Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologias
Curso de Ciência da Computação

HANS NEWTON FONSECA CANTANHEDE

APLICANDO A MÉTRICA PONTOS DE FUNÇÃO À METODOLOGIA *SCRUM*

São Luís
2016

HANS NEWTON FONSECA CANTANHEDE

APLICANDO A MÉTRICA PONTOS DE FUNÇÃO À METODOLOGIA *SCRUM*

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Profº Dr. Carlos de Salles Soares Neto

São Luís

2016

Cantanhede, Hans Newton Fonseca
APLICANDO A MÉTRICA PONTOS DE FUNÇÃO À METODOLOGIA *SCRUM*/ Hans Newton Fonseca Cantanhede. – São Luís, 2016.
52f.

Orientador: Dr. Carlos de Salles Soares Neto

Monografia (Graduação) – Universidade Federal do Maranhão, Curso de Ciência da Computação, 2016.

1. SCRUM 2. Pontos de Função 3. Estimativa 4. Sprints

CDU 004.4

Hans Newton Fonseca Cantanhede

APLICANDO A MÉTRICA PONTOS DE FUNÇÃO À METODOLOGIA *SCRUM*

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado. São Luís, 15 de JULHO de 2016:

Carlos de Salles Soares Neto

Prof. Dr. Carlos de Salles Soares Neto
(Orientador)
Universidade Federal do Maranhão

Prof. Dr. Luciano Reis Coutinho
Universidade Federal do Maranhão

Maria Auxiliadora Freire

Prof. Msc. Maria Auxiliadora Freire
Universidade Federal do Maranhão

São Luís
2016

"Dedico a todos que me apoiaram e acreditaram na minha capacidade para realizar este trabalho"

Agradecimentos

Primeiramente quero agradecer a **Deus** pelo dom da vida e por me dar toda a vontade e a sabedoria para escrever.

Agradeço a toda a minha família: minha mãe **Ana Maria**, meu pai **Raimundo Cantanhede**, minha irmã **Aretha Lorena** e meu cunhado **Nelson Carneiro** por todo apoio e incentivo que deram durante todo o curso e sobretudo durante a realização deste tabalho.

Agradeço aos meus amigos: **Willian Mano**, **André Felipe** e **Thayane Soares**, que sempre se fizeram presentes na minha vida e auxiliaram para finalizar esta monografia.

Agradeço ao meu orientador **Carlos de Salles Soares Neto** pela paciência e por ter me guiado nos momentos, esclarecendo dúvidas, sugerindo idéias durante a realização do trabalho.

"Muitos me perguntam
qual o segredo para o meu sucesso...
Existem apenas duas regras:
1: Trabalhe pra valer;
2: Nunca ouça aos que só dizem não."
(Arnold Schwarzenegger)

Resumo

Produzir software de qualidade, em tempo hábil, respondendo rápido as mudanças no escopo e sem perder a qualidade do produto é, sem dúvidas, um desafio. Este trabalho mostra a viabilidade da integração entre a metodologia SCRUM e a métrica Pontos de Função em projetos de desenvolvimento de software.

A metodologia SCRUM, uma metodologia ágil focada na adaptabilidade, flexibilidade e produtividade unida a Pontos de Função, uma métrica simples e consistente e ainda flexível a mudança de escopo do sistema mostrou bons resultados para aqueles que precisam de organização e velocidade na tomada de decisões quanto ao projeto e desenvolvimento de software. Boas disciplinas que geralmente ao serem utilizadas de forma isolada mostraram-se excelentes em conjunto.

Palavras-chaves: scrum, pontos de função, estimativa, sprints.

Abstract

Produce a good software and in time, responding quickly to changes in the scope without losing the quality of the final product is undoubtedly a challenge. This work shows the viability of integration between the SCRUM methodology with the metric Function Points in software development projects.

The SCRUM methodology, an agile methodology focused on adaptability, flexibility and productivity united to Function Points, a simple and consistent metric and yet flexible switching system scope showed good results for those who need organization and speed in making decisions in design and development of software. Good disciplines that are usually used in isolated form proved to be great together.

Key-words: scrum. function point. estimate. sprints

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

A automação de processos manuais através de sistemas já é uma realidade atual e praticamente parte do sucesso de qualquer empresa. Algumas vezes encontra-se uma solução pronta que se encaixa perfeitamente à realidade do negócio da empresa, outras vezes não. Quando não, começa-se a procura por especialistas no desenvolvimento de sistemas que possam criar um sistema próprio à empresa.

Um fator importante no desenvolvimento de sistemas é o tempo necessário para entrega de um sistema pois, geralmente, as empresas contratantes deste tipo de serviço demandam uma conclusão rápida, já que necessitam que o sistema entre em ação para auxílio dos processos internos o quanto antes.

Outro aspecto importante é a estimativa das funcionalidades que serão incorporadas no sistema. Se o especialista não se servir de um método eficiente, ele pode errar a estimativa de produção do software para o cliente, atrasando dessa forma os cronogramas e até aumentando custos.

1.1 Motivação

Produzir um sistema de forma rápida, que atenda às necessidades da empresa, seja flexível a mudanças no escopo e resulte em um produto de qualidade é de fato um desafio.

Este trabalho tem por objetivo mostrar um método fácil de projeto e organização de sistemas que vai desde a concepção, passando pelo desenvolvimento, até a entrega de softwares de qualidade e o oferecimento de um acompanhamento interativo ao cliente.

Uma metodologia compreende um conjunto de atividades ordenadas, com o objetivo de obter um produto. Ao desenvolver softwares, um problema recorrente é qual metodologia de desenvolvimento de software utilizar. Como já foi mencionada, uma condição que chega quase a ser uma obrigação do desenvolvedor é a rapidez na entrega do produto, não importando o tamanho ou complexidade do software. Dado que existem diversas metodologias disponíveis, é interessante escolher aquela que resolva tal condição.

O *SCRUM* enfatiza o uso de um conjunto de padrões de processos de software que provaram serem eficazes para projetos com prazos de entrega apertados, requisitos mutáveis e críticos de negócio. Cada um desses padrões de processos define um conjunto de ações de desenvolvimento (PRESSMAN, 2011, pg. 95).

Definida a metodologia, chega-se a outro problema: essa metodologia ágil, apesar de possuir um método arbitrário para medição, não possui um método formal de medição

das funcionalidades desenvolvidas. A métrica de tamanho funcional Pontos de Função (PF) afere o tamanho de um projeto ou aplicação de software, considerando as funcionalidades requisitadas e recebidas pelo usuário, independente da tecnologia e do processo de desenvolvimento utilizados (FERREIRA; HAZAN, 2008, pg. 2).

Um dos principais benefícios de se utilizar esta métrica é aferir o tamanho de uma funcionalidade, antes que esta seja desenvolvida e assim verificar sua viabilidade e saber o esforço necessário para realizá-la. Assim, unir estes conceitos de forma que trabalhem juntos é um desafio que se alcançado tornará muito mais eficiente o desenvolvimento de software.

Mas como uma técnica que é tão formal a ponto de exigir um detalhamento considerado dos requisitos do software, quando uma das principais exigências da metodologia utilizada preza justamente pela diminuição de toda essa "burocracia"?

Esta dúvida surge quando se vê o desenvolvimento de software como um processo completo e indivisível. Se fosse necessário definir todos os requisitos do sistema para fazer as medições, antes de começar a desenvolver o software, este questionamento faria todo sentido, mas quando se dividem as tarefas maiores em sub-tarefas, a quantidade de tempo necessária diminui consideravelmente, possibilitando que as mesmas sejam entregues interativamente e com a interação por parte do cliente.

Uma metodologia que possui pressupostos leves e objetivos auxiliados por um processo de medição simples e consistente tornam essas ferramentas indispensáveis para a correta finalização e sucesso dos projetos de software. É importante acrescentar que o *SCRUM* possui uma forma de dimensionar o que está sendo produzido, mas o que se deseja é otimizar esse processo (estimativa) em projetos de software da melhor forma possível, ficando ainda mais fundamentado para o cliente.

1.2 Objetivos

Geral

De modo geral o objetivo é demonstrar a viabilidade da integração entre a metodologia *SCRUM* e a métrica Pontos de Função, em projetos de desenvolvimento de *software*.

Específicos

Especificamente, os objetivos a serem destacados são:

- Apresentar conceitos e a fundamentação necessária ao entendimento sobre a metodologia *SCRUM* e a métrica Pontos de Função.

- Desenvolver um estudo de caso que proporcione o melhor entendimento da

integração entre o *SCRUM* e Pontos de Função.

- Evidenciar pontos positivos e negativos da integração, a partir do estudo de caso.

1.3 Organização do Trabalho

Inicialmente, será feita a coleta de uma referência bibliográfica sobre a metodologia *SCRUM* assim como Análise de Pontos de Função, a fim de compreender os conceitos, técnicas e saber aplicá-los corretamente. Assim, será possível elaborar etapas para a correta utilização da integração entre estes conceitos. Em seguida, será feito um estudo de caso baseado em um sistema real de forma que a integração possa ser aplicada e então analisar as vantagens e desvantages observadas.

É importante citar que como se trata de um método de projeto, algumas etapas relativas à parte técnica (e por técnica, refere-se a linguagens de programação, testes de software, questões de infraestrutura e qualquer conceito que se encaixe na parte do técnica do software a ser desenvolvido) e de produção de sistemas não será aprofundada em detalhes, por não se tratar do foco deste trabalho.

2 Pontos de Função

A atividade de desenvolver soluções tecnológicas de sistemas já faz parte da realidade atual. O aumento da demanda de novos projetos de softwares, além da manutenção necessária em softwares já existentes só comprova que organização e gerenciamento são requisitos para as empresas desse ramo que almejam o sucesso.

O desenvolvimento de sistemas é um ramo dinâmico no qual novas tecnologias surgem rapidamente, portanto, medir o tamanho de um sistema, a partir dessas tecnologias seria não-conforme, visto que cada tecnologia tem a sua particularidade.

Dado que o usuário sabe parte daquilo que deseja no seu software, é mais conveniente utilizar as requisições do usuário como base inicial, assim o alto nível de abstração será depois especificado para a linguagem ou tecnologia escolhida, sem que se perca a essência do software a ser desenvolvido. Dessa forma, a métrica Pontos de função foi criada para medir o tamanho funcional de uma funcionalidade implementada, a partir do ponto de vista do usuário.

2.1 Definição e caracterização

A métrica de tamanho funcional Pontos de Função (PF) afere o tamanho de um projeto ou aplicação de software, considerando as funcionalidades requisitadas e recebidas pelo usuário, independente da tecnologia e do processo de desenvolvimento utilizados (FERREIRA; HAZAN, 2008, pg. 2).

O IFPUG (*International Function Point User Group*) é o órgão que regimenta o CPM (Counting Practice Manual) que é um manual de práticas de contagem no qual as regras de contagem de pontos de função são devidamente explicadas, qualquer que seja o propósito da contagem (VAZQUEZ; SIMÕES; ALBERT, 2013, pg. 38).

Os principais benefícios obtidos pelas organizações com a utilização da métrica PF são os seguintes (VAZQUEZ; SIMÕES; ALBERT, 2013, pg. 57) :

- Possibilita a análise de produtividade e qualidade;
- Pode ser utilizada como padrão para comparação entre produtos de software;
- Apoia o gerenciamento de requisitos;
- Suporta análise de pacotes de software, considerando as necessidades da organização e as funcionalidades entregues pelo pacote;

- Pode ser usada na avaliação do porte de sistemas em Planos Diretores de Tecnologia da Informação (PDTI);
- Pode ser usada como base para o estabelecimento de contratos de software;
- Serve como insumo para as estimativas de esforço, prazo e custos.

2.2 Etapas

Conforme o manual de práticas de contagem (CPM) de pontos de função existem sete etapas para realizar a correta contagem, como explicado em (IFPUG, 2010, pg. 9), são elas:

2.2.1 Reunir informações necessárias

Nesta etapa todos os requisitos do usuário são definidos, pois é a partir deles que outras informações são retiradas. Há diversas técnicas de reunião, entrevista e *brainstorm* que são adotadas nessa etapa.

2.2.2 Determinar o tipo da contagem

Existem três tipos principais:

- Projeto de desenvolvimento: projeto para desenvolver e entregar a primeira versão de uma aplicação de software.
- Projeto de melhoria: a partir de um software, realiza-se apenas uma manutenção em alguma parte dele. Pode ser manutenção evolutiva ou melhoria funcional.
- Aplicação: compreende medir uma aplicação instalada.

2.2.3 Determinar o escopo da contagem e fronteiras da aplicação

O escopo da contagem refere-se a quais funcionalidades serão incluídas no processo de contagem. Essa etapa faz mais sentido quando o tipo de contagem é o de projeto de melhoria, pois somente algumas funcionalidades são consideradas na contagem.

As fronteiras da aplicação são importantes para definir o limite entre o usuário e o sistema, na visão do usuário. Vale lembrar que toda contagem de pontos de função é feita sobre uma fronteira estabelecida.

