UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ANDRÉ LUAN CHIQUETTO NASCIMENTO

MINERAÇÃO DE DADOS APLICADA AO CONTROLE DE PRAZOS E PRIORIDADES EM PROJETOS DE SOFTWARE

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2017

ANDRÉ LUAN CHIQUETTO NASCIMENTO

MINERAÇÃO DE DADOS APLICADA AO CONTROLE DE PRAZOS E PRIORIDADES EM PROJETOS DE SOFTWARE

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, do Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. André Pinz Borges

PONTA GROSSA

2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Ponta Grossa

Diretoria de Pesquisa e Pós-Graduação
Departamento Acadêmico de Informática
Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

MINERAÇÃO DE DADOS APLICADA AO CONTROLE DE PRAZOS E PRIORIDADES DE PROJETOS DE SOFTWARE

por

ANDRÉ LUAN CHIQUETTO NASCIMENTO

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 23 de fevereiro de 2017 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. André Pinz Borges
Prof. Orientador

_____
Prof. Me. Luiz Rafael Schmitke
Membro titular

_____
Prof. Me. Vinícius Camargo Andrade
Membro titular

_____
Prof. Dr. Ionildo José Sanches
Responsável pelo Trabalho de Conclusão de Curso

_____
Profª. Dra. Mauren Louise Sguario
Coordenadora do curso

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

RESUMO

NASCIMENTO, André Luan Chiquetto. **Mineração de Dados Aplicada ao Controle de Prazos e Prioridades em Projetos de Software**. 2017. 66 f. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) – Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2017.

Atualmente, os produtos de software são produzidos em nível industrial e com complexidade cada vez maior, de forma que sua produção requer um número considerável de pessoas trabalhando simultaneamente e seguindo os mesmos processos. Entre os principais problemas encontrados nestes cenários estão a atribuição adequada de prioridades, prazos e como definir a importância dos requisitos, bem como estimar o prazo máximo para sua entrega de forma que se produza com eficiência e segurança. Apesar de atualmente existirem sistemas de gestão de projetos, estes deixam unicamente a cargo do gestor do projeto as ações a serem tomadas para que o projeto seja concluído com sucesso. As funcionalidades ofertadas nesses sistemas não auxiliam o gestor em suas decisões mais estratégicas, como a negociação de um prazo maior para entrega de um requisito que não irá ser concluído a tempo. Por outro lado, existem algoritmos e técnicas computacionais capazes de auxiliar a tomada de decisão nos mais diferentes cenários. Pressupõe-se que o uso de Mineração de Dados, combinada com as metodologias de gestão de cronogramas e riscos, possa oferecer aos gestores sugestões e/ou alternativas para problemas encontrados no processo de desenvolvimento de um software. Nesta linha, este projeto teve por objetivo aplicar algoritmos de Mineração de dados para a Descoberta de Conhecimento em Bases de Dados, especificamente algoritmos de Classificação por Árvores de Decisão, no desenvolvimento de um sistema de gestão de projetos, que irá recomendar ações tratativas na gestão de prazos e prioridades de projetos. Resultados mostram que o sistema auxilia os gestores na designação de prioridades e prazos, por meio de registros de comportamento do próprio gestor em situações críticas, a fim de obterem maior sucesso em seus projetos.

Palavras-chave: Gestão de Projetos. Mineração de Dados. Descoberta de Conhecimento em Bases de Dados

ABSTRACT

NASCIMENTO, André Luan Chiquetto. **Data Mining Applied to Deadlines and Priorities Control on Software Projects.** 2017. 66 p. Work of Conclusion Course (Graduation in Systems Analysis and Development Technology) – Federal Technological University of Paraná. Ponta Grossa, 2017.

Currently, software products are produced at an industrial level and with increasing complexity, so that their production requires a considerable number of people working simultaneously and following the same processes. Among the main problems encountered in these scenarios are the proper allocation of priorities, deadlines and how to define the importance of the requirements, as well as estimate the maximum deadline for delivery so that it is produced efficiently and safely. Although there are currently project management systems, they leave only the project manager responsible for the actions to be taken in order for the project to be successfully completed. The features offered in these systems do not assist the manager in his strategic decisions, such as negotiating a longer deadline for delivery of a requirement that will not be completed in time. On the other hand, there are computational algorithms and techniques capable of assisting decision making in different scenarios. It is assumed that the use of Data Mining, combined with methodologies for management of schedules and risks, can offer managers suggestions and / or alternatives for problems encountered in the software development process. In this line, this project aimed to apply Data Mining algorithms for Knowledge Discovery in Databases, specifically Decision Tree Classification algorithms, in the development of a project management system, which will recommend management actions in management Project deadlines and priorities. Results show that the system assists managers in the designation of priorities and deadlines, through records of the manager's own behavior in critical situations, in order to obtain greater success in their projects.

Keywords: Project Management. Data Mining. Knowledge Discovery in Databases

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE SIGLAS E ACRÔNIMOS

API	Application Programming Interface
PMI	Project Management Institute
XP	Extreme Programming

SUMÁRIO

1 INTRODUÇÃO

Atualmente, os produtos de software são produzidos em nível industrial e com complexidade cada vez maior, de forma que sua produção requer um número considerável de pessoas trabalhando simultaneamente e seguindo os mesmos processos (PRESSMAN, 2011). Um dos principais problemas encontrados nestes cenários é como administrar situações críticas, especialmente questões relacionadas ao prazo de entrega ou à priorização dos requisitos do projeto para que este seja seguro e eficiente. Segundo o PMI (2008), é necessário que um projeto seja monitorado, periodicamente ou em situações críticas, para analisar seu desempenho e identificar mudanças que devem ser realizadas para a performance do projeto retorne ao ritmo de desenvolvimento (PROJECT MANAGEMENT INSTITUTE, 2008).

Um projeto é definido como “um empreendimento temporário com o objetivo de criar um produto ou serviço único” (PROJECT MANAGEMENT INSTITUTE, 2008) e é composto por uma sequência de atividades executadas por pessoas dentro de um limite de tempo, recursos e custos.

O PMI (2008) também estabelece uma sequência de etapas a serem seguidas para a gestão de tempo no ciclo de vida de um projeto, sendo elas: a elaboração de um planejamento, a definição e priorização das atividades, o cálculo de estimativas de recursos e duração dessas atividades, o desenvolvimento do cronograma e, por fim, o seu controle. Neste trabalho serão abordadas apenas questões relacionadas à priorização das atividades e cronograma, etapas serão melhores detalhadas no capítulo 2.2.

Observando as etapas envolvidas no processo de projetos, alguns softwares foram desenvolvidos nos últimos anos para auxiliar no planejamento, realização e controle de projetos, bem como na gestão de tempo de suas atividades. Dentre os principais estão: Runrun.it (RUNRUN.IT, 2015), ActiveCollab (ACTIVECOLLAB, 2015) e Wrike (WRIKE INC, 2015). Os sistemas citados buscam prover um ambiente colaborativo, de *feedback* rápido, onde os usuários podem trabalhar e se comunicar ao longo do desenvolvimento do software. Entretanto, cabe somente ao próprio gestor tomar as decisões de atribuir e controlar as atividades, recursos, prazos e prioridades, não havendo ferramenta ou recurso que

apoie sua decisão em relação a esses aspectos, além de gráficos que devem ser analisados pelo gestor para avaliar o desempenho do cronograma estipulado.

Por outro lado, existem técnicas computacionais capazes de auxiliar a tomada de decisão nos mais diferentes cenários, como a Mineração de Dados. A Mineração de Dados visa também descobrir conhecimentos novos em conjuntos de dados. A Descoberta de Conhecimento em Bases de Dados tem como finalidade a extração de informação útil em um conjunto de dados por meio do uso de técnicas de Mineração de Dados para descobrir, por exemplo, padrões estruturais nos dados analisados (WITTEN, FRANK, & HALL, 2011). Os padrões identificados nos dados podem ser utilizados para obter previsões sobre situações futuras. Dentre as tarefas utilizadas na Mineração de Dados, destaca-se a Classificação, que visa analisar um conjunto de dados com o objetivo de determinar a melhor forma de mapear uma instância a uma determinada categoria. Este trabalho propõe que, com o uso de tais técnicas combinadas com as metodologias de gestão de cronograma em projetos, os gestores tenham um conjunto maior de sugestões e/ou alternativas para o tratamento de desvios de planejamento em relação ao planejado no cronograma durante a execução de um projeto.

Observando as ferramentas atuais e a ausência de mecanismos de apoio à decisão aos gestores de projetos, este trabalho objetiva-se a aplicar os métodos de Mineração de Dados baseados em técnicas de Classificação por Árvores de Decisão no desenvolvimento de um sistema de gestão de projetos que irá recomendar ações tratativas para adequar os cronogramas dos projetos.

A ferramenta proposta visa auxiliar o gestor na tomada de decisões, com foco na priorização de determinado projeto de acordo com seu desvio no cronograma, para que esse possa ser concluído dentro do previsto. Espera-se que o sistema possibilite uma maior precisão e chance de sucesso ao processo de desenvolvimento de software.

1.1 OBJETIVOS

Para melhor delineamento do trabalho foram definidos os seguintes objetivos:

1.2 OBJETIVO GERAL

Desenvolver um sistema de gestão de projetos de software que forneça recomendações de ações tratativas relacionadas à priorização e prazo de projetos visando o cumprimento de seus cronogramas.

1.2.1 Objetivos Específicos

Em conjunto com o objetivo geral deste trabalho buscou-se alguns objetivos específicos, citados abaixo:

- Identificar as principais metodologias e fatores considerados para o monitoramento e controle de prazos e prioridades;
- Identificar e analisar as diferentes abordagens e tarefas usadas em de Mineração de Dados para a tarefa de Classificação que possam ser usadas para gerir prazos e prioridades de projetos;
- Estabelecer um conjunto de métricas para monitoramento de irregularidades em projetos;
- Desenvolver uma ferramenta mínima viável para gestão de prioridades e prazos em projetos de software;
- Incorporar a Mineração de Dados na ferramenta implementada, baseando-se nas métricas estabelecidas, para auxiliar na tomada de decisões do gestor;
- Avaliar a solução desenvolvida e seu impacto no gerenciamento de situações de irregularidade em prioridades e prazos de projetos.

1.3 ESTRUTURA DO TRABALHO

Este trabalho apresentou no primeiro capítulo uma introdução do assunto abordado, o que motivou o desenvolvimento deste e os objetivos esperados com o desenvolvimento. O segundo capítulo apresenta as metodologias de desenvolvimento de software, bem como aborda técnicas de gestão de cronograma

em projetos e apresenta os conceitos utilizados no processo de Descoberta de Conhecimento em Bases de Dados. O terceiro capítulo descreve a metodologia usada para elaboração deste trabalho. O quarto capítulo apresenta os resultados obtidos e, por fim, o quinto capítulo apresenta as considerações finais e sugere trabalhos futuros relacionados ao resultado deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda inicialmente algumas metodologias de desenvolvimento comumente usadas. Em seguida, são abordados conceitos relacionados aos processos envolvidos em projetos de software, com o objetivo de mostrar a importância da realização de um planejamento bem estruturado ao desenvolver um produto de software. Após, são apresentados modelos de processos de desenvolvimento e gestão de projetos de software. Por fim, este capítulo apresenta uma breve introdução à Mineração de Dados, com foco no ciclo e nas tarefas usadas na Descoberta de Conhecimento em Bases de Dados.

2.1 ENGENHARIA DE SOFTWARE

A Engenharia de Software trata, dentre outros temas, dos processos, métodos, técnicas e ferramentas utilizados no desenvolvimento de software (PRESSMAN, 2011). Por meio da Engenharia de Software é possível aumentar a confiança, a facilidade de manutenção, a chance de aceitação do produto final, uma vez que esta preza por uma abordagem sistemática e organizada para seu desenvolvimento (SOMMERVILLE, 2007). Estes fatores fazem com que o produto seja melhor não somente para o cliente, mas também para a equipe de programadores e analistas que o desenvolve.

Segundo Pressman (2011), toda equipe de desenvolvimento de software está sujeita a problemas no decorrer do processo de software. Entretanto, a Engenharia de Software provê soluções comprovadas para esses problemas com a finalidade de solucioná-los de forma rápida e eficaz (PRESSMAN, 2011), tais como modelos de processos, gestão da qualidade e testes de software. O que a Engenharia de Software prega, portanto, é que seja estabelecida e seguida uma série de passos previsíveis e de eficiência comprovada, teórica e praticamente, que ajudem a criar um resultado de alta qualidade e dentro do prazo estabelecido. Essa sequência de passos previsíveis é denominada “processo de software”, abordado na seção seguinte.

2.1.1 Processos de Desenvolvimento de Software

Os processos de software têm por objetivo produzir um software de alta qualidade ao estabelecer uma metodologia para as atividades, ações e tarefas envolvidas no processo de desenvolvimento (PRESSMAN, 2011). Os processos auxiliam na padronização da qualidade do produto gerado, descrevem problemas comumente encontrados durante o ciclo de vida de projetos de software, bem como oferecem uma ou mais soluções comprovadas para esses problemas.

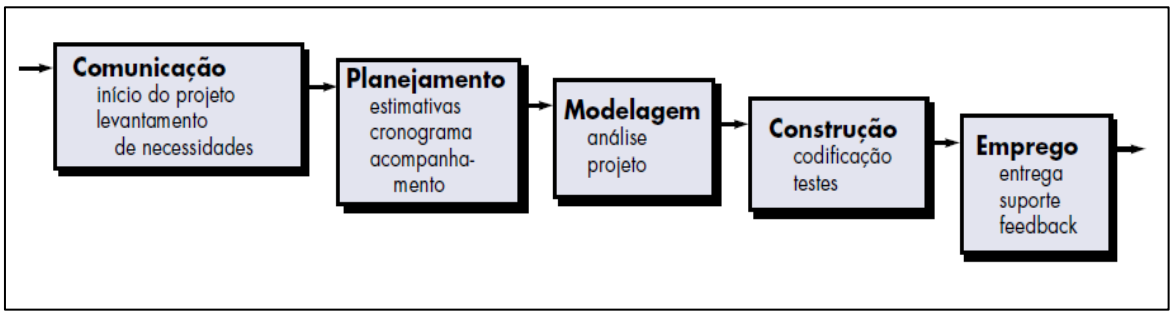
Os processos podem especificar desde a solução de um problema completo ou problemas específicos encontrados nas etapas do projeto, como no planejamento ou na análise de requisitos. Processos diferentes podem também ser combinados para aumentar ainda mais o volume de problemas solucionados no processo (PRESSMAN, 2011). Pode-se, por exemplo, utilizar-se de protótipos na metodologia de cascata.

Dentre os principais tipos de processos de desenvolvimento de software pode-se citar os modelos: Cascata, Incremental, Evolucionário, Concorrente e os modelos ágeis. Cada modelo se diferencia dos demais em relação por possuir regras próprias a serem seguidas durante o desenvolvimento do software, discutidas a seguir.

2.1.1.1 Modelo cascata

O modelo de cascata ou ciclo de vida clássico, sugere que as etapas do desenvolvimento sigam um fluxo sequencial. A Figura 1 demonstra o fluxo em cascata, que se inicia com o levantamento de necessidades por parte do cliente, avançando pelas fases de planejamento, modelagem, construção, emprego e encerrando no suporte contínuo do software concluído (PRESSMAN, 2011).

Figura 1 - O modelo de cascata



Fonte: Pressman (2011)

No modelo de cascata o processo inicia com uma comunicação junto ao cliente para realizar o levantamento dos requisitos. Nessa etapa, também, os envolvidos no projeto são comunicados sobre aspectos como as metas do projeto e seus requisitos. Em seguida é realizado um planejamento para obter estimativas de esforço e recursos necessários para a execução do projeto, bem como a definição de prazos.

Somente com as duas etapas anteriores concluídas dá-se início ao trabalho de análise dos requisitos levantados e de modelagem do sistema. Por meio da modelagem a equipe passa a ter uma abstração das funcionalidades solicitadas ou erros identificados pelo cliente. Ao final desta etapa os programadores podem iniciar seu trabalho na fase de Construção, onde o software é implementado e testado.

Finalmente, o produto é entregue ao cliente, que o irá avaliar e prover seu *feedback* a respeito do produto. Caso necessário, o produto deverá ser ajustado de acordo com as especificações do cliente.

