pelos modelos, dificuldade em expressar mais de um estilo arquitetônico para o sistema, decomposição prematura ou excesso de ênfase em um único aspecto, [KRUCHTEN-95] propôs organizar a descrição da arquitetura de software usando diversas visões simultâneas, cada uma abordando um conjunto específico de considerações. A figura 2-3 apresenta de forma macro o modelo de visões 4+1.

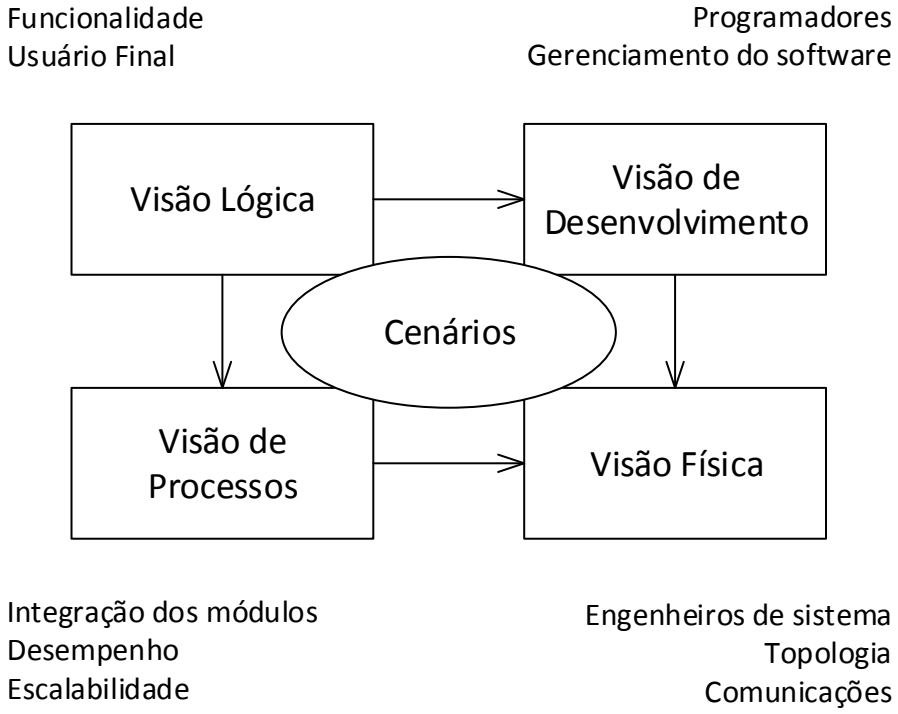


Figura 2-3 Modelo de visões 4+1 [KRUCHTEN-95]

A descrição de uma arquitetura e as decisões tomadas podem ser organizadas ao redor dessas quatro visões e ilustradas por casos de uso ou cenários que estabelecem a quinta visão. Cada visão é descrita por um diagrama que usa sua notação particular. Além disso, para cada visão, o arquiteto pode designar um estilo

arquitetural, permitindo assim a coexistência de múltiplos estilos em um único sistema [KRUCHTEN-95].

2.10.1 Arquitetura Lógica

A arquitetura lógica[12] apoia principalmente os requisitos funcionais. O sistema é decomposto no conjunto de abstrações principais obtidas a partir do domínio do problema na forma de classes ou objetos, onde são explorados os princípios de abstração, encapsulamento e herança [KRUCHTEN-95].

A ênfase do diagrama de classes é voltada para as principais características e operações. Caso seja importante definir os comportamentos internos, tais comportamentos podem ser retratados por diagramas de máquina de estados [KRUCHTEN-95]. Aplicações que são orientadas por dados podem representar, como alternativa à abordagem orientada a objetos, uma visão lógica utilizando diagramas entidade-relacionamento.

Como diretriz principal para a modelagem da visão lógica, é conveniente tentar manter um modelo de objetos simples e coerente considerando todo o sistema, evitando a especialização prematura das classes [KRUCHTEN-95].

2.10.2 Arquitetura de Processos

A visão de processos considera alguns requisitos não funcionais como desempenho e disponibilidade[13]. Ele aborda questões de concorrência, sincronização e distribuição, integridade, tolerância a falhas, e como as principais abstrações da visão lógica se enquadram na arquitetura de processos [KRUCHTEN-95].

A visão de processos pode ser descrita por diversos níveis de abstração, onde cada nível aborda diferentes considerações dos interessados. Em nível macro, a arquitetura de processos pode ser vista como um conjunto de redes de comunicação

---

[12] Nos termos da [ISO/IEC/IEEE 42010:2011], a arquitetura de um sistema é uma concepção holística das propriedades fundamentais do sistema, melhor entendida através de múltiplas visões tal como visão de negócios, visão física e visão técnica.

[13] No contexto do framework proposto, os requisitos não funcionais são abordados como requisitos arquiteturalmente significativos e expressados por meio de cenários de qualidade.

lógicas de programas executando independentemente e distribuídos através de um conjunto de recursos de hardware interligados. Múltiplas redes lógicas podem existir simultaneamente compartilhando os mesmos recursos físicos [KRUCHTEN-95].

Processos são agrupamentos de tarefas que formam uma unidade executável. Eles representam o nível no qual a arquitetura de processos pode ser controlada taticamente. Adicionalmente processos podem ser replicados para aumento de distribuição da carga de processamento ou para a melhoria de disponibilidade [KRUCHTEN-95].

O software é particionado em um conjunto de tarefas independentes. Uma tarefa é uma pilha de controle[14], que pode ser individualmente agendada para execução em um nó de processamento. Pode-se distinguir entre tarefas maiores, que são elementos arquitetônicos que podem ser abordados unicamente e tarefas menores introduzidas localmente por razões de implementação [KRUCHTEN-95].

As maiores tarefas se comunicam por um conjunto de mecanismos inter-processos bem definidos, serviços de comunicação baseado em mensagens síncronas e assíncronas e chamadas de procedimento remoto. As tarefas menores podem se comunicar através de memória compartilhada. As medidas de desempenho, fluxos das mensagens e carga de processos podem ser estimadas com base em um diagrama de processos [KRUCHTEN-95].

2.10.3 Arquitetura de Desenvolvimento

A visão de desenvolvimento centra-se na organização modular do software no ambiente de desenvolvimento. O software é empacotado em blocos pequenos, bibliotecas ou subsistemas que podem ser desenvolvidos por um único desenvolvedor ou por um grupo pequeno de desenvolvedores. Os subsistemas são organizados em uma hierarquia de camadas, onde cada camada provê interfaces bem definidas para as camadas superiores [KRUCHTEN-95].

---

14 Do inglês - thread of control.

A arquitetura de desenvolvimento completa pode ser descrita somente após a identificação de todos os elementos do software. Entretanto, é possível listar as regras que regem a arquitetura de desenvolvimento: particionamento, agrupamento e visibilidade [KRUCHTEN-95].

Na maioria das vezes a arquitetura de desenvolvimento leva em conta requisitos internos relacionados com a facilidade de desenvolvimento, gerenciamento do software, reutilização e as restrições impostas por um conjunto de ferramentas ou linguagem de programação [KRUCHTEN-95].

Deste modo, a visão de desenvolvimento serve como fundamento para alocação de requisitos, alocação de atividades para as equipes ou organizações, estimativas de custos e planejamentos, monitoramento do progresso dos projetos, raciocínio sobre o reuso de software, portabilidade e segurança, sendo a base para o estabelecimento de uma linha de produtos [KRUCHTEN-95].

É recomendável a adoção de um estilo em camadas para a visão de desenvolvimento, definindo quatro ou seis camadas de subsistemas, onde cada camada possui uma responsabilidade bem definida. Para minimizar o desenvolvimento de uma rede complexa de módulos dependentes, aplica-se a regra de modelagem em que um subsistema somente pode depender de outro subsistema que se encontra na mesma camada ou em camadas inferiores [KRUCHTEN-95].

2.10.4 Arquitetura Física

A arquitetura física descreve o mapeamento do software para o hardware e reflete os aspectos de distribuição, considerando principalmente os requisitos não funcionais do sistema tal como disponibilidade, confiabilidade, tolerância a falhas, desempenho (vazão) e escalabilidade [KRUCHTEN-95].

O software executa em uma rede de computadores ou nós de processamento. Os diversos elementos identificados, isto é, redes, processos, tarefas e objetos precisam ser mapeados sobre diversos nós. Espera-se que diferentes tipos de configurações físicas sejam utilizados, algumas para desenvolvimento e teste, outras

para desenvolvimento do sistema para diversos sites ou diferentes clientes. Portanto, o mapeamento do software para os nós precisam ser altamente flexíveis e implicar em impacto mínimo sobre o código fonte [KRUCHTEN-95].

2.10.5 Cenários

Os elementos das quatro visões apresentadas são trabalhados de forma integrada através do uso de um pequeno conjunto de cenários importantes. Os cenários são, de certa forma, uma abstração dos requisitos mais importantes. Sua modelagem é expressa por diagramas de objetos e diagramas de interação. A visão de cenários é redundante com respeito às outras, porém tem como propósitos principais [KRUCHTEN-95]:

- Conduzir a descoberta de elementos arquitetônicos durante a modelagem.
- Validar e ilustrar os elementos quando a modelagem da arquitetura está completa, servindo como ponto de partida para os testes de um protótipo arquitetural.

2.10.6 Correspondência entre as visões

As diversas visões não são ortogonais e independentes. Elementos de uma visão são conectados a elementos de outras visões, de acordo com certas regras e heurísticas [KRUCHTEN-95].

2.10.6.1 Visão lógica para visão de processos

Partindo da lógica para a visão de processos identificam-se importantes características das classes da arquitetura lógica. Embora na visão lógica considera-se cada objeto como ativo e potencialmente concorrente sem prestar atenção no grau exato de concorrência necessária, considerando deste modo apenas os aspectos funcionais dos requisitos, durante a definição da arquitetura de processos, implementar cada objeto em sua própria pilha de controle não é muito prático no estado atual da tecnologia [KRUTCHEN-95].

Por outro lado, são necessárias múltiplas pilhas de controle por diversas razões:

- Reagir rapidamente a certas classes de estímulos externos, inclusive eventos relacionados ao tempo.
- Se beneficiar de múltiplas CPU's em um nó ou múltiplos nós em um sistema distribuído.
- Aprimorar a utilização da CPU, alocando a CPU para outras atividades enquanto uma pilha de controle é suspensa aguardando até que alguma atividade finalize.
- Priorizar atividades e melhorar a capacidade de resposta.
- Suportar a escalabilidade do sistema com processos adicionais compartilhando a carga.
- Separar as considerações entre diferentes áreas do software.
- Alcançar um alto grau de disponibilidade através de cópias de processos.

2.10.6.2 Visão lógica para visão de desenvolvimento

Uma classe é usualmente implementada como um módulo. Classes maiores são decompostas em múltiplos pacotes. Coleções de classes estritamente relacionadas são agrupadas em subsistemas. Restrições adicionais devem ser consideradas para a definição de subsistemas, tal como organização das equipes, magnitude esperada do código como o número de linhas típicas por subsistema, grau esperado para reuso, princípios de camadas rígidas, política de implantação e gerenciamento de configuração. Consequentemente, acabam-se concebendo visões de desenvolvimento que não possuem correspondência de um para um com a visão lógica [KRUCHTEN-95].

2.11 Considerações do Capítulo

Este capítulo apresentou as definições fundamentais de arquitetura de software e o seu âmbito tanto em um contexto profissional quanto em um contexto técnico.

No contexto profissional foi possível perceber que a arquitetura de software, em princípio, tem como meta resolver os conflitos de interesse entre diferentes grupos de interessados pelo sistema, tomando decisões que dizem respeito aos requisitos arquiteturalmente significativos.

É preciso expressar os requisitos arquiteturalmente significativos apropriadamente para diminuir a probabilidade de ocorrerem mudanças de magnitude mais ampla que implicam em custos altos com manutenção.

Foi possível perceber que a arquitetura de software é designada por meio do uso de métodos e técnicas de modelagem com base em classes de problemas existentes e que as diferentes suposições que os interessados fazem sobre o sistema podem conduzir o projeto à incompatibilidade arquitetural inviabilizando seu sucesso.

Esse capítulo também apresentou algumas técnicas e princípios amplamente aceitos na indústria como especificação de cenários, modelagem dirigida por atributos de qualidade, atributos de qualidade, padrões de projeto, interfaces econômicas, visibilidade e espaçamento. Essas técnicas são fundamentais para orientar os arquitetos de software à resolver os problemas inerentes com as arquiteturas.

Finalmente este capítulo apresentou as principais definições de um framework de arquitetura padronizado pela ontologia da [IEEE-1471-2000] e extendido pela [ISO/IEC/IEEE 42010:2011] que introduziu o conceito de regras de correspondência entre modelos.

Adicionalmente foi apresentado o modelo de visões 4+1 de Krutchen para descrição de arquiteturas utilizando múltiplas visões simultâneas tendo como ponto focal as especificações de cenários que contribuem para a avaliação e os testes das arquiteturas propostas. Com base nesse referencial teórico o próximo capítulo apresenta o desenvolvimento do framework de arquitetura.

3. DESENVOLVIMENTO DO FRAMEWORK DE ARQUITETURA

3.1 Considerações iniciais

Este capítulo tem como finalidade detalhar as atividades de definição da arquitetura, contemplando os modelos que devem ser concebidos e as regras de correspondência que se aplicam entre modelos para expressar consistência, dependência e rastreabilidade entre modelos distintos.

3.2 Atividades de definição da arquitetura

Com base no referencial teórico apresentado, é especificado o método para arquitetar[15] sistemas de software. Tal método se inicia pela identificação dos interessados, identificação das considerações associadas com a arquitetura, mapeamento das considerações identificadas para um conjunto de atributos de qualidade que servirão como apoio para as especificações de cenários de qualidade, bem como a aplicação dos pontos de vista no decorrer do estudo de caso.

No que diz respeito à adequação para o negócio [BUSCHMANN-11a], o método apoia o estabelecimento de diálogo contínuo com os interessados no domínio através de atividades de especificação de cenários funcionais e cenários de qualidade desempenhados em iterações curtas.

Com relação ao escopo do sistema, as especificações de requisitos funcionais, brevemente descritas na aplicação do ponto de vista funcional, são refinadas na medida em que a aplicação do ponto de vista de informação contribui para a análise das propriedades essenciais dos objetos de domínio, retratadas diretamente no modelo de domínio.

[15] De acordo com [ISO/IEC/IEEE 42010:2011] arquitetar diz respeito aos processos de concepção, definição, expressão, documentação, comunicação, assegurando uma implementação apropriada, mantendo e aprimorando uma arquitetura ao longo do ciclo de vida de um sistema.

Utilizando modelos de domínio e cenários[16] como uma orientação, a especificação e a clareza do escopo do sistema e dos requisitos serão baseadas em menos suposições e mais em necessidades reais e valor agregado [BUSCHMANN-11a].

As atividades restringem a estrutura subjacente do framework para contemplar os pontos de vista funcional, de informação e de concorrência designados através dos modelos propostos e das regras de correspondência especificadas para governar a correspondência expressando de forma explícita as relações de dependência que se aplicam entre modelos distintos.

Embora a restrição dos pontos de vista estabelecida pelo método proposto seja contraditória com a definição de bibliotecas de pontos de vista da [ISO/IEC/IEEE 42010:2011] que tem como propósito estabelecer um conjunto de pontos de vista reutilizáveis, tal restrição de pontos de vista propõe uma base inicial para descrever arquiteturas, podendo ser continuamente evoluído através da incorporação de pontos de vista específicos para o domínio em questão.

Ao final da atividade de identificação dos interessados, o mapeamento das considerações identificadas com os atributos de qualidade inerentes a estas pode ser conduzido com o auxílio da ISO-25010 (2011) que padroniza os atributos de qualidades comumente associados aos produtos de software.

Juntos, o método de definição de arquiteturas, os modelos e as regras de correspondência entre os modelos estabelecem a base fundamental para a especificação dos cenários funcionais, cenários de qualidade e modelagem candidata. A figura 3-1 resume essas atividades de forma macro.

16 O framework proposto utiliza como apoio descrições de casos de uso para especificação de requisitos funcionais e o método de especificação de cenários de qualidade de [BASS-12] para especificação de requisitos não funcionais expressados em cenários de qualidade.

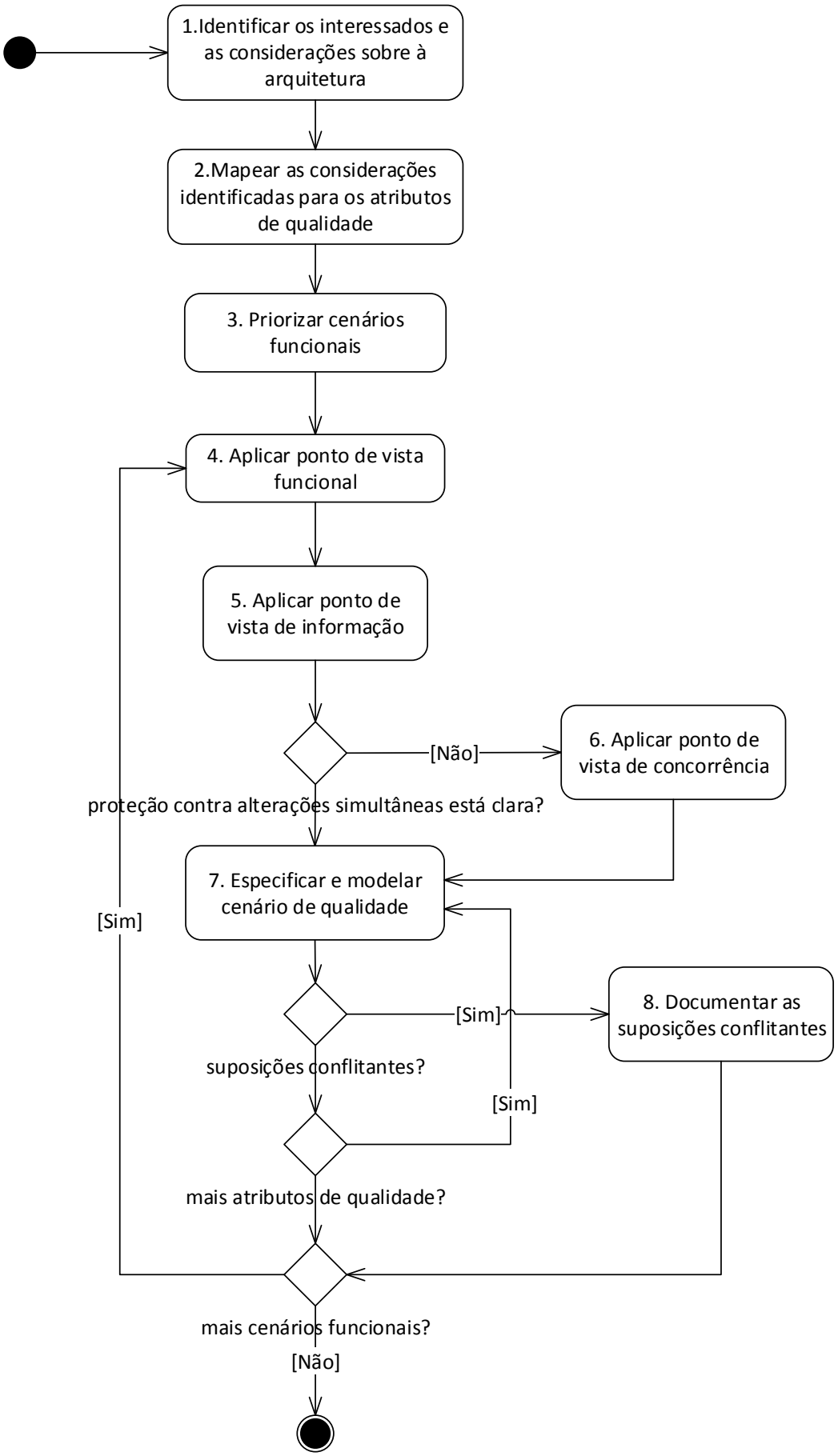


Figura 3-1 Atividades propostas para definição de arquiteturas de software.

3.2.1 Identificar os interessados e as considerações sobre a arquitetura

Uma das formas de se obter as informações sobre os interessados e as considerações relacionadas à arquitetura é realizar entrevistas com os interessados pela arquitetura. Tal atividade é opcional, caso o desenvolvedor do framework tenha conhecimento sobre os principais envolvidos e as considerações relacionadas com a arquitetura. A tabela 3-1 apresenta o modelo proposto, VREQ-IARQ, para ser preenchido durante a atividade de identificação dos interessados e suas considerações.

Tabela 3-1 Modelo para mapear os interessados pela arquitetura

Papel	Responsabilidade	Principais considerações
Cód. modelo	VREQ-IARQ	
[Nome do papel]	*[Descrever sucintamente as responsabilidades do interessado]*	*[Listar as principais considerações do interessado associadas ao sistema]*

Os interessados e as considerações são os motivadores para a identificação e o mapeamento apropriado dos atributos de qualidade que, no contexto do framework proposto, são fundamentais para as especificações e modelagem de cenários de qualidade.

3.2.2 Mapear as considerações identificadas para os atributos de qualidade

Essa atividade tem como entrada principal o modelo VREQ-IARQ utilizado para identificar as considerações dos principais interessados pelo sistema tendo como propósito identificar um conjunto de atributos de qualidade que abordam as principais considerações sobre a qualidade do sistema como um todo.

Embora não exista uma propriedade que indique se um requisito é arquiteturalmente significativo ou não [LIANGING-13], durante essa atividade os interessados são entrevistados para levantar questões que implicam em custos maiores para mudança, questões que não são negociáveis para o sucesso do sistema e questões difíceis de resolver.

Após o levantamento das questões, o mapeamento adequado para os atributos de qualidade é realizado com o apoio da [ISO-25010:2011]. A tabela 3-2 apresenta o modelo de documento proposto que deve ser preenchido após o mapeamento das questões significativas para os atributos de qualidade.

Tabela 3-2 Modelo para mapeamento dos atributos de qualidade

Interessado	Atributos de qualidade
Cód. modelo	VREQ-ATTR
[Nome do interessado]	*[Listar os nomes dos atributos de qualidade referenciados pelas principais considerações dos interessados para apoiar o responsável pela especificação de requisitos arquiteturalmente significativos a formular questões que o ajude a identificá-los.]*

O mapeamento apropriado das principais considerações para os atributos de qualidade é fundamental para apoiar a identificação dos requisitos arquiteturalmente significativos.

3.2.3 Priorizar cenários funcionais

A priorização de cenários funcionais tem como meta designar sinteticamente um conjunto inicial de cenários funcionais para especificação de requisitos funcionais. A definição dos critérios de priorização de cenários é de responsabilidade dos interessados pelo sistema. A tabela 3-3 apresenta o modelo de documento proposto para ser preenchido durante a atividade de priorização de cenários funcionais.

Tabela 3-3 Modelo para justificar a priorização de cenários funcionais

Prioridade	Cenário funcional	Justificativa
[Código de priorização]	*[Nome do cenário]*	*[Descrever as principais motivações e razões pelas quais o cenário funcional em questão deve ser especificado e modelado na ordem definida]*

Após priorização dos cenários funcionais, inicia-se o projeto arquitetural pela aplicação do ponto de vista funcional que abrange o escopo inicial através da descrição dos principais eventos entre o sistema e seus atores.

3.2.4 Aplicar ponto de vista funcional

Essa atividade tem como finalidade especificar os requisitos funcionais do sistema, tendo como foco os fluxos essenciais entre o sistema e seus atores, servindo de base para as atividades subsequentes. Para descrever requisitos funcionais é adotada como padrão a descrição de casos de uso[17].