2.2.4 Identificar funções de dados e transação

Antes de definir as funções de dados e transação, é necessário entender o conceito de processo elementar que pode ser entendido como a menor unidade de atividade, que mantém um estado consistente após sua execução e é reconhecida pelo usuário.

A seguir é possível identificar outros elementos de extrema importância durante a contagem de pontos de função, que são os tipos funcionais de dados e transação:

- Arquivo Lógico Interno (ALI): grupo de dados logicamente organizado, que é reconhecido pelo usuário e mantido através de um processo elementar da aplicação que está sendo contada. A Figura 1 exemplifica a identificação de um ALI (venda) que é mantido pela aplicação Vendas, que está sendo contada.

- Arquivo de Interface Externa (AIE): grupo de dados logicamente organizado que é reconhecido pelo usuário e mantido através de um processo elementar de uma aplicação externa, mas que é referenciado pela aplicação que está sendo contada. A Figura 1 exemplifica a identificação de um AIE (funcionario) que é referenciado pela aplicação que está sendo contada.

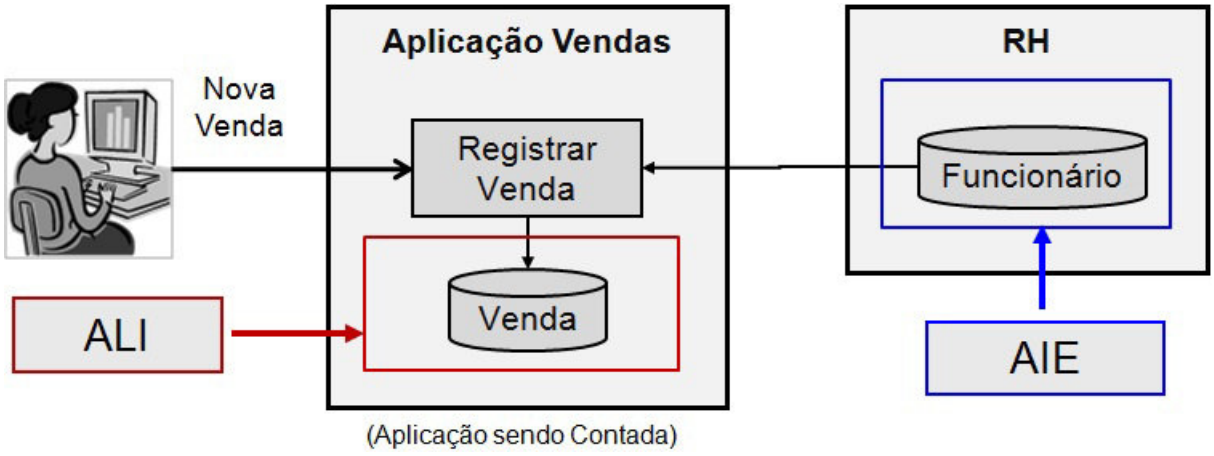


Figura 1: Identificação das funções de dados

Fonte: Adaptado de (CAMPOS, 2010)

- Consulta Externa (CE): é um processo elementar que envia dados ou informações de controle para fora da fronteira da aplicação. Seu objetivo principal é enviar informações através de sua recuperação por ALI ou AIE.

- Entrada Externa (EE): é um processo elementar que envia dados ou informações através da fronteira da aplicação. Seu objetivo principal é manter um ou mais ALI ou alterar o comportamento do sistema.

- Saída Externa (SE): é um processo elementar que envia dados ou informação de controle para fora da fronteira da aplicação. Seu objetivo principal é apresentar informação para um usuário ou outra aplicação através de um processamento lógico adicional à recuperação de dados ou informação de controle. O processamento lógico deve conter cálculo, criar dados derivados, manter ALI ou alterar o comportamento do sistema.

A Figura 2 exemplifica a identificação de um EE, CE e SE em uma aplicação que está sendo contada de acordo com objetivo da funcionalidade.

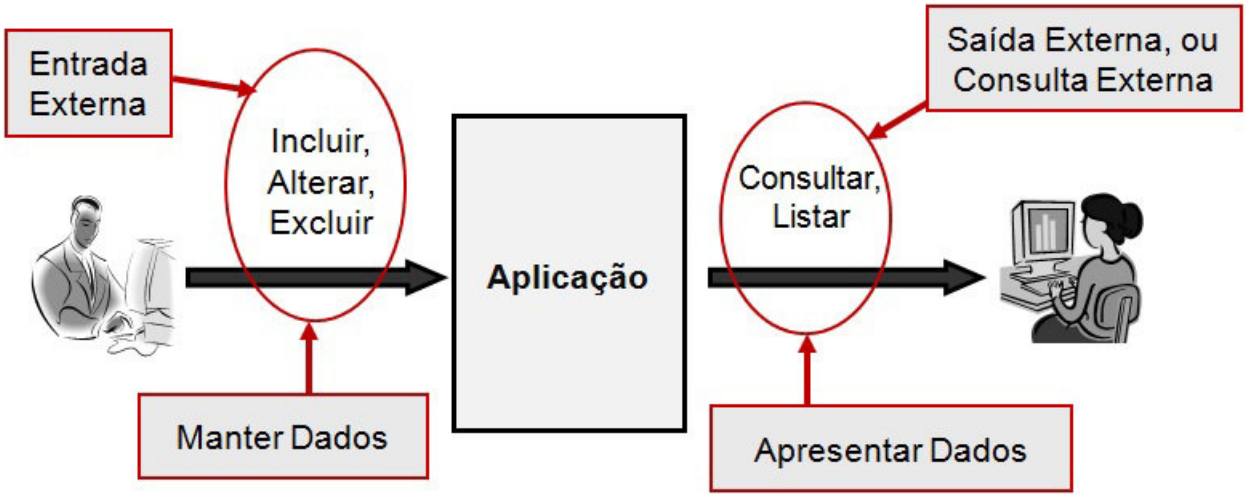


Figura 2: Identificação das funções de transação

Fonte: Adaptado de (CAMPOS, 2010)

2.2.5 Complexidade funcional

Antes de calcular o tamanho funcional é necessário estabelecer a contribuição de cada função que pode ser baixa, média ou alta. A complexidade funcional da função de dados depende da quantidade dos tipos de registros elementares da própria função e da quantidade de campos (tipos de dados) que são armazenados. A Tabela 1 auxilia na determinação da complexidade funcional de função de dados.

Tipo de dados(TDs): Campo único, não repetido e reconhecido pelo usuário. Tipo de registro elementar: Subgrupo de tipos de dados, reconhecido pelo usuário e componente de ALI ou AIE.

A complexidade funcional da função de transação depende da quantidade de arquivos referenciados (ARs) e da quantidade dos campos (Tipos de Dados - TDs). As Tabelas 2 e 3 auxiliam na determinação da complexidade funcional de função de transação.

Tabela 1: Complexidade funcional dos ALI e AIE

Tabela de Complexidade (ALI e AIE)			
Quantidade de tipos de registros	Tipo de dados (TDs)		
	<20	20 - 50	>50
1	Baixa	Baixa	Média
De 2 ate 5	Baixa	Média	Alta
Mais de 5	Média	Alta	Alta

Fonte: Adaptado de (VAZQUEZ; SIMÕES; ALBERT, 2013, pg. 85)

Tabela 2: Complexidade para entradas externas (EEs)

Tipos de Dados (TDs)				
Arquivos Referenciados (ARs)		<5	5 - 15	>15
	<2	Baixa	Baixa	Média
	2	Baixa	Média	Alta
	>2	Média	Alta	Alta

.

Fonte: Adaptado de (VAZQUEZ; SIMÕES; ALBERT, 2013, pg. 124)

Tabela 3: Tabela de complexidade para saídas externas (SEs) e consultas externas (CEs)

Tipos de dados (TDs)				
Arquivos Referenciados (ARs)		<6	6 - 19	>19
	<2	Baixa	Baixa	Média
	2 - 3	Baixa	Média	Alta
	>3	Média	Alta	Alta

Fonte: Adaptado de (VAZQUEZ; SIMÕES; ALBERT, 2013, pg. 124)

2.2.6 Calcular tamanho funcional

O tamanho funcional representa uma medida dos requisitos funcionais do usuário que foram classificados como funções de dados e de transações e que foi transformado num valor que corresponde a uma medida para cada uma das funções (dados e transações), como pode ser verificado na Tabela 4.

Para cada tipo funcional realiza-se um cálculo simples de multiplicar o número de elementos pelo valor de pontos de função, associado à sua complexidade e depois somado. A Tabela 5 mostra um resumo desse cálculo.

Tabela 4: Contribuição funcional dos Tipos Funcionais de dados e transação

	Complexidade		
Tipo Funcional	**Baixa**	**Média**	**Alta**
Arquivo Lógico Interno (ALI)	7 PF	10 PF	15 PF
Arquivo de Interface Externa (AIE)	5 PF	7 PF	10 PF
Entrada Externa (EE)	3 PF	4 PF	6 PF
Saída Externa (EE)	4 PF	5 PF	7 PF
Consulta Externa (CE)	3 PF	4 PF	6 PF

Fonte: Adaptado de (IFPUG, 2010, pgs. 6-9, 7-20)

Tabela 5: Cálculo da contribuição para o tamanho funcional

Tipo de Função	**Complexidade Funcional**	**Totais de Complexidade**	**Totais tipo função**
ALI	_____ Baixa	X 7 = _____	
	_____ Média	X 10 = _____	
	_____ Alta	X 15 = _____	
			_____
AIE	_____ Baixa	X 5 = _____	
	_____ Média	X 7 = _____	
	_____ Alta	X 10 = _____	
			_____
EE	_____ Baixa	X 3 = _____	
	_____ Média	X 4 = _____	
	_____ Alta	X 6 = _____	
			_____
CE	_____ Baixa	X 3 = _____	
	_____ Média	X 4 = _____	
	_____ Alta	X 6 = _____	
			_____
SE	_____ Baixa	X 4 = _____	
	_____ Média	X 5 = _____	
	_____ Alta	X 7 = _____	
			_____
		Tamanho funcional total	_____

Fonte: Arquivo do autor (2016).

2.2.7 Documentar e relatar

Esta etapa consiste em documentar e relatar a estimativa feita, seja com documentações em papel ou utilizando sistemas para armazenamento em bancos de dados. O importante é que estes dados estejam diponíveis para futuras iterações e possam ser atualizados, caso seja necessário.

3 SCRUM

A ideia principal da metodologia *SCRUM* é a de cumprir os princípios do manifesto ágil (BECK; BEEDLE et al., 2013):

- **Indivíduos e interações** mais que processos e ferramentas
- **Software em funcionamento** mais que documentação abrangente
- **Colaboração com o cliente** mais que negociação de contratos
- **Responder a mudanças** mais que seguir um plano

Esse manifesto foi uma resposta ao fracasso de diversos projetos de software que utilizavam metodologia tradicionais de software. Ele ainda afirma que não devemos esquecer o valor dos itens à direita, mas que o foco maior deve ser o de atender os itens da esquerda (aqueles em negrito citados anteriormente).

3.1 Definição e Caracterização

Uma metodologia ágil com uma comunidade grande de usuários é a *SCRUM*, propondo uma forma de trabalho flexível que se adapte a ambientes em constante mudança. O *SCRUM* apresenta uma abordagem empírica que aplica algumas idéias da teoria de controle de processos industriais para o desenvolvimento de software, re-introduzindo as idéias de flexibilidade, adaptabilidade e produtividade (KOSCIANSKI; SOARES, 2007, pg. 200).

O *SCRUM* enfatiza o uso de um conjunto de padrões de processos de software que provaram ser eficazes para projetos com prazos de entrega apertados, requisitos mutáveis e críticos de negócio. No qual cada um desses padrões de processos define um conjunto de ações de desenvolvimento (PRESSMAN, 2011, pg. 95).

No *SCRUM* os projetos são divididos em ciclos (tipicamente mensais) chamados de Sprints. Ao final de cada ciclo, partes (incrementos) de software são produzidas em um curto espaço de tempo. As funcionalidades que serão desenvolvidas estão descritas no Product Backlog. Em todo início de uma *sprint* há uma *Sprint Planning Metting* na qual os itens que são desenvolvidos nessa *sprint* são priorizados.

Os itens são representados e mensurados com alguma métrica, geralmente utilizam-se as histórias de usuário. As histórias de usuário não fazem parte do framework *SCRUM*,

mas devido a sua popularidade aparecem vinculadas à metodologia. O time de desenvolvimento deve encontrar a melhor forma de representar os itens do *Product Backlog*.

No decorrer de uma *sprint*, a *Daily Scrum*, que é uma reunião breve e geralmente pela manhã, é realizada para disseminar o que foi feito no dia anterior, planejamento/priorização das atividades do dia, identificar recursos necessários e até mesmo impedimentos possíveis para as tarefas serem realizadas. Ao final de uma *sprint* a equipe apresenta as funcionalidades através de uma *Sprint Review Metting*. Seguido pelo *Sprint Retrospective*, a equipe inicia uma nova *sprint*. Este ciclo de vida e artefatos são melhor visualizados na Figura 3.

