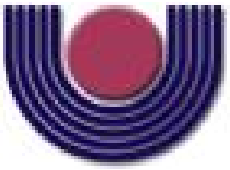


Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

Desenvolvimento da ferramenta LiProMA para o auxilio na construção de escopo de linhas de produto com o apoio de metodologias Ágeis

Tharle Josefi de Camargo

CASCAVEL
2013

THARLE JOSEFI DE CAMARGO

DESENVOLVIMENTO DA FERRAMENTA LIPROMA PARA O AUXILIO NA CONSTRUÇÃO DE ESCOPO DE LINHAS DE PRODUTO COM O APOIO DE METODOLOGIAS ÁGEIS

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Prof. Dr. Ivonei Freitas da Silva

CASCAVEL
2013

THARLE JOSEFI DE CAMARGO

Desenvolvimento da ferramenta LiProMA para o auxilio na construção de escopo de linhas de produto com o apoio de metodologias Ágeis

Monografia apresentada como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Dr. Ivonei Freitas da Silva (Orientador)
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Dr. Victor Francisco Araya Santander
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Me. Elder Schemberger
Colegiado de Ciência da Computação,
UNIOESTE

Cascavel, 5 de dezembro de 2013

DEDICATÓRIA

Dedico este trabalho ao meu namorado Henrique Costa Rakssa pela sua dedicação, pelo seu companheirismo e por me fazer acreditar no amor verdadeiro.

Somos o que há de melhor. Somos o que dá pra fazer. O que não dá pra evitar e não se pode escolher. (Humberto Gessinger)

AGRADECIMENTOS

Primeiramente, agradeço aos meus professores, por todo conhecimento compartilhado, pela amizade e dedicação do seu tempo e por ser a minha segunda família durante o tempo da minha graduação. Ao professor Ivonei Freitas da Silva pelas suas orientações e ajudas nesse trabalho e, principalmente, por acreditar no meu potencial.

Agradeço à minha Mãe, Nilda, pelo apoio, incentivo, tanto financeiro quanto emocional, e por ser a melhor mãe do mundo. Agradeço também às minhas irmãs, Camila e Tatiane, que eu amo, apesar de nossos eventuais desentendimentos e diferenças, sempre estiveram ao meu lado não importando a situação.

Aos meus amigos e colegas de curso, que, nesses anos, me ajudaram nos estudos, aguentaram amargamente os meus "comentários hilários", e, principalmente, por trazer sorrisos ao meu rosto nos momentos que mais precisei.

Agradeço ao Anderson Roberto Slivinsk e ao Julio Cesar Lazzarim, aos quais foram os melhores amigos que tive durante esses anos de curso. Eu nunca irei esquecer o que vocês fizeram por mim.

E a todos que, de alguma forma, me auxiliaram durante minha formação.

Muito Obrigado!

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AD	*Análise de Domínio.*
AGM	*Arcade Game Maker.*
CASE	*Computer-Aided Software Engineering*
DAO	*Data Acess Object.*
EA	*Engenharia de Aplicação.*
ED	*Engenharia de Domínio.*
EJB	*Enterprise JavaBeans.*
JDBC	*Java Data Base Conector.*
JSON	*JavaScript Object Notation*
LiProMA	*Linha de Produtos e Metologias Ágeis.*
LPS	*Linha de Produto de Software.*
MVC	*Model View Control.*
PC	*Computador Pessoal.*
SD	*Diagrama de Dependências Estratégicas.*
SR	*Diagrama de Estratégias Internas.*
Xml	*eXtensible Markup Language.*

Sumário

Resumo

Desenvolvedores de sistemas são obrigados a enfrentar, no dia a dia, uma competição no mercado. Essa competição muitas vezes envolve a diversidade das necessidades de mercado, o crescente aumento e diversidade das *features* dos seus produtos, as rápidas mudanças de tecnologias sem contar com as pressões na entrega do produto. Para se adequar a essas adversidades, é útil um processo para mapear as *features* dentro de uma linha de produto, identificando as *features* principais junto a seus domínios, da Silva [6] propõe o uso de práticas do método ágil Scrum. Em um processo que envolve gerar um mapa de produto da LPS com o apoio das interações das *Sprints* do Scrum.

Entretanto, as ferramentas disponíveis [7][8] não oferecem suporte completo a essa abordagem. Esse trabalho apresenta o desenvolvimento de uma ferramenta chamada LiProMA (Linha Produto e Metodologias Ágeis) para oferecer o apoio a construção de escopo de uma linha de produto juntamente com a metodologia ágil Scrum.

Palavras-chave: *Linhas de Produto de Software, LPS, Metodologias Ágeis, Scrum, Escopo de Produto*.

Capítulo 1

Introdução

1.1 Contexto

Linhas de produto de software (LPS) possibilitam o gerenciamento de um conjunto de sistemas similares por meio da análise de similaridade e variabilidade [9] [2]. A engenharia de LPS possui dois ciclos de desenvolvimento de software, um para o desenvolvimento para reuso, e, outro para o desenvolvimento com reuso [2]. Como há um processo de engenharia para e com reuso, atividades como, engenharia de requisitos, arquitetura, implementação, testes e evolução de artefatos estão presentes para o desenvolvimento da linha.

A primeira atividade que deve ser realizada ao construir uma linha de produto de software é o escopo. O escopo define as características e os procedimentos que caracterizam os produtos de uma linha em domínio [10]. Essa atividade define quais produtos de software serão endereçados pela linha e especifica quais *features* (características funcionais e não funcionais visíveis pelo usuário) formarão esses produtos através da análise de similaridade e variabilidade. É também uma atividade de tomada de decisão referente aos produtos e *features* com a finalidade de gerar um escopo de linha de produtos similares que seja economicamente otimizado para a organização [6].

Esses produtos e *features* estão dentro de um domínio, por exemplo, domínio médico ou domínio de telecomunicações. Muitos produtos e *features* podem ser identificados para o domínio, mas somente os produtos e *features* que interessam ao negócio da organização e que ofereçam bom potencial de reuso devem ser especificados na atividade de escopo da linha.

Normalmente, há muitas *features* que devem ser identificadas, analisadas e especificadas gerando desmotivação e esforço duplicado se o domínio não é tão estável e maduro ou se a

organização não tem bem definido como deve ser o escopo. Neste cenário de instabilidade ou imaturidade do domínio ou da falta de conhecimento da organização das variabilidades que devem ser implementadas na linha, é necessário o uso de técnicas que melhor lidem com este tipo de contexto. Métodos ágeis têm sido adotados com linhas de produto de software com o objetivo de endereçar cenários não tão estáveis e maduros para aplicar engenharia de LPS tradicional [11].

1.2 Motivação e Problema

Linhas de produto de software desenvolvem um conjunto de *assets* de software e documentação para reuso sobre os produtos que constituem a linha de produtos, na qual cada um dos *assets* são definidos como artefatos de qualquer natureza, gerados em qualquer momento do processo de desenvolvimento e capturam conhecimentos que são importantes para a organização e, consequentemente, possuem um valor potencial [12] com uma análise, projeto, implementação e testes "*up-front*"[2]. Por outro lado, métodos ágeis focam no desenvolvimento de código enquanto reduz a análise e projeto *"upfront"* reduzindo o *overhead* do processo de desenvolvimento.

Algumas iniciativas estão em desenvolvimento para incluir os processos ágeis em desenvolvimento em larga escala, incluindo linhas de produtos [13][14][15][16][17][18][19][20][21], porém os resultados, embora tenham algum valor empírico, ainda precisam ser confirmados em outros trabalhos de replicação e novos estudos reais na indústria. Da integração de LPS com métodos ágeis espera-se benefícios como a redução de tempo de entrega de produtos, melhorias na qualidade, redução no custo de desenvolvimento e manutenção, redução de barreiras de adoção de LPS, etc.

Em especial a redução da barreira de adoção de LPS para pequenas e médias empresas, essa integração pode ser fundamental. Essa integração pode resultar em processos, com apoio ferramental, mais leves e apropriados para um domínio não tão maduro e com requisitos voláteis.

Em um cenário na qual questões de gestão em LPS geram problemas para a adoção, o método Scrum pode fornecer estratégias que permitam um gerenciamento mais eficaz para lidar com domínios não tão maduros e requisitos voláteis, bem como, para pequenas e médias empresas.

1.2.1 Ferramentas existentes

Dentro do cenário apresentado nessa proposta, existem ferramentas que parcialmente cobrem o método citado. Essas ferramentas são as mais ativas dentro do cenário científico e empresarial, no qual o seu objetivo é auxiliar a Engenharia de Domínio. A tabela 1.1, baseada no estudo feito em [8], aponta similaridades e diferenças entre elas, incluindo a ferramenta LiProMA que é proposta nesse trabalho.

Tabela 1.1: Ferramentas existentes e um Funcionalidades

Funcionalidades/ Ferramentas	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
Feature-IDE			x	x	x	x			x	x		x					
Pure :: Variants			x	x	x	x	x	x	x	x		x					
ToolDay	x	x	x	x	x	x	x	x	x	x	x	x					
LiProMA	x	x	x	x	x	x	x	x	x				x	x	x	x	x

1. **Documentação de análise de mercado:** Armazenamento de informações do mercado que buscam ajudar a identificação de quais características fazem parte do domínio dentro de uma LPS.

2. **Definição de escopo:** Identifica as *features* que devem fazer parte do domínio.

3. **Representação do domínio:** Representação do escopo definido por meio de tabelas, modelos árvores, entre outros.

4. **Variabilidade:** Representação da variabilidade que a *feature* pode assumir.

5. ***Feature* Obrigatória:** Representa *features* que sempre irão fazer parte do produto.

6. **Feature Pai:** Possibilidade de cadastrar hierarquicamente as *features* e *subfeatures* no sistema.

7. **Documentação de domínio:** Providencia a documentação para o domínio, normalmente consistem em descrição, contexto, dependências entre o sistema e o domínio, entre outras. Cada ferramenta tem um ponto de vista diferente para essa documentação.

8. **Documentação das *features*:** Providencia uma documentação para cada *feature* dentro do domínio, com informações a respeito de nome, descrição, prioridades, etc. Cada ferramenta tem um ponto de vista diferente para essa documentação.

9. **Mapa de produto:** Identifica quais *features* do domínio estão relacionadas com o produto.

10. **Modelo de *feature* em grafos:** A existência de uma interface gráfica interativa que exibe a hierarquia das *features* em forma de grafo. Faz da experiência mais intuitiva para os usuários.

11. **Importação e Exportação de outras aplicações:** Importar ou exportar documentos genéricos para outros tipos de arquivos, como XML, XMI, PDF, XLS, JPG entre outros.

12. **Geração de código-fonte:** Permite que, de acordo com a Engenharia de domínio seja possível gerar códigos fontes automaticamente de *features* cadastradas no sistema.

13. **Dados em um Banco de dados:** Os dados estão armazenados em um Sistema Gerenciador de Banco de Dados (SGBD).

14. **Múltiplas LPS:** É possível cadastrar mais de uma LPS no sistema dentro da mesma base de dados.

15. **Integração com o Scrum:** Para o apoio no gerenciamento do escopo, a ferramenta fornece estrutura para trabalhar com o método ágil Scrum.

16. **Documentação do Scrum:** Documentação do processo completo efetuado pela metodologia ágil Scrum, junto com as suas *sprints*, tarefas, responsáveis e etc.

17. **Plataforma Web:** A ferramenta não está restrita a um único dispositivo e sim pode operar em um servidor remoto e ser acessada por diversos usuários.

Ao ser comparada com outras ferramentas em atividades, a ferramenta LiProMA possui algumas limitações como não conter geração de códigos fontes, falta um modelo de *features* disposto em grafo e ser limitada a formulários e tabelas, importação de arquivos de terceiros ou até mesmo ser limitada ao gerenciamento de escopo, todavia, a ferramenta proposta contém

um conjunto interessante funcionalidades distintas. Essas funcionalidades foram pensadas para suprir as necessidades de uma única ferramenta de apoio ao método de LPS com o apoio do método ágil Scrum, tais como o gerenciamento e documentação das *sprints* vinculadas as *features* e produtos cadastrados no sistema. A LiProMA também oferece como diferencial o fato de ser uma ferramenta Web, ou seja, os dados cadastrados são armazenados em um servidor, que podem ser acessados por diferentes computadores e em diferentes lugares, além de fornecer uma centralização da informação.

1.3 Objetivo

Este trabalho tem o objetivo de apresentar uma proposta e implementação de uma ferramenta que visa automatizar tarefas no método que integra a metodologia ágil Scrum com LPS proposto por da Silva[6], mais especificamente na etapa de gestão escopo da LPS. A ferramenta visa automatizar tarefas que são específicas da parte do escopo.

Capítulo 2

Linhas de Produto de Software

2.1 Introdução

Uma linha de produto de software (LPS) é um conjunto de sistemas de software formado pela agregação de suas *features* em comum, as quais tem por objetivo acatar as necessidades de um determinado segmento de mercado ou escopo que são desenvolvidos na mesma base (*core*) [2].

Para Czarnecki e Eisenecker[22], uma *feature* é uma característica funcional e não funcional visível do sistema que possui grau de importância para os *stakeholders* (compreende todos os envolvidos em um processo [23]). Uma *feature* pode derivar em outras *subfeatures* visando o seu melhor detalhamento, como por exemplo, temos uma *feature* "Gerenciamento de Paciente", entretanto o especialista decide que essa *feature* por sua vez pode ser divida em outras *subfeatures* como "Cadastro de Paciente", "Agendamento de Pacientes", entre outras, e cada uma dessas *subfeatures* possui a *feature* "Gerenciamento de Paciente"como sua *feature* pai.

A abordagem LPS concentra-se na utilização de técnicas da engenharia de software para possibilitar a criação de um grupo de sistemas de softwares com *features* similares partindo de um conjunto de *features* já existentes.

É possível, junto da análise de similaridade e variabilidade das *features* dos sistemas, definir e construir *core* com *features* comuns a todos os produtos estudados, de modo que possa ser utilizado no desenvolvimento de todos os sistemas do grupo.

Segundo Clements[9], o objetivo das LPS é juntar semelhanças e gerir as variações, que originam em melhorias na qualidade, tempo e esforço de desenvolvimento, custo e complexidade reduzidos na criação e manutenção dos produtos.

A LPS é vantajosa ao longo de todas as etapas da engenharia de software, entretanto ao ser aplicada na análise e especificação de requisitos, sendo que por sua vez, são elaborados generalizando os requisitos para um grupo de sistemas em geral que serão usados em um nível genérico de construção. Assim fica mais facilitado a tarefa de documentar as *features* comuns e, por conseguinte, prover melhor similaridade e variabilidade nos sistemas e a reutilização estratégica do software. A LPS pode é divida em dois subprocessos: Engenharia de Domínio e Engenharia de Aplicação [2].

Enquanto a Engenharia de Domínio (ED) trabalha com o processo de construção de sistemas pertencentes a uma família de aplicações reusáveis em um domínio em particular de problema, a Engenharia de Aplicação (EA) visa construir aplicações específicas em um dado domínio reusando artefatos de software gerados da ED [24].

2.2 Engenharia de Domínio

A Engenharia de Domínio tem por objetivo prover e definir a simplicidade, a variabilidade e o escopo de uma LPS e definir e construir artefatos reusáveis que possam gerar a variabilidade desejada [2].

Os objetivos da engenharia de domínio são acompanhados por sub processos. Esses sub processos existem para detalhar e refinar a variabilidade determinada pelo procedimento do sub processo e prover *feedback* sobre a viabilidade de realizar a variabilidade requerida no sub processo [2].

A ED é constituída por quatro sub processos: Análise, Projeto, Código e Teste de Domínio. Através desses sub processos a engenharia de domínio gera os requisitos, a arquitetura, os componentes e os testes no domínio do sistema [2]. Veja figura 2.1.

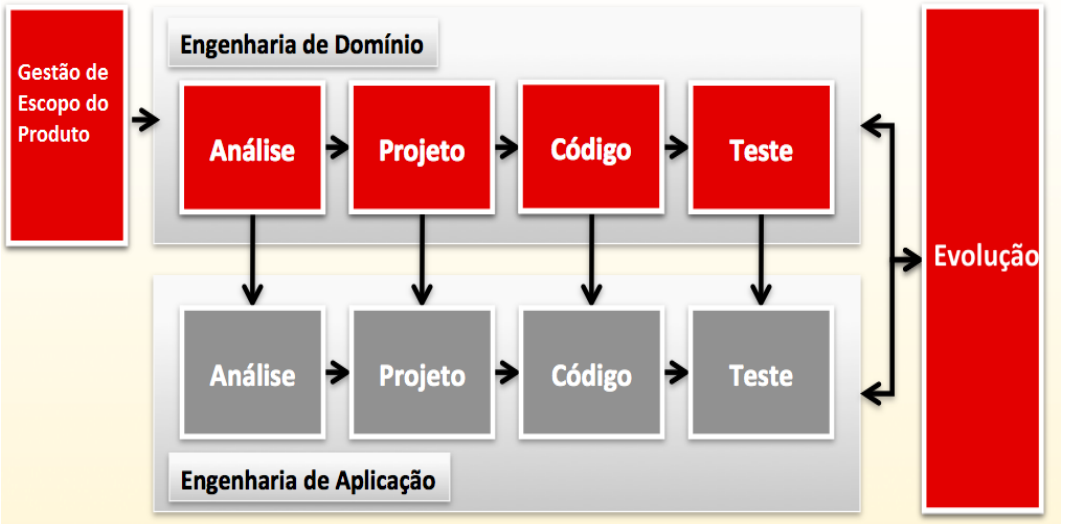


Figura 2.1: O processo de engenharia de domínio [1].

Nesse trabalho vamos detalhar somente os sub processos de Gestão de Escopo do Produto por se tratar do foco da aplicação.