De acordo com a priorização de cenários realizada anteriormente, os principais eventos do cenário principal de sucesso são descritos. A tabela 3-4 apresenta o modelo de documento proposto.

Tabela 3-4 Modelo proposto para descrição de casos de uso

Elemento	Descrição
Cód. modelo	VREQ_FUN
Id. Caso de Uso	*[Identificador único do caso de uso. Pode ser numérico ou não.]*
Descrição	*[Uma descrição sucinta do caso de uso.]*
Interessados	*[Identificar os interessados pelo cenário funcional, considerando que um ou mais desses interessados servirão como ponto focal para entrevista durante aplicação do ponto de vista de informação, de concorrência e especificação de cenários de qualidade].*
Evento Iniciador	*[Identificar o evento que inicia o caso de uso.]*
Atores	*[Identificar os atores participantes do caso de uso: pessoas ou outros sistemas.]*
Pré-condição	*[Os estados relevantes em que o sistema deverá estar antes da execução do caso de uso.]*
Sequência de eventos	*[Descrever as interações entre os atores e o sistema na forma de uma sequencia de passos do cenário principal do caso de uso.]*
Pós-Condição	*[Os estados relevantes que o sistema deverá estar após a execução do caso de uso.]*

A modelagem candidata proveniente do cenário funcional deve ser concebida em um modelo de classes da UML [FOWLER-05]. A concepção do modelo de classes é

[17] A especificação de requisitos funcionais por meio de casos de uso é apresentada em detalhes por [LARMAN-07].

realizada através da identificação dos principais substantivos e adjetivos descritos na sequência de eventos do caso de uso.

3.2.5 Aplicar ponto de vista de informação

Embora diversos conceitos sobre o domínio sejam derivados durante a aplicação do ponto de vista funcional, usualmente, não é factível tornar visível os conceitos importantes para o negócio em um modelo de domínio utilizando como subsídio apenas o cenário principal de um caso de uso.

A proposta do ponto de vista de informação é contribuir para a assimilação do conhecimento iniciando pela entrevista com cada um dos interessados declarados pelo cenário funcional correspondente. Essa decisão tem como objetivo dar enfoque nas questões sobre o sistema que, a princípio, não são de conhecimento de um único interessado no domínio.

Para alcançar tal objetivo, a aplicação do ponto de vista de informação busca enriquecer a modelagem do sistema materializando conceitos importantes, de modo que se tornem expressíveis no modelo, contribuindo para redução de margens para interpretações ambíguas sobre as questões significativas para o negócio.

Além de tornar visível os conceitos no modelo, o ponto de vista de informação aplica o espaçamento adequado entre os conceitos com pouca ou nenhuma proximidade semântica contribuindo para uma modularização versátil que promove alterações com impacto reduzido e praticidade para evolução do negócio com qualidade e produtividade.

Para retratar os conceitos relevantes e projetar os fluxos de trabalho diretamente no modelo utiliza-se como apoio essencial os padrões de projeto propostos para o framework. A tabela 3-5 apresenta o modelo proposto para a modelagem do ponto de vista de informação.

Tabela 3-5 Modelo proposto para o ponto de vista de informação

Elemento	Declaração
Cód. Modelo	VINF_MOD
Ponto de vista	Informação
Questão	*[Questão que não pode ser eliminada do sistema, mas pode ser abordada de modo à reduzir o impacto sobre o sistema, a magnitude das mudanças e consequentemente os custos com manutenção.]*
Motivação	*[Identificar a principal motivação para a modelagem em termos de redução de impacto e de custos com manutenção.]*
Correspondência	*[Descrever de que modo este modelo corresponde com um cenário funcional VREQ_FUN, designando uma composição, refinamento, rastreabilidade ou restrição entre os objetos dos modelos.]*
[Apresentar um esboço de modelo de domínio através de um diagrama de classes da UML. Em princípio, utiliza-se como apoio os padrões 'Agregado' para contribuir com o espaçamento entre conceitos distintos e o padrão 'Objeto de valor' para materializar os conceitos simples do domínio contribuindo para visibilidade. Tais padrões são identificados por meio dos estereótipos <<raíz agregado>> e <<objeto valor>>. Adicionalmente o espaçamento pode se tornar mais claro através da representação de um conjunto de classes rigorosamente associadas em um diagrama de pacotes da UML. Além disso, o projetista é livre para incorporar qualquer padrão de projeto que contribua para a resolução eficaz da questão.]	

O espaçamento é obtido com o apoio do padrão Agregado de [EVANS-03], para representar de forma abstrata os relacionamentos entre objetos que são alterados como uma única unidade conceitual, normalmente modelados como uma composição do tipo todo-parte. Em outras palavras, o padrão agregado contribuirá para o isolamento[18] de classes que fazem sentido no sistema somente quando manipuladas em conjunto, designando desta forma o espaçamento mais apropriado com agregados conceituais distinguíveis e visíveis.

Com a aplicação do ponto de vista de informação busca-se aprimorar a qualidade da informação, estruturando–as apropriadamente para reduzir o impacto de modificações e transmitir mais segurança para os responsáveis pela manutenção do

[18] Na prática, não é possível isolar completamente os objetos, pois eles precisam colaborar entre si, cada um representando o seu papel durante a execução dos fluxos de trabalho.

sistema através de um modelo de domínio auto-documentado, com a materialização de módulos coesos que encapsulam objetos semanticamente associados.

3.2.6 Aplicar ponto de vista de concorrência

A atribuição dos elementos funcionais para pilhas de controle ou processos tem como propósito conceber um modelo de concorrência que oriente os desenvolvedores a proteger o estado interno de objetos compartilhados entre tarefas que precisam alterar o estado do objeto simultaneamente. A tabela 3-6 apresenta o modelo de documento proposto para aplicar ao ponto de vista de concorrência.

Tabela 3-6 Modelo proposto para o ponto de vista de concorrência

Elemento	Declaração
Cód. modelo	VCON_PASS
Finalidade	Esse modelo tem como finalidade retratar os estados e as transições entre estados em um objeto compartilhado, destinando-se a abordar como a chegada de eventos paralelos, será coordenada e controlada para garantir que o estado interno do objeto não seja corrompido. Além disso, retratar de que modo o objeto compartilhado será protegido contra tais alterações simultâneas.
Ponto de vista	Concorrência
Interessados	*[Informar quem são os principais interessados em garantir a consistência da informação pertinente ao objeto.]*
Objeto	*[Informar o nome do objeto compartilhado para análise das seções críticas.]*
[Elaborar diagrama de máquina de estados retratando as condições estáveis em um objeto compartilhado e os possíveis eventos responsáveis por disparar as transições entre estados.]	
[Elaborar diagrama de classes retratando pilhas de controle, processos, o objeto compartilhado e os mecanismos para protegê-lo.]	

Na prática qualquer elemento funcional é atribuído a uma pilha de controle ou processo. No entanto, o framework proposto considera que objetos que sofrem pouca ou nenhuma alteração simultânea não precisam ser analisados e modelados no ponto de vista de concorrência. Portanto, o ponto de vista de concorrência é

aplicado somente em circunstâncias onde a proteção contra alterações simultâneas em um objeto compartilhado não está claramente definida.

3.2.7 Especificar cenários de atributos de qualidade

Tendo como principais entradas os modelos VREQ_FUN que expressa os requisitos funcionais associados ao cenário de qualidade e VREQ-ATTR que lista os principais atributos de qualidade relacionados aos principais interesses pelo sistema, essa atividade tem como finalidade investigar requisitos arquiteturalmente significativos, negligenciados ou ocultos que sejam significantes no contexto do cenário funcional em questão, ou seja, requisitos que implicam em custos maiores para mudança, caso não sejam resolvidos de forma apropriada.

Para cada cenário funcional especificado, podem existir um ou mais cenários expressando requisitos arquiteturalmente significativos. A rastreabilidade entre o cenário de qualidade e o cenário funcional é obtida pela devida identificação do cenário funcional ao qual se aplica o cenário de qualidade. A tabela 3-7 mostra o modelo de documento proposto, VREQ_QUA, que dever ser utilizado para elicitar cenários de qualidade.

Tabela 3-7 Modelo proposto para especificação de cenários de qualidade

Elemento	Declaração
Cód. modelo	VREQ_QUA
Cenário funcional	*[Identificar o cenário funcional ao qual se aplica o cenário de qualidade sendo descrito.]*
Interessados	*[Listar os grupos dos interessados pelo cenário.]*
Atributo de qualidade	*[Declaração do atributo de qualidade associado ao cenário.]*
Estímulo	*[Descreve a chegada de um evento no sistema ou ação que motivou o estímulo.]*
Ambiente (Opcional)	*[Qualificação do estímulo, destacando as circunstâncias em que o cenário ocorre.]*
Resposta (Opcional)	*[Responsabilidades executadas para responder ao estímulo.]*
Medida de resposta	*[Medida que tem como finalidade validar se o requisito arquiteturalmente significativo foi resolvido.]*

Os campos [Ambiente], [Resposta] e [Medida de resposta] são definidos pelo método de elicitação de cenários de qualidade de BASS (2012).

O cenário funcional relaciona o caso de uso correspondente com o cenário de qualidade. Tal rastreabilidade é fundamental para orientar o projetista a modelar o cenário de qualidade sem desacomodar as funcionalidades atribuídas para as classes derivadas do cenário funcional.

Conforme apresentado na seção 2.5.5 os usuários frequentemente nada mencionam sobre requisitos arquiteturalmente significativos, no entanto, a identificação apropriada dos interessados pelo cenário auxilia nas especificações das medidas de resposta servindo de base para avaliação de modelagens candidatas.

A identificação do atributo de qualidade associado ao cenário bem como as especificações da chegada do evento ao sistema e do ambiente fornecem subsídios para a contextualização do ambiente e a identificação apropriada de padrões de projeto ou táticas que apoiam a modelagem do cenário.

O raciocínio guiado por atributos de qualidade contribui para o enfoque no que é mais importante na perspectiva do problema em particular, orientando o projetista a formular o questionamento apropriado para identificação das tarefas relevantes e os fluxos de trabalho mais importantes para os interessados pelo cenário.

3.2.8 Documentar as suposições conflitantes

As diferentes suposições que os interessados fazem sobre o sistema e levam a questões de arquitetura que implicam significativamente na escolha entre diferentes soluções devem ser documentadas em um modelo de decisões de arquitetura, conforme mostra a tabela 3-8.

Tabela 3-8 Modelo para documentar as decisões de arquitetura

Modelo para documentar as decisões de arquitetura			
Cód. modelo	DARQ-01		
Questão	*[A questão que levou o arquiteto ao raciocínio no que diz respeito à magnitude que uma decisão arquitetônica pode causar, em termos de custos de mudança.]*		
Interessado 1	*[Identificar os interessados na questão. Adicionar uma linha para cada grupo de interessados]*	**Suposições 1**	*[Descrever as suposições ou premissas que, se forem verdadeiras poderão influenciar positivamente ou negativamente a decisão tomada].*
Proposta	*[A proposta para resolver a questão, podendo incluir esboços de modelagem candidata.]*		
Decisão e justificativa	*[A decisão tomada pelo arquiteto para resolver a questão.]*		

3.3 Regras de correspondência entre os modelos

Essa seção destina-se a especificar as regras que governam as correspondências entre os modelos concebidos tal como coerência, rastreabilidade, composição, consistência e completeza, tornando esses relacionamentos mandatórios para que um documento de descrição de arquitetura concebido com a aplicação do framework proposto possa ser avaliado com respeito à conformidade com tais correspondências.

3.3.1 Correspondência entre os modelos VREQ_FUN e VREQ-ATTR

Considerando que, no contexto do framework proposto, os atributos de qualidade orientam o projetista durante as especificações de requisitos arquiteturalmente significativos, os interessados pelo cenário funcional precisam ser rastreáveis até o modelo VREQ-ATTR que mapeia os interessados com os atributos de qualidade.

Tal correspondência contribui para a identificação apropriada dos atributos de qualidade e dos interessados que podem ser entrevistados durante o levantamento

de requisitos arquiteturalmente significativos. A tabela 3-9 governa a correspondência entre os interessados pelo cenário VREQ_FUN e os interessados declarados em um modelo VREQ-ATTR.

Tabela 3-9 Correspondência entre os modelos REQ_FUN e VREQ-ATTR

Elemento	Declaração
Cód. Regra	Reg_fun_attr
Finalidade	Essa regra de correspondência tem como finalidade governar a correspondência entre os interessados declarados em um cenário funcional e os interessados declarados em um modelo VREQ-ATTR que tem como propósito realizar o mapeamento das considerações desses interessados para os atributos de qualidade.
Descrição	Os interessados declarados em um cenário funcional precisam ser rastreáveis até um modelo VREQ-ATTR, que lista os nomes dos atributos de qualidade identificados para abordar as considerações dos interessados.

Tal rastreabilidade é fundamental para auxiliar o arquiteto a identificar os interessados e questioná-los sobre requisitos arquiteturalmente significativos durante as atividades de especificação de cenários de qualidade. A violação da regra de correspondência pode implicar em mudanças de magnitude ampla com a negligência de requisitos estrategicamente importantes para o negócio.

3.3.2 Correspondência entre os modelos VREQ_FUN e VCON_PASS

Embora todos os objetos sejam designados para pilhas de controle durante a execução do sistema, o mapeamento de tais objetos para as pilhas de controle, assim como a modelagem detalhada do ponto de vista de concorrência não é mandatória pelo framework proposto. No entanto, quando existe uma preocupação maior com relação aos objetos que sofrem alterações simultâneas, a concorrência e a sincronização precisam ser apropriadamente abordadas e solucionadas. Para estar em conformidade com o framework proposto tal correspodência não pode ser violada. A tabela 3-10 declara tal regra de correspondência.

Tabela 3-10 Correspondência entre um modelo de domínio e VCON_PASS

Elemento	Declaração
Cód. Regra	Reg_fun_con
Finalidade	Essa regra de correspondência tem como finalidade governar as correspondências entre os elementos funcionais compartilhados entre múltiplas pilhas de controle e a modelagem que designa o método de coordenação e controle das alterações em tal objeto compartilhado.
Descrição	Ao aplicar o ponto de vista de concorrência, o objeto compartilhado sendo representado deverá corresponder com um objeto de domínio derivado de um cenário VREQ_FUN, no qual existe uma preocupação com as seções críticas do objeto.

Conforme especificação da regra, a correspondência é válida somente quando o ponto de vista de concorrência é aplicado.

3.3.3 Correspondência entre a derivação do modelo VREQ_FUN e VINF_MOD

Para evitar as armadilhas inerentes aos sistemas de informação complexos, onde a falta de especialização no domínio pode desencadear uma série de problemas, desde o aumento nos custos com manutenação até o corrompimento da modelagem proposta, o ponto de vista de informação dedica-se a extrair a terminologia essencial do processo de negócio associado ao cenário funcional para adicionar ao modelo de domínio, tornando-o mais expressivo e valioso. Assim, o esboço de modelagem derivado do ponto de vista de informação deve refinar o esboço de modelagem derivado do ponto de vista funcional. A tabela 3-11 apresenta a declaração da regra de correspondência.

Tabela 3-11 Correspondência entre um modelo de domínio e VINF_MOD

Elemento	Declaração
Cód. Regra	Reg_dminf
Finalidade	Essa regra de correspondência tem como finalidade governar as correspondências como refinamento e rastreabilidade entre um modelo de domínio derivado de um cenário funcional VREQ_FUN e um modelo de domínio derivado de um modelo VINF_MOD no ponto de vista de informação.

Descrição	Para estar em conformidade com o framework proposto o ponto de vista de informação, representado por um modelo VINF_MOD, precisar refinar um modelo de classes derivado do cenário funcional VREQ_FUN correspondente.

Essa regra de correspondência formaliza a relação de dependência existente entre o ponto de vista de informação e o ponto de vista funcional. Contudo, o próprio modelo VINF_MOD contempla o campo [Correspondência] para que a rastreabilidade entre os modelos seja claramente definida.

3.3.4 Correspondência entre os modelos VREQ_QUA e VREQ-ATTR

Um dos modelos utilizados como subsídio para as especificações de cenários de qualidade são os modelos VREQ-ATTR utilizados para mapear as considerações dos interessados para os atributos de qualidade. Assim, para que um modelo VREQ_QUA esteja em conformidade com o framework proposto ele deve corresponder com pelo menos um dos atributos de qualidade listados em um dos modelos VREQ-ATTR. A tabela 3-12 declara a regra de correspondência.

Tabela 3-12 Correspondência entre os modelos VREQ_QUA e VREQ-ATTR

Elemento	Declaração
Cód. Regra	Reg_Atrib
Finalidade	Essa regra de correspondencia tem como finalidade governar a correspondência entre um modelo VREQ_QUA, representando um cenário de qualidade e um modelo VREQ-ATTR que declara o atributo de qualidade correspondente.
Descrição	Cada cenário de qualidade especificado deve identificar um atributo de qualidade mapeado e declarado por pelo menos um modelo do tipo VREQ-ATTR.

Caso seja identificada uma preocupação relevante para o negócio que implique na especificação de um cenário de qualidade, no qual o atributo de qualidade correspondente não foi definido em um modelo VREQ-ATTR, tal atributo de qualidade deve ser apropriadamente documentado por um modelo VREQ-ATTR,

para manter a consistência entre os modelos e evitar a violação da regra de correspondência.

3.3.5 Correspondência entre VREQ_FUN e cenários de interoperabilidade

Um projeto de sistemas corporativo, tipicamente, possui diversas questões importantes que não são analisadas durante a modelagem do ponto de vista funcional. Por exemplo, ao derivar um conjunto de classes conceituais do domínio de um cenário que possui como ator um sistema externo, normalmente não se desvia o foco dos eventos funcionais para abordar os complexidades particulares de integração com o sistema externo.

Questões como a troca de informações com aplicações externas requerem suporte adicional de infraestrutura técnica e tratamento de considerações inerentes à esse tipo de comunicação, como a redução do número de mensagens ida e volta devido à alta latência de rede, custo para gerenciamento de conexões, confiabilidade das mensagens transmitidas por uma rede não confiável, confiabilidade na integração de transações distribuídas, confidencialidade e integridade das mensagens. Tais considerações podem ser abordadas pelas especificações de cenários de interoperabilidade. A tabela 3-13 define a regra de correspondência entre cenários funcionais e cenários de interoperabilidade[19].

Tabela 3-13 Correspondência entre os modelos VREQ_FUN e VREQ_QUA

Elemento	Declaração
Cód. Regra	Reg_Fint
Finalidade	Essa regra de correspondencia tem como finalidade governar a rastreabilidade entre um cenário funcional e um cenário de interoperabilidade, contribuindo com a análise e documentação das questões inerentes à troca de mensagens com aplicações externas de modo à evidenciar a necessidade de conceber estruturas não associadas diretamente ao negócio para comportar requisitos de qualidade como desempenho, confiabilidade, tolerância a falhas,

[19] Interoperabilidade sintática é o grau no qual dois ou mais sistemas podem trocar dados significativos através de interfaces em um contexto particular e a interoperabilidade semântica diz respeito à habilidade de interpretar esses dados sendo trocados [BASS-12].

	confidencialidade e integridade das mensagens.
Descrição	Os modelos de casos de uso, VREQ_FUN, que declararem atores representando a troca de informações com outros sistemas, devem possuir, ao menos, um cenário de interoperabilidade que aborde os requisitos arquiteturalmente significativos para integração entre os sistemas, com o propósito de evitar a negligência sobre as considerações inerentes a esse tipo de comunicação.

Para evitar a violação de tal regra de correspondência, durante as especificações de cenários funcionais é importante identificar apropriadamente os atores que representam sistemas fora da fronteira com o sistema em questão implicando na necessidade de integração.

3.3.6 Correspondência entre VCON_PASS e cenários de escalabilidade

A modelagem do ponto de vista de concorrência pode levar à introdução de pontos de alta contenção que limitam a capacidade do sistema de escalar de forma apropriada. Para abordar essa questão, o tratamento apropriado dos pontos de alta contenção é governado pela regra de correspondência Reg_Pcont, que estabelece a rastreabilidade entre um modelo VCON_PASS e um ou mais cenários de escalabilidade. A tabela 3-14 apresenta a declaração da regra que governa essa relação.

Tabela 3-14 Correspondência VCON_PASS e cenários de escalabilidade

Elemento	Declaração
Cód. Regra	Reg_Pcont
Finalidade	Essa regra de correspondência tem como finalidade governar a dependência entre um modelo de concorrência VCON_PASS e um ou mais modelos VREQ_QUA que especifiquem cenários de escalabilidade apropriados com o objetivo de reduzir pontos de alta contenção.
Descrição	Para garantir a consistência durante alterações simultâneas em um objeto compartilhado, um modelo de concorrência VCON_PASS pode introduzir bloqueios, onde um ou mais pilhas de controle aguardam a liberação do bloqueio para alterar o estado do objeto, aumentando a latência e reduzindo a vazão das transações, na medida em que o

	número de acessos simultâneos aumenta. Sendo assim, uma pilha de controle que bloqueia um objeto compartilhado deve desempenhar seu trabalho o mais rápido possível e liberar o bloqueio do objeto compartilhado para que possa ser alterado por outra pilha de controle.

De certo modo a especificação dessa regra implica na incorporação do atributo de qualidade 'Escalabilidade' ao framework proposto. Contudo a intenção principal da regra de correspondência é reduzir eventuais suposições[20] sobre a queda de desempenho no sistema à medida que o número de acessos simultâneos aumenta.

3.4 Padrões de projeto para apoio à modelagem

Conforme apresentado no tópico 2.8, o método de modelagem dirigida por atributos de qualidade utiliza como apoio um conjunto de hipóteses, sendo tais hipóteses provenientes de sistemas existentes, frameworks, padrões, táticas, decomposição do domínio ou verificações.

No contexto do framework proposto essas hipóteses são consideradas como soluções comprovadas para problemas em contextos semelhantes e correspondem com um grupo de padrões de projeto constituido por padrões de [GAMMA-94], [BUSCHMANN-96], [SCHMIDT-00], [HOHPE-03], [FOWLER-03] e os padrões táticos da modelagem dirigida por domínios de [EVANS-03]. A tabela 3-15 apresenta, em resumo, a definição dos padrões.