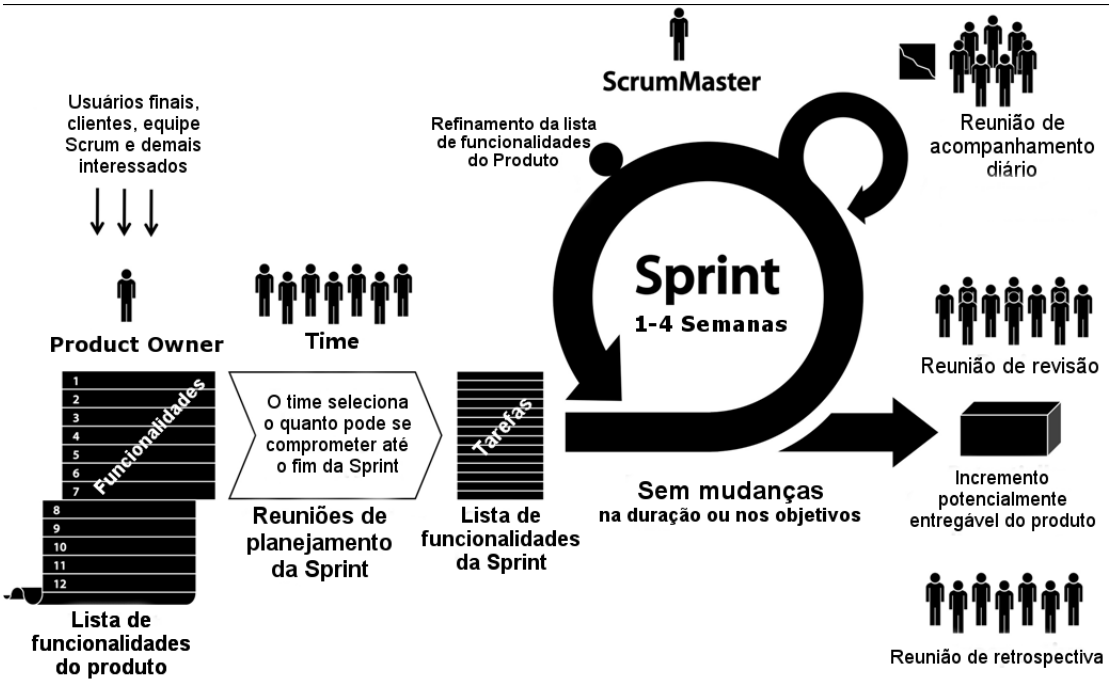


Figura 3: Ciclo de Vida do *SCRUM*

Fonte: Adaptado de (SUTHERLAND; SCHWABER, 2014, pg. 26)

Os eventos do *SCRUM* – a própria *Sprint* e as reunioes de *Sprint Planning*, de *Daily Scrum*, de *Sprint Review Metting* e de *Sprint Retrospective* – possuem uma duração máxima ou fixa defina que é chamada de *timebox*. Os *timeboxes* servem para evitar desperdício do tempo, e ajudam a criar um ritmo para o trabalho realizado.

3.2 Papéis no *SCRUM*

Existem três papéis principais nessa metodologia que são (RUBIN, 2012, pg. 14-16):

- *Product Owner* (Dono do produto) – O Dono do Produto tem a responsabilidade de informar ao *Scrum Master* os requisitos do sistema, gerenciar o *Product Backlog*, bem como priorizar o que deve ser feito pelo time de desenvolvimento durante a

sprint. Entende bem das regras de negócio e dá suporte ao time de desenvolvimento sempre que necessário.

- *SCRUM Master* (Perito em SCRUM) – Faz a intermediação entre os membros do Time Desenvolvimento e o Dono do Produto identificando as necessidades, fazendo com que os prazos sejam cumpridos. É responsável por remover possíveis impedimentos para a equipe e faz com que a metodologia seja cumprida.

- *Developement Team* (Time de Desenvolvimento) – Planeja e prioriza o que será feito junto ao Dono do Produto. Tem autonomia para realizar o desenvolvimento de cada item do *Product Backlog* de acordo com o ciclo de vida definido anteriormente. É suficientemente pequeno, contemplando todas as habilidades necessárias para a entrega do produto.

3.3 Técnicas Complementares

Existem algumas técnicas que, geralmente, são utilizadas juntamente com a metodologia *SCRUM*, são elas:

3.3.1 Metas no *SCRUM*

Quando se organizam as necessidades do usuário, definem-se metas nas quais essas necessidades serão entregues. Essa meta é como um objetivo alcançado e pode ser de três tipos: **meta da *sprint***, **meta da *release*** ou ***roadmap*** ou ainda **visão do produto**. É importante utilizar metas, pois estas servem como marcos mostrando a consistência e a evolução do sistema desenvolvido.

A **meta da *sprint*** é estabelecida com o Dono do Produto e o Time de Desenvolvimento e serve como guia para a produção dos ítens do *backlog*. É, portanto, o caminho para atinigir a meta de *release* ou *roadmap* (SABBAGH, 2014, pg. 162).

Release pode ser entendida como a entrega de um incremento de *software* homologado pronto para utilização pelo usuário e geralmente serve como base para desenvolvimento de novas funcionalidades. A **Meta da *Release*** geralmente existe quando se considera uma *Release* planejada. Possui uma data da *Relase* definida e tem um conjunto de ítens a serem entregues (SABBAGH, 2014, pg. 164).

Roadmap é um plano organizado por funcionalidades a serem atingidas. A **Meta de Roadmap** é um objetivo ou necessidade de negócios de alto nível em direção a entrega do produto e é semelhante a Meta de Release, diferenciando-se apenas no fato da segunda possuir uma data aproximada para entrega, e não é, necessariamente, totalmente alcançada no momento da entrega. (SABBAGH, 2014, pg. 164).

A visão do produto é o objetivo maior que compreende as necessidades de alto nível com contexto claro e simples, e proporciona interesse para o desenvolvimento do produto durante o projeto. Neste nível, todos os envolvidos do projeto podem ser atualizados quanto ao andamento do projeto. É definida antes do desenvolvimento do produto e permanece estática até o término do mesmo.

Em termos de valor atingido, das metas à visão do produto, pode-se verificar na Figura 4 que cada uma possui quantidades diferentes.

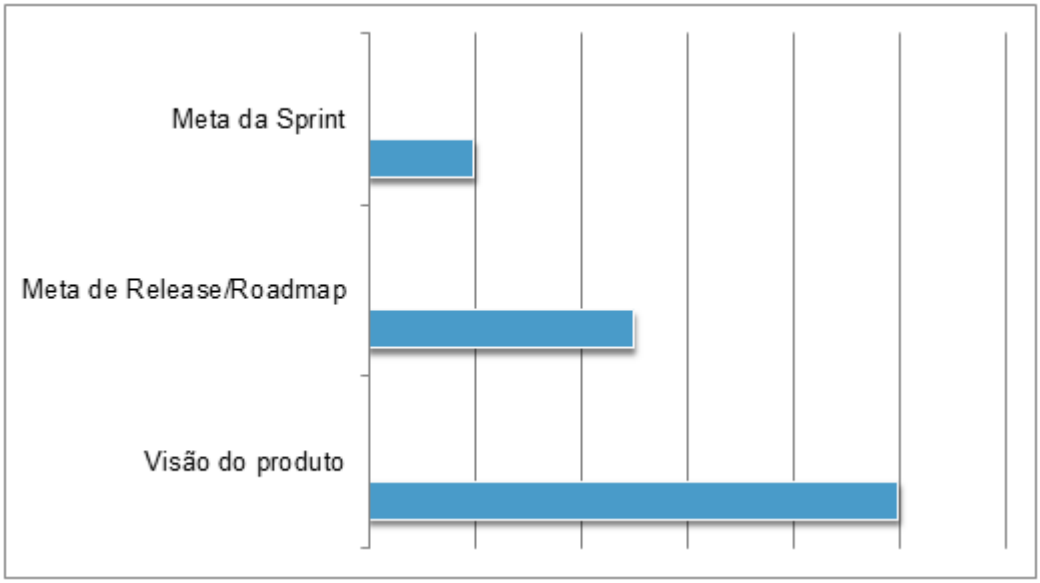


Figura 4: Quantidades de valor atingido das metas à visão do produto

Fonte: arquivo do autor (2016)

3.3.2 Gráficos

Geralmente o time de desenvolvimento nas primeiras *sprints* não conhece a sua produtividade então espera-se que, após algumas *sprints*, já seja possível ter uma noção melhor de quantas funcionalidades podem ser entregues a cada *sprint*. *Story Points* é uma unidade de medida bastante utilizada nas estimativas. O principal a se fazer é usar a quantidade desenvolvida e calcular a média de trabalho entregue nas *sprints*, sobre essa unidade de medida.

Isso possibilitará, independente da unidade de medida utilizada, encontrar a velocidade do time de desenvolvimento. Uma forma bem eficiente de verificar visualmente essa velocidade é através dos gráficos Release *burndown* e *Sprint burndown*.

Gráfico *Sprint* ou *Realease Burndown* - Servem para mostrar o andamento do projeto, a velocidade do time de desenvolvimento, e a previsão de quando este se encerrará de forma gráfica, tendo como entrada informações relativas à *sprint* e às tarefas realizadas.

A diferença entre os dois está na métrica do eixo–x, sendo dias para o *sprint burndown* (pois a unidade de medida geralmente é em dias trabalhados, mas nada impede de utilizar-se o número de semanas, por exemplo) enquanto que para a release *burndown* são *sprints* necessárias para a entrega da release verificados nas Figuras 5 e 6.

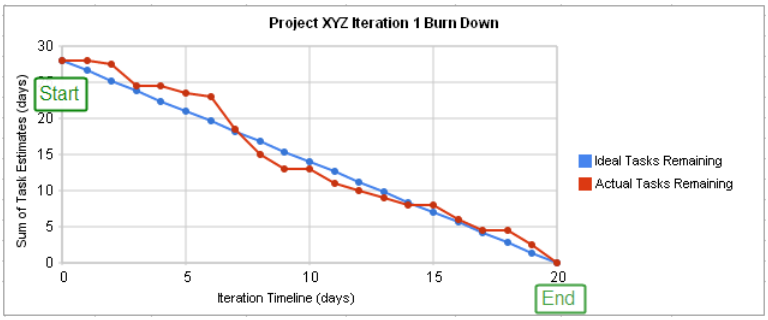


Figura 5: Gráfico *Release Burndown*

Fonte: Disponível em <http://en.wikipedia.org/wiki/Burn_down_chart>. Acesso em 24/11/2014

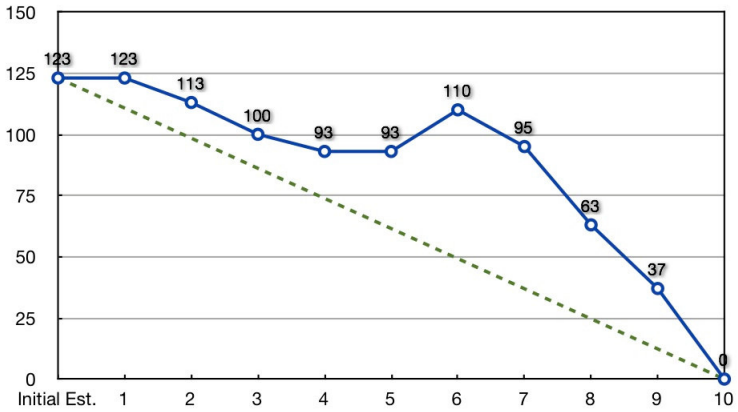


Figura 6: Gráfico *Sprint Burndown*

Fonte: Disponível em <https://lookforwardconsulting.com/2010/10/30/using-a-sprint-burndown-chart/>. Acesso em 11/11/2015

Observando as figuras percebe-se que idealmente a linha seria reta. As suas inclinações mostram algum tipo de anomalia naquele período, acresentando funcionalidades (ocasioando aumento inclinação para cima) ou entregando funcionalidades em menos tempo (ocasionando inclinação para baixo). É claro que reestimativas são esperadas e que mudanças ou problemas eventualmente acontecem, mas o gráfico serve para alertar onde essas situações estranhas ocorrem e, caso seja necessário, tomar alguma atitude.

Os gráficos *burndown* mostram o trabalho que falta no decorrer do tempo. Ou seja, o gráfico começa em cima e cai até tocar o eixo do tempo, indicando que todas as tarefas foram concluídas. Mas existem algumas variações desse gráfico, que são chamadas

de *burnup*, na qual a relação é com o trabalho já realizado no tempo e o trabalho total a ser realizado. Na Figura 7, é possível verificar exemplo do gráfico *burnup*.

