FABRICIO PAULO DE SOUZA

MÉTODO ODP ÁGIL PARA A DEFINIÇÃO DE ARQUITETURA DE SISTEMAS

Monografia apresentada a Escola Politécnica da Universidade de São Paulo para a conclusão do curso de MBA em Tecnologia da Informação

São Paulo
2013

FABRICIO PAULO DE SOUZA

MÉTODO ODP ÁGIL PARA A DEFINIÇÃO DE ARQUITETURA DE SISTEMAS

Monografia apresentada a Escola Politécnica da Universidade de São Paulo para a conclusão do curso de MBA em Tecnologia da Informação

Área de Concentração: Tecnologia da Informação

Orientador: Prof. Nelson Tanomaru

São Paulo
2013

FICHA CATALOGRÁFICA

Souza, Fabricio Paulo de
Método ODP ágil para a definição de arquitetura de sistemas / F.P. de Souza. -- São Paulo, 2013.
70 p.

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

1.Arquitetura de software 2.Métodos ágeis 3.Métodos de desenvolvimento de software I.Universidade de São Paulo. Escola Politécnica. Programa de Educação Continuada em Engenharia II.t.

Dedico este trabalho à minha família e amigos pelo apoio e compreensão ao longo desta jornada e trabalho.

AGRADECIMENTOS

Ao professor Nelson Tanomaru, pela orientação e motivação, sem o qual não haveria este trabalho.

Ao professor Jorge Luis Risco Becerra pelas discussões e experiência agregados ao trabalho.

Aos amigos e familiares que colaboraram de alguma forma na execução deste trabalho, em especial a Beatriz Ribeiro pela ajuda direta.

RESUMO

A utilização de métodos ágeis na área de Engenharia de Software tem crescido consideravelmente nos últimos tempos a fim de atender as necessidades de desenvolvimento com qualidade no menor tempo possível. Uma das principais metodologias ágil é o Extreme Programming (XP), que prima pela qualidade no desenvolvimento do software procurando atender as reais necessidades do cliente.

Nesse contexto, a Arquitetura de Software tem tido uma importância cada vez maior no auxilio para uma melhor qualidade de desenvolvimento, através da descrição dos elementos estruturais que compões o sistema.

Dessa forma, o principal objetivo deste trabalho foi o estabelecimento de um processo ágil de desenvolvimento de software aplicando as vantagens da Arquitetura de Software, através do relacionamento do método ágil de desenvolvimento XP com o padrão RM-ODP para Arquitetura de Software, criando assim o Método ODP Ágil para a definição de Arquitetura de Sistemas.

Como uma forma de comprovar o seu funcionamento o Método ODP Ágil foi aplicado em um experimento dentro de uma Instituição Financeira.

Palavras-chave: RM-ODP, XP, Arquitetura de Software, Métodos Ágeis.

ABSTRACT

Recently the use of agile methods in the field of Software Engineering has grown considerably in order to meet the development needs with quality levels in the shortest possible time. One of the main agile methodologies is the Extreme Programming (XP) which strives for quality in software development seeking to meet the real needs of the customer.

In this context, Software Architecture has had a growing importance in the helping to better quality of development through the description of the structural elements that compose the system.

Thus, the main objective of this study is to establish an agile software development process by applying the benefits of Software Architecture through the relationship of XP Agile Development Method with the RM- ODP standard for Software Architecture, resulting in the creation of the ODP Agile Method for the definition of System Architecture.

As a way to check its operation we will apply the ODP Agile Method in an experiment within a Financial Institution.

Keywords: RM-ODP, XP, Software Architecture, Agile Methods.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ANA	Agência Nacional de Águas
BC	Banco Central
BPMN	Business Process Modeling Notation
ER	Modelos Entidade-Relacionamento
ESRI	Environmental Systems Research Institute
IEC	International Electrotechnical Commission
ISO	International Organization Standardization
ITU	International Telecommunication Union
ODP	Open Distributed Processing
RM-ODP	Reference Model for Open Distributed Processing
SPEM	Software Process Engineering Metamodel
UML	Unified Modeling Language
XP	Extreme Programming

SUMÁRIO

1 INTRODUÇÃO

Neste capitulo, apresentam-se as considerações iniciais, o objetivo, a motivação, a abrangência, a metodologia e a estrutura do trabalho. A consideração inicial apresenta o contexto da pesquisa e o problema a ser tratado. O objetivo descreve as metas pretendidas. A motivação traz contribuições acadêmicas que fundamentam o desenvolvimento. A abrangência define os limites tanto acadêmicos quanto práticos desta pesquisa. A metodologia apresenta as atividades de pesquisa definidas para a monografia. A estrutura do trabalho descreve como a monografia será apresentada.

1.1 Considerações iniciais

A indústria de software vem passando por diversas transformações nos últimos tempos, entre elas o desafio de desenvolver softwares de qualidade no menor tempo possível atendendo as necessidades dos clientes.

Com o intuito de atender esses desafios, observa-se uma crescente do uso de métodos ágeis na área de Engenharia de Software. Entre as características destas metodologias está atender as reais necessidades do cliente, na velocidade e praticidade por ele desejada. Uma destas metodologias de desenvolvimento ágeis mais utilizadas é o Extreme Programming (XP), que prima pela qualidade do software desenvolvido atendendo as reais necessidades do cliente no prazo por ele definido.

Por outro lado, a dificuldade de desenvolvimento destes sistemas é um ponto crítico devido à complexidade dos sistemas serem cada vez maior. Nesse cenário entra a área de Arquitetura de Software, que tem tido uma importância cada vez maior devido a sua colaboração na qualidade de software, através da descrição dos elementos estruturais que compõe o sistema, seus relacionamentos e propriedades (Bass, Clements e Kazman, 2003). Segundo Bass, Clements e Kazman (2003), a arquitetura reúne um conjunto de requisitos de negócios e técnicos, com influências de diversas áreas do negócio, dando apoio ao desenvolvimento do sistema para que atinja seus objetivos. Como uma forma de auxilio à arquitetura tem-se o padrão RM-ODP, que tem como característica a implementação de sistemas Open Distributed Processing (ODP) de forma consistente e confiável através de um conjunto de

pontos de vista, cada qual refletindo um conjunto de atributos diferentes (Gungi, 1995).

Considerando-se este cenário, observa-se então que as áreas de pesquisa de Arquitetura de Software e Métodos Ágeis possuem objetivos em comum visando o aumento da qualidade e produtividade.

1.2 Objetivo

O objetivo principal deste trabalho é propor um método prático e confiável de desenvolvimento de software que explora as vantagens de métodos ágeis e de um padrão para definição da Arquitetura de Sistemas.

Para isto, estabeleceu-se um relacionamento entre a definição da arquitetura de sistemas através do padrão RM-DP e o método de desenvolvimento ágil Extreme Programming (XP). Com esse relacionamento pretende-se identificar as competências e deficiências de cada uma a fim de integra-las e como resultado se obter um novo método chamado Método ODP Ágil.

1.3 Motivação

Levando-se em conta a grande preocupação com arquitetura de sistema no contexto de métodos ágeis de desenvolvimento de software, várias abordagens têm surgido ultimamente. Entretanto, essas pesquisas não abordam com a devida importância a definição e utilização de arquiteturas de sistemas em métodos ágeis, e é nesse ponto que este trabalho tem o seu foco: os benefícios da definição da arquitetura de sistema no contexto de métodos ágeis.

Entre os benefícios, destacam-se a melhoria da qualidade do software desenvolvido e o aumento da produtividade relacionada ao reuso do conhecimento documentado no desenvolvimento de software através de métodos ágeis.

A motivação para o experimento (Gestão de Contratos Financeiros) se dá pela intenção de comprovar que a utilização do Método ODP Ágil é viável.

Entre os trabalhos correlatos publicados, foi possível identificar os seguintes autores e suas contribuições para esta monografia:

a) Galasini (2004) propõe um modelo ODP-UP para definição de arquiteturas distribuídas. Este modelo é baseado no relacionamento de duas abordagens arquiteturais: Metodologia ODP e o Processo Unificado.

Nesta monografia será utilizada a ideia principal de Galasini, de criação de um método para definição de arquiteturas distribuídas a partir da integração de dois recursos distintos, porém no caso desta monografia será abordado o método de desenvolvimento ágil de software XP no lugar do recurso Processo Unificado.

b) Sampaio (2004) propõe um processo XWEBPROCESS ágil para o desenvolvimento de aplicações Web. Tem como foco a construção de sistemas Web de qualidade de forma eficiente e rápida, baseando-se no método de desenvolvimento ágil de software XP.

O Método ODP Ágil propõe a integração do padrão RM-ODP com o método de desenvolvimento ágil de software XP. Nesta monografia foi feita a modelagem por completo do método XP por Sampaio que será utilizada na integração. Esta modelagem é apresentada através de diagramas de classe estereotipados criados através de uma analise interpretativa da literatura de XP. Será em cima desta modelagem que será relacionado o padrão de abordagem arquitetural RM-ODP com o método de desenvolvimento ágil de software XP para a criação do Método ODP Ágil para definição da arquitetura de sistemas.

c) Zani (2013) propõe um processo ágil de desenvolvimento de software, denominado AGIRA, que explora as vantagens do uso de arquiteturas de referência.

Nesta monografia será utilizada como referência a forma de definição da arquitetura de sistema dentro de um processo de desenvolvimento de software. Zani define as arquiteturas que compõe a arquitetura de sistemas e como cada uma delas deve ser definida dentro de uma organização (quais artefatos e seus relacionamentos, por exemplo).

Por fim, objetivou-se contribuir para as áreas de Arquitetura de Software com um método ágil de software com a definição da arquitetura do sistema

1.4 Abrangência

Este trabalho aborda a aplicação dos conceitos dos pontos de vista do padrão RM-ODP nas etapas do método XP com o objetivo de criar um método ágil de desenvolvimento de software para sistemas distribuídos. É importante esclarecer que este trabalho esta focado nas etapas de definição de arquitetura e não nas etapas de programação, já bem definidas pelo método XP.

Este trabalho aborda também uma aplicação prática do Método ODP Ágil para um sistema de Gestão de Contratos em uma instituição financeira de âmbito nacional. Esse experimento foi escolhido a fim de comprovar que a utilização do Método ODP Ágil é viável.

1.5 Metodologia da pesquisa

A metodologia da pesquisa se refere à forma que foi desenvolvida essa monografia e seus procedimentos científicos, conforme abaixo:

1. Definição do referencial teórico: nesta fase foi feita uma pesquisa e coletadas as informações relevantes ao tema do trabalho. Foram analisados os conceitos existentes e como poderiam fornecer os fundamentos teóricos necessários à definição do Método ODP Ágil;

2. Estruturação da metodologia: esta etapa consiste na criação do método e seu guia de aplicação contendo as atividades e sua sequência fundamentada nas teorias estudadas;

3. Execução e análise do experimento: esta fase aplica a metodologia proposta na elaboração da arquitetura do sistema de Gestão de Contratos de uma instituição financeira;

4. Elaborar a conclusão: a conclusão é baseada na avaliação da metodologia proposta, levando em consideração o experimento e as referências teóricas

utilizadas para fundamentar esse método. A conclusão inclui também a indicação de trabalho futuros complementares a este trabalho de pesquisa.

1.6 Estrutura do trabalho

Este trabalho esta estruturado como segue:

- No Capitulo 1 tem-se a introdução ao tema, com as considerações iniciais e apresentação dos objetivos do trabalho, bem como as motivações, a abrangência, a metodologia da pesquisa e a estrutura do trabalho.
- No Capitulo 2 apresentamos os fundamentos teóricos envolvidos no trabalho, ou seja, apresenta os fundamentos do padrão RM-ODP e o método ágil de desenvolvimento de software XP. Esses fundamentos serão combinados para compor a proposta do método apresentado no capitulo subsequente.
- No Capitulo 3 é apresentada a proposta do trabalho. É o detalhamento do Método ODP Ágil, sua forma estrutural e seus fluxos de trabalho.
- No Capitulo 4 apresenta o experimento onde será colocado em prática o Método ODP Ágil, cujo contexto é a Gestão de Contratos Financeiros.
- No Capitulo 5 tem-se a conclusão onde são apresentadas as contribuições do trabalho, a conclusão obtida pela elaboração deste trabalho e ainda possíveis trabalhos futuros que podem ser desenvolvidos motivados por este.

2 FUNDAMENTAÇÃO CONCEITUAL

Este capítulo tem como objetivo apresentar os conceitos utilizados para a criação do Método ODP Ágil através de um conceito e um padrão já difundidos na indústria do desenvolvimento de software: o padrão RM-ODP e o método ágil de desenvolvimento de software XP. Também será apresentada o meta-modelo SPEM, utilizado para descrever os processos de desenvolvimento de software do método ágil XP.

2.1 O padrão RM-ODP

O padrão RM-ODP é um padrão internacional ISO/IEC 10746 criado pela International Organization Standardization (ISO) em conjunto com a International Electrotechnical Commission (IEC) e a International Telecommunication Union (ITU) que propõe um *framework* para facilitar o entendimento e construção de sistemas abertos e distribuídos e prover suporte para a integração da distribuição, interoperabilidade e portabilidade, segundo a norma ISO (98).

É um *framework* estruturado em pontos de vista que ao invés de trabalhar com toda a complexidade de um sistema distribuído de forma unificada, tem por objetivo observa-lo por diferentes pontos de vista, cada um com características e complexidades diferentes. Ele é baseado nos conceitos derivados das pesquisas e desenvolvimentos na área de processamento distribuído.

Ele é constituído de quatro partes distintas:

- A primeira parte apresenta uma visão geral do padrão e os conceitos de sistemas distribuídos;
- A segunda parte consiste nos documentos com terminologias padronizadas fundamentais para sistemas distribuídos;
- A terceira parte consiste em um documento de arquitetura que define os pontos de vista para arquiteturas com regras estruturais claras;
- A quarta parte provê uma descrição matematicamente formal da segunda e terceira partes.

O padrão RM-ODP provê práticas de projeto para a definição de arquiteturas de sistemas distribuídos, sendo adequadas a vários tipos de negócio.

Neste trabalho será utilizada a terceira parte, onde se tem a definição dos pontos de vista para arquiteturas de sistemas distribuídos.

2.1.1 Motivações para o seu uso

A abordagem utilizada pelo padrão RM-ODP é a de que os objetivos e regras de negócio são os responsáveis para se criar um sistema. Por isso o *framework* consiste na definição do escopo do negócio, seus objetivos e regras para orientar a evolução da arquitetura do sistema. Sendo assim, a especificação de cada um destes elementos proporciona a especificação da arquitetura do sistema e reforça a condição de que um sistema só existe para suportar um negócio.

A figura 1 apresenta como os objetivos e regras de negócio são os responsáveis por guiar as necessidades e motivações para a construção de um sistema.

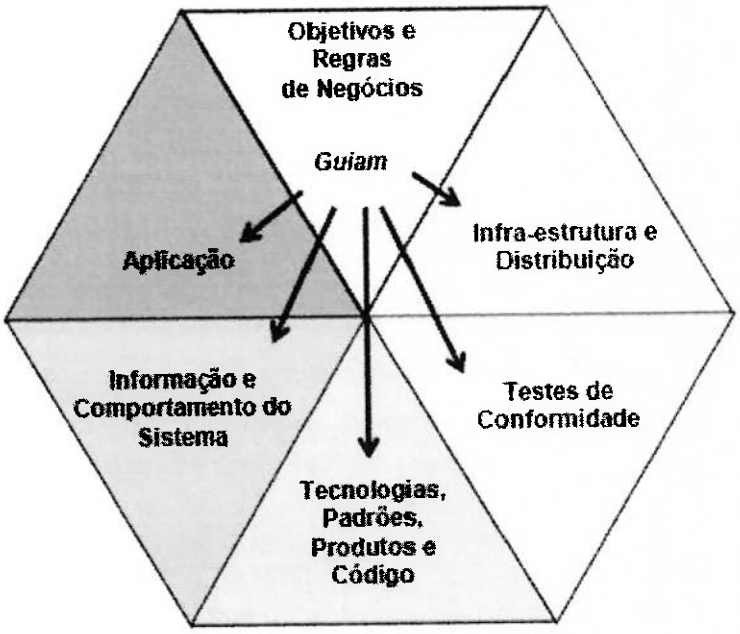


Figura 1 – Elementos da arquitetura do RM-ODP. Fonte Galasini (2004).

Segundo Gungi (1995), o padrão RM-ODP estabelece uma base para a implementação de sistemas, e por isso ele deve:

- Prover um modelo consistente que possa ser mantido entre múltiplos padrões desenvolvidos separadamente;
- Definir as áreas em que o padrão RM-ODP é necessário, assim como o relacionamento entre os padrões ODP e outros padrões;
- Descrever um conjunto mínimo de ferramentas a serem usadas na definição do modelo (visão geral, guia de uso, modelo descritivo, modelo prescritivo e semântica arquitetônica);
- Prover um conjunto consistente de conceitos e terminologias comuns.

2.1.2 Benefícios de sua utilização

Segundo Galasini (2004), os benefícios da utilização do padrão RM-ODP passam pela melhor definição do negócio envolvido e pela melhor definição dos aspectos de distribuição do sistema. Pretende-se com isto reduzir os custos do projeto, assim como seus prazos e o aumento da eficiência no seu desenvolvimento.

Para isto, é importante obter a aderência entre as arquiteturas obtidas a partir dos pontos de vista do modelo.

2.1.3 Pontos de vista do padrão RM-ODP

O padrão RM-ODP trata de vários aspectos e características de diversos sistemas e para isso ele possui cinco abstrações diferentes (pontos de vista), a partir das quais os sistemas distribuídos são modelados.

Essas abstrações partem do pressuposto que um problema complexo pode ser melhor entendido e solucionado se dividido em problemas menores. Os pontos de vista servem exatamente para isto: dividir um problema complexo em cinco visões distintas, independentes entre si, porém complementares, cada qual absorvendo um conjunto de atributos diferentes.

Segundo Gungi (1995), um ponto de vista leva a uma representação do sistema com ênfase em um determinado assunto, ou seja, um ponto de vista reconhece algumas distinções (relevantes para o seu nível em questão) e ignora outras (não relevantes para o seu nível em questão).

Os cinco pontos de vista são: Empresarial, Informação, Computacional, Engenharia e Tecnológico. A figura 2 apresenta os pontos de vista e a independência deles aos aspectos de distribuição e implementação, assim como os principais assuntos de cada ponto de vista ao lado. Segundo Galasini (2004), os pontos de vista de Empresa, Informação e Computação são independentes de aspectos físicos que envolvem distribuição e o ponto de vista de Engenharia não deve sofrer influências de aspectos de tecnologia. Ao lado de cada ponto de vista observam-se seus respectivos assuntos a serem desenvolvidos.

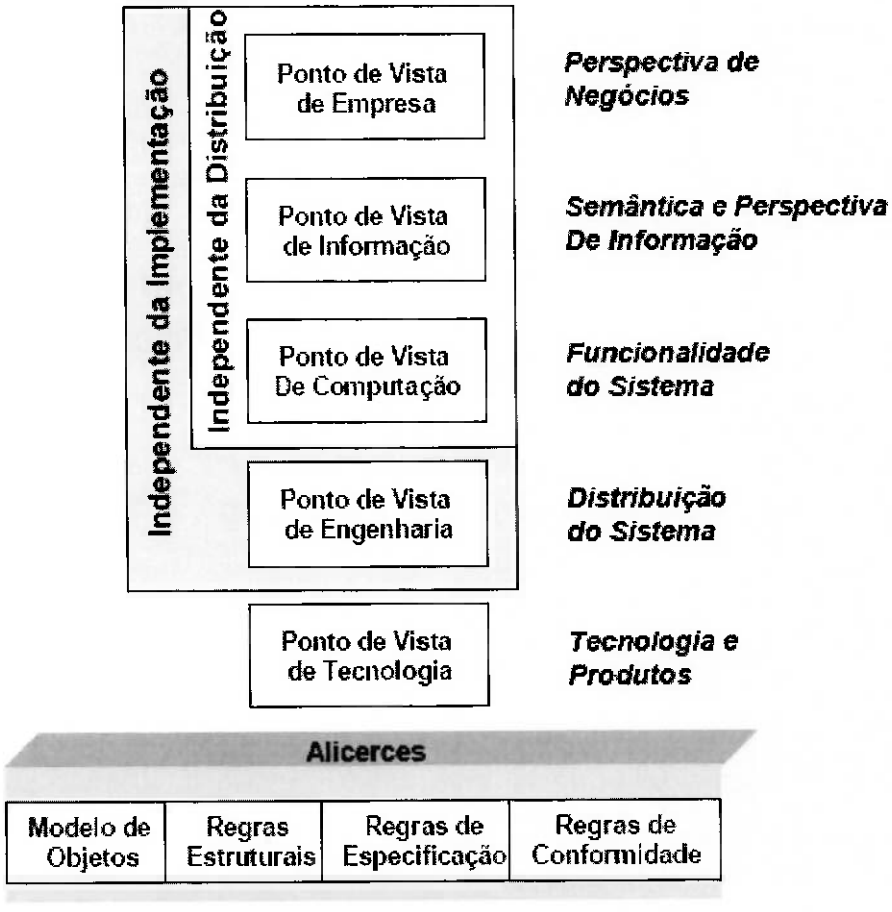


Figura 2 – Modelo arquitetural do padrão RM-ODP. Fonte: Galasini (2004).