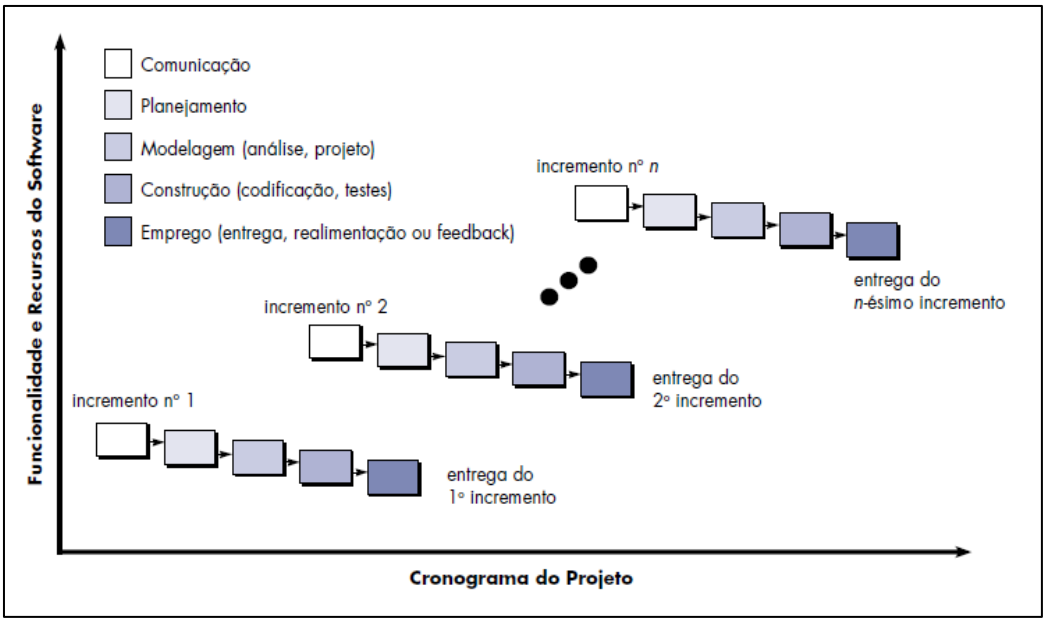
O modelo de cascata propõe que o resultado de uma etapa seja utilizado como entrada para a etapa seguinte, de forma que ela não seja iniciada enquanto sua predecessora não for concluída. Entretanto, o projeto todo fica em risco de atraso ou de não atender os requisitos caso haja um erro em uma das etapas iniciais, aumentando o custo necessário para realizar as correções nas fases finais do processo (SOMMERVILLE, 2007).

2.1.1.2 Modelo incremental

O modelo incremental, mostrado na Figura 2, emprega as mesmas etapas utilizadas no modelo de cascata, porém de forma escalonada. Neste modelo

o projeto é dividido em partes menores e cada parte segue o ciclo de cascata individualmente (PRESSMAN, 2011).

Figura 2 - O modelo incremental



Fonte: Pressman (2011)

O modelo de processo incremental segue a mesma sequência de etapas que o modelo de cascata, porém tem seu foco voltado para a entrega de um produto operacional com cada incremento, ou seja, os requisitos são levantados, planejados, analisados, implementados e entregues em pequenos grupos ou módulos (PRESSMAN, 2011).

2.1.1.3 Modelos evolucionários

A abordagem evolucionária visa rapidez na entrega de um produto inicial ao cliente. Esse produto inicial é validado e o cliente provê *feedback* a respeito do que foi desenvolvido até o momento. Após esse *feedback,* o produto inicial é melhorado de acordo com os requisitos faltantes e um novo *feedback* é realizado. O processo é repetido e gera versões cada vez mais completas do produto, até que este esteja completo (SOMMERVILLE, 2007).

Dentre os principais modelos de processo evolucionários utilizados por equipes de desenvolvimento de software pode-se citar o modelo de prototipação (SOMMERVILLE, 2007). Nele, uma versão inicial do software é rapidamente

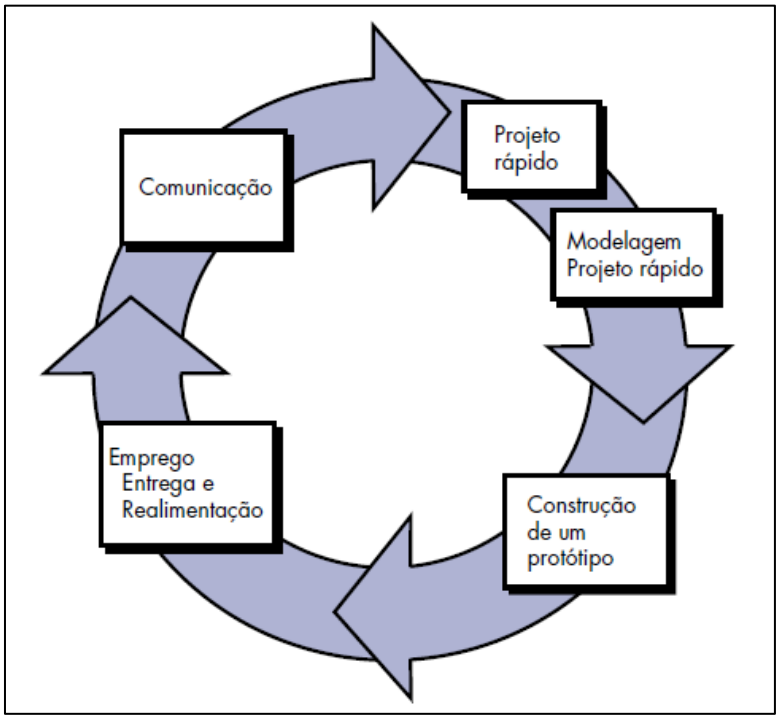
disponibilizada ao cliente, que provê *feedback* sobre o produto desenvolvido. Esse *feedback* é usado como entrada para melhorar o protótipo inicial, tornando-o mais completo.

Segundo Pressman (2011), o paradigma da prototipação, mostrado na Figura 3, começa com a comunicação entre o cliente e a equipe de desenvolvimento de software. Faz-se uma reunião com os envolvidos para definir os objetivos e identificar os requisitos já conhecidos do software.

Após a definição dos objetivos e requisitos, ocorre a modelagem e elaboração de um projeto rápido, denominado protótipo, que se concentra nos aspectos do software que serão visíveis aos usuários finais (por exemplo, o layout da interface com o usuário ou a interação entre as telas) (PRESSMAN, 2011).

O protótipo é então empregado e avaliado pelos envolvidos, os quais fornecerão um retorno que servirá para aprimorar os requisitos. A cada iteração o protótipo é ajustado às necessidades do cliente, sendo acredito de novas necessidades que devem ser atendidas (PRESSMAN, 2011). O produto de software é dado como concluído uma vez que todos os requisitos do cliente tenham sido atendidos.

Figura 3 - O modelo de prototipação

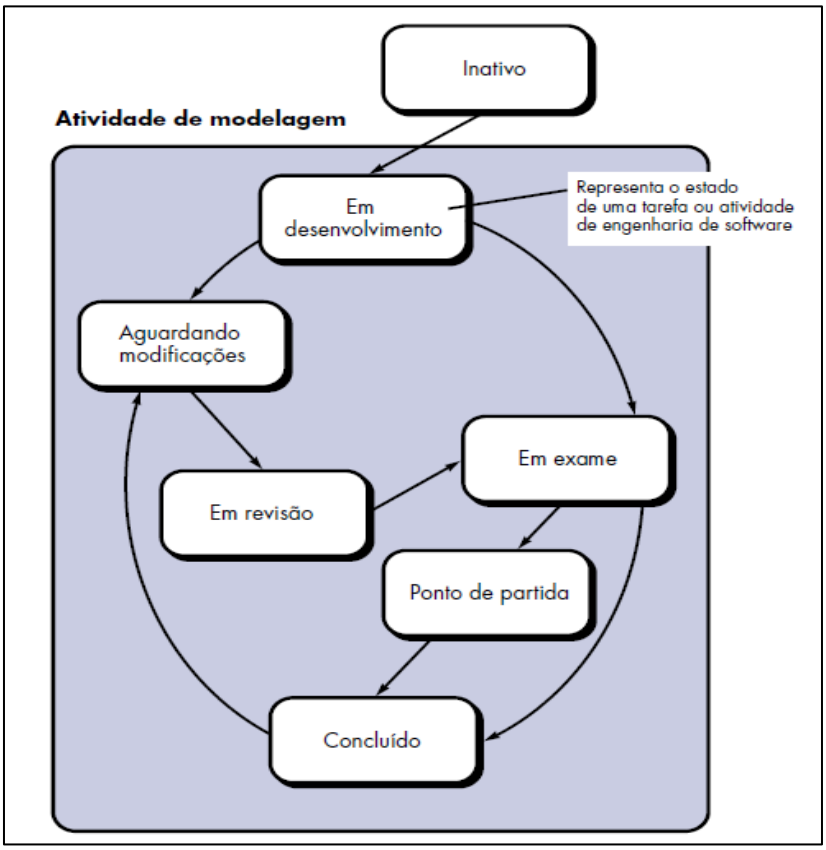


Fonte: Pressman (2011)

2.1.1.4 Modelo concorrente

O modelo de desenvolvimento concorrente apresenta os processos de desenvolvimento de software na forma de estados que podem ocorrer de forma concorrente e iterativa (PRESSMAN, 2011). Na Figura 4 é apresentado o esquema de uma atividade de modelagem de software utilizando a abordagem concorrente.

Figura 4 - O modelo concorrente



Fonte: Pressman (2011)

No modelo concorrente uma tarefa pode se encontrar em qualquer determinado estado a qualquer momento de seu ciclo de vida. Uma tarefa pode iniciar como inativa, passar ao estágio de desenvolvimento e, caso existam alterações nos requisitos, a atividade pode ir para o estado em que aguarda modificações ou para exame do que foi executado. Caso seja necessário, uma tarefa também pode voltar ao ponto de partida, caso contrário é dada como concluída até que novas alterações sejam necessárias (PRESSMAN, 2011).

2.1.2 Modelos Ágeis

Segundo Sommerville (2007), entre os anos 1980 e 1990 houve a difusão da ideia de que era necessário um cuidadoso planejamento e cumprimento de processos formalizados para a garantia da qualidade de um projeto de software. O descontentamento com os modelos existentes até então levou à criação de novos modelos mais ágeis de desenvolvimento de software, que permitissem aos desenvolvedores focar no software em si ao invés de projeto e documentação.

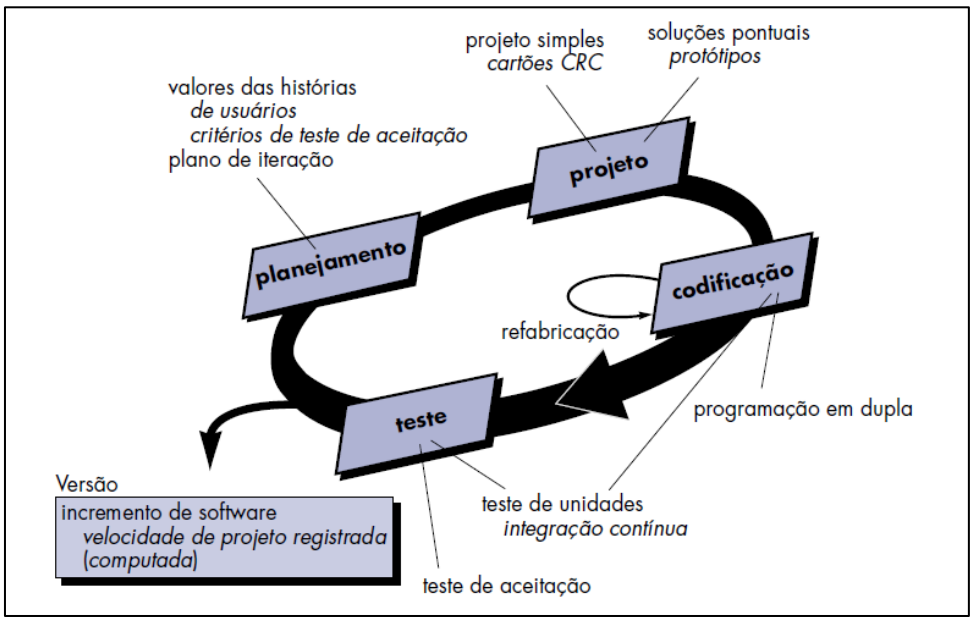
O foco dos modelos ágeis é a satisfação do cliente, a motivação da equipe, o uso de métodos menos formais e a simplicidade nos processos e no produto desenvolvido (PRESSMAN, 2011). Segundo Sommerville (2007), o envolvimento ativo do cliente no processo de desenvolvimento é de grande importância, especialmente na provisão e priorização dos requisitos, bem como na avaliação da evolução do projeto**.** Além disso, o sistema deve ser projetado para acomodar mudanças, pois elas devem ser esperadas.

Os princípios ágeis priorizam a entrega mais do que a análise e o projeto, embora essas atividades não sejam desencorajadas. Eles também priorizam a comunicação ativa e contínua entre desenvolvedores e clientes. De acordo com Sommerville (2007), dentre os modelos de processos ágeis mais comumente utilizados podem ser citados o Extreme Programming (XP) e o Scrum.

2.1.2.1 Extreme programming

O *Extreme Programming* (XP) é o mais difundido e utilizado dos métodos ágeis (SOMMERVILLE, 2007). O XP expressa todos os requisitos como cenários (ou histórias) que são implementadas como uma série de tarefas. A Figura 5 apresenta o esquema do modelo XP.

Figura 5 - O modelo XP



Fonte: Pressman (2011)

O processo de desenvolvimento inicia-se pela seleção das histórias dos usuários (cenários) de forma a entender o ambiente de negócios do cliente e suas necessidades (PRESSMAN, 2011). Cada história descrita pelo cliente possui um valor (prioridade) atribuído a ela pelo cliente ou gestor do projeto e então a equipe de desenvolvimento de software atribui um custo àquela história, que pode ser medido em horas, dias, semanas, etc.

Na fase de projeto, cada história é analisada com o objetivo de gerar um guia de implementação que atenda estritamente às necessidades daquela história (PRESSMAN, 2011). No XP, qualquer funcionalidade extra é desencorajada, pois o modelo preza pela simplicidade, focando apenas nas necessidades imediatas e não futuras.

Na etapa de codificação, as histórias são implementadas, segundo o roteiro elaborado na etapa de projeto. Assim como no projeto, durante a codificação funcionalidades extras não são encorajadas pelo XP (SOMMERVILLE, 2007). Finalmente, testes são realizados para avaliar a solução implementada e caso seja aceita, o requisito é dado como concluído. O cliente também realiza um teste de aceitação no produto gerado. Após sua aprovação, o software pode ser entregue ao ambiente de produção.

2.1.2.2 Scrum

O Scrum visa a facilidade de entendimento e adaptabilidade a diferentes cenários. Ele é formado por um time que obedece a um conjunto de papéis, artefatos, regras e eventos, sendo que cada um desses componentes serve a um propósito específico (PRESSMAN, 2011).

Dentre os artefatos utilizados pelo método Scrum, o mais notável é o *Backlog* do produto, que é composto, assim como no XP, por uma lista ordenada por prioridade dos requisitos do produto (SCHWABER & SUTHERLAND, 2013). A cada iteração, itens do *Backlog* do produto são selecionados para compor as funcionalidades da próxima iteração, denominada no Scrum como *Sprint*.