2.3 Gestão de Escopo do Produto

Durante todo o clico de vida de uma LPS, a gestão de escopo do produto tem que lidar com vários aspectos de desenvolvimento dentro do mercado, como as mudanças que o cliente solicitar, o surgimento de novas tecnologias, competidores apresentando novas *features* e a troca na demanda e preços [23].

A gestão de escopo do produto lida com aspectos econômicos das LPS, mais especificamente com as estratégias de mercado, que são definidas e representadas pelo especialista de negócio. Seu objetivo principal é o gerenciamento do portfólio do produto da empresa ou unidade de negócio. Em linhas de produtos de software, a gestão de escopo de produto se encarrega de técnicas para definir o que está dentro do escopo de linha de produto e o que está fora [25].

Podemos observar a partir da figura 2.1 que a gestão de escopo de produto é um processo que ocorre antes da ED e trabalha em atividades como definição de mercado e estratégia do produto, definição do produto, suporte ao produto, introdução ao mercado, observação do mercado e controle do produto [22].

A gestão de escopo do produto, baseado na organização de mercado observada pelos espe-

cialista de negócio, define um portfólio de produto juntamente com um *roadmap* e sua maior contribuição é a diversidade de *features* dos produtos planejados na LPS.

O maior resultado da gestão de escopo do produto com respeito a LPS é o *roadmap* do produto. O *roadmap* do produto apresenta toda uma LPS quanto às previsões de produto, ele define também as mais comuns e variáveis *features* em todas as aplicações da LPS [2].

As *features* especificadas no *roadmap* do produto afetam diretamente a análise de domínio. Veja figura 2.2.

Figura 2.2: Fluxo de informação entre Gestão de Escopo do Produto e Análise de Domínio. Adaptado de [2].

A análise de domínio (AD) é um sub processo que engloba todas as atividades para resumir e documentar os requisitos comuns e variáveis da LPS. A AD utiliza da *roadmap* do produto junto com a lista de artefatos existentes como informações de entrada para gerar como retorno sugestões da adição ou alteração *features* na LPS [25].

A AD analisa os requisitos para identificar o quão eles são comuns entre todas as aplicações e quais são específicos dentro de uma LPS. Existe então a possibilidade das escolhas a partir de requisitos explicitamente documentados e como recebem informações da gestão de escopo de produto, esta etapa pode antecipar possíveis futuras mudanças nos requisitos, tais como leis, mudança em tecnologias, padrões e necessidade de mercado para uma aplicação futura [24].

2.4 Exemplo de uma Análise de Domínio/Engenharia de Escopo

A linha de produto *Arcade Game Maker* (AGM)[1] possui o interesse de criar uma série de jogos do gênero arcade. Cada um desses jogos será de um único jogador, no qual ele controla alguns objetos de forma a pontuar ao embater em outros objetos estáticos. Os jogos estarão disponíveis em diversas plataformas (consoles, PC e dispositivos móveis).

O contexto em que a LPS AGM se enquadra é ilustrado na figura 2.3. As quatros derivações de jogos inclusos no círculo constituem no Domínio da linha de produto. Já as que estão fora não correspondem ao escopo de Domínio da linha, e por sua vez não são consideradas na construção de *assets* para linha de produto.

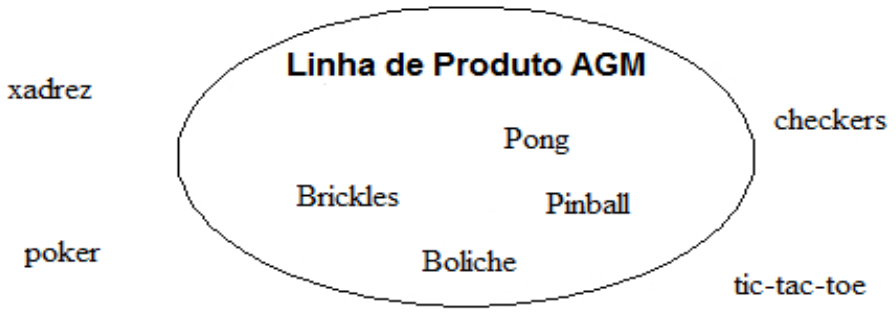


Figura 2.3: Diagrama de Contexto. Adaptado de [3].

As principais diferenças de um produto para outro dentro da LPS da AGM consistem nas variações das regras, tipo e números de objetos envolvidos, comportamentos desses objetos e a física ambiental com a qual o jogo opera.

O escopo das *features* envolve em que cada produto possa fornecer ao usuário (jogador) a habilidade de interagir com o jogo para controlar alguma parte da ação. O produto irá operar de acordo com as regras de cada jogo e resultará em uma representação gráfica do seu estado atual. Os produtos controlam o hardware através de um sistema operacional existente e o tratamento das variações de hardware é de responsabilidade do Sistema Operacional.

As descrições acima deram uma primeira ideia da captura do escopo da linha de produto. A tabela 2.1 refere-se a variações nas *features* de acordo com o mercado aplicado do produto,

[1]Para maiores detalhes e informações referencias, etapas e diagramas do exemplo utilizado nesse capítulo acesse *https://www.sei.cmu.edu/productlines/ppl/scope.html* [3].

por exemplo, a *feature* "Exibição"trabalha com *bitmaps* padrão quando o objetivo de mercado é jogos de *arcade* gratuitos, mas, quando o objetivo de mercado é atingir os jogos comerciais PC o foco dessa *feature* é disponibilizar gráficos vetoriais.

Tabela 2.1: Exemplo de *Features* por Necessidades de Mercado. Adaptado de [3]

Feature	**Jogos de Arcade Grátis**	**Jogos comerciais de PC**	**Jogos Comerciais sem-fio**
Exibição	*bitmaps* padrão	gráficos vetoriais	protocolo de acesso sem fio WAP bitmap
Interação	mouse/botões	dispositivo de indicação	botões
Objetos do Jogo	ícones simples	ícones dinâmicos	ícones dinâmicos
Placar	recordes armazenados localmente	armazenamento local/armazenamento compartilhado com inscrição	Placar salvo na internet/inscrição requerida

Ná figura 2.4 é representado, em alto nível, o modelo de *features* do domínio "Jogos de Arcade". Esse modelo exibe as *features* dos produtos que são necessárias para um produto ser completo no mercado e satisfazer os consumidores. Essas *features* existem em todos os produtos da LPS.

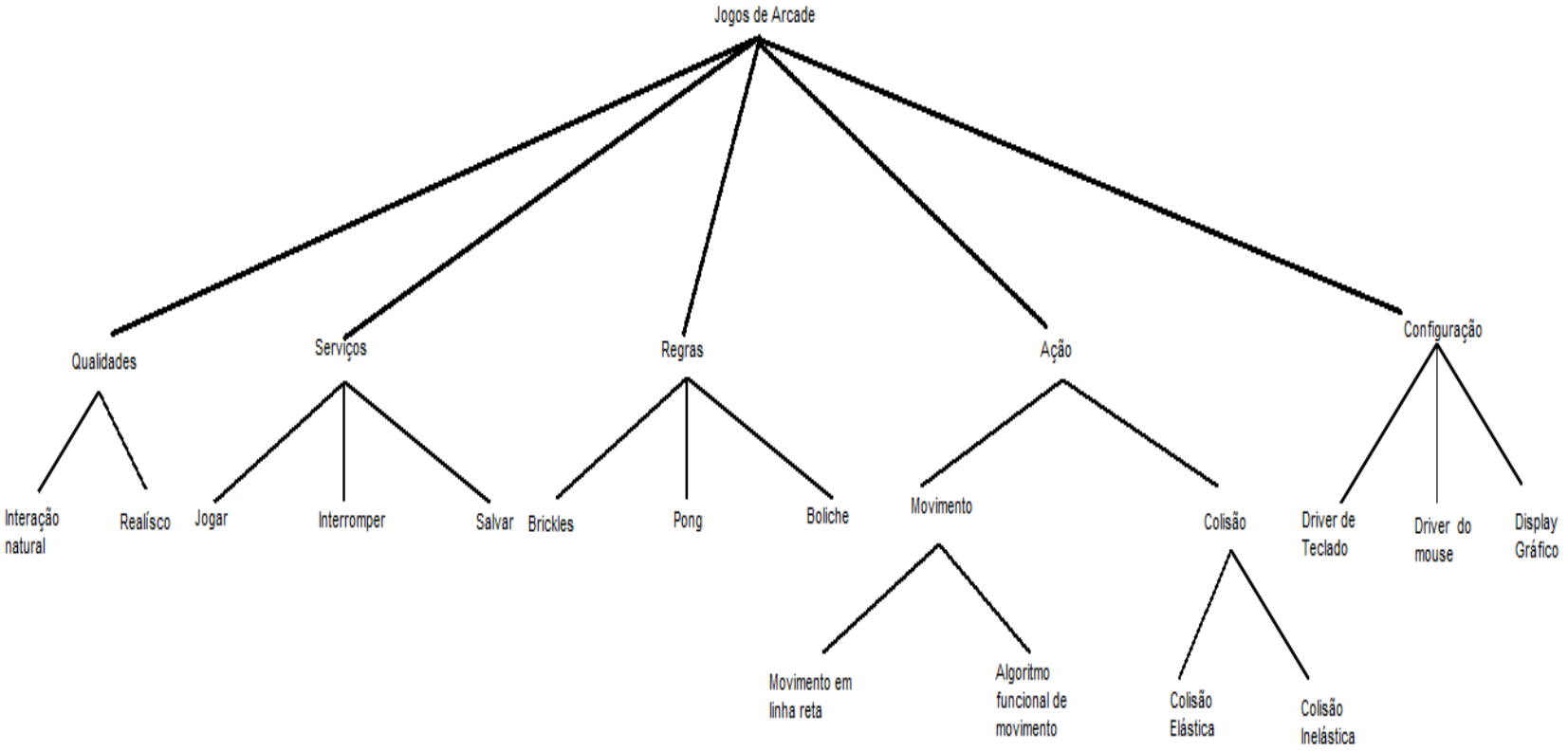


Figura 2.4: Modelo de *features* do Domínio "Jogos de Arcade". Adaptado de [3].

Para um novo produto ser criado usando os *assets* da LPS, o produto deve ser avaliado de acordo com o escopo da linha. Esse processo depende de como formalmente o escopo foi definido. Na LPS da AGM o escopo foi definido nas regras vistas acima.

Se esse novo produto se encaixa no escopo LPS, então ele é avaliado de acordo com a descrição de arquitetura para maiores detalhes técnicos. Entretanto, se o novo produto não se enquadra completamente, mesmo que parcialmente, é efetuado uma nova análise de mercado para descobrir se existe a necessidade de expandir o escopo da LPS para poder incluir o novo produto. Veja o fluxo grama 2.5.

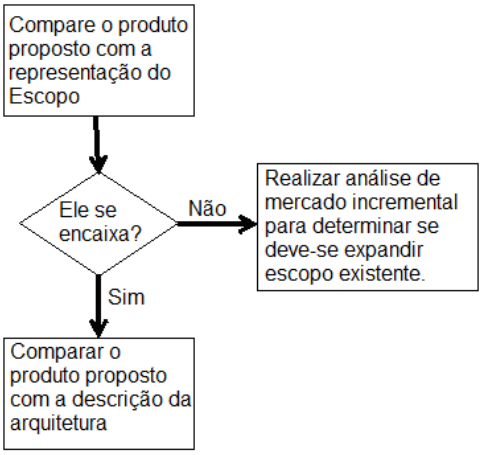


Figura 2.5: Fluxograma para a adição de novos produtos na LPS. Adaptado de [3].

Capítulo 3

Métodologias Ágeis

3.1 Introdução

Nas décadas de 80 e 90, o foco do desenvolvimento era os sistemas complexos e com alto grau de responsabilidade que levava a precisar de maior garantia de qualidade, uso reforçado de vários métodos de análise e projetos apoiados por ferramentas CASE, com uma supervisão e controle rígido do processo de desenvolvimento de software. Gerando um *overhead* de informações necessário, por tratar-se de um sistema dito crítico [5]

Entretanto, ao aplicar as mesmas metodologias em sistemas de médio e pequeno porte, nos quais os requisitos iniciais do sistema podem sofrer grandes mudanças, o tempo necessário para o desenvolvimento é considerado curto aos sistemas críticos. O mesmo *overhead* de informações gerava descontentamento por parte da equipe ao analisar o tempo gasto com a documentação contra o de desenvolvimento, na qual o primeiro era muitas vezes maior que o segundo [26].

Essa insatisfação levou um grupo de desenvolvedores na década de 90 a propor os métodos ágeis. Esses métodos, como objetivo principal, direcionam o foco para o desenvolvimento de sistema e menos para a documentação.

Em geral, métodos ágeis contam com uma abordagem interativa para a especificação, desenvolvimento e entrega de software e foram concebidos principalmente para apoiar o desenvolvimento de aplicações nos quais os requisitos de sistema são ditos mutáveis, ou seja, os requisitos funcionais e não funcionais podem sofrer mudanças e adaptações com o decorrer do desenvolvimento do sistema [5].

Existem vários métodos ágeis atualmente no mercado, entre os quais podemos citar o *ex-*

treme programming, *Crystal*, *Adaptative Software Development, Feature Driven Development* e Scrum. Apesar de que estes métodos ágeis sejam todos baseados na intenção do desenvolvimento ágil e entregas incrementais, eles sugerem processos diferentes, entretanto, compartilham de um grupo de princípios. Esses princípios podem ser observados na tabela 3.1.

Tabela 3.1: Princípios dos métodos ágeis. Adaptado de Sommerville[5]

Principio	Descrição
Envolvimento do Cliente.	Clientes devem ser profundamente envolvidos no processo de desenvolvimento. Seu papel é fortalecer e priorizar novos requisitos do sistema e avaliar as iterações do sistema.
Entrega incremental.	O software é desenvolvido em incrementos e o cliente especifica os requisitos a serem incluídos em cada incremento.
Pessoas, não processos.	As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas. Os membros da equipe devem desenvolver suas próprias maneiras de trabalhar em processos prescritos.
Aceite as mudanças.	Tenha em mente que os requisitos do sistema vão mudar, por isso projete o sistema para acomodar essas mudanças.
Mantenha a simplicidade.	Concentre-se na simplicidade do software que está sendo desenvolvido e do processo de desenvolvimento. Sempre que possível, trabalhe ativamente para eliminar a complexidade do sistema.

O Scrum oferece práticas que sistematizam a gestão das atividades da engenharia de escopo e LPS. Além disso, Srum é o método mais popular[27], e assim ele será abordado adiante nesse capítulo.

3.2 Scrum

O Scrum é uma metodologia de desenvolvimento baseado em iterações incrementais focada no gerenciamento de projetos de desenvolvimento ágil de sistemas. Desenvolvido por Ken Schwaber e Jeff Sutherland [28], o Scrum não é um processo que descreve as ações em cada situação dentro do desenvolvimento de software, entretanto, é usada para projetos com requisitos complexos nos quais é praticamente impossível prever seu desenvolvimento.

O objetivo do Scrum não é prover uma técnica ou um processo para construir um produto, entretanto, é um framework onde é possível empregar vários processos ou técnicas. Dentro do Scrum é possível observar a eficácia das práticas de gerenciamento e desenvolvimento de produtos, de modo que os indivíduos envolvidos no projeto possam melhorá-las [4].

Segundo Schuwaber e Sutherland [28], o Scrum é baseado nas teorias empíricas de controle de processo, ou empirismo. O empirismo afirma através da experiência e tomada de decisões baseadas no que é conhecido é possível formar o conhecimento. Com as abordagens iterativas e incrementais, o Scrum visa ao longo do processo aperfeiçoar a previsibilidade e o controle de riscos.

Existem três pilares que amparam a prática de controle empírico: transparência, inspeção e adaptação. No primeiro, todos os aspectos significativos do processo devem ser de conhecimento de todos os responsáveis. Já o segundo pilar, sugere que os usuários Scrum devem, frequentemente, vistoriar as ferramentas Scrum e o andamento em direção ao escopo, para que seja possível detectar variações indesejáveis. Por último, o terceiro pilar sugere ajustes aos aspectos que foram identificados pelo inspetor para que seja possível contornar o problema identificado [29].

O Scrum é um framework constituído por equipes associadas a papéis, eventos, artefatos e regras. Cada componente dentro do framework incide a um propósito específico e é essencial para o uso e o sucesso do projeto. Os componentes do Scrum podem ser classificados Time Scrum e a *Sprint* [28].

3.2.1 Time Scrum

O Time Scrum é a equipe de pessoas que estão trabalhando no desenvolvimento do sistema. As principais características do Time Scrum é a sua auto-organização e sua multifuncionalidade. As equipes auto organizáveis podem escolher a melhor forma de trabalhar e concluir determinada tarefa. O Time Scrum é constituído pelo *Product Owner*, a equipe de desenvolvimento e o Scrum *Master* [1].

O *Product Owner* (dono do produto) é o responsável por aumentar ao máximo o valor do produto e do trabalho em equipe de desenvolvimento. Ele é o responsável por gerenciar o backlog do produto, expressar claramente os itens do *backlog*, organizar os itens do *backlog* de modo a alcançar melhor os objetivos, garantir que o *backlog* seja concreto e claro para todos e mostrar o que o Time Scrum vai trabalhar a seguir e garantir que a equipe de desenvolvimento compreenda os itens do *backlog* do produto [1].

A equipe de desenvolvimento é formada por aqueles que trabalham na construção do sistema. Eles têm por objetivo principal entregar uma versão usável que potencialmente incrementa o produto pronto a cada final de uma Sprint (para saber mais sobre Sprint veja a 3.2.2).

As equipes de desenvolvimento são estruturadas e liberadas pela organização para montar e gerenciar seu próprio método de trabalho. Eles são auto organizáveis, multifuncionais, qualquer integrante da equipe de desenvolvimento possui somente o titulo de desenvolvedor, apesar da individualidade de especialização de cada membro a responsabilidade pertente a equipe como um todo e não possuem subequipes dedicadas [28].