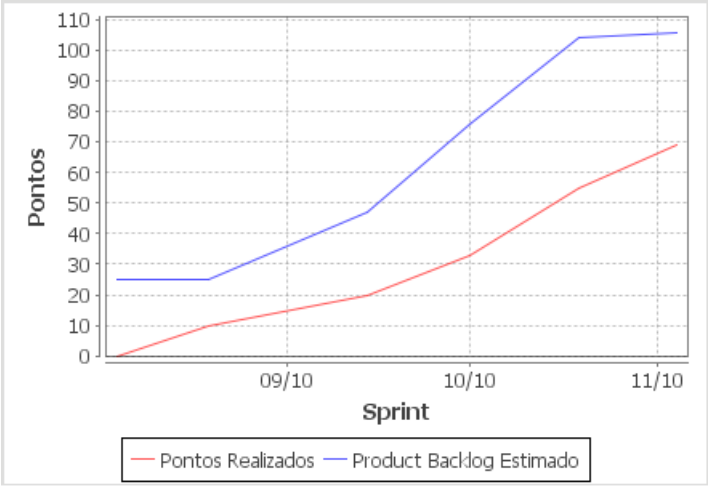


Figura 7: Gráfico Release Burnup

Fonte: Disponível em <http://blog.myscrumhalf.com/2011/07/novidades-no-scrumhalf/>. Acesso em 19/11/2015

3.3.3 *Kanban*

Técnica que utiliza um quadro organizacional para apresentar visualmente o que deve ser feito, o que está sendo feito e o que já foi feito. Exemplo desta técnica é apresentado na Figura 8.

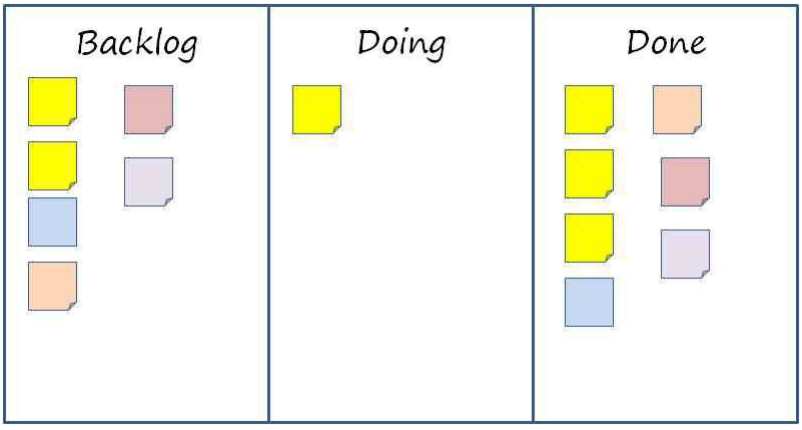


Figura 8: Técnica *KanBan*

Fonte: Disponível em <http://www.guidedinnovation.com/si/2010/08/23/managing-your-constraint/>. Acesso em 24/11/2014

O Kanban é uma técnica para ter uma visibilidade maior em relação à atual situação de cada item desenvolvido. É geralmente composta por três colunas: *Backlog*(o que será

feito), *Doing* (Fazendo) e outra *Done* (Feito). Uma coluna de "testes", "homologação", "especificação" pode ser acrescentada para facilitar a visualização da situação real do item, se o time de desenvolvimento desejar.

É importante lembrar que as técnicas complementares apresentadas não fazem parte do *SCRUM*, mas são eficientes para visualização e acompanhamento da metodologia.

4 A integração entre as disciplinas

A intenção principal na integração dessas disciplinas já apresentadas é de complementar valores de uma com a outra. O ponto que foi encontrado para essa união é o momento de produção do *Product Backlog* na metodologia *SCRUM*.

O planejamento desse importante artefato da metodologia *SCRUM* é utilizado e/ou atualizado praticamente em toda a vida do projeto, pois é ele que define os itens a serem feitos, a quantidade de tempo necessária e possivelmente outras informações como valor e *ROI* (Retorno de Investimento).

São utilizadas estimativas para a realização de cada item do Backlog do Produto, (SABBAGH, 2014, pg. 121) exemplifica 3 (três) estimativas que podem ser utilizadas: tempo real, dias ou horas reais de trabalho; tempo ideal, dias ou horas ideais de trabalho, sem quaisquer interrupções; *story point*, unidade relativa de tempo criada pelo Time de Desenvolvimento, sendo a última a mais utilizada por equipes ágeis, esta última devido a sua importância será melhor descrita a seguir.

(SABBAGH, 2014, pg. 122), define *Story Point* como uma unidade relativa utilizada pelo Time de Desenvolvimento para estimar o tempo necessário para se desenvolver um item do *Backlog do Produto*, no qual ao invés de usar horas ou dias, o próprio Time de Desenvolvimento cria e escolhe uma escala e, a partir de um ou mais pontos de referência selecionados, passa a estimar os itens do *Backlog do Produto* dentro dessa escala, estimando os próximos itens com os pontos de referência dos últimos itens já estimados.

As escalas utilizadas variam como uma sequência de um a dez (escala 1), potências de dois, a famosa sequência de *Fibonacci*(escala 2) e a *t-shirt sizing*(escala 3) por exemplo, como podem ser visualizadas na Tabela 6.

A vantagem dessa abordagem é que o item estimado como unidade menor, serve de referência para os outros itens, no sentido de que um item precisa de duas vezes menos/mais esforço ou tempo do que a unidade menor. Assim, considerando que todos os itens são estimados a partir da referência, ela permanece correta.

A desvantagem dessa abordagem é que ela não mede risco, esforço ou tamanho,

Tabela 6: Exemplos de escalas utilizadas em story points

Escala 1	1,2,3,4,5,6,7,8,9,10..
Escala 2	1,2,3,5,8,13,21,34,55 89..
Escala 3	PP,P,M,G,GG,XG

Fonte : adaptado de (SABBAGH, 2014).

mas apenas uma estimativa de trabalho necessário para realizar o item e também deixa a aferição baseada na experiência e sem uma fundamentação consolidada, baseada apenas no senso comum.

É exatamente aqui que a métrica pontos de função se encaixa. Dado que os itens do nosso *Product Backlog* podem ser estimados por qualquer métrica, não há impedimentos para que a métrica pontos de função possa ser utilizada.

Na Figura 9 apresenta-se um exemplo de *Product Backlog*. A ideia é que cada item, após definida sua prioridade com o Dono do Produto siga as atividades definidas no capítulo 2. Assim ao final de cada ciclo de atividades ter-se-á um valor aproximado em pontos de função, para cada funcionalidade a ser implementada.

PRODUCT BACKLOG

Produto	Software clínica sorriso no nordeste		
Dono do Produto	Nelson Carneiro	Atualizado em	27 de setembro de 2015

ID	História de Usuário / Requisito / Item	Categoria	Tipo	Prioridade	Estimativa
6	Eu quero cadastrar meus pacientes com informações pessoais deles.	Cadastros	Funcionalidade	500	5
3	Como administrador da clínica eu quero ter um controle dos meus orçamentos feitos.	Gerencial	Funcionalidade	460	5
13	Preciso cadastrar as consultas feitas e quais delas geram o inicio de tramento.	Gerencial	Funcionalidade	450	8
3	Preciso de um relatório de orçamentos do mês.	Relatórios	Funcionalidade	440	13

Figura 9: Exemplo de *Product Backlog*

Fonte: Adaptado de <http://www.tiexames.com.br/>

É necessário portanto que os itens do Product Backlog sejam escritos como funcionalidades entregues, caso seja necessário um item pode e deve ser desmembrado em mais itens para que o método seja efetivo.

5 Estudo de caso

O presente capítulo visa demonstrar a integração entre a metodologia *Scrum* e a métrica pontos de função, na prática, em um caso fictício de utilização. Será utilizado como exemplo um sistema de controle de clínica odontológica fictícia.

Começa-se por definir um roadmap do produto a ser desenvolvido. A título de exemplo, apenas quatro objetivos foram estabelecidos, na Figura 10. Mais importante é que a partir desse *roadmap*, pode-se definir melhor os itens que farão parte do *Product Backlog*.

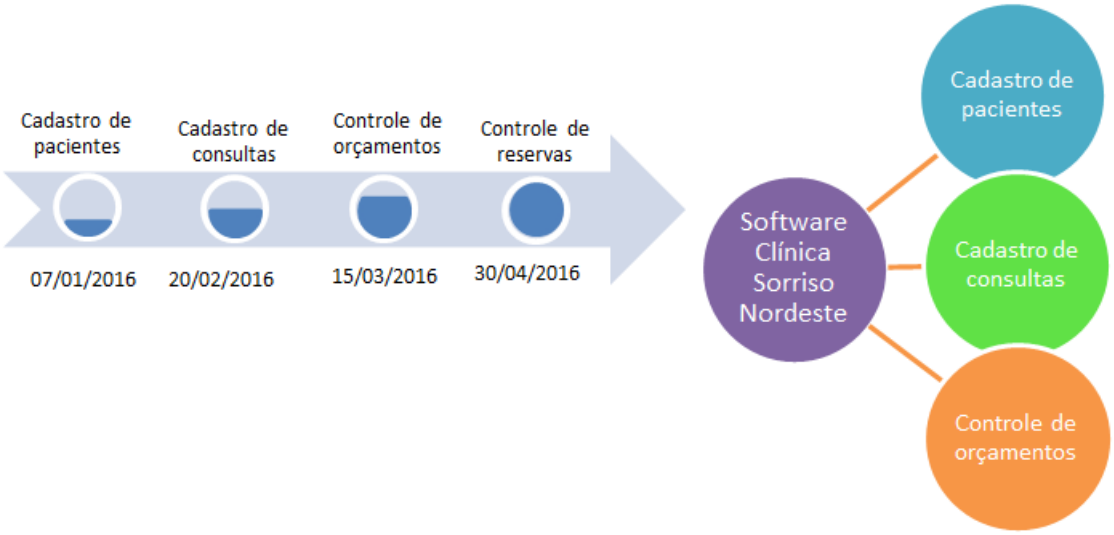


Figura 10: Roadmap do produto

Fonte: arquivo do autor (2016)

Em uma Reunião de Planejamento da *Sprint* (*Sprint Planning Metting*) define-se que somente o primeiro marco do roadmap seria trabalhado na primeira sprint. Um conjunto de itens que farão parte do *Product Backlog* dessa primeira *Sprint* e das seguintes pode ser visto na Tabela 7.

Após definidos quais marcos serão focados primeiro, pode ser interessante subdividir a história de usuário em sub-tarefas. A Tabela 8 apresenta essa subdivisão, ou modularização, da funcionalidade em outras funcionalidades menores, para a primeira *Sprint*. Essa modularização serve para dividir melhor o tempo a ser utilizado, fazendo uma análise mais precisa das tarefas que fazem parte do todo (incremento de software), facilita o acompanhamento e ajuda o desenvolvedor a ter uma melhor visão de progresso.

O *Product Backlog* em si pode não ser suficiente para contemplar todos os requisitos do sistema, se necessário devem-se descrever bem as história de usuário de forma que não

Tabela 7: *Product Backlog* em construção

Código	História de Usuário/Requisito/Item	Estimativa
SPRINT 1		
1	Cadastro público de pacientes	-
SPRINTs SUBSEQUENTES		
2	Controle de consultas	-
3	Controle de orçamentos	-
4	Controle de reservas	-

Fonte: arquivo do autor(2016)

Tabela 8: *Product Backlog* com *Sprint* 1 modularizada

Código	História de Usuário/Requisito/Item	Estimativa
SPRINT 1		
1	Cadastro público de pacientes	-
1.1	Criação da página de inserção/atualização de dados do paciente com informações mínimas necessárias	-
1.2	Possibilitar que o paciente informe dados adicionais para enriquecer seu cadastro	-
1.3	Consultar endereço através da API dos correios	-
1.4	Enviar e-mail de autenticação do usuário	-
SPRINTs SUBSEQUENTES		
2	Controle de consultas	-
3	Controle de orçamentos	-
4	Controle de reservas	-

Fonte: arquivo do autor(2016)

fiquem dúvidas ou acabem sendo feitas desconforme o desejo do usuário final.

Se necessário, alguém bem informado das regras de negócio (geralmente o Dono do Produto) pode auxiliar no detalhamento dessas histórias de usuário. Também podem ser utilizados diagramas de modelagem para fluxos complexos, a UML é uma linguagem que pode ser utilizada para este fim.

Exemplo disso foi o que no cadastro do paciente o Dono do Produto solicitou que ao fazer o cadastro público, o endereço do mesmo pudesse ser capaz de ser consultado através da API de consultas do correio e que um email de autenticação do usuário fosse enviado para seu *email* pessoal.

Para que a equipe *scrum* tivesse total segurança no momento da implementação um diagrama de caso de uso Cadastro Público do Paciente (Apêndice A) foi feito e assim repassado para a equipe, tirando desta forma qualquer dúvida sobre a funcionalidade.