Ainda segundo Galasini (2004), todos os pontos de vista utilizam de alicerces básicos do padrão RM-ODP, que são:

- Modelo de objetos: a modelagem de objetos no padrão RM-ODP utiliza-se dos conceitos de objetos e suas interações, levando-se em conta identidade, encapsulamento, herança, polimorfismo e instanciação;

- Regras estruturais: visa à organização de objetos, propriedades do sistema, contexto de nomes, comportamento e gerenciamento de ações;
- Regras de especificação: visam a composição, decomposição, componentização, tipos de dados, classes, objetos, herança, template, deleção, invariantes, rastreabilidade, interfaces, compatibilidade de comportamento, refinamento, instanciação, criação, pré-condição e pós-condição;
- Regras de conformidade: correspondência entre os pontos de vista, visando obter integridade conceitual em cada ponto de vista.

Na sequência será detalhada a descrição dos pontos de vista baseada no padrão ISO/IEC 10746 que propõe uma abordagem do sistema por meio destes pontos de vista para o desenvolvimento de sistemas de informações.

2.1.3.1 Ponto de vista Empresarial

O ponto de vista Empresarial define os requisitos básicos do sistema considerando papéis, politicas de negócio e de gerenciamento da empresa assim como as necessidades dos usuários. Com isso, o sistema é modelado em termos dos domínios envolvidos, funcionalidade requerida, os atores e seus papéis.

2.1.3.2 Ponto de vista de Informação

O ponto de vista da Informação descreve os recursos que apoiam os requisitos do ponto de vista Empresarial. Tem como foco o modelamento da informação, provendo uma visão cobrindo as fontes da informação, os destinos e o fluxo da mesma.

2.1.3.3 Ponto de vista Computacional

O ponto de vista Computacional tem como enfoque os algoritmos e estruturas de dados e a definição dos requisitos de transparência e distribuição. É tratado através da decomposição funcional do sistema e da distribuição das funções realizadas dentro do mesmo.

2.1.3.4 Ponto de vista de Engenharia

O ponto de vista de Engenharia define os aspectos físicos de distribuição do sistema complementando o ponto de vista Computacional. É nele que se define as necessidades de comunicação, desdobramento das funcionalidades e as características de desempenho, interconectividade e distribuição.

2.1.3.5 Ponto de vista Tecnológico

O ponto de vista Tecnológico define as tecnologias e produtos a serem utilizados pelo sistema. Este ponto de vista deve demonstrar qual o software e hardware que devem ser utilizados, assim como a configuração, instalação e manutenção dos mesmos.

2.1.4 O uso dos pontos de vista no ciclo de vida de um sistema

Segundo Gungi (1995), os pontos de vista do padrão RM-ODP devem ser utilizados ao longo de todo o ciclo de vida de um sistema. A figura 3 mostra o uso dos pontos de vista ao longo do ciclo de vida de um sistema.

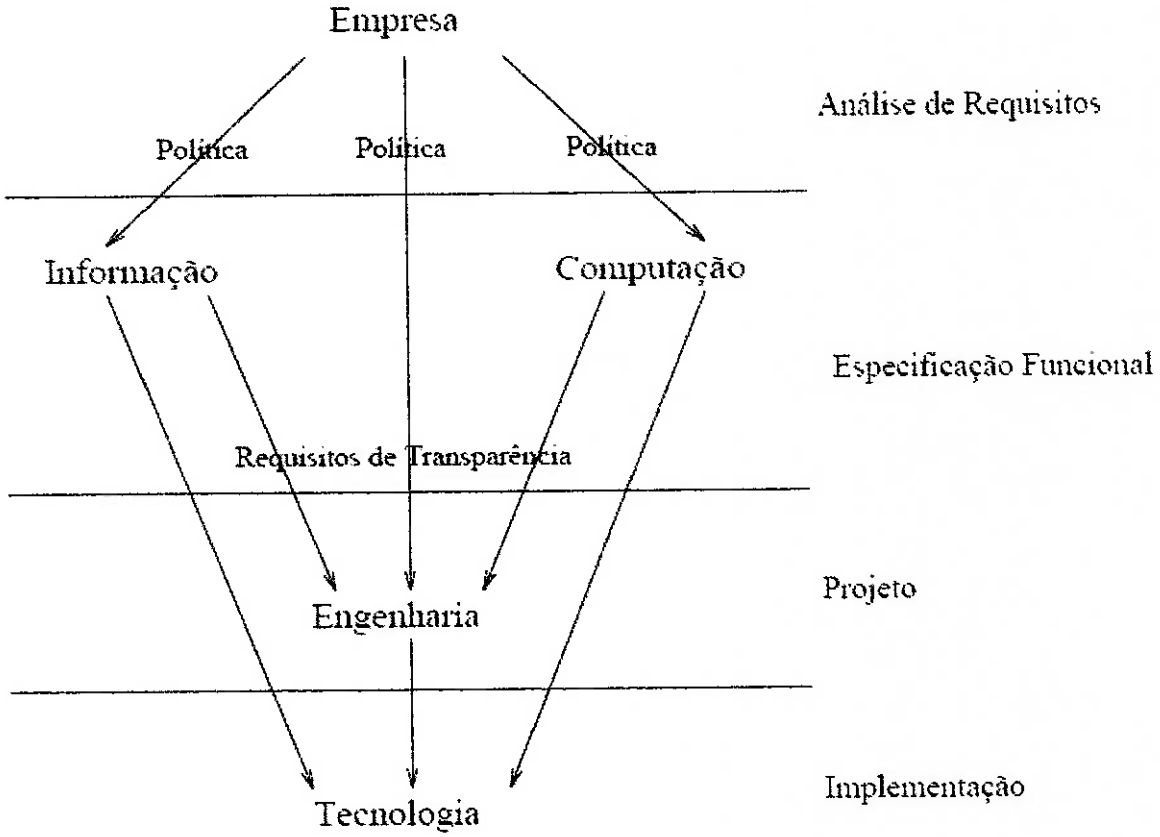


Figura 3 – Uso dos pontos de vista no ciclo de vida. Fonte: Gungi (1995).

O ponto de vista Empresarial leva a analise de requisitos e politicas do sistema. Já os pontos de vista de Informação e Computacional fornecem documentos válidos para a fase de especificação funcional. O ponto de vista de Engenharia permite a especificação das funções de processamento, armazenamento e comunicação necessárias para a implementação do sistema. Por fim, no ponto de vista Tecnológico deverão ser formulados requisitos de implementação, identificadas as tecnologias a serem usadas, casos de teste, etc.

2.2 Software Process Engineering Metamodel (SPEM)

O SPEM é um meta-modelo que pode ser utilizado para descrever um processo de desenvolvimento de software. Ele utiliza UML como notação e adota uma abordagem orientada a objetos e foi utilizada neste trabalho na modelagem do processo XP, criada por Sampaio (2004).

Segundo Sampaio (2004), o objetivo do SPEM é abranger os processos de desenvolvimento de software já existentes e não exclui-los devido a sua complexidade como excesso de elementos e restrições.

A linguagem SPEM disponibiliza algumas representações e estereótipos para modelar seus principais elementos em diagramas UML. A tabela 1 apresenta um resumo dos principais elementos de SPEM, seus conceitos e suas representações gráficas.

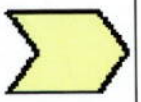
Estereótipo	Comentário	Notação
Artefato	É uma descrição de algo que contém informação ou é uma entidade física produzida ou usada por atividades do processo.	
Conjunto de trabalho	É um elemento do modelo que descreve a execução, as operações realizadas e as transformações feitas nos Artefatos. Representa um conjunto de atividades, como, por exemplo: especificar requisitos e fazer projeto do sistema.	
Guia	É um elemento do modelo que se associa a outros elementos e pode conter descrições adicionais, tais como técnicas, guias e templates.	
Atividade	É um conjunto de trabalho que descreve o que um papel realiza. A diferença para Conjunto de trabalho é sutil mas pode ser entendida como sendo uma única atividade bem delimitada, e não um conjunto, de responsabilidade de um determinado papel.	
Papel	Descreve os papéis, responsabilidades e competências que um determinado indivíduo tem dentro do processo.	
Disciplina	É um agrupamento coerente de elementos do processo (artefatos, papéis, atividades) cujas atividades são organizadas segundo algum ponto de vista ou tema comum.	

Tabela 1 – Descrição de alguns elementos de SPEM. Fonte: Sampaio (2004).

2.3 Extreme Programming (XP)

Segundo Soares (2004), a Extreme Programming (XP) é uma metodologia Ágil para equipes médias que desenvolvem software baseado em requisitos vagos e que se modificam rapidamente. Ela veio suprir as necessidades da indústria de software em desenvolver softwares com qualidade, no menor tempo possível e que atendam as necessidades dos clientes.

A proposta de iterações curtas do método de XP surgiu como uma alternativa as deficiências dos processos de desenvolvimento existentes, sendo que nesse contexto pode-se citar o estudo de Beck (1999), que revela o fato de que ciclos grandes de desenvolvimento de software não permitem mudanças rápidas e que o custo dessas mudanças crescia de acordo com o tempo do projeto.

Na figura 4 pode-se ver a evolução dos modelos: o modelo tradicional (cascata) com seus longos ciclos, passando pelo modelo iterativo e seus ciclos de tempo médio até chegar ao modelo XP com seus ciclos curtos.

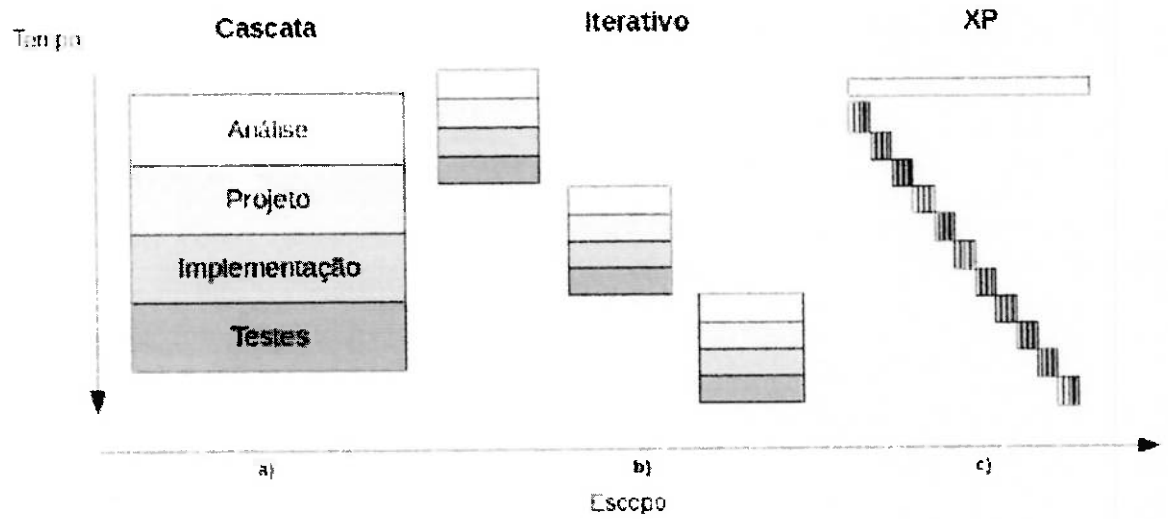


Figura 4 – Modelo Cascata (a); Modelo Iterativo (b); e XP (c). Fonte Beck (1999).

Segundo Kuhn e Pamplona (2004), o método XP busca o máximo de valor de cada dia de trabalho e em um curto espaço de tempo o cliente terá um produto que possa ser utilizado, podendo aprender e avaliar se foi realmente desenvolvido o que ele deseja a cada ciclo.

A metodologia XP aborda somente ações de desenvolvimento de software. Portanto, não fazem parte do seu escopo áreas como gestão de portfolio, gestão financeira, operação de produto, marketing e vendas.

2.3.1 Práticas, Valores e Princípios

Segundo Zani (2013), o método XP esta fundamentado em três grandes alicerces: práticas, valores e princípios. Práticas estão relacionadas com técnicas que são utilizadas por indivíduos sendo claras e objetivas, sendo assim extremamente benéficas. Valores são subjetivos, sendo a causa de haver inclinação para certos hábitos. Os valores devem ser explicitados para que as práticas não caiam em

desuso. Já os princípios fazem a união entre valores e práticas, sendo diretrizes específicas de domínio. Os princípios apresentam diretrizes especificas de como lidar com os desenvolvedores de uma equipe, sendo sustentados pelos valores e pelas práticas. Eles auxiliam na escolha de alternativas para a solução de problemas durante o desenvolvimento.

Segundo Kuhn e Pamplona (2004), o método XP adota quatro valores como critérios para as pessoas envolvidas no desenvolvimento do sistema, que são

- Comunicação: no método XP a comunicação ocorre de forma mais direta e eficaz, oferecendo agilidade aos assuntos. Como forma de evitar qualquer mal-entendido, é recomendado o contato direto entre o cliente e desenvolvedor;

- Simplicidade: no método XP é adotado que cada membro da equipe adote a solução mais simples que possa funcionar em que o custo para mudança no futuro seja baixo evitando assim a construção antecipada de funcionalidades que acabam muitas vezes nem sendo usadas;

- Feedback: o cliente conduz o desenvolvimento através do seu feedback da sua reavaliação do produto recebido. Ë um valor importante pois possibilita que as pessoas aprendam cada vez mais sobre o sistema, corrigindo falhas e assim melhorando o mesmo;

- Coragem: é necessária para que se aplique a metodologia de forma correta, trazendo melhorias ao projeto. É preciso ter coragem para, entre outras coisas, manter o sistema simples, fazer desenvolvedores trabalharem em pares, propor a adoção de um processo novo e implantar uma nova versão do sistema no cliente semanalmente.

Segundo Sampaio (2004), os princípios fundamentais são:

- Feedback rápido: a ideia do método XP é que os participantes do projeto estejam sempre se comunicando para facilitar assim o aprendizado sobre o projeto e também para solucionar rapidamente dúvidas, riscos e problemas;

- Assumir simplicidade: todos os problemas devem ser resolvidos da forma mais simples possível e deve-se confiar na capacidade de adicionar complexidade no futuro caso necessária;

- Mudança incremental: as mudanças devem ser incrementais e feitas aos poucos a fim de se obter um bom resultado;

- Abraçando mudanças: no método XP mudanças são sempre bem vinda independente da fase do projeto e para isso procura facilitar o ato de incluir alterações através de princípios e práticas;

- Trabalho de qualidade: qualidade no método XP é um critério obrigatório. A metodologia XP trata qualidade no sentido de que um sistema atenda aos requisitos do cliente, rodando 100% dos testes com sucesso e agregando o maior valor possível para o negócio do cliente.

As práticas são conjunto de atividades que deverão ser seguidas pelas equipes de desenvolvimento e devem ser claras e objetivas. Elas foram criadas para serem utilizadas em conjunto de forma a reduzir as fraquezas umas das outras. Segundo Beck (1999), as mais importantes são:

- Sentar juntos: a equipe de desenvolvimento deve estar próxima fisicamente para que haja sucesso no projeto;

- Equipe completa: diversificar os membros da equipe de acordo com suas capacidades técnicas;

- Ambiente de trabalho informativo: deve-se organizar o ambiente de modo a ter um painel sempre visível com informações ativas para que se possa identificar o projeto e seu estado geral rapidamente;

- Trabalho energizado: estimular a equipe a trabalhar o número de horas que sejam produtivas pois o excesso de trabalho leva a uma queda de rendimento prejudicando a qualidade do desenvolvimento;

- Programação em pares: fazer com que dois membros da equipe compartilhem uma estação de trabalho para que possam discutir soluções,

analisar o código e chegar as melhores soluções para os problemas. Com isso tem-se um ganho no conhecimento e melhor qualidade no desenvolvimento;

- Histórias: o planejamento deve ser feito pelo cliente através de histórias, que devem ser descritas e estimadas em um cartão;

- Ciclo semanal: o planejamento deve ser feito semanalmente, durante uma reunião no inicio da semana que descreverá o que foi feito até aquele momento e definirá o que deve ser feito na semana atual de acordo com a seleção feita pelo cliente. O objetivo do ciclo semanal é a entrega de software funcional, passível de utilização e implantação no cliente;

- Ciclo trimestral: o trabalho deve ser planejado em trimestres, através de objetivos maiores. Em uma reunião devem-se identificar os impedimentos, o que será desenvolvido, iniciar reparos e selecionar os temas para o trimestre;

- Folga: os integrantes da equipe devem se sentir confortáveis para o desenvolvimento e para isso devem passar uma estimativa realista para os superiores, minimizando a tensão dentro da equipe;

- Construção em dez minutos: a compilação do sistema deve ser concluída em no máximo 10 minutos, para que assim seja estimulado o seu uso. Otimizações devem ser realizadas de modo a diminuir o tempo de compilação caso necessárias;

- Integração continua: busca-se automatizar o processo de integração e construção a fim de que todas as mudanças realizadas sejam testadas assim que submetidas ao repositório;

- Testes constantes: o desenvolvimento dos testes é realizado antes do desenvolvimento do código da funcionalidade em si, melhorando a coesão e minimizando o acoplamento de código;

- Projeto incremental: o projeto do sistema deve ser visto diariamente para garantir que seu proposito esteja sendo atendido de acordo com as necessidades do negócio;

- Jogo do planejamento: o planejamento de uma release e das iterações é feitos com base nas histórias e conta com a colaboração de toda a equipe, inclusive do cliente;

- Projetos simples: esta prática enfatiza que o projeto deve se concentrar em soluções simples e bem estruturadas para os problemas de hoje e que não deve perder tempo procurando soluções genéricas que procurem atender a funcionalidade futuras;

- Refatoramento: são melhorias constantes no projeto para aumentar a capacidade de mudanças do mesmo;

- Código de propriedade coletiva: procura encorajar a equipe inteira a trabalhar mais unida em busca de qualidade no código fazendo melhorias e refatoramentos em qualquer parte do código a qualquer momento;

- Padrões de codificação: tem como objetivo que todos programem da mesma forma, facilitando o entendimento do código e as alterações.

2.3.2 Papéis e Responsabilidades

Segundo Zani (2013), os principais papéis e responsabilidades em uma equipe que utiliza o método XP são:

- Programador: tem como responsabilidade a estimativa do esforço das histórias, a escrita dos testes e o desenvolvimento do código de forma mais simples possível;

- Arquiteto: tem como responsabilidade observar e refatorar partes do código que estejam impactando no sistema, escrever testes para validar a arquitetura do sistema e auxilia com sua clara visão na seleção das histórias a serem desenvolvidas;

- Cliente: tem como responsabilidade a elaboração das histórias, o acompanhamento do desenvolvimento para que lhe satisfaça e definir as prioridades dos requisitos;

- Testador: tem como responsabilidade a escrita dos testes funcionais, executar os testes e comunicar os resultados;
- Treinador: é o responsável por orientar os membros da equipe a manter o processo como um todo, seguindo suas atividades;
- Gerente de Projeto: tem como responsabilidade verificar e comunicar o andamento do projeto, tomar decisões e auxiliar na comunicação entre equipe e cliente;
- Gerente de Produto: tem a responsabilidade de elaborar as histórias, selecionar as histórias a serem desenvolvidas e atua no detalhamento dos requisitos.

2.3.3 Processo

Nesta seção será apresentado o modelo do processo XP definido por Sampaio (2004). A figura 5 mostra um diagrama que representa a dinâmica de um projeto realizado seguindo o processo XP.

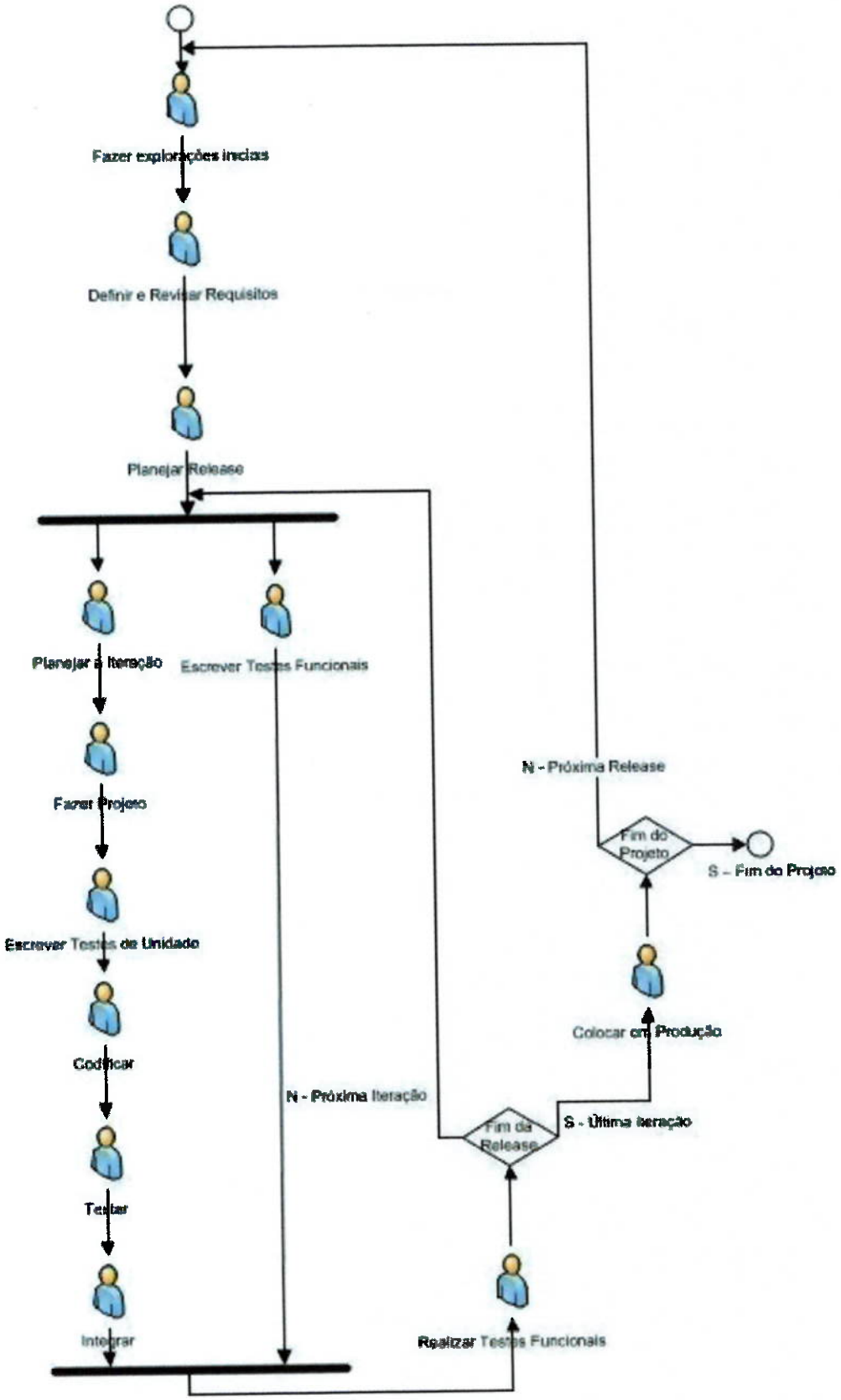


Figura 5 – Modelo do processo XP.