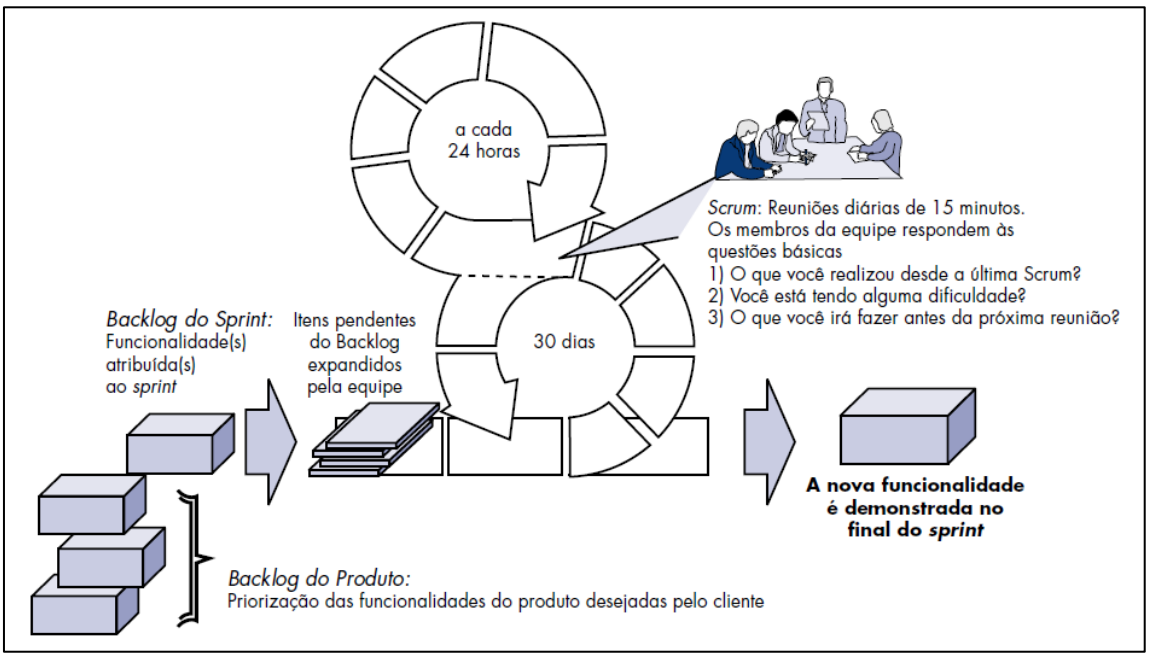
Ao final da *Sprint*, é obtido um incremento composto pela soma de todos os itens no *Backlog* da *Sprint* concluída, adicionado ao valor dos incrementos anteriores. O incremento deve ser julgado como pronto (utilizável) para que possa ser adicionado ao produto (SCHWABER & SUTHERLAND, 2013).

Após o incremento do produto estar pronto para ser integrado, é realizada uma reunião de revisão da *Sprint* em que o time e, preferencialmente, o cliente revisam e discutem o incremento, bem como identificam o que deve ser feito em seguida, alimentando o *Backlog* do produto e também o planejamento da próxima *Sprint*.

De forma geral, o processo de gestão no método Scrum, como mostrado na Figura 6, segue a seguinte sequência de etapas:

- Elaboração do *Backlog* do produto;
- Reunião de planejamento de *Sprints*;
- Reunião diária;
- Desenvolvimento;
- Incremento do produto;
- Revisão da *Sprint*;
- Retrospectiva da *Sprint*;

Figura 6 - O modelo Scrum



Fonte: Pressman (2011)

2.2 GESTÃO DE PROJETO DE SOFTWARE

Segundo Pressman (2011), a gestão de projeto de software é uma atividade de apoio à Engenharia de Software e deve ser iniciada antes de qualquer outra atividade técnica. A gestão deve continuar ao longo da modelagem, construção e até mesmo utilização do software desenvolvido.

De acordo com o PMI (2008), um dos pontos fundamentais a se gerenciar em um projeto é seu cronograma. O PMI prevê uma sequência de etapas na gestão do cronograma um projeto, sendo elas: a elaboração de um planejamento, a definição e priorização das atividades, o cálculo de estimativas de recursos e duração dessas atividades, o desenvolvimento do cronograma e por fim o seu controle.

Na fase de planejamento, o gestor deve identificar as atividades a serem executadas, bem como definir a forma como essas serão monitoradas e controladas para que o projeto seja bem-sucedido.

Na fase de definição das atividades, o gestor decompõe o escopo do projeto em partes menores, denominadas pacotes de trabalho, que poderão ser gerenciadas mais facilmente. A priorização das atividades tem como objetivo identificar atividades que dependem da conclusão de outras atividades para serem

iniciadas, de forma que o trabalho possua uma sequência lógica. Pressman (2011) prevê que atribuição da prioridade de um item pode levar em consideração sua importância para o cliente. Caso o cliente não especifique explicitamente a prioridade de um item ou projeto sobre outro, essa tarefa deve ser feita pela equipe ou gestor do projeto.

Quando vários projetos são executados simultaneamente e concorrendo por recursos, eles devem ser constantemente revisados de forma que mais recursos sejam alocados a projetos com maior prioridade. Projetos em execução podem ser cancelados, atrasados ou acelerados por esta prática (NICHOLAS & STEYN, 2012). Os critérios utilizados para essa priorização podem variar desde retorno financeiro até benefícios intangíveis como aumento do portfólio da organização.

Segundo Nicholas & Steyn (2012), existem métodos heurísticos para a determinação de prioridades, sendo eles:

- **O mais cedo possível:** a prioridade das atividades ou projetos é determinada de acordo com sua previsão de início, de forma que aqueles que possam ser iniciados antes recebem prioridade maior;
- **O mais tarde possível:** a prioridade é determinada de acordo com a previsão de término, de forma que itens ou projetos que possam ser concluídos mais tarde recebem prioridade menor;
- **Quantidade de recursos:** projetos ou itens que necessitem de mais recursos, tais como pessoas ou máquinas recebem prioridade maior;
- **Menor tempo de tarefa:** projetos ou itens de duração mais curta recebem prioridade maior que aqueles com maior tempo de execução;
- **Menor folga:** a prioridade é determinada de acordo com a folga estimada para cada projeto ou item, de forma que aquelas com menor folga recebem maior prioridade;
- **Ordem de chegada:** projetos ou itens que chegam primeiro recebem prioridade maior que aqueles que chegam mais tarde;
- **Menor prazo:** a definição da prioridade é determinada de acordo com o prazo de entrega do projeto ou item, de forma que aqueles cujo prazo de entrega está menor recebe maior prioridade.

Na sequência definida pelo PMI (2008), após a definição dos prazos e prioridade, deve-se calcular as estimativas de recursos e duração. O gestor define a

quantidade de recursos necessários para a realização de cada atividade bem como a duração necessária para que o trabalho seja concluído com os recursos estimados.

Antes mesmo de serem iniciadas as atividades técnicas de um projeto, como análise de requisitos, modelagem e implementação, é imprescindível para o planejamento e posterior controle que o trabalho a ser realizado seja estimado, bem como os recursos necessários e o tempo de duração de cada tarefa. Uma estimativa é uma previsão confiável dos recursos e materiais que serão necessários para a conclusão de uma determinada tarefa e é considerada de importância fundamental para o planejamento do projeto (PRESSMAN, 2011).

Quando se fala em estimativas, está-se tratando na realidade de diversas medidas diferentes, como tamanho, esforço, recursos, tempo e custo. Estimativas de tamanho geralmente são usadas para avaliar a complexidade de uma tarefa, ao passo que estimativas de esforço são usadas para avaliar o tempo necessário para concluí-la. A partir das estimativas o gestor possui fundamentos para negociação de prazos com o cliente, bem como para alocar os recursos necessários. Com isso é possível elaborar o cronograma do projeto de forma mais confiável, bem como obter uma previsão do custo do projeto.

Segundo Pressman (2011), os resultados medidos em uma iteração ou projeto anterior podem ser utilizados como base para a definição das estimativas para o projeto que está sendo planejado. O PMI (2008) também prevê o uso de informação histórica para fundamentar o planejamento de novos projetos, pois essas estatísticas podem ser úteis em decisões tais como o escopo e viabilidade do projeto.

Após o estabelecimento das estimativas, o PMI prevê o desenvolvimento do cronograma, ou seja, a conversão das diversas atividades a serem realizadas no projeto, juntamente com suas prioridades e estimativas em um modelo lógico que determina, não apenas a sequência e duração das atividades, mas também deve prever potenciais atrasos decorrentes de problemas no projeto. Finalmente o cronograma determinado para o projeto passa a ser monitorado ao longo do tempo, de forma a identificar desvios na execução do projeto em relação ao planejado para que seja possível tomar ações corretivas para que o fluxo de trabalho volte à normalidade (PROJECT MANAGEMENT INSTITUTE, 2008).

O cronograma do projeto deve ser controlado com frequência para que problemas sejam detectados por análises de desempenho. Essa análise tem como finalidade medir o desempenho do projeto e compará-lo com o desempenho projetado na fase de planejamento de forma a manter as datas de início e término das atividades, bem como o percentual de conclusão do projeto dentro dos limites aceitáveis por meio de ações corretivas (PROJECT MANAGEMENT INSTITUTE, 2008).

Segundo Nicholas & Steyn (2012), o monitoramento de um projeto deve incluir a coleta e interpretação de dados relacionados aos projetos. Tais dados devem refletir os objetivos do projeto, bem como seu andamento e devem se basear nos processos, políticas e padrões utilizados no desenvolvimento do produto com a finalidade de determinar estatisticamente se os objetivos do projeto estão sendo atingidos (NICHOLAS & STEYN, 2012). Na Engenharia de Software, tal coleta e interpretação de dados segue um processo denominado medição.

As métricas de projeto e processos de software são medidas quantitativas que permitem ao gestor de projeto ter uma estimativa da eficácia dos processos de software nos projetos que são executados (PRESSMAN, 2011). Como entrada para as medições, são coletados dados a respeito da qualidade do produto de software e da produtividade da equipe que o produz, como, por exemplo, o número de linhas de código ou quantidade de erros. Esses dados são analisados e comparados com medições passadas ou metas pré-estabelecidas e avaliados para determinar se ocorreram melhorias na qualidade e produtividade.

Segundo Pressman (2011), as métricas podem ser classificadas nas seguintes categorias:

- **Orientadas a Tamanho:** número de erros, custo monetário, quantidade de documentação por milhar de linhas de código;
- **Orientadas a Função:** pontos por função contados na aplicação e complexidade do software;
- **Orientadas a Objeto:** número de classes-chave, número de classes de apoio, complexidade de métodos;
- **Orientadas a Caso de Uso:** funções e características visíveis diretamente ao usuário.

Segundo Sommerville (2007), o processo de medição de software deve obedecer a uma sequência de etapas:

- **Escolha das métricas:** o objetivo a ser alcançado pela da medição, ou seja, as perguntas que ela irá responder;
- **Seleção dos componentes auditados:** devem ser selecionadas partes individuais do software ou do processo, geralmente as partes mais críticas;
- **Medição das características:** os componentes definidos na etapa anterior têm seus atributos medidos e um resultado numérico derivado é registrado;
- **Identificação de anomalias:** os valores obtidos na medição são analisados e comparados a indicadores anteriores ou metas para identificar áreas com problemas.

O método GQM (*Goal-Question-Metric*) proposto por Basili (2002) pode ser aplicado para abordar a etapa de escolha das métricas. GQM consiste no estabelecimento de uma meta inicial (*goal*) para a avaliação a ser realizada para então levantar questões relevantes norteadas por essa meta (*question*) com o objetivo de atingi-la e, finalmente, identificas as métricas e fórmulas (*metric*) que irão responder a essas perguntas (PRESSMAN, 2011).

De acordo com Basili (2002), um objetivo ou meta de medição deve consistir em uma questão, um objeto, um ponto de vista e um propósito, por exemplo: “diminuir o volume de erros nos produtos de software do ponto de vista do cliente final”. Nesse objetivo, identificam-se os seguintes elementos:

- **Questão:** volume de erros;
- **Objeto:** produtos de software;
- **Ponto de vista:** cliente final;
- **Objetivo:** diminuir.

Uma vez que o objetivo esteja definido, consideram-se as perguntas que irão auxiliar a equipe a alcançar esse objetivo, tais como:

- Qual o volume de erros atual?
- Quantos erros o cliente final percebe?
- Quantos dos erros são críticos?

Com as perguntas formuladas, pode-se finalmente conceber as métricas a serem calculadas para responder cada uma das perguntas. Para responder o questionamento sobre o volume de erros que o cliente percebe, pode-se contar o número de reclamações ou suporte técnico solicitado pelo cliente onde este aponta algum erro.

Métricas são especialmente úteis na identificação de áreas com problemas no software ou no processo de desenvolvimento, pois fornecem resultados que podem ser comparados com medições ou metas anteriores. Desse modo, as correções podem ser aplicadas a tempo de prevenir potenciais aumentos de custo e esforço, melhorando o processo de software e, como consequência, o produto entregue ao cliente final (PRESSMAN, 2011).

As métricas especificadas pela equipe de projeto podem também ser transmitidas para os responsáveis sobre a qualidade e pelo aperfeiçoamento do processo de software, tais como: auditores internos de Sistemas de Gestão da Qualidade ou engenheiros de processos, por exemplo. Com isso, as mesmas métricas podem ser utilizadas em diversos domínios do processo e do projeto (PRESSMAN, 2011).

Segundo Pressman (2011) metodologias focadas em aperfeiçoamento de processos, tais como o CMMI, bem com sua versão adaptada à realidade brasileira, o MPS.BR utilizam-se das métricas com a finalidade de acompanhar a eficácia dos processos utilizados pela equipe. Embora sejam metodologias não prescritivas, ou seja, não forneçam uma metodologia específica para a realização dos processos, elas fornecem guias e objetivos a serem alcançados pelos processos, o que determinará o nível alcançado pela organização por uma entidade certificadora.

A partir da avaliação das métricas, pode-se obter uma estimativa sobre a situação do produto, projeto ou processo para determinar se seu resultado está dentro de um limite aceitável pré-determinado. Caso o valor esteja fora do aceitável, cabe aos responsáveis tomar as ações necessárias para corrigir os problemas.

2.3 MINERAÇÃO DE DADOS

Ao longo das últimas décadas, o volume de dados armazenados digitalmente tem crescido de forma exponencial. Empresas dos mais variados

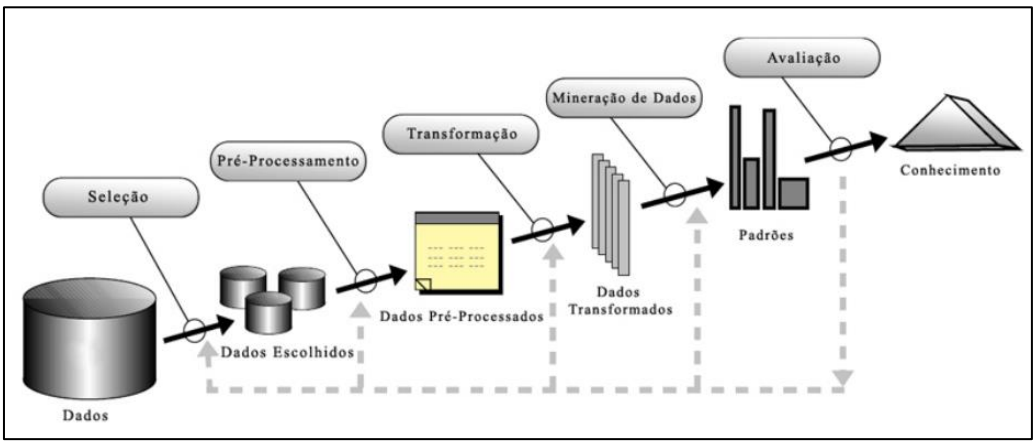
setores têm buscado coletar mais dados de seus negócios e clientes, registrando digitalmente suas menores atividades. Ao mesmo passo, a capacidade de *hardware* acompanha essa demanda, oferecendo dispositivos com potencial de armazenamento cada vez maior. Além disso, existem também serviços de hospedagem em nuvem que oferecem armazenamento de arquivos a preços tão baixos quanto uma fração de centavo por Gigabyte hospedado (WITTEN, FRANK, & HALL, 2011).

Entretanto, resta a questão do que fazer com esses grandes volumes de dados, uma vez que apenas coletá-los e armazená-los não serve de grande ajuda em decisões estratégicas (WITTEN, FRANK, & HALL, 2011). A Mineração de Dados serve ao propósito de extrair informação relevante de conjuntos de dados e transformá-las em conhecimento que pode ser utilizado de forma estratégica, como identificar o perfil de clientes e prever seus padrões de comportamento de forma a melhor atendê-los, por exemplo.

Balsera et al. (2012) utilizam-se de conceitos de Mineração de Dados por meio do uso do algoritmo de Curvas de Regressão Adaptativa Multivariada (MARS) sobre um conjunto de dados estatísticos referentes a projetos coletados pelo Grupo Internacional de Padrões de Melhoria de Software (ISBSG). Com a Mineração de Dados, identificou-se relações entre trabalho estimado e realizado, bem como número de defeitos em relação a aspectos, tais como: o tipo de linguagem ou metodologia de desenvolvimento utilizadas ou o tamanho da equipe (BALSERA, Joaquin V. et al., 2012). Desta forma, pode-se notar que o processo de Descoberta de Conhecimento em Bases de Dados pode também trazer benefícios estratégicos quando aplicada à área de Gestão de Projetos.

De acordo com Fayyad (1996), o processo de extração e descoberta de conhecimento em bases de dados ilustrado na Figura 7 segue uma série de etapas que devem ser executadas em sequência. Cada etapa provê informações a serem utilizadas pela etapa seguinte e passam por decisões feitas pelo usuário, evidenciando sua natureza interativa e iterativa.