Antes de iniciar o desenvolvimento, realiza-se a estimativa dos itens do *Product Backlog* utilizando como métrica pontos de função.

***SPRINT* 1 - Realizar Estimativas**

Após definidos os requisitos do sistema a ser desenvolvido deve-se iniciar a contagem dos pontos de função que guiará o processo de desenvolvimento do que é menos/mais simples/complexo, barato/custoso de ser feito, mas não esquecendo que o aceite do usuário deve ser prezado.

Seguindo as etapas para o processo de contagem de ponto de função, conforme o Capítulo 2, inicia-se determinando o tipo de contagem a ser feito. Como se tratam de funcionalidades novas, o tipo de contagem será o de projeto de desenvolvimento. Caso já existisse algo pronto e que precisasse ser alterado, a única diferença seria considerar, no projeto de melhoria, as comunicações necessárias, atributos acrescentados, ou seja, o que já está pronto não entraria na contagem.

Estimativa em Pontos de Função:

1. Determina-se o tipo da contagem como projeto de desenvolvimento, pois trata-se de uma funcionalidade totalmente nova.

2. O escopo da contagem, compreende informações sobre o paciente (dados cadastrais), totalizando um sub-sistemas que entrará no escopo. A fronteira é da aplicação de cadastro de informações pessoais. A Figura 11 ilustra os conceitos de fronteira e escopo.

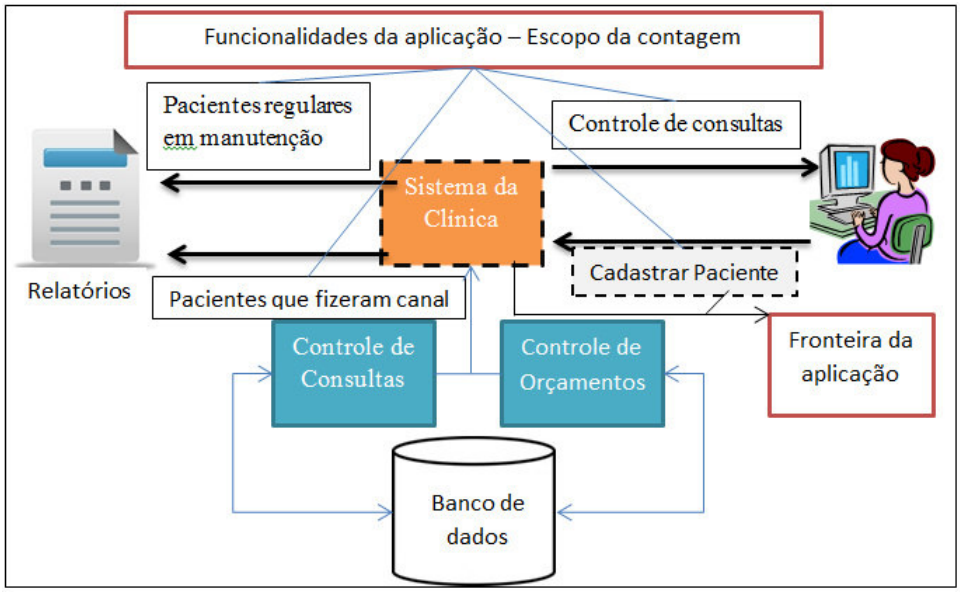


Figura 11: Fronteira e escopo da contagem do sistema

Fonte: arquivo do autor (2016)

3. Tabelas e seus campos, com consultas quais filtros serão necessários e relatórios sobre quais tabelas serão feitas.

Tabela 9: Complexidade das funcionalidades da *SPRINT* 1

Código	Funcionalidades	Tipo	ARs	TDs	Complexidade
1	Cadastro público de pacientes	EE	2	18	Média
2	Confirmação de Email	EE	1	1	Baixa

Fonte: arquivo do autor(2016)

Agora definem-se as entidades, atributos e relacionamentos necessários para as funcionalidades. Pode-se visualizar o diagrama de entidade-relacionamento no nível lógico na Figura 12.

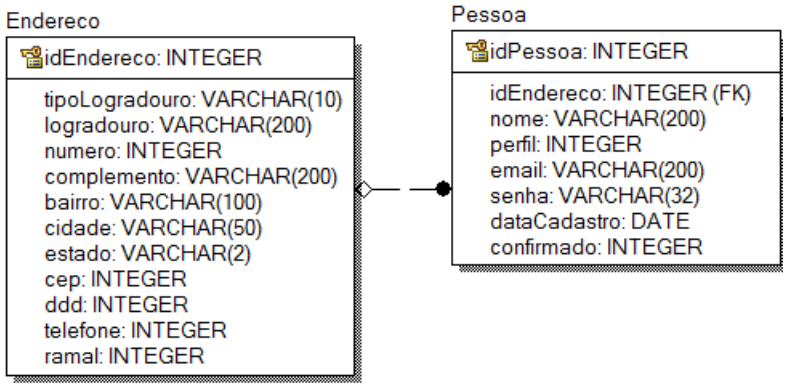


Figura 12: Modelo lógico contendo, atributos e relacionamentos necessários.

Fonte: arquivo do autor (2016)

4. Determinar complexidade funcional de cada funcionalidade.

   Para a funcionalidade 'Cadastro público de pacientes' (Compreende as sub-funcionalidades 1.1, 1.2 e 1.3, na Tabela 9) temos uma EE com 2 ARs (Pessoa, Endereco) , 18 TDs (idPessoa, nome, perfil, email, senha, dataCadastro, idEndereco, tipoLogradouro, logradouro, numero, complemento, bairro, cidade, estado, cep, ddd, telefone, ramal).

   Em especial temos uma funcionalidade de "Confirmação de email" (compreende a sub-funcionalidade 1.4, Tabela 8) tem-se uma EE com 1 AR (Pessoa) e 1 TD (confirmado).

   É importante salientar que o método dispensa uma precisão tão alta na contagem de tipos de dados, mas preza por uma procupaação maior com a quantidade de funcionalidades contadas.

5. Utilizando a tabela descrita no capítulo 2 inserem-se os dados obtidos e faz-se as contas necessárias para estabelecer a complexidade funcional da Sprint, apresentada na Tabela 10.

Tabela 10: Complexidade funcional da *Sprint* 1

Tipo de Função	Complexidade Funcional	Totais de Complexidade	Totais tipo de função
EE	___1____ Baixa		
	___1___ Média	X 4 = _____4_____	
	___0___ Alta	X 6 = _____0_____	
			_____7_____
		Tamanho funcional total da sprint:	_____3+4 = 7_____

6. Documentar e partir para o desenvolvimento Sabendo que a *Sprint* 1 tem um número de 7 pontos de função para serem entregues. Este ciclo é feito no inicio de toda *Sprint* para determinar o número de pontos de função a serem desenvolvidos.

   Definir a produtividade em Pontos de Função da Equipe.

   Conforme o número de sprints aumentar, pode-se determinar uma média de pontos entregue por uma equipe de desenvolvimento.

   A produtividade real da equipe só será conhecida após o inicio do desenvolvimento, mas estipula-se, de acordo com a complexidade, um valor entre 1 e 3 semanas para a realização da *Sprint*. Após algumas *sprints*, a quantidade de tempo necessária vai se tornando um valor mais próximo da realidade.

   Definir a ordem de execução de atividades conforme a necessidade do usuário.

   A ordem de execução das atividades deve obedecer ao desejo do usuário, para que ele possa acompanhar, verificar e validar que o sistema desenvolvido está conforme o que precisa. Caso seja necessária alguma alteração o usuário não precisa esperar até o final do projeto para se manifestar.

   Neste exemplo o cliente optou pelo cadastro da pessoa primeiro, para que pudesse ir realizando cadastros, enquanto as outras funcionalidades eram desenvolvidas.

   Realizar o acompanhamento utilizando o gráfico *burn down chart*.

   O gráfico *burndown* e o quadro *kanban* são boas técnicas de acompanhamento do projeto. A primeira melhor entendida pelo usuário, pois acompanha de fato o desenvolvimento do sistema e quanto falta para terminá-lo, já o segundo melhor entendido pelo time de desenvolvimento pois sabe-se através destes quantas funcionalidades faltam para o fim do projeto. Para o estudo de caso podemos visualizar estes artefatos nas Figuras 13 e 14 referentes a primeira *Sprint*.

7. Reuniões Diárias

   Se necessário, o time de desenvolvimento deve relatar qualquer impedimento ao *Scrum Master* para que este possa, o mais breve possível, resolver a situação para

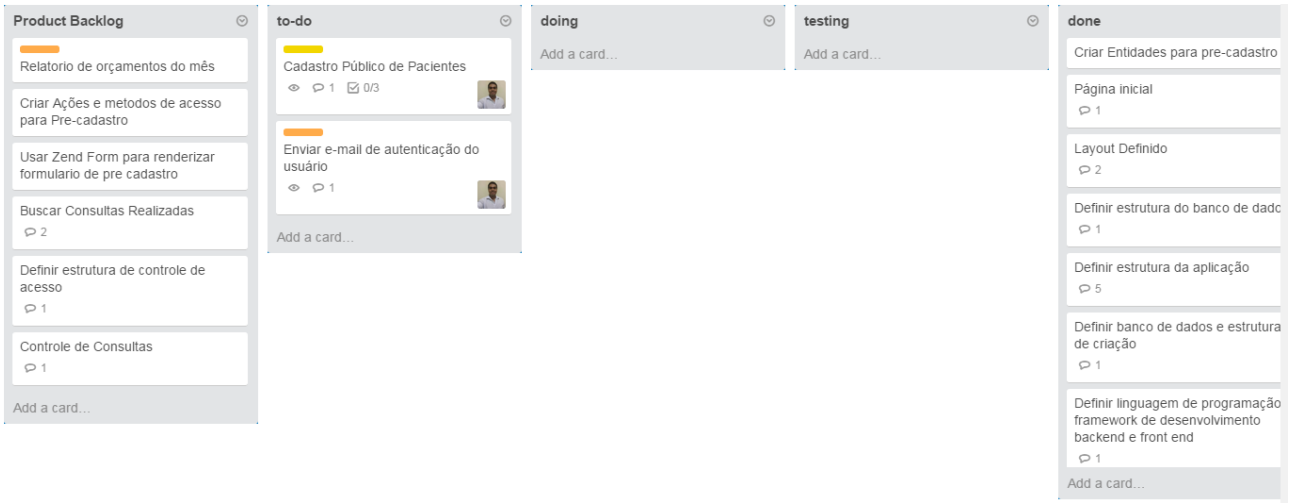
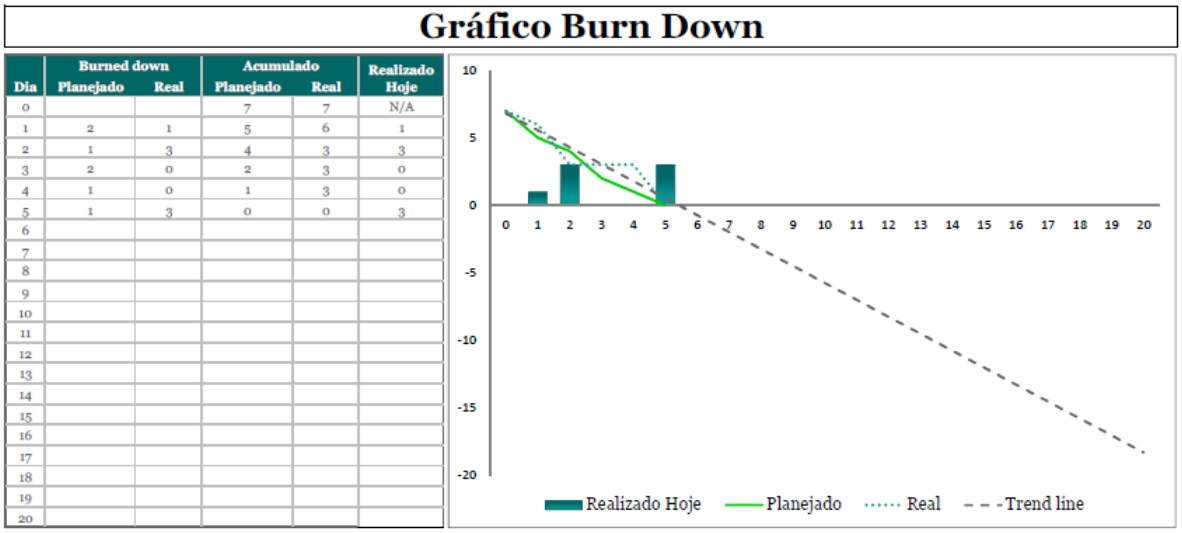


Figura 13: Técnica Kanban implementada pelo Trello para acompanhamento das tarefas do projeto.