E por último temos o papel de Scrum *Master*: garantir que o Scrum seja compreendido e executado de maneira correta desde sua teoria até as práticas e regras. Ele tem o papel de integrar os acontecimentos a todos os membros do Time Scrum e gerenciar as iterações para que seja maximizada sua execução [29].

O Scrum *Master* interage com o *Product Owner* para auxiliar a construção de *backlogs* de forma sucinta, entender em longo prazo a idealização do produto no ambiente empírico e simplificar os eventos Scrum conforme determinados ou necessários [28].

A interação com a equipe de desenvolvimento é exigida para treinar o auto gerenciamento e interdisciplinaridade, ensinar e liderar na criação do produto, remover quaisquer empecilhos no progresso de desenvolvimento e simplificar os eventos Scrum conforme determinados ou

necessários [28].

3.2.2 *Sprint*

O coração do Scrum é a *Sprint*, uma reserva de tempo de um mês ou menos, na qual o produto, em uma versão incremental potencialmente utilizável, é criado. O inicio de *Sprint* ocorre logo após o termino de outra [1]. Veja figura 3.1.

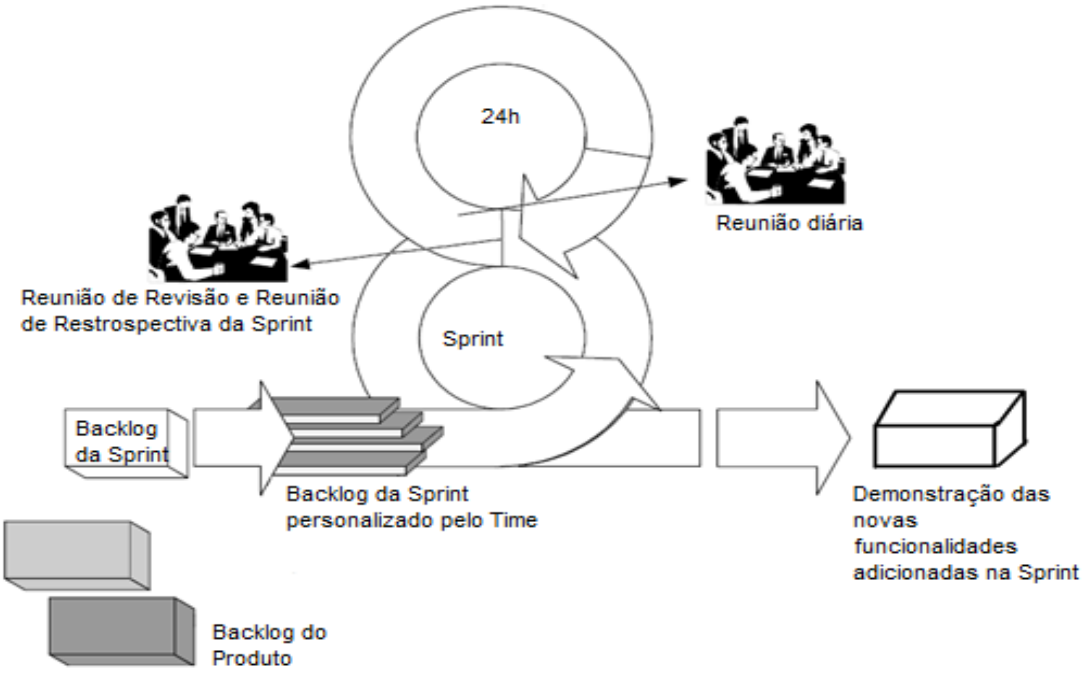


Figura 3.1: Scrum. Adaptado de [4]

Durante a execução de uma *Sprint* não são efetuadas mudanças que podem resultar de forma negativa no objetivo previsto da Sprint. Os arranjos de equipes permanecem os mesmos. O escopo pode ser reavaliado e renegociado entre o *Product Onwer* e a Equipe de desenvolvimento ao decorrer do aprendizado do produto [28].

A divisão do desenvolvimento do produto é dividida em várias Sprints, nas quais é fornecido um plano flexível de construção do produto que pode ser visto como um projeto, não maior que um mês, e que tenha como resultado ir incrementado para o produto final [28].

O Scrum prescreve cinco eventos para inspeção e adaptação de uma *Sprint*: reunião de planejamento da *Sprint*, reunião diária (*daily Scrum*), reunião de revisão da *Sprint*, retrospectiva da *Sprint* e entrega do produto [4].

Na reunião e planejamento da *Sprint* o *Product Owner* junto ao Scrum *Master* decidem qual é a porção do *backlog* do produto será adicionada no *backlog* da *Sprint*. Esse *backlog* terá como finalidade servir de objetivos para a *Sprint* que virá em seguida [4]. A reunião diária ocorre a cada 24 horas dentro de uma *Sprint*. Dentro dessa reunião são informados o que cada integrante das equipes está fazendo e quais dificuldades, se existirem, está passando. É nesse momento em que podem ser adicionadas novas tarefas no *backlog* da *Sprint* e identificar problemas que impeçam a conclusão dos objetivos traçados no planejamento [1].

A revisão *Sprint* ocorre no inicio do último dia. Nesse evento é discutido, com todos os envolvidos na *Sprint*, o que cada um construiu e como foi o andamento da *Sprint* [28].

No final do último dia tem-se a retrospectiva da *Sprint*. É efetuada uma nova reunião como o Time Scrum onde são capturados o *feedbacks* positivos ou negativos gerados na última *Sprint*. Esse *feedback* é necessário para evitar que determinados problemas voltem a ocorrer em outras *Sprints* ou para avaliar se determinadas adaptações nas rotinas serviram para melhorar o progresso de construção do produto [1].

Logo após acabar a *Sprint* é efetuada a sua entrega. É disponibilizado aquilo que foi concluído, construído e planejado no início da *Sprint*. Esse conjunto de eventos dentro do Scrum fornece uma rotina e minimizam a necessidade de reuniões imprevistas. Esses eventos são baseados em reserva de tempo. A não inclusão de um dos eventos resultará na diminuição da transparência e da perda de chance para inspecionar e adaptar o processo [28].

Capítulo 4

Proposta

4.1 Modelagem

Para o desenvolvimento da ferramenta, foi necessário coletar requisitos e expressá-los na forma de documentos. Nessa sessão iremos abordar a documentação inicial do sistema através da modelagem criada na etapa análise e pré-construção da ferramenta LiProMA. Essa etapa foi importante para rastrear as *features* da ferramenta bem como assegurar o desenvolvimento correto da ferramenta.

4.1.1 Modelagem Organizacional I*

A modelagem organizacional I* adotada nesse trabalho facilita o entendimento dos atores (especialistas) no processo e a relação de dependência com as tarefas dentro da ferramenta.

No diagrama de dependências estratégicas (SD), veja figura 4.1, podemos observar os relacionamentos entre os especialistas e a ferramenta proposta. Existem cinco especialistas no processo e cada um interage com a ferramenta buscando a gestão do escopo da LPS, por exemplo, podemos citar o especialista de escopo que é responsável por aplicar critérios de potencial de reuso, gerenciar o mapa de produto e gerenciar o *backlog* de escopo, entretanto ele espera da ferramenta uma boa experiência em usabilidade e que a ferramenta forneça a centralização de informações sobre a gestão do escopo da LPS.

Figura 4.1: Modelo Organizacional SD

O diagrama a seguir exibe o detalhamento das estratégias internas (SR), veja figuras 4.2 e 4.3, dos especialistas em termos dos elementos do processo, das alternativas e das decisões por trás do processo na ferramenta. Por exemplo, o objetivo de gerenciar *backlog* de escopo, descrito no digrama SD (figura 4.1), é dividido em quatro tarefas: identificar domínios da LPS, gerenciar os responsáveis do *backlog* de escopo, gerenciar *backlog* das *Sprints* do *backlog* de escopo, e identificar *features* principais para trabalhar no *backlog* de escopo.

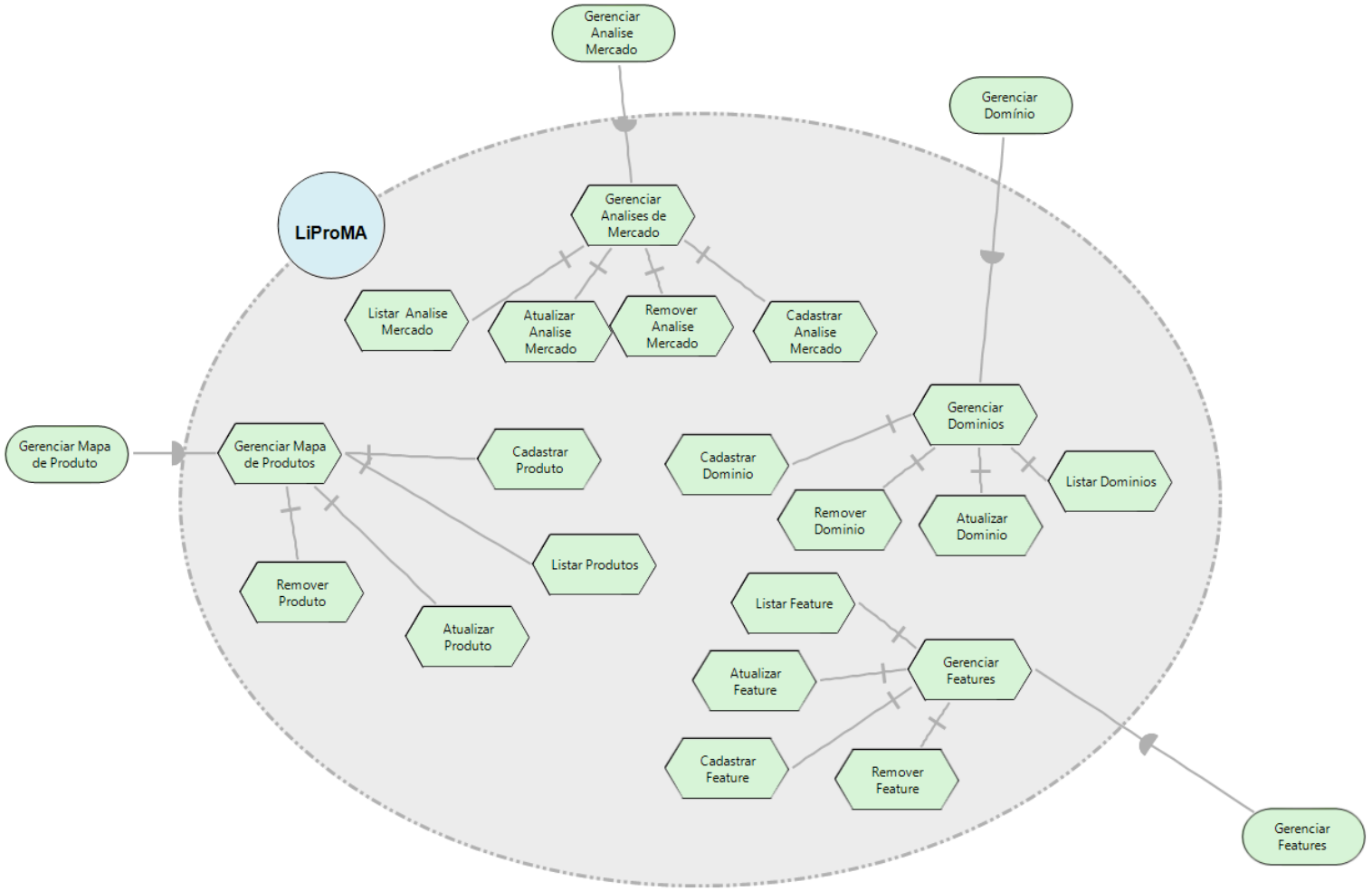


Figura 4.2: Modelo Organizacional SR - Parte 1

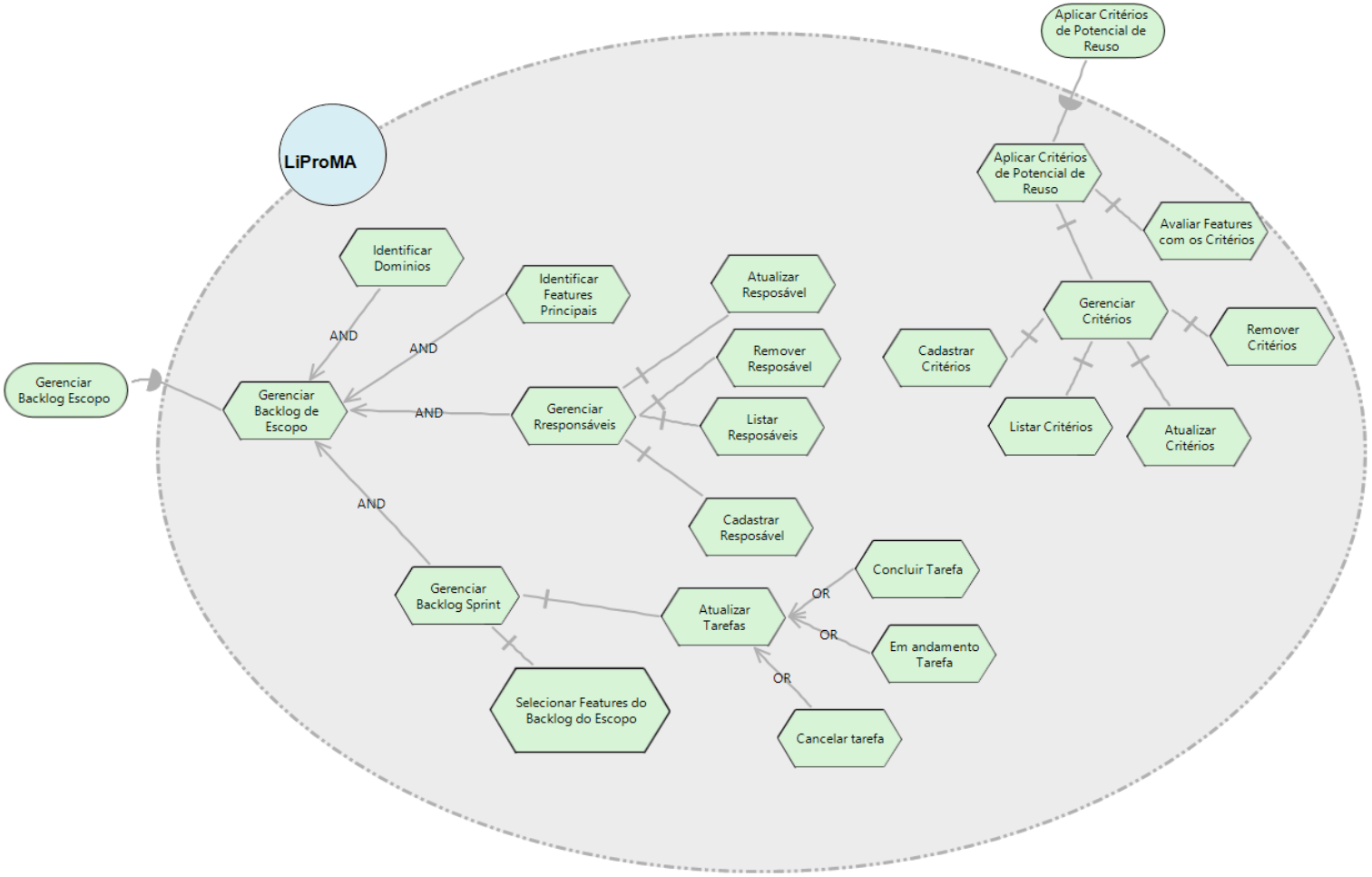


Figura 4.3: Modelo Organizacional SR - Parte 2

4.1.2 Diagramas de Casos de Uso

O diagrama de casos de uso, veja imagem 4.4, foi construído visando descoberta e registro dos requisitos do sistema. Com esse diagrama foi possível identificar cenários que pareciam confusos na primeira coleta de requisitos. Esse diagrama tem propósito similar aos digramas apresentados na sessão 4.1.1. Entretanto, foi construído para elucidar os requisitos das funcionalidades principais que a ferramenta LiProMA deve conter.