Figura 7 - O processo de KDD



Fonte: Adaptado de Fayyad (1996)

Após a obtenção dos dados inicia o a etapa de Seleção, onde é definido o objetivo a ser alcançado com o processo de Mineração de Dados, levando em consideração os desejos do usuário, bem como entender o domínio da aplicação. Uma vez que os objetivos estejam definidos, pode-se utilizar essas informações para a criação de um conjunto de dados alvo (Dados Escolhidos) em que a Mineração será aplicada.

Com um conjunto inicial de dados obtidos, pode ser necessário realizar um pré-processamento dessas informações de forma a eliminar problemas que possam interferir negativamente na execução dos algoritmos de Mineração. São exemplos de problemas registros fora do padrão e registros gravados erroneamente. Nesta etapa são feitos também o tratamento de informações faltantes e formatação dos dados, tais como número de casas decimais, remoção de caracteres especiais, etc. Caso o conjunto de dados possua informações faltantes, pode-se simplesmente remover as instâncias com valores faltantes ou substituir esses valores globalmente no conjunto de dados pela média, caso trate-se de valores numéricos ou pela moda em caso de valores nominais (WITTEN, FRANK, & HALL, 2011).

Caso seja necessário, os dados pré-processados podem passar pelo processo de Transformação. Nesta etapa busca-se a redução dos dados (em faixas específicas), eliminação ou agrupamento de dados de forma a melhor representar o problema em questão. A Transformação pode ser realizada por meio da discretização de valores numéricos, agrupamento de múltiplas classes em classes mais sumarizadas ou a generalização de valores muito específicos em valores mais genéricos (WITTEN, FRANK, & HALL, 2011). Após essa etapa, o conjunto de dados

irá ser composto apenas pelos registros (instâncias) e colunas (atributos) mais relevantes ao contexto problematizado na etapa de Seleção.

Uma vez que o conjunto de dados foi pré-processado e transformado, ele é submetido a um ou mais algoritmos de Mineração de Dados. Dependendo do objetivo especificado na primeira etapa, pode-se escolher a tarefa mais adequada para o problema. Ao final do processo de Mineração um conjunto de padrões ou regras relacionadas aos dados é gerado, cabendo ao usuário realizar a avaliação das regras geradas com a finalidade de verificar sua validade e utilidade.

Os algoritmos de Mineração de Dados podem ser classificados em algumas categorias ou Tarefas, tais como: Classificação, Regressão, etc. (FAYYAD, SHAPIRO, & SMYTH, 1996). Cada tarefa pode ser classificada em dois grandes grupos de acordo com o modo de aprendizagem do algoritmo:

- **Aprendizagem Supervisionada:** requer que o usuário informe um atributo alvo, o qual deve estar presente no conjunto de dados. A Aprendizagem Supervisionada possui natureza preditiva, ou seja, a partir de um conjunto de amostras relacionadas à correta avaliação de um atributo alvo (classe), os algoritmos conseguem identificar padrões e regras. Essas regras podem ser utilizadas para prever o atributo classe em instâncias ainda não vistas;
- **Aprendizagem Não Supervisionada:** ao contrário da Aprendizagem Supervisionada, essa categoria não necessita que uma classe alvo seja estabelecida. A aprendizagem não supervisionada possui natureza de caráter mais exploratório, onde busca-se a relação entre os atributos do conjunto de dados sem o conhecimento prévio da classificação da instância.

Dentre as Tarefas mais comumente utilizadas na Aprendizagem Supervisionada, pode se citar (WITTEN, FRANK, & HALL, 2011):

- **Classificação:** tem por objetivo analisar um conjunto de dados em busca de padrões nos atributos de forma a estabelecer regras para classificar novas instâncias em classes pré-definidas, por exemplo: determinar as condições ideais para se realizar um jogo ao ar livre;
- **Regressão:** similar à Classificação, exceto pelo fato de ser utilizada para prever valores numéricos ao invés de categóricos. Pode ser utilizada para prever um valor futuro de um atributo classe com base nos demais atributos, por exemplo:

analisar o perfil de um consumidor e determinar quanto ele irá gastar a partir de um histórico de consumo.

Dentre as Tarefas mais comumente utilizadas na Aprendizagem Não Supervisionada, pode se citar (WITTEN, FRANK, & HALL, 2011):

- **Agrupamento (*Clustering*):** na Tarefa de agrupamento, busca-se separar instâncias com atributos semelhantes em conjuntos. Um conjunto, ou *cluster*, contém registros similares entre si, porém diferentes dos demais. Um exemplo de sua utilização seria a identificação de grupos de consumidores com comportamentos semelhantes;
- **Associação:** nesta Tarefa, o objetivo é identificar atributos que estejam relacionados, bem como determinar as regras para esses relacionamentos. Um exemplo de sua utilização seria identificar quais produtos são comumente comprados em conjunto com outros.

Para cada tarefa existem diferentes abordagens em sua aplicação. Na Tarefa de Classificação, por exemplo, pode-se utilizar:

- **Árvores de Decisão:** realiza testes em cada atributo (exemplo: temperatura > 23). Cada teste é representado como um nó (não folha) na árvore e é ligado a outro teste até que se chegue a um nó folha representando a classe à qual uma instância pertence (QUINLAN, 1994). Com a árvore gerada, pode-se utilizar as regras definidas por ela para classificar novos registros. Dentre os algoritmos comumente utilizados para geração de árvores de decisão em Mineração de Dados, pode-se citar o ID3 (QUINLAN, 1986) e C4.5 (QUINLAN, 1994).

O algoritmo ID3 verifica recursivamente qual o atributo mais importante em um conjunto de exemplos, ou seja, qual teste melhor divide o conjunto. Para cada resultado possível do teste é criado um nó folha. Em cada nó folha a verificação do melhor teste continua a ser aplicada e novos nós folhas a serem criados até que todos os possíveis resultados de um teste sejam idênticos (QUINLAN, 1990). Nesse caso, chega-se ao final da árvore onde a classe é estabelecida.

Para a verificação do melhor teste para o nó raiz ou para os nós folhas, o algoritmo ID3 utiliza-se de um cálculo para determinar o ganho de informação com a

finalidade de medir a efetividade de um teste em classificar um conjunto de dados. Em cada etapa do algoritmo ID3, todos os atributos ainda não testados têm seu ganho de informação medido e o atributo cujo ganho for maior é selecionada para representar o nó da árvore.

A medida de ganho de informação de um teste é representada por:

$$\mathrm{I}\,(\mathrm{S}) = -\sum_{j=1}^{x} FR\big(C_j, S\big)\ \log_2(FR(C_j, S)) \qquad (1)$$

Onde I representa o ganho de informação de um conjunto S. O cálculo é dado pela soma da frequência de todos os valores x. Calcula-se a probabilidade ou frequência relativa FR de um item no conjunto S pertencer a uma classe C_j. Esse valor é então multiplicado por seu logaritmo em base 2.

A medida de ganho de informação serve ao propósito de identificar qual atributo melhor separa o conjunto e é utilizada para separar o conjunto S em subconjuntos $S_1, S_2, \ldots, S_n$. O teste passa então pelo cálculo da taxa de ganho para cada subconjunto, dada por:

$$\mathrm{G}\,(\mathrm{S}, \mathrm{B}) = I(S) - \sum_{i=1}^{t} \frac{|S_i|}{|S|}\ I(S_i) \qquad (2)$$

Onde $\mathrm{G}\,(\mathrm{S}, \mathrm{B})$ determina o ganho de informação G de um teste B em no conjunto S. O ganho de informação $I(S)$ do conjunto é subtraído do somatório das razões entre o número de instâncias do subconjunto S_i e o número total de instâncias do conjunto multiplicadas pelo ganho de informação do subconjunto.

Tomando-se como base um problema clássico de Classificação ilustrado por Symeonidis & Mitkas (2005), onde busca-se determinar se as condições climáticas são propícias para a prática de golfe. Os dados analisados são tempo, temperatura, umidade e ocorrência de ventos. O Quadro 1 apresenta o conjunto proposto pelos autores.

Quadro 1 – Condições para prática de golfe

Instância	Tempo	Temperatura	Umidade	Ventos	Jogar
1	Ensolarado	Alta	Alta	Não	Não
2	Ensolarado	Alta	Alta	Sim	Não
3	Nublado	Alta	Alta	Não	Sim
4	Chuvoso	Moderada	Alta	Não	Sim
5	Chuvoso	Baixa	Normal	Não	Sim
6	Chuvoso	Baixa	Normal	Sim	Não
7	Nublado	Baixa	Normal	Sim	Sim
8	Ensolarado	Moderada	Alta	Não	Não
9	Ensolarado	Alta	Alta	Não	Não
10	Chuvoso	Moderada	Normal	Não	Sim
11	Ensolarado	Moderada	Normal	Sim	Sim
12	Nublado	Moderada	Alta	Sim	Sim
13	Nublado	Alta	Normal	Não	Sim
14	Chuvoso	Moderada	Alta	Sim	Não

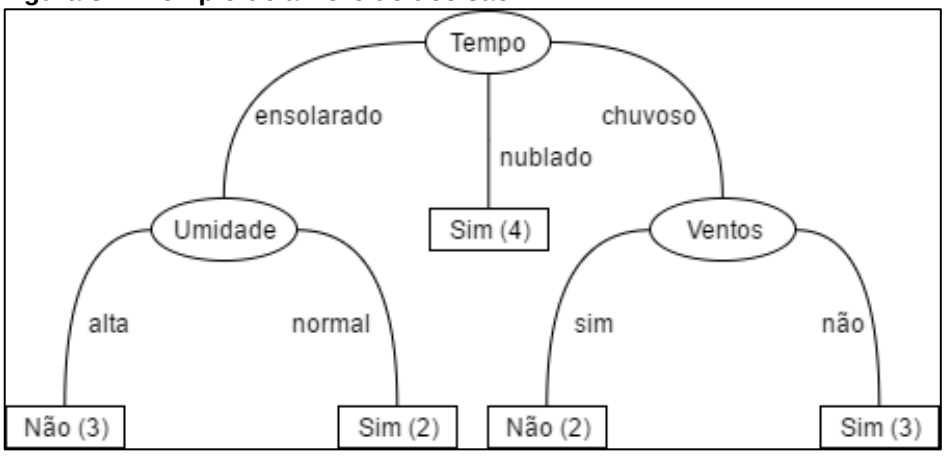
Fonte: Adaptado de Symeonidis & Mitkas (2005)

O conjunto acima possui 14 instâncias, das quais 9 são positivas e 4 negativas. Calculando-se o ganho de informação de cada atributo no conjunto, obtêm-se o valor de 0.246 para o atributo *tempo*, 0.029 para o atributo *temperatura*, 0.151 para o atributo *umidade* e 0.048 para o atributo *ventos*. Portanto, o atributo *tempo* é o que melhor separa o conjunto. Por este motivo ele passa a ser a raiz da árvore.

O conjunto é então separado em três subconjuntos menores agrupados de acordo com cada valor diferente do atributo *tempo*. Nota-se que, em todas as quatro instâncias (3, 7, 12 e 13) onde o atributo *tempo* é igual a *nublado*, a classe é igual a *sim*, portanto neste momento já é possível classificar os dados de acordo com este atributo.

Para os demais subconjuntos o cálculo de ganho de informação é realizado novamente, excluindo-se os atributos que já foram testados. O resultado final é ilustrado na árvore de decisão mostrada na Figura 8. O número ao lado de cada nó-folha representa a quantidade de instâncias que obedecem a todas as regras apresentadas nos demais nós desde a raiz.

Figura 8 - Exemplo de árvore de decisão



Fonte: Adaptado de Symeonidis & Mitkas (2005)

Para ilustrar a utilização da árvore de decisão gerada, considere a seguinte questão: “devo praticar golfe se o tempo estiver ensolarado e com a umidade normal?”. O primeiro passo é identificar os valores de cada atributo da questão: tempo = ensolarado e umidade = normal, para formar a instância a ser classificada. A instância é então submetida ao classificador, que irá comparar, de maneira *top-down*, os valores dos atributos da instância com os valores dos nós da árvore. Para esta questão o classificador irá retornar o valor s*im* como resposta ao questionamento.

Uma das principais desvantagens do algoritmo ID3 é a de que ele trata apenas atributos nominais, ignorando atributos numéricos. O algoritmo C4.5 é considerado uma evolução do ID3 e permite o tratamento de atributos tanto nominais quanto numéricos. O processo de geração da árvore de decisão no C4.5 segue o mesmo cálculo de ganho de informação do ID3 para valores nominais, entretanto quando o algoritmo encontra um valor numérico, ele realiza os testes nesses atributos a partir de um determinado limite. Os resultados, então são separados em valores menores ou iguais a esse limite e os maiores que o limite.

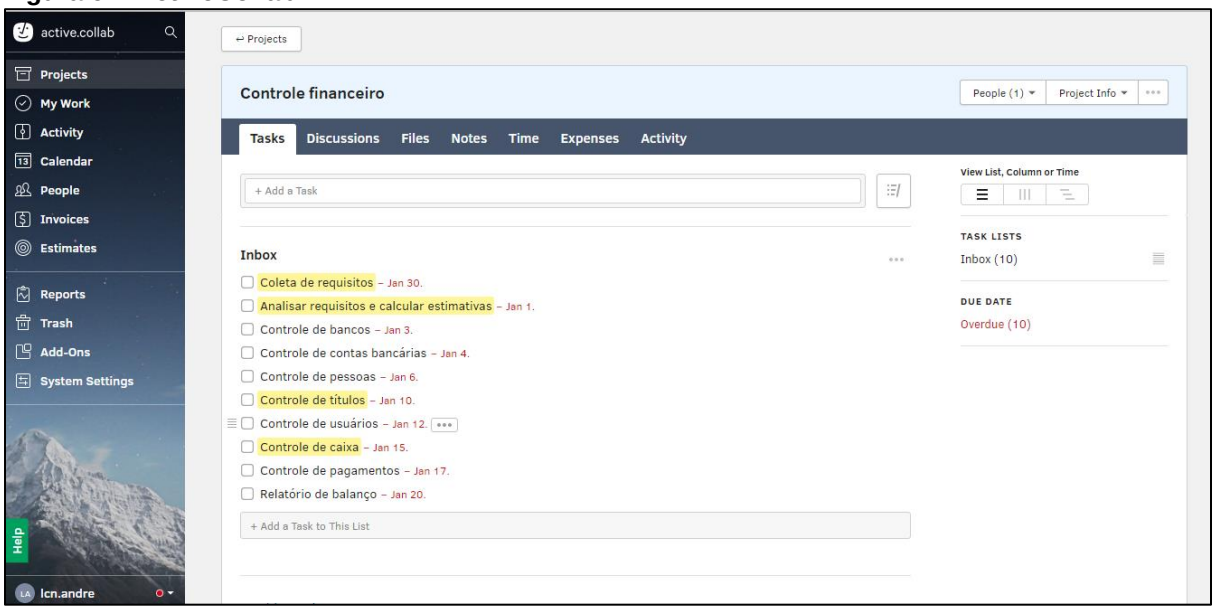
Para a determinação do limite, o algoritmo primeiro ordena todos os valores daquele atributo em ordem crescente na forma $\{v_1, v_2, \dots, v_n\}$. Cada valor v_n tem seu ganho de informação calculado com base em dois testes: valores menores ou iguais a v_n e valores maiores que v_n. O valor do atributo que possuir o maior ganho de informação é utilizado como nó na árvore de decisão.

2.4 SISTEMAS EXISTENTES

Alguns sistemas foram desenvolvidos nos últimos anos para auxiliar no planejamento, realização e controle de projetos, bem como na gestão de tempo de suas atividades. Dentre os principais comumente utilizados por gestores de projetos estão: ActiveCollab (ACTIVECOLLAB, 2015), Wrike (WRIKE INC, 2015) e Runrun.it (RUNRUN.IT, 2015).

O ActiveCollab, ilustrado na Figura 9, destaca-se por permitir a criação de projetos e incluir itens a um projeto. Cada item possui um prazo de entrega, bem como pode ser definido como prioritário. O sistema oferece controle de tempo gasto em cada item por tipo de atividade, como documentação, desenvolvimento, etc.