Fonte: <http://www.trello.com>



Dia	Burned down Planejado	Burned down Real	Acumulado Planejado	Acumulado Real	Realizado Hoje
0			7	7	N/A
1	2	1	5	6	1
2	1	3	4	3	3
3	2	0	2	3	0
4	1	0	1	3	0
5	1	3	0	0	3
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					

Figura 14: Grafico *Burndown* da primeira *Sprint*

Fonte: Adaptado de <http://www.tiexames.com.br/>

que o desenvolvimento da *Sprint* não seja interrompido. É importante também para verificar o andamento das *sprints* e verificar o que já foi feito até o presente momento.

8. Terminada a Sprint

   Após o Time de Desenvolvimento entregar o resultado proposto pela *Sprint*, será necessário validar junto ao Dono do Produto se a mesma é aceita. Também é necessário aprender com os erros, e na *Sprint Review* todos os erros, impedimentos ocorridos durante a sprint são levantados, para que estes nao reincidam em próximas sprints.

9. Entregar Incremento de Software

   O incremento de *software*, devidamente validado junto ao pessoal interessado, é incorporado ao sistema e lançado para utilização efetiva dos clientes.

10. Hora de recomeçar

    Entregue o incremento de software, é possível visualizar no Roadmap e com o próximo item inicia-se novamente o processo descrito anteriormente até que todos os itens sejam devidamente entregues ao cliente.

SPRINT 2 - REALIZAR ESTIMATIVAS

O tipo da contagem continua como projeto de desenvolvimento, pois ainda trata-se de funcionalidades totalmente novas.

O escopo da contagem muda para consulta de dados cadastrais e organização lógica obedecendo a regra de negócio da história de usuário "Controle de consultas" (Apêndice B).

A velocidade de entrega obtida na primeira Sprint indica que a produtividade da equipe teria capacidade para receber mais de 1 (uma) funcionalidade, portanto a história de usuário "Controle de orçamentos" (Apêndice C) também poderá entrar na Sprint, totalizando dois sub-sistemas que entrarão no escopo.

A decisão de adicionar mais uma funcionalidade na *sprint* será feita ao final da contagem, em reunião com a equipe e o Dono do Produto. As fronteiras são da aplicação de cadastro de consulta e da aplicação de consultas do paciente, a Figura 15 ilustra os conceitos de fronteira e escopo.

Após a Reunião de Planejamento da *Sprint* obteve-se como resultado a Tabela 11 já modularizada.

Agora definem-se as entidades, atributos e relacionamentos necessários para as funcionalidades. A Figura 16 apresenta o diagrama de entidade-relacionamento no nível lógico.

Determina-se assim complexidade funcional de cada funcionalidade. Para a funcionalidade 'controle de orçamentos' (Código 2, na Tabela 11) :

- uma CE: Consulta de orçamentos que possui 3 ARs (Orcamento, OrcamentoDetalhes, Servicos, Pessoa, Endereco) e 15 TDs (idOrcamento, idMedico, nomeMedico, idPessoa, nomePessoa, observacaoOrcamento, queixaPrincipal, dataPrevista, valorTotal, formaPagamento, dataValidade, idServico, valorServico, descricaoServico).
- três EE:

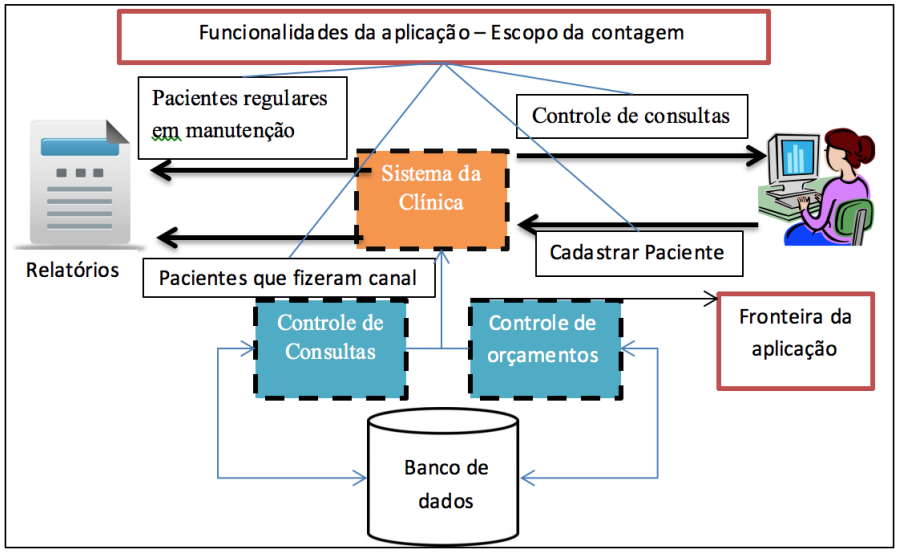


Figura 15: Fronteira e escopo da contagem do sistema *Sprint* 2

Fonte: arquivo do autor (2016)

Tabela 11: Sprint 2 modularizada

Código	História de Usuário/Requisito/Item	Estimativa
SPRINT 1		
1	Cadastro público de pacientes	7
SPRINT 2		
2	Controle de orçamentos	-
2.1	Consulta de orçamentos	-
2.2	Criação de orçamento	-
2.3	Atualização de orçamento	-
2.4	Remoção de orçamento	-
3	Controle de consultas	-
3.1	Consulta de consultas feitas	-
3.2	Criação de consulta	-
3.3	Atualização de consultas	-
3.4	Remoção de consultas	-
SPRINTs SUBSEQUENTES		
4	Controle de reservas	-

Fonte: arquivo do autor(2016)

– Criação de orçamento: possui 4 ARs (Pessoa, Orcamento, OrcamentoDetalhes, Servicos) e 13 TDs (idOrcamento, idMedico, dataOrcamento, idPessoa, observação, valorTotal, queixaPrincipal, dataPrevista, formaPagamento, dataValidade, idServico, descricaoServico, valoServico).

– Atualização de orçamento: possui 4 ARs (Pessoa, Orcamento, OrcamentoDetalhes, Servicos) e 11 TDs (idMedico, dataOrcamento, observação, valorTotal, queixaPrincipal, dataPrevista, formaPagamento, dataValidade, idServico, des-

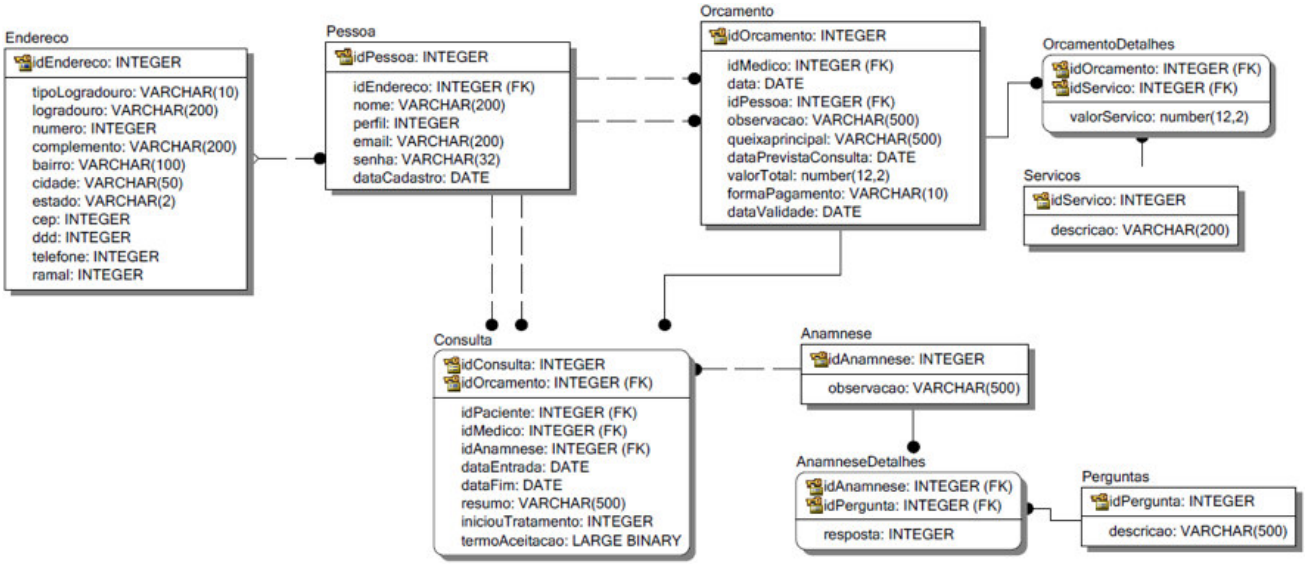


Figura 16: Modelo logico contendo, atributos e relacionamentos necessários.

Fonte: arquivo do autor (2016)

cricaoServico, valorServico).

- Remoção de orçamento: possui 2 ARs (Orcamento, OrcamentoDetalhes) e 1 TDs(idOrcamento)

Para a funcionalidade 'controle de consultas' (Codigo 3, na Tabela 11) verifica-se as seguintes funções de transação :

- uma C.E.:
  - Consulta de consultas feitas: composta por 5 ARs (Consulta, Orcamento, OrcamentoDetalhes, Servicos, Pessoa), 11 TDs (idConsulta, idOrcamento, idMedico, nomeMedico, idPaciente, nomePaciente, dataEntrada, dataValidade, valorTotalOrcamento, idServico, descricaoServico)
- três E.E.:
  - Criação de consulta: compreende 3 ARs (Orcamento, Consulta, Anamnese, AnamneseDetalhes, Perguntas) e 11 TDs, (idConsulta, idOrcamento, idPaciente, idMedico, termoAceitacao, resumo, dataEntrada, idAnamnese, observacaoAnamnese, idPergunta, resposta).
  - Atualização de consulta: possui 5 ARs (Orcamento, Consulta, Anamnese, AnamneseDetalhes, Peguntas) e 8 TDs (resumo, iniciouTratamento, termoAceitacao, idAnamnese, observacaoAnamnese, idPergunta, resposta, descricaoPergunta).
  - Remoção de consulta: compreende 1 AR (Consulta) e 1 TD (idConsulta).

Tabela 12: Complexidade das funcionalidades da *SPRINT* 2

Codigo	Funcionalidade	Tipo	ARs	TDs	Complexidade
2.1	Consulta de orçamentos	CE	3	15	Média
2.2	Criação de orçamentos	EE	4	13	Alta
2.3	Atualização de orçamento	EE	4	11	Alta
2.4	Remoção de orçamento	EE	2	1	Baixa
3.1	Consulta de consultas feitas	CE	5	11	Alta
3.2	Criação de consultas	EE	3	11	Alta
3.3	Atualização de consultas	EE	5	5	Alta
3.4	Remoção de consultas	EE	1	1	Baixa

Fonte: arquivo do autor(2016)

Tabela 13: Complexidade funcional da segunda *Sprint*

Tipo de Função	Complexidade Funcional	Totais de complexidade	Totais tipo de função
EE	___2___ Baixa	X 3 = ____6_____	
	___0___ Média	X 4 = ____0_____	
	___4___ Alta	X 6 = ____24____	
			_____30_____
CE	___0___ Baixa	X 3 = ____0_____	
	___1___ Média	X 4 = ____4_____	
	___1___ Alta	X 6 = ____6_____	
			_____10_____
		Tamanho funcional total da sprint:	_(10+30) =40_

Fonte: arquivo do autor(2016)

Conforme a Tabela 4, apresentada no capítulo 2, verifica-se a complexidade para as funcionalidades a serem desenvolvidas preenchidas na Tabela 12.

Utilizando a Tabela 5, descrita no capitulo 2, inserem-se os dados obtidos pela contagem dos Pontos de Função e as operações matemáticas necessárias são feitas para estabelecer a complexidade funcional da segunda *Sprint*, como mostra a Tabela 13.

Segue-se com a documentação e parte-se para o desenvolvimento. Nesta segunda *Sprint*, tem-se um número de 40 pontos de função para serem entregues. Comparada com a primeira *Sprint* de 7 pontos de função entregues em 1 semana, houve um aumento de 571% no número de pontos de função entregues por semana.