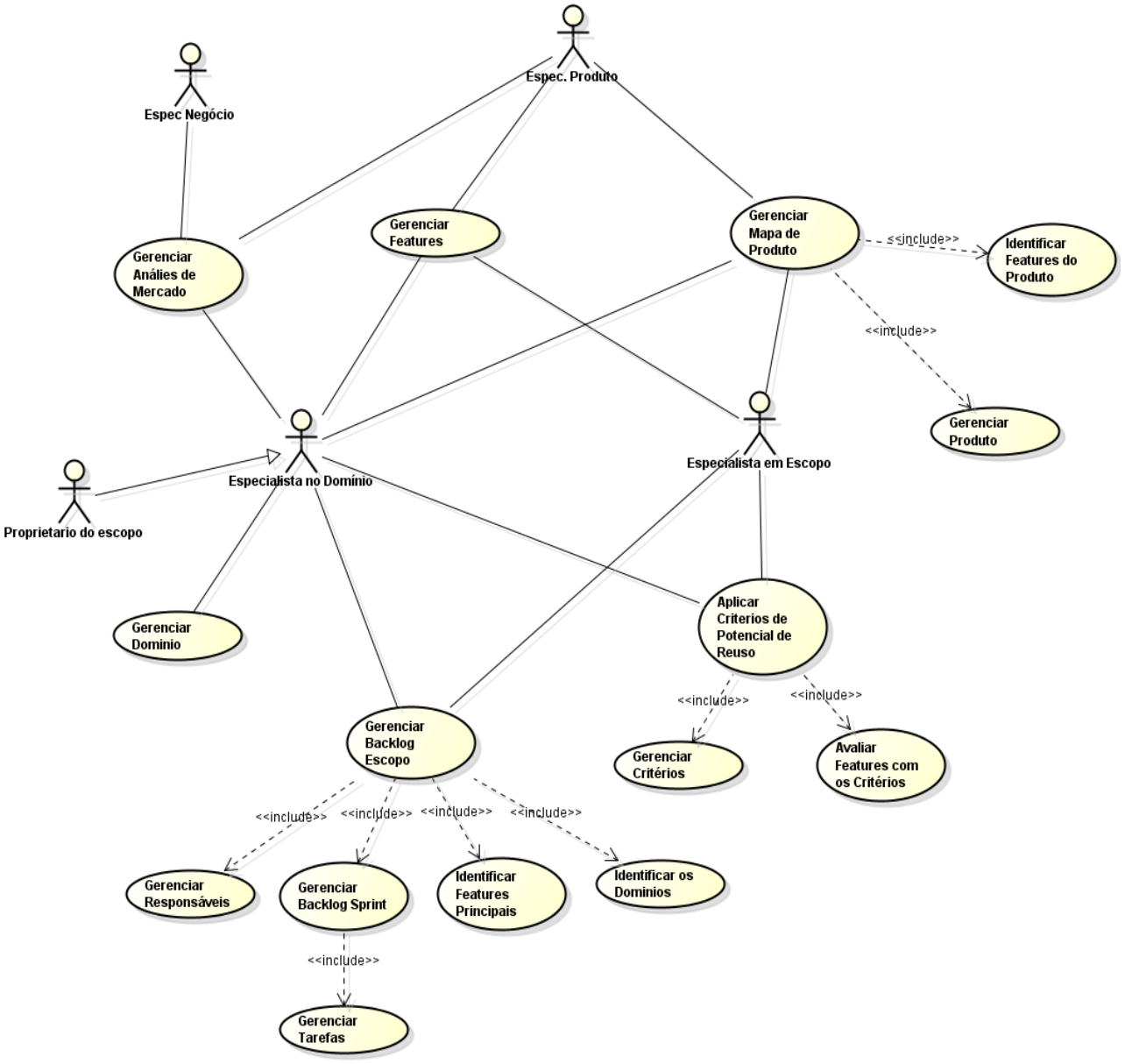


Figura 4.4: Diagrama de Casos de Uso

4.1.3 Diagramas de Classe

Diagrama de classes da fase de análise (figura 4.5) é o começo para definição de um futuro meta-modelo que representará a rastreabilidade entre os conceitos adotados na ferramenta Li-PriMA. Logo abaixo no digrama 4.5 uma descrição breve da construção de cada entidade nesse modelo.

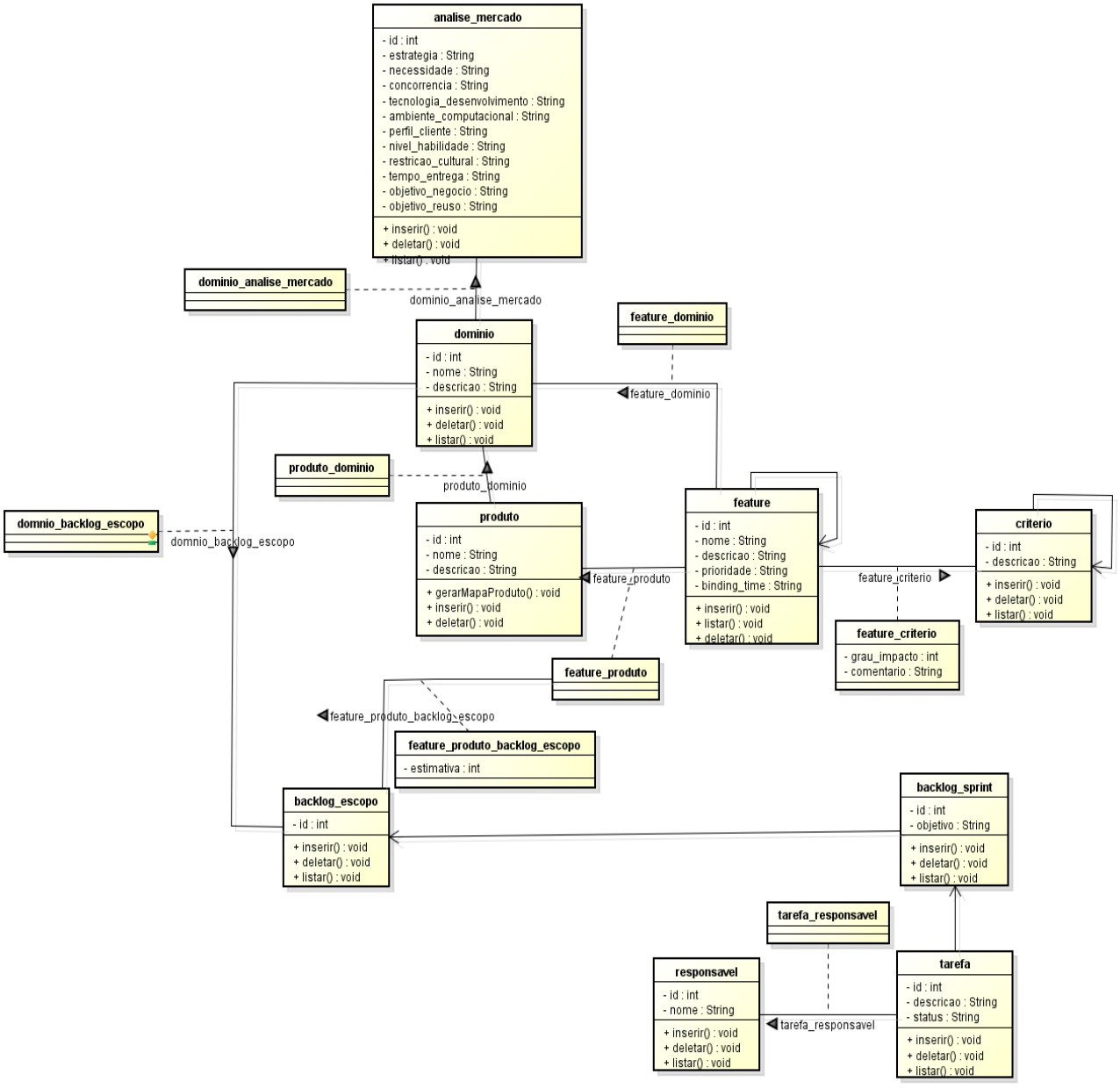


Figura 4.5: Diagrama de classes da fase de analise

4.1.3.1 Classe analise_mercado

A entidade *analise_mercado* representa os dados efetuados da análise de mercado sobre os domínios da LPS. Seus atributos são uma representação do formulário 4.6.

Análise de mercado										
Estratégia marketing (método de entrega de aplicações)	Necessidades de mercado	Concorrência	Tecnologias desenvolvimento	Características dos usuários			Restrições culturais/legais	Tempo de entrega	Objetivos de negócio	Objetivo do reuso
				Ambiente computacional	Perfil dos clientes	Níveis de habilidades				
Entrega do mesmo produto para todos os clientes (não-customização)	Versão web com acesso via tablet ou smartphone			Ambiente desktop, web, tablet e smartphone.	Funcionários das unidades operacionais subordinadas à Petrobras	Equipe de suporte da organização instala o sistema, além de treinar os usuários	Responder às normas da ANP	sem média de tempo	Reduzir custos de desenvolvimento	Maximizar customização dos serviços/produtos
Para clientes já existentes, "pequenas" entregas por conjuntos de features (web), com integração incremental ao sistema desktop existente				Infraestrutura de TI adequada		Usuários treinados		Rápida entrega para manutenções corretivas	Redução de custos de manutenção (reparo e suporte)	Aumentar a eficiência/produtividade
									Entregar serviços/produtos aos clientes utilizando tecnologia web	Redução de custos
									Entregar serviços/produtos customizados à curto prazo (semanas/meses/bimestres)	Melhorar qualidade
									Reusar serviços/produtos previamente desenvolvidos/estruturados	Rápida entrada aos clientes
										Difunfir conhecimento entre equipes

Figura 4.6: Imagem exemplo da análise de mercado.

4.1.3.2 Classe dominio

A entidade *dominio* representa os Domínios cadastrados no sistema. Não existe um único formulário em que foi baseado para criação dos seus atributos, entretanto, faz-se necessário seu mapeamento como uma entidade, pois, é um termo presente em grande parte dos documentos do processo.

4.1.3.3 Classe feature

A classe *feature* corresponde as *features* dos produtos ou dos domínios existentes no sistema. Seus atributos foram baseados na figura 4.7.

Lista de Features (domínio, features principais, sub-features)				
Domínio	Features principais	Sub-features		Descrição
Exploração Petróleo/Gás	Dados Geofísicos			Responsável por adquirir, tratar e avaliar dados geofísicos
		Obtenção Dados Geofísicos		Gerencia o processo de obtenção dos dados geofísicos
		Processamento Dados Geofísicos		Gerencia as técnicas e atividades envolvidas no processamento dos dados geofísicos obtidos
	Oportunidades Exploratórias			...
		Avaliar Carteira Prospectos Exploratórios		...
		Seleção Prospectos para Avaliação		...
	Poços Exploratórios			...
		Perfuração Poços		...
		Testes de Formação		...
Desenvolvimento Produção	Plano Desenvolvimento Concessão			...
	Projeto/Planejamento			...
		Projeto/Planejamento Poços		...
		Projeto/Planejamento Sistemas		...
			Sistemas Elevação	...
			Sistemas Coleta	...
			Sistemas Injeção	...
	Implantação			...
		Construção Poços		...
		Construção Sistemas Elevação		...
		Implantação Instalações Produção		...
	Comissionamento			...

Figura 4.7: Imagem exemplo da lista de domínio das *features*

4.1.3.4 Classe produto

A entidade *produto* corresponde ao cadastro de produtos da LPS. Pode ser utilizado para cadastro tanto dos produtos existentes como dos produtos pretendidos. Seus atributos foram baseados na figura 4.8.

Template para lista de produtos

Produto	Descrição
Hospital	sistema de gerenciamento de hospitais
Laboratório	sistema de gerenciamento de laboratórios
Clínica	sistema de gerenciamento de clínicas
Consultório médico	sistema de gerenciamento de consultórios médicos

Figura 4.8: Imagem exemplo da Lista de Produtos

4.1.3.5 Classe backlog_escopo

A entidade *backlog_escopo* é responsável por gerenciar e indexar os processos de gestão do escopo através do método ágil Scrum na Ferramenta LiProMA.

4.1.3.6 Classes backlog_Sprint, tarefa e responsavel

A Entidade *backlog_Sprint* foi criada para organizar as *Sprints* do processo de gestão do escopo. Para a construção de cada *Sprint* é necessário cadastrar as tarefas e designá-las aos seus respectivos responsáveis, sendo assim, foi modelado as entidades de *tarefa* e *responsavel*. Os atributos e comportamento das classes backlog_Sprint, tarefa e responsável foram baseados na figura 4.9.

Especialista domínio ou dono do produto:	Kwanwoo Lee, Krzysztof Czarnecki				
Integrantes da equipe	Bob, Timão, WillyKit				
Features principais selecionadas		**Tarefas**	**Proprietário**	**Status**	
Atendimento		identificar sub-features dos experts do domínio	Bob	feito	
		especificar sub-features	Bob, Timão	feito	
		Agrupar, decompor, especializar features	Bob, Timão	feito	
		Organizar diagrama hierárquico (modelo de features)	Bob, Timão	feito	
		Organizar as sub-features no mapa de produto	Bob, Timão	feito	
		Minerar artefatos legados	Timão	feito	
		Refinar features	WillyKit	feito	
		Inspecionar modelos	WillyKit	feito	
		Validar features	Lee, Czarnecki	feito	

Figura 4.9: Imagem exemplo do backlog de uma *Sprint*

O diagrama de classes da fase de *design*, figura 4.10, foi elaborado para elucidar o comportamento a arquitetura. É possível observar o fluxo de cadastro de Produto onde é nítido a arquitetura MVC do sistema, tanto na camada lógica quanto na camada de visão. Esse fato ocorre porque a implementação da camada de visão (web) é baseada no desenvolvimento padrão

de aplicações web do framework ExtJs [30].

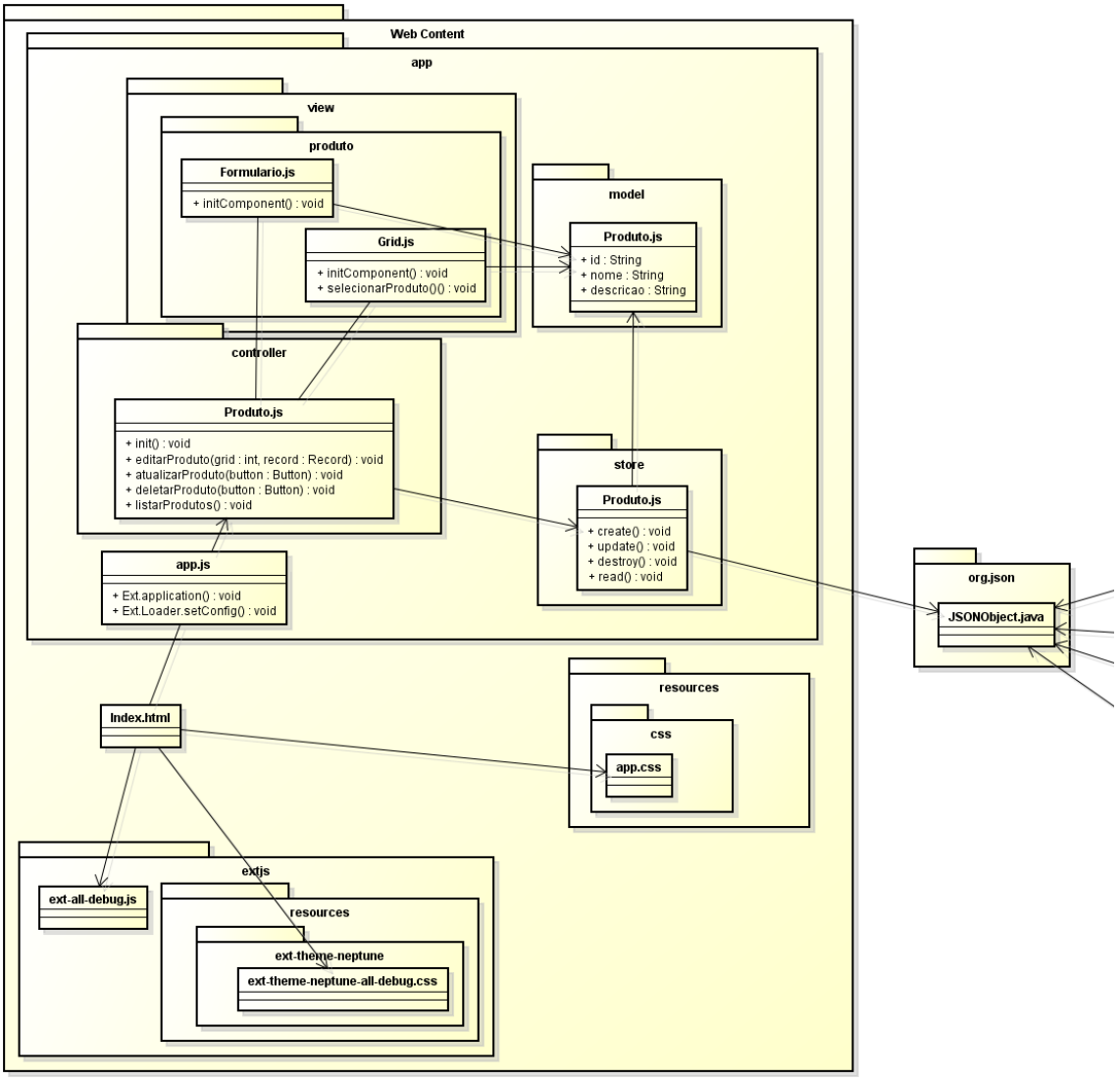


Figura 4.10: Diagrama de classes pertencente a fase de *design* - parte 1

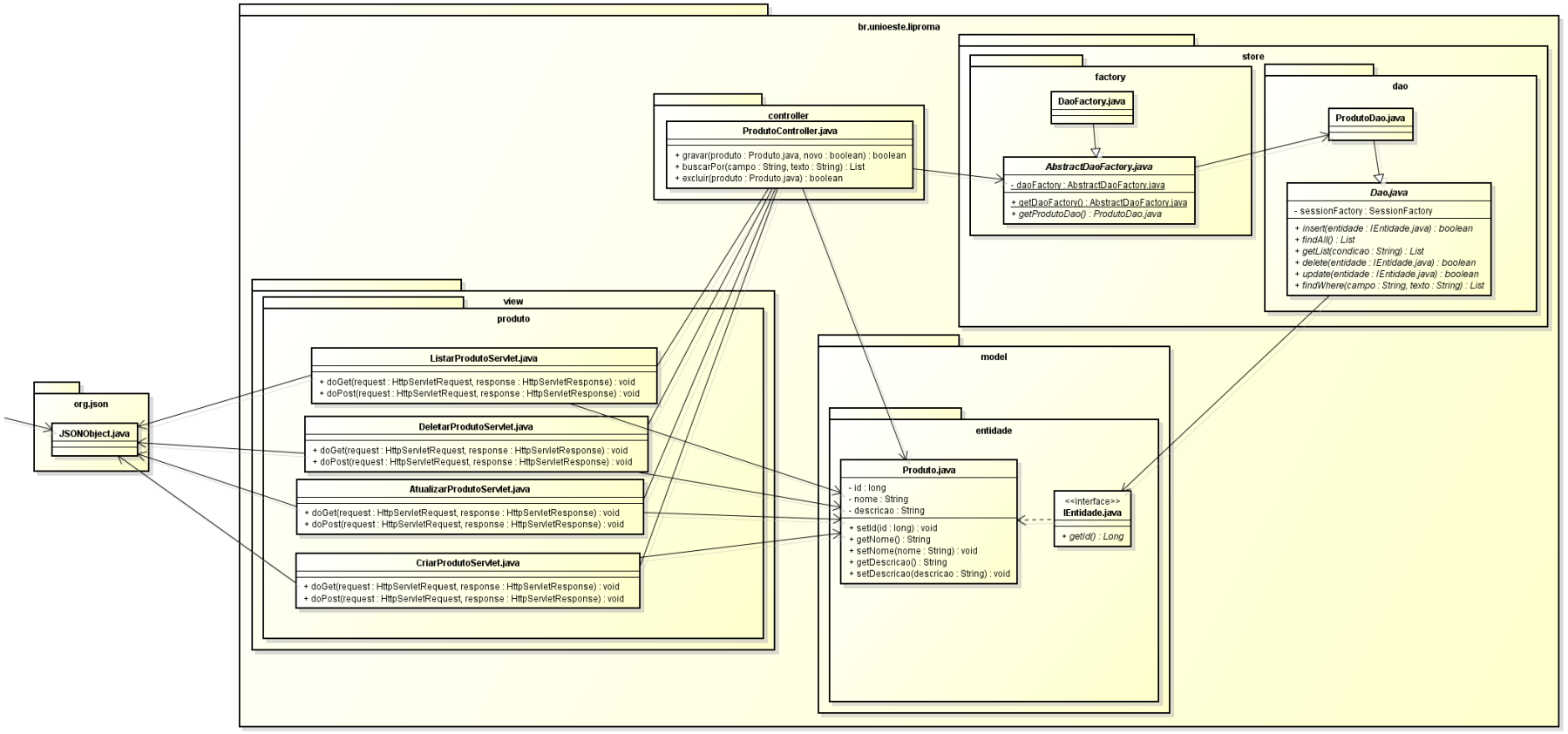


Figura 4.11: Diagrama de classes pertencente a fase de *design* - parte 2

4.1.4 Diagramas de Sequência

Esse diagrama exibe o comportamento da troca de mensagens da ferramenta LiProMA. O diagrama de sequência foi dividido em duas partes: diagrama de sequência do cliente, veja diagrama 4.12, que exibe o comportamento do processo de troca de mensagens entre os *scripts javascript* executado no navegador do cliente e o digrama de sequência do servidor, veja diagrama 4.13, que explana o fluxo do processo de trocas de mensagens entre as classes Java no servidor de aplicação Web. Os dois diagramas são unidos pelo objeto JSONObject cuja responsabilidade na comunicação dos dados entre o cliente e servidor.