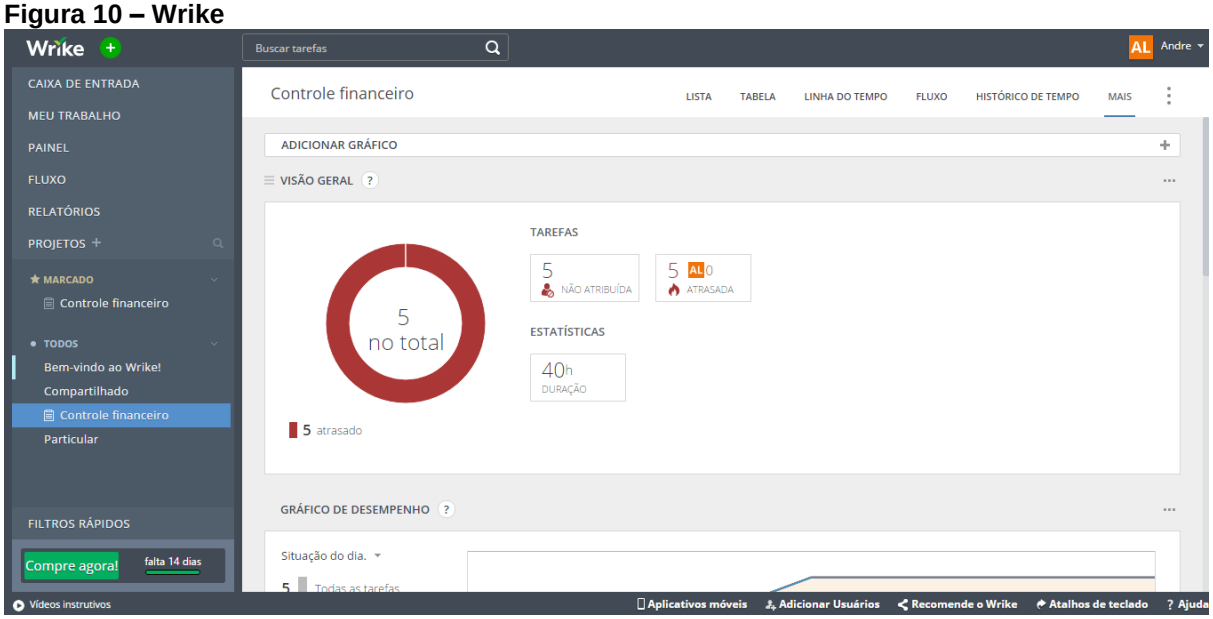
Figura 9 – ActiveCollab



Fonte: ACTIVECOLLAB (2016)

O Wrike, ilustrado na Figura 10, tem como destaque a possibilidade de definir dependências entre itens de um projeto, bem como determinar uma situação para projetos e itens, tais como: pendente ou em risco. O Wrike oferece registro automatizado de tempo gasto, sendo possível clicar em um botão para iniciar uma tarefa e, uma vez concluída, clicar novamente para que o tempo seja registrado. O principal fator de destaque no sistema é o uso de métricas automáticas e relatórios que oferecem uma visão geral dos projetos.

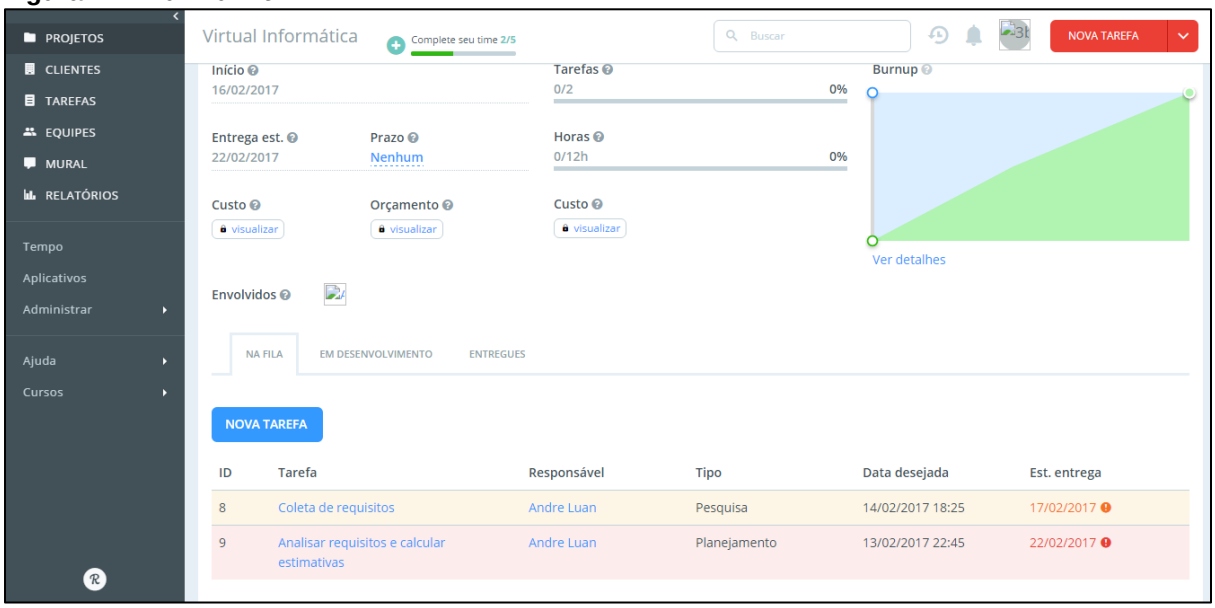
Figura 10 – Wrike



Fonte: WRIKE INC (2016)

O RunRun.it, exibido na Figura 11, oferece recursos similares ao ActiveCollab e Wrike, tais como: criação de projetos, criação de itens em projetos, registro automatizado de tempo, especificação de tipos de atividades e dependências, bem como a situação de projetos e itens. Seu diferencial é a forma inteligente com que trata o tempo, pois oferece suporte a estimativas por item e é capaz de priorizar automaticamente itens de acordo com o prazo restante e o tempo estimado.

Figura 11 - RunRun.it



Fonte: RUNRUN.IT (2016)

O Quadro 2 mostra a comparação entre os recursos oferecidos pelos três sistemas apresentados. Nota-se que o que possui maior número de recursos é o RunRun.it, portanto este será utilizado como o sistema de referência para este trabalho.

Quadro 2 – Comparação entre os sistemas existentes

Recurso	ActiveCollab	RunRun.it	Wrike
Criação de projetos	Sim	Sim	Sim
Vincular itens a projetos	Sim	Sim	Sim
Definição de prazos	Sim	Sim	Sim
Organização por tipo de atividade	Sim	Sim	Não
Registro de tempo gasto	Manual	Automático	Automático
Priorização de itens	Sim	Sim	Sim
Dependências entre itens	Não	Sim	Sim
Estimativas	Não	Sim	Não
Métricas automáticas	Não	Sim	Sim
Relatórios	Não	Sim	Sim
Situação por projeto	Não	Sim	Sim
Situação por item	Sim	Sim	Sim

Fonte: Autoria própria

3 METODOLOGIA

Este capítulo apresenta os processos executados na criação de um sistema para gestão de projetos, bem como a aplicação de uma implementação do algoritmo C4.5 no reconhecimento de padrões de comportamento de gestores do projeto.

3.1 ESCOLHA DA METODOLOGIA UTILIZADA

Após a análise das diferentes metodologias e abordagens utilizadas no processo de Descoberta de Conhecimento em Bases de Dados, optou-se pelo uso da técnica de Classificação por meio de Árvores de Decisão tendo como fonte de dados medições obtidas dos projetos por meio de cálculos automáticos. Para a geração de predições de ações tratativas nos projetos, optou-se por utilizar uma implementação em Javascript do algoritmo C4.5 disponibilizada como software livre (MOTA, 2017). A escolha da implementação em Javascript deu-se em função da arquitetura do sistema ser totalmente baseada em Javascript.

Pelo fato de tratar-se de um sistema web com requisitos provenientes da observação dos sistemas similares, optou-se pelo uso da Prototipação como metodologia de desenvolvimento, uma vez que ela permite a elaboração de um protótipo básico, não funcional, que vai tendo seus requisitos faltantes implementados até que o sistema esteja concluído.

As seções seguintes descrevem as tecnologias utilizadas para o desenvolvimento, bem como uma visão geral da Arquitetura do sistema desenvolvido.

3.2 TECNOLOGIAS UTILIZADAS

Para a interface do *front-end* (cliente da aplicação) foi utilizada a linguagem HTML5 em conjunto com CSS3, utilizando o framework visual Bootstrap, desenvolvido e mantido pelo Twitter (TWITTER, 2016). O framework foi usado por prover uma interface moderna e com maior compatibilidade com diferentes navegadores, bem como uma padronização visual na aplicação. A lógica da aplicação no *front-end* foi escrita utilizando a linguagem Javascript, com o

framework Angular (GOOGLE, 2016). A escolha do framework se deu principalmente por sua agilidade em tratar a comunicação cliente-servidor, sua fácil utilização, bem como sua confiabilidade, uma vez que este é desenvolvido e distribuído pela Google, uma das principais líderes em inovação tecnológica atualmente (GOOGLE, 2016).

A tecnologia escolhida para o *back-end* (servidor da aplicação) foi a Node (NODE.JS FOUNDATION, 2016), que utiliza o motor de Javascript do Google Chrome para executar a linguagem Javascript fora do navegador, bem como para possibilitar o uso da linguagem tanto no cliente como no servidor. O framework escolhido para o *back-end* foi o Express (NODE.JS FOUNDATION, 2016), que possibilita a rápida criação de APIs. A escolha do Express se deu por sua alta performance quando usada em servidores web e facilidade de interpretação de código-fonte. O framework Express também é desenvolvido e mantido pela mesma fundação que mantêm o Node.

O banco de dados eleito para a aplicação foi o MongoDB (MONGODB INC, 2016), um Sistema Gerenciador de Banco de Dados que utiliza o paradigma não-relacional, o qual é bastante difundido e explorado atualmente por conta de sua alta performance e flexibilidade no armazenamento de dados complexos. Sua escolha foi devido à grande quantidade de documentação disponível, bem como por sua fácil e rápida integração com a tecnologia Node, uma vez que armazena os dados em formato JSON, o formato padrão para representação de objetos na linguagem Javascript, bem como pela alta performance com grandes conjuntos de dados.

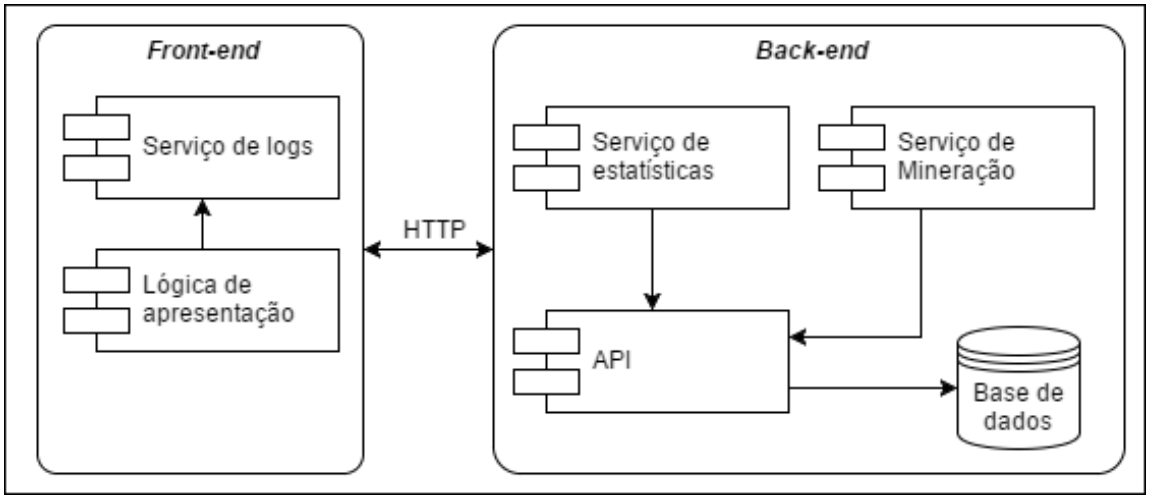
Após a escolha das tecnologias a serem utilizadas, o desenvolvimento da aplicação proposta se iniciou pela modelagem conceitual dos componentes do sistema, demonstrada na Figura 12 da seção seguinte. Após concluída essa modelagem, foi iniciado o desenvolvimento do sistema proposto, o qual deveria estar concluído e populado com dados de amostra para que somente então se pudesse iniciar o processo de Descoberta de Conhecimento na base de dados gerada.

3.3 ARQUITETURA DO SISTEMA

Por tratar-se de um sistema web, o modelo de arquitetura utilizado foi o Cliente-Servidor (SOMMERVILLE, 2007), em que o servidor é responsável por gerenciar os dados vindos do cliente e conectar-se à base de dados para retornar dados a este. O cliente responsabiliza-se apenas pela apresentação e validação das informações a serem enviadas ou recuperadas do servidor, além de executar rotinas específicas a seu ambiente de execução, como o serviço de logs. O sistema aqui proposto foi dividido em cinco partes fundamentais:

- ***Back-end*:** servidor da aplicação que fornece conexão com a base de dados, bem como a API a ser utilizada pelo *front-end*;
- ***Front-end*:** cliente da aplicação que fornece a interface a ser utilizada diretamente pelo usuário para manipulação dos métodos disponibilizados pela API do sistema;
- **Serviço de Logs:** atua no *front-end*, enviando informações ao *back-end* de forma independente da vontade do usuário, registrando determinadas ações realizadas por ele, tais como a alteração da prioridade de um projeto;
- **Serviço de Estatísticas:** atua no *back-end*, utilizando os dados dos projetos para geração de medições e estatísticas automáticas;
- **Serviço de Mineração:** utiliza os logs armazenados para analisar a estatística atual dos projetos e prever ações tratativas para problemas encontrados.

Figura 12 - Componentes do sistema



Fonte: Autoria própria

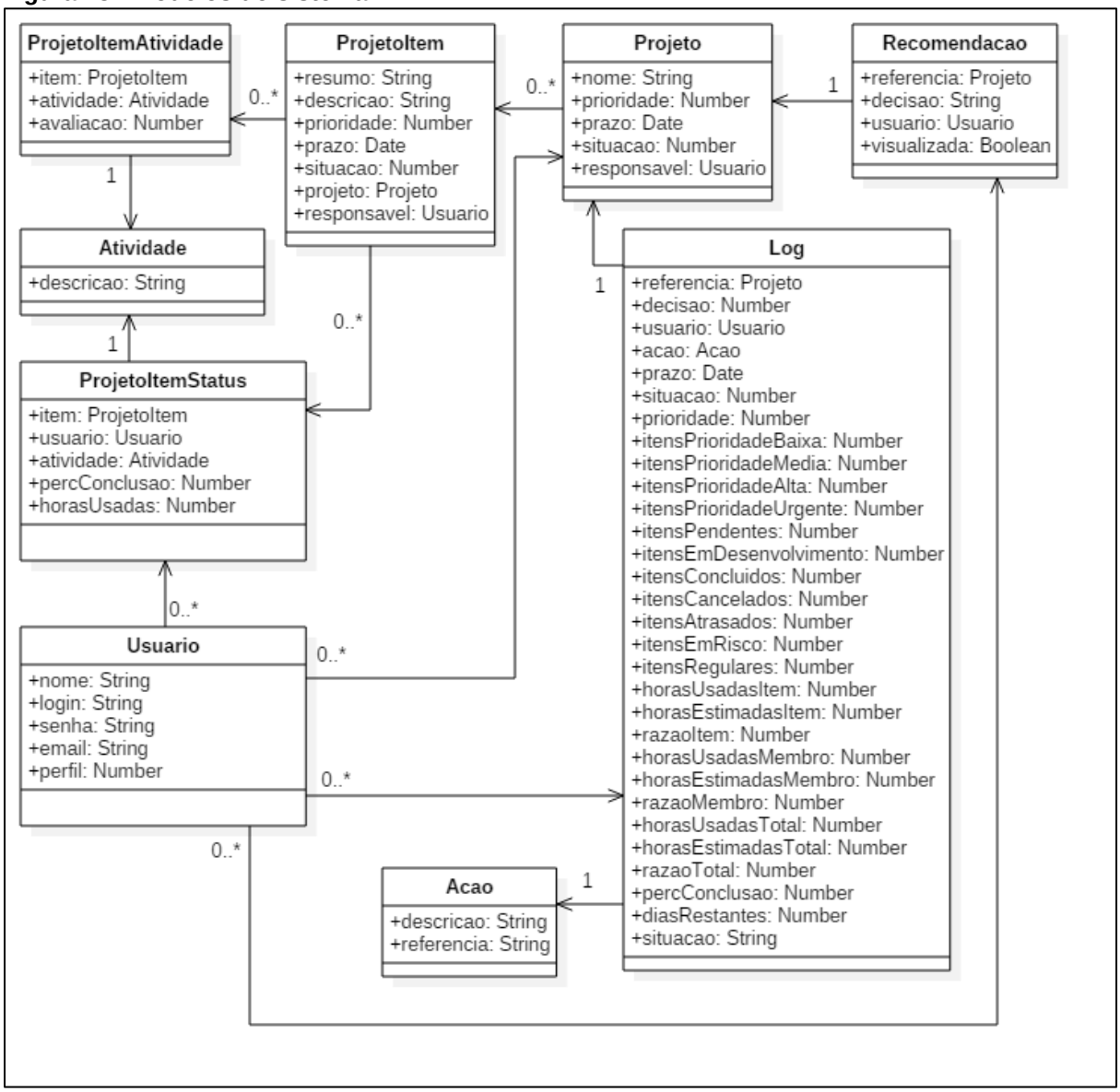
3.4 LEVANTAMENTO E UTILIZAÇÃO DOS REQUISITOS

Os requisitos do sistema proposto foram levantados a partir de uma análise dos requisitos presentes nos sistemas apresentados na introdução deste trabalho, sendo o Runrun.it (RUNRUN.IT, 2015) utilizado como sistema de referência para a determinação dos requisitos, pois mostrou-se o sistema mais completo, conforme demonstrado no Quadro 2. Pela análise do funcionamento de tais sistemas, concluiu-se que os requisitos mínimos necessários para um sistema de gestão de projetos são:

- **Controle de acesso:** visa identificar os usuários para separá-los de acordo com sua função, como gestor de projeto, membro da equipe ou administrador do sistema;
- **Controle de tipos de atividade:** tem por objetivo controlar os tipos de atividades a serem realizadas no projeto, tais como documentação, desenvolvimento, testes, etc.;
- **Controle de projetos:** deve ser possível controlar projetos, atribuí-los a um gestor e permitir a criação de itens ou tarefas em um projeto;
- **Controle de itens de projeto:** deve ser possível controlar itens (tarefas) de um projeto, atribuí-los a membros da equipe, permitir a definição de estimativas para cada tipo de atividade e acompanhamento de progresso de cada item.