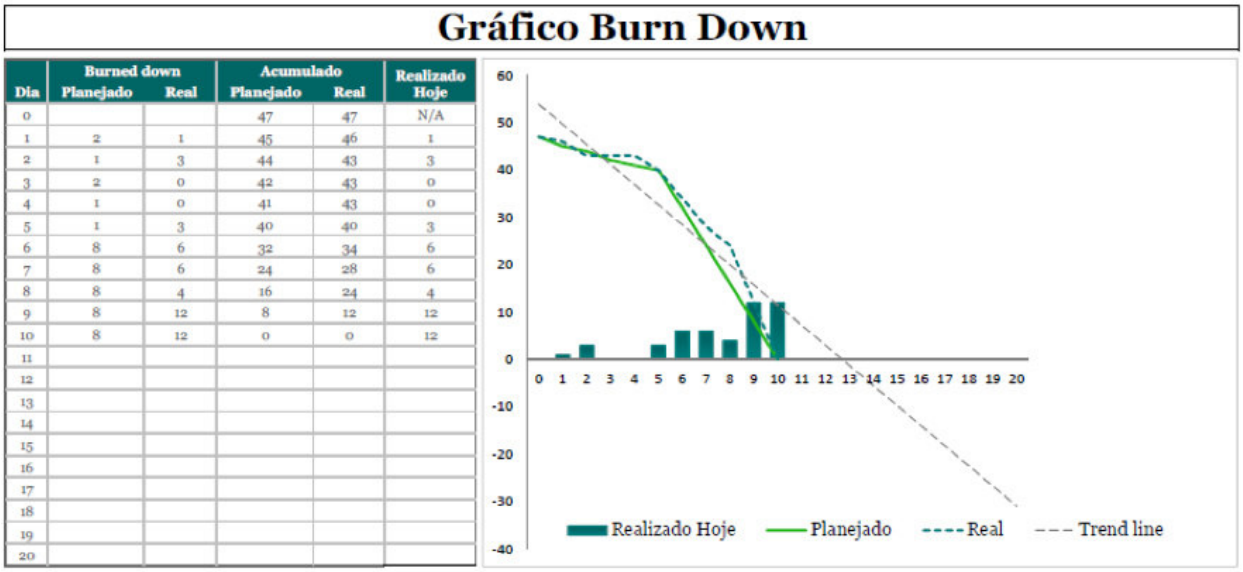
Percebe-se um aumento da produtividade da equipe. Na primeira *Sprint* foram entregues 7 pontos de função em 1 semana, o que resultaria uma média de 1,2 pontos de função por semana. Já na segunda *Sprint*, obteve-se um número bem maior de 40 pontos de função entregues em 1 semana, totalizando uma média de 8 pontos de função entregues.

Apartir desse estágio no qual já consegue-se uma boa comparação, em *sprints* futuras já é possível estimar bem mais facilmente a quantidade de tempo necessário para entregar uma funcionalidade baseada no número de Pontos de Função definidos para esta.

O gestor responsável pela equipe pode fazer um cálculo mais preciso e assim apresentar um prazo bem próximo do real, ainda que impreciso.

O Dono do Produto define que por vir primeiro no fluxo de suas atividades, o controle de orçamentos seja desenvolvido antes do controle de consultas.

As figuras 17 e 18 apresentam o gráfico *burndown* e o diagrama *kanban* para a segunda *Sprint*.



Dia	Burned down Planejado	Burned down Real	Acumulado Planejado	Acumulado Real	Realizado Hoje
0			47	47	N/A
1	2	1	45	46	1
2	1	3	44	43	3
3	2	0	42	43	0
4	1	0	41	43	0
5	1	3	40	40	3
6	8	6	32	34	6
7	8	6	24	28	6
8	8	4	16	24	4
9	8	12	8	12	12
10	8	12	0	0	12
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					

Figura 17: Grafico *Burndown* da *Sprint* 2.

Fonte: Adaptado de <http://www.tiexames.com.br/>

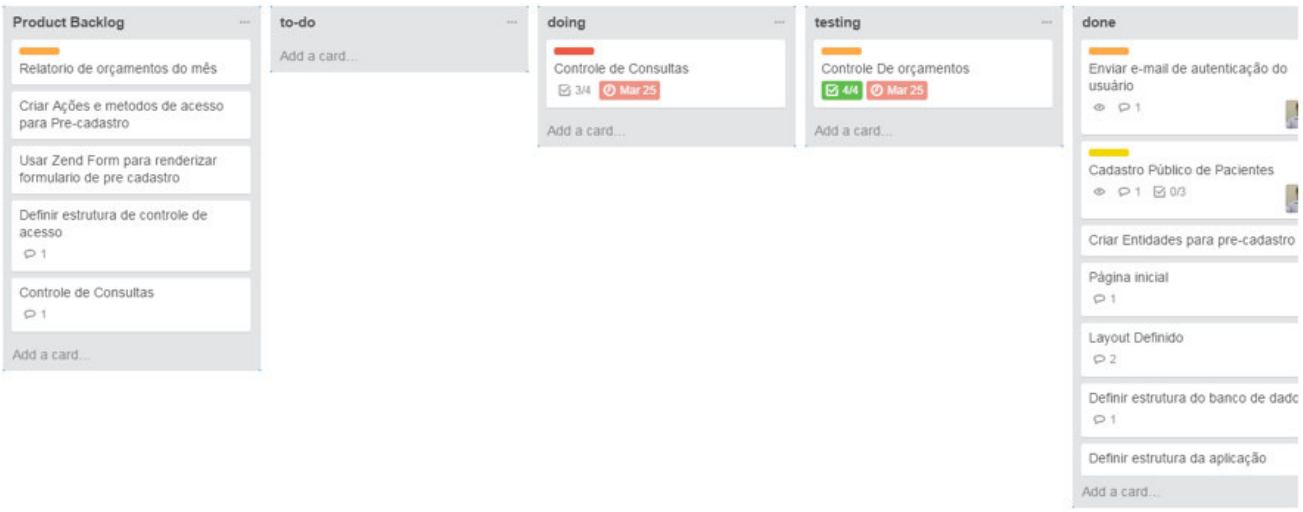


Figura 18: Técnica *Kanban* implementada pelo *Trello* para acompanhamento das tarefas do projeto.

Fonte: Adaptado de <http://www.trello.com/>

Em suas reuniões diárias, a equipe pode perceber um aumento da sua produtividade, pois não se preocupava com complexidade do sistema, mas com o término da sua atividade. É necessária uma interação entre a equipe para alinhar os conhecimentos relativos às funcionalidades, assim que todos os envolvidos conhecessem bem e soubessem como utilizar exatamente as funcionalidades a serem entregues, a etapa de desenvolvimento se torna menos complexa.

Ao entregar uma tarefa a equipe não se preocupava em apenas cumprir o cronograma (que ainda assim é importante), mas tinha a tranquilidade e o discernimento de fazer o possível para entregar as funcionalidades no tempo previsto sem que isso prejudicasse as partes envolvidas.

O incremento de software é apresentado ao Dono do Produto e este aprova as funcionalidades, só assim é incorporado ao sistema como parte final do produto.

Entregue o incremento de software pode-se visualizar o *Roadmap* e com outro item inicia-se novamente o processo descrito anteriormente até que todos os itens sejam devidamente entregues ao cliente.

Conclusão

O *SCRUM* acolhe a incerteza e a criatividade, permitindo que as equipes avaliem o que já criaram e, o mais importante, como o criaram. A estrutura do *SCRUM* busca aproveitar a maneira como as equipes realmente trabalham, dando a elas as ferramentas para se auto organizar e aprimorar rapidamente a velocidade e a qualidade do seu trabalho, afirma (SUTHERLAND, 2015, pg. 17).

Carlos Vazquez em (VAZQUEZ; SIMÕES; ALBERT, 2013) esclarece que a Análise de Pontos de Função (APF) é uma técnica de medição das funcionalidades fornecidas por um *software*, do ponto de vista do seu usuário. Ponto de função é entendido como a unidade medida desta técnica que tem por objetivo tornar a medição independente da tecnologia utilizada para a construção do software. Ou seja, a APF busca medir o que o software faz e não como ele foi construído.

A metodologia *SCRUM* vista com bons olhos na atualidade por resolver um problema crítico nas demandas por software que é a rapidez na entrega assim como a métrica de Pontos de Função auxilia numa eficiente, bastante consistente medição e ainda por cima flexível à mudança do escopo do sistema. O trabalho pôde mostrar duas técnicas que geralmente são utilizadas de formas isoladas, mas que integradas possibilitam melhorar ainda mais o desenvolvimento de *software*.

O Time de Desenvolvimento deve possuir o conhecimento necessário para o efetivo término do produto. O alinhamento e a separação de tarefas por habilidades foi importante para o sucesso do desenvolvimento. O *Scrum Master*, sempre atento às atividades da equipe, e sempre marcando reuniões com o Dono do Produto quando eram necessárias. A aceitação por parte do Dono do Produto envolveu reuniões de no máximo 1 hora, ocasiões nas quais todas as suas sugestões e críticas eram anotadas de forma a serem repassadas para o Time de Desenvolvimento.

Resultados Obtidos

O estudo de caso mostrou que a integração entre a metodologia /textitSCRUM e a métrica Pontos de Função é viável e proporciona uma medição bem razoável e bem fundamentada quando comparada com a medição feita através do conhecimento empírico utilizado pelas histórias de usuário.

As desvantagens encontradas durante o projeto, eram relacionadas à necessidade de um conhecimento prévio das disciplinas, e caso o processo de negócio da funcionalidade a ser entregue fosse complexo, foi necessária a utilização de documentação, o que desacelerou

a produção.

Trabalhos Futuros

Como sugestões de trabalhos futuros, podemos enumerar:

- Sistema de gerenciamento do projeto com os conceitos apresentados

  Nenhuma ferramenta foi encontrada que abrangesse os conceitos das duas disciplinas, futuros trabalhos que pudessem ter a técnica apresentada neste trabalho gerenciada por um sistema que implementasse os conceitos tanto da metodologia (SCRUM), quanto da métrica (Pontos de Função).

- Controle de entregas de incremento de software (Manutenção)

  Percebeu-se também no decorrer das entregas, a necessidade de refazer o processo de publicação/incorporação da funcionalidade ao sistema principal. O controle de entregas é extremamente importante e poderia ser aprofundado no sentido de ter uma vistoria maior e técnicas de contingência para entregas com erros que só fossem percebidos após serem utilizados pelo usuário final.

- Desenvolvimento orientado a testes

  Inserir uma técnica de desenvolvimento orientado a testes, visto que essa prática pode ajudar ainda mais na rapidez da entrega do incremento de software.

- Agilizar a documentação

  A própria tarefa de documentação é extensa e dependendo da complexidade pode levar bastante tempo, seria interessante uma forma de agilizar esse processo e assim aumentar ainda mais a produção.

Referências

BECK, K.; BEEDLE, M. C. et al. Manifesto para desenvolvimento ágil de software. 2001. *Disponivel em*, 2013. Citado na página 22.

CAMPOS, C. J. E. *MS Windows NT Kernel Description*. 2010. Disponível em: <http://carloscamposinfo.com/cjec/?p=205>. Citado 2 vezes nas páginas 18 e 19.

FERREIRA, R. C. da C.; HAZAN, C. Uma aplicação da análise de pontos de função no planejamento e auditoria de custos de projetos de desenvolvimento de sistemas. 2008. Citado 2 vezes nas páginas 14 e 16.

IFPUG, C. Counting practices manual, release 4.3.1 1. *IFPUG–International Function Point Users Group*, 2010. Citado 2 vezes nas páginas 17 e 20.

KOSCIANSKI, A.; SOARES, M. dos S. *Qualidade de software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software*. 2st. ed. São Paulo: Novatec Editora, 2007. ISBN 9788575221129. Citado na página 22.

PRESSMAN, R. S. *Engenharia de software*. [S.l.]: AMGH Editora, 2011. Citado 2 vezes nas páginas 13 e 22.

RUBIN, K. S. *Essential Scrum: A practical guide to the most popular Agile process* [S.l.]: Addison-Wesley, 2012. Citado na página 23.

SABBAGH, R. *Scrum: Gestão ágil para projetos de sucesso*. [S.l.]: Editora Casa do Código, 2014. Citado 2 vezes nas páginas 24 e 29.

SUTHERLAND, J. *Scrum-a arte de faze o dobro de trabalho na metade do tempo*. [S.l.]: Leya, 2015. Citado na página 43.

SUTHERLAND, J.; SCHWABER, K. The scrum papers: Nut, bolts, and origins of an agile framework (2011). *SCRUM Training Institute*, 2014. Citado na página 23.

VAZQUEZ, C. E.; SIMÕES, G. S.; ALBERT, R. M. Análise de pontos de função–medição, estimativas e gerenciamento de projetos de software. *Editora Érica, São Paulo*, v. 13, 2013. Citado 3 vezes nas páginas 16, 20 e 43.

A CASO DE USO: Cadastro público do paciente

A Figura 19 mostra o caso de uso Cadastro Publico do Paciente utilizando a notação em UML.