No diagrama de sequência cliente, veja diagrama 4.12, foi utilizado um exemplo de requisição da listagem de produto por meio do especialista do produto, a requisição ativa o *controller* que envia uma requisição de leitura para a camada de *store*. A camada *store* converte a requisição em um objeto Json e envia para o servidor web e espera como retorno uma lista de *models* de produto e, por sua vez, envia a camada de *controller*. Com a lista de *models*, o *controller* instancia o *script* Grid.js que exibe para o especialista do produto.

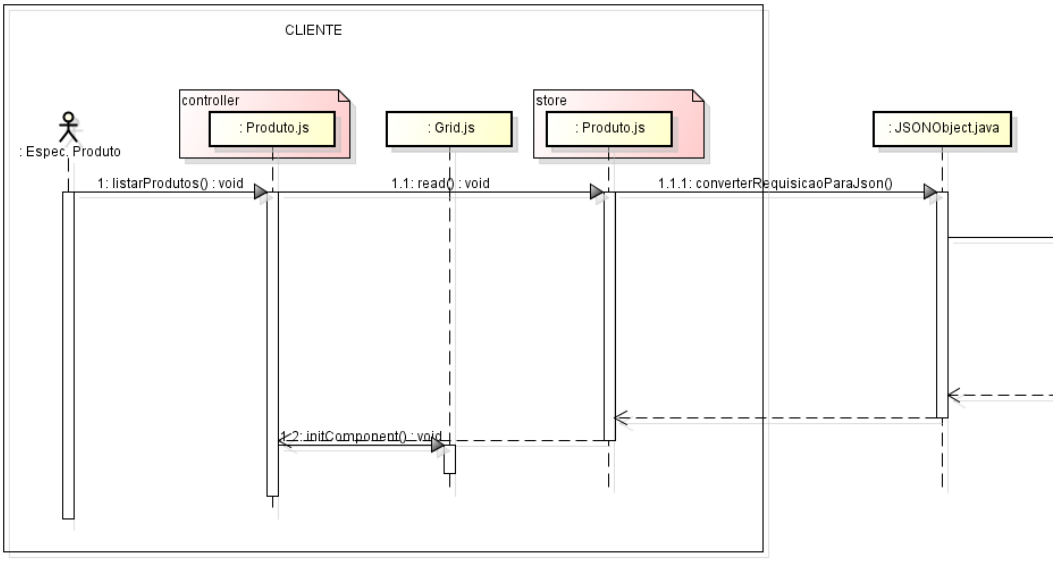


Figura 4.12: Diagrama de sequência cliente

O diagrama de sequência servidor, veja diagrama 4.13, começa com a requisição efetuada pelo cliente e enviada como objeto Json ao servidor. O *servlet ListarProdutoServlet.java* recebe essa requisição, traduz o objeto Json os parâmetros campo e texto e os envia ao objeto

ProdutoController.java, que pertence a camada *controller*.

O objeto *ProdutoController.java* valida essa requisição e solicita ao *AbstractDaoFactory.java* uma instância do *DaoFactory.java*, para que este último produza o *Data acess object* (Dao) da entidade do banco produto chamado *ProdutoDao.java.* Assim, o controlador acessa o ProdutoDao.java, com os parâmetros campo e texto, e requisita a busca. o *ProdutoDao.java* acessa o banco, por meio do drive *Java Data Base Conector* (JDBC), executando a consulta construída a partir dos parâmetros campo e texto. O resultado dessa consulta é enviado ao *ProdutoController.java* que delega a tarefa ao *Servlet ListarProdutoServlet.java* converter a lista de objetos consultados em objetos Json e, por fim, envia ao cliente.

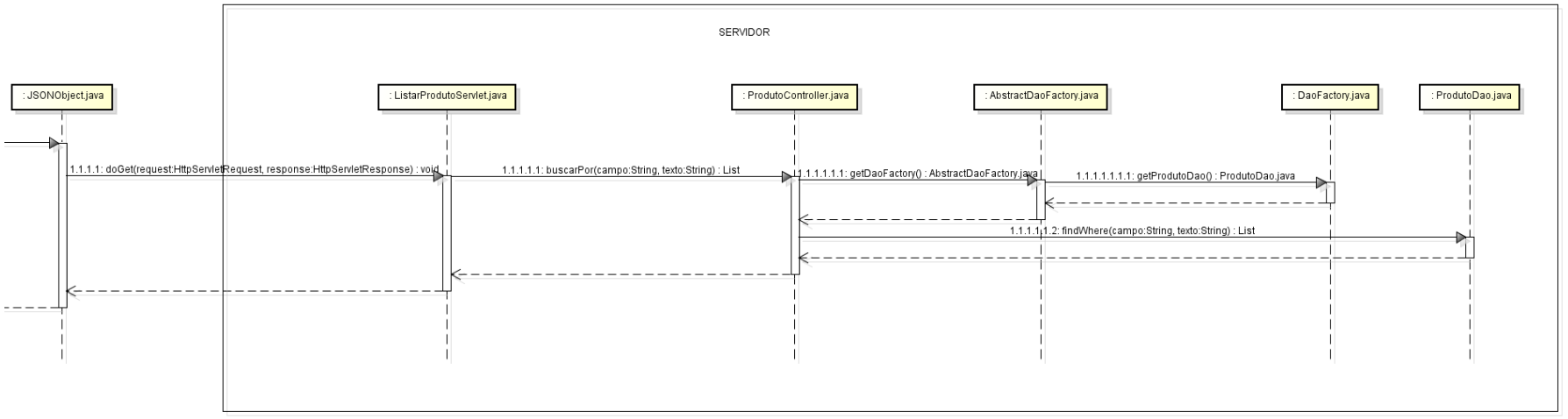


Figura 4.13: Diagrama de sequência servidor

4.2 Ferramentas utilizados no desenvolvimento da LiProMA

Nessa seção será abordada as ferramentas e as principais tecnologias envolvidas no desenvolvimento do sistema.

4.2.1 Linguagens

No sistema apresentado neste trabalho foi utilizado o padrão JSON para a comunicação entre o cliente (linguagem Javascript) e o servidor (linguagem Java). JSON é uma formatação leve de troca de dados baseado em um subconjunto da linguagem de programação JavaScript. Esse formato foi padronizado por Douglas Crockford [31], que por conta por sua simplicidade e eficiência foi difundindo como solução para o antigo padrão Xml [32] .

4.2.2 *Framework* Sencha ExtJs 4.2

ExtJs é uma biblioteca em Javascript pertencente a Sencha sobre a licença GPL para a construção de aplicativos web com interfaces ricas. Atualmente sobre a versão 4.2., esse framework fornece uma enorme biblioteca de componentes em uma estruturação MVC (Model View Control). Além disso, possui boas prática de uma boa usabilidade, seus componentes oferecem um leque de iterações e facilitações no desenvolvimento de sistemas complexos [30].

4.2.3 Banco de dados

O banco de dados foi desenvolvido seguindo o diagrama de classe de análise (veja sessão 4.1.3). Os atributos e as relações contida nas entidades foram construídos segundo os diagramas de classe. O diagrama de entidade-relacionamento resultante é exibido na figura 4.14.

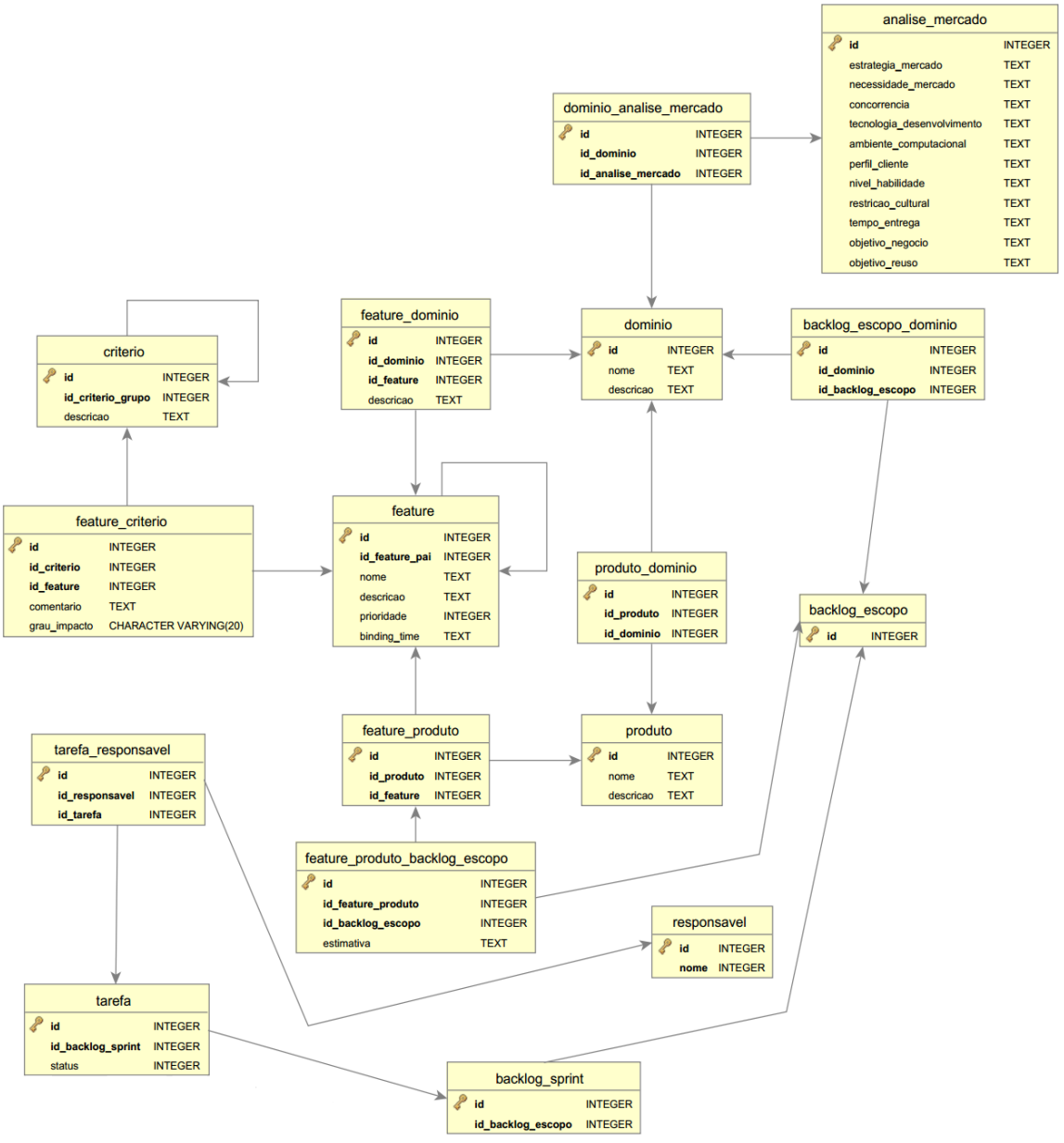


Figura 4.14: Modelo Relacional do Banco de dados

4.2.3.1 *Postgres*

Todo o desenvolvimento da ferramenta foi baseado no Sistema de Gerenciamento de Banco de Dados *Postgres* [33], por tratar-se de uma ferramenta livre e robusta. Entretanto, o sistema é independente de um SGBD específico.

4.2.3.2 Hibernate

O Hibernate é uma ferramenta para o mapeamento objeto-relacional escrito na linguagem Java. No sistema LiProMA ele tem por objetivo facilitar o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objeto de uma aplicação (DAO), mediante o uso de arquivos (XML)[34].

4.2.4 Servidor Tomcat

Desenvolvido pela *Apache Software Foundation*, o Tomcat é um servidor Web de aplicações Java de código aberto, entretanto, não chega a ser um EJB. No sistema LiProMA, o Tomcat serve de *conteiner* para os arquivos *servlets* da aplicação. É nele que é efetuada a camada lógica e estratégica.

4.2.5 Sistema Operacional

Como visto nos tópicos anteriores, o LiProMA foi desenvolvido pensando na independência de plataforma, ou seja, o sistema desenvolvido é independente S.O. do cliente.

4.3 Protótipo

Como já discutido nos capítulos anteriores, a ferramenta LiProMA o processo de gestão ágil de escopo da linhas de produtos. Na figura 4.15 podemos analisar mais claramente o fluxo do método proposto da Silva [6]. O método consiste em duas etapas: Estudo e análise do escopo da LPS e a gestão do escopo da LPS e suas *features* através de iterações do Scrum.

Primeiramente, na primeira etapa, o especialista de negócio e o especialista de domínio efetuam a pré-análise da LPS, isso pode envolver um estudo sobre o mercado do domínio a ser explorado ou documentado resultando no documento de análise de mercado. Com os dados coletados por esses dois especialistas, é tarefa do especialista de escopo elucidar o mapa de aplicações da LPS, ou seja, desenhar em uma matriz os produtos existentes e pretendidos na LPS junto com a identificação de suas *features* e *subfeatures*. Com o mapa de produto e a pré-análise feita pelos especialistas anteriores, é tarefa do especialista de domínio separar e enfatizar quais as *features* dentro do mapa de produto são as principais (qual *features* devem ser prioridades dentro da análise da LPS).

A análise de escopo gerada é seguida pela etapa de gestão do escopo da LPS. Essa gestão é efetuada levando em conta o método Scrum e suas características de gestão, na qual, o proprietário do escopo, que faz o papel de *product owner*, tem a função principal de gerenciar o *backlog* de escopo, no qual é definido quais das *features* principais identificadas na etapa anterior vai fazer parte da gestão. Com o *backlog* de escopo definido é possível gerenciar o escopo por meio de iterações incrementais, conhecidas dentro do Scrum como *Sprints*.

Cada começo de *Sprint* é realizada uma reunião de planejamento com o proprietário do escopo e o *Scrum master*, tendo por objetivo criar o *backlog* da *Sprint*. Dentro de cada *backlog* de *Sprint* é inserido um conjunto de *features*, que pertencem ao *backlog* de escopo, e são vinculadas tarefas para cada uma dessas *features* tarefas para ser executadas, como por exemplo refinar *features*, inspecionar modelos, especificar *subfeatures*, entre outros que sejam planejados pelos especialistas. Para cada resolver as tarefas é direcionado um ou mais responsável(is).

As etapas de reunião diária, reunião de revisão e reunião de retrospectivas acontecem conforme a metodologia ágil Srum prescreve, como já abordado no capítulo 3.