Com base nos requisitos levantados, realizou-se a modelagem conceitual dos modelos a serem utilizados no desenvolvimento do sistema aqui proposto. A Figura 13 apresenta um diagrama de classes dos modelos do sistema aqui proposto pelo fato de até o momento ainda não haver uma notação padronizada para representação visual de bases NoSQL.

Figura 13 - Modelos do sistema



Fonte: Autoria própria

O modelo *Acao* representa uma decisão tomada no projeto. Ele contém a referência a um objeto do tipo Projeto, bem como um identificador textual da ação realizada. Os valores possíveis para o atributo descrição são: *prioridadeProjeto*, *deadlineProjeto*, *responsavelProjeto* e *situacaoProjeto*.

O modelo *Atividade* representa um tipo de atividade a ser realizada no projeto, tendo o atributo *descricao* como texto livre a ser informado pelo usuário em tela específica.

O modelo *Projeto* representa um projeto no sistema proposto. Para o atributo *prioridade* considera-se os valores genéricos: (0) baixa, (1) média, (2) alta e (3) urgente. Para o atributo *situacao*, considera-se os valores comumente

encontrados em sistemas de gestão de projetos de software, como o Runrun.it (RUNRUN.IT, 2015), por exemplo: (0) pendente, (1) em desenvolvimento, (2) encerrado e (3) cancelado.

O modelo *ProjetoItem* representa um item em um projeto. Para o atributo *prioridade* considera-se os valores: (0) baixa, (1) média, (2) alta, (3) urgente. Para o atributo *situacao*, considera-se os valores: (0) pendente, (1) em desenvolvimento, (2) encerrado e (3) cancelado.

O modelo *ProjetoItemAtividade* representa uma estimativa para um tipo de atividade em um item de projeto. O atributo *avaliacao* é utilizado para informar a quantidade de horas estimadas para a conclusão de um item.

O modelo *ProjetoItemStatus* representa uma atualização no item de projeto realizada pelo membro da equipe, contendo o percentual completo daquele item, bem como o número absoluto de horas gastas nessa atualização.

O modelo *Usuario* representa um usuário no sistema, juntamente com seu perfil de acesso. Os valores possíveis para o atributo *perfil* são: (1) administrador, (2) gestor de projetos e (3) membro de equipe.

O modelo *Recomendacao* representa uma ação tratativa a ser apresentada ao gestor em um determinado projeto. Ele contém a referência a um projeto e será exibida apenas ao usuário definido como responsável por esse projeto, apresentando uma sugestão textual de ação a ser realizada.

O modelo *Log* representa o registro de uma alteração realizada em um projeto por um gestor de acordo com as ações pré-definidas no modelo *Acao*. Além da alteração realizada também é armazenada uma medição da situação atual do projeto.

Por meio do método GQM (BASILI, CALDIERA, & ROMBACH, 2002), foram propostas as métricas a serem utilizadas no sistema proposto. Essas métricas têm por finalidade fornecer um panorama geral do projeto, do desempenho dos membros da equipe e do andamento do trabalho. A principal pergunta a ser respondida por essas métricas é a respeito da necessidade de dar prioridade maior a um projeto, alterar seu prazo de entrega ou mesmo cancelar o projeto.

Neste trabalho as medições são realizadas de forma automatizada pelo serviço de estatísticas apresentado na próxima seção.

3.5 IMPLEMENTAÇÃO

Inicialmente o foco do desenvolvimento foi direcionado à implementação dos requisitos essenciais de um sistema de gestão de projetos apresentados na seção 3.4, tais como controle de usuários, controle de projetos e itens de projeto, bem como a vinculação dos responsáveis (membros da equipe), gestão de tempo e percentual de conclusão de cada item.

O desenvolvimento iniciou-se com a elaboração de um protótipo não funcional utilizando-se apenas o *framework* CSS Bootstrap. Esse protótipo teve como finalidade a elaboração do *layout* inicial do sistema, bem como a determinação das telas onde as informações seriam gerenciadas pelo usuário.

Após a avaliação do protótipo inicial, iniciou-se a criação do servidor básico em Node com o *framework* Express para que os arquivos HTML gerados no protótipo pudessem ser servidos por uma porta na rede local conforme ilustrado na Figura 14.

Figura 14 - Serviço de arquivos do front-end

```
var express = require('express');
var path = require('path');

var app = express();

app.use(express.static(path.join(__dirname, 'public')));
```

Fonte: Autoria própria

A função *require* realiza a importação de um módulo no Node, neste caso são importados o módulo nativo denominado *path*, que tem a finalidade de facilitar o tratamento de caminhos no sistema de arquivos. Também é importado o módulo do *framework* Express, o qual é instanciado através da função *express,* a qual cria um servidor de aplicação. O módulo *path* é utilizado para se obter o caminho do diretório onde a aplicação está sendo executada e combiná-la com seu subdiretório *public*, onde são armazenados os arquivos do *front-end*. Esse caminho é utilizado pelo serviço de arquivos estáticos do Express para que o *front-end* torne-se acessível publicamente.

Com a base do *front-end* e *back-end* concluídos, iniciou-se a implementação dos serviços no *back-end* a serem utilizados pelo *front-end* para manipulação de dados, tais como: usuários, projetos, tipos de atividades, etc.

Cada serviço consiste em um módulo que especifica rotas ou URLs servidas pelo *back-end.* Serviços que fazem uso de conexão com a base de dados podem utilizar um ou mais dos modelos apresentados na Figura 13. Um modelo tem a estrutura básica apresentada na Figura 15.

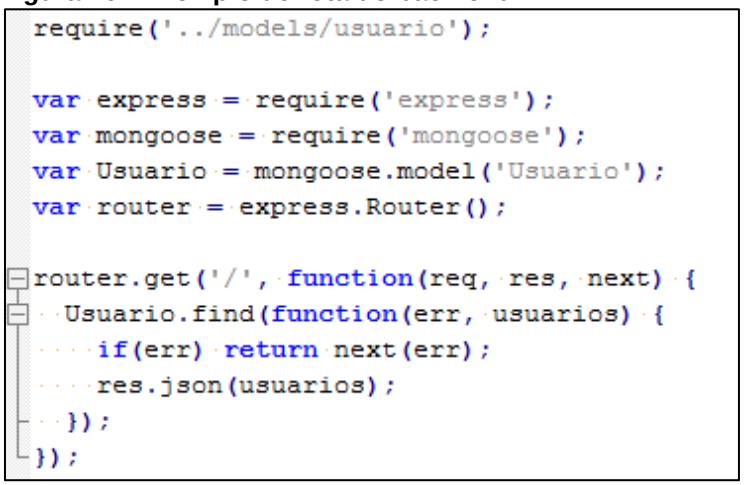
Figura 15 - Exemplo de modelo do back-end

```
var mongoose = require('mongoose');

var UsuarioSchema = new mongoose.Schema({
    nome: String,
    login: String,
    senha: String,
    email: String,
    perfil: {type: Number, default: 0}
});

mongoose.model('Usuario', UsuarioSchema);
```

Fonte: Autoria própria

A Figura 16 apresenta um exemplo de uma rota em um serviço no *back-end* que consulta todos os usuários existentes na base de dados, representados pelo modelo *Usuario*, e os retorna ao *front-end* em formato JSON. O módulo Mongoose provê um conjunto de utilitários que fornecem uma camada de modelo que elimina o uso de consultar diretas à base de dados, bem como provê uma forma de definir esquemas dos dados armazenados na base através de sua classe *Schema*. Cada esquema é registrado globalmente no módulo pela função *model* e pode ser reutilizado em outras partes do código-fonte. Cada serviço é mapeado a um padrão de URL através da classe *Router* do Express e pode responder a um dos tipos padrão de requisição HTTP, tais como: GET, POST, DELETE, etc.

Figura 16 - Exemplo de rota do back-end

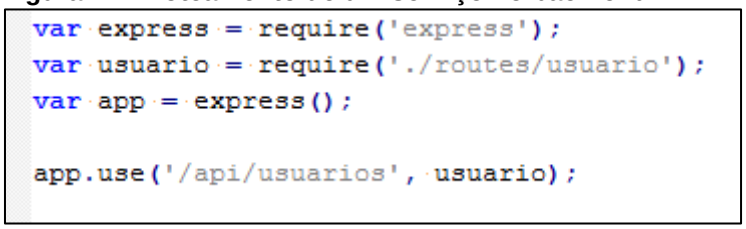


```
require('../models/usuario');

var express = require('express');
var mongoose = require('mongoose');
var Usuario = mongoose.model('Usuario');
var router = express.Router();

router.get('/', function(req, res, next) {
  Usuario.find(function(err, usuarios) {
    if(err) return next(err);
    res.json(usuarios);
  });
});
```

Fonte: Autoria própria

Como mencionado anteriormente, um modelo registrado pelo Mongoose provê uma camada de abstração para consultar à base de dados. Para consultar todos os usuários na base, utiliza-se o método *find,* que retorna a lista de usuários, a qual é devolvida ao cliente em formato JSON através do método *json* do objeto que representa a resposta da requisição tratada pela rota. Cada serviço é adicionado à base do *back-end* e atribuído a um determinado padrão de URL por meio do método *use* do Express. Com isso, o serviço passa a ser acessível ao *front-end.* No exemplo mostrado na Figura 17, é possível acessar o serviço pela da URL <*http://localhost/api/usuarios*>.

Figura 17 - Roteamento de um serviço no back-end



```
var express = require('express');
var usuario = require('./routes/usuario');
var app = express();

app.use('/api/usuarios', usuario);
```

Fonte: Autoria própria

Os serviços são acessados no *front-end* com o *framework* Angular, por meio de chamada HTTP, conforme demonstrado na Figura 18. Com os dados carregados no cliente é possível realizar sua exibição e controle nos documentos HTML.

Figura 18 - Chamada a um serviço no front-end

```
$http.get('./api/usuarios').then(function(usuarios) {
    $scope.usuarios = usuarios.data;
    delete $scope.registro;
});
```

Fonte: Autoria própria

Uma vez que os serviços utilizados pelos requisitos básicos estavam concluídos, tais como: controle de usuários, projetos, tipos de atividade, registro de tempo e estimativas, deu-se início ao desenvolvimento do serviço de medições e estatísticas. Sua implementação seguiu o mesmo padrão aqui demonstrado, porém sua finalidade é diferente. As medições geradas fornecem um panorama sobre o projeto, os itens e a equipe.

O serviço de medições também é utilizado na geração de gráficos na tela inicial, apresentada ao gestor e ilustrada na Figura 25, para que este tenha uma visão geral do andamento de cada projeto. Para a geração das medições são consideradas as seguintes fórmulas:

- **Itens com prioridade baixa:** contagem simples;
- **Itens com prioridade média:** contagem simples;
- **Itens com prioridade alta:** contagem simples;
- **Itens com prioridade urgente:** contagem simples;
- **Itens pendentes:** contagem simples;
- **Itens em desenvolvimento:** contagem simples;
- **Itens concluídos:** contagem simples;
- **Itens cancelados:** contagem simples;
- **Itens atrasados:** contagem simples;
- **Itens em risco:** contagem simples;
- **Itens em situação regular:** contagem simples;
- **Horas usadas por item:** soma das horas usadas no projeto dividida pelo total de itens no projeto;
- **Horas estimadas por item:** soma das horas estimadas no projeto dividida pelo total de itens no projeto;
- **Horas usadas x estimadas por item:** razão entre os dois anteriores multiplicada por cem;

- **Horas usadas por membro:** soma das horas usadas no projeto dividida pelo total de membros do projeto;
- **Horas usadas por membro:** soma das horas estimadas no projeto dividida pelo total de membros do projeto;
- **Horas usadas x estimadas por membro:** razão entre os dois anteriores multiplicada por cem;
- **Dias restantes:** diferença de dias entre o prazo do projeto e a data atual;
- **Percentual concluído:** número de itens concluídos ou cancelados dividido pelo número total de itens do projeto. O resultado é multiplicado por cem;
- **Horas usadas totais:** soma simples;
- **Horas estimadas totais:** soma simples;
- **Horas usadas x estimadas totais:** razão entre os dois anteriores multiplicada por cem; e
- **Situação:** caso o prazo final do projeto tenha passado, é igual a *atrasado*, caso restem três dias para o final do prazo e menos de 90% dos itens estejam concluídos, é igual a *em risco*, caso contrário é igual a *normal*.

O serviço de medições pode ser invocado a qualquer dado momento, gerando as medições por demanda. A Figura 19 apresenta um exemplo de saída do serviço de medições:

Figura 19 - Saída do serviço de medições


```
{
    "_id": "588cd99028e6a81831d25252",
    "nome": "Clínica Estética",
    "deadline": "2017-02-28T03:00:00.000Z",
    "status": 1,
    "prioridade": 1,
    "itensPriorBaixa": 1,
    "itensPriorMedia": 6,
    "itensPriorAlta": 4,
    "itensPriorUrgente": 0,
    "itensSitPendente": 0,
    "itensSitDesenvolv": 1,
    "itensSitEncerrado": 10,
    "itensSitCancelado": 0,
    "itensAtrasados": 1,
    "itensRisco": 0,
    "itensNormais": 10,
    "horasUsadasPorItem": 8.727272727272727,
    "horasEstimadasPorItem": 6.545454545454546,
    "razaoEstimativaPorItem": 133,
    "horasUsadasPorMembro": 96,
    "horasEstimadasPorMembro": 72,
    "razaoEstimativaPorMembro": 133,
    "diasRestantes": 31,
    "percConcluido": 90.9090909090909,
    "horasUsadas": 96,
    "horasEstimadas": 72,
    "razaoEstimativa": 133,
    "situacao": "N"
}
```


Fonte: Autoria própria

Uma vez concluído o desenvolvimento dos requisitos fundamentais do sistema de gestão de projetos e do serviço de medições automatizadas, foi iniciado o desenvolvimento de um sistema de *logs* para registrar determinadas ações realizadas na tela de projetos do sistema. Todos os dados persistentes do sistema são armazenados em coleções no MongoDB, manipulados e transmitidos entre o cliente e o servidor no formato JSON, uma vez que este é o formato padrão da linguagem Javascript para representação de objetos. A Figura 20 ilustra o exemplo de um registro no log de ações de usuários.