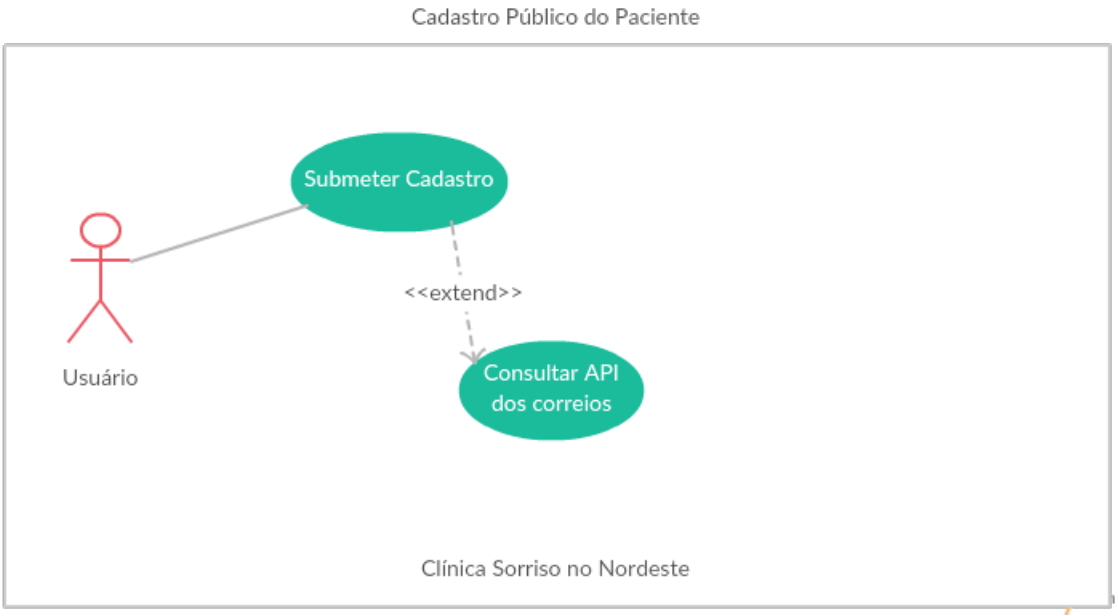


Figura 19: Caso de uso Cadastro Público do Paciente

Fonte: Fonte: arquivo do autor (2016)

Detalhamento do caso de uso:

- Atores: Usuário
- Descrição: Ao visualizar a página inicial do website o usuário poderá submeter um formulário com informações pessoais de forma a ter acesso a novas funcionalidades do sistema.
- Pré-condição: O visitante deve redirecionar para a página principal do site da clínica, ao visualizar o menu, selecionar o botão de “Cadastrar” e preencher corretamente os campos do formulário ao qual é redirecionado.
- Pós-condição: O usuário recebe uma mensagem de confirmação, informando que os dados foram devidamente enviados e cadastrados no banco de dados do sistema e um e-mail é enviado ao e-mail cadastrado pelo usuário para finalizar confirmação o cadastro.

- Cenários

    1. Submete cadastro - Principal
        (a) O usuário seleciona o botão de cadastro do sistema;
        (b) O usuário preenche as informações necessárias;
        (c) O usuário envia o formulário;
        (d) Ao preencher o endereço, após informar o campo CEP, os campos relativos são autocompletados pelo retorno da API de consulta dos correios.
        (e) Uma mensagem de cadastro com sucesso é enviada.
        (f) O sistema envia um e-mail para confirmação do cadastro do usuário.
    2. O usuário esquece algum campo – Exceção 1
        (a) No passo b do cenário 1, se o usuário esquecer algum campo obrigatório, como nome, curso, e-mail, telefone, endereço;
        (b) É exibida uma mensagem de atenção para que ele preencha os dados corretamente.
    3. O usuário informa o CEP, mas não recebe o preenchimento automático – Exceção 2
        (a) Caso o CEP informado não esteja cadastrado, ou seja informado corretamente a API não conseguirá autocompletar os campos relativos ao endereço.
    4. O usuário não recebe retorno do sistema – Exceção 3
        (a) Se por algum motivo o passo do cenário 1 não ocorrer, algum problema do sistema gerenciador de banco de dados não conseguir incluir as informações, enviar uma mensagem de erro de sistema, e contatar os administradores do sistema.

B CASO DE USO: Controle de consultas

A Figura 20 mostra o caso de uso Controle de consultas utilizando a notação em UML.

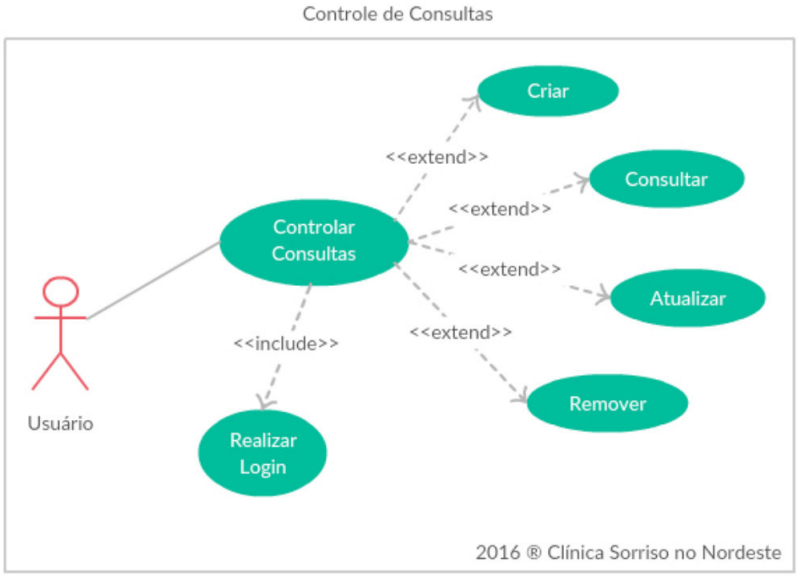


Figura 20: Caso de uso controle de consultas

Fonte: arquivo do autor(2016)

Detalhamento do caso de uso

- Ator: Usuário
- Descrição: O usuário deverá redirecionar para a tela de login do sistema através do botão “Login” para ter acesso ao controle de consultas.
- Pré-condição: O visitante deve redirecionar para a página principal do site da clínica e ao visualizar o menu, selecionar o botão de “Login” informar os dados de usuário e senha para entrar no sistema.
- Pós-condição: O usuário é redirecionado para a tela inicial do sistema, e através do menu consegue acesso a funcionalidade de controle de consultas.
- Cenários

  1. Criar Consulta - Principal
     (a) O usuário clica no botão “cadastrar nova consulta” e é redirecionado para outra tela para o preenchimento do formulário com informações relativas a consulta do paciente.

(b) O usuário confirma os dados e clica no botão “salvar”.

(c) O sistema envia uma mensagem de sucesso em que a consulta foi criada

2. O usuário esquece de preencher campos obrigatorios– Exceção 1

   (a) No passo a) do cenário 1 o sistema informará quais campos obrigatórios não foram preenchidos, como o orçamento vinculado, a anamnésia ou algum dado cadastral do paciente.

3. Consultar consultas do paciente – Principal

   (a) O usuário preenche campos de filtro e clica no botão “Consultar”.

   (b) O sistema processa a requisição e retorna um conjunto de registros a serem verificados pelo usuário.

4. Atualizar consulta – Principal

   (a) O usuário realiza o cenário 3

   (b) O usuário escolhe um dos registros, clicando no ícone relacionado a funcionalidade de atualização.

   (c) O usuário é redirecionado para a tela de atualização, aonde os campos permitidos para atualização, com os dados originais, permitem a edição.

   (d) O usuário confirma a edição

   (e) O sistema emite uma mensagem de confirmação dos dados atualizados.

5. Remover consulta – Principal

   (a) O usuário realiza o cenário 3

   (b) O usuário escolhe um dos registros, clicando no ícone relacionado a funcionalidade de remoção.

   (c) O sistema emite uma mensagem de confirmação para remoção

   (d) O usuário confirma a remoção

   (e) O sistema emite uma mensagem de confirmação dos dados removidos.

C CASO DE USO: Controle de orçamentos

A figura 21 mostra o caso de uso Controle de orçamentos utilizando a notação em UML.

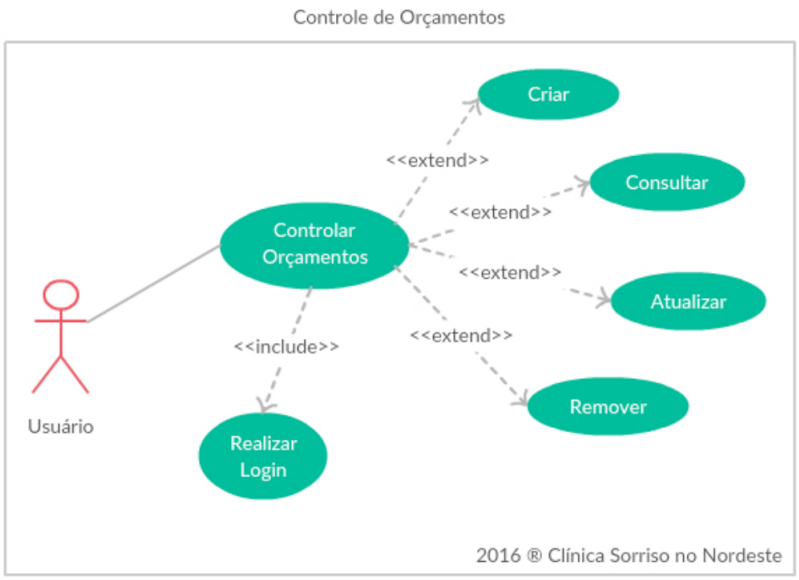


Figura 21: Caso de uso Controle de Orçamentos

Fonte: arquivo do autor(2016)

Detalhamento do caso de uso

- Ator: Usuário.
- Pré-condição: O visitante deve redirecionar para a página principal do site da clínica e ao visualizar o menu, selecionar o botão de "Login" informar os dados de usuário e senha para entrar no sistema.
- Pós-condição: O usuário é redirecionado para a tela inicial do sistema, e através do menu consegue acesso a funcionalidade de controle de orçamentos.
- Cenários

1. Criar orçamento - Principal
   (a) O usuário clica no botão "cadastrar novo orçamento" e é redirecionado para outra tela para o preenchimento do formulário com informações relativas ao orçamento do tratamento ao paciente.
   (b) O usuário confirma os dados e clica no botão "salvar".

   (c) O sistema envia uma mensagem de sucesso em que o orçamento foi criado com sucesso.

2. O usuário esquece de preencher campos obrigatórios– Exceção 1

   (a) No passo a) do cenário 1 o sistema informará quais campos obrigatórios não foram preenchidos, como o médico vinculado, o paciente a receber o tratamento, algum serviço faltante, valores ou algum dado cadastral do paciente.

3. Consultar orçamentos – Principal

   (a) O usuário preenche campos de filtro e clica no botão “Consultar”.

   (b) O sistema processa a requisição e retorna um conjunto de registros a serem verificados pelo usuário.

4. Atualizar orçamentos – Principal

   (a) O usuário realiza o cenário 3

   (b) O usuário escolhe um dos registros, clicando no ícone relacionado a funcionalidade de atualização.

   (c) O usuário é redirecionado para a tela de atualização, aonde os campos permitidos para atualização, com os dados originais, permitem a edição.

   (d) O usuário confirma a edição

   (e) O sistema emite uma mensagem de confirmação dos dados atualizados.

5. Remover orçamento – Principal

   (a) O usuário realiza o cenário 3

   (b) O usuário escolhe um dos registros, clicando no ícone relacionado a funcionalidade de remoção.

   (c) O sistema emite uma mensagem de confirmação para remoção.

   (d) O usuário confirma a remoção.

   (e) O sistema emite uma mensagem de confirmação dos dados removidos.

D Modelagem do Processo de Negócio da Empresa

No sentido de auxiliar no efetivo entendimento do funcionamento do fluxo principal da empresa, o processo de negócio principal, foi modelado utilizando a notação BPMN (*Business Process Modeling*). As atividades indicadas no modelo servem para facilitar a compreensão dos processos internos da empresa. A Figura 22 apresenta a modelagem do fluxo principal desde o momento em que o paciente tem a intenção de fazer o atendimento até o final da consulta.

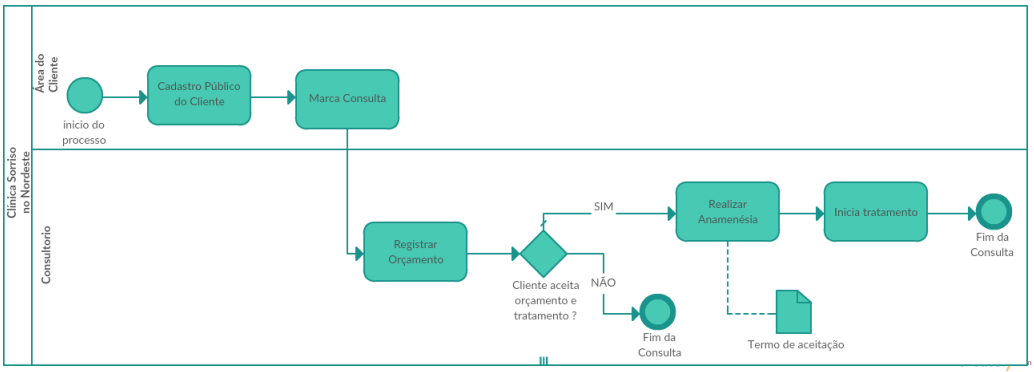


Figura 22: Modelagem do processo de negócio da empresa

Fonte: arquivo do autor(2016)