Seguindo a definição de Sampaio (2004), serão agora detalhadas as atividades envolvidas em cada uma das etapas do processo XP.

2.3.3.1 Fazer explorações iniciais

É onde são feitas as primeiras explorações e investigações sobre o projeto. Conforme mostrado na figura 6, os clientes são consultados e incorporados à equipe para realizar a definição prévia dos requisitos e do escopo do sistema, gerando como artefato uma prévia dos User Stories. Os programadores realizam experimentos com a possível tecnologia e infraestrutura para verificar a viabilidade tecnológica da solução e elaborar uma primeira ideia de arquitetura para o sistema (metáfora). É importante levantar as informações necessárias para decidir se o projeto é viável ou não.

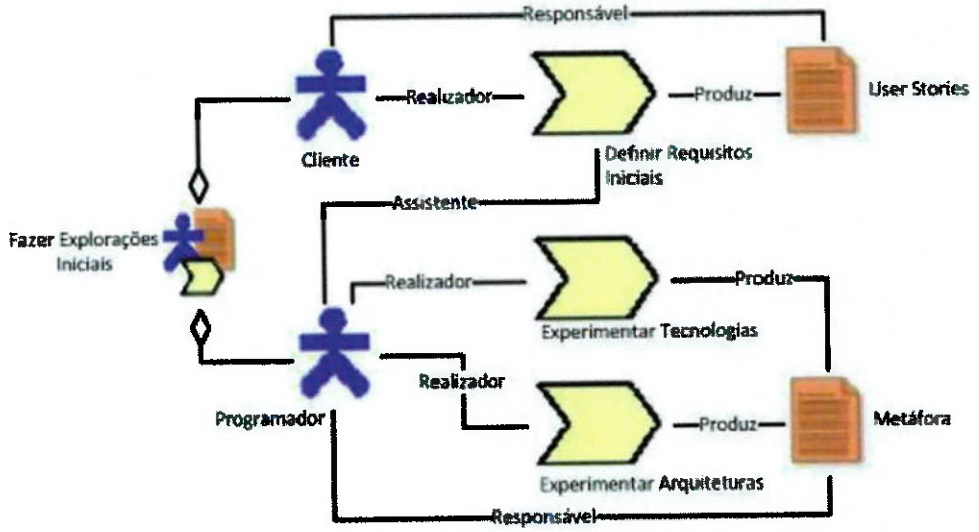


Figura 6 – Diagrama de classe estereotipado da etapa Fazer Explorações Iniciais. Fonte: Sampaio (2004).

2.3.3.2 Definir e revisar requisitos

Esta etapa é responsável por definir e revisar os requisitos que farão parte do release subsequente do sistema. Esses requisitos podem sofrer alterações a qualquer momento e por isso serão revistos na etapa que trata de requisitos na iteração. Seguindo o diagrama da figura 7, o cliente é o responsável por escrever e priorizar as histórias, auxiliado pela equipe de desenvolvimento (programadores).

Como resultado terá os cartões de história (User Stories), que contém a descrição das funcionalidades e as estimativas de esforço.

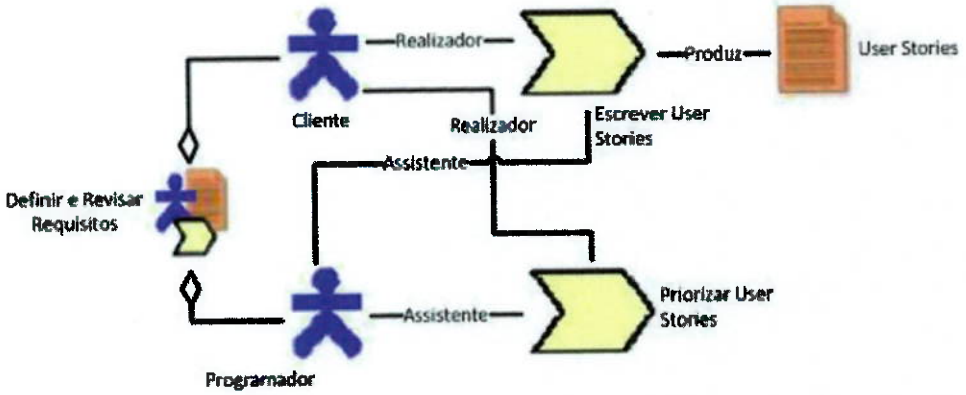


Figura 7 - Diagrama de classe estereotipado da etapa Definir e Revisar Requisitos. Fonte: Sampaio (2004).

2.3.3.3 Planejar o release

É onde se faz o planejamento do que vai ser entregue no release (versão) subsequente. Esse planejamento pode ser alterado caso as prioridades do cliente mudem. Conforme mostrado no diagrama da figura 8, os programadores estimam a dificuldade de implementação de cada história baseado nos cartões escritos anteriormente e orientados pela prática "jogo do planejamento", que descreve que uma release deve conter as histórias que agregam maior valor para o negócio do cliente. O release deve contar às histórias que agregam maior valor para o negócio do cliente e o cliente, auxiliado pelo programador, então define quantas histórias poderão implementar nas diversas iterações do release, elaborando com isso um plano do release (Release Plan).

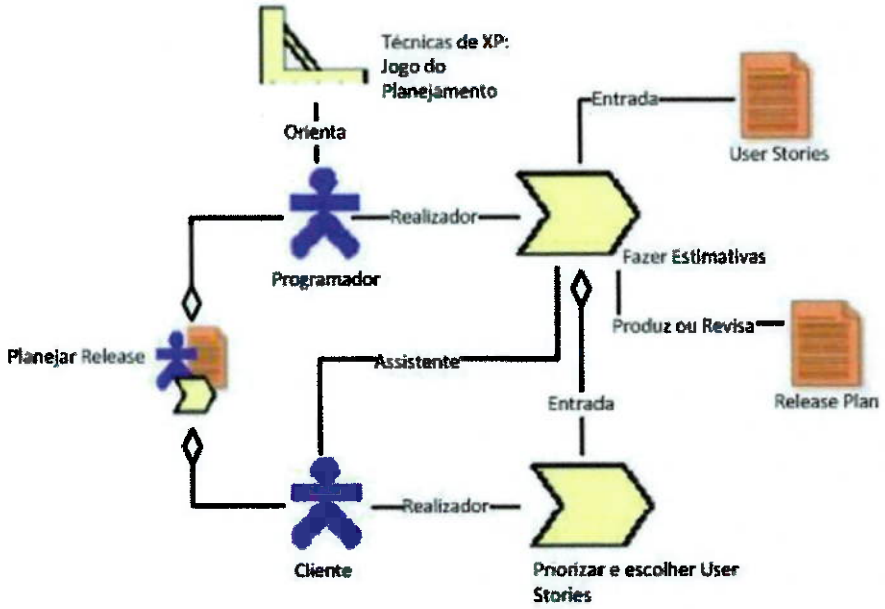


Figura 8 - Diagrama de classe estereotipado da etapa Planejar Release. Fonte: Sampaio (2004).

2.3.3.4 Planejar iteração

É onde o planejamento da iteração é feito. Em um release podem ocorrer várias iterações e com isso o planejamento da iteração procura refinar as estimativas feitas anteriormente e definir quantas e quais histórias serão implementadas na iteração. Na figura 9 tem-se o diagrama que representa as ações desta etapa. Primeiramente, o programador procura estimar o esforço de forma mais precisa, melhorando assim a estimativa anterior para posteriormente auxiliar o cliente a escolher a histórias a serem implementadas na iteração subsequente. Leva-se em consideração que a atividade Estimar Iteração é orientada pela prática Jogo do Planejamento, vista anteriormente.

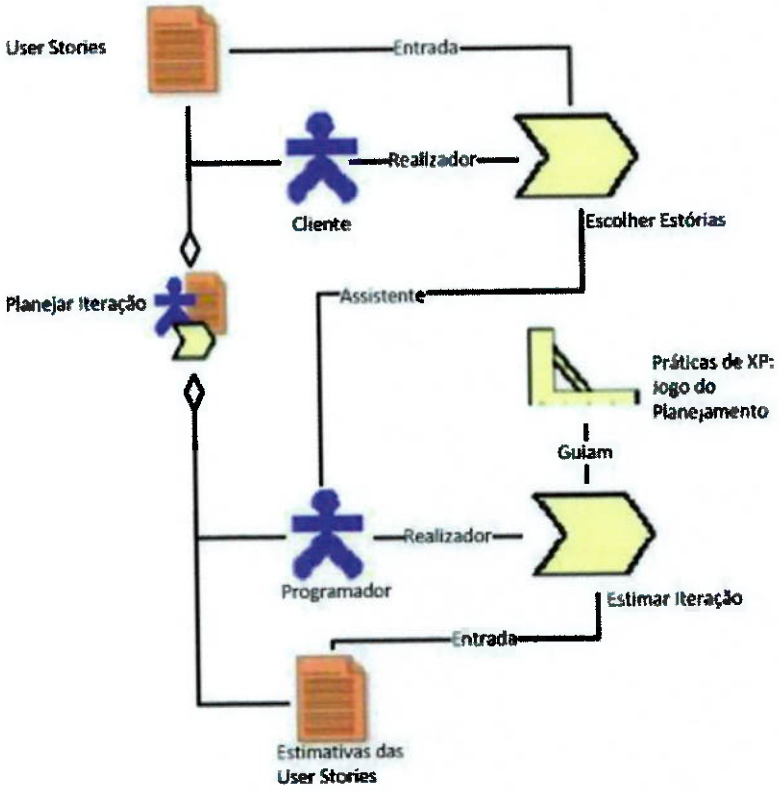


Figura 9 - Diagrama de classe estereotipado da etapa Planejar Iteração. Fonte: Sampaio (2004).

2.3.3.5 Escrever testes funcionais

É onde se realiza a escrita dos testes funcionais. Na figura 10 tem-se a representação desta etapa, onde os clientes, auxiliados pelos programadores, escrevem os cenários de testes necessários para validar as funcionalidades a serem implementadas. Esta etapa é orientada pela prática Testes Constantes.

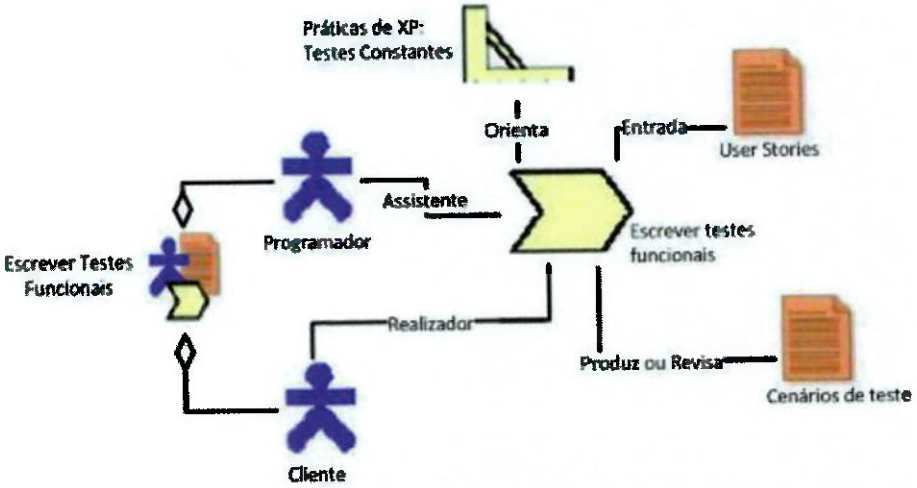


Figura 10 - Diagrama de classe estereotipado da etapa Escrever testes Funcionais. Fonte: Sampaio (2004).

2.3.3.6 Fazer projeto

É onde é produzido o projeto para as classes que serão implementadas. O objetivo é que o projeto esclareça as dúvidas entre os programadores da melhor estratégia para a implementação das classes do sistema. Na figura 11 vê-se que a atividade Projetar é orientada pela prática Projeto Simples, a fim de que os modelos não sejam criados de forma complexa como diagramas de classe Unified Modeling Language (UML), e sim de forma simples, como rascunho para esclarecer dúvidas sendo o mais simples possível. Seguindo o diagrama, caso algum programador vislumbre uma melhor solução, o mesmo deve aplicar a técnica de refatoramento no código em desenvolvimento, para melhoria e melhor entendimento do mesmo.

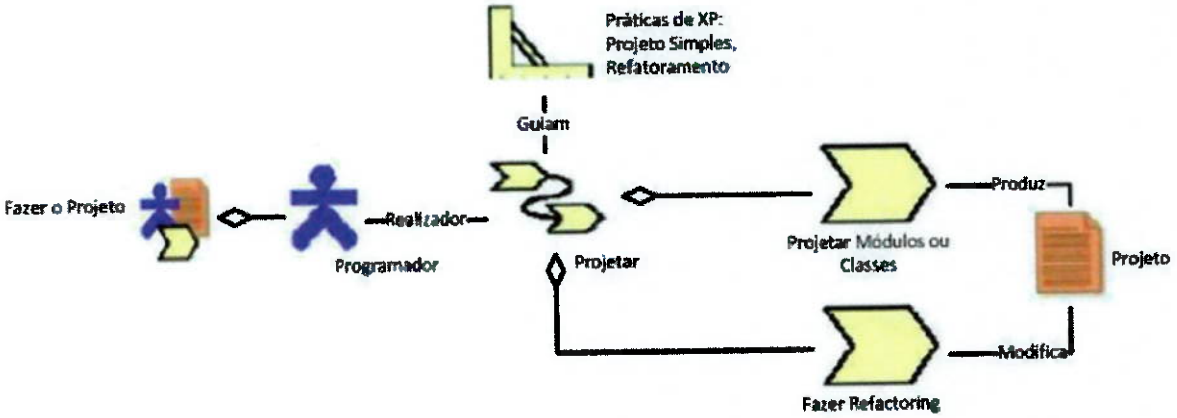


Figura 11 - Diagrama de classe estereotipado da etapa Fazer o Projeto. Fonte: Sampaio (2004).

2.3.3.7 Escrever testes de unidade

É onde se produz o código dos testes unitários, que correspondem aos testes com granularidade mais fina que os testes funcionais. Seguindo o diagrama da figura 12, os programadores escrevem o código para os testes que depois será validado quando as classes forem implementadas. Nesta atividade os programadores se orientam pela prática de Programação em pares para a escrita dos testes, assim como, seguem a orientação da prática Testes Unitários que diz que tudo que possa dar errado deve ser testado.

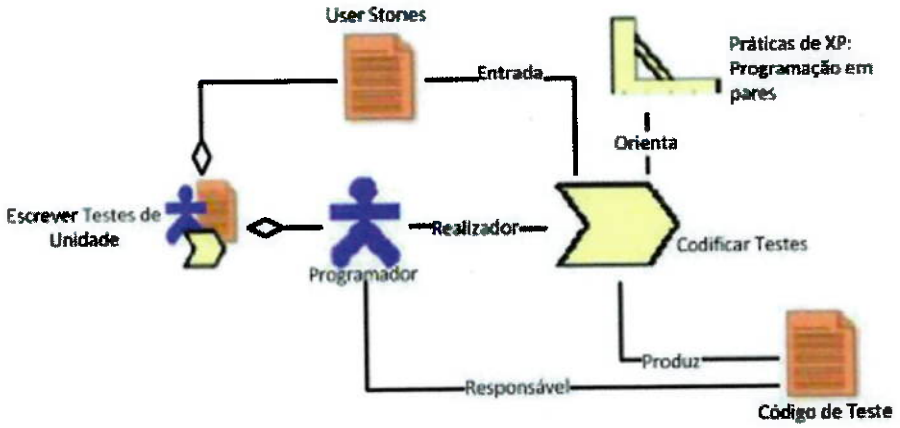


Figura 12 - Diagrama de classe estereotipado da etapa Escrever Testes de Unidade. Fonte: Sampaio (2004).

2.3.3.8 Codificar

É nesta etapa que os programadores implementam as histórias escritas anteriormente, sempre trabalhando em pares (orientação da prática Programação em pares). Como se pode ver no diagrama da figura 13, os programadores seguem um padrão de codificação (orientação da prática Padrão de Codificação) e não possuem domínio total sobre nenhuma parte do código que é de posse coletiva (orientação da prática Código de Propriedade Coletiva). As duplas devem ser constantemente trocadas, assim como o papel de cada programador a fim de que todos os membros da equipe tenham uma visão geral do sistema.

Figura 13 - Diagrama de classe estereotipado da etapa Codificar. Fonte: Sampaio (2004).

2.3.3.9 Testar

Nesta etapa são executados os testes de unidade codificados anteriormente. O diagrama da figura 14 demonstra que os responsáveis pela execução são os

programadores (orientados pela prática Testes Unitários) e caso encontrem algum erro também se responsabilizam pela sua correção.

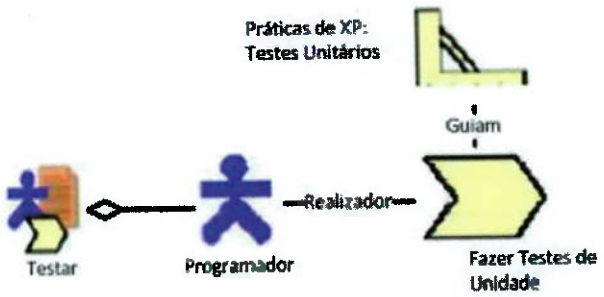


Figura 14 - Diagrama de classe estereotipado da etapa Testar. Fonte: Sampaio (2004).

2.3.3.10 Integrar

É onde as classes construídas e testadas devem ser integradas para compor os módulos do sistema. É importante verificar se os testes de unidade respeitam o que foi especificado nos cenários de teste e a cada integração, todos os casos de testes unitários devem ser executados e caso algum erro seja encontrado deve ser corrigido de imediato. Como se pode ver no diagrama da figura 15, os programadores são os responsáveis por integrar as funcionalidades, sendo orientados pela prática Integração Contínua.

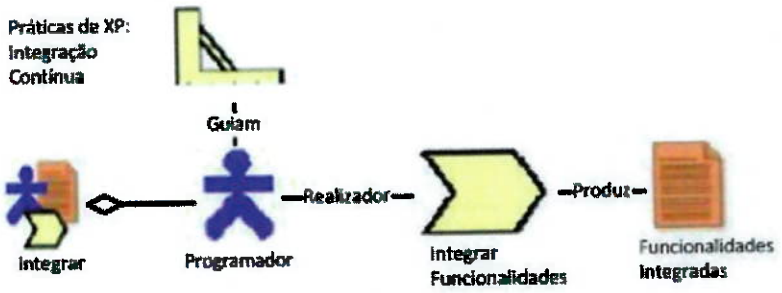


Figura 15 - Diagrama de classe estereotipado da etapa Integrar. Fonte: Sampaio (2004).

2.3.3.11 Fazer testes funcionais

Nesta etapa os programadores realizam os testes funcionais. O objetivo desta etapa é testar as funcionalidades como um todo e não somente os métodos a fim de validar se as funcionalidades estão de acordo com as expectativas do cliente. Como se pode ver no diagrama da figura 16, os programadores têm o auxilio do cliente e

os testes são baseados nos cenários e no código de teste produzido anteriormente. Um relatório de testes é produzido com o resultado dos testes.

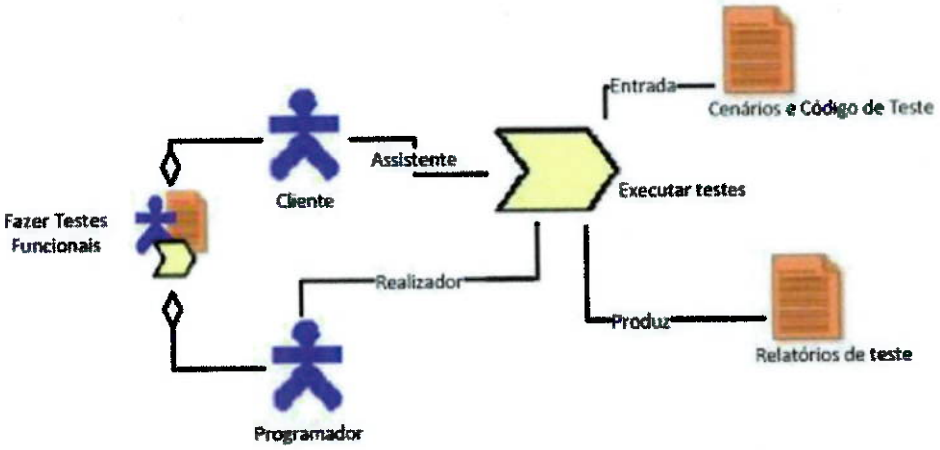


Figura 16 - Diagrama de classe estereotipado da etapa Fazer Testes Funcionais. Fonte: Sampaio (2004).