Tabela 3-15 Padrões de projeto de apoio para o estudo de caso

Padrão	Definição
Agregado	Agrupamento de objetos associados que são tratados como uma unidade para fins de alterações nos dados. Referências externas são restritas a um único membro do Agregado, designado como raiz. Um conjunto de regras de consistência se aplica dentro dos limites do Agregado [EVANS-03].
Bloqueio de alta	Bloqueia um conjunto relacionado de objetos que podem

[20] Tipicamente, a primeira impressão que os interessados têm sobre a queda de desempenho no sistema está associada à falta de poder computacional do hardware. Contudo, a redução do desempenho pode ser proveniente de, além de outras coisas, estratégias de bloqueio mal elaboradas.

granularidade	frequentemente ser editados como um grupo, com uma simples trava [FOWLER-03].
Bloqueio otimista	O padrão bloqueio otimista (do inglês, optimistic offline lock) previne conflitos entre transações concorrentes através da descoberta de um conflito e reversão da transação [FOWLER-03].
Entidade	Objeto fundamentalmente definido não por seus atributos, mas por uma cadeia de continuidade e identidade [EVANS-03].
Entrega garantida	O padrão entrega garantida (do inglês, guaranteed delivery) torna as mensagens persistentes para que elas não sejam perdidas mesmo se o sistema de mensagens falhar [HOHPE-03].
Estratégia	Define uma família de algoritmos, encapsula cada um e os torna intercambiáveis. O padrão Estratégia deixa o algoritmo variar independentemente dos clientes que o utilizam [GAMMA-94].
Fábrica	Mecanismo para encapsular a complexa lógica de criação e abstrair o tipo de objeto criado em função de um cliente [EVANS-03].
Fábrica Abstrata	A fábrica abstrata fornece uma interface para criar famílias de objetos dependentes ou relacionados, sem especificar classes concretas [GAMMA-94].
Interceptador	O padrão arquitetural interceptador permite que funcionalidade seja adicionada à um framework de aplicação de forma transparente. Essa funcionalidade é invocada automaticamente quando eventos internos do framework ocorrem [SCHMIDT-00].
Mediador	O padrão arquitetural mediador (do inglês, broker) pode ser usado para estruturar sistemas distribuídos com componentes desacoplados que interagem através de invocações de métodos remotos, sendo responsável por coordenar a comunicação, tal como redirecionar as solicitações, bem como para transmitir resultados e exceções [BUSCHMANN-96].
Objeto valor	Objeto que descreve alguma característica ou atributo, não traz consigo nenhum conceito de identidade, não é rastreável e tipicamente é imutável [EVANS-03].
Plugin	Essencialmente o padrão plugin é uma variação do padrão estratégia, com a introdução de uma classe responsável por criar o plugin, que vincula classes em tempo de configuração ao invés de em tempo de compilação [FOWLER-03].
Repositório	Mecanismo para encapsular armazenamento, recuperação e

	comportamento de busca que imita uma coleção de objetos [EVANS-03].
Script de transação	Scripts de transação (do inglês, transaction script) organizam a lógica do negócio por procedimentos onde cada procedimento manipula uma solicitação da apresentação [FOWLER-03]

A adoção aos padrões listados na tabela 3-15 tem como propósito contribuir com a redução de uma parte da variabilidade que leva à incompatibilidade arquitetural [GARLAN-09] através da segregação dos conceitos que variam com mais frequência, de conceitos que variam com menos frequência.

A assimilação do conhecimento, a materialização dos conceitos simples do domínio e o enriquecimento do modelo de domínio com informação sobre o negócio, são exercidos pelo apoio fundamental dos padrões táticos do DDD de [EVANS-03].

3.5 Relação com o modelo de visões 4 + 1

No que diz respeito à qualidade dos sistemas projetados, o framework proposto habilita a descoberta dos atributos de qualidade por projeto realizado. Essa abordagem difere um pouco do modelo 4 + 1 onde os pontos de vista, de certa forma, antecipadamente designam os atributos de qualidade para abordagem por um ou mais pontos de vista. Conforme apresentado na parte conceitual, no modelo 4 + 1, desempenho, disponibilidade, integridade e tolerância a falhas são tratados pela visão de processos. Assim como disponibilidade, confiabilidade, tolerância a falhas, desempenho e escalabilidade são abordados na visão física. Fazendo uma analogia com a orientação a objetos, as visões propostas pelo modelo 4 + 1 parecem estar sobrecarregadas de responsabilidades.

O catálogo de [ROZANSKI-11], com base no modelo 4 + 1, apresenta uma abordagem onde os atributos de qualidade são desconectados dos pontos de vista. Para externalizar tais considerações arquiteturalmente significativas, conjuntos de atividades e boas práticas são definidos e catalogados como perspectivas. Ao contrário dos pontos de vista, as perspectivas não geram modelos diretamente, elas se relacionam com um ou mais pontos de vista no sentido de que qualquer modelo de qualquer visão é sujeito à incorporação de novos elementos ou modificações nos

elementos existentes para responder apropriadamente aos requisitos arquiteturalmente significativos especificados para atender às diferentes considerações que os interessados fazem sobre o sistema.

3.6 Considerações do capítulo

Este capítulo apresentou diretrizes para definição de um framework de arquitetura em um domínio específico, detalhando as atividades que devem ser realizadas, reunindo o conhecimento fundamental para que um documento de descrição de arquitetura possa ser consistentemente concebido.

A padronização do projeto arquitetural através da definição de modelos e regras de correspondência entre modelos designados através de um conjunto de atividades estabelece uma estrutura de base que compreende os pontos de vista funcional, de informação e de concorrência.

Embora o framework proposto não estabeleça os atributos de qualidade para atribuição durante as especificações de cenários de qualidade, a conformidade com as regras de correspondência [Reg_Fint] e [Reg_Pcont] implicam em designar respectivamente os atributos de Interoperabilidade e Escalabilidade.

Justifica-se a adição dessas restrições a contribuição para reduzir as margens para interpretações errôneas com respeito à integração entre sistemas e com respeito à queda de desempenho devido à introdução de pontos de alta contenção que podem inibir as capacidades do sistema de escalar nas medidas estimadas.

4. ESTUDO DE CASO

Este capítulo tem como finalidade experimentar o framework em um domínio específico de acordo com as diretrizes propostas, identificando os interessados pelos sistemas, as considerações associadas à arquitetura, realizando o mapeamento das considerações para os pontos de vista e atributos de qualidade que orientem a especificação e modelagem de cenários. A seção 4.1 inicia-se com a identificação dos interessados em um domínio de recuperação de ativos financeiros. A partir da seção 4.2 o projeto arquitetural é apresentado em detalhes.

4.1 Identificação dos interessados e considerações relacionadas à arquitetura

Conforme o método proposto, a atividade inicial tem como meta levantar quem são os principais interessados pela arquitetura e quais são as suas principais considerações sobre o sistema. A tabela 4-1 resume os principais interessados e suas considerações no domínio da empresa de recuperação de ativos financeiros.

Tabela 4-1 Resumo das considerações relacionadas ao sistema

Papel	Responsabilidade	Principais considerações
Administrador de banco de dados	Responsável pelo monitoramento e controle das operações realizadas no banco de dados.	Assegurar que as permissões adequadas sejam concedidas ou revogadas para cada perfil de usuário acessando o sistema. Assegurar a integridade e a consistência da informação. Identificar consultas que precisam ser otimizadas, monitorar e dar suporte aos desenvolvedores para reduzir os pontos de alta contenção nas tabelas do departamento de cobrança.
Administrador de Redes	Responsável pelo ambiente operacional do sistema e controle de backups.	Manter senhas de acesso aos sistemas, implantar os artefatos nos nós, monitorar o ambiente de execução, aplicar regras de roteamento, fazer a instrumentação dos ambientes, monitorar o tráfego pela rede, instalar e configurar sistemas operacionais, identificar possíveis invasores, recuperar os dados em caso de perda de informações.
Analista Desenvolvedor	Responsável pela análise, projeto e codificação do sistema.	Analisar e refinar a modelagem do projeto arquitetual, assegurar que o projeto detalhado está aderente com o projeto arquitetural, identificar pontos onde existem variações e encapsulá-las, evitar o espalhamento de código, evitar atribuir mais de uma responsabilidade para uma única

		classe, enviar para o repositório de versões os artefatos elaborados, assegurar que o código segue as boas práticas recomendadas pelo arquiteto.
Arquiteto de Software	Responsável pelo projeto arquitetural.	Negocia e prioriza requisitos conflitantes com as áreas usuárias e os clientes. Impoê restrições de modelagem no projeto para reduzir suposições conflitantes existentes entre os diferentes tipos de interessados. Especifica, analisa, modela e avalia cenários de qualidade e arquiteturas candidatas. Realiza prova de conceitos, entrevista os interessados com respeito aos requisitos arquiteturalmente significativos. Certifica-se de que o sistema está sendo desenvolvido conforme as decisões de arquitetura.
Auxiliar de cobrança	Operador da cobrança.	Realiza a cobrança de devedores inadimplentes, localiza endereços, negocia o pagamento de dívidas, recalcula a dívida inadimplente, emite boletos de cobrança, realiza a baixa de parcelas no sistema, edita e consulta as ocorrências.
Credor	Cliente pessoa jurídica que concede o crédito direto ao consumidor, realiza financiamento de veículos, etc.	Principal contato com a diretoria da empresa. Negocia a cobrança das carteiras e produtos, acompanha o andamento das cobranças, especifica campanhas estratégicas, realiza a baixa de parcelas pagas em outras assessorias, consulta faturas, consulta borderôs de prestação de contas.
Devedor	Pessoa física ou jurídica que possui dívidas em atraso ou não.	Consultar dívidas, emite a segunda via de boleto, realiza o pagamento das parcelas em atraso.
Diretor	Principal executivo da empresa, responsável por manter relações empresariais e contatar novos clientes.	Contrata novos credores, analisa os resultados, toma decisões estratégicas com base nos resultados obtidos.
Faturamento	Departamento responsável pelo faturamento.	Emite faturas, processa relatórios analíticos de faturamento, emite o borderô de prestação de contas dentro do prazo do fechamento contábil.
Financeiro	Departamento responsável pelo	Responsável pelo fluxo de caixa, recebimento de pagamentos, baixa de parcelas pagas na própria assessoria,

	controle das finanças.	emissão de recibo de pagamento.
Gerente de Projetos	Responsável pelo planejamento e controle do projeto.	Planejamento de custos, recursos, elaboração de cronogramas, realização de análises de risco.
Supervisor da cobrança	Responsável pela distribuição e controle das cobranças.	Cria estratégias de cobrança para aprimorar o potencial dos auxiliares de cobrança, gera relatórios de desempenho da cobrança.

Com base na tabela 4-1, são identificados três grupos de interessados: o grupo técnico, o grupo estratégico e o grupo operacional.

4.1.1 Considerações do grupo técnico de interessados

O administrador de banco de dados deve entender a semântica e o ciclo de vida da informação para assegurar que a integridade e as regras de consistência sejam obedecidas. Tipicamente os operadores de cobrança tem relatado que o processo de negociação com os devedores tem sido prejudicado devido ao aumento no tempo de resposta do sistema atual.

Ao monitorar as operações executadas nas tabelas do ambiente operacional da cobrança, o administrador de banco de dados percebeu que algumas tabelas da cobrança permanecem bloqueadas por um longo período de tempo, prejudicando a desempenho da cobrança como um todo. Para abordar essas questões é essencial aplicar o ponto de vista de concorrência em conjunto com especificações de cenários de escalabilidade.

Além disso, o administrador de banco de dados deve assegurar que permissões apropriadas sejam concedidas ou revogadas para cada perfil de usuário. As orientações para tratar das considerações com a segurança da informação são obtidas através da especificação de cenários de segurança.

Uma das principais considerações do administrador de redes é certificar-se de que o sistema esteja disponível. Também é de sua responsabilidade elaborar estratégias

para escalabilidade dos sistemas. Em relação à segurança, assegurar a integridade e a confidencialidade dos dados transmitidos pelas redes pública e privada.

O analista desenvolvedor especifica requisitos funcionais, analisa e projeta modelos detalhados considerando o mapeamento dos objetos de domínio para as pilhas de controle e o mapeamento dos artefatos produzidos para o ambiente de implantação. No que diz respeito aos requisitos arquiteturalmente significativos, ele se preocupa com a modificabilidade do sistema, elaborando interfaces bem definidas para evoluírem com confiança e, na medida do possível, atribuindo responsabilidades únicas para os componentes, diminuindo as dependências entre eles.

Considerando que os sistemas desenvolvidos são integrados a outros sistemas, é de responsabilidade dos analistas desenvolvedores especificar os contratos de operação para troca de informações com aplicações externas, descrevendo a sintaxe e a semântica das informações sendo trocadas.

Os desenvolvedores também são responsáveis pelo refinamento de processos e tarefas, trabalhando para reduzir os pontos de alta contenção. As orientações para abordar tais considerações são obtidas através das especificações de cenários de escalabilidade que tem como meta analisar o tempo ocioso em espera por recursos compartilhados, considerando que tal espera inibe a escalabilidade do sistema.

Em nível macro, o arquiteto de software se preocupa em entender os processos de negócio, as principais funções do sistema, a semântica das informações, o mapeamento dos elementos funcionais para o ambiente de execução e a implantação dos artefatos em seus respectivos nós. Além de apoiar nas especificações, análise, modelagem e avaliação de cenários de qualidade, o arquiteto dá suporte ao planejamento do projeto e às negociações entre requisitos conflitantes.

Conforme a análise das considerações do grupo técnico de interessados, a tabela 4-2 resume os atributos de qualidade identificados que representam a base inicial para apoiar as especificações dos cenários de qualidade.

Tabela 4-2 Atributos de qualidade designados ao grupo técnico

Interessado	Atributos de qualidade
Administrador de banco de dados	Escalabilidade, disponibilidade e segurança.
Administrador de redes	Escalabilidade, disponibilidade, segurança.
Analista Desenvolvedor	Modificabilidade, interoperabilidade, escalabilidade.
Arquiteto de Software	Modificabilidade, interoperabilidade, escalabilidade, disponibilidade, segurança e usabilidade.

4.1.2 Considerações do grupo operacional de interessados

O grupo operacional é caracterizado por contemplar os principais usuários do sistema. O auxiliar de cobrança realiza a negociação do pagamento das dívidas contatando o devedor por telefone. Com relação às qualidades do sistema, o auxiliar de cobrança precisa que o sistema esteja disponível e responda rapidamente às solicitações para renegociação das dívidas. Além disso, possui considerações associadas à usabilidade para que seja possível cumprir com suas responsabilidades com menos sobrecarga de memória, mais eficiência e eficácia.

Os devedores desejam pagar suas dívidas através da internet, em qualquer dia da semana e em qualquer horário. Além disso, os devedores também apreciam respostas rápidas. A experiência dos colaboradores da empresa tem mostrado que grande parte do público de devedores são usuários inexperientes em informática, portanto é desejável que as interfaces projetadas para esse perfil de usuário sejam concebidas com o raciocínio voltado para usabilidade. Além disso, os devedores não desejam que as informações de sua dívida estejam desprotegidas em uma rede pública como a internet, portanto a segurança também é fundamental.

A tabela 4-3 apresenta os atributos de qualidade que servem como apoio para as especificações de cenários de qualidade do grupo operacional.

Tabela 4-3 Atributos de qualidade designados ao grupo operacional

Interessado	Atributos de qualidade
Auxiliar de cobrança, Faturamento, Financeiro	Escalabilidade, disponibilidade e usabilidade.
Devedor	Escalabilidade, disponibilidade, segurança e usabilidade.

4.1.3 Considerações do grupo estratégico de interessados

Todos os interessados possuem algum conhecimento sobre o negócio da empresa. Em particular, o diretor, o gerente de projetos e o supervisor da cobrança estão cientes dos riscos de mercado inerentes a uma operadora de cobrança.

Para aprimorar o potencial da recuperadora de ativos financeiros, contribuir para o planejamento e evitar a perda da carteira por atrasos indesejados nas prestações de contas, esses grupos de interessados consideram essencial à facilidade de manutenção, a evolução do sistema e a disposição para estabelecer comunicação com outros sistemas. A indisponibilidade do sistema é um fator significativo para o negócio, pois afeta diretamente as transações da cobrança e implica na perda de credibilidade pela queda do desempenho na posição das maiores assessorias de cobrança.

O credor, além de ser um usuário exigente, é o principal cliente da empresa. É imprescindível que a cobrança de seus ativos seja feita de forma confiável. A disponibilização da funcionalidade para baixa de parcelas pagas para os credores introduz uma preocupação maior com relação à disponibilidade do sistema.

No que diz respeito à segurança da informação, o credor se preocupa com a confidencialidade e integridade de seus ativos financeiros. Além de tudo, o tempo que o credor tem disponível para realizar suas atividades é curto. Portanto, para ser aceitável pelo credor o sistema precisa ser usual. A tabela 4-4 apresenta os atributos de qualidade que apoiam as especificações de cenários de qualidade do grupo estratégico.

Tabela 4-4 Atributos de qualidade designados ao grupo estratégico

Interessado	Atributos de qualidade
Diretor, Gerente de Projetos e Supervisor da cobrança	Modificabilidade, disponibilidade, interoperabilidade, escalabilidade, segurança e usabilidade.
Credor	Modificabilidade, disponibilidade, escalabilidade, segurança e usabilidade.

4.2 Projeto da arquitetura

Uma das principais atividades realizadas pela assessoria de cobrança é a cobrança amigável de ativos financeiros propriamente dita. Em particular, os interessados do grupo estratégico e interessados do grupo operacional acreditam que o valor agregado que a assessoria visa alcançar, depende do apoio de um sistema de informação aliado aos processos da cobrança, faturamento e prestação de contas.

Embora existem outras complexidades associadas ao domínio de cobrança, como por exemplo, a efetividade na localização de devedores e a cobrança judicial, este estudo de caso dedica-se, em princípio, à automatização dos processos de cobrança e faturamento.

4.3 Priorização de cenários funcionais

Em uma reunião com a diretoria executiva, gerentes, supervisores e usuários chave estabelecem o escopo inicial do projeto. Conforme a proposta do framework, tal projeto é conduzido por meio de um método iterativo. Para não desviar o enfoque dos objetivos principais do projeto, a priorização de cenários é designada através da orientação obtida pelos interessados. Por simplicidade, são priorizados os fluxos principais de quatro casos de uso, representados pelas especificações de quatro cenários funcionais conforme mostra a tabela 4-5.

Tabela 4-5 Priorização de cenários funcionais

Prioridade	Cenário funcional	Justificativa
1	Realizar cobrança de produto massificado	A cobrança de produto da carteira do massificado representa mais do que 50% do faturamento da assessoria de cobrança. Não só a diretoria, mas

		também os gerentes e supervisores concordam que caso a cobrança não seja automatizada apropriadamente, o risco para o negócio como um todo é enorme.
2	Realizar transferência on-line	A assessoria de cobrança têm trabalhando arduamente para elevar a recuperação dos ativos financeiros. Uma das metas designadas pela gerencia é modernizar o sistema para usufruir de alguns dos benefícios que a tecnologia pode propiciar como o pagamento de contas on-line.
3	Efetuar a baixa de uma parcela	As atividades de baixas das parcelas, estritamente associadas com o processo de pagamento, possuem peculiaridades que não podem ser abordadas somente ao término do projeto.
4	Emitir relatório de previsão de receitas	Para que seja possível realizar o acompanhamento das cobranças, monitoramento, controle e gerenciamento é indispensável que o sistema possua um mecanismo bem elaborado que suporte a incorporação de diversos tipos de relatórios gerenciais, sumarizados ou analíticos.

Embora o escopo designado seja extremamente reduzido para um projeto real, é suficiente para propósitos deste estudo de caso. No decorrer das quatro iterações, os cenários funcionais especificados fornecem os subsídios iniciais para modelagem dos pontos de vista. Após aplicação dos pontos de vista, os requisitos arquiteturalmente significativos são expressados através de especificações de cenários de qualidade que podem implicar em refinamentos em um ou mais modelos elaborados no decorrer da aplicação dos pontos de vista.

4.4 Realizar cobrança de produto massificado

De acordo com a priorização definida, as especificações funcionais se iniciam pelo cenário principal de cobrança de produto massificado[21]. Para realizar tal

[21] Diz respeito às carteiras que oferecem linhas de crédito para um público em massa, como crédito direto ao consumidor (CDC) e cartões de crédito.

especificação, o arquiteto entrevista os usuários chave da cobrança. A tabela 4-6 resume os principais eventos que ocorrem em um processo de cobrança desempenhado pelo auxiliar de cobrança.

Tabela 4-6 Cenário: realizar cobrança de produto massificado

Elemento	Declaração
Cód. modelo	VREQ_FUN
Id. Caso de Uso	UC-001
Descrição	O objetivo deste caso de uso é realizar a cobrança de um produto da carteira do massificado.
Interessados	Supervisor da cobrança, Gerente de Projetos, Diretor, Arquiteto de Software, Analista Desenvolvedor.
Evento Iniciador	Seleção da operação de cobrança de um produto da carteira do massificado.
Atores	Auxiliar de cobrança.
Pré-condição	Usuário autenticado como auxiliar de cobrança e sistema no modo de cobrança de um produto da carteira do massificado.
Sequência de Eventos	1. Usuário solicita pesquisa de devedores por nome e CPF. 2. Sistema lista os devedores inadimplentes para cobrança, atribuidos ao usuário logado. Cada registro contém os campos [Número do contrato, nome do produto, nome do devedor, CPF, telefone e e-mail]. 3. Usuario seleciona um registro e confirma. 4. Sistema lista as dívidas do devedor. Cada registro contém o número da parcela, o número de dias em atraso, valor do principal, valor dos juros, multa, honorários, despesas, descontos, valor corrigido, data de vencimento, data de baixa, data de previsão de pagamento, número do recibo. 5. Usuário seleciona uma dívida e informa: o número das parcelas para cálculo, data de previsão de pagamento, taxa de honorário cobrada, taxa de juros cobrada, taxa de multa cobrada e solicita uma nova cobrança. 6. Sistema inicia uma nova cobrança e recalcula os dias em atraso, despesas, descontos, taxas, honorários, multa, sendo que a taxa de juros varia por credor ou por credor e produto, sendo calculada pelo método de juros simples ou juros compostos.

	7. Usuario solicita emissão de boleto de cobrança. 8. Sistema emite boleto de cobrança em nome do devedor, contendo o CPF do devedor, endereço, número das parcelas cobradas, data de emissão, valor total cobrado, juros totais cobrados e multa total cobrada. 9. Usuário solicita envio de boleto de cobrança para o e-mail do devedor e registra as ocorrências no sistema. 10. Sistema envia boleto para o e-mail do devedor.
Pós-Condição	1. Boleto emitido com sucesso. 2. Data de previsão de pagamento registrada com sucesso.

Usualmente, os credores tomam suas próprias decisões à respeito do método para cálculo dos encargos devidos. Em particular, cada credor possui uma taxa de juros e um método para cálculo que varia entre juros simples e juros compostos.

Supondo que a assessoria de cobrança possua cerca de 70 credores, as expectativas indiquem um aumento de 30% da carteira nos próximos dois anos e a responsabilidade em efetuar os cálculos dos encargos é atribuída à classe Cobranca. Sendo assim, 21 novos métodos poderiam ser adicionados à classe Cobranca, implicando em perda de coesão e consequentemente inviabilizando uma manutençao confiável que, tipicamente, precisa ser realizada em um curto período de tempo.

Para proteger a classe Cobranca de tal variação, o modelo proposto abstrai o método para cálculo dos juros de modo à diminuir possíveis transtornos ao receber novas especificações de cálculo no decorrer do ciclo de vida do sistema.

Com base na análise dos eventos do caso de uso, são derivadas as classes de domínio específicas do cenário funcional, conforme mostra a figura 4-1.