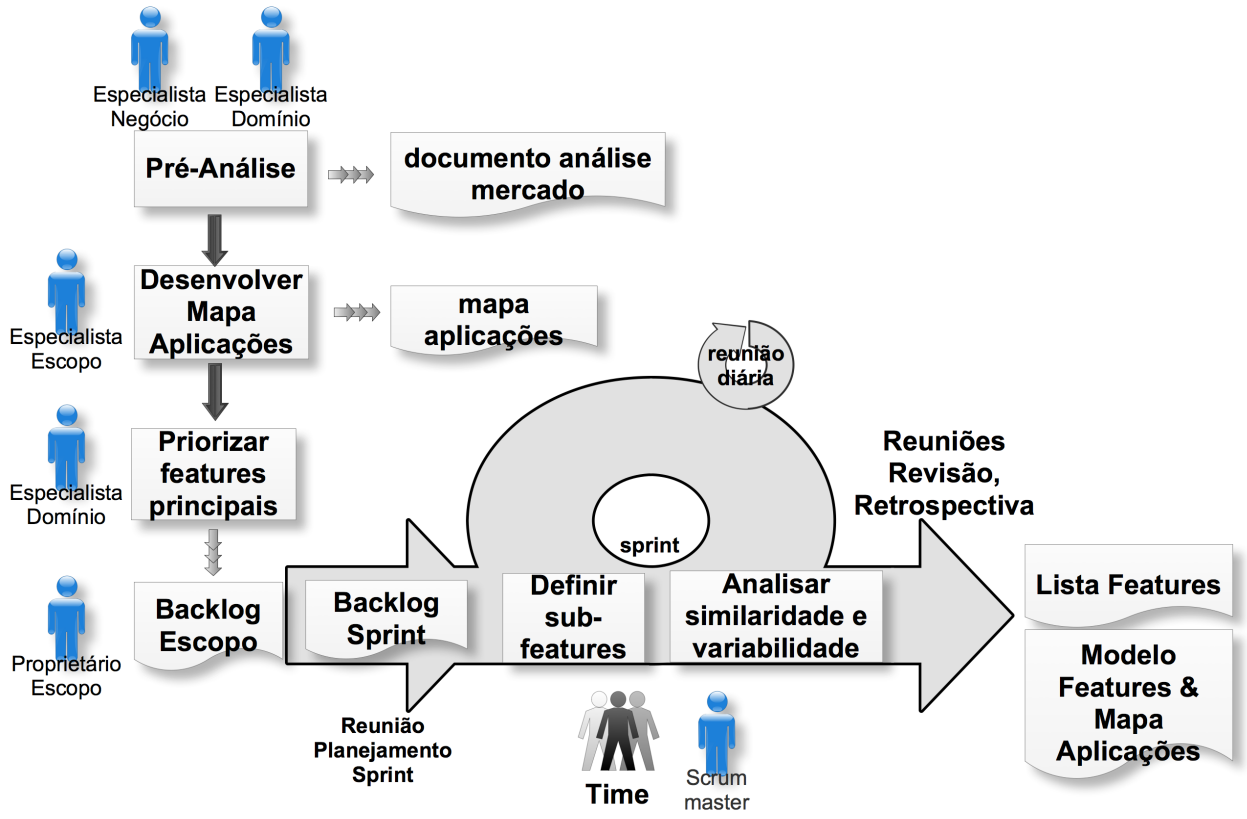


Figura 4.15: Ciclo do processo de LPS com Scrum

Nas sessões a seguir iremos trabalhar o exemplo da LPS da empresa AGM inserido na ferramenta LiProMA e explicando os passos e o seu funcionamento. O exemplo é o encontrado na sessão 2.4.

4.3.1 Gerenciamento de Domínios

Primeiro é necessário o cadastro dos domínios da LPS na ferramenta. A tela simples onde é necessário somente o nome e uma descrição do domínio a ser cadastrado. Na AGM possui um único domínio "Jogos de Arcade".Veja figuras 4.16 e 4.17

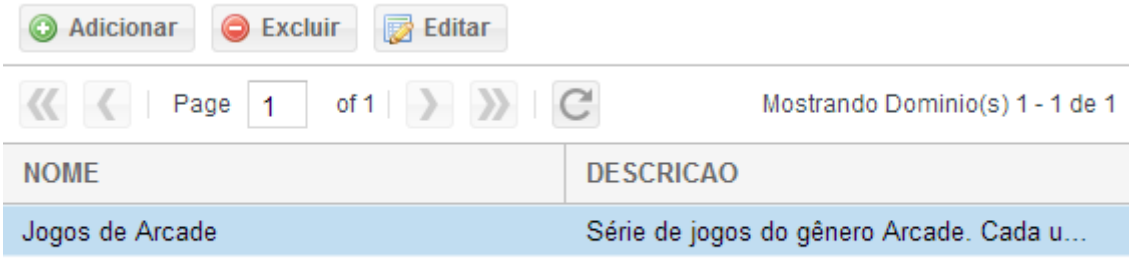


Figura 4.16: Captura da tela de dominío no sistema LiProMA.

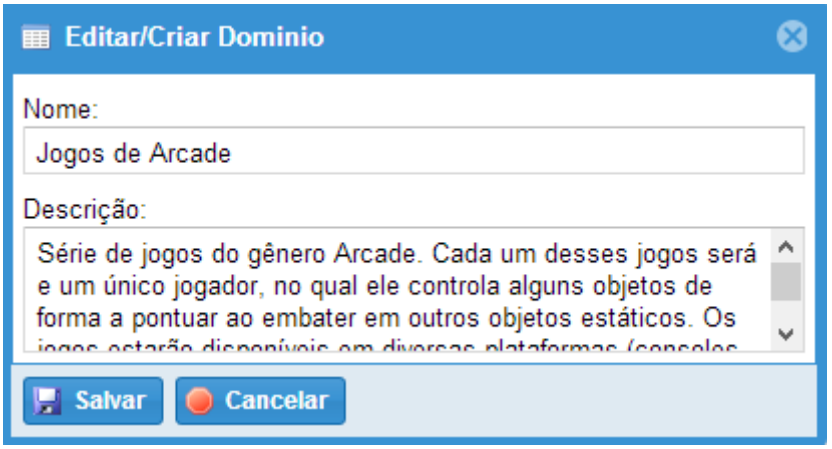


Figura 4.17: Captura da tela do formulário de cadastro/edição de dominío no sistema LiProMA

4.3.2 Análise de Mercado

Com o domínio cadastrado é possível o especialista de negócio inserir a análise de mercado no sistema. No cadastro de análise de mercado na ferramenta possibilitar o engajamento de domínios na analise de mercado. As figura 4.19 e 4.18 exibem como foi implementado no LiProMA essa tabela e o exemplo dentro do contexto da empresa AGM.

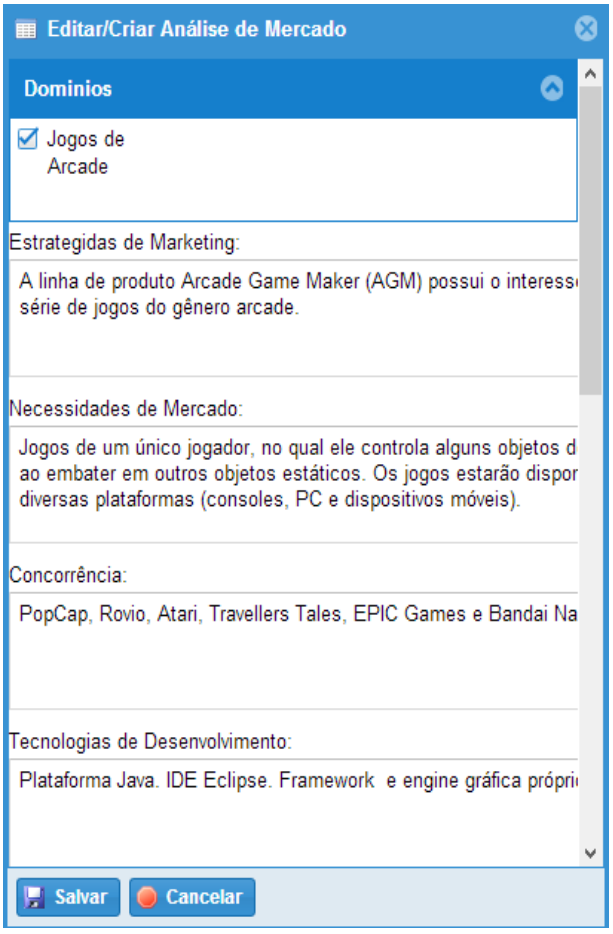


Figura 4.18: Captura da tela do formulário de análise de mercado do sistema LiProMA

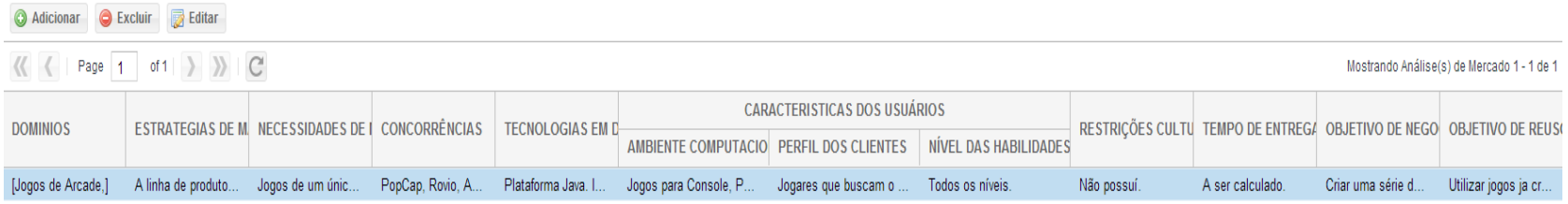


Figura 4.19: Captura da tela de listagem de análise de mercado do sistema LiProMA.

4.3.3 Gerenciamento de *Features*

Próxima etapa para a gestão de escopo na ferramenta LiProMA é o cadastro de *features* na LPS. As *features* são cadastradas levando em conta se ela é principal, quais são os seus domínios, em caso existir uma *feature* pai, ou seja, ela pertence a uma árvore de derivações de uma *feature*, é selecionado, caso não exista basta deixar a caixa de texto em branco. A prioridade de uma *feature* varia de 0 (Muito alta) até 4 (Muito baixa), esse campo é necessário para o especialista que estiver criando uma *Sprint* para o processo identificar quais as *features* devem ser processadas primeiro.

Na empresa AGM as features cadastradas são diversas, e podem ser visualizadas nas imagens 4.21 e 4.20.

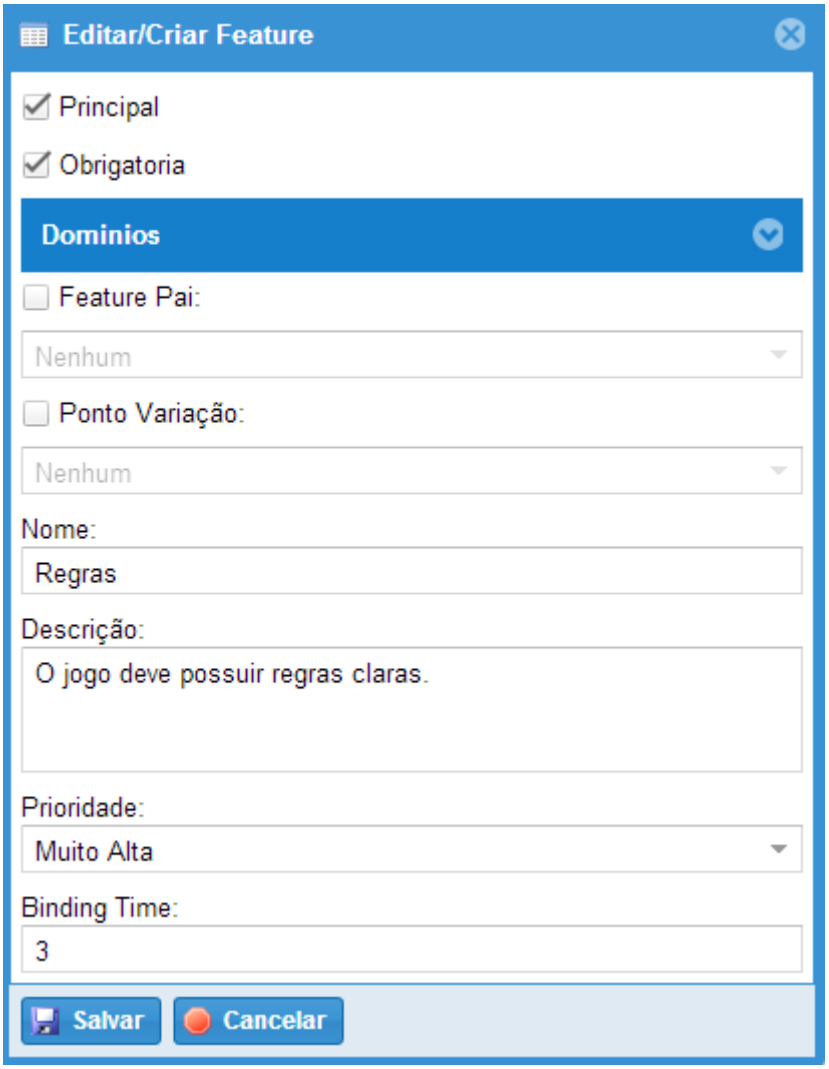
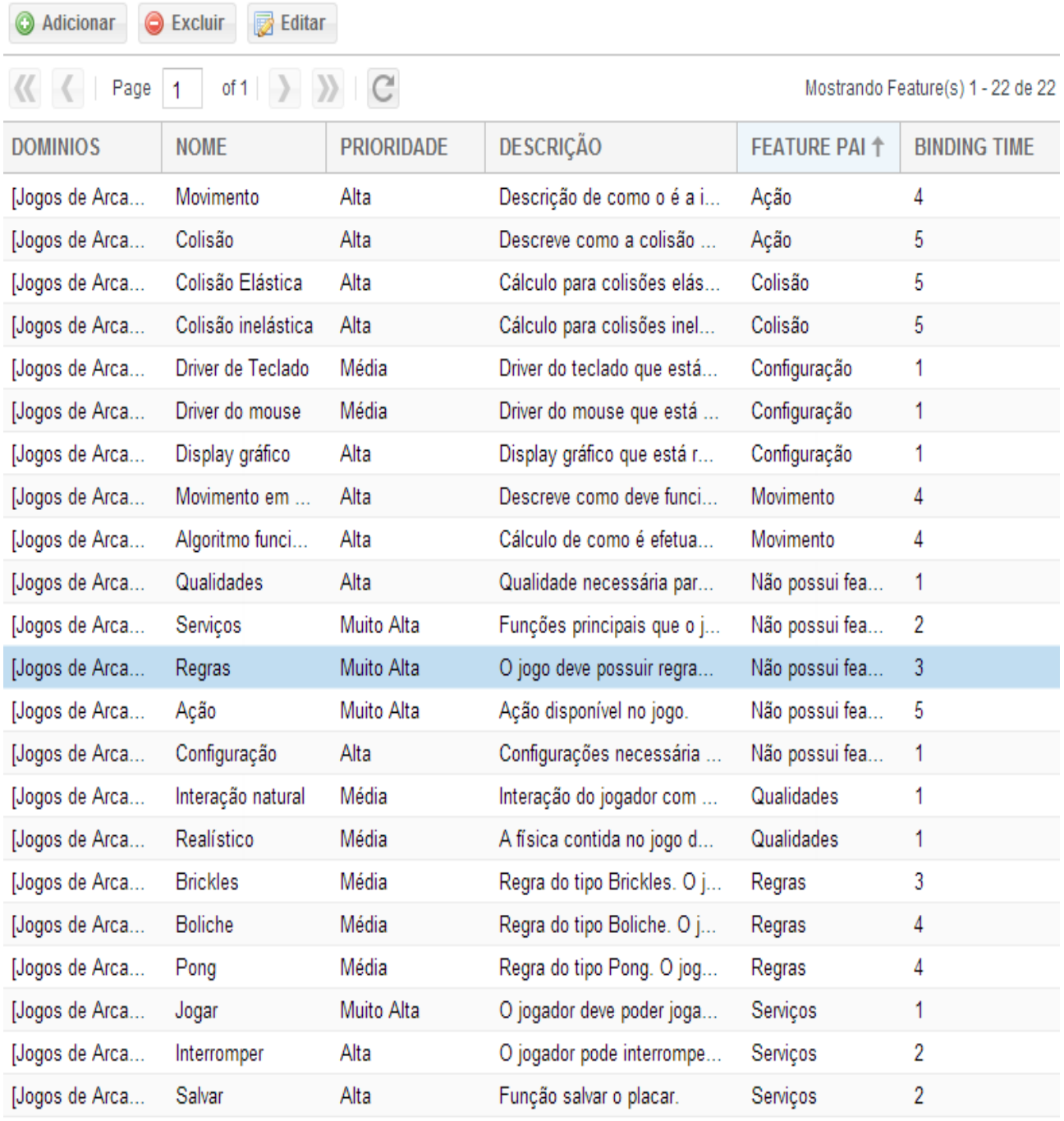


Figura 4.20: Captura da tela do formulário de cadastro/edição de uma *feature* no sistema Li-ProMA



DOMINIOS	NOME	PRIORIDADE	DESCRIÇÃO	FEATURE PAI	BINDING TIME
[Jogos de Arca...	Movimento	Alta	Descrição de como o é a i...	Ação	4
[Jogos de Arca...	Colisão	Alta	Descreve como a colisão ...	Ação	5
[Jogos de Arca...	Colisão Elástica	Alta	Cálculo para colisões elás...	Colisão	5
[Jogos de Arca...	Colisão inelástica	Alta	Cálculo para colisões inel...	Colisão	5
[Jogos de Arca...	Driver de Teclado	Média	Driver do teclado que está...	Configuração	1
[Jogos de Arca...	Driver do mouse	Média	Driver do mouse que está ...	Configuração	1
[Jogos de Arca...	Display gráfico	Alta	Display gráfico que está r...	Configuração	1
[Jogos de Arca...	Movimento em ...	Alta	Descreve como deve funci...	Movimento	4
[Jogos de Arca...	Algoritmo funci...	Alta	Cálculo de como é efetua...	Movimento	4
[Jogos de Arca...	Qualidades	Alta	Qualidade necessária par...	Não possui fea...	1
[Jogos de Arca...	Serviços	Muito Alta	Funções principais que o j...	Não possui fea...	2
[Jogos de Arca...	Regras	Muito Alta	O jogo deve possuir regra...	Não possui fea...	3
[Jogos de Arca...	Ação	Muito Alta	Ação disponível no jogo.	Não possui fea...	5
[Jogos de Arca...	Configuração	Alta	Configurações necessária ...	Não possui fea...	1
[Jogos de Arca...	Interação natural	Média	Interação do jogador com ...	Qualidades	1
[Jogos de Arca...	Realístico	Média	A física contida no jogo d...	Qualidades	1
[Jogos de Arca...	Brickles	Média	Regra do tipo Brickles. O j...	Regras	3
[Jogos de Arca...	Boliche	Média	Regra do tipo Boliche. O j...	Regras	4
[Jogos de Arca...	Pong	Média	Regra do tipo Pong. O jog...	Regras	4
[Jogos de Arca...	Jogar	Muito Alta	O jogador deve poder joga...	Serviços	1
[Jogos de Arca...	Interromper	Alta	O jogador pode interrompe...	Serviços	2
[Jogos de Arca...	Salvar	Alta	Função salvar o placar.	Serviços	2

Figura 4.21: Captura da tela da listagem de *features* no sistema LiProMA.