Figura 20 - Registro no log de ações


```
{
    "acao": "prioridadeProjeto",
    "decisao": 2,
    "usuario": "56f9abb26a54a5141740f77e",
    "deadline": "2017-02-28T03:00:00.000Z",
    "status": 1,
    "prioridade": 2,
    "itensPriorBaixa": 1,
    "itensPriorMedia": 5,
    "itensPriorAlta": 4,
    "itensPriorUrgente": 0,
    "itensSitPendente": 4,
    "itensSitDesenvolv": 1,
    "itensSitEncerrado": 5,
    "itensSitCancelado": 0,
    "itensAtrasados": 0,
    "itensRisco": 0,
    "itensNormais": 10,
    "horasUsadasPorItem": 4.2,
    "horasEstimadasPorItem": 7.9,
    "razaoEstimativaPorItem": 0,
    "horasUsadasPorMembro": 42,
    "horasEstimadasPorMembro": 79,
    "razaoEstimativaPorMembro": 0,
    "percConcluido": 50,
    "horasUsadas": 42,
    "horasEstimadas": 79,
    "razaoEstimativa": 53,
    "diasRestantes": 21,
    "situacao": "N"
},
```


Fonte: Autoria própria

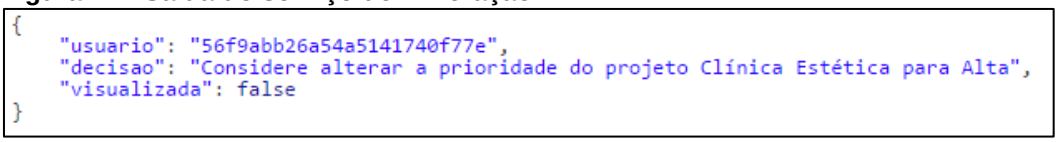
No *log* é armazenado o tipo de modificação realizada pelo usuário (campo *decisao*), como alterar a prioridade do projeto, por exemplo. Além disso, também é armazenado o valor escolhido pelo usuário para a modificação, bem como o identificador único (ID) daquele usuário. No momento da geração do *log*, é realizada uma chamada ao serviço de medições para a coleta das estatísticas daquele projeto e o objeto resultante dessa chamada é mesclado com o objeto do log. Uma particularidade do MongoDB é que este utiliza cifras codificadas em MD5 para representar identificadores únicos, como o valor do atributo *usuario*.

No caso da ação ser uma alteração de *deadline*, o valor passa a conter a diferença de dias entre a data previamente armazenada e a nova data escolhida. Caso ocorra alteração na prioridade do projeto, o valor passa a conter o novo valor da prioridade, sendo 1 o valor de menor prioridade e 4 o maior.

Para o desenvolvimento do serviço de Mineração de Dados foi utilizada uma implementação de código aberto do algoritmo C4.5 (MOTA, 2017) para realizar a

classificação das informações constantes no *log*. A saída do serviço de Mineração é ilustrada na Figura 21.

Figura 21 - Saída do serviço de Mineração



Fonte: Autoria própria

O processo de Descoberta de Conhecimento em Bases de Dados ocorre em uma série de etapas no serviço de Mineração implementado. Para que o serviço seja ativado é necessária a realização de uma chamada HTTP manual. Essa chamada pode ser agendada por meio de ferramentas do sistema operacional para ser realizada diariamente, por exemplo.

Na primeira etapa, obtenção dos dados, todos os *logs* do usuário são recuperados da base de dados por meio de uma chamada ao serviço de estatísticas. Em seguida, são capturadas as medições atualizadas dos projetos atribuídos a esse mesmo usuário para compor a base de dados que será usada na mineração.

De posse dos dados, os logs são separados de acordo com seu atributo *acao*, de forma que as ações tratativas são geradas separadamente para cada tipo de ação, exemplo: alteração de prazo ou prioridade do projeto.

O serviço de Mineração gera Árvores de Dados específicas para cada tipo de ação. Os logs referentes a um determinado tipo de ação são separados em um subconjunto e então submetido para ser usado no treinamento de uma instância do algoritmo C4.5.

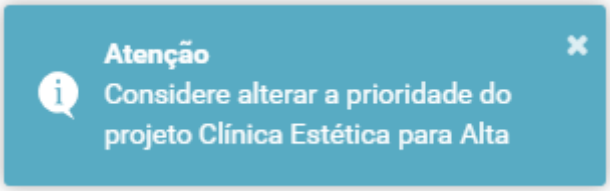
No processo de pré-processamento dos conjuntos de dados, algumas informações constantes no *log* que não foram consideradas relevantes para a geração das recomendações, tais como o identificador codificado ou nome do projeto, foram removidas para evitar ruídos potencialmente negativos no treinamento do algoritmo. Uma vez que os dados estejam pré-processados, eles são submetidos à Classificação pelo algoritmo C4.5, usando o atributo *decisao* como a classe avaliada. O algoritmo recebe o conjunto de dados de treinamento, a classe a ser avaliada e uma lista com o nome dos atributos como parâmetro em seu construtor. Uma vez inicializado, o algoritmo pode predizer a classe *decisao* de um

objeto do tipo *Log*. Desta forma, o algoritmo avalia cada registro e determina quais atributos foram os determinantes para a decisão do usuário.

Com a árvore treinada, o objeto contendo as medições de um projeto é submetido a ela para prever o valor do atributo *decisao* para o cenário atual medido. O valor retornado pela Árvore de Decisão é convertido em texto legível, conforme demonstrado na Figura 21. O serviço de Mineração possui ativação manual, podendo ser invocado a qualquer dado momento. Essa decisão teve como base o fato de que equipes diferentes podem utilizar processos diferentes, baseados em marcos semanais ou mensais. Dessa forma, fica a critério da equipe a configuração da periodicidade da execução do serviço.

Para evitar acumulo de informações, cada ação gerada só é apresentada uma vez. Esse controle é realizado por meio do atributo *visualizada* da modelo *Recomendacao* e é utilizado para evitar que um gestor veja uma mesma recomendação toda vez que utilizar o sistema aqui desenvolvido. Finalmente, o serviço retorna uma potencial decisão de acordo com a situação atual do projeto baseado em seu treinamento feito com as decisões anteriores do usuário em questão mostrada na Figura 22.

Figura 22 - Exemplo de recomendação gerada



Fonte: Autoria própria

4 RESULTADOS

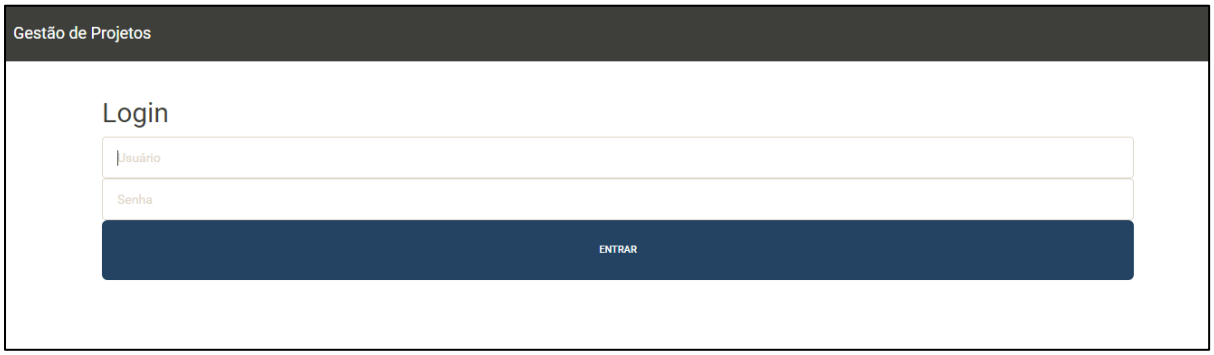
Neste capítulo serão apresentados os procedimentos e dados utilizados na validação do sistema desenvolvido neste trabalho, bem como a avaliação dos resultados obtidos com sua implementação.

4.1 SISTEMA DESENVOLVIDO

O controle de permissão de acesso do sistema desenvolvido neste trabalho foi baseado em perfis pré-determinados com o objetivo de melhor separar os usuários em grupos distintos, de forma que apenas gestores de projeto possam alterar qualquer informação do projeto. Quanto aos membros da equipe, eles podem apenas visualizar e atualizar itens que ainda não foram concluídos ou cancelados e que estejam atribuídos diretamente a eles. Também foi adicionado um perfil de administrador do sistema que tem permissões absolutas para realizar qualquer alteração em itens, projetos e membros da equipe.

Para a utilização do sistema de gestão de projetos é necessário prover autenticação via usuário e senha que estejam previamente cadastrados pelo administrador do sistema, o qual é o único a possuir a autoridade para cadastrar novos usuários no sistema. A tela inicial da aplicação apresenta um formulário para autenticação e é ilustrado na Figura 23.

Figura 23 - Autenticação de usuário

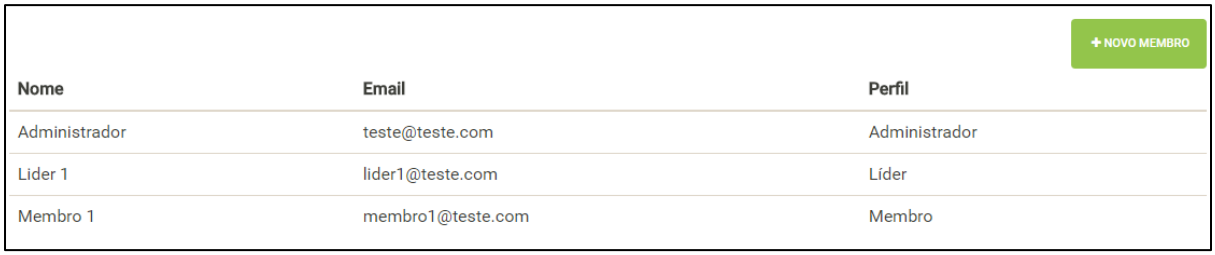


Fonte: Autoria própria

Uma vez autenticado e, caso possua permissão para tal, o usuário pode acessar a tela de gestão de usuários, em que são cadastrados e geridos os

usuários do sistema, bem como definidos seus perfis. O sistema possui três perfis fixos de permissões, sendo eles: administrador do sistema, líder de projeto e membro da equipe, conforme ilustrado na Figura 24.

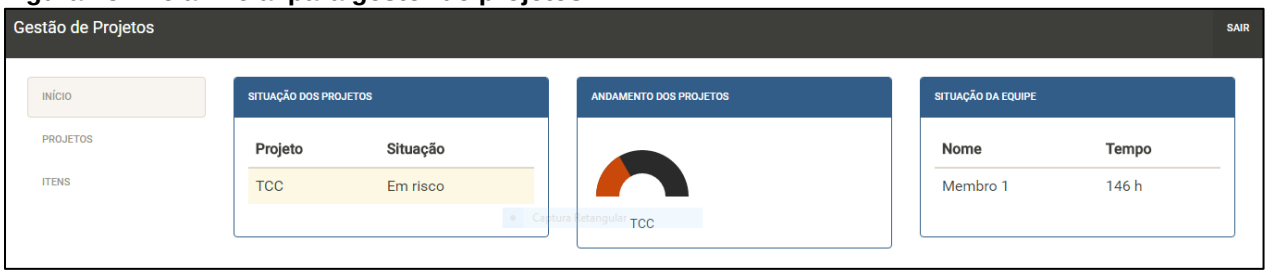
Figura 24 - Tela de gestão de usuários



Fonte: Autoria própria

Cada usuário tem acesso somente às funcionalidades permitidas pelo perfil ao qual foi vinculado. Caso o usuário seja gestor de um projeto, ele irá visualizar uma página inicial personalizada com informações relevantes sobre os projetos que estiver gerindo, sendo elas: a situação avaliada para cada projeto, um gráfico exibindo o percentual de conclusão e as horas trabalhadas por cada membro da equipe, como demonstrado na Figura 25.

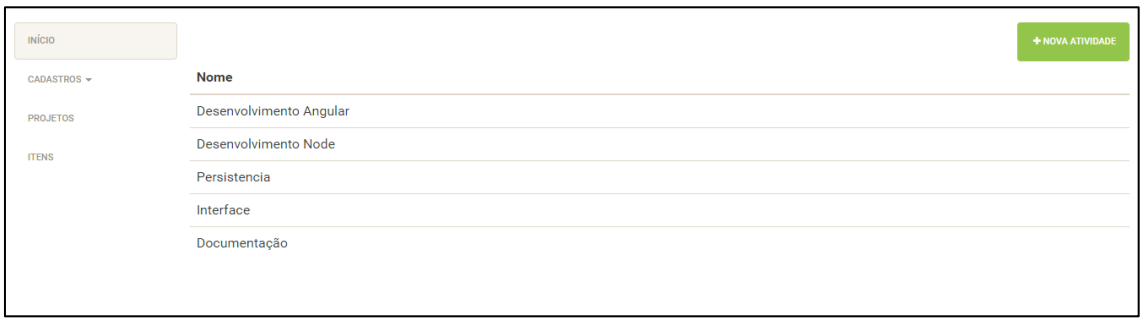
Figura 25 - Tela inicial para gestor de projetos



Fonte: autoria própria

Caso o usuário possua perfil de administrador do sistema, este poderá acessar a opção responsável por gerir usuários, mostrada na Figura 24, e gerenciar tipos de atividades, como por exemplo: “Análise de Requisitos” ou “Desenvolvimento HTML”, mostrada na Figura 26.

Figura 26- Tela para cadastro de atividades

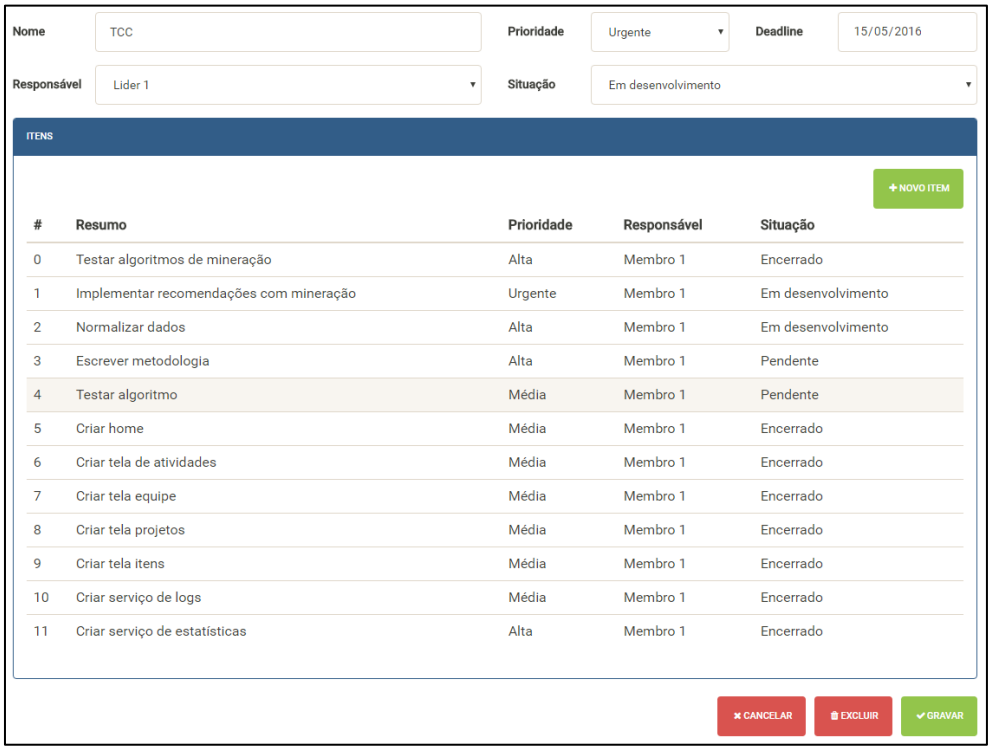


Fonte: Autoria própria

Finalmente, caso o usuário possua perfil de membro da equipe, ele não terá acesso ao painel de gestão de projetos, podendo acessar os itens do projeto atribuídos a ele por uma tela específica para usuário com esse perfil.

Pelo do painel de projetos, demonstrado na Figura 27, o gestor é capaz de incluir um novo projeto no sistema, bem como consultar e alterar projetos existentes, podendo inclusive excluir um projeto. A tela também permite a inclusão e gestão de itens dentro de um determinado projeto, bem como suas estimativas separadas por cada tipo de atividade e o acompanhamento da conclusão de cada item.