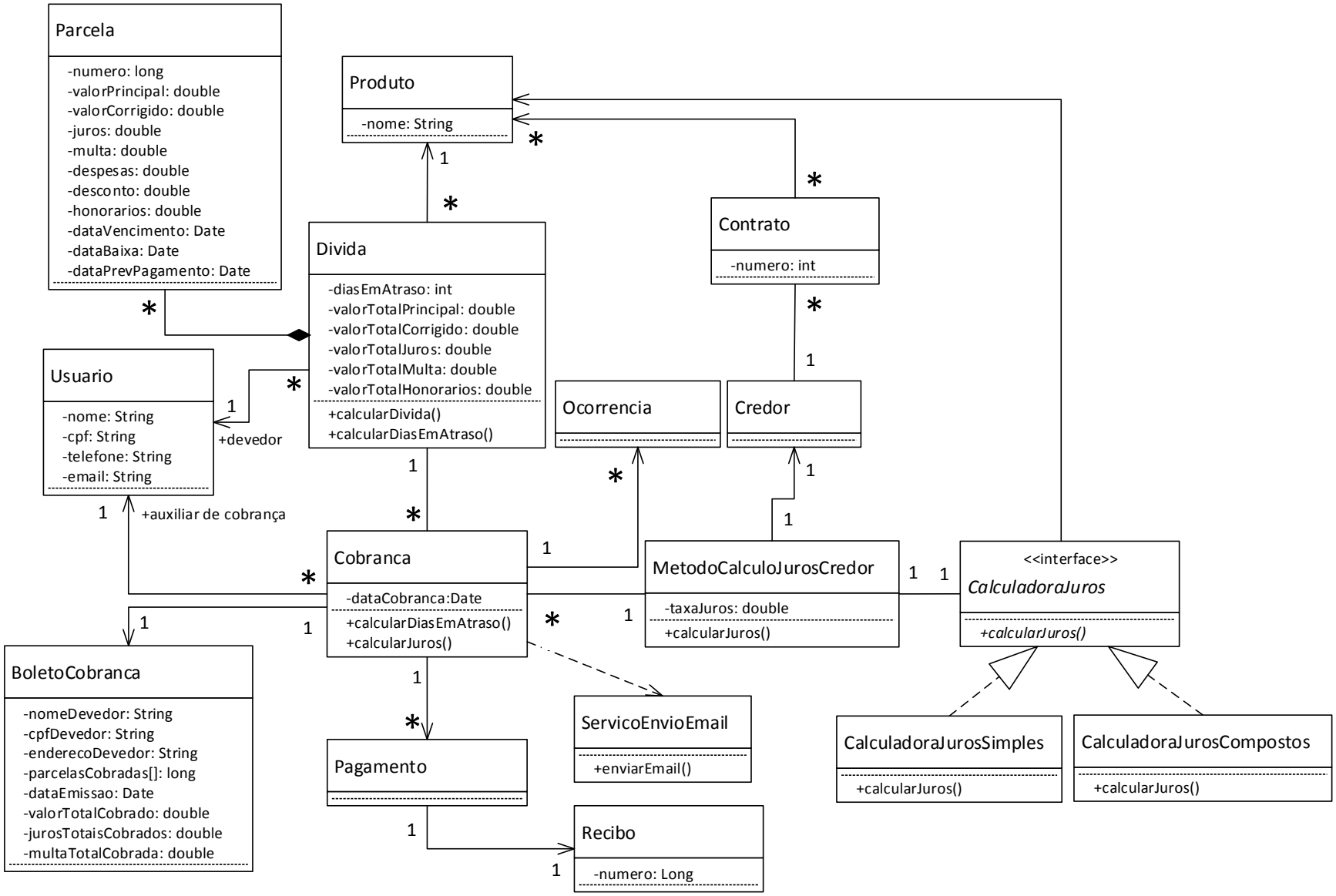


Figura 4-1 Classes derivadas do caso de uso UC-001.

4.4.1 Aplicação do ponto de vista de informação

Conforme apresentado pelas considerações do grupo estratégico de interessados, é inadimissível à uma assessoria de cobrança prestar contas de informações não confiáveis. Do ponto de vista do credor a cobrança deve obedecer criteriosamente as regras estabelecidas por ele, desde o cálculo da dívida vencida até a emissão dos borderôs para prestar contas.

Tipicamente, os credores especificam campanhas para cobrança com o propósito de elevar a recuperação dos ativos. Embora diversas campanhas contemplem regras semelhantes, cada uma possui restrições específicas que se aplicam sobre diferentes propriedades das parcelas, implicando em diferentes valores de comissão e de repasse. Além de variar por credor, as campanhas variam pelo número de dias em atraso das parcelas vencidas. Os diretores e o arquiteto concordam que caso os requerimentos designados pelo credor não sejam cumpridos o risco de perder a carteira é muito grande.

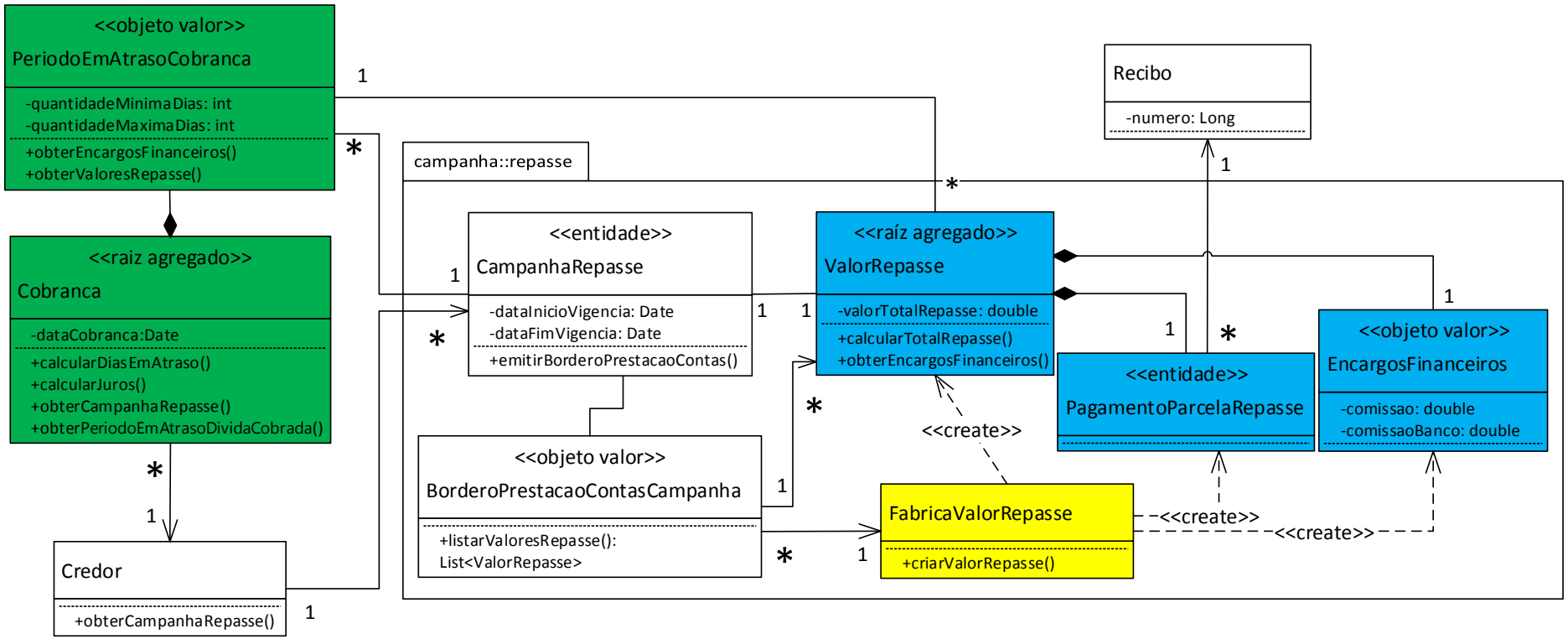
As campanhas estão rigorosamente associadas aos valores de repasse e de comissão necessários para a emissão dos borderôs de prestação de contas. Embora na prática o cálculo dos encargos seja efetuado somente após o pagamento das parcelas cobradas, uma importante regra de consistência se aplica entre o período da dívida em atraso, calculado com base na data da cobrança, e os valores de comissão e de repasse.

Tais invariantes evidenciam a existência de uma preocupação maior no que diz respeito aos valores de repasse e de comissão calculados com base nas regras das campanhas.

A tabela 4-7 apresenta a modelagem candidata para o ponto de vista de informação.

Tabela 4-7 Modelo do ponto de vista de informação.

Elemento	Declaração
Questão	A complexidade para cálculo dos valores de comissão e de repasse é relativamente alta. As campanhas estratégicas podem variar por credor e pelo numero de dias de atraso das parcelas vencidas.
Motivação	Reduzir a probabilidade de ocorrerem problemas durante a emissão dos borderôs de prestação de contas e o esforço com manutenções de última hora caso sejam detectadas inconsistências nos relatórios enviados aos credores.
Correspondência	Este modelo representa um refinamento do modelo de classes derivado do cenário funcional UC-001, onde a classe Cobranca é remodelada para tornar visível o conceito de periodo em atrado da divida cobrada, contribuindo para a redução de duplicidade de código e para o cálculo apropriadado dos valores de repasse e de comissão.



É pouco provável que os conceitos associados representem apenas números arbitrários. Utilizando como apoio os padrões da modelagem dirigida por domínios, os conceitos assimiliados são materializados e retratados diretamente no modelo.

Utilizando como hipótese de modelagem o padrão ‘Agregado’, o conceito de Valor de Repasse é explicitamente tipado no modelo de domínio. Nesse caso, o repasse é designado como raíz de seu próprio agregado, sendo responsável por assegurar a consistência entre os valores de comissão e de repasse, de acordo com as regras da campanha vigente.

Além disso, a campanha tendo como principal argumento o período em atraso da divida cobrada, tal informação é claramente definida no modelo para ser melhor compreendida pelos desenvolvedores. Com o apoio do padrão ‘Objeto Valor’, o período em atraso da divida cobrada também é materializado no modelo como um objeto imutável, uma vez que o período em atraso se refere à data em que a cobrança foi realizada e não a data atual.

A modelagem proposta contribui para o espaçamento através do isolamento conceitual entre o agregado da cobrança, representado pelas classes de cor verde, e o agregado do repasse representado pelas classes de cor azul. Além disso, retrata as classes específicas da campanha de repasse em um pacote separado.

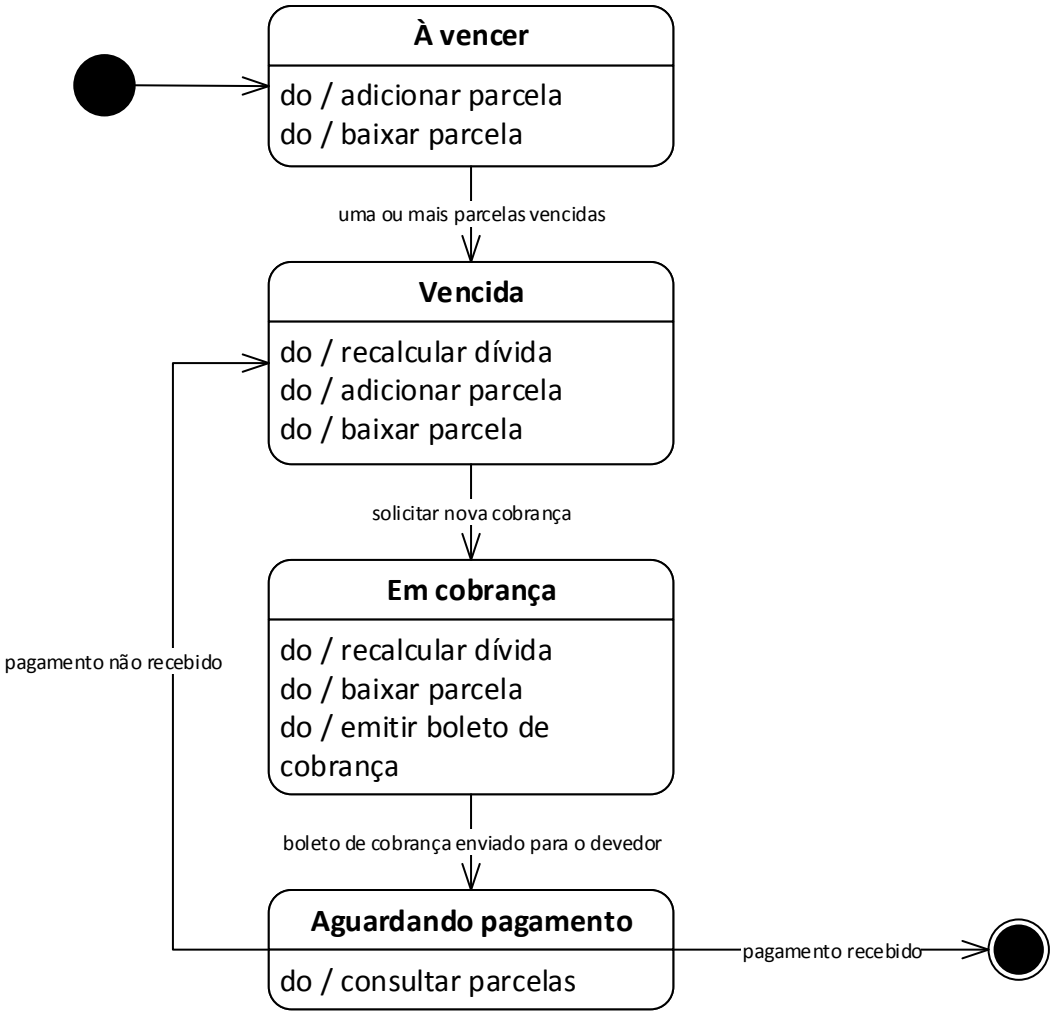
4.4.2 Aplicação do ponto de vista de concorrência

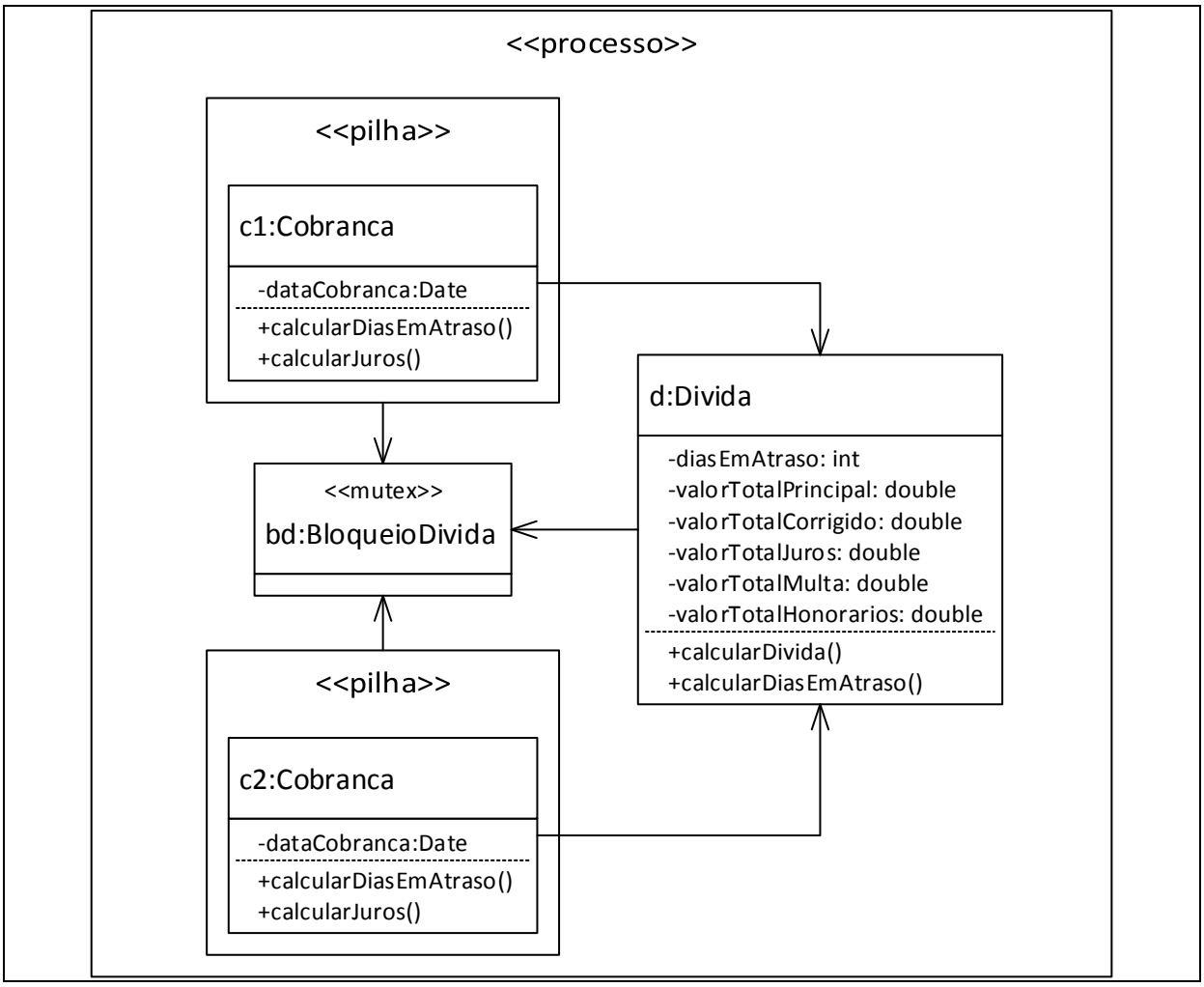
No que diz respeito às operações da recuperadora de ativos financeiros, a atividade mais executada é a negociação de dívidas inadimplentes. Durante as cobranças das carteiras de produto massificado a operação para soma da dívida inadimplente pode ser invocada simultaneamente entre um usuário cobrador e um usuário credor ou entre um usuário cobrador e o supervisor da cobrança.

Considerando que alterações simultâneas nas parcelas da dívida podem conduzir a divida à um estado indeterminado, o estado interno do objeto é estabelecido como um ponto crítico que precisa ser explícitamente representado pelo modelo. Assim, o modelo VCON_PASS é concebido conforme mostra a tabela 4-8.

Tabela 4-8 Proteção contra alterações simultâneas em uma instância da Divida

Elemento	Declaração
Cód. modelo	VCON_PASS
Ponto de vista	Concorrência
Interessados	Supervisor da cobrança, Gerente, Analista desenvolvedor, Arquiteto de software.
Objeto	Instância da classe Divida





A máquina de estados da Divida torna explícito que durante os estados [Vencida] e [Em cobrança] tanto as atividades para o recálculo quanto as atividades para baixa de parcelas podem ser simultâneamente desempenhadas. Assim, introduz-se como ponto de contenção uma instância da classe BloqueioDivida para proteger o estado interno de um objeto Divida por meio de exclusão mútua. Quando uma pilha de controle tenta adquirir o bloqueio para adicionar uma parcela, efetuar o recálculo da dívida ou a baixa de uma parcela paga, o bloqueio é concedido caso outra pilha de controle não o tenha adquirido.

4.4.3 Cenário 1: Campanhas para reparcelamento

Aprimorar a performance de uma empresa de cobrança não é uma tarefa trivial, pois o lucro depende essencialmente de devedores inadimplentes. No entanto, supõe-se que muitos dos devedores estão inadimplentes devido à uma circunstância, por exemplo o desemprego. O lucro da empresa depende da volta desses consumidores

ao mercado de crédito. Uma das estratégias que a empresa têm adotado é a especificação de campanhas estratégicas, onde são concedidos meios de pagamento diversificados para atender a demanda de devedores de diferentes perfis econômicos. A tabela 4-9 apresenta o cenário para modificabilidade das campanhas de reparcelamento.

Tabela 4-9 Cenário de modificabilidade para as campanhas estratégicas

Elemento	Declaração
Cód. modelo	VREQ_QUA
Cenário funcional	UC-001
Interessados	Analista Desenvolvedor, Arquiteto de Software, Diretor, Gerente de Projetos, Supervisor da cobrança, Credor.
Atributo de qualidade	Modificabilidade
Estímulo	Implementar campanha estratégica para reparcelamento de dívidas para devedores que estão com períodos entre 90 e 149 dias em atraso, 150 e 239 dias em atraso, 240 e 359 dias em atraso, 360 e 720 dias em atraso, oferecendo parcelamento de até 9 vezes, com ou sem correção, sendo que a entrada é obrigatória caso o número de dias em atraso seja superior à 90 dias. A regra para cálculo dos encargos deve ser bastante flexível.
Medida de resposta	Codificação, homologação e implantação em até 20 horas.

A princípio manifestam-se duas opções para o problema: a primeira e mais trivial seria atribuir a responsabilidade para as definições e regras de campanhas à um módulo separado onde a cada nova campanha lançada o mesmo módulo seria alterado para incorporar as regras da nova campanha. A outra opção menos trivial seria isolar as campanhas em uma abstração para flexibilizar a incorporação de diversas estratégias de campanha ao sistema. A tabela 4-10 documenta essa questão e a decisão tomada pelo arquiteto.

Tabela 4-10 Decisões de arquitetura para modificabilidade das campanhas

Decisão de arquitetura			
Cód. modelo	DARQ-01		
Questão	Alguns credores podem lançar novas campanhas com a intenção de aumentar a recuperação dos ativos, criando estratégias específicas para refinanciamento e amortização de dívidas.		
Interessado 1	Supervisor de cobrança, gerente da cobrança.	**Suposição 1**	A quantidade de credores que lançam novas campanhas é pequena. Além disso, talvez cada um deles especifique uma nova campanha a cada semestre.
Interessado 2	Analista desenvolvedor.	**Suposição 2**	Ao conceber, através do polimorfismo, um mecanismo que facilite a configuração dos módulos de cada campanha, é possível implantar uma nova campanha e disponibilizá-la para o usuário decidir, em tempo de execução, qual campanha está ativa.
Proposta	Opção 1: Escrever as regras das campanhas diretamente no código “hard coded”, considerando que a frequência de mudanças é baixa. Opção 2: Desenvolver um componente na forma de um plugin, que pode ser facilmente atualizado ou substituído.		
Decisão e justificativa	Desenvolver um componente separado ao invés de escrever rigidamente as regras no código. - A opção 1 produz código de baixa qualidade, viola a coesão e incentiva a duplicidade de código. - A opção 2 é mais robusta, promove variação protegida com o desenvolvimento de um componente separado, isolando o restante do sistema da variação sobre as novas regras das campanhas.		

O cenário introduz desafios para o projeto por não especificar antecipadamente todas as opções possíveis de parcelamento e as regras para efetuar os cálculos dos encargos. Outra questão é que no momento em que o credor lança uma nova campanha não há muito tempo para perder em esforços com codificação. Para contribuir para a flexibilidade exigida, a solução proposta abstrai as opções de acordo através de uma interface econômica que contém apenas a operação reparcelarDivida() para efetivar os acordos. Adota-se como hipótese de modelagem

o padrão plugin de [FOWLER-03]. A figura 4-2 apresenta o esboço da modelagem candidata.

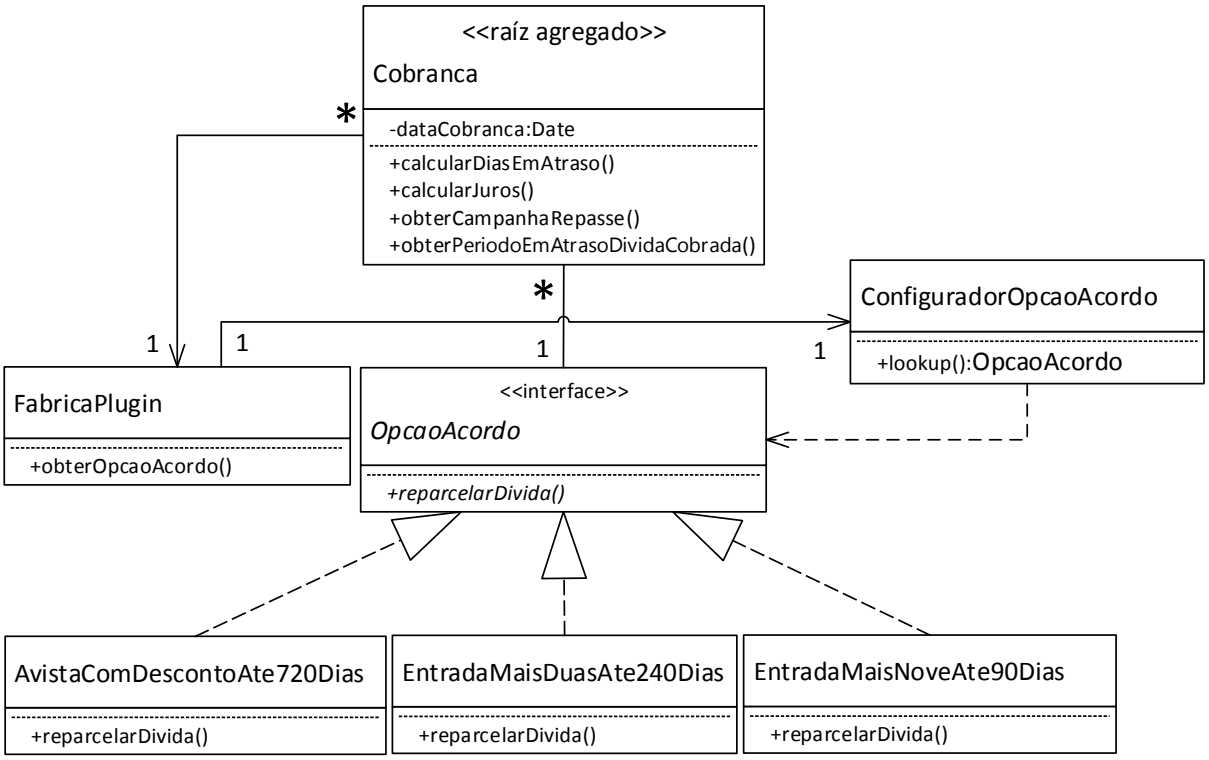


Figura 4-2 Estratégia para modificabilidade das campanhas.