4.3.4 Gerenciamento de Produto

Após cadastrar as *features* da LPS e seus domínios, a próxima etapa é o gerenciamento de Produto. Seu cadastro é constituído campo nome, descrição, qual domínio o produto pertence

e quais as *features* estão inclusas no produto. No exemplo da AGM, nessa etapa é cadastrado o jogo "Pong Supremo"como visto nas imagens 4.22 e 4.23.

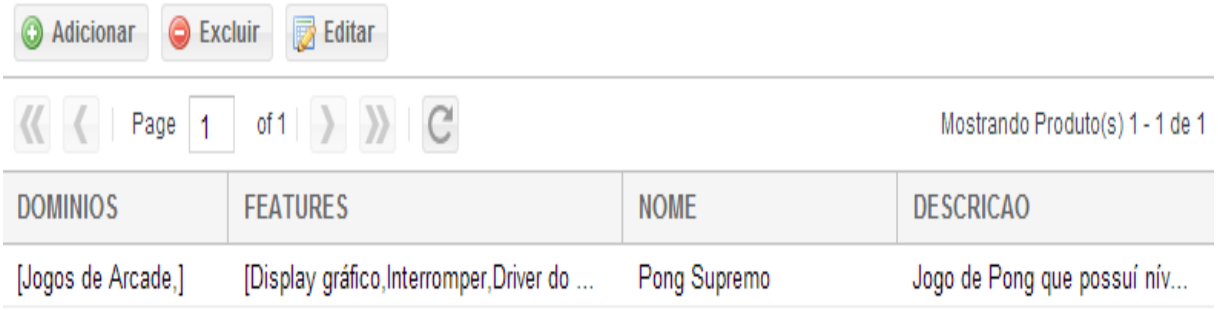


Figura 4.22: Captura da tela de Produto no sistema LiProMA.

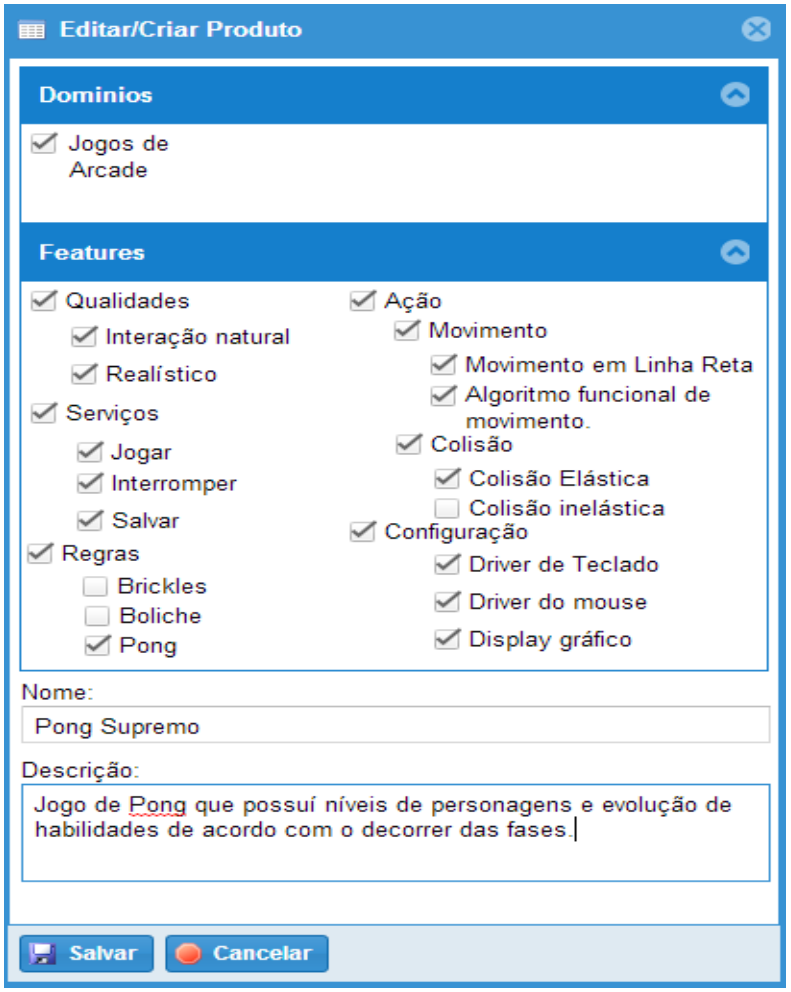


Figura 4.23: Captura da tela do formulário de cadastro/edição de Produto no sistema LiProMA

4.3.5 Gerenciamento do *Backlog* de Escopo

Com as *features* e os produtos devidamente cadastrados no sistema pode-se então gerenciar o escopo da LPS através da metodologia ágil Srum. Seu cadastro envolve os campos de descrição, identificar qual é o escopo que está sendo trabalhado, e a seleção das *features* principais. Também é possível selecionar um produto no cadastro de *backlog* de escopo, essa ação adiciona todas as *features* principais vinculadas ao produto selecionado.

Ainda no exemplo da LPS da empresa AGM, criamos um *backlog* de escopo com o nome de "Estudo sobre o produto "Pong Supremo, nele é adicionado o produto "Pong Supremo", com isso todas as features principais são automaticamente selecionadas. Veja figuras 4.24 e 4.25.

Figura 4.24: Captura da tela de *backlog* de Escopo no sistema LiProMA.

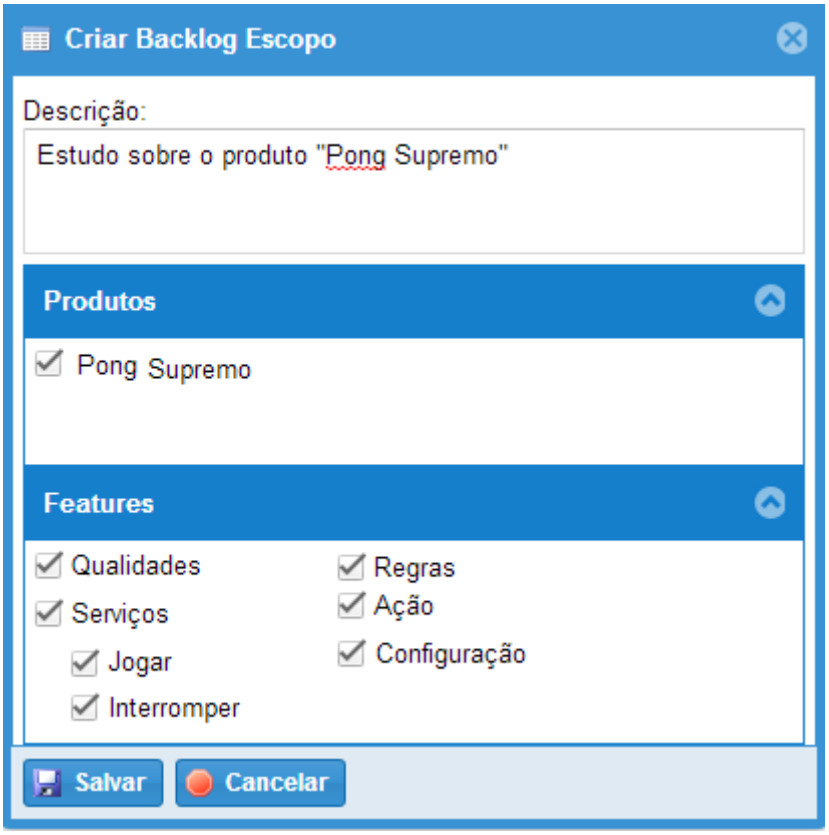


Figura 4.25: Captura da tela do formulário de cadastro/edição de um *Backlog* de Escopo no sistema LiProMA

Após criar um *backlog* de escopo é possível acessar a tela de *Feature Backlog* de Escopo. Nessa parte da ferramenta existe uma *combo box*, onde o especialista pode selecionar o escopo desejado. Após selecionar o escopo, o sistema exibe todas as *features* juntamente com seus domínios e prioridades e ao clicar em uma *feature* é possível definir suas estimativas. As estimativas são dados baseados em "pontos de história"e por sua vez são definidos por um especialista adequado.

Trabalhando no escopo do nosso exemplo da LPS AGM, nessa etapa são definidas as estimativas para cada uma dessas *features* da seguinte forma: em uma reunião, a equipe decide que a *feature* Serviços tem maior impacto, riscos e dificuldade de desenvolvimento se comparada a outras *features*, sendo assim tem maior peso. Então, é sugerido um peso 50 fixo para essa *feature*. As demais *features* são ponderadas comparando os esforços de desenvolvimento, quantidade de riscos e impacto se comparada a *feature* Serviços. Os resultados podemos observar nas figuras 4.26 e 4.27.

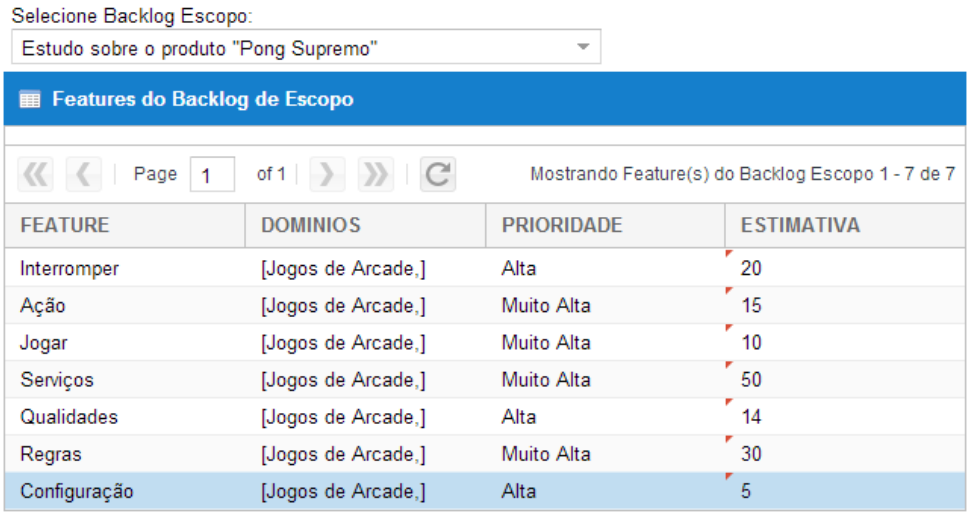


Figura 4.26: Captura da tela da listagem de *features* pertencentes ao *backlog* de Escopo no sistema LiProMA.

Figura 4.27: Captura da tela do formulário de edição de uma estimativa para uma *feature* contida no *Backlog* de Escopo no sistema LiProMA

4.3.5.1 Gerenciamento do *backlog* de *Sprints*

Com o *backlog* de escopo todo definido, a próxima etapa é criar e gerenciar as *Sprints*. Para definir e criar uma *Sprint* é necessário preencher os campos de descrição da própria *Sprint*, definir e descrever o local de reuniões, as datas de início e término da *Sprint* e quais as *features* do *backlog* de escopo que farão parte dessa iteração. O funcionamento da listagem de *Sprint* ocorre da mesma forma da especificação *Feature Backlog* de Escopo, uma *combo box* com os *backlog* de escopo cadastrado, que quando selecionando exibe a lista de *Sprints* cadastradas

para o *backlog* de escopo.

Tendo em vista nosso exemplo da LPS da AGM, cadastramos uma *Sprint* identificada pela descrição "Sprint 1 - Analisar feature Ação"que irá trabalhar a *feature* "Ação". As reuniões irão acontecer na Sala de Reunião da empresa. E essa *Sprint* tem as datas de começo como dia 8 de novembro de 2013 e encerramento em 22 de novembro de 2013. Conforme imagens 4.28 e 4.29.

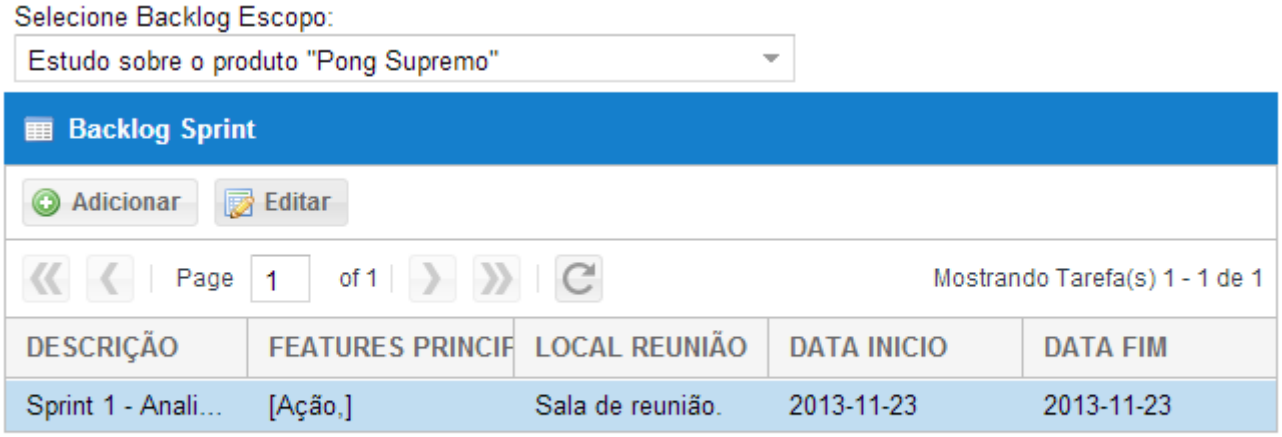


Figura 4.28: Captura da tela de backlog de uma *Sprint* no sistema LiProMA.

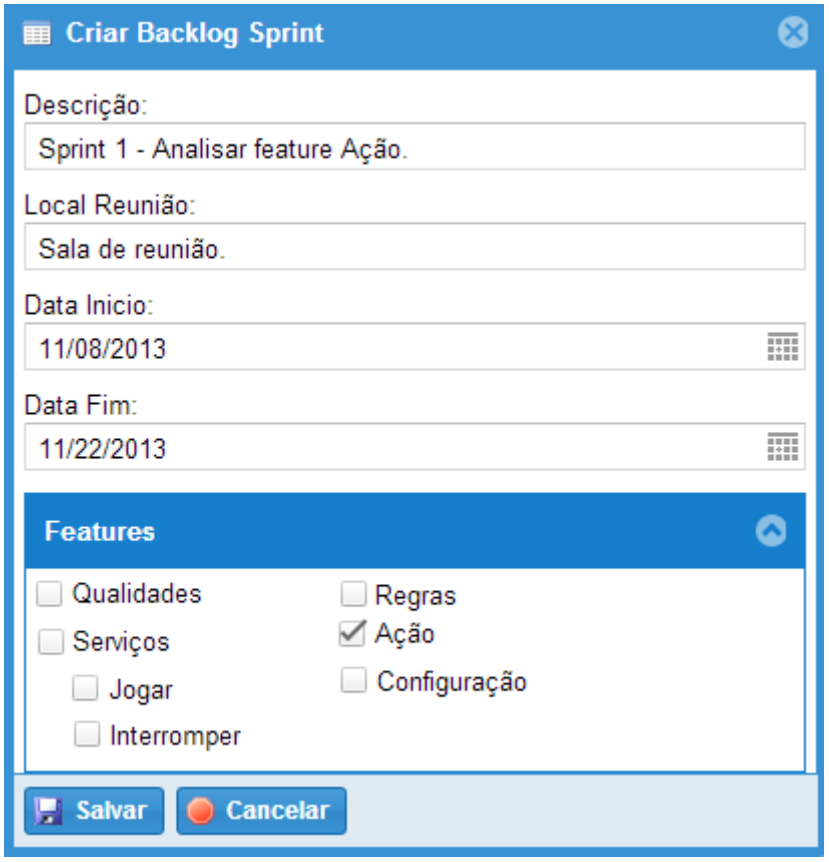


Figura 4.29: Captura da tela do formulário de cadastro/edição de uma *Sprint* no sistema LiProMA

4.3.5.2 Gerenciamento dos Responsáveis

A criação e gerenciamento de responsáveis dentro da ferramenta LiProMA podem ocorrer a qualquer momento, ou seja, sem a necessidade de outras etapas vistas nas sessões anteriores. Seu cadastro na ferramenta é somente inserir o nome completo em um formulário simples. Os responsáveis são aqueles indivíduos que o *Srum Master* definiu na reunião como "responsável por determinada tarefa"dentro da *Sprint*.

No exemplo da LPS da AGM, os responsáveis pelas tarefas atribuídas nas *Sprints* são: Elder, Henrique, Ivonei, Tharle e Victor. Veja as imagens 4.30 e 4.31.