2.3.3.12 Colocar em produção

Após a conclusão das iterações tem-se uma versão produzida e que deve ser colocada em produção. Conforme o diagrama da figura 17, a equipe de suporte auxiliada pelo programador configura a infraestrutura na qual o sistema irá ser executado e coloca a versão atual do sistema em execução, podendo ser auxiliada por guias e checklists produzidos para a instalação.

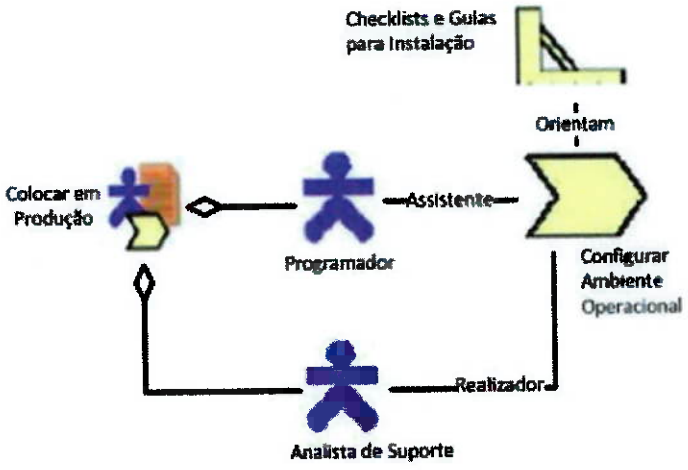


Figura 17 - Diagrama de classe estereotipado da etapa Colocar em Produção. Fonte: Sampaio (2004).

3 O MÉTODO ODP ÁGIL

A partir das tecnologias apresentadas no capitulo anterior será apresentado neste capitulo o Método ODP Ágil cujo objetivo é relacionar o método de desenvolvimento ágil XP com o padrão RM-ODP com uma estrutura estratégica.

O Método ODP Ágil consiste em relacionar as etapas e atividades do método XP com as visões do padrão arquitetural RM-ODP a fim de facilitar a definição da arquitetura para o sistema a ser desenvolvido de forma consistente e clara.

Na figura 18 tem-se uma ideia do como será feito esse relacionamento: de um lado a definição da Arquitetura de Sistema através do padrão RM-ODP e do outro o desenvolvimento ágil de software através do XP.

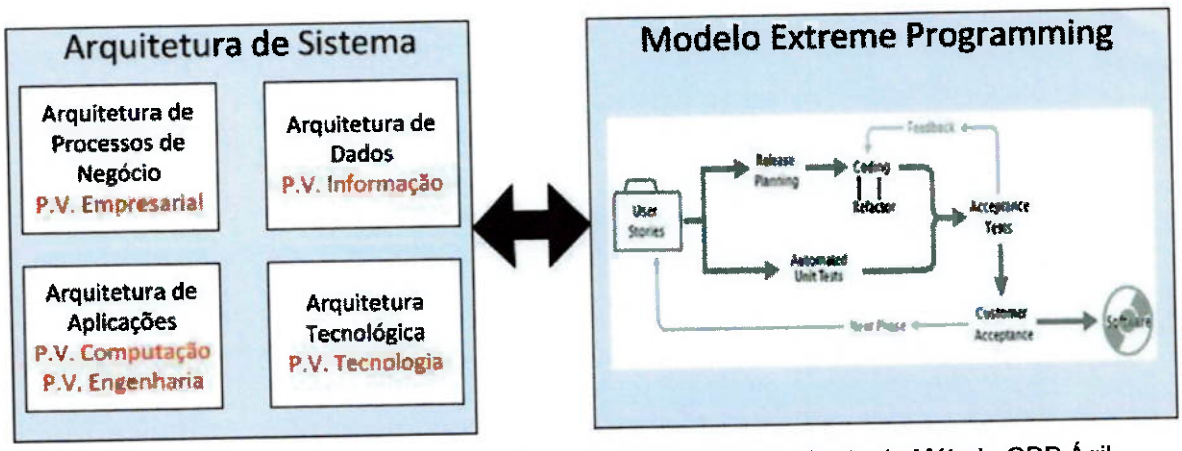


Figura 18 – Ideia de relacionamento entre as tecnologias para criação do Método ODP Ágil.

Primeiramente será definida cada uma das arquiteturas que compõe o Método ODP Ágil, que será apresentado na sequência.

3.1 Definição das Arquiteturas

3.1.1 Arquitetura de Processos de Negócio

A Arquitetura de Processos de Negócio possibilita documentar claramente os processos de negócio e com isso identificar os principais indicadores de sucesso dos mesmos. O gerenciamento do processo de negócios demanda de um ciclo fechado composto de quatro fases fundamentais: modelagem/melhoria, implantação, execução e controle. Tem por objetivo gerar vantagens competitivas sustentáveis e

duradouras e depende da organização de processos da organização e da continuidade com a qual é aplicada.

A modelagem de um processo consiste em identificar seu propósito, seus principais processos e indicadores de sucesso e detalha-los em subprocessos até chegar a um nível de atividades individuais.

Com as constantes mudanças dos processos de negócio são necessárias revisões cada vez mais frequentes.

Como exemplo, na figura 19 tem um modelo de processo de negócio de uma compra de um jornal.

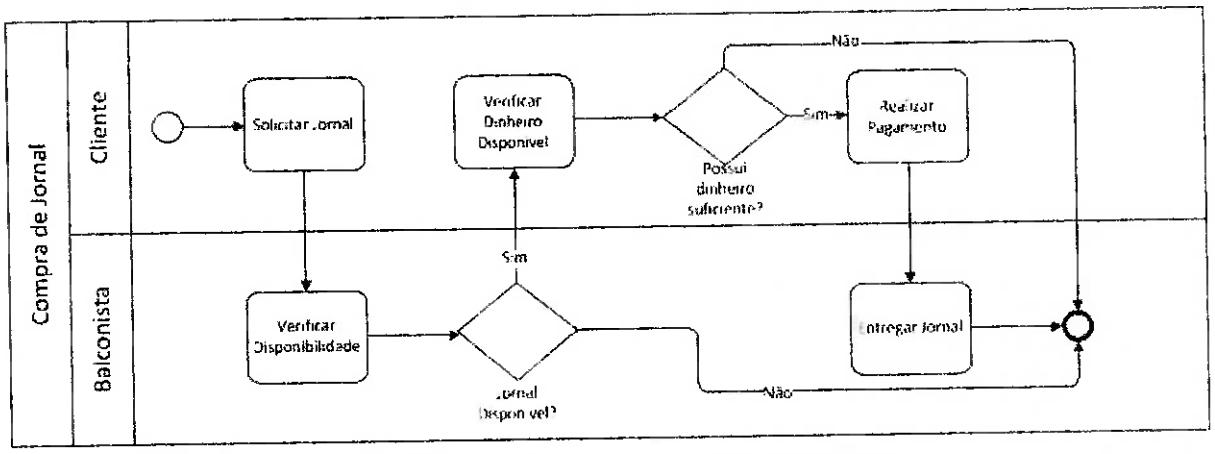


Figura 19 – Modelagem de Processo de Negócio.

3.1.2 Arquitetura de Dados

Segundo definição de Weill, Ross e Robertson (2006), a Arquitetura de Dados modela o conteúdo dos processos de negócios da empresa, tratando das propriedades, interação e comportamento associados a essa informação. Descreve como os dados são processados, armazenados e utilizados em um determinado sistema fornecendo os critérios para as operações de processamento de dados. Ele se baseia no que foi definido na Arquitetura de Processos de Negócio e seus artefatos resultantes (como por exemplo, o modelo BPMN dos processos).

A modelagem do repositório de dados tem sido pouco alterada ao longo do tempo desde a proposta de Modelos Entidade-Relacionamento (ER) de Chen (1976). Como linguagem para a modelagem, tem-se o UML que é altamente difundida que tem

como característica seu alto grau de aderência aos modelos e diagramas de especificação de aplicações.

Durante a tarefa de definição do estado de destino, o arquiteto de dados decompõe a informação até o seu nível atômico para em seguida fazer o caminho inverso e compor o contexto desejado. Para isso, existem três processos tradicionais de arquitetura: conceitual, lógico e físico (figura 20).

3.1.2.1 Arquitetura Conceitual de Dados

Visão de alto nível que dá suporte ao atendimento das necessidades de negócio de uma organização, direcionando as decisões sobre as soluções de tecnologia.

3.1.2.2 Arquitetura Lógica de Dados

Descreve com precisão os relacionamentos e as propriedades de cada uma das entidades de dados envolvidas em um domínio organizacional ou problema de negócio a ser resolvido com apoio de TI. Possui um estreito alinhamento ao modelo corporativo previamente concebido e preocupação com padrões de implementação da arquitetura de dados.

3.1.2.3 Arquitetura Física de Dados

É parte de um plano de tecnologia que esta focada em elementos reais e tangíveis a serem utilizados na implementação da arquitetura de dados. Engloba um esquema da tecnologia de banco de dados utilizado para viabilizar a realização de um projeto de arquitetura de dados.

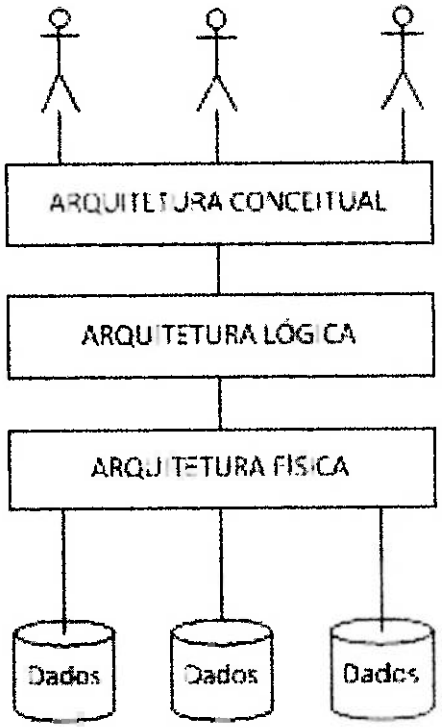


Figura 20 – Processos de Arquitetura de Dados.

3.1.3 Arquitetura de Aplicações

Segundo Weill, Ross e Robertson (2006), Arquitetura de aplicações é a definição de como as aplicações farão o gerenciamento dos dados e o fornecimento de informações para as pessoas que executam as funções de negócios, ou seja, é o mapa corporativo dos sistemas de aplicação e suas interfaces.

Existem diversas etapas que envolvem a modelagem de um sistema, como a modelagem de diagrama de casos de uso (mapeamento das atividades de interação entre usuários e módulos dos sistemas), passando por modelagem de classes (onde se estabelecem as estruturas lógicas e funções) além de diagramas de sequência (especificam a ordem de acionamento de cada componente de um sistema).

Pode-se ainda utilizar outros tipos de diagramas para auxilio na modelagem da arquitetura de aplicações como diagramas de colaboração, de estados e componentes.

Na modelagem da arquitetura de aplicações também é muito difundido o uso do UML para definir todos os diferentes tipos de diagramas citados.

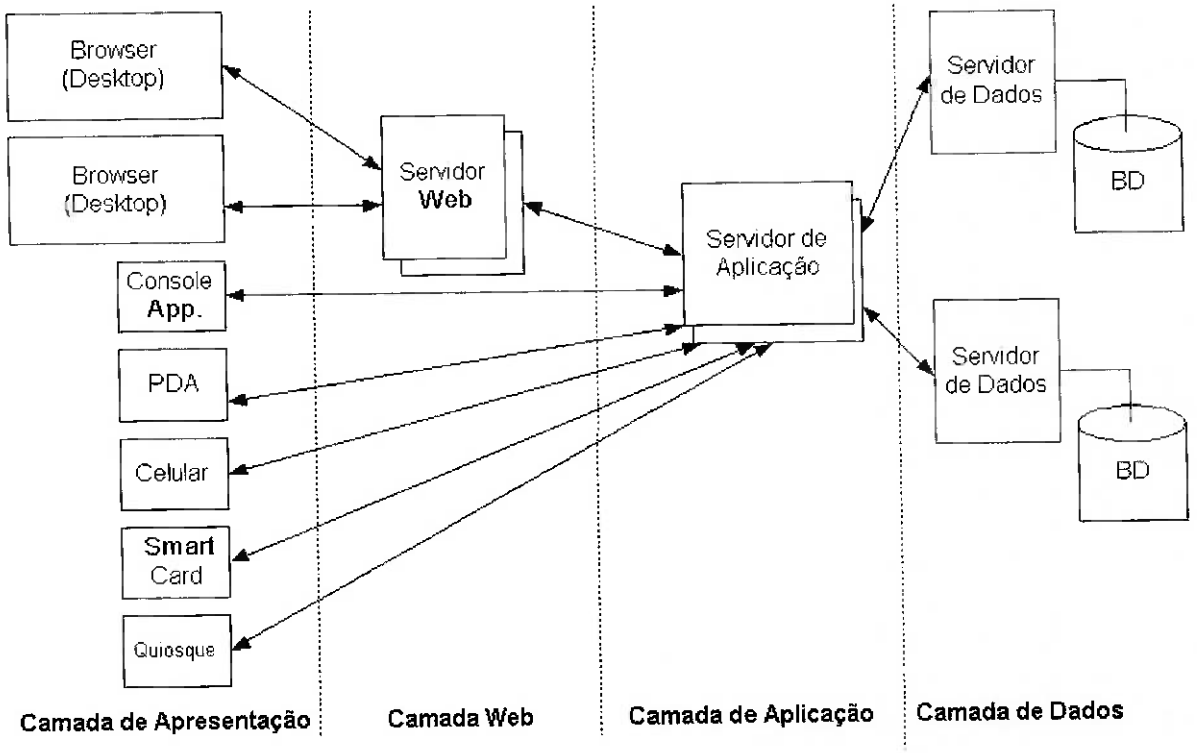


Figura 21 – Exemplo de Arquitetura de aplicações em N camadas.

Na figura 21 tem-se um exemplo de Arquitetura de aplicações em N camadas, representada pela camada de Apresentação (interface com o cliente), camada Web (servidor que atendem as requisições Web), camada de aplicação (onde são tratadas as regras de negócio) e a camada de dados (repositório de informações).

3.1.4 Arquitetura Tecnológica

Segundo Rosa (2008), por se tratar da camada mais próxima do operacional e da infraestrutura de TI, a Arquitetura tecnológica tem por responsabilidade a tomada de decisões quanto ao investimento em infraestrutura (lidando, por exemplo, com hardware, software, ferramentas, redes, politicas de uso e segurança).

Para isso, as decisões são baseadas em três fatores principais: necessidade mínima de desempenho dos componentes para atender aos sistemas, necessidade de conectividade entre empresas distribuídas fisicamente e limitações financeiras.

Devido à importância cada vez maior da disponibilidade dos sistemas para a viabilização dos processos de negócio, a escolha de componentes de infraestrutura e definição de serviços de suporte se tornam fundamentais.

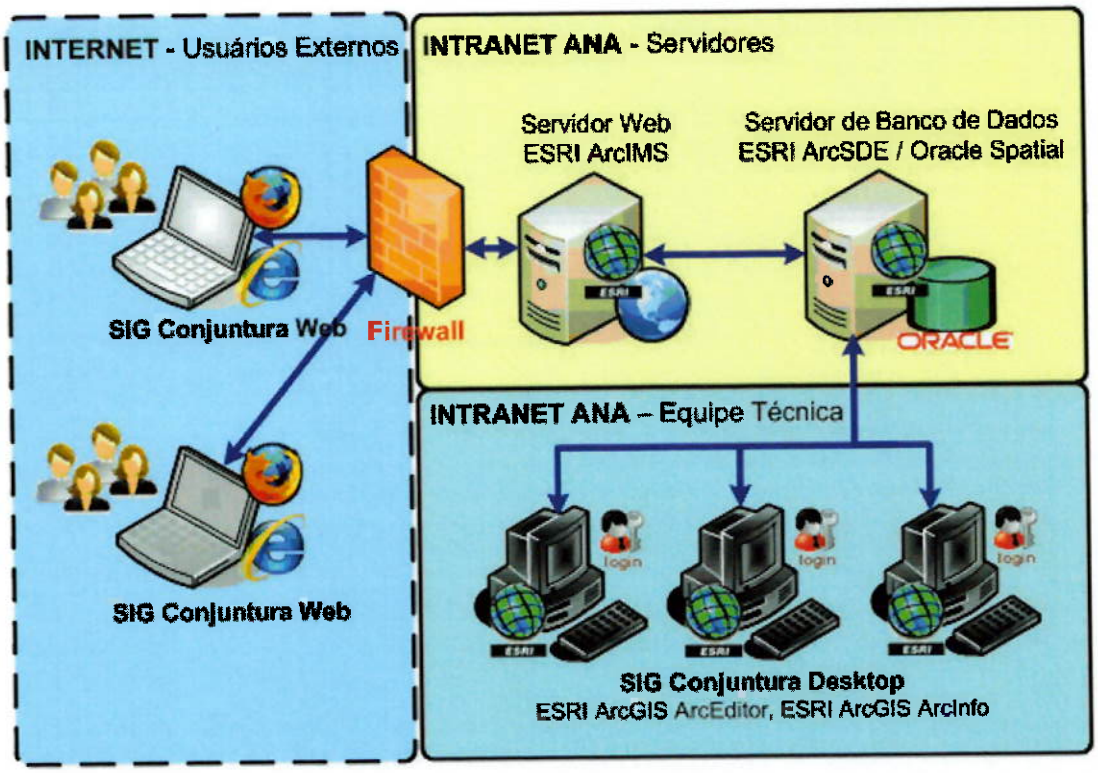


Figura 22 – Arquitetura Tecnológica do projeto Conjuntura. Fonte: Bem e Hashimoto (2010).

A figura 22 apresenta um exemplo de Arquitetura Tecnológica. O projeto é o Conjuntura da Agência Nacional de Águas (ANA) e segundo Bem e Hashimoto, se baseia na plataforma Environmental Systems Research Institute (Esri). A estruturação e o armazenamento dos dados foram feitos no sistema de gerenciamento de banco de dados Oracle com a extensão Oracle Spatial.

3.2 Introdução ao Método ODP Ágil

Como apresentado no capitulo 2, a abordagem arquitetural RM-ODP possui uma forte visão arquitetural. Em contrapartida, o método ágil de desenvolvimento XP tem por característica a entrega do sistema em partes, ou seja, o desenvolvimento iterativo bem definido através de suas etapas.

A partir desta perspectiva será apresentada a hipótese de relacionamento entre as etapas de desenvolvimento do método XP e os pontos de vista do padrão RM-ODP. Este relacionamento irá se basear nas fases de desenvolvimento do método XP, onde serão definidos quais delas serão relacionadas com quais pontos de vista do padrão RM-ODP.

Como resultado, se terá a definição de um método de desenvolvimento com estrutura ágil otimizada para definir, criar e manter a arquitetura de sistemas.

3.3 Relacionamento entre a metodologia XP e o padrão RM-ODP

Segundo Sommerville (2011), a metodologia XP foi concebida para ser um *framework* com objetivo de ser uma forma controlada e integrada de elaboração de sistemas de forma iterativa e ágil. Já o padrão RM-ODP foi concebido para ser um *framework* para a elaboração de sistemas distribuídos, com foco na definição de seus pontos de vista e respectivamente suas arquiteturas.

O objetivo do relacionamento de ambos é a soma das atividades das etapas de desenvolvimento da metodologia XP com as atividades para a definição das arquiteturas de cada ponto de vista do padrão RM-ODP conforme apresentado na figura 23.

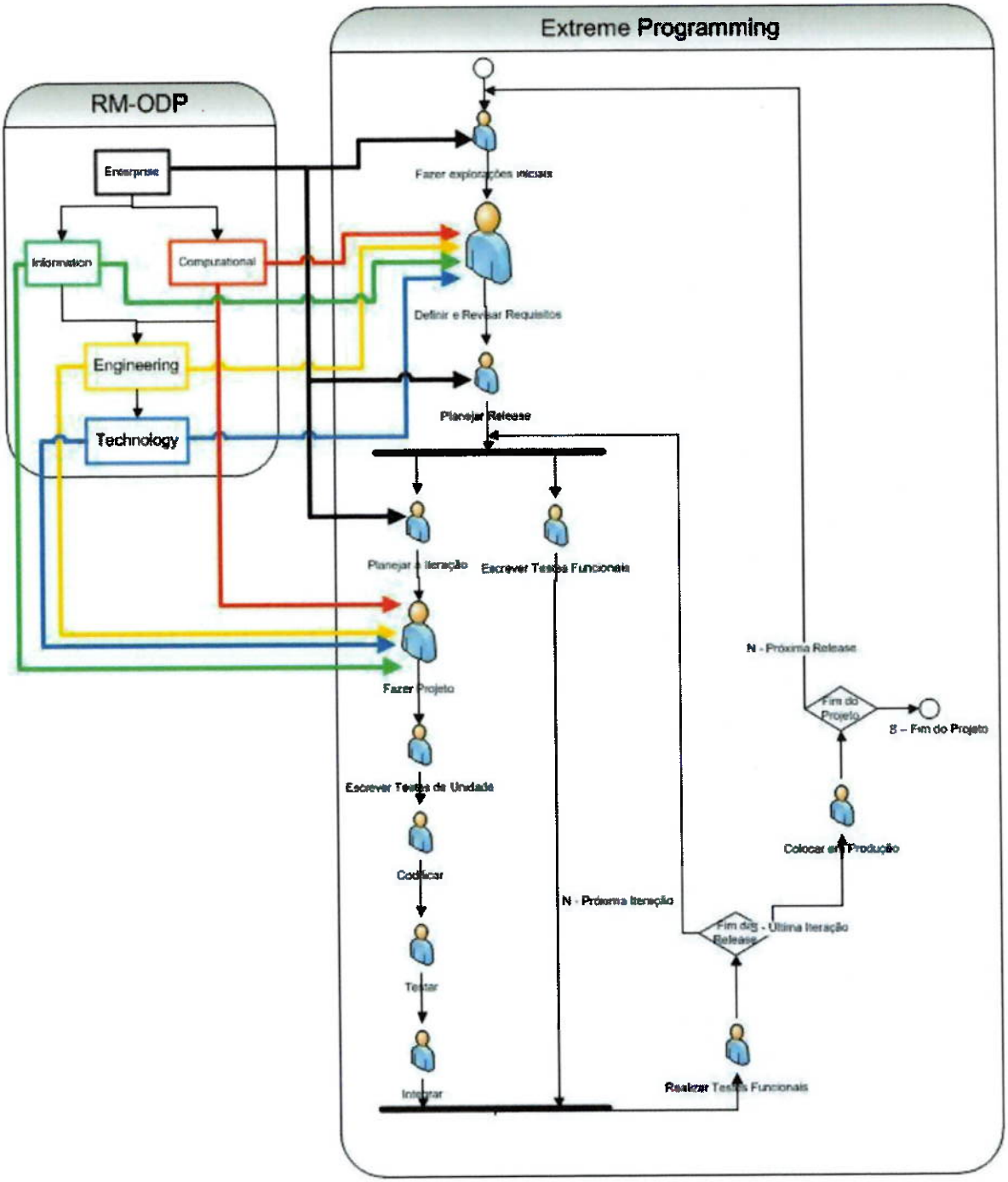


Figura 23 – Relacionamentos entre Metodologia XP e os pontos de vista do padrão RM-ODP.