A abstração dos acordos permite que novos tipos de parcelamento sejam incorporados à qualquer momento ao sistema bastando encapsular as novas regras em uma implementação da interface OpcaoAcordo e modificando o plugin para encontrar a nova estratégia do acordo.

4.4.4 Cenário 2: Campanhas para devedores sem credibilidade

Alguns devedores inadimplentes, descumprem com as obrigações por motivo de crises financeiras. Quando são concedidas outras formas de pagamento adequando na medida do possível ao seu perfil econômico, esses devedores usualmente quitam as suas dívidas. Na tentativa de ajudá-los a voltar ao mercado, a recuperadora de ativos formula a seguinte estratégia: oferecer até 50% de desconto para quitação do débito corrigido à vista. A tabela 4-11 apresenta o cenário de modificabilidade.

Tabela 4-11 Cenário de modificabilidade para campanha com desconto à vista

Elemento	Declaração
Cód. modelo	VREQ_QUA
Cenário funcional	UC-001
Interessados	Analista Desenvolvedor, Arquiteto de Software, Diretor, Gerente de Projetos e Supervisor da cobrança.
Atributo de qualidade	Modificabilidade
Estímulo	Implementar campanha para devedores sem credibilidade (SPC ou Serasa) que não honraram seus acordos, oferecendo até 50% de desconto para quitação do débito corrigido à vista. A campanha é válida somente para alguns tipos de produtos. A aplicação do desconto dependerá da vigência da campanha.
Medida de resposta	Codificação, homologação e implantação em até 16 horas.

O cenário especifica a estratégia para oferecimento de desconto para devedores sem credibilidade no mercado. Sabemos que a regra para calculo de desconto pode variar. Para proteger o sistema de tal variação, é adotado como hipótese de modelagem, o padrão estratégia de [GAMMA-94]. A figura 4-3 apresenta a aplicação do padrão.

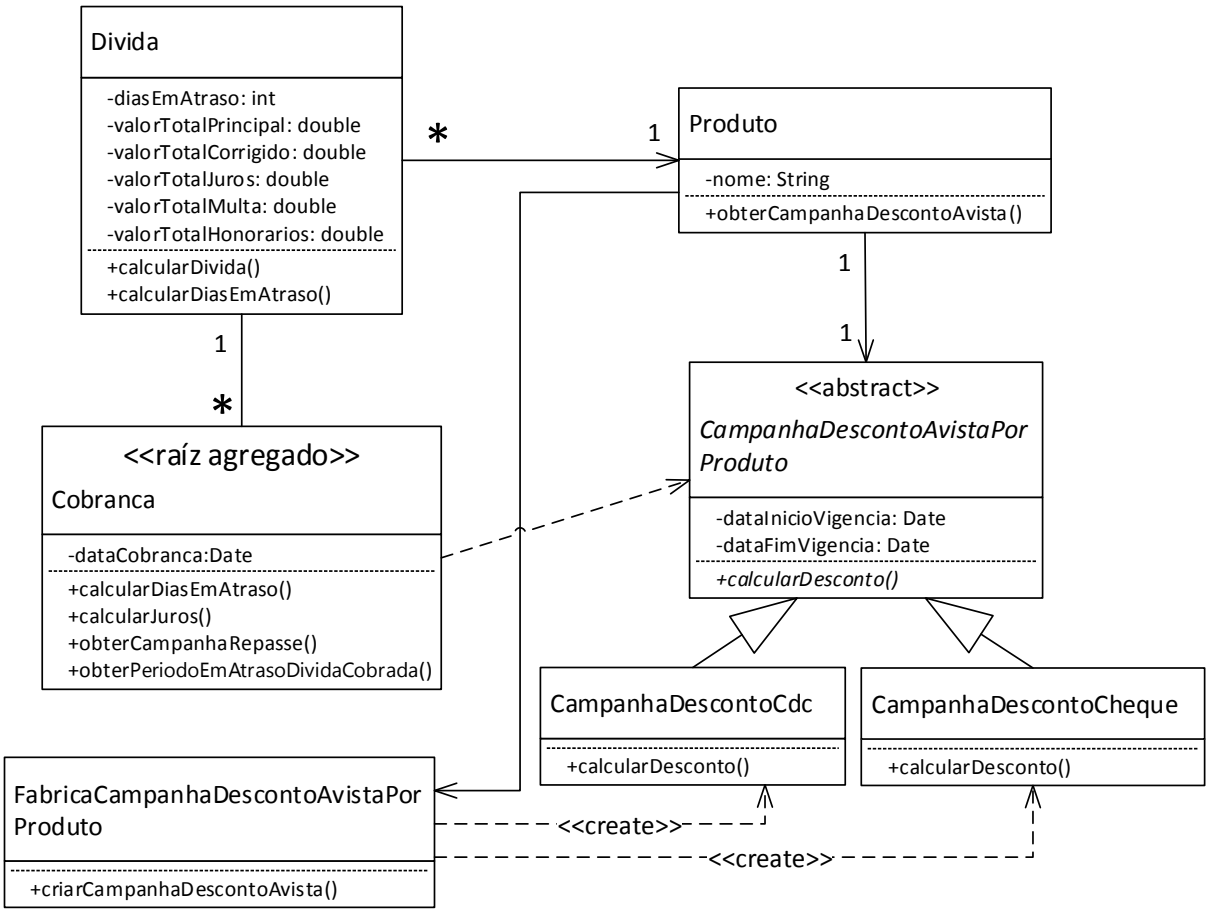


Figura 4-3 Campanha de desconto à vista.

A solução proposta contribui para que a incorporação de novas estratégias de desconto por produto possam ser adicionadas ao sistema com implicações mínimas sobre o código existente, através do encapsulamento das regras específicas e as taxas de desconto designadas à cada produto específico.

4.4.5 Cenário 3: Baixa latência nas transações da cobrança

A modelagem do ponto de vista de concorrência introduziu um ponto de contenção para instâncias da classe Divida através de exclusão mútua. A principal questão associada com a solução é que, na medida em que o número de usuários simultâneos aumenta, o tempo de espera pela liberação do bloqueio também aumenta.

Além disso, existe um ponto de conformidade com o framework proposto governado pela regra de correspondência Reg_Pcont. Tal regra de correspondência visa

contribuir para a escalabilidade do sistema, eliminando os pontos de alta contenção introduzidos na modelagem do ponto de vista de concorrência. Para abordar a questão o arquiteto especifica o cenário de escalabilidade, conforme mostra a tabela 4-12.

Tabela 4-12 Cenário de escalabilidade da cobrança

Elemento	Declaração
Cód. modelo	VREQ_QUA
Cenário funcional	UC-001
Interessados	Supervisor de Cobrança, Gerente de Projetos, Diretor, Auxiliar de cobrança, Analista Desenvolvedor, Arquiteto de Software, Administrador de Banco de dados.
Atributo de qualidade	Escalabilidade
Estímulo	Um auxiliar de cobrança entra em contato com o devedor para cobrar as parcelas vencidas de um produto massificado como CDC. Simultâneamente, o credor acessa o sistema pela internet a fim de incluir uma baixa parcial da mesma dívida, que foi paga pelo devedor em outra assessoria.
Ambiente	Carga de 150 transações por minuto.
Resposta	Sistema verifica divida indadimplente obtendo os dias em atraso, calcula os encargos de cada parcela vencida e sumariza os valores totais da divida, mantendo a consistência entre os valores totais da dívida e os valores totais das parcelas, independentemente do número de acessos simultâneos na mesma divida.
Medida de resposta	90% das transações retornam o controle para o usuário em até 3 segundos.

O cenário requer um modo de garantir a consistência da dívida, pois no momento em que o sistema calcula o valor de inadimplência, são desconsideradas todas as parcelas baixadas. Para propósitos de alteração consideram-se a dívida e as parcelas como sendo uma única unidade. A alteração de uma parcela implica em recalcular os valores totais da dívida. Assim, ao realizar a baixa de uma parcela, a dívida inteira precisa ser bloqueada para garantir sua integridade.

Além da garantia de invariantes, o cenário especifica que mesmo com uma carga alta, 90% das transações deveriam possuir uma latência média de até 3 segundos. Com a aplicação do padrão bloqueio de alta granularidade[22] em conjunto com o padrão bloqueio otimista de [FOWLER-03], a raiz de um agregado[23] é utilizada como ponto de contenção único, de modo que um conjunto de objetos associados é bloqueado pelo mesmo objeto de versão. A figura 4-4 apresenta a aplicação dos padrões bloqueio de alta granularidade e bloqueio otimista.

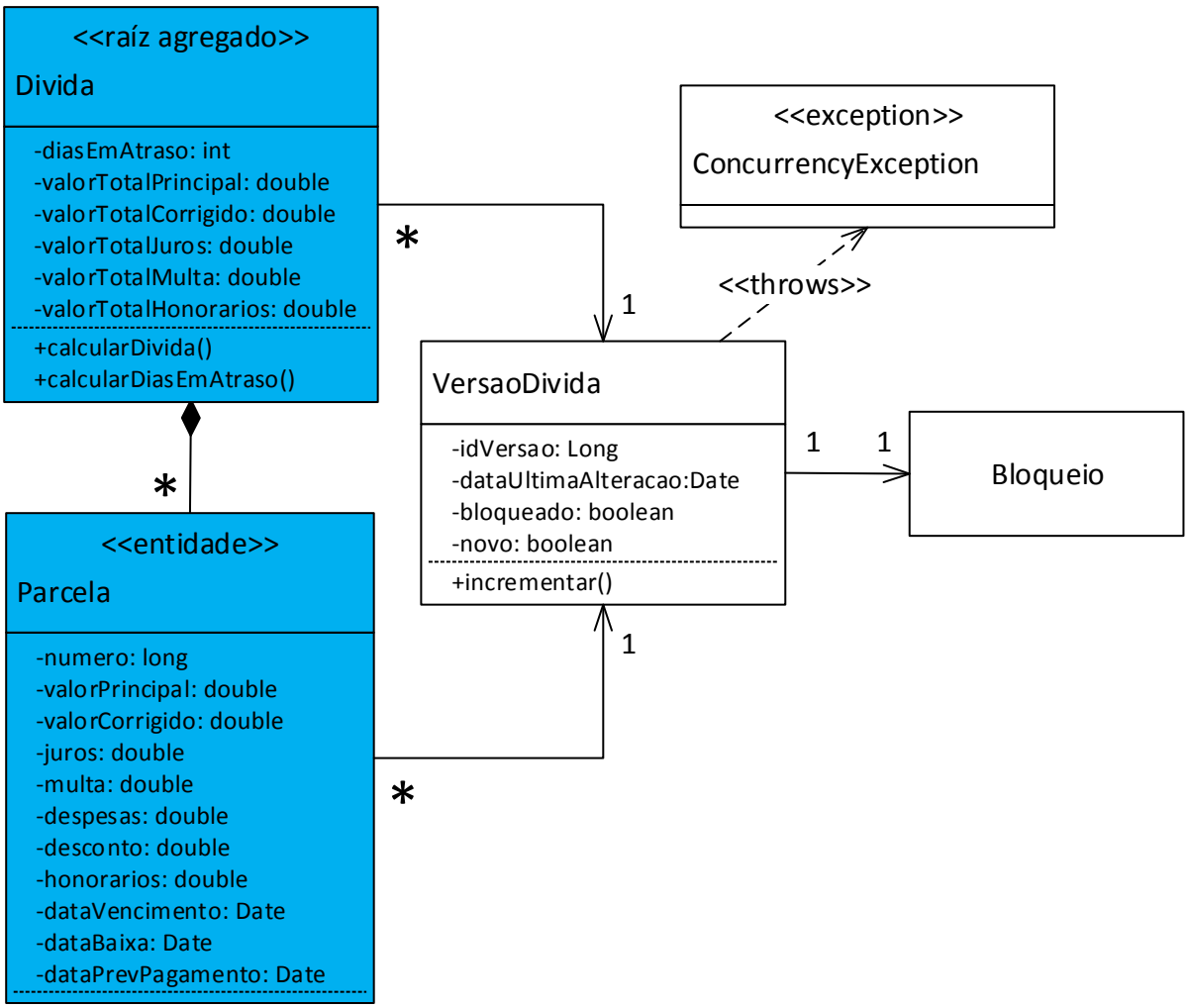


Figura 4-4 Aplicação do bloqueio de alta granularidade e bloqueio otimista.

A solução proposta elimina a necessidade de consultar todas as parcelas associadas a divida para adquirir bloqueios individuais, implicando em menos sobrecarga de memória e processamento para gerenciamento dos bloqueios. Nesse

[22] do inglês, coarse-grained lock.
[23] Um contorno conceitual representado por associações entre objetos em que um deles é estabelecido como a raiz responsável pelos membros do agregado e pelas invariantes que se aplicam entre os objetos do agregado [EVANS-03].

caso, somente a dívida, sendo a raiz do agregado, é bloqueada durante as transações. As parcelas são bloqueadas utilizando o mesmo bloqueio da dívida. Com a aplicação dos padrões o custo de operação para aquisição e remoção de bloqueios em todas as parcelas da dívida é menor, com a introdução do ponto de contenção único, contribuindo para uma latência média de três segundos[24] sem abrir mão da consistência e escalabilidade exigidos pelo cenário.

4.5 Realizar transferência on-line

Uma das grandes expectativas dos diretores, gerente e supervisor da cobrança é a liberação dos pagamentos online na internet. Essa estratégia visa atingir o público de devedores que dispõe de pouco tempo para deslocar-se até a agência de cobrança ou ao banco para pagarem suas dívidas. Para contribuir com tais expectativas, o arquiteto realizada entrevistas com os usuários chave, em particular do departamento financeiro para especificar o cenário funcional de transferência on-line, conforme mostra a tabela 4-13.

Tabela 4-13 Cenário: realizar transferência on-line

Elemento	Descrição
Cód. modelo	VREQ_FUN
Id. Caso de Uso	UC-002
Descrição	O objetivo deste caso de uso é realizar o pagamento de uma ou mais parcelas vencidas.
Interessados	Financeiro, Supervisor da cobrança, Gerente de Projetos, Diretor, Arquiteto de Software, Analista Desenvolvedor.
Evento Iniciador	Seleção da operação de pagamento via transferência on-line.
Atores	Devedor, Sistema de pagamento.
Pré-condição	Usuário autenticado como devedor e sistema no modo de operação de pagamento via transferência on-line.
Sequencia de eventos	1. Usuário solicita a pesquisa de dívidas. 2. Sistema pesquisa as dívidas do usuário e apresenta uma lista contendo as parcelas, sendo que em cada registro é exibido o

[24] Na prática existem outros fatores que contribuem para a redução de latência, inclusive o layout físico de implantação, a eficiência do hardware, a configuração de memória, a latência da rede e o ambiente como um todo. É necessário o apoio de ferramentas de teste para assegurar que tal medida de resposta seja efetivamente alcançada no contexto particular.

	número da parcela, data de vencimento, data de baixa, número do recibo, valor do principal, juros calculados, multa calculada e descontos calculados. 3. Usuário seleciona uma ou mais parcelas vencidas e solicita o recalculo da dívida. 4. Para cada parcela vencida, o sistema calcula o valor total do principal, o valor total dos juros, o valor total da multa, o valor total dos honorários, os descontos, realiza a somatória dos totais da dívida e apresenta o resultado. 5. Usuário solicita o pagamento das parcelas calculadas via transferência on-line. 6. Sistema inicia novo pagamento para as parcelas contendo os números das parcelas e o valor total calculado. Sistema apresenta as opções de banco disponíveis para transferência on-line. 7. Usuário informa o código do banco e confirma. 8. Sistema solicita o número da agência, o número da conta, o dígito da conta para débito, o e-mail do usuário e o número do CPF. 9. Usuário informa o número da agência, número da conta, dígito da conta para débito, e-mail, número do CPF e confirma. 10. Sistema valida dados da conta para débito, e-mail e CPF do usuário, armazena dados do pagamento e envia uma mensagem para o sistema de pagamento contendo o valor da transferência, o código do banco, o número da agência, o número da conta e o dígito da conta para débito, o código do banco, o número da agência, o número da conta e o dígito da conta para crédito, o número do CPF e a senha do usuário. Sistema retorna mensagem informando que a confirmação do pagamento será enviada por email em alguns instantes.
Pós-Condição	1. Dados do pagamento armazenados com sucesso. 2. Data de previsão de pagamento registrada com sucesso.

Para suportar o processo de transferência online do ponto de vista funcional, são identificados os conceitos associados à operação de pagamento como transferência on-line, informações do banco e tipos de operações comuns para transferência entre contas como débito e crédito. A figura 4-5 mostra o esboço de modelagem candidata.

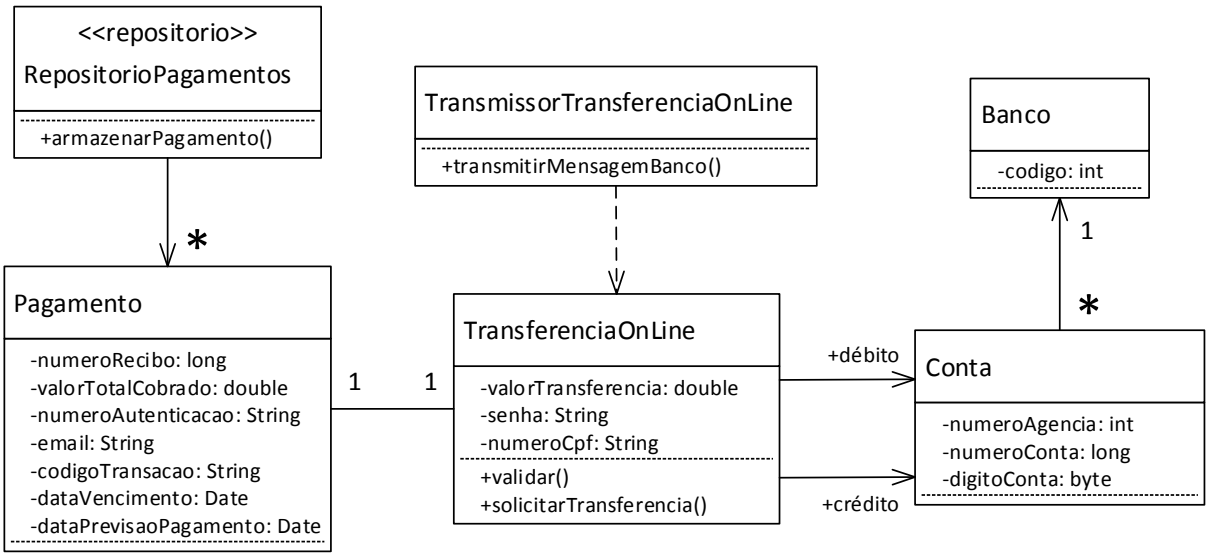


Figura 4-5 Classes derivadas do cenário UC-002.

A classe RepositorioPagamentos é estereotipada pelo padrão Repositorio de [EVANS-03]. Tal repositório é responsável pelo armazenamento dos pagamentos. Cada transferência possui um objeto Pagamento correspondente. A classe TransferenciaOnLine possui dois acoplamentos com a classe Conta, um que representa a conta para débito e outro que representa a conta para crédito.

4.5.1 Aplicação do ponto de vista de informação

Através de entrevistas realizadas com os interessados pelo cenário funcional de transferência on-line, em particular o supervisor da cobrança, gerente de projetos e diretor, é posssível extrair uma série de conceitos ainda implícitos, com relação aos fluxos do cenário funcional de transferências on-line.

Embora a principal meta da assessoria de cobrança seja elevar os índices de recuperação dos ativos financeiros, não há como garantir que as cobranças sendo exercidas sejam efetivamente pagas pelos devedores. Isso significa que uma parcela pode ser cobrada diversas vezes até que a baixa possa ser efetivada.

Na medida em que o tempo de inadimplência aumenta, as regras da cobrança mudam para refletir a correção dos novos valores calculados para juros, multa, comissão, repasse, etc.

Tecnicamente quando um objeto representando a divida cobrada está no estado ‘Aguardando pagamento’ e a data do vencimento é alcançada, o objeto volta ao estado ‘Vencida’. Esse ciclo de vida precisa ser mantido para o gerenciamento apropriado das cobranças. De modo que, novas cobranças da mesma divida não devem sobrepor o ciclo de vida de cobranças anteriores.

Para contribuir com a gestão das cobranças, assegurando integridade, consistência e confiabilidade dos valores cobrados, a modelagem candidata propõe a segregação do agregado da Dívida, identificado pelas classes em cor azul, com um novo agregado Cobrança, identificado pelas classes em cor de laranja conforme mostra o esboço de modelagem candidata da tabela 4-14.

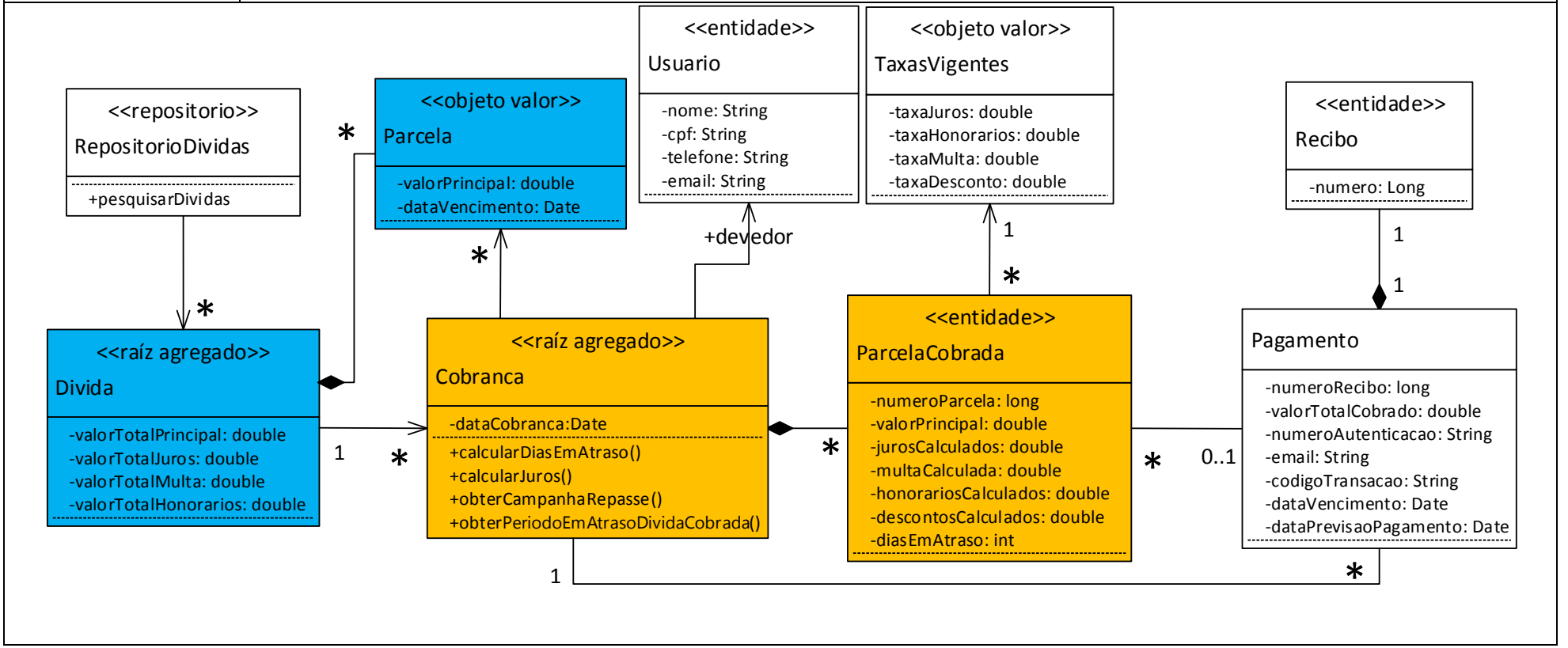
A solução proposta implica na implementação da classe Parcela como um objeto imutável, responsável por armazenar os valores do principal e a data de vencimento.