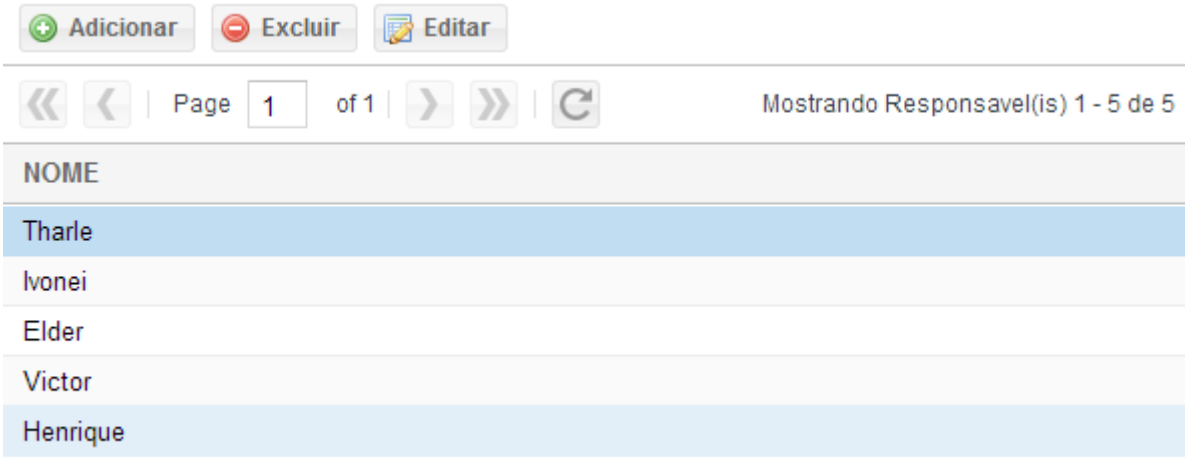


Figura 4.30: Captura da tela de listagem de responsáveis no sistema LiProMA.

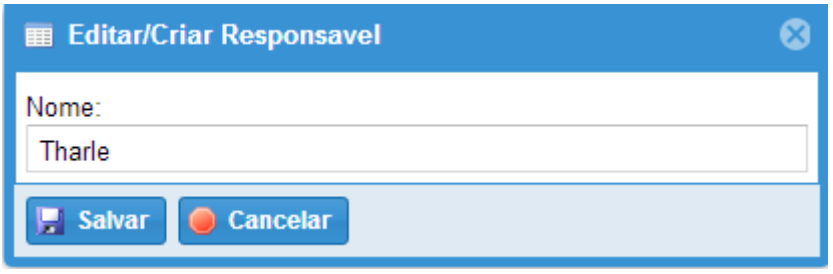


Figura 4.31: Captura da tela do formulário de cadastro/edição de um responsável no sistema LiProMA

4.3.5.3 Gerenciamento das Tarefas

As tarefas são descriões de atividades existentes na Sprint. Para criar uma tarefa na ferramenta LiProMA é necessário preencher sua descrição, detalhando brevemente uma ação ou objetivo. A tarefa é vinculada ao selecionar o *backlog* de escopo desejado e, consequentemente, o *backlog* da Sprint desejado. Por último são selecionados quais são os responsáveis por essa tarefa. As novas tarefas são automaticamente setadas com o status igual a "Nova".

As tarefas cadastradas são exibidas em uma listagem. Ao editar uma tarefa, não se pode alterar os campos já cadastrado, entretanto, abre-se uma tela de diálogo para edição do status: Nova, A Fazer, Em Andamento, Feita e Cancelada. Não é possível excluir uma tarefa, para isso é necessário alterar o status da mesma para cancelada.

A *Sprint* "*Sprint* 1 - Analisar *feature* Ação"LPS da AGM é divida em 3 tarefas: tarefa "Verificar se as *subfeatures* de Ação podem ser principais."responsabilidade do Ivonei, tarefa "Verificar se existe uma *subfeature* candidata para a *feature* de Ação."responsabilidade do Henrique, Tharle do Victor e a tarefa "Documentar o Processo da *feature* Ação dentro do produto Pong Supremo"responsabilidade do Tharle e do Elder. Veja figuras 4.32, 4.33 e 4.34.

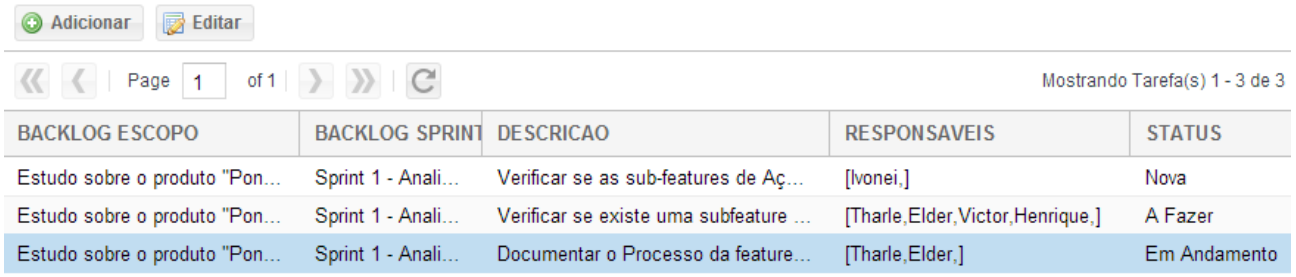


Figura 4.32: Captura da tela de listagem de tarefas no sistema LiProMA.

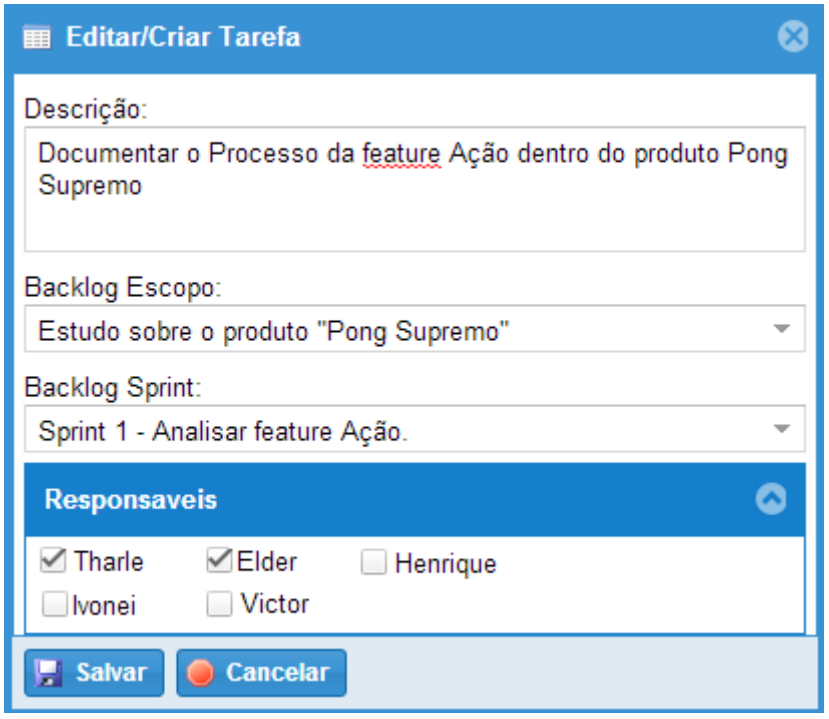


Figura 4.33: Captura da tela do formulário de cadastro/edição de uma tarefa no sistema LiProMA

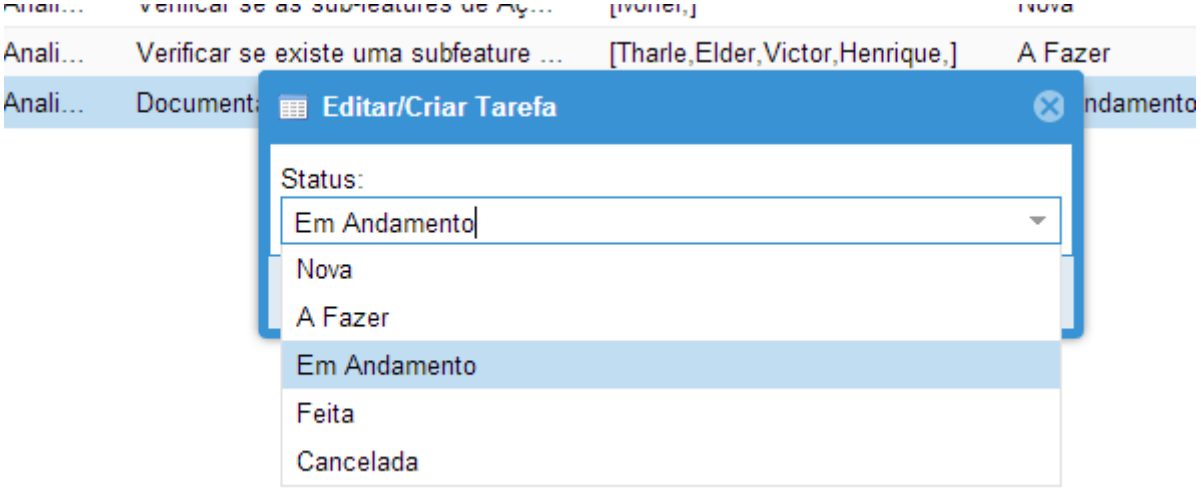


Figura 4.34: Captura da tela do formulário de alteração de status de uma tarefa no sistema LiProMA

Capítulo 5

Conclusão

Trabalhar com uma linha de produto junto as metodologias ágeis possui vários benefícios para os especialistas em escopo e de domínio, como ter uma estrutura consolidada como uma metodologia ágil para a gestão de seu escopo. Entretanto, ao explorar essas duas abordagens de desenvolvimento de sistemas juntas resulta uma diversidade de informações com forte ligação entre si dispersas em várias ferramentas e documentos separados.

A ferramenta proposta LiProMA busca integrar a documentação da análise de escopo e ED com o método ágil Scrum. Contudo, a ferramenta proposta não abrange toda engenharia de domínio, não possui um mapa de *feature* interativo e outras funcionalidades se comparada com ferramentas que trabalhem com escopo e LPS ativas no mercado e no âmbito científico.

Todavia, a LiProMA contém seu diferencial das outras ferramentas, como a possibilidade de aplicar várias LPS concorrentemente, integrar todos padrões de gestão de escopo de uma LPS com a metodologia ágil Scrum e ser uma ferramenta de ambiente Web, na qual tanto o sistema quanto sua base de dados pertencem a um servidor web e seu acesso pode ocorrer de forma remota gerando assim uma base de dados rica em informações que podem ser aproveitados para estudos ou relatórios.

5.1 Trabalhos Futuros

Esse trabalho teve seu foco na construção da base para ferramenta LiProMA e suas funcionalidades em relação a gestão de escopo de uma LPS. Esse trabalho é somente o primeiro passo dessa ferramenta, ainda existem muitas funcionalidades a ser implementadas e melhoradas no LiProMA, tais como:

- Gerar relatório de similaridade;
- Utilizar de código fonte legado dos produtos;
- Avaliação Empírica da ferramenta em laboratórios e em indústrias de software (cenário real);
- Criar/Manipular diagramas, na forma de interface para melhorar a usabilidade e experiência do usuário.
- Trabalhar as outras etapas da ED.

Glossário

Overhead	Um processamento ou armazenamento em excesso, seja de tempo, de materiais, de informações ou condições impeditivas para executar uma determinada tarefa. Como consequência, pode piorar o desempenho organizacional.
Backlog do Produto	é um conjunto de requitos e *features* elaborados previamente que, quando satisfeitos durante as *Sprints*, geram o produto pronto.
CASE	é uma classificação que abrange todas ferramentas baseadas em computadores que auxiliam atividades de engenharia de software, desde análise de requisitos e modelagem até programação e testes.

Referências Bibliográficas

[1] MORAES, M. B. S. *RiPLE SC: An Agile Scoping Process for SoftwareProduct Lines.* Dissertação (Mestrado) — Universidade Federal de Pernambuco, Recife, PE, Agosto 2010.

[2] POHL K.; BOCKLE, G.; LINDEN, F. V. D. *Software product line engineering: foundations, principles, and techniques*. [S.l.]: Springer, 2005.

[3] INSTITUTE, S. E. *Arcade Game Maker Pedagogical Product Line: Scope*. Consultado na Internet em: 18/11/2013. Disponível em: https://www.sei.cmu.edu/productlines/ppl/scope.html.

[4] SCHUWABER, K. *Agile Project Management with Scrum*. [S.l.]: Microsoft Press, 2004. ISBN 978-0-7356-1993-7.

[5] SOMMERVILE, I. *Engenharia de Software*. 8. ed. [S.l.]: Addison-Wesley Longman Publishing Co., 2007.

[6] SILVA, I. F. e. a. Agile software product lines: a systematic mapping study. software: Practice and experience. 2011.

[7] AL, L. H. Vulcan: A workbench for feature-oriented product line software development. *Korea-Japan Joint Workshop on ICT, Paper 08*, Setembro 2012.

[8] LISBOA, B. L. *ToolDAy: A Tool for DomainAnalysis*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, Recife, PE, Novembro 2008.

[9] CLEMENTS, L. N. *Software product lines: Praticles and Patterns*. Boston, MA, USA: AddisonWesley Longman Publishing Co., 2001.

[10] VARGAS, R. V. *Gerenciamento de Projetos (6a edição)*. [S.l.]: Brasport, 2006.

[11] BALBINO M.; ALMEIDA, E. S.; MEIRA, S. R. L. A scoping process for software product lines. *23rd International Conference on Software Engineering and Knowledge Engineering*, Miami - USA.

[12] EZRAN, M.; MORISIO, M.; TULLY, C. J. *Practical software reuse*. [S.l.]: Springer, 2002.

[13] HANSSEN, G. K.; FÆGRI, T. E. Process fusion: An industrial case study on agile software product line engineering. *The Journal of Systems and Software*, 2008. V. 81, n. 6, 2008, pp. 843 - 854.

[14] NOOR M. A.; RABISER, R.; GRÜNBACHER, P. Agile product line planning: A collaborative approach and a case study. *The Journal of Systems and Software*, 2008. V. 81, n. 6, 2008, pp. 868 - 882.

[15] TIAN, K.; COOPER, K. Agile and software product line methods: Are they so different? *1st International Workshop on Agile Product Line Engineering (APLE'06), IEEE Computer Society*, Baltimore, Maryland, USA, Agosto 2006.

[16] TRINIDAD, P. e. a. Automated error analysis for the agilization of feature modeling. *The Journal of Systems and Software*, 2008. V. 81, n. 6, 2008, pp. 883 - 896.

[17] CARBON, R. e. a. Integrating product line engineering and agile methods: Flexible design up-front vs. incremental design. *1st International Workshop on Agile Product Line Engineering (APLE'06)*, Baltimore, Maryland, USA, Agosto 2006. V. 81, n. 6, 2008, pp. 883 - 896.

[18] GHANAM, Y. e. a. A test-driven approach to establishing and managing agile product lines. *The 5th Software Product Lines Testing Workshop (SPLiT)*, Setembro 2008.

[19] PAIGE, R. F. e. a. Process fusion: An industrial case study on agile software product line engineering. *Lecture Notes in Computer Science, v. 4044, Proceedings from 7th International Conference eXtreme Progamming*, 2006. Pp. 198-199.

[20] PETTERSSON, U.; JARZABEK, S. Industrial experience with building a web portal product line using a lightweight, reactive approach. *ACM SIGSOFT Software Engineering*. V. 30, n. 5, September, 2005, pp. 326-335.

[21] RAATIKAINEN, M. e. a. Integrating product family modeling with development management in agile methods. *Proceedings of the 1st international workshop on Software development governance*, 2008.

[22] CZARNECKI, K.; EISENECKER, U. W. *Generative Programming: Methods, Tools, and Applications*. [S.l.]: Addison-Wesley Longman Publishing Co., 2000.

[23] FREEMAN, R. *Strategic Management: A Stakeholder Approach*. [S.l.: s.n.], 1984.

[24] LEITE, A.; GIRARDI, R. Um processo para a engenharia de domínio e de aplicações multiagente: As fases de projeto de domínio e de aplicações. *III Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de software*, 2009.

[25] MACHADO, M. S.; AL. et. Uma ferramenta de apoio à aquisição cooperativa de conhecimento em engenharia de domínio. *Caderno de Ferramentas do XIII Simpósio Brasileiro de Engenharia de Software*, Florianópolis - SC, 1999.

[26] STEFFEN, J. B. *O que são essas tais de metologias Ágeis*. Janeiro 2012. Consultado na Internet em: 05/05/2013. Disponível em: https://www.ibm.com/developerworks.

[27] ONE, V. *Version One*. Consultado na Internet em: 11/11/2013. Disponível em: http://www.versionone.com/.

[28] SCHUWABER, K.; SUTHERLAND, J. *Um guia definitivo para o Scrum: As regras do jogo*. 2011. Consultado na Internet em: 08/05/2013. Disponível em: http://www.scrum.org.

[29] AUDY, J. H. *Uma breve introdução ao método ágil Scrum*. 2011. Consultado na Internet em: 20/05/2013. Disponível em: http://www.baguete.com.br/colunistas/colunas/1173/jorge-horacio-audy/18/01/2013/uma-breve-introducao-ao-metodo-agil-scrum.

[30] SENCHA. *Sencha Extjs: Javascript Framework for Rich Desktop Apps*. Consultado na Internet em: 19/08/2013. Disponível em: http://www.sencha.com/products/extjs.

[31] CROCKFORD, D. *RFC 4627: The application/json Media Type for JavaScript Object Notation (JSON)*. Julho 2006. Disponível em: http://www.ietf.org/rfc/rfc4627.txt?number=4627.

[32] JSON.ORG. *Introdução ao JSON*. Consultado na Internet em: 08/08/2013. Disponível em: http://json.org/json-pt.html.

[33] POSTGRES. *Comunidade Brasileira de Postgres*. Consultado na Internet em: 8/10/2013. Disponível em: http://www.postgresql.org.br/.

[34] JBOSS. *HIBERNATE*. Consultado na Internet em: 12/9/2013. Disponível em: http://www.hibernate.org/.