O relacionamento apresentado na figura 23 parte do principio de que as fases da metodologia XP seguem, em linhas gerais, os pontos de vista do padrão RM-ODP. Consequentemente, cria-se um relacionamento lógico entre as atividades de ambos assim como seus artefatos.

A soma das atividades não representa simplesmente a execução sequencial de cada uma das atividades do relacionamento. Ela representa uma estratégia de

execução, onde as atividades das etapas da metodologia de desenvolvimento XP podem influenciar nas atividades para definição dos pontos de vista do padrão RM-ODP e vice-versa. Desta forma, as estratégias de execução da soma das atividades das etapas da metodologia XP e dos pontos de vista do padrão RM-ODP serão escopo deste trabalho.

A descrição das estratégias de execução com o relacionamento das atividades da metodologia de desenvolvimento XP com o padrão RM-ODP serão detalhadas na sequência.

3.4 Fluxos de atividade do Método ODP Ágil

Nesta etapa do trabalho serão apresentadas os fluxos de etapas e suas atividades para a implementação do Método ODP Ágil. O fluxo segue o padronizado na metodologia de desenvolvimento XP, tendo a aplicação dos pontos de vista do padrão RM-ODP em algumas etapas especificas para que se possa definir as arquiteturas do sistema. Nos itens a seguir apenas serão indicados quais etapas e atividades serão realizadas para a definição das arquiteturas e, no capitulo seguinte, serão apresentadas a forma de implementação dos mesmos através de um experimento.

Nosso objetivo nesta etapa não é o de detalhar a metodologia de desenvolvimento XP (vista com detalhes no capitulo 2), e sim mostrar onde os pontos de vista do padrão RM-ODP devem ser aplicados, ou seja, descrever quais pontos de vistas serão aplicados e em quais etapas cada um deles se refere. Portanto, somente as etapas da metodologia de desenvolvimento XP relacionadas com os pontos de vista do padrão RM-ODP serão detalhadas. As demais etapas seguem conforme especificação da metodologia ágil de desenvolvimento de software XP.

As etapas do desenvolvimento e o seu relacionamento com os pontos de vista teve seus processos detalhados através de diagramas de classes estereotipados utilizando a notação UML.

A criação dos processos se deu através da utilização da modelagem das etapas da metodologia ágil de desenvolvimento XP criada por Sampaio (2004) utilizando a linguagem *Software Process Engineering Metamodel* (SPEM), que é uma meta-

linguagem de modelagem de processos que oferece algumas representações e estereótipos para modelar seus principais elementos em diagramas UML, através de uma analise interpretativa das literaturas do padrão RM-ODP.

Sendo assim, conforme a figura 23, as etapas da metodologia de desenvolvimento XP a serem detalhadas e relacionadas com o padrão RM-ODP são:

- Fazer explorações iniciais;
- Definir e Revisar Requisitos;
- Planejar Release;
- Planejar a iteração;
- Fazer projeto.

3.4.1 Etapa Fazer Explorações Iniciais

Na etapa Fazer Explorações iniciais o importante é decidir se o projeto é ou não viável. Para isso são feitas as explorações iniciais e investigações sobre o projeto a ser implantado.

Os clientes participam ativamente dentro da equipe de desenvolvimento para a breve definição dos requisitos e do escopo do sistema. Já os programadores realizam experimentos iniciais com as possíveis tecnologias e infraestrutura a serem escolhidas para verificar a viabilidade tecnológica da solução obtendo-se uma ideia inicial da tecnologia necessária para o sistema.

É nesta etapa que será iniciada a aplicação do ponto de vista da Empresa e como consequência estará sendo definida a Arquitetura de Processos de Negócios.

Segundo descrito no padrão ISO/IEC 10746, o ponto de vista da empresa define as perspectivas de negócios do sistema através de seu propósito, escopo e políticas. Nele o sistema é modelado através de suas funcionalidades requeridas, os domínios envolvidos, os atores e seus papéis.

Como ao longo do desenvolvimento existem alterações e criações de novos requisitos, a Arquitetura de Processos de Negócios será concluída na etapa de

Planejar Release e revisada em uma etapa de planejamento dentro das iterações definidas pela metodologia de desenvolvimento XP.

Para a definição do ponto de vista da Empresa é necessário o levantamento dos requisitos de negócio que são os objetivos que a empresa pretende obter com o desenvolvimento do sistema. Para representar este ponto de vista nesta etapa será utilizado o Diagrama de Caso de Uso e a construção do artefato Business Process Modeling Notation (BPMN) (em português Notação de Modelagem de Processos de Negócio).

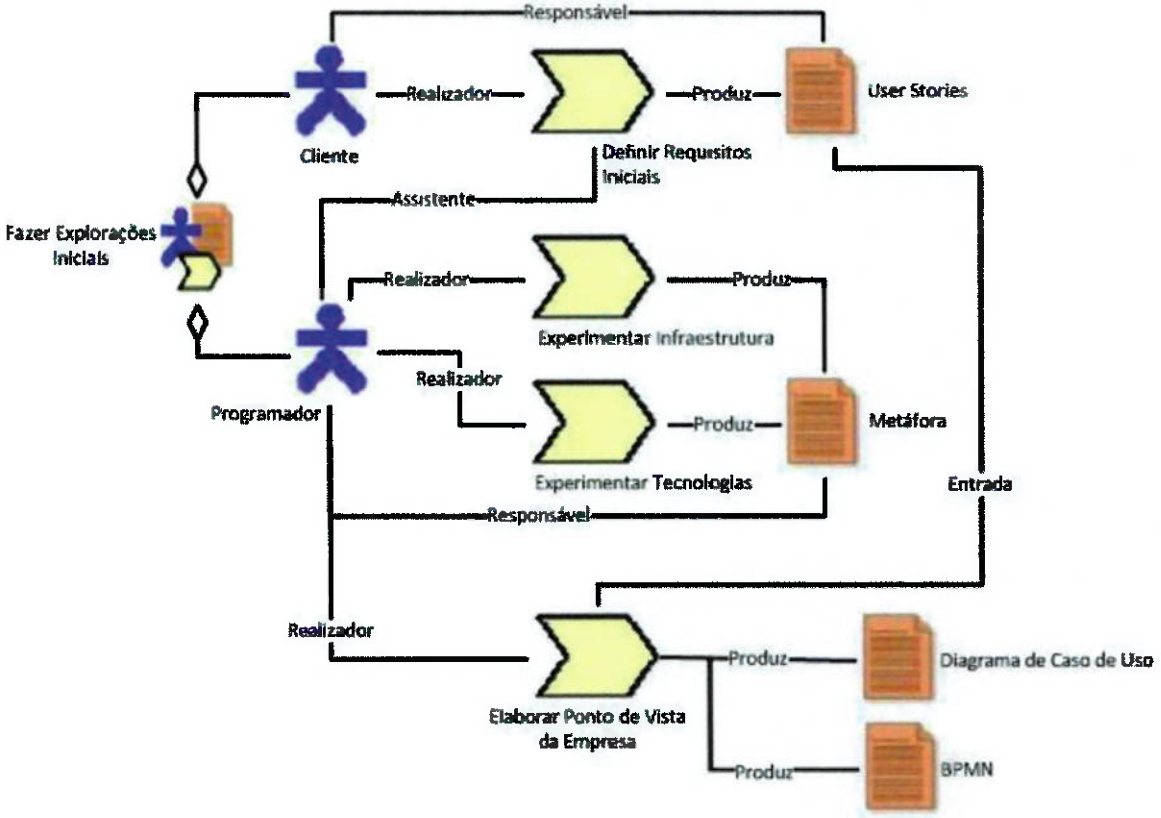


Figura 24 – Diagrama de classe estereotipado da etapa Fazer Explorações Iniciais.

Analisando o diagrama da figura 24, tem-se as atividades Definir Requisitos Iniciais, Experimentar Infraestrutura, Experimentar Arquiteturas e Elaborar Ponto de Vista da Empresa. O objetivo da atividade Definir Requisitos Iniciais é produzir os User Stories tendo como realizador o Cliente, que é assistido pelo Programador. Para a elaboração do Ponto de vista da Empresa, o Programador tem o papel de realizador. Ele se baseia nos User Stories produzidos anteriormente para elaborar o Diagrama de Caso de Uso e o artefato BPMN. Na sequência, o Programador é o realizador

das atividades Experimentar Infraestrutura e Experimentar Tecnologias, a fim de realizar experimentos iniciais com as possíveis tecnologias e infraestrutura e produzir uma Metáfora como resultado (que nada mais é uma analise básica da arquitetura sugerida para se ter uma ideia da tecnologia necessária para o sistema).

3.4.2 Etapa Definir e Revisar Requisitos

Na etapa Definir e Revisar Requisitos é onde são definidos e revisados os requisitos que farão parte da release do sistema.

Os clientes são novamente integrados a equipe de desenvolvimento com o objetivo de escrever e priorizar as histórias com o auxilio dos programadores. Os clientes escrevem as histórias em cartões contendo as descrições da mesma, e também define a prioridade da mesma. Os programadores auxiliam na criação dos cartões e por fim definem uma estimativa de esforço para cada uma.

Com isso, têm-se como resultado desta atividade os cartões de histórias com as descrições detalhadas de suas funcionalidades e suas estimativas de esforço.

Com os requisitos e o ponto de vista da Empresa definidos, serão aplicados os pontos de vista Computacional, Informação, Engenharia e Tecnologia a fim de obtermos a definição das arquiteturas de Dados, de Aplicações e Tecnológica. Cada um dos pontos de vistas tem sua descrição definida pelo padrão ISO/IEC 10746, conforme abaixo:

- Ponto de vista da Informação define a estrutura, a semântica e o relacionamento entre os elementos de informação do sistema. Ele descreve uma visão comum dos recursos que apoiam os requerimentos de informação definidos no ponto de vista da Empresa.

  Para representar este ponto de vista será utilizado o Diagrama Entidade-Relacionamento;

- Ponto de vista da Computação define as funcionalidades do sistema e o distribui em componentes.

Para representar este ponto de vista será utilizado o Diagrama de Classe, a fim de se obter a identificação dos seus objetos e interfaces, e o Diagrama de Sequência;

- Ponto de vista da Engenharia define os mecanismos e funções necessárias para suportar o sistema complementando o ponto de vista da Computação e Informação. Define também as necessidades de comunicação e o desdobramento das funcionalidades além da distribuição dos objetos através da infraestrutura física.

  Para representar este ponto de vista será utilizado o Diagrama de Implantação;

- Ponto de vista da Tecnologia define as tecnologias a serem utilizadas no sistema, detalhando seus componentes e suas plataformas estabelecendo uma ligação entre os conjuntos de especificações dos pontos de vistas empresa, informação, computação e engenharia.

  Para representar esse ponto de vista será utilizado as definições dos padrões de especificação e a plataforma tecnológica.

Lembrando que a metodologia de desenvolvimento XP absorve as mudanças constantes dos requisitos e com isso as histórias definidas nesta etapa podem ser alteradas a qualquer instante e serão revistas em outra etapa que trata de requisitos dentro das iterações, assim como as arquiteturas de Dados, de Aplicações e Tecnológica.

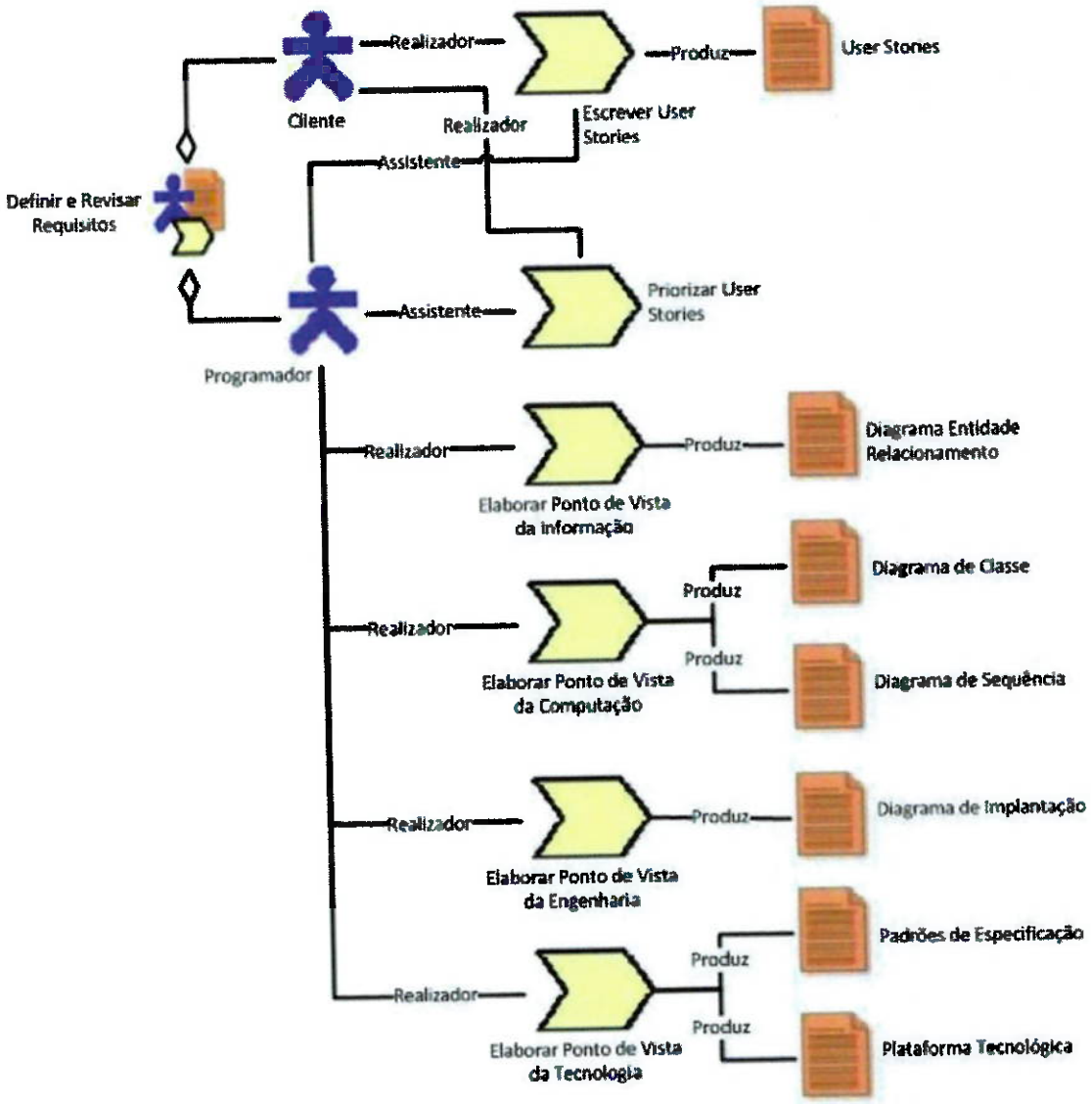


Figura 25 - Diagrama de classe estereotipado da etapa Definir e Revisar Requisitos.

Analisando o diagrama da figura 25, tem-se como atividades Escrever User Stories, Priorizar User Stories, Elaborar Ponto de Vista da Informação, Elaborar Ponto de Vista da Computação, Elaborar Ponto de Vista da Engenharia e Elaborar Ponto de Vista da Tecnologia. A atividade Escrever User Stories tem o Cliente como realizador assistido pelo Programador tendo como resultado o artefato User Stories. Uma vez escrito, tem-se a atividade de Priorizar os User Stories, que tem o auxilio do Programador estimando o esforço de cada User Storie, para que o Cliente defina a prioridade dos mesmos. Na sequência o Programador realiza as atividades relacionadas à elaboração dos pontos de vista e seus artefatos: Ponto de Vista da Informação (produz artefato Diagrama Entidade Relacionamento), Ponto de Vista da Computação (produz artefatos Diagrama de Classe e Diagrama de Sequência),

Ponto de Vista da Engenharia (produz artefato Diagrama de Implantação), Ponto de Vista da Tecnologia (define os Padrões de Especificação e a Plataforma Tecnológica).

3.4.3 Etapa Planejar Release

Na etapa Planejar Release é onde é planejado o que será entregue na próxima release (versão).

O programador tem o papel de estimar a dificuldade de implementação das histórias definidas anteriormente e orientados pela prática "jogo do planejamento". O cliente por sua vez tem o papel de priorizar as histórias, dando maior prioridade a aquelas que agregam maior valor para o negócio.

Com a definição da primeira release a ser desenvolvida, o ponto de vista da Empresa é revisado através da validação do artefato Business Process Modeling Notation (BPMN) (em português Notação de Modelagem de Processos de Negócio), sendo atualizado caso necessário. Na releases subsequentes o BPMN será revisado e atualizado de acordo com as alterações e novas funcionalidades agregadas ao sistema.

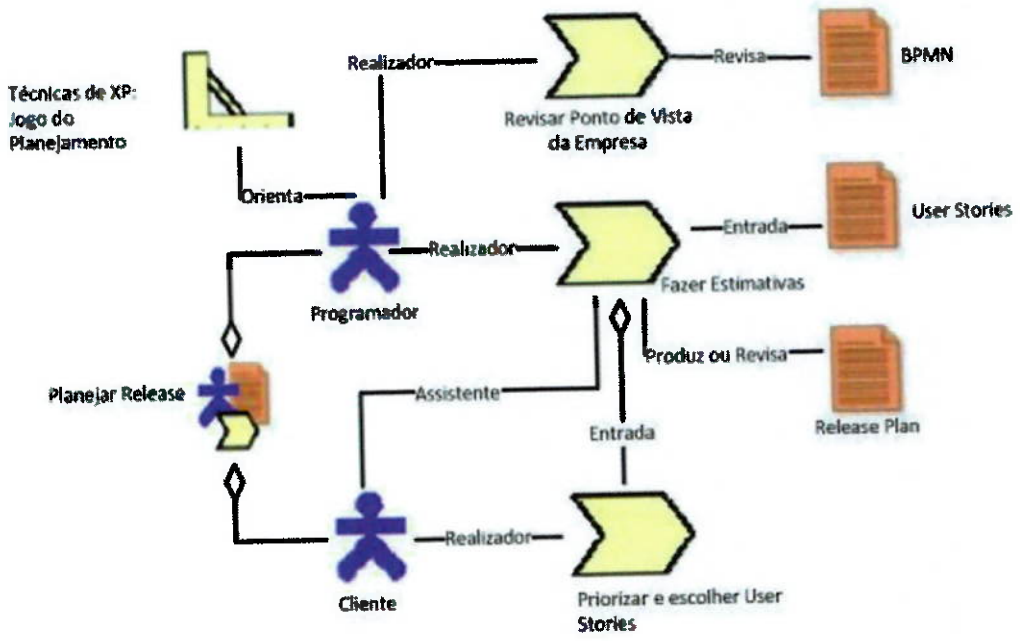


Figura 26 - Diagrama de classe estereotipado da etapa Planejar Release.

Analisando o diagrama da figura 26, tem-se como atividades Revisar Ponto de Vista da Empresa, Fazer Estimativas e Priorizar e escolher User Stories. A atividade Priorizar e escolher User Stories tem como realizador o Cliente, que define a prioridade de cada User Storie de acordo com a sua importância para o negócio. Uma vez definidas as prioridades, o Programador realiza a estimativa da release, definindo quantas e quais histórias farão parte da release levando em conta as prioridades e dificuldades de cada uma delas e orientados pela prática "jogo do planejamento". O resultado desse trabalho é um plano para a release a ser desenvolvida. Por fim, a atividade de Revisar o Ponto de Vista da Empresa é realizada pelo Programador baseado na definição da release e nos requisitos definidos anteriormente, tendo como resultado o artefato BPMN revisado.

3.4.4 Etapa Planejar a Iteração

Na etapa Planejar a Iteração procura-se refinar as estimativas feitas no planejamento da release e verificar quantas e quais histórias serão implementadas na iteração. Em uma release (versão do sistema) pode ocorrer uma ou mais iterações.