Os valores calculados com base no número de dias em atraso[25] são de responsabilidade da classe ParcelaCobrada que deve assegurar que as taxas de juros, honorários, multa e descontos cobrados correspondam com as taxas aplicáveis no período vigente.

25 Calculado pela diferença entre a data atual e a data de vencimento da parcela.

Tabela 4-14 Modelagem do ponto de vista de informação.

Elemento	Declaração
Questão	O recálculo dos valores de cobrança com base no valor do principal e no período vigente, considerando dias em atraso, juros, multa, honorários e descontos não deve implicar na perda dos cálculos efetuados em cobranças anteriores.
Motivação	Assegurar que os valores das parcelas cobradas sejam mantidos no sistema mesmo após efetuar diversas vezes a cobrança da mesma dívida para apoiar o gerenciamento e controle apropriado das cobranças.
Correspondência	Esse modelo corresponde com o modelo VREQ_FUN, cenário UC-002, onde a classe Cobranca é segregada para representar a raiz conceitual de seu próprio agregado, responsável por controlar o ciclo de vida das parcelas cobradas.



Cada vez que uma cobrança é realizada, duas novas entidades Cobranca e ParcelaCobrada são criadas para representar o progresso da operação atual da cobrança, independentemente de cobranças já realizadas anteriormente.

4.5.2 Cenário 4: Integração com sistema de pagamento

O cenário funcional UC-002 declara um ator que representa o sistema de pagamento externo. Para estar em conformidade com a regra de correspondência Reg_FInt, é necessário especificar, ao menos, um cenário de interoperabilidade para evidenciar os requisitos arquiteturalmente significativos inerentes à esse tipo de transação.

De fato, a modelagem atual das classes Pagamento, TransferenciaOnLine, Conta e Banco esclarecem pouco sobre requisitos arquiteturalmente significativos como tolerância a falhas, disponibilidade, confiabilidade e desempenho, tipicamente associadas com a troca de mensagens entre aplicações distintas. A tabela 4-15 apresenta o cenário de interoperabilidade com o sistema de pagamento externo.

Tabela 4-15 Troca de mensagem com sistema de pagamento externo

Elemento	Declaração
Cód. modelo	VREQ_QUA
Cenário funcional	UC-002
Interessados	Administrador de redes, Analista Desenvolvedor, Arquiteto de Software, Diretor, Gerente de Projetos e Supervisor da cobrança.
Atributo de qualidade	Interoperabilidade
Estímulo	Envio de mensagem para sistema de pagamento externo para realização de pagamento de parcelas vencidas.
Resposta	Sistema de pagamento recebe dados da conta do devedor e as credenciais de segurança, realiza a autenticação, realiza débito da conta informada e o respectivo crédito na conta da recuperadora de ativos financeiros e retorna mensagem de confirmação da transferência. É desejável que o sistema esteja protegido contra a natureza e localização do provedor do serviço.
Medida de resposta	Operação realizada com sucesso em 99% das transferências.

O cenário caracteriza-se pela troca de informações com o sistema de um banco. Além disso, é desejável que o sistema esteja protegido contra a natureza e localização do provedor do serviço. Nesse contexto, o padrão mediador de [BUSCHMANN-96], se enquadra perfeitamente como hipótese de solução.

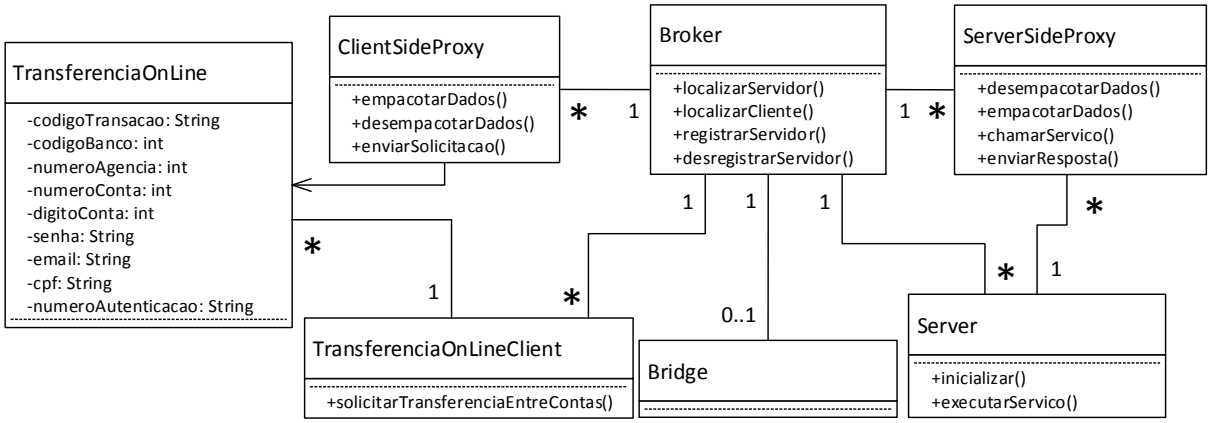


Figura 4-6 Integração com um sistema de pagamento externo.

O mediador torna-se responsável pela comunicação com o provedor do serviço. A indireção protege o módulo de pagamento interno de variações nos elementos ou na localização do sistema externo. Além disso, o isolamento do objeto de transferência online promove a substituição, em tempo de execução, do mediador, caso o sistema remoto esteja indisponível.

4.5.3 Cenário 5: Sistema de pagamentos indisponível

No que diz respeito à disponibilidade do sistema de pagamentos externo, o cenário especifica o comportamento que deve ser desempenhado quando o sistema do banco apresentar falhas. É importante que em tal circunstância as operações de transferência transmitidas permaneçam íntegras e consistentes. Além disso, o usuário deve ser informado sobre possíveis atrasos na conclusão da operação de transferência. A tabela 4-16 resume o cenário de disponibilidade.

Tabela 4-16 Cenário de disponibilidade em operações de transferência on-line

Elemento	Declaração
Cód. modelo	VREQ_QUA
Cenário funcional	UC-002
Interessados	Arquiteto de Software, Administrador de redes, Devedor, Diretor,

	Gerente de Projetos, Supervisor da cobrança e Credor.
Atributo de qualidade	Disponibilidade
Estímulo	Solicitar pagamento via operação de transferência on-line.
Ambiente	Sistema de pagamentos do banco indisponível.
Resposta	Mesmo que o sistema do banco apresente falhas nas primeiras tentativas de entrega, é preciso assegurar que outras tentativas de entrega sejam exercidas. Além disso, ao expirar o tempo necessário para efetivação da operação, uma mensagem de notificação é disparada para o devedor.
Medida de resposta	Mensagem de confirmação por e-mail em até 10 minutos.

O cenário caracteriza-se pelo comportamento tolerante à falhas, onde diversas tentativas de entrega são exercidas até que uma confirmação de sucesso seja recebida ou até que o tempo necessário para conclusão da operação seja atingido.

Como hipótese de modelagem, a solução adota o padrão de entrega garantida de [HOPE-03]. A figura 4-7 apresenta o esboço de modelagem com a aplicação do padrão.

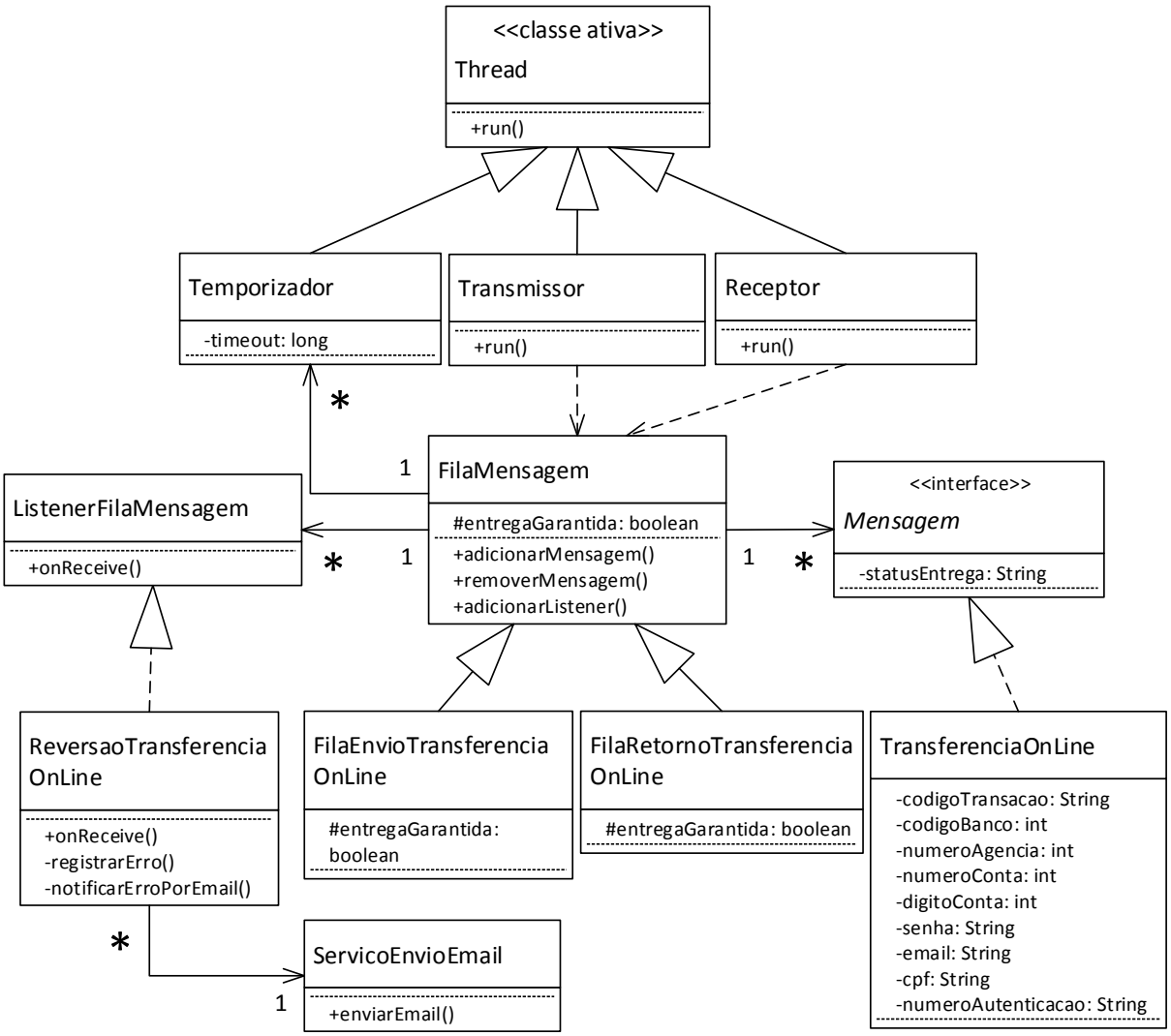


Figura 4-7 Entrega garantida e notificação por email das transferências on-line.

A modelagem candidata consiste em tornar persistentes as mensagens que são transmitidas e recebidas do sistema de pagamentos. Devido à natureza assíncrona das mensagens é necessário conceber uma fila para envio e uma fila para retorno. A interface Mensagem designa um contrato para o cliente que deseja enviar uma nova mensagem para sistema de mensagens. Após dez minutos uma instância da classe Temporizador desperta para que o sucesso ou a falha da operação seja registrado e um e-mail seja encaminhado para o devedor.

4.6 Efetuar a baixa de parcelas pagas

Além de receber pagamentos via internet, o sistema de cobrança contempla opções para pagamento em espécie (dinheiro), depósito e cheque. Neste caso, o ator responsável por receber o pagamento e efetuar a baixa das parcelas pagas é um

funcionário do departamento financeiro. Além do departamento financeiro, o supervisor de cobrança e o gerente de projetos são entrevistados para elicitação do cenário funcional para baixa de parcelas. A tabela 4-17 apresenta o cenário.

Tabela 4-17 Caso de uso: Baixar parcelas pagas

Elemento	Declaração
Cód. modelo	VREQ_FUN
Id. Caso de Uso	UC-003
Descrição	O objetivo deste caso de uso é realizar a baixa de uma ou mais parcelas pagas.
Interessados	Supervisor da cobrança, Gerente de Projetos, Faturamento, Financeiro, Diretor, Arquiteto de Software, Administrador de banco de dados, Analista Desenvolvedor.
Evento Iniciador	Seleção da operação de baixa de parcelas pagas.
Atores	Financeiro
Pré-condição	Usuário autenticado como financeiro e sistema no modo de baixa de parcelas pagas.
Sequência de Eventos	1. Usuário pesquisa a dívida utilizando como critério o nome do devedor e/ou o CPF. 2. Sistema pesquisa as dívidas do usuário e apresenta uma lista contendo as parcelas, sendo que em cada registro é exibido o número da parcela, data de vencimento, data de baixa, número do recibo, valor do principal, juros calculados, multa calculada e descontos calculados. 3. Usuário seleciona uma ou mais parcelas cobradas e solicita a baixa. 4. Sistema solicita a forma de pagamento: em dinheiro, cheque ou depósito. 5. Usuário escolhe a forma de pagamento sendo que caso o usuário selecione cheque deverá ser informado o número do cheque, número do banco e número da agência. 6. Sistema calcula a data para envio do recibo (prestação de contas), que varia pela data do pagamento e dia da semana, emite o recibo de pagamento em nome do credor, realiza a baixa de cada parcela paga, registrando o código do usuário que solicitou a baixa e exibe uma mensagem informativa de confirmação da emissão do recibo.
Pós-Condição	1. Data de baixa registrada em todas as parcelas pagas.

	2. Recibo emitido com sucesso.

Embora muitas das classes relacionadas ao cenário já tenham sido derivadas de cenários anteriores, tais classes são adicionadas no modelo para evidenciar o relacionamento adjacente entre os objetos responsáveis pelo processo da cobrança e os objetos responsáveis pelo processo de pagamento. A figura 4-8 apresenta o esboço de modelagem candidata.

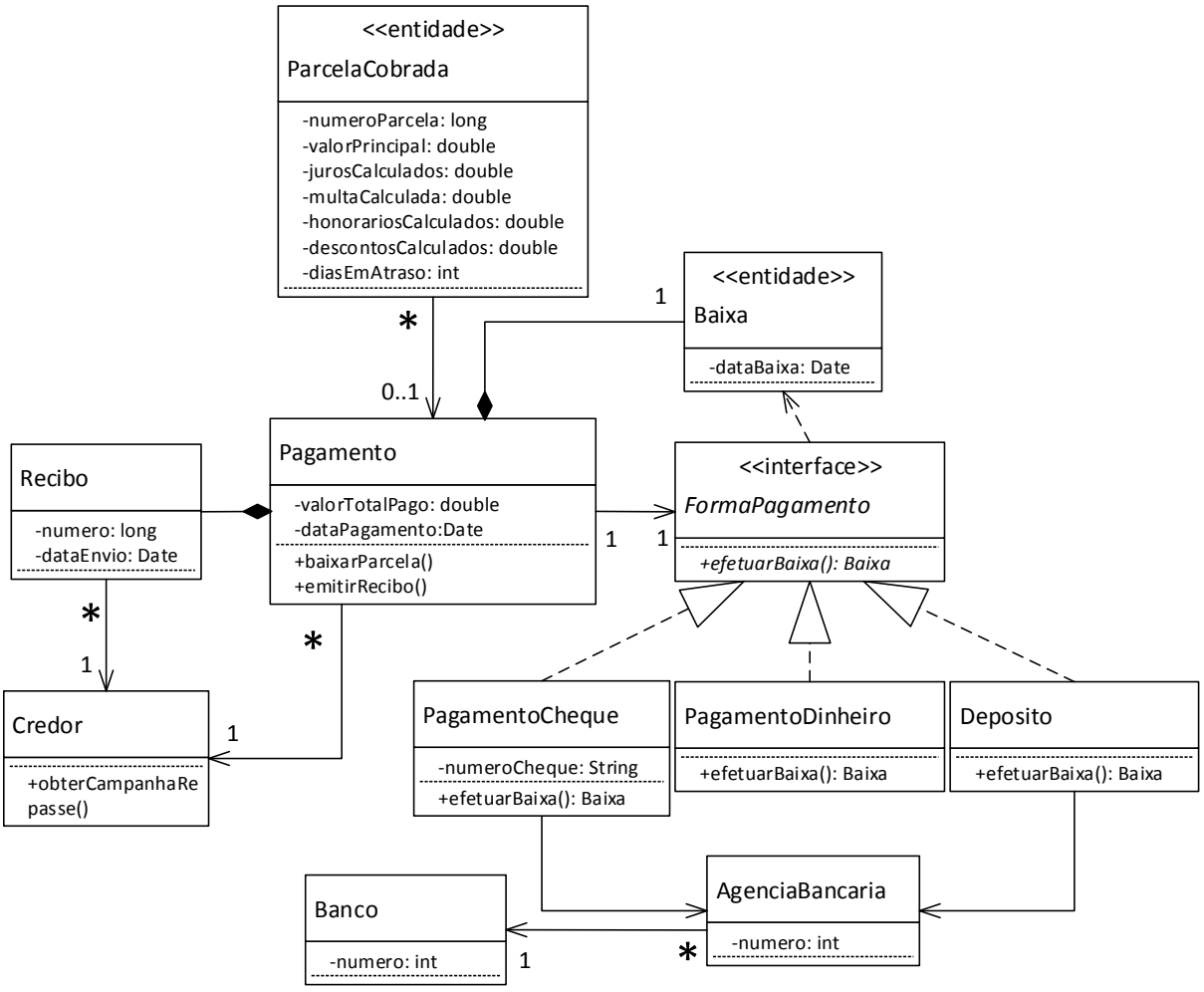


Figura 4-8 Classe derivadas do cenário UC-003.

Conforme mostra o esboço de modelagem da figura 4-8, a baixa das parcelas pagas está estritamente associada com os pagamentos realizados. Para representar as diferentes formas de pagamentos, o pagamento é abstraído para facilitar a incorporação das estratégias de pagamento existentes ou eventualmente novas.

Utilizando como apoio o padrão estratégia de [GAMMA-94] a variação nas formas de pagamento é encapsulada. Embora o departamento financeiro tenha contribuído com a especificação apropriada do cenário funcional para baixa de parcelas pagas, ao entrevistar os outros interessados pelo cenário, em particular o usuário chave do faturamento, um conhecimento mais profundo sobre o domínio é assimilado.

4.6.1 Aplicação do ponto de vista de informação

Embora o cenário funcional UC-003 aparentemente seja simples, ele possui algumas peculiaridades associadas com as parametrizações de datas para envio das prestações de contas. Ao questionar o usuário chave do departamento de faturamento sobre o fluxo das baixas, ele levanta uma questão importante sobre a data de envio das prestações de conta, considerando que o atraso no envio de tais prestações de contas pode implicar em transtornos durante o período de fechamento contábil e perda de credibilidade diante do credor.

Uma entrevista realizada com o responsável pelo faturamento revela conceitos que, embora do ponto de vista funcional possam ser codificados apropriadamente, durante as manutenções podem significar apenas cálculos sem significado para os interessados sem conhecimento especializado do negócio.

Em particular, os interessados do grupo técnico não são especialistas no domínio de pagamentos para projetar um modelo rico em conhecimento e auto documentado, implicando em falta de clareza para os responsáveis pelas manutenções do sistema e consequentemente em riscos maiores de não enviar as prestações de conta nas datas estabelecidas pelos credores. A tabela 4-18 apresenta o documento VINF_MOD.

Tabela 4-18 Modelagem do ponto de vista de informação

Elemento	Declaração
Cód. Modelo	VINF_MOD
Ponto de vista	Informação
Questão	Alguns credores em particular possuem a sua própria estratégia para recebimento dos borderôs de prestação de contas de acordo com seu

	período de fechamento contábil. Por exemplo, caso a data para o repasse (envio da prestação de contas) seja dia 15 e o pagamento tenha sido efetuado até o dia 5 do mesmo mês, a data do envio é registrada para o dia 15, sendo que caso o dia 15 seja domingo, a data para o repasse é D+2, ou seja, dia 17. Caso o dia da semana seja segunda-feira, a data para o envio é D+1, ou seja, dia 16. Para pagamentos efetuados após o dia 5 e antes do dia 20 a estratégia é ligeiramente diferente.
Motivação	Esclarecer as estratégias para computar a data do envio dos borderôs de prestações de contas de acordo com o período de fechamento contábil, contribuir para a visibilidade dos conceitos associados à essas questões importantes para evitar atrasos indesejados no envio das prestações de contas e perda de credibilidade.
Correspondência	Esse modelo refina o modelo derivado pelo cenário funcional VREQ_FUN, cenário UC-003, através da materialização de diversos conceitos associados com a estratégia para computar a data de envio dos borderôs de prestação de contas, designadas pelos credores.

Para evidenciar a importância da data do fechamento contábil a classe FechamentoContabil é modelada como um ‘Objeto de Valor’ [EVANS-03] para tornar visível esse importante conceito associado ao credor, o qual foi modificado para representar a entidade raiz de seu próprio agregado conceitual, conforme destacado pelas classes em vermelho na modelagem candidata, exibida pela figura 4-9.

Através do apoio do padrão fábrica abstrata de [GAMMA-94] os conceitos assimilados são materializados no modelo de modo a não ser de livre interpretação pelo grupo técnico de interessados. Conforme mostra a figura 4-9, é possível distinguir facilmente que cada credor especifica a sua própria estratégia para computar a data de envio das prestações de conta de acordo seu período de fechamento contábil.

A figura 4-9 apresenta o esboço de modelagem candidata que materializa os conceitos implícitos no modelo derivado do cenário funcional UC-003.