O programador tem o papel de analisar cada história de forma refinada, que tem a sua estimativa revisada, resultando em um valor mais preciso e de menor risco de erro. Com o resultado desta revisão, o programador apresenta ao cliente o que pode ser feito na iteração. O cliente por sua vez tem a responsabilidade de decidir quais histórias serão implementadas na iteração subsequente e qual será deixada para uma próxima iteração. Resumindo: o programador tem a responsabilidade da estimativa de esforço de cada história enquanto o cliente é responsável por atribuir prioridade a cada uma das histórias e com isso realizar a escolha de quais delas entrarão na iteração. Leva-se em consideração que a atividade Estimar Iteração é orientada pela prática Jogo do Planejamento, vista anteriormente.

Como nesta etapa existem alterações e criações de novos requisitos, o ponto de vista da Empresa será revisado e atualizado se necessário através do Diagrama de Caso de Uso. Consequentemente, a arquitetura de processo de negócios também será revisada e atualizada caso necessite.

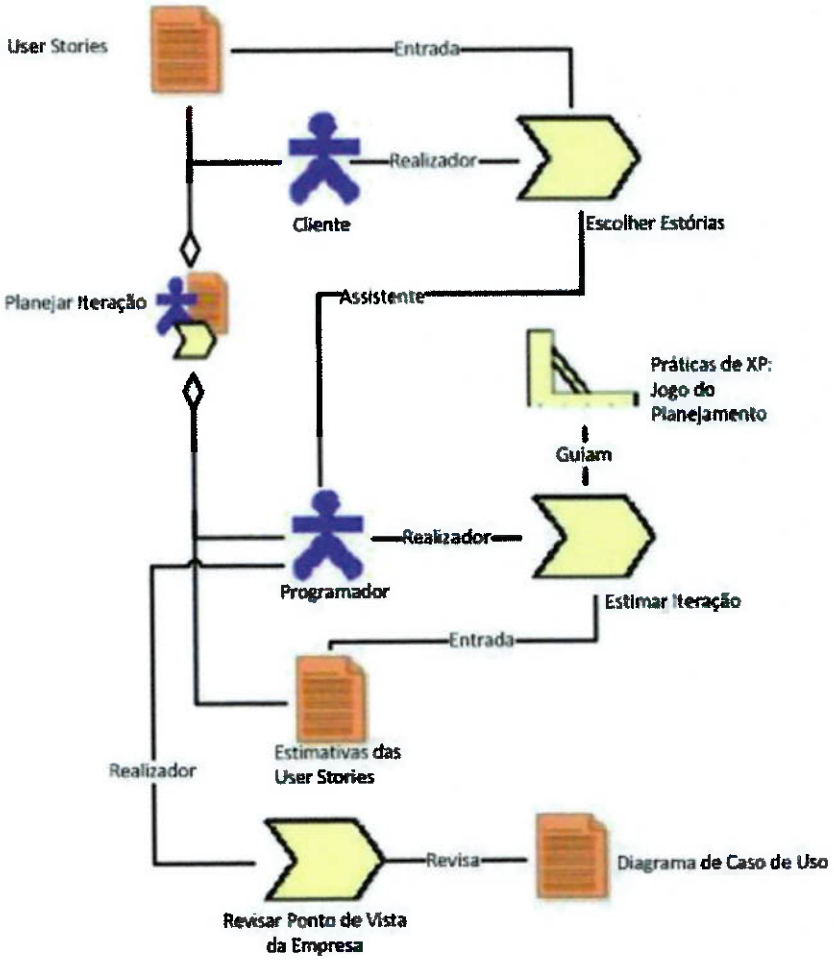


Figura 27 - Diagrama de classe estereotipado da etapa Planejar Iteração.

Analisando o diagrama da figura 27, tem-se como atividades Escolher Histórias e Revisar o Ponto de Vista da Empresa. A atividade Estimar a Iteração tem como realizador o Programador, que refina as estimativas de esforço das User Stories orientado pelas práticas de Xp (Jogo do Planejamento, visto anteriormente). Uma vez refinadas as estimativas de esforços, é iniciada a atividade Escolher Stories, que tem como realizador o Cliente assistido pelo Programador que lhe apresenta as estimativas refinadas a fim de auxilia-lo na decisão de quais Histórias irão pertencer à iteração subsequente. Por fim, a atividade de Revisar o Ponto de Vista da Empresa realizada pelo Programador baseado nas novas estimativas refinadas dos User Stories, produzindo alterações no Diagrama de Caso de Uso.

3.4.5 Etapa Fazer o Projeto

Na etapa Fazer o Projeto é onde é criado de fato o projeto com o objetivo de esclarecer as dúvidas entre os programadores da melhor estratégia para implementar as classes do sistema.

Por definição da metodologia de desenvolvimento XP não é necessária à produção de modelos como diagramas de classe UML, e sim a produção de simples modelos através de rascunhos com o objetivo de esclarecer dúvidas, conforme orientação da prática Projeto Simples. Porém, com a integração dos pontos de vista do padrão RM-ODP, criou-se o relacionamento desta etapa com os pontos de vista da Informação, Computação, Engenharia e Tecnologia, assim como as suas respectivas arquiteturas de Dados, de Aplicações e Tecnológica. Com a aplicação dos pontos de vista será obtido como resultado a implementação e/ou revisão de alguns artefatos criados anteriormente na etapa Definir e Revisar Requisitos que irão nos auxiliar na definição das arquiteturas, como será visto a seguir.

Esta etapa irá absorver as mudanças constantes dos requisitos e com isso as histórias definidas na etapa Definir e Revisar Requisitos podem ser aqui alteradas, tendo assim os pontos de vistas Informação, Computação, Engenharia e Tecnologia sendo revistos assim como as arquiteturas de Dados, de Aplicações e Tecnológica. A figura 28 apresenta o diagrama de classe estereotipado da etapa Fazer o Projeto.

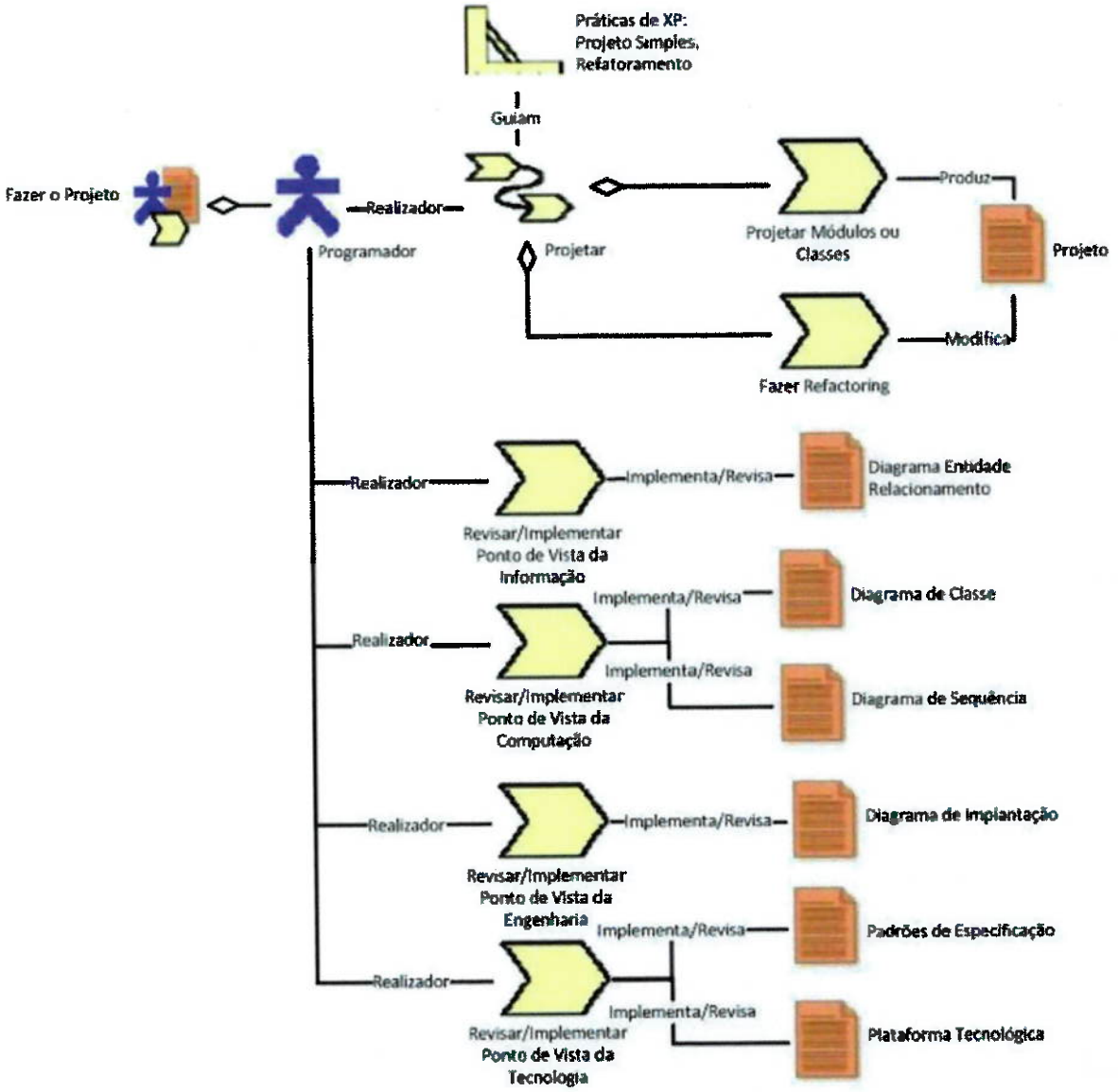


Figura 28 - Diagrama de classe estereotipado da etapa Fazer o Projeto.

Analisando o diagrama da figura 28, tem-se o módulo Projetar que contém as atividades Planejar Módulos ou Classes, Fazer Refactoring e também às atividades Revisar/Implementar Ponto de Vista da Informação, Revisar/Implementar Ponto de Vista da Computação, Revisar/Implementar Ponto de Vista da Engenharia e Revisar/Implementar Ponto de Vista da Tecnologia. O módulo Projetar (que contém as atividades Projetar Módulos ou Classes e Fazer Refactoring) é realizado pelo Programador orientado pelas práticas do XP (Projeto Simples e Refatoramento) a fim de gerar como artefato o Projeto. Existem também as atividades relacionadas às revisões e implementações dos pontos de vista realizadas pelo Programador. A cada iteração, essas atividades têm como objetivo revisar os artefatos gerados

anteriormente por cada ponto de vista devido as constantes alterações/inclusões de requisitos no sistema e na sequência implementar os mesmos. Sendo assim, o Programador tem as seguintes atividades: Revisar/Implementar Ponto de Vista da Informação (revisar e implementar Diagrama Entidade Relacionamento), Revisar/Implementar Ponto de Vista da Computação (revisar e implementar Diagrama de Classe e Diagrama de Sequência), Revisar/Implementar Ponto de Vista da Engenharia (revisar e implementar Diagrama de Implantação), Revisar/Implementar Ponto de Vista da Tecnologia (revisar e implementar os Padrões de Especificação e a Plataforma Tecnológica).

4 EXPERIMENTO

4.1 Introdução

O objetivo deste capítulo é o de apresentar a forma prática do Método ODP Ágil através da realização de um experimento, cujo contexto esta relacionado à Gestão de Contratos Consignados de uma instituição financeira.
O foco deste capítulo não é o de apresentar o projeto como um todo e sim verificar que o processo continua ágil apesar de inserir e modificar elementos cruciais, como a definição das arquiteturas.

4.2 Cenário

Segundo o Banco Central (BC) (2013), o empréstimo consignado cresceu 17,4% no ano de 2012. Graças às facilidades e o baixo risco que a linha de crédito oferece aos credores, essa modalidade de financiamento tem tido um crescimento exponencial ao longo dos anos (gráfico 1). Com isso, o empréstimo consignado tem sido tratado como a menina dos olhos pelas instituições financeiras.

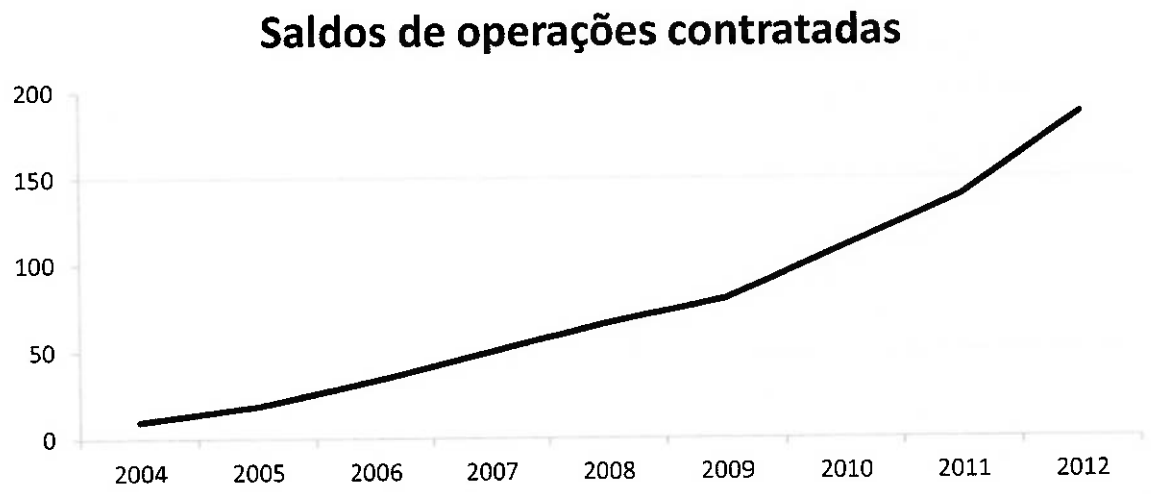


Gráfico 1 – Gráfico do crescimento real do crédito consignado. Fonte: Banco Central do Brasil (2013).

As instituições financeiras visualizam que ainda possuem um grande espaço para crescimento, e por isso os investimentos em campanhas nessa modalidade tem sido cada vez maior.

O empréstimo consignado é um produto que as instituições financeiras oferecem aos funcionários públicos, funcionários privados e pensionistas que permite aos mesmos que sejam realizados empréstimos bancários com descontos das parcelas diretamente na folha salarial do mesmo.

Esse mesmo empréstimo pode ser estendido através da pratica do refinanciamento, onde o cliente solicita mais dinheiro e o prazo do empréstimo é ampliado, desde que ocorram as aprovações necessárias.

Dessa maneira, a oportunidade de manter o cliente fiel à instituição financeira é grande e com uma boa Gestão de Contratos uma melhor administração dessa carteira é visível e produtiva.

Atualmente, não ter um controle específico ou metodologia para gerir contratos é sinal de problemas dentro de qualquer empresa, principalmente nas instituições financeiras.

Segundo a consultora IDC Brasil, o mercado brasileiro possui algumas ferramentas especificas para gestão de contratos, consideradas eficientes e que geralmente são implementadas nas empresas para suprir a necessidade de uma gestão eletrônica de documentos. No entanto, atualmente não existe nenhuma ferramenta especifica que possa gerir os contratos financeiros do tipo consignado.

O maior problema que as instituições financeiras enfrentam é a falta de um modelo de arquitetura de sistemas que possa auxiliar na integração de diversas tecnologias e, ao mesmo tempo, integrar diversos sistemas de negócios, com o objetivo de gerir contratos financeiros firmados junto a seus clientes.

Foi possível identificar que o Método ODP Ágil pode auxiliar devido a sua facilidade de desenvolvimento de sistemas distribuídos e o seu foco para a definição das arquiteturas.

Dentro deste escopo, o autor apresenta neste capitulo a implementação do Método ODP Ágil a fim de obter orientação para o desenvolvimento do sistema de Gestão de Contratos Consignados e suas arquiteturas.

4.3 Requisitos do Experimento

Este experimento teve como objetivo principal melhorar o processo de Gestão de Contratos Consignados propondo o desenvolvimento de um sistema levando em conta toda a complexidade do processo e principalmente atendendo os requisitos específicos da instituição financeira utilizando o Método ODP Ágil.

Esta proposta foi aplicada em uma instituição financeira nacional com matriz situada em São Paulo. Essa instituição não tinha seus contratos financeiros de Consignado gerenciados de forma eficiente, sendo assim um ambiente propicio a se aplicar um sistema para gestão de contratos. Com a aplicação visou-se ter uma gestão de contratos de Consignados eficiente (figura 29), identificando os contratos de Consignado passiveis de refinanciamento de forma clara e automatizar procedimentos hoje manuais como a indicação de promotoras, geração de base para telemarketing e relatórios gerenciais.

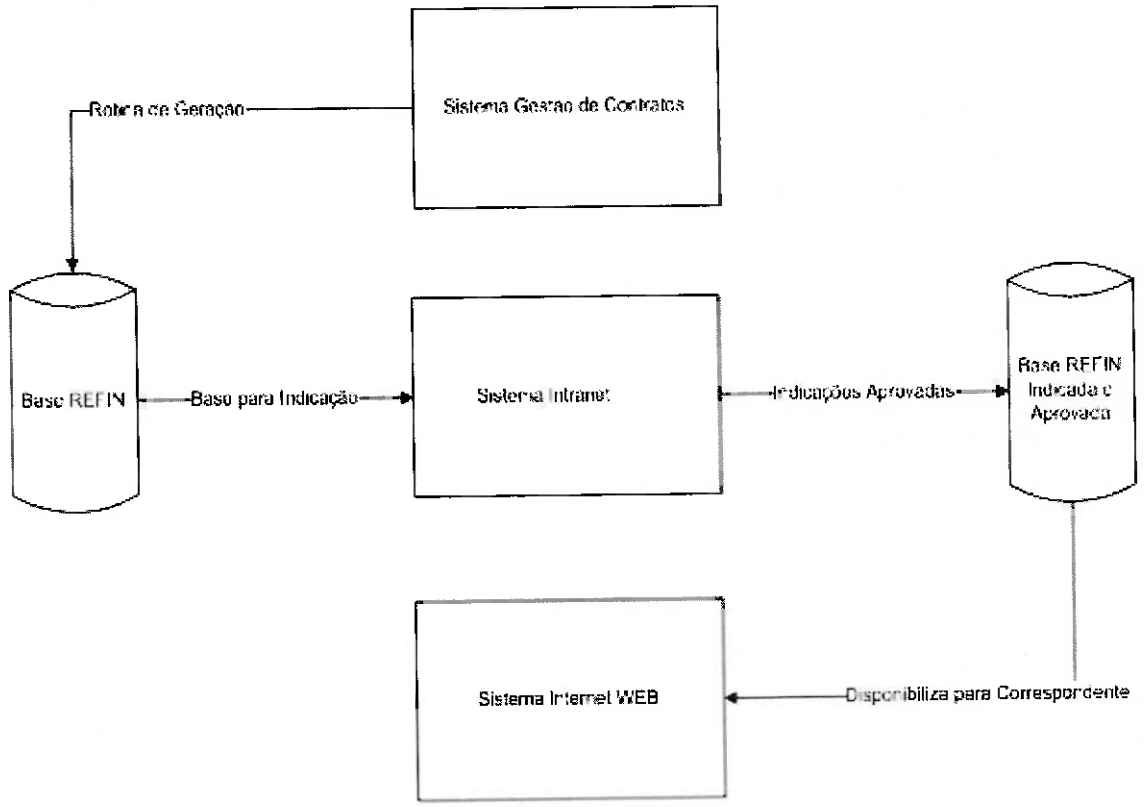


Figura 29 – Diagrama de Contexto.

4.4 Aplicação do Método ODP Ágil

Neste item as etapas do Método ODP Ágil foram desenvolvidas, tornando prática à aplicação do mesmo.

A fim de controlar o seu escopo, o experimento foi focado nas etapas do método onde foram inseridos e/ou modificados elementos cruciais seguindo assim a linha de resolver apenas o problema de definição das arquiteturas para o sistema. As etapas que não sofreram alterações foram ignoradas.

O desenvolvimento do sistema e suas etapas foram descritos de forma resumida (alguns pontos cruciais foram escolhidos), a fim de manter o foco do experimento e também de preservar a instituição financeira e seus dados confidenciais.

Como o projeto de desenvolvimento do sistema de gestão de contratos é grande e aborda vários pontos distintos, foi escolhido uma pequena parte para expor na forma de experimento. A parte escolhida foi à automatização da rotina para a geração de base de contratos passiveis de refinanciamento. Esta rotina faz com que contratos que possuam as características necessárias para um possível refinanciamento sejam automaticamente selecionados e atribuídos a uma promotora, para que a mesma entre em contato com o cliente titular do contrato.

A seguir apresenta-se a descrição das etapas onde foram definidas as arquiteturas para o sistema.

4.4.1 Etapa Fazer Explorações Iniciais

Esta etapa teve inicio com a definição dos requisitos iniciais. O cliente auxiliado pelo programador produziram os User Stories, descrevendo as definições dos requisitos neles. Levaram-se em conta as perspectivas de negócios da empresa para com o sistema.

Baseado nos User Stories criados, os programadores elaboraram o Diagrama de Casos de Uso, detalhando os requisitos e definindo o escopo do sistema. Também foi produzido o artefato BPMN do sistema. Por fim, o programador realizou testes com tecnologias e infraestrutura a fim de se obter uma análise básica da tecnologia necessária para o sistema e assim auxiliar na definição da arquitetura tecnológica.

Abaixo segue o como exemplo o Diagrama de Caso de Uso para a Geração da Base de Refinanciamento e na sequência o artefato BPMN, descrevendo os processos do sistema (como exemplo pode-se ver uma parte do modelo na figura 31). Com a definição destes artefatos tivemos a definição do ponto de vista Empresarial e assim obteve-se a Arquitetura de Processo de Negócio.

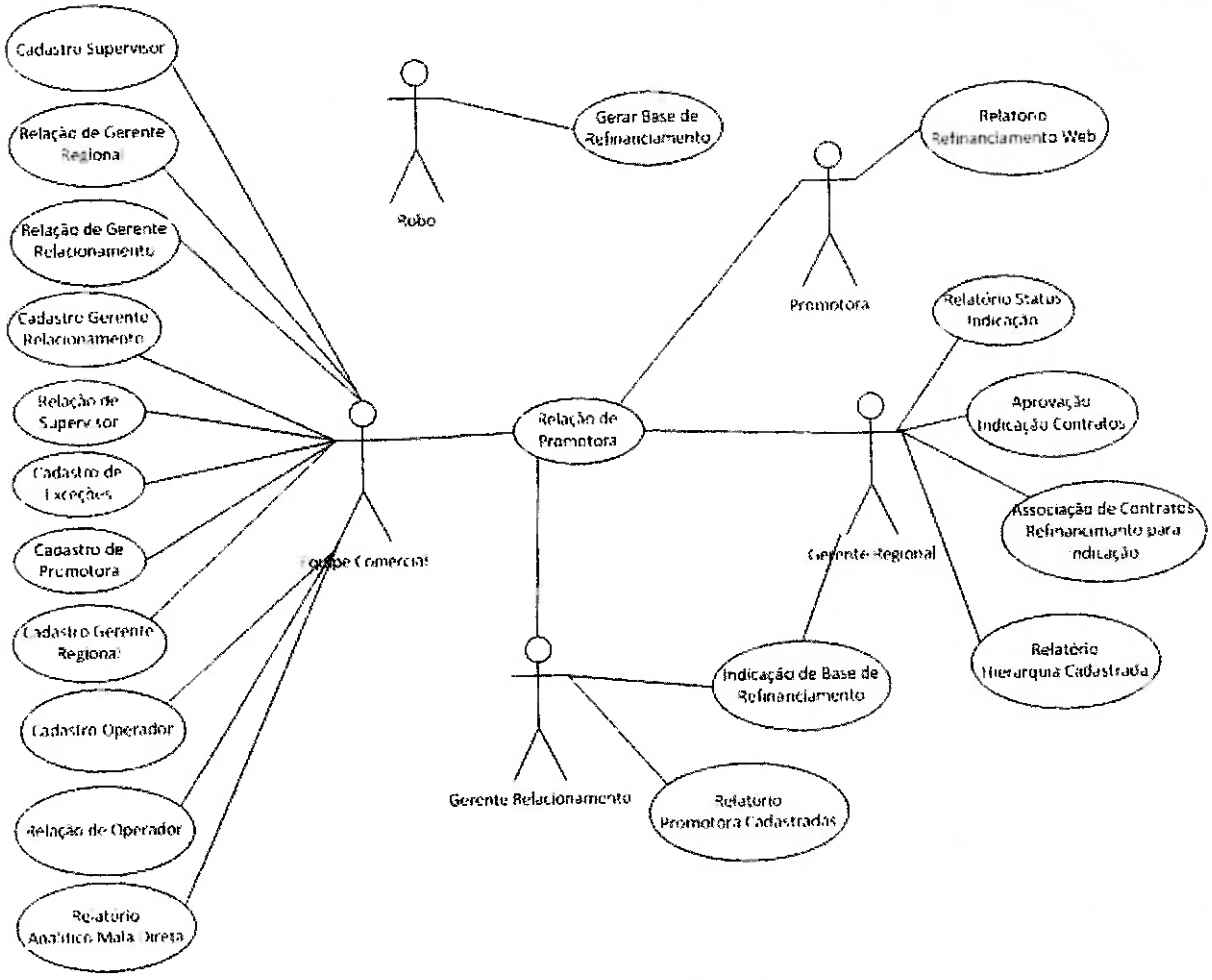


Figura 30 – Diagrama de Caso de Uso.

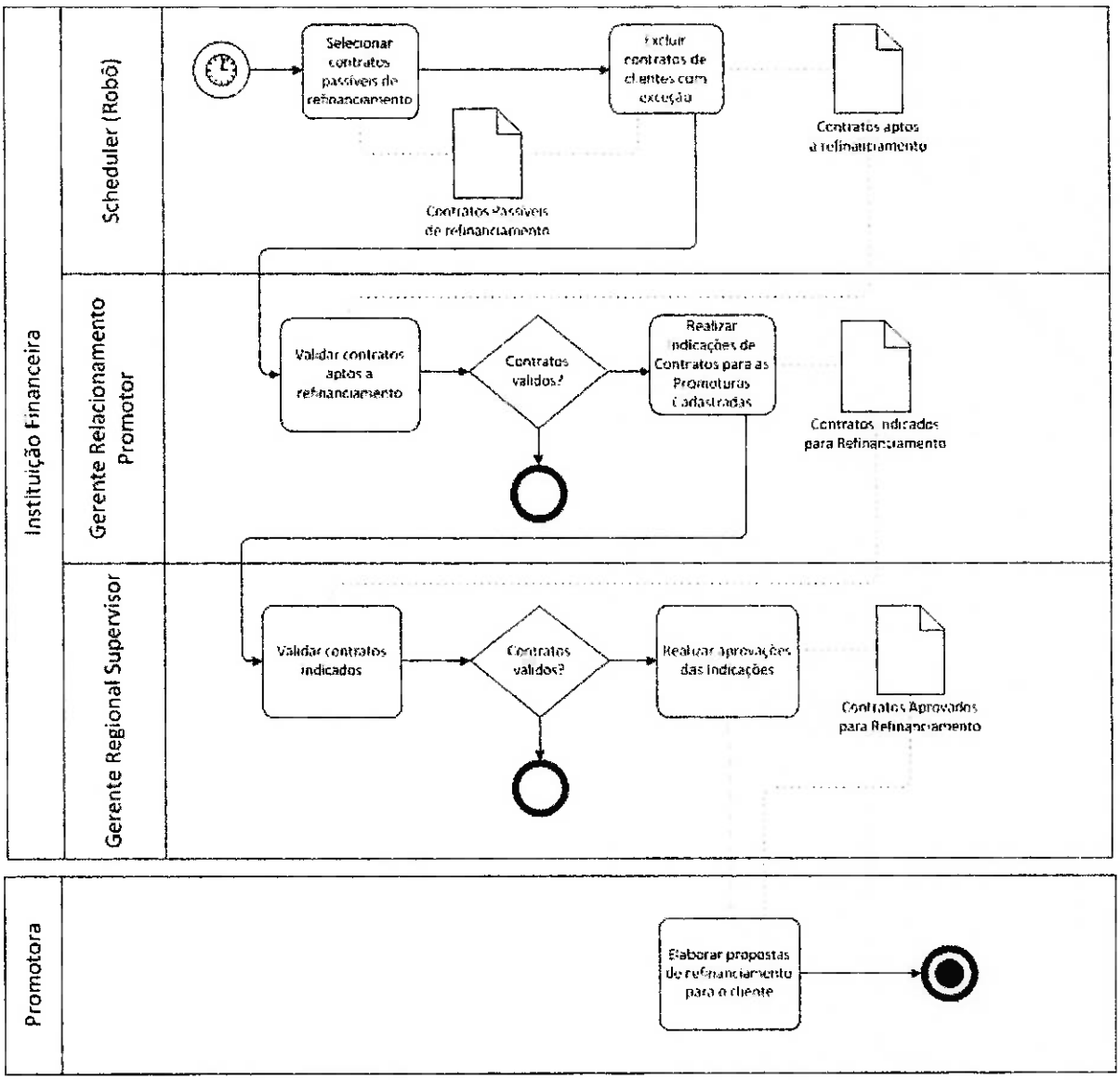


Figure 31 – Notação de Modelagem de Processos de Negócio (BPMN).

4.4.2 Etapa Definir e Revisar Requisitos

Esta etapa se iniciou com os clientes escrevendo e priorizando as histórias com o auxilio dos programadores, que definiram as estimativas de esforço para cada uma delas.

Posteriormente foram definidos os pontos de vista da Informação, Computação, Engenharia e Tecnologia sempre se levando em consideração a aderência à arquitetura de processo de negócio, definida na etapa anterior, Fazer explorações iniciais.

O primeiro ponto de vista definido foi o da Informação através do diagrama Entidade Relacionamento (como exemplo pode-se ver uma parte do diagrama na figura 32), descrevendo uma visão dos recursos que apoiam os requerimentos da informação. Com a definição do ponto de vista da Informação definido, obteve-se a Arquitetura de Dados.

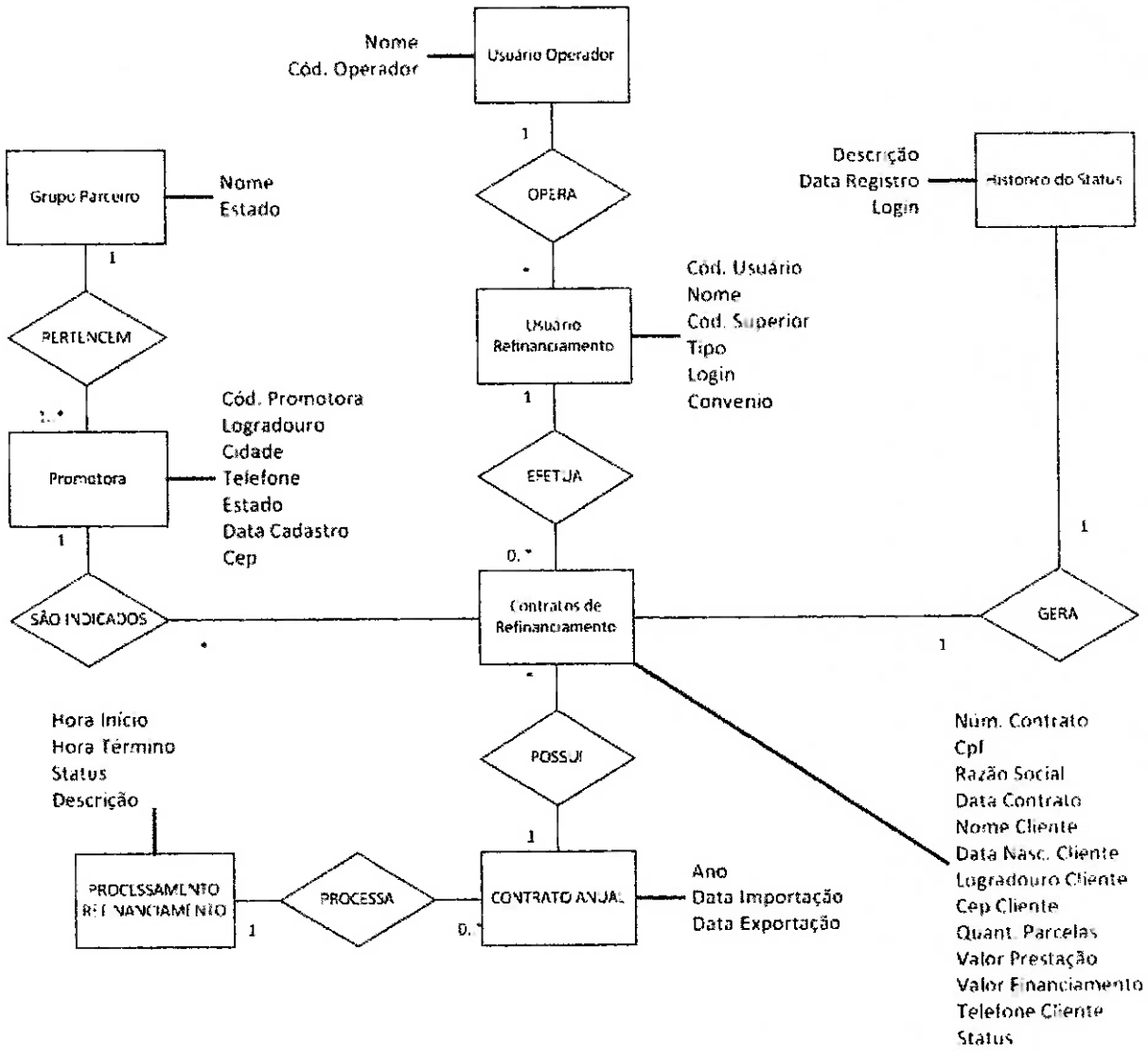


Figura 32 – Diagrama Entidade Relacionamento.

A definição do ponto de vista da Computação se deu através da elaboração do Diagrama de Classe (como exemplo pode-se ver uma parte do modelo na figura 33), onde se obteve a identificação dos objetos e interfaces, e os Diagramas de Sequência (figura 34 apresenta o Diagrama de Sequência de seleção de contratos). Com a definição do ponto de vista da Computação obteve-se parte da Arquitetura de Aplicações, que será completada com a definição do ponto de vista de Engenharia.

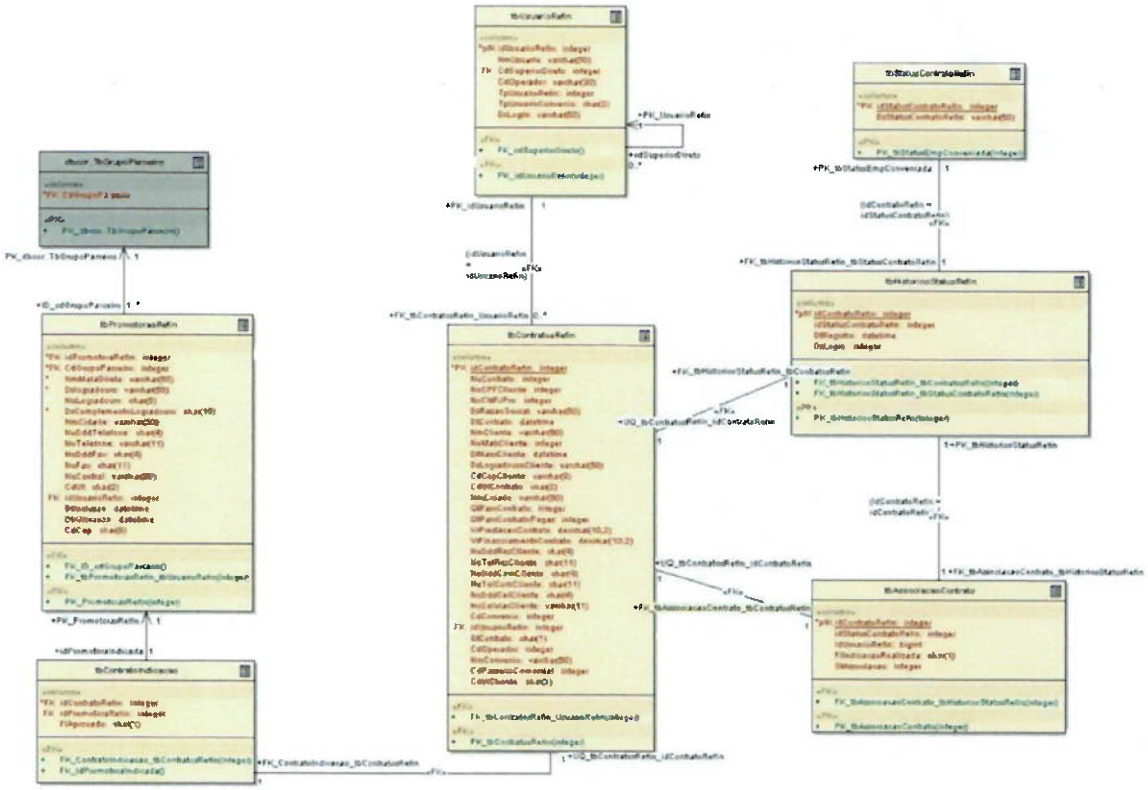


Figura 33 – Diagrama de Classe.

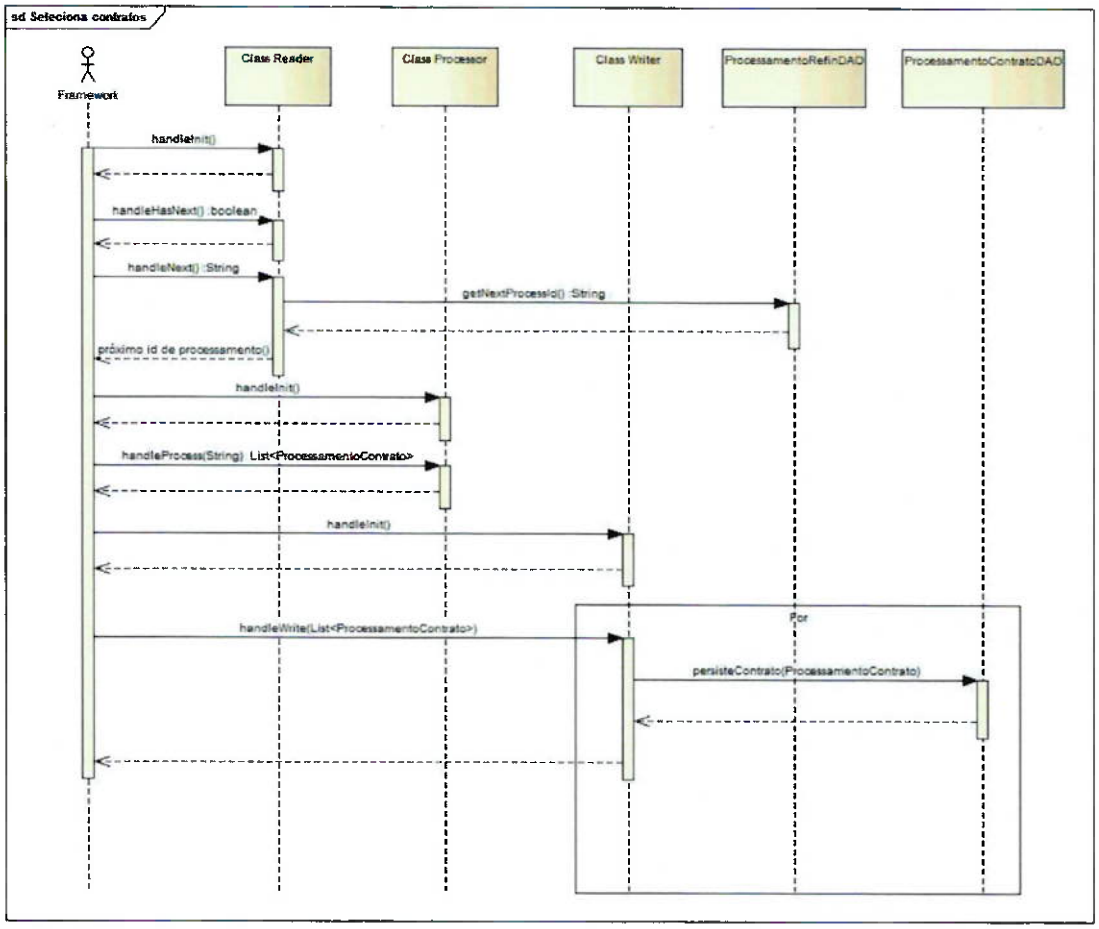


Figura 34 – Diagrama de Sequência Seleciona Contratos.

Já o ponto de vista da Engenharia foi definido através do Diagrama de Implantação (figura 35), onde se foi definido os mecanismos e as funções necessárias para suportar o sistema. Com a definição do ponto de vista da Engenharia juntamente com o ponto de vista da Computação definido anteriormente obteve-se a Arquitetura de Aplicações.

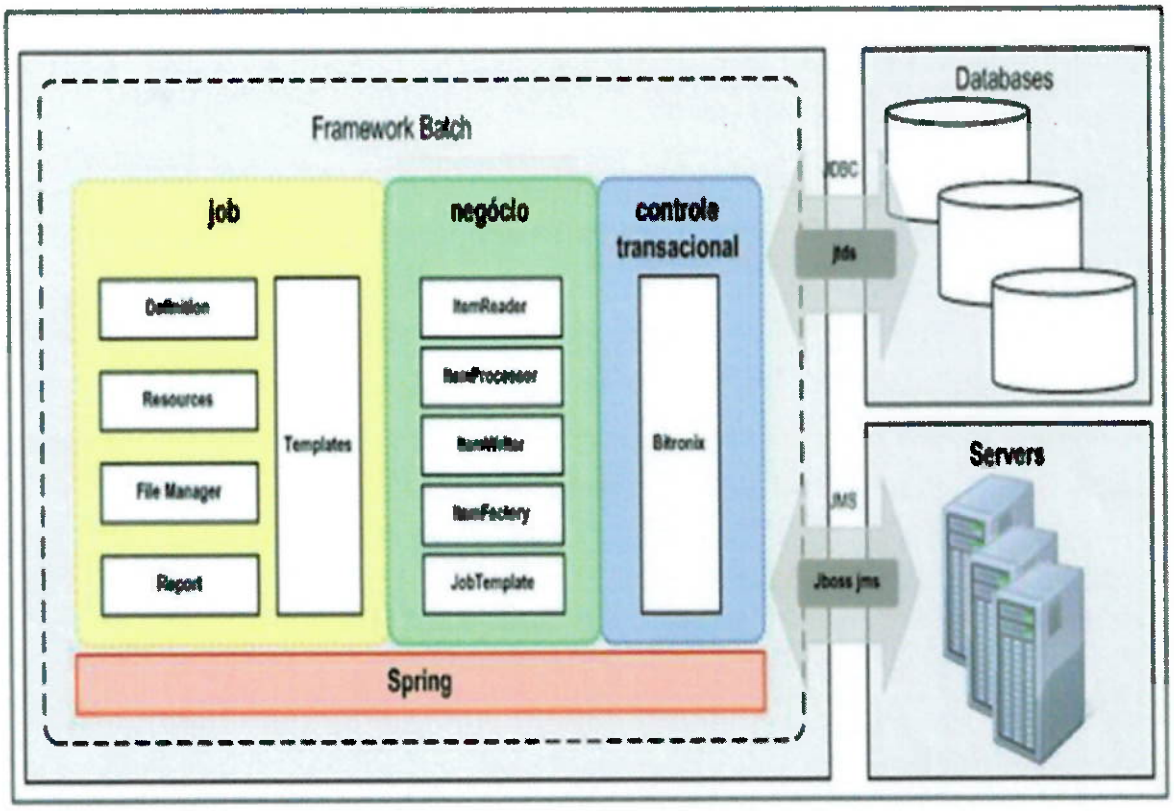


Figura 35 – Diagrama de Implantação.