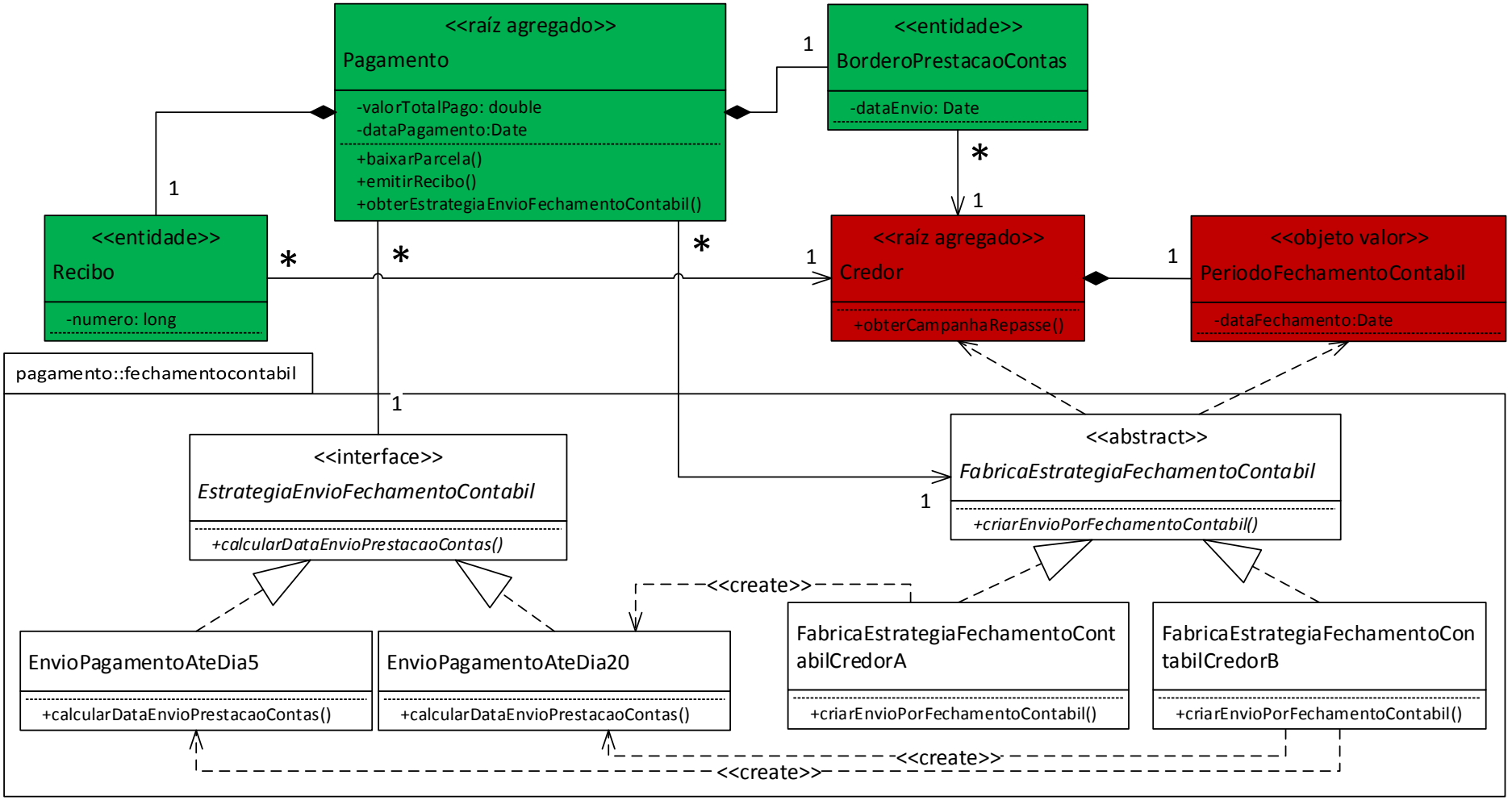


Figura 4-9 Modelagem proposta para o ponto de vista de informação.

Para dar ênfase à distinção semântica existente entre as classes do domínio, as classes derivadas pelo padrão fábrica abstrata são reunidas em um pacote separado, contribuindo para o espaçamento apropriado entre os objetos representando o processo de pagamento e os objetos representando as regras específicas de fechamento contábil dos credores.

Além de promover a visibilidade e o espaçamento adequado, a modelagem proposta promove flexibilidade para substituir as estratégias de envio das prestações de conta, devido à indireção entre a classe Pagamento e as estratégias concretas designadas a cada credor. Caso seja necessário mudar a estratégia de um credor em particular, tal estratégia é facilmente incorporada ao módulo de pagamentos que possui acoplamento baixo com as estratégias, causando impacto mínimo sobre o código existente.

4.6.2 Cenário 6: Restrições para baixa de parcelas

Disponibilizar um serviço de pagamento para o público através da internet implica em diversas considerações associadas à segurança. Operações como baixa de parcelas devem ser acessíveis somente para alguns papéis específicos designados pelo administrador do sistema. O cenário de segurança apresentado pela tabela 4-19 especifica o requisito arquiteturalmente significativo para autorização à baixa das parcelas.

Tabela 4-19 Usuário malicioso tenta baixar uma parcela indevidamente

Elemento	Declaração
Cód. modelo	VREQ_QUA
Cenário funcional	UC-003
Interessados	Administrador de banco de dados, Arquiteto de Software Devedor, Diretor, Gerente de Projetos, Supervisor da cobrança e Credor.
Atributo de qualidade	Segurança
Estímulo	Baixar parcela vencida indevidamente, como se já tivesse sido paga.
Resposta	Sistema bloqueia o acesso à operação de baixa

Medida de resposta	99% das tentativas de baixa de usuários maliciosos mal sucedidas.

As restrições para baixa de parcelas, essencialmente, são critérios adicionados às operações de acesso aos dados. Com respeito à coesão, a responsabilidade dos objetos de acesso a dados deve ser somente acessar dados. Portanto a responsabilidade pelos critérios adicionais de segurança é atribuída a uma classe separada. A hipótese de modelagem para responder ao cenário é a aplicação de um filtro[26] responsável pelos critérios adicionais. Ao invocar operações restritas em um objeto de acesso a dados, o interceptador [BUSCHMANN-11b] é acionado para ativar o filtro responsável pela aplicação dos critérios adicionais de segurança. A figura 4-10 apresenta o esboço de modelagem candidata.

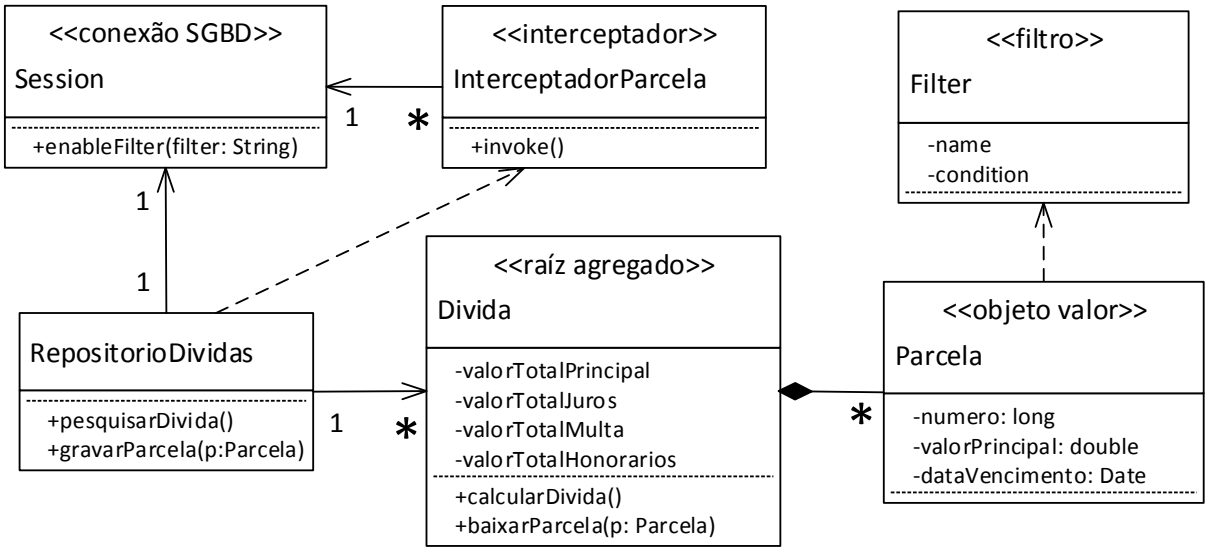


Figura 4-10 Filtros de segurança para as parcelas.

Para desacoplar as reponsabilidades de segurança da informação e conceber uma solução mais robusta é mais factível adotar um framework ou arquitetura de referência[27] que suporte recursos como os interceptadores.

26 O framework de persistência Hibernate possui filtros parametrizáveis que auxiliam no atendimento de aspectos transversais como a segurança [BAUER-07].
27 Arquitetura que representa um ativo reusável associado com um domínio de interesse particular que, tipicamente, inclui padrões aplicados em diferentes áreas da estrutura [EELES-10].

4.6.3 Cenário 7: Mensagem informativa ao ocorrer conflito de versão na dívida

A introdução do padrão de bloqueio de alta granularidade e o bloqueio otimista propostos no cenário 3 implica em uma questão inerente aos bloqueios otimistas que é o tratamento dos conflitos de versão gerados quando dois ou mais usuários alteram simultâneamente o mesmo objeto.

Embora o sistema possa assegurar apropriadamente a integridade e consistência em uma instância da classe Divida, no que diz respeito à usabilidade, o usuário que possui a versão mais antiga da Dívida não deseja receber como resposta uma mensagem contendo o rastreamento da pilha proveniente de uma exceção propagada pelo sistema, quando ocorre um conflito de versão.

É conveniente que o sistema envie uma mensagem informativa instruindo-o a reiniciar a transação devido à informação desatualizada. A tabela 4-20 mostra a especificação do cenário de usabilidade.

Tabela 4-20 Mensagem amigável ao ocorrer um conflito de versão na dívida

Elemento	Declaração
Cód. modelo	VREQ_QUA
Cenário funcional	UC-003
Interessados	Arquiteto de Software, Credor, Supervisor da cobrança, Gerente de projetos
Atributo de qualidade	Usabilidade
Estímulo	Baixar uma ou mais parcelas de um devedor que realizou o pagamento em outra assessoria de cobrança. No mesmo instante, um auxiliar de cobrança decide realizar a cobrança de uma ou mais parcelas da mesma dívida. O credor confirma a baixa com sucesso, mas ao recalcular a dívida, o auxiliar de cobrança recebe uma mensagem propagada pelo sistema proveniente do conflito de versão na dívida.
Ambiente	Carga normal
Resposta	Um dos usuários que alteram o mesmo objeto Divida receberá uma mensagem informativa instruindo-o a reiniciar a transação para obter

	as informações atualizadas recentemente pelo outro usuário.
Medida de resposta	75% de usuários satisfeitos com o tratamento de erro amigável.

Ao solicitar a baixa da parcela da dívida com a versão mais antiga, uma exceção é lançada. Para evitar que o rastreamento da pilha seja apresentado na interface do credor, o objeto controlador se responsabilizará em capturar e tratar a exceção, gravar os detalhes no log e enviar uma mensagem amigável instruindo o credor a reiniciar a operação de baixa. A figura 4-11 apresenta o esboço de modelagem.

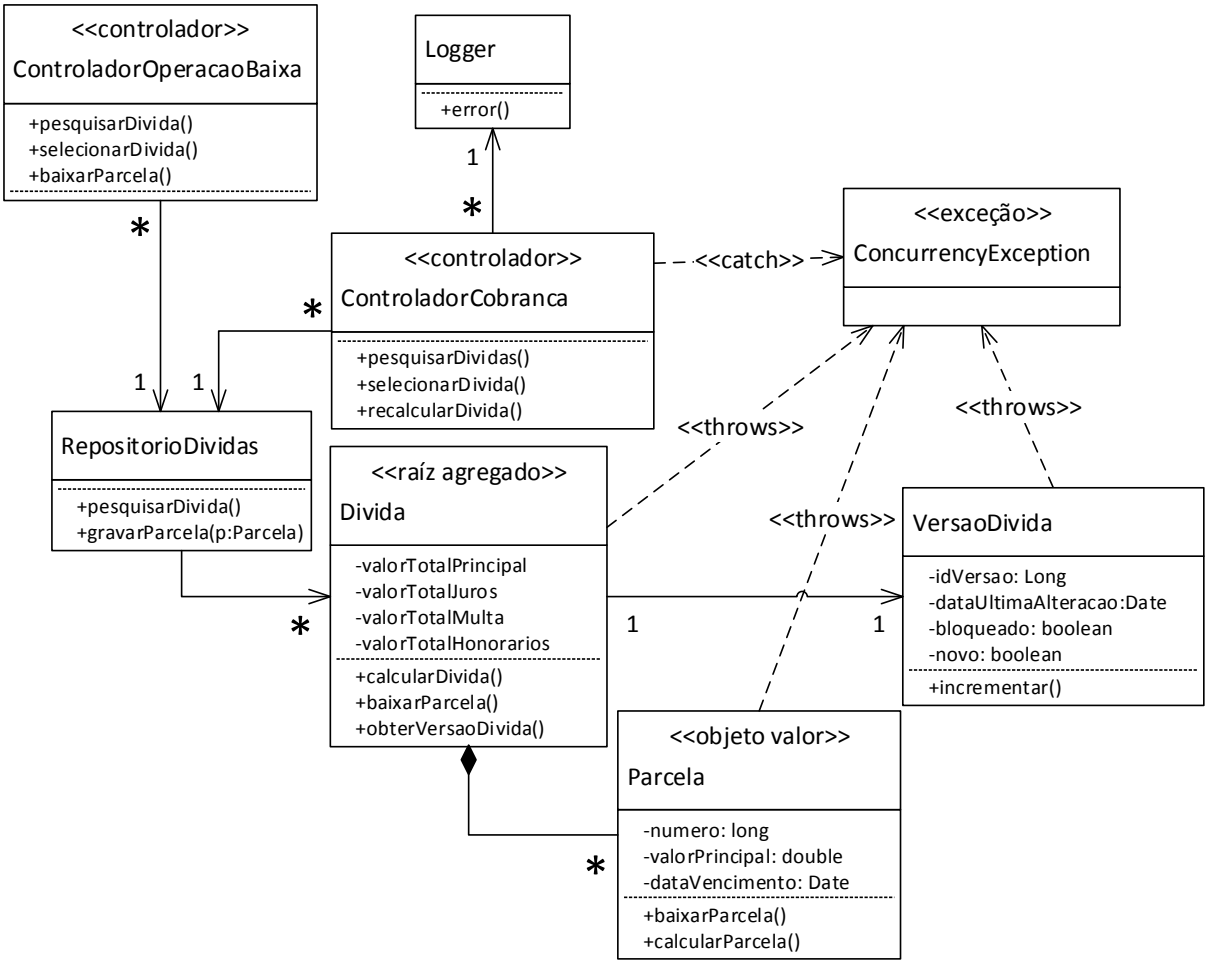


Figura 4-11 Abordagem para o conflito de versão na dívida.

Para simplificação, algumas classes são omitidas do esboço da figura 4-11 para dar enfoque ao tratamento do conflito de versão. A figura 4-12 apresenta o comportamento em tempo de execução como medida preventiva do conflito de versão.

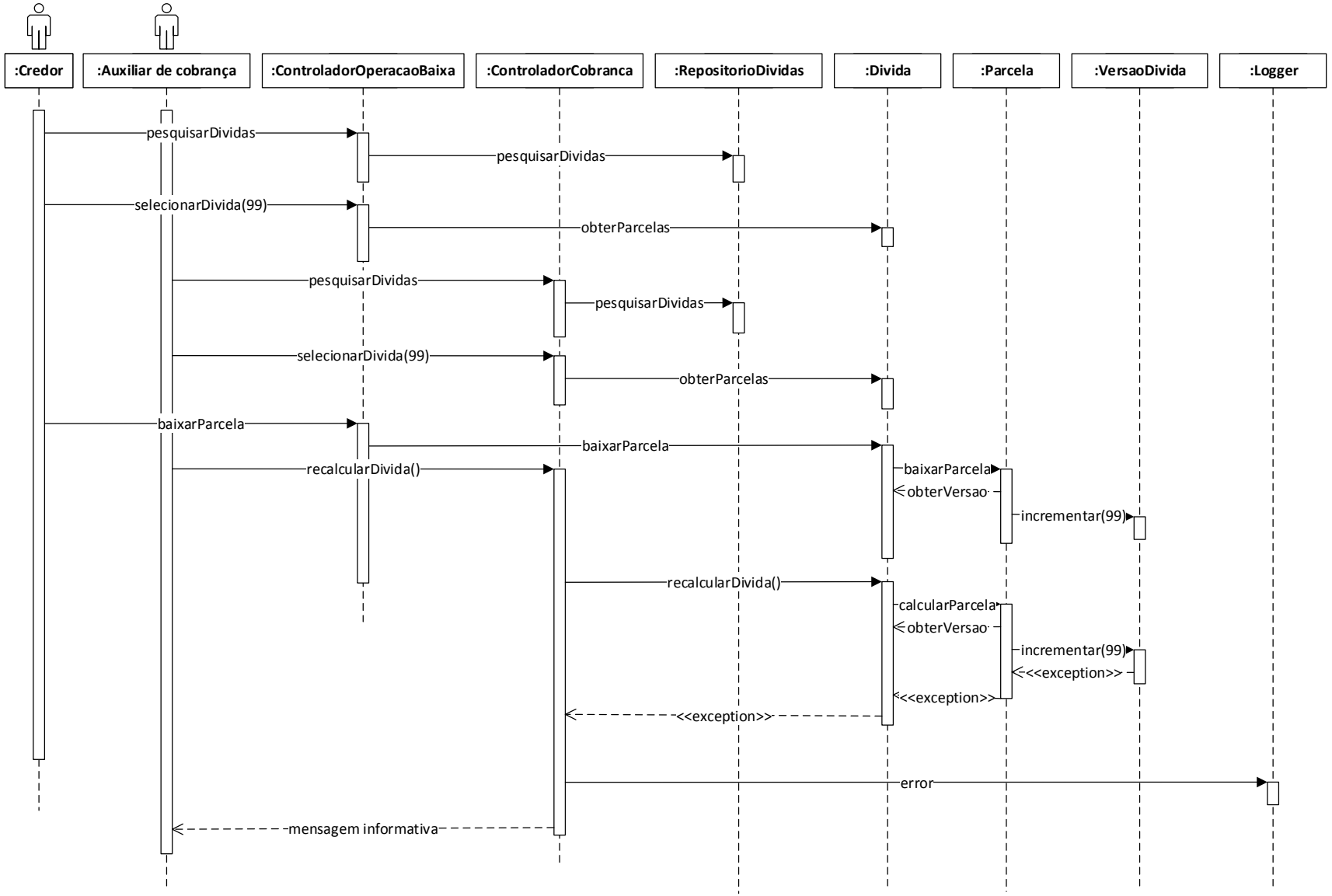


Figura 4-12 Mensagem informativa ao ocorrer um conflito de versão na dívida.

Inicialmente o credor pesquisa a dívida para baixa. Supoe-se que o credor consulta as parcelas da dívida de versão número 99. O auxiliar de cobrança escolhe uma das parcelas da mesma dívida, versão 99, para cobrar.

Antes mesmo de entrar em contato com o devedor, o auxiliar de cobrança decide recalcular a dívida, sendo que no mesmo instante o credor escolhe uma parcela paga e solicita a baixa.

O sistema operacional suspende a pilha de controle designada para o auxiliar de cobrança e seleciona para execução a pilha de controle designada para o credor que conclui a operação de baixa com sucesso implicando no incremento de versão da dívida para o número 100. A partir de então, qualquer transação na antiga dívida, versão 99, será inválida.

A pilha de controle designada para o auxiliar de cobrança é selecionada para execução pelo sistema operacional. O objeto VersaoDivida detecta o conflito de versão e lança uma exceção. A exceção é propagada até o controlador da cobrança que se responsabiliza tratá-la, reverter a transação, registrar a mensagem detalhada em um arquivo de mensagens e configurar uma mensagem amigável para instruir o auxiliar de cobrança à reiniciar a transação.

4.7 Emitir relatórios de previsão de receitas

Frequentemente, os gerentes e supervisores da cobrança desejam ter uma previsão das receitas geradas pela cobrança antes mesmo da confirmação do pagamento. Para suprir tais necessidades gerenciais é preciso realizar consultas diversificadas com critérios variados. A tabela 4-21 apresenta o cenário funcional para emissão de de relatório gerencial.

Tabela 4-21 Cenário: Relatório de ocorrencias registradas na cobrança

Elemento	Descrição
Cód. modelo	VREQ_FUN
Id. Caso de Uso	UC-004
Descrição	O objetivo deste caso de uso é emitir um relatório gerencial.

Interessados	Supervisor da cobrança, Arquiteto de Software, Analista Desenvolvedor.
Evento Iniciador	Seleção da operação de emissão de relatórios gerenciais.
Atores	Supervisor da cobrança.
Pré-condição	Usuário autenticado como supervisor da cobrança operação de emissão de relatórios gerenciais.
Sequencia de eventos	1. Usuário solicita relatório gerencial de cobrança. 2. Sistema solicita o período da cobrança (data inicial e data final). 3. Usuario informa o período da cobrança. 4. Sistema emite relatório de cobrança.
Pós-Condição	Relatório gerencial gerado com sucesso.

Relatórios são caracterizados principalmente por serem consultas de informações estruturadas, não são transacionais e não sofrem alterações após serem implantados em produção. Além disso, são tão específicos que o reuso em outros cenários é pouco provável. Eles possuem essencialmente a lógica organizada por procedimentos. Nessas circunstâncias é conveniente adotar como hipótese de modelagem o padrão script de transação de [FOWLER-03]. A figura 4-13 mostra a aplicação do padrão script de transação para apoio da modelagem de relatórios gerenciais.

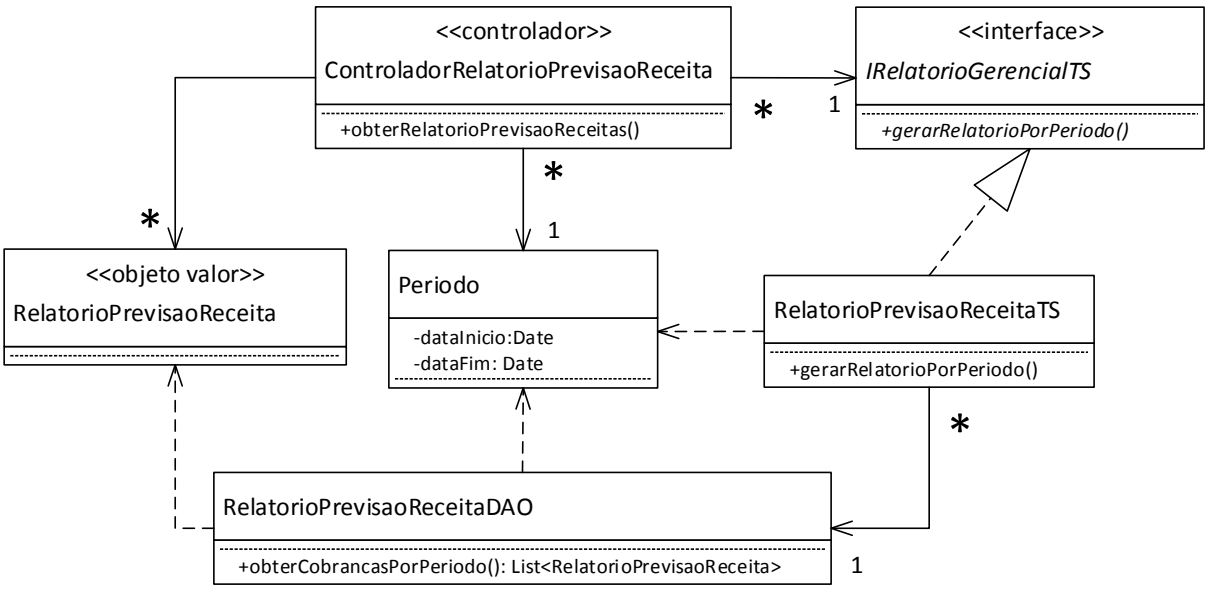


Figura 4-13 Classes derivadas do cenário UC-004.

O relatório de previsão de receitas é responsável por receber a solicitação do usuário, enviar uma mensagem para o objeto de acesso a dados que realiza a consulta no banco de dados subjacente.