Por fim, foi definido o ponto de vista da Tecnologia, onde foram definidos os padrões de especificação e a plataforma tecnológica (figura 36). Com a definição do ponto de vista da Tecnologia obteve-se a Arquitetura Tecnológica, tendo assim a Arquitetura do Sistema definida por completo.

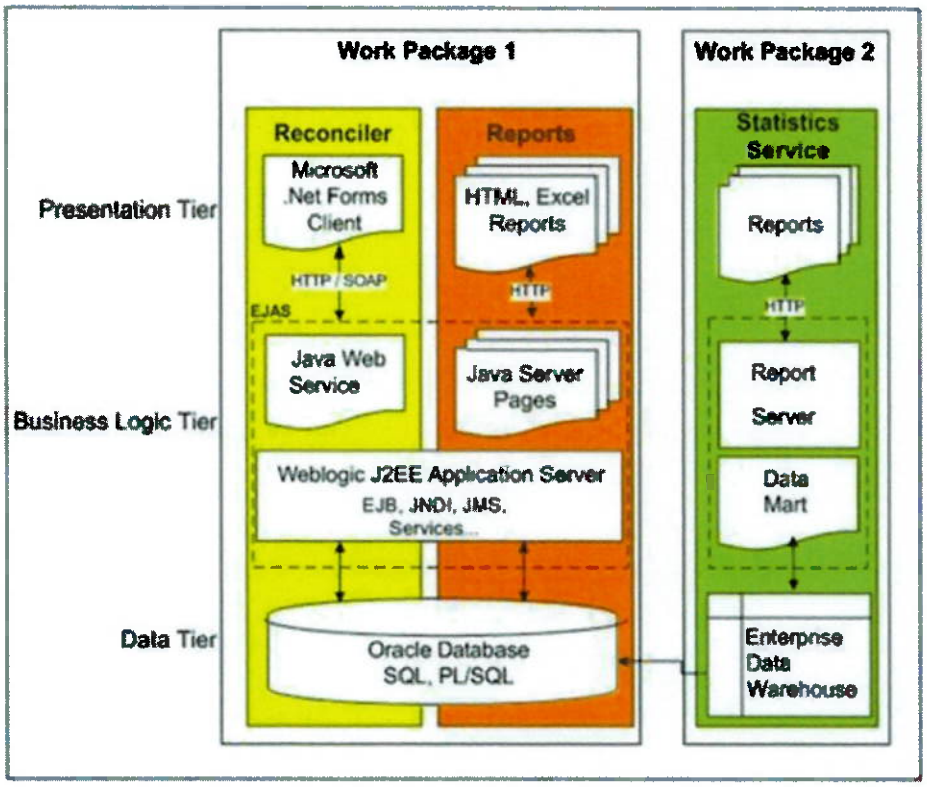


Figura 36 – Padrões de especificação e a plataforma tecnológica.

4.4.3 Etapa Planejar Release

Nesta etapa os programadores estimaram a dificuldade da implementação das histórias para que na sequência o cliente priorizasse aquelas que agregam maior valor ao negócio.

Ao mesmo tempo, a cada release criada, foi feita a revisão do ponto de vista da empresa, através da revisão do artefato BPMN, descrevendo os processos do sistema de acordo com o planejamento da release. Com isso temos a Arquitetura de Processo de Negócio sempre atualizada e controlada.

4.4.4 Etapa Planejar Iteração

Nesta etapa foram verificadas e refinadas as estimativas das histórias para definir quais serão implementadas na iteração seguinte. Com isso, os requisitos sofreram alterações e até mesmo novos requisitos foram incorporados para atender as necessidades do negócio. Como consequência, foram realizadas as revisões dos artefatos do ponto de vista da Empresa (Diagrama de Caso de Uso e BPMN),

mantendo assim o sistema sempre atualizado e de acordo com as mudanças constantes dos requisitos. Com as revisões realizadas temos a Arquitetura de Processo de Negócio sempre aderente e atualizada de acordo com o sistema a cada iteração realizada.

Com as revisões e estimativas dos requisitos concluídas, o cliente decidiu quais histórias cada iteração iria implementar concluindo a etapa.

4.4.5 Etapa Fazer Projeto

Na etapa Fazer Projeto teve-se dois momentos distintos: a passagem dela na primeira iteração e a passagem nela nas iterações restantes. Na primeira iteração, os artefatos criados anteriormente na etapa Definir e Revisar Requisitos são revisados através dos pontos de vista (Diagrama Entidade Relacionamento, Diagrama de Classe, Diagrama de Sequência, Diagrama de Implantação, Padrões de Especificação e a Plataforma Tecnológica) e atualizados caso necessário (devido as constantes mudanças nos requisitos). Uma vez revisados, as definições foram implementadas. É neste momento que são consolidadas as arquiteturas no sistema, atingindo assim a proposta da aplicação deste método: a definição da arquitetura do sistema através das arquiteturas Processos de Negócio, Dados, Aplicações e Tecnológica.

Nas iterações seguintes, os pontos de vistas são revisados devido aos novos requisitos ou por mudanças dos já existentes. Com isso, foi atingido o segundo objetivo da proposta deste método que é a absorção das constantes mudanças dos requisitos, mantendo os pontos de vista e seus artefatos sempre atualizados e de acordo com o sistema. Com isso as arquiteturas que compõem a Arquitetura do Sistema estarão sempre atualizadas e de acordo com o sistema a cada iteração realizada.

4.4.6 Análise e interpretação dos resultados

Nesta seção será mostrada uma análise quantitativa feita com base em dados coletados durante o experimento e também será apresentada uma análise qualitativa baseada nas observações diárias da equipe envolvida no experimento.

4.4.6.1 Análise quantitativa

Este experimento teve uma estimativa inicial de 1928 horas que representam o esforço gasto nas atividades realizadas. Esta estimativa foi feita baseando-se na previsão de termos 2 releases, cada uma com duas iterações cada e será implementada por uma equipe de 6 pessoas.

A primeira release foi estimada em 1000 horas e com isso nosso prazo para ela foi de 21 dias (considerando uma equipe de 6 pessoas). Ela foi concluída em 1080 horas (23 dias). Sendo assim, houve um atraso de 80 horas, contabilizando dois dias a mais (um erro de 8%). No gráfico 2 pode-se verificar a comparação do que foi estimado para com a evolução do desenvolvimento da primeira release.

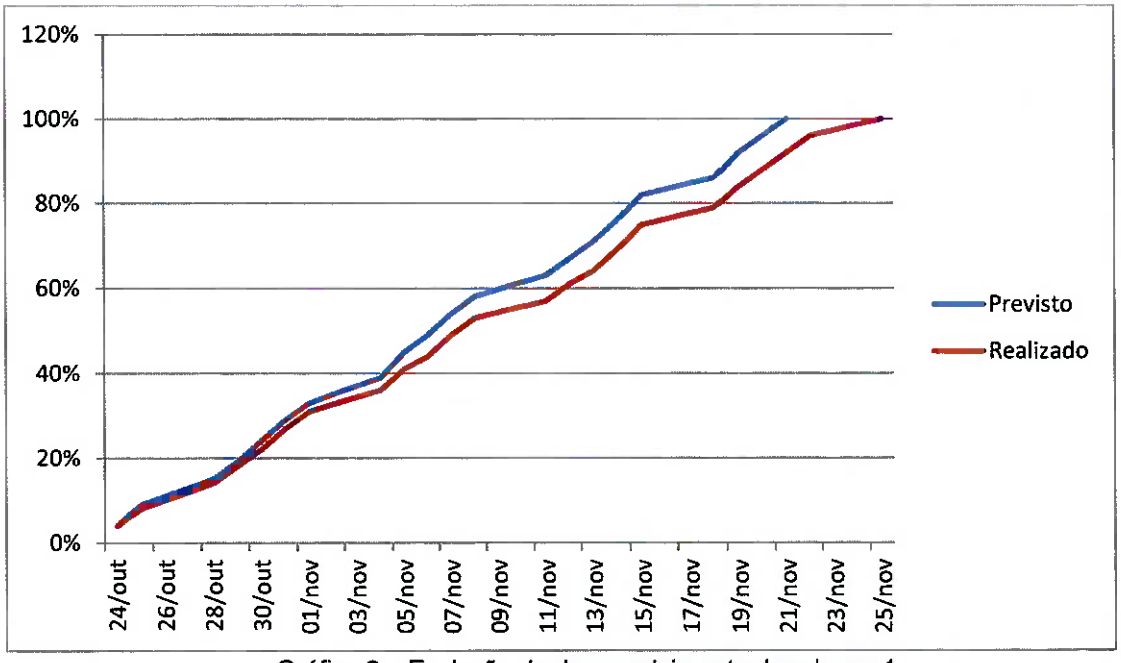


Gráfico 2 – Evolução do desenvolvimento da release 1.

A segunda release foi estimada em 928 horas e com o prazo de 20 dias (considerando uma equipe de 6 pessoas). Ela foi concluída em 1000 horas (21 dias). Sendo assim, houve um atraso de 72 horas, contabilizando um dia a mais (um erro de 7,75%). No gráfico 3 pode-se verificar a comparação do que foi estimado para com a evolução do desenvolvimento da segunda release.

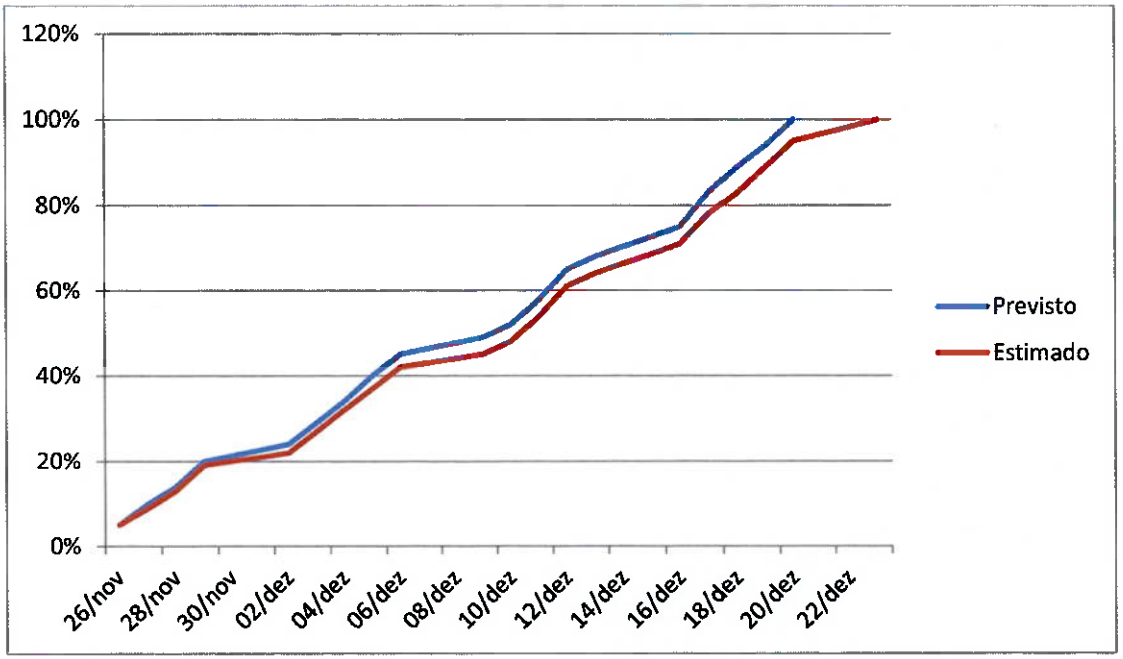


Gráfico 3 – Evolução do desenvolvimento da release 2.

4.4.6.2 Análise qualitativa

A análise qualitativa se baseou nas observações e conversas do dia a dia durante o desenvolvimento do sistema. Uma reunião com todos envolvidos também foi feita após a conclusão e a principal intenção era obter um feedback sobre o sistema e a metodologia utilizada no desenvolvimento.

O foco eram pontos como benefícios, desvantagens e facilidade. Com isso, chegou-se a alguns resultados importantes:

- Todos da equipe mostraram-se satisfeitos com a metodologia e demonstraram interesse em utiliza-la novamente, assim como auxiliar na evolução da mesma;
- As principais vantagens foram o menor rigor da documentação em relação a outros métodos como o RUP por exemplo e uma maior e melhor interação dentro da equipe. Além disso, a presença constante do cliente junto à equipe traz uma maior segurança do que deve ser feito e de que forma ser feito a fim de agregar valor ao negócio;
- A definição da arquitetura do sistema é de suma importância para o melhor entendimento e andamento do desenvolvimento do sistema, assim como o

controle da arquitetura na evolução do sistema. Esse é um fator considerável no fator de facilidade na manutenção, auxiliando na rapidez e clareza do atendimento;

- Como desvantagem foram citados pontos como a falta de experiência na utilização de métodos ágeis e até mesmo na evolução/manutenção constante da arquitetura do sistema. Fato este compreensível já que atualmente a empresa não faz uso de metodologias ágeis no desenvolvimento.

Por fim, o experimento se mostrou de grande importância para a validação do Método ODP Ágil, mostrando que apesar da adaptação de diversas etapas para a definição da arquitetura do sistema, não foi perdida a característica de um processo ágil, possibilitando a construção de sistemas de boa qualidade, segurança e com agilidade.

5 CONSIDERAÇÕES FINAIS

Neste capitulo serão apresentadas as principais conclusões encontradas neste trabalho de pesquisa, assim como suas contribuições, recomendações e trabalhos futuros que poderão ser realizados a partir desta pesquisa.

5.1 Conclusões

O desenvolvimento desta monografia culminou com a elaboração de um método de desenvolvimento ágil para definição de arquiteturas de sistemas.

O método proposto tem como objetivo especificar uma forma de desenvolvimento ágil através da metodologia XP integrado ao padrão RM-ODP para a definição da arquitetura do sistema.

Com o relacionamento destes dois padrões, conclui-se que o Método ODP Ágil se mostrou uma ideia viável, pois se utiliza dos pontos de vista do padrão RM-ODP para a definição da arquitetura do sistema dentro de um processo de desenvolvimento de software ágil que é o XP.

Com isto, agregou-se de forma consistente e confiável a definição da arquitetura do sistema através de um conjunto de pontos de vista do padrão RM-ODP dentro das etapas do método XP, tido como base. Tem-se então no Método ODP Ágil características dos dois padrões, sendo as principais:

- Melhor definição do negócio envolvido (característica do padrão RM-ODP);
- Suporte para a integração da distribuição, interoperabilidade e portabilidade ideais para o sistema (característica do padrão RM-ODP);
- Iterações curtas, possibilitando mudanças rápidas sem que o custo cresça durante o desenvolvimento (característica do padrão XP);
- A utilização de práticas, valores e princípios (característica do padrão XP).

Durante o experimento prático, perceberam-se alguns pontos positivos e negativos, como:

- Uma maior facilidade de implementação do sistema por parte dos programadores devido à definição dos pontos de vista e seus artefatos, reduzindo assim o esforço para a implementação do sistema;
- A definição dos pontos de vista atrelados às etapas do desenvolvimento fez com que os esforços aumentassem, já que programadores tiveram que executar tais tarefas;
- A definição dos pontos de vista não fez com que fosse perdida a agilidade durante o desenvolvimento de software, mantendo a possibilidade de mudanças sem o aumento do custo ao longo do desenvolvimento do sistema;
- Feedback sobre o produto e boa comunicação entre o cliente e desenvolvedor é primordial para a agilidade aos assuntos durante o desenvolvimento;

Por fim, obteve-se um resultado satisfatório da utilização do método para o que ele propõe, uma vez que se obteve sucesso ao desenvolver um sistema de forma ágil com a definição de sua arquitetura, mantendo a qualidade e um controle maior quanto ao seu valor junto ao negócio, facilitando inclusive sua manutenção.

5.2 Principais contribuições

Neste trabalho foi proposto um método ágil para a definição de arquitetura de sistemas. O Método ODP Ágil foi desenvolvido baseado no método ágil de desenvolvimento XP e o padrão RM-ODP e procurou levar em consideração as competências e deficiências de cada uma a fim de integra-las para a definição da arquitetura de sistemas.

O método Extreme Programming é uma metodologia ágil popular e que tem como características a interação constante, pouca documentação, pequenas equipes e projetos simples, favorecendo assim a dinâmica inerente ao desenvolvimento de sistemas.

Já o padrão RM-ODP apresenta uma excelente alternativa para a especificação de arquiteturas de sistemas abertos e distribuídos através de suas diferentes visões computacionais.

O Método ODP Ágil para a definição de arquitetura de sistemas distribuídos foi criado procurando respeitar os princípios e práticas do XP para que suas características fossem mantidas. As alterações realizadas para a integração com o padrão RM-ODP visaram tratar as questões de definição da arquitetura do sistema.

Com isso, o método proposto constitui uma novidade com relação a simples utilização do método de desenvolvimento XP, porque é capaz de modelar a arquitetura do sistema através da integração com o padrão RM-ODP e seus pontos de vista.

O experimento conduzido mostrou que as integrações entre os padrões não representaram a perda das características básicas do padrão XP, pois não apresentaram um aumento significativo no esforço empregado pelas equipes. Em contrapartida, ficou evidente que a integração com o padrão RM-ODP é adequada para o desenvolvimento de sistemas distribuídos, pois a questão da definição da arquitetura do sistema é de fundamental importância para atender as expectativas do cliente.

Essa monografia apresenta uma proposta prática que pode ser utilizada pelas indústrias de software.

5.3 Trabalhos futuros

Com este trabalho acredita-se ser possível iniciar uma série de pesquisas relacionadas à engenharia de software a partir do refinamento e detalhamento deste método, procurando focar em pontos específicos como, por exemplo, disponibilidade e segurança.

Outro ponto que pode se desenvolvido diz respeito à criação do Método ODP Ágil a partir do relacionamento entre a metodologia ágil XP e o padrão RM-ODP. Um tema relevante seria uma pesquisa que busque criar e/ou aprimorar o método utilizando outros métodos, como, por exemplo, SCRUM, TOGAF entre outras.

No que diz respeito à validação do Método ODP Ágil pode ser interessante à aplicação do mesmo em experimentos mais longos, de áreas diferentes, com

equipes e diretrizes distintas. Assim será possível avaliar possíveis dificuldades nestas circunstancias.

REFERÊNCIAS

BANCO CENTRAL DO BRASIL. **Empréstimos Consignados.** www.bcb.gov.br - Ago., 2013.

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software architecture in practice**. Addison-Wesley, 2003, p. 528.

BECK, K. **Embrancing change with extreme programming**. Version 32. IEEE Computer, 1999, p. 70-77.

BEM, G.; HASHIMOTO, A. M. **InfoGEO**. Edição 62. MundoGEO, 2010, p. 38.

CHEN, P. P. **The entity-relationship model-toward a unified view of data**. ACM Transactions on Database Systems – portal.acm.org, 1976.

GALASINI, M. **O método ODP-UP para a definição de arquiteturas de sistemas distribuídos.** São Paulo, 2004. Dissertação (mestrado) – Escola Politécnica da Universidade de São Paulo.

GUNGI, T. **Processamento distribuído aberto:** ODP. Campinas, 1995.

ISO. Recommendation X.901/ISO/IEC 10746-1 : Basic Reference Model of Open Distributed Processing – Part 1 : Overview and Guide to Use, 1998.

ISO. Recommendation X.902/ISO/IEC 10746-2 : Information Technology – Open Distributed Processing – Referencence Model : Foundations, 1996.

ISO. Recommendation X.903/ISO/IEC 10746-3 : Information Technology – Open Distributed Processing – Referencence Model : Architecture, 1996.

KHUSIDMAN, V.; ULRICH, W. **Architecture-Driven modernization**: transforming the enterprise. Burlington, 2007.

KUHN,G. R.; PAMPLONA, V. F. **Apresentando XP**: encante seus clientes com Extreme Programming. Blumenau, 2004.

ROSA, F. A. J. **Método de modelagem de arquitetura corporativa.** São Paulo, 2008. Dissertação (mestrado) – Escola Politécnica da Universidade de São Paulo.

SAMPAIO, A. T. F. **Xwebprocess**: um processo ágil para o desenvolvimento de aplicações web. Recife, 2004. Dissertação (mestrado) – Universidade Federal do Pernambuco.

SOARES, M. S. **Metodologias ágeis Extreme Programming e Scrum para o desenvolvimento de software.** Conselheiro Lafaiete, 2004.

SOARES, M. S. **Comparação entre metodologias ágeis e tradicionais para o desenvolvimento de software.** Conselheiro Lafaiete, 2004.

SOMMERVILLE, I. **Engenharia de software.** 8° edição. São Paulo: Pearson Addison Wesley, 2011.

WEILL, P.; ROSS, J. W.; ROBERTSON, D. C. **Enterprise architecture as strategy**: Creating a foundation for business execution. Harvard Business School Press, 2006.

ZANI, V. A. T. **Agira:** um processo ágil de desenvolvimento de software baseado em arquiteturas de referência. São Carlos, 2013. Dissertação (mestrado) – Instituto de Ciências Matemáticas e de Computação – ICMC-USP.