4.8 Aplicação do ponto de vista de informação

Ao entrevistar os gerentes e supervisores da cobrança, um aspecto sobre a emissão de relatórios gerenciais é identificado. Algumas informações presentes nos relatórios gerenciais dizem respeito somente ao grupo estratégico de interessados. Outra questão importante é que, tipicamente, o usuário que solicita o relatório nem sempre realiza uma análise imediata. Portanto o relatório emitido precisa ser armazenado para que possa ser consultado posteriormente.

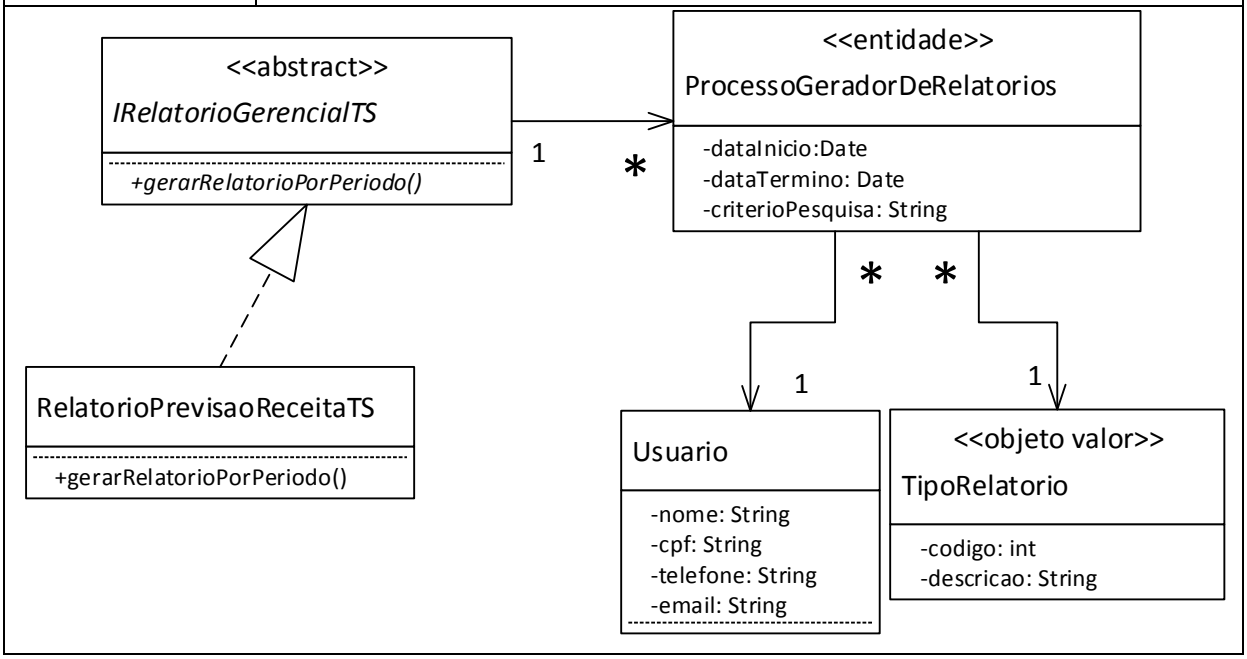
Em algumas circunstâncias os interessados do grupo estratégico precisam recuperar um relatório emitido alguns dias anteriores para propósitos de análise da genuidade das informações, desempenho dos processos ou de padrões que são combinados para análise do progresso das operações da cobrança.

Assim, do ponto de vista de informação, além do usuário que solicitou a emissão do relatório, é preciso registrar a data em que foi solicitado, a data de conclusão, o tipo de relatório e os critérios específicos selecionados pelo usuário no momento da emissão. A tabela 4-22 apresenta o modelo VINF_MOD referente ao ponto de vista de informação.

Tabela 4-22 Modelo do ponto de vista de informação

Elemento	Declaração
Questão	A emissão de relatórios gerenciais requer controles adicionais com respeito ao estado do processo específico que processa o relatório.
Motivação	Permitir que os usuários consultem os relatórios emitidos anteriormente, contribuindo para que o progresso das operações sendo desempenhadas possam ser apropriadamente sumarizadas e analisadas posteriormente para propósitos de tomada de decisão estratégica.
Correspondência	Este modelo refina o modelo derivado pelo cenário funcional VREQ_FUN, UC-004 para adicionar informações específicas ao domínio

	de geração de relatórios gerenciais.



Para suprir as necessidades referentes aos relatórios gerenciais, a solução proposta utiliza como hipótese de modelagem os padrões 'Entidade' e 'Objeto valor' de [EVANS-03] que materializam os conceitos inerentes ao domínio específico de geração de relatórios. Através da solução proposta, os usuários podem recuperar os relatórios solicitados logo após sua geração, indendentemente da data em que foi processado.

4.8.1 Cenário 8: Manter a vazão ao solicitar o processamento de relatórios

Uma das tarefas realizadas frequentemente pelos supervisores da cobrança é a emissão de relatórios analíticos das cobranças. Além dos supervisores, os diretores, gerentes e alguns operadores da cobrança também desempenham essas atividades diariamente.

Uma das preocupações do grupo técnico de interessados diz respeito à perda de vazão das transações sendo realizadas pela cobrança com o aumento do número de solicitações de relatórios. No que diz respeito à escalabilidade, o sistema deve suportar a solicitação de diversos relatórios simultâneos, sem queda de desempenho aparente nas operações da cobrança. Para suprir essa necessidade, o cenário de escalabilidade é especificado, conforme mostra a tabela 4-23.

Tabela 4-23 Cenário de escalabilidade ao solicitar relatórios gerenciais

Elemento	Declaração
Cód. modelo	VREQ_QUA
Cenário funcional	UC-004
Interessados	Supervisor de cobrança, Auxiliar de cobrança, Gerente, Arquiteto de Software.
Atributo de qualidade	Escalabilidade
Estímulo	Solicitar o processamento de um ou mais relatórios gerenciais.
Ambiente	Carga de 150 transações por minuto
Medida de resposta	Latência média de 2 segundos ao solicitar relatórios gerenciais e vazão média de 150 transações por minuto nas operações da cobrança.

Para reduzir a sobrecarga[28] adota-se como solução um pool de pilhas de controle parametrizado para alocar um número reduzido de pilhas. Adicionalmente, é selecionada como hipótese a tática de agendamento recursos de [BASS-12] e uma biblioteca para agendamento de tarefas[29]. Os relatórios serão programados para execução em horários fora de expediente.

Para simplificação, detalhes da modelagem são omitidos do diagrama, tais como filas de espera e estratégia para aquisição e liberação de bloqueios. O tipo de tarefa é abstraído pela interface TarefaExecutavel.

Quando uma tarefa agendada é despertada pelo temporizador, o controlador de tarefas agendadas é acionado. O controlador tenta adquirir uma pilha de controle disponível do pool que repassa a solicitação para o semáforo[30]. Caso o número de permissões disponíveis no semáforo seja alcançado, a pilha de controle do temporizador é bloqueada até que outra pilha seja liberada para o poll. A figura 4-14 apresenta o esboço de modelagem.

28 Do inglês overhead.

29 Ver quartz-scheduler em [WALLS-08].

30 Um mecanismo de bloqueio que mantém um contador. Caso o contador seja maior que zero, a thread pode adquirir o semáforo sem bloquear. Após o contador chegar a zero as threads bloqueiam no semáforo até que se contador seja incrementado como resultado de outra thread liberando o semáforo [SCHMIDT-00].

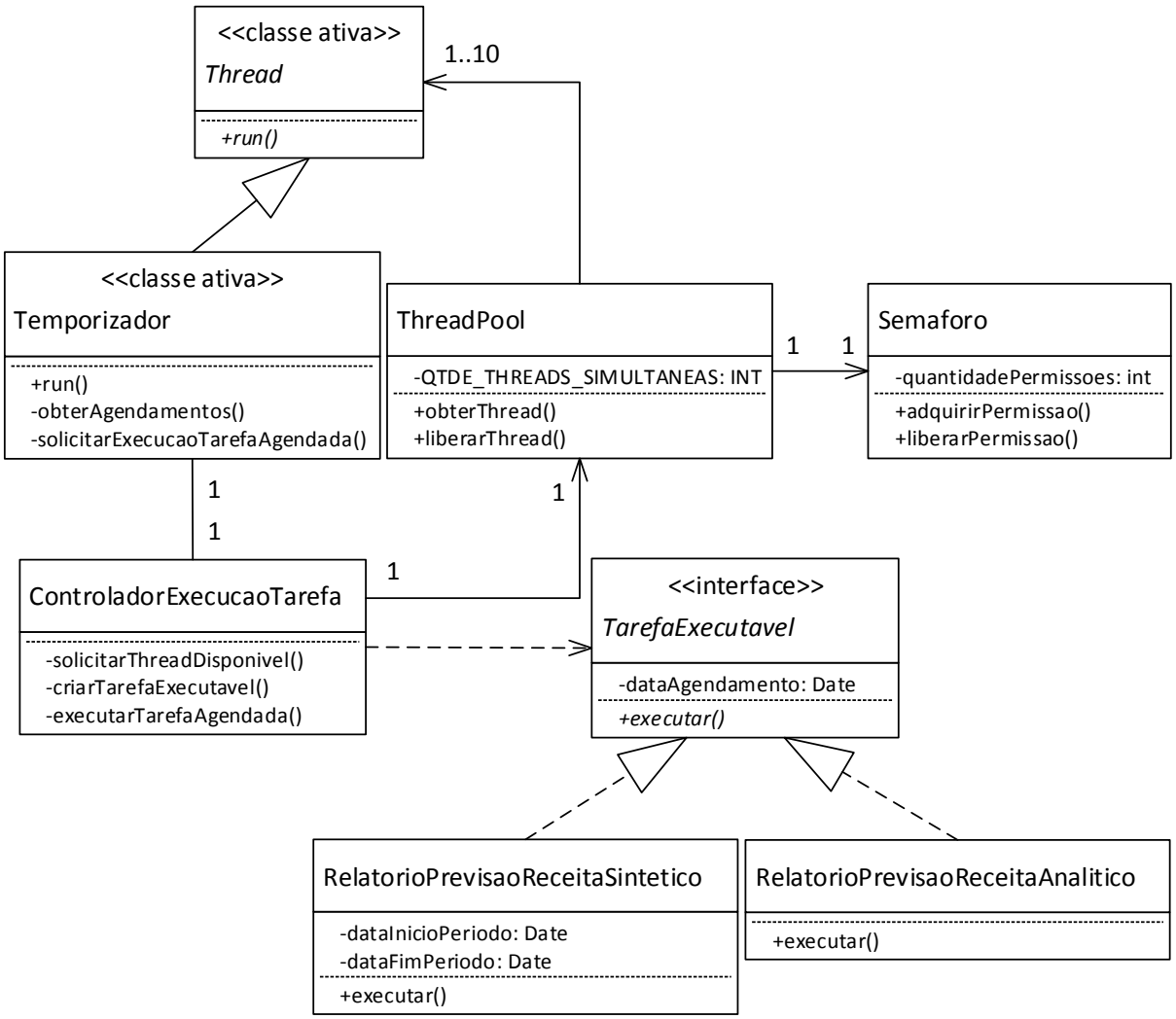


Figura 4-14 Aplicação da tática agendamento de recursos.

As táticas de agendamento de execução em horários específicos e controle do número de pilhas simultâneas proporcionam um mecanismo eficiente para que o sistema suporte números maiores de solicitações de relatórios ou outras tarefas, contribuindo para manter os tempos de latência próximos dos 2 segundos além de contribuir para uma vazão maior no que diz respeito ao número de transações por minuto conforme especificado como medida de resposta para o cenário.

5. CONSIDERAÇÕES FINAIS

5.1 Contribuições do Trabalho

A aplicação do framework de arquitetura é uma prática valiosa para padronizar as atividades de especificação, análise, elaboração e concepção de modelagens expressivas que descrevem a arquitetura, comunicando claramente as intenções inicias pelas quais o sistema é projetado e o público alvo interessado no projeto.

A atividade de modelagem das classes de domínio segregada entre os pontos de vista funcional e de informação, facilitou a produção de modelos de domínio que representam conceitos claramente distinguíveis e representando de maneira mais consistente a terminologia particular do negócio, sem negligenciar os fluxos das principais tarefas dos usuários representados pelas especificações de cenários funcionais.

No que diz respeito à descrição dos casos de uso, embora seja uma técnica importante para especificar requisitos funcionais, não é prático extrair o conhecimento mais profundo sobre o negócio de um documento de descrição de casos de uso, que usualmente é descrito em um nível alto de abstração. Além disso, eles estão associados com as funções do sistema, que tem uma influência menos significativa na arquitetura de um sistema. Embora tipicamente sejam adotados por projetos orientados à objetos, eles são independentes do uso deste paradigma de modelagem. No contexto do framework proposto, o paradigma orientado à objetos é essencial para a aplicação dos princípios de visibilidade e espaçamento ao ponto de vista de informação.

Os casos de uso foram úteis para representar a visão do usuário em alto nível e fornecer os subsídios iniciais para a especificação de alguns dos requisitos arquiteturalmente significativos. Por exemplo, a necessidade de elicitar requisitos de interoperabilidade, ocultos em uma descrição de casos de uso que possue atores representando outros sistemas, tornou-se mandatória através da declaração da regra de correspondência Reg_Fint, que governa a relação implícita entre os

requisitos de interoperabilidade e os atores do caso de uso que representam outros sistemas.

A adoção aos padrões da modelagem dirigida por domínios para aplicar os princípios de visibilidade e espaçamento ao ponto de vista de informação também demonstrou ser valiosa, na medida em que os padrões contribuíram tanto para a clareza semântica, quanto para a separação de termos distintos. O encapsulamento das restrições importantes para o negócio na forma de invariantes dentro das raízes dos agregados contribuiu para a redução de duplicidade de código e redução de impacto sobre as mudanças em tais restrições.

A especificação de regras de correspondência entre os modelos foi de grande utilidade para tornar explícito algumas das restrições que os modelos de um documento de descrição de arquitetura devem obedecer para serem consistentes. Essa técnica, além de contribuir para a redução de suposições implícitas sobre os modelos, contribuiu para o entendimento dos tipos de dependência existentes entre os modelos, eliminando uma parte da variabilidade que poderia conduzir o projeto à incompatibilidade arquitetural, conforme definido em [GARLAN-09].

Outra questão importante é a identificação apropriada dos interessados nos modelos concebidos. A prática demonstrou ser fundamental para a condução do projeto com enfoque nos interessados pelo domínio em todas as tarefas, agregando valor para o negócio.

De fato, o raciocínio sobre requisitos arquiteturalmente significativos foi eficaz através do apoio das especificações de cenários de qualidade, orientadas pelos modelos que identificam os atributos de qualidade provenientes das considerações dos interessados pelo sistema. De uma forma geral, a incorporação dos atributos de qualidade como heurísica, conforme definido em LIANPING (2013), para a descoberta de requisitos arquiteturalmente significativos demonstrou ser uma técnica bastante eficaz.

5.2 Trabalhos Futuros

As especificações de cenários de qualidade são práticas valiosas para elicitar requisitos arquiteturalmente significativos com consistência, coerência e sem ambiguidades. Embora as medidas especificadas, a princípio, sejam subjetivas, elas promovem o comprometimento e a busca por estimativas mais precisas. No entanto, é preciso experimentar essas atividades e diretrizes em outros domínios para comprovar sua verdadeira eficácia.

No que diz respeito à maturidade dos processos, ele pode servir de subsídio para as equipes que passam por auditorias ou desejam obter uma certificação de maturidade para aprimorar sua credibilidade no mercado. Nesse contexto, a incorporação de um ponto de vista de verificação e validação[31] poderia agregar valor ao estabelecer as diretrizes e os modelos de documento para elaboração de casos de teste, documentos de revisão e inspeção, contribuindo inclusive para as métricas de qualidade[32] e consequentemente para o planejamento assertivo de projetos de futuros.

Muitas das decisões tomadas durante a aplicação do framework, inclusive a aplicação de padrões, podem ser mais facilmente abordadas com a adoção a uma arquitetura de referência que, de certo modo, implica na incorporação de algumas das decisões de arquitetura, conforme relatado em CLEMENTS (2009).

Ao delegar uma parte do trabalho técnico para um framework ou arquitetura de referência, o arquiteto é livre para raciocinar em uma visão macro sobre questões estratégicas para o negócio, podendo apresentar resultados economicamente mais favoráveis para as empresas. Nesse contexto, a especificação de um framework de arquitetura corporativa utilizando as definições da norma [ISO/IEC/IEEE 42010:2011] poderia contribuir para a descoberta de novos métodos para resolver classes de problemas em um contexto mais amplo, ultrapassando os limites do software e abrangendo a corporação como um todo.

---

31 Em seu livro que detalha um processo para definição de arquiteturas, [EELES-10] atribui ao seu framework o ponto de vista de validação, designado para avaliar se o sistema contempla as funcionalidades requeridas, apresenta as qualidades necessárias e acomoda as restrições definidas.

32 Não requerido pela norma [ISO/IEC/IEEE 42010:2011].

Enfim, existe uma infinidade de opções que podem ser abordadas para adaptar o framework a um domínio específico. O mais importante é manter o foco no que efetivamente agrega valor para o negócio. Assim, as diretrizes para o desenvolvimento do framework de arquitetura são essencialmente designadas após o entendimento das principais considerações que os interessados fazem sobre o sistema para alcançar os objetivos do negócio.

REFERÊNCIAS

[BASS-12] BASS, L; CLEMENTS, P; KAZMAN, R. **Software Architecture in Practice**. 3. ed. United States of America: Addison-Wesley, 2012.

[BAUER-07] BAUER, C.; KING, G. **Java Persistence com Hibernate**. 1. ed. Revisada. Rio de Janeiro: Ciência Moderna, 2007.

[BUSCHMANN-07] BUSCHMANN, F.; HENNEY, K.; SCHMIDT, D. C. **Past, Present, and Future Trends in Software Patterns**. Biblioteca Digital de Software, IEEE, v. 24, n. 4, p. 31-37, 2007. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4267600>. Acesso em: 12 mar. 2013.

[BUSCHMANN-10a] BUSCHMANN, F. **On Architecture Styles and Paradigms.** Biblioteca Digital de Software, IEEE, v. 27, n. 5, p. 92-94, 2010. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5551019>. Acesso em: 05 mar. 2013.

[BUSCHMANN-10b] BUSCHMANN, F.; HENNEY, K. **Five Considerations for Software Architecture:** Part 1. Biblioteca Digital de Software, IEEE, v. 27, n. 3, p. 63-65, 2010. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5452148>. Acesso em: 05 mar. 2013.

[BUSCHMANN-10c] BUSCHMANN, F. **Value-Focused System Quality**. Biblioteca Digital de Software, IEEE, v. 27, n. 6, p. 84-86, 2010. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5604364>. Acesso em: 12 mar. 2013.

[BUSCHMANN-11a] BUSCHMANN, F. **Unusable Software Is Useless:** Part 1. Biblioteca Digital de Software, IEEE, v. 28, n. 1, p. 92-94, 2011. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5672524>. Acesso em: 05 mar. 2013.

[BUSCHMANN-11b] BUSCHMANN, F. **Unusable Software Is Useless:** Part 2. Biblioteca Digital de Software, IEEE, v. 28, n. 2, p. 100-102, 2011. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5720717>. Acesso em: 05 mar. 2013.

[BUSCHMANN-96] BUSCHMANN, F. et al. **Pattern-Oriented Software Architecture:** A system of patterns. England: John Wiley & Sons, 1996.

[CLEMENTS-09] CLEMENTS, P.; SHAW, M. **The Golden Age of Software Architecture Revisited.** Biblioteca Digital de Software, IEEE, v. 26, n. 4, p. 70-72, 2009. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5076462>. Acesso em: 05 mar. 2013.

[EELES-10] EELES, P.; CRIPPS, P. **The Process of Software Architecting**. 1. ed. United States of America: Addison-Wesley, 2010.

[EMERY-09] EMERY, D.; HILLIARD, R. **Every architecture description needs a framework.** Conferência sobre Arquitetura de Software na Europa, Cambridge, p. 31-40, 2009. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5290789>. Acesso em: 19 mar. 2013.

[ERL-07] ERL, T. **SOA:** Principles of Service Design. 1. ed. United States of America: Prentice Hall, 2007.

[EVANS-03] EVANS, E. **Domain Driven Design:** Atacando as Complexidades no Coração do Software. 2. ed. Rio de Janeiro: Alta Books, 2003.

[FOWLER-03] FOWLER, M. **Patterns of Enterprise Application Architecture**. 1. Ed. United States of America: Addison-Wesley, 2003.

[FOWLER-05] FOWLER, M. **UML Essencial**. 3. ed. São Paulo: Bookman, 2005.

[GAMMA-94] GAMMA, E. et al. **Design Patterns:** Elements of Reusable Object-Oriented Software. 1. ed. United States of America: Addison-Wesley, 1994.

[GARLAN-09] GARLAN, D.; ALLEN, R.; OCKERBLOOM, J. **Architectural Mismatch:** Why Reuse Is Still So Hard. Biblioteca Digital de Software, IEEE, v. 26, n. 4, p. 66-69, 2009. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5076461>. Acesso em: 05 mar. 2013.

[HOHPE-03] HOHPE, G.; WOOLF, B. **Enterprise Integration Patterns:** Designing, building, and deploying messaging solutions. United States of America: Addison-Wesley, 2003.

[IEEE-1471-2000] IEEE STANDARDS ASSOCIATION. **IEEE Recommended practice for architectural description of software-intensive systems**. 2000. Disponível em: <http://standards.ieee.org/findstds/standard/1471-2000.html>. Acesso em: 17 dez. 2012.

[ISO/IEC/IEEE 42010:2011] SYSTEM AND SOFTWARE ENGINEERING. **Architecture Descriptions in ISO/IEC/IEEE 42010.** Disponível em: <http://www.iso-architecture.org/ieee-1471/ads/>. Acesso em: 26 mai. 2013.

[ISO-25010:2011] ISO/IEC 25010:2011. **Systems and software Quality Requirements and Evaluation (SQuaRE)**. Disponível em: <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733>. Acesso em: 23 fev. 2013.

[KRUCHTEN-95] Kruchten, P. **The 4+1 View Model of architecture**. Biblioteca Digital de Software, IEEE, v. 12, n. 6, p. 42-50, 1995. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=469759>. Acesso em: 26 mai. 2013.

[LARMAN-07] LARMAN, C. **Utilizando UML e Padrões**. 3. Ed. São Paulo: Bookman, 2007.

[LIANPING-13] LIANPING, C.; MUHAMMAD, A. B.; BASHAR, N. **Characterizing Architecturally Significant Requirements.** Biblioteca Digital de Software, IEEE, v. 30, n. 2, p. 38-45, 2013. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6365165>. Acesso em: 05 mar. 2013.

[ROZANSKI-11] ROZANSKI, N.; WOODS, E. **Software Systems Architecture:** Working with Stakeholders Using Viewpoints and Perspectives. 2. ed. United States of America: Addison-Wesley, 2011.

[SCHMIDT-00] SCHMIDT, D. et al. **Pattern-Oriented Software Architecture:** Patterns for Concurrent and Networked Objects. 1. ed. England: John Wiley & Sons, 2000.

[SHAW-09] SHAW, M. **Continuing Prospects for an Engineering Discipline of Software**. Biblioteca Digital de Software, IEEE, v. 26, n. 6, p. 64-67, 2009. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5287012>. Acesso em: 05 mar. 2013.

[WALLS-08] WALLS, C.; BREIDENBACH, R. **Spring em ação**. 2. ed. Rio de Janeiro: Alta Books, 2008.