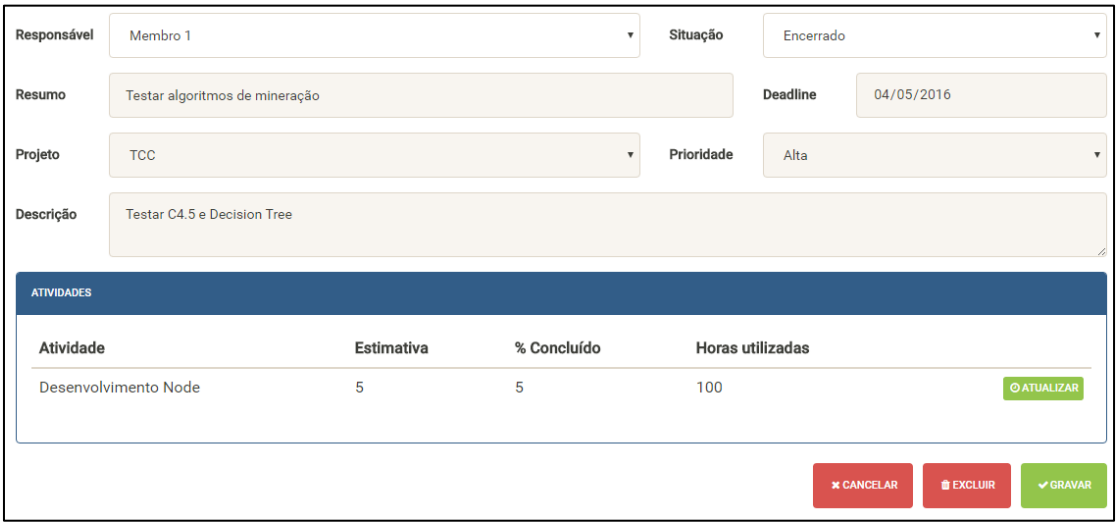
Figura 27 - Detalhes de um projeto



Fonte: Autoria própria

Cada membro da equipe que possua itens atribuídos a si poderá visualizar e atualizar a situação desses itens em uma tela específica, conforme Figura 28. Nessa tela, é possível incluir o registro de horas utilizadas para cada atividade estimada pelo gestor, bem como especificar o percentual de conclusão daquele item.

Figura 28 - Item de projeto para membro da equipe



Fonte: Autoria própria

Cada projeto possui um *deadline*, ou seja, um prazo máximo para ser concluído estabelecido e mantido pelo gestor do projeto. Essa informação é utilizada pelo serviço de estatísticas para gerar as medições automatizadas dos projetos. Além do *deadline*, também são considerados: quantidade de itens concluídos, percentual de conclusão total do projeto e horas estimadas restantes.

4.2 CENÁRIOS DOS EXPERIMENTOS

Com o sistema de gestão e o serviço de Mineração concluídos, foi realizado um teste de validação das recomendações geradas com o objetivo de verificar se, após uma quantidade mínima de treinamento, o sistema capaz de gerar uma ação tratativa que auxiliasse no sucesso do projeto.

Para tal finalidade, foram criados dois projetos fictícios (exemplificados no Quadro 3), contendo itens, estimativas e realizado o correto registro de tempo gasto.

Os projetos representam sistemas distintos em seus requisitos, porém o projeto “Clínica estética” depende do projeto “Controle financeiro”. Cada item do projeto foi estimado de acordo com o tipo de atividade necessária, como “documentação” ou “desenvolvimento de interface”, por exemplo.

Quadro 3 – Projetos para validação do sistema

Projeto	Prazo	Prioridade
Controle financeiro	20/02/2017	Média
Clínica estética	20/02/2017	Média

Fonte: Autoria própria

O Quadro 4 mostra os itens do projeto “Controle financeiro” com seus prazos, prioridades e dependências entre os itens.

Quadro 4 – Itens do projeto 1 - “Controle financeiro”

Item	Prazo	Dependência	Prioridade
1- Coleta de requisitos	30/01/2017	Nenhuma	Alta
2- Analisar requisitos e calcular estimativas	01/02/2017	1	Alta
3- Controle de bancos	03/02/2017	2	Média
4- Controle de contas bancárias	04/02/2017	3	Média
5- Controle de pessoas	06/02/2017	2	Média
6- Controle de títulos	10/02/2017	5	Alta
7- Controle de usuários	12/02/2017	2	Média
8- Controle de caixa	15/02/2017	6	Alta
9- Controle de pagamentos	17/02/2017	8	Média
10- Relatório de balanço	20/02/2017	9	Baixa

Fonte: Autoria própria

O Quadro 5 mostra os itens do projeto “Clínica estética” com seus prazos, prioridades e dependências entre os itens.

Quadro 5 – Itens do projeto 2 - “Clínica estética”

Item	Prazo	Dependência	Prioridade
1- Coleta de requisitos	30/01/2017	Nenhuma	Alta
2- Analisar requisitos e calcular estimativas	01/02/2017	1	Alta
3- Controle de usuários	04/02/2017	2	Média
4- Controle de clientes	05/02/2017	2	Média
5- Controle de profissionais	07/02/2017	2	Média
6- Controle de tratamentos	08/02/2017	2	Média
7- Controle de pacotes	10/02/2017	6	Baixa
8- Controle de convênios	12/02/2017	4	Média
9- Controle de espaços	15/02/2017	2	Alta
10- Controle de agenda	18/02/2017	4, 5, 7, 9	Alta
11- Integração com controle financeiro	20/02/2017	Projeto 1	Média

Fonte: Autoria própria

Para fins de simulação, considerou-se que ambos os projetos pertenciam a um mesmo cliente e deveriam ser entregues juntos. Os projetos formariam um sistema único com a finalidade de gerir uma clínica estética, provendo controle financeiro e operacional do negócio. Entretanto, optou-se por agrupar as diferentes regras de negócio em projetos separados. Sendo assim, o item 11 do projeto 2 tem como dependência o projeto 1 completo.

4.3 EXPERIMENTOS

Foi realizada uma simulação do andamento dos projetos ao longo da duração de seu prazo estabelecido e o tempo gasto em cada tipo de atividade para cada item foi registrado, bem como o percentual de conclusão de cada item. Com a introdução proposital de atrasos nos itens dos projetos, a prioridade de cada item foi revisada e aumentada em um nível de forma que itens que eram de prioridade baixa passaram a ser de prioridade média e assim por diante. Similarmente, a prioridade de ambos os projetos foi alterada de forma que o projeto que estivesse em situação mais crítica tivesse maior prioridade.

O Quadro 6 mostra as decisões tomadas pelo gestor que foram registradas nos *logs* da aplicação, bem como o motivo de tais decisões pelo ponto de vista do gestor.

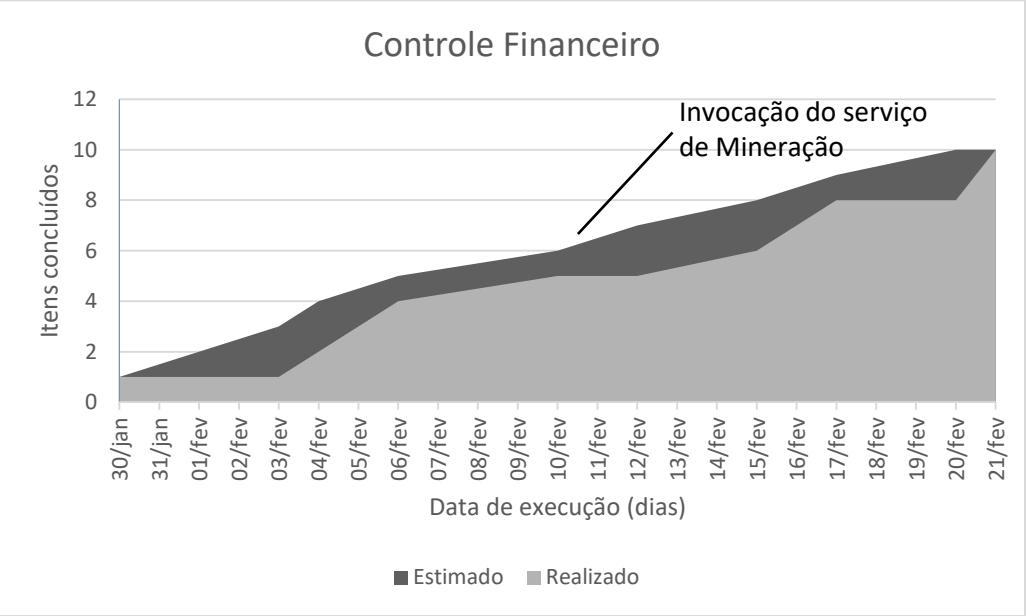
Quadro 6 – Registro de logs dos projetos

Evento	Data	Projeto	Decisão	Motivo
1	03/02	1- Controle financeiro	Prioridade alta	Usadas x estimadas totais em 53% com 18 dias restantes
2	07/02	1- Controle financeiro	Prioridade média	Usadas x estimadas totais em 74% com 16 dias restantes
3	17/02	1- Controle financeiro	Prioridade urgente	Projeto em risco (avaliado pelo serviço de medições)
4	20/02	1- Controle financeiro	Aumentar o prazo final em 1 dia	Projeto atrasado com itens não concluídos
5	10/02	2- Clínica estética	Prioridade alta	11 dias restantes e 4 itens pendentes

Fonte: Autoria própria

Quando o projeto 1 estava com pouco esforço sendo realizado, decidiu-se por aumentar sua prioridade para que fossem cumpridas as horas estimadas totais do projeto. Quando as horas estimadas começaram a ser atingidas, a prioridade do projeto foi baixada novamente. Quando o serviço de medições alertou que o projeto 1 estava em situação de risco, sua prioridade foi aumentada novamente, desta vez em regime de urgência. Mesmo com o aumento do esforço no projeto, ele não foi concluído no prazo, portanto, decidiu-se aumentar o prazo final em um dia com a finalidade de simular uma renegociação de prazo com um cliente. A performance do trabalho realizado no projeto “Controle financeiro” em relação ao trabalho estimado é ilustrado na Figura 29. Os eventos 1, 2, 3 e 4 destacados representam mudanças no planejamento do projeto registradas conforme registradas no log apresentado no Quadro 6.

Figura 29 - Performance do projeto “Controle financeiro”



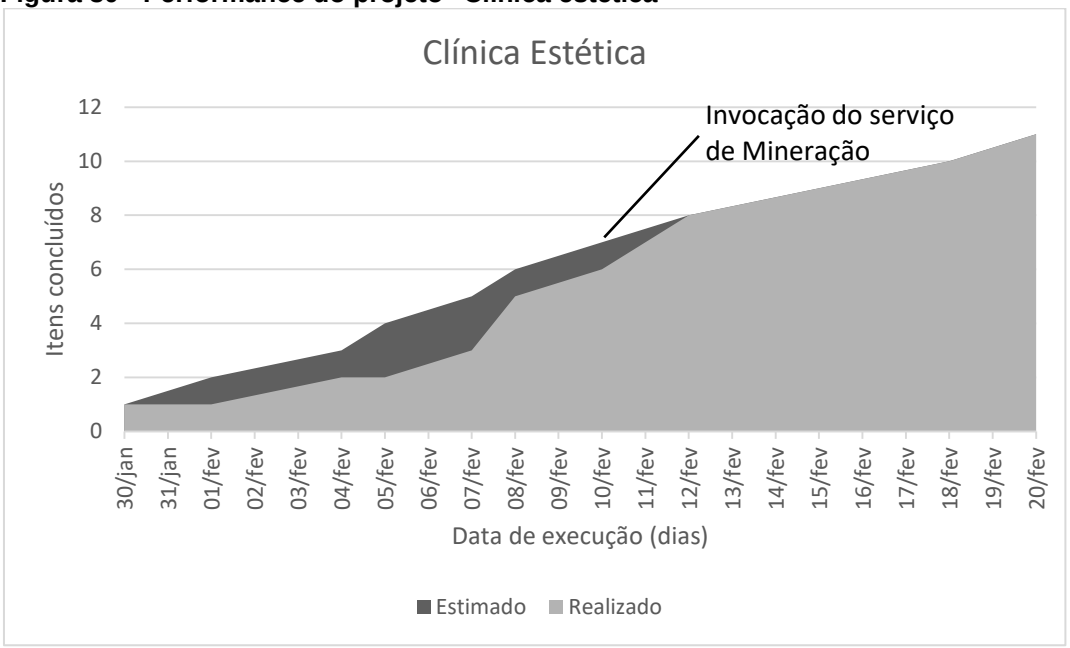
Fonte: Autoria própria

Com o esforço sendo priorizado para o projeto 1, o projeto 2 passou a deixar muitos itens pendentes com relação ao prazo final. O serviço de Mineração foi ativado e recomendou que se alterasse a prioridade do projeto 2 para *alta*. Esta recomendação foi considerada útil, pois foi baseada no tratamento dado ao gestor para o projeto 1 quando este começou a apresentar baixo rendimento.

Para o projeto 2, realizou-se uma simulação dos mesmos eventos ocorridos no projeto 1 até o momento em que o fim do prazo do projeto estava próximo. O sistema gerou uma recomendação para alterar a situação do projeto para *urgente*. A eficiência da recomendação gerada foi relevante pois, novamente, foi baseada no tratamento dado ao projeto 1 quando este estava em situação de risco.

A performance do projeto “Clínica estética” é ilustrado na Figura 30. O evento 5 representa a mudança no planejamento registrada no log apresentado no Quadro 6. Essa mudança foi realizada seguindo a recomendação gerada pelo serviço de Mineração.

Figura 30 - Performance do projeto "Clínica estética"



Fonte: autoria própria

Por meio da comparação da performance dos dois projetos utilizados como cenários de teste nota-se que no primeiro caso, onde não houve acionamento do sistema de Mineração para recomendar ações tratativas, o projeto teve seu desempenho normalizado apenas perto de sua conclusão. No cenário onde houve o acionamento do serviço de Mineração para o projeto “Clínica estética”, o desempenho do trabalho voltou ao estimado com maior antecedência em relação ao projeto “Controle Financeiro”.

5 CONCLUSÃO

Verificou-se que o uso de técnicas de Mineração de Dados na área de Gestão de Projetos tem o potencial de auxiliar gestores de projetos em sua tomada de decisão avaliando a situação de projetos em andamento e verificando cenários semelhantes encontrados por um gestor, oferecendo a ele recomendações de ações com base em suas decisões anteriores. Com isso é possível antecipar situações críticas que, caso fossem identificadas tardiamente, poderiam gerar prejuízo ao projeto aumentando, por exemplo, o custo de horas trabalhadas e reduzindo o orçamento final, bem como atrasando o prazo de conclusão.

O uso de métricas no monitoramento de projetos (NICHOLAS & STEYN, 2012) mostrou-se benéfico no acompanhamento do desempenho de projetos de software. O uso da metodologia *Goal-Question-Metric* na definição de tais métricas auxiliou no levantamento das questões que determinam quando determinado projeto necessita de maior prioridade ou de aumento em seu prazo. O uso do algoritmo C4.5 sobre o registro de ações do gestor nos projetos mostrou-se uma solução eficaz na tarefa de prever as decisões que tal gestor poderia tomar, pois os dados providos para treinar o algoritmo são as decisões tomadas pelo próprio gestor para cada cenário anteriormente encontrado por ele em outros projetos.

Também foi verificado que a qualidade das recomendações tende a aumentar conforme o sistema é utilizado. Conforme cada novo projeto é gerido pelo sistema desenvolvido neste trabalho, o registro de logs aumenta o que oferece ao algoritmo C4.5 uma amostra maior de informações, aumentando a qualidade de seu treinamento.

Observou-se que as recomendações geradas quando o algoritmo tinha poucas instâncias para treinamento refletiram apenas as últimas decisões tomadas devido à falta de variedade no conjunto de dados, enquanto que aquelas geradas com um número maior de instâncias passaram a possuir mais significância e a fazer mais sentido do ponto de vista de um gestor de projetos.

Conclui-se também que embora a quantidade de métricas e estatísticas utilizadas na análise automatizada da situação do projeto tenha sido suficiente para demonstrar a eficiência do sistema aqui apresentado, torna-se necessária a

implementação de outros tipos de medições para que o sistema tenha significância real se usado em ambiente profissional.

Como indicação de trabalho futuro, recomenda-se a expansão do sistema de geração de recomendações para que este também contemple a emissão de recomendações para cada item em um projeto e para cada membro da equipe.

Como trabalho futuro também pode-se simplificar o processo de registro de horas trabalhadas pela equipe, tornando o cálculo automatizado por meio da criação de um cronômetro na aplicação, de forma que os membros da equipe tenham apenas de inicia-lo ou pará-lo, não tendo que calcular por conta própria as horas trabalhadas.

Outra recomendação é a melhoria na interface visual da aplicação, preferencialmente utilizando-se dos paradigmas do *Material Design*, criado e utilizado pela Google em suas aplicações web e móveis, uma vez que esses paradigmas propiciam maior acessibilidade e facilidade de entendimento e de interação do usuário com a aplicação.

Também se propõe a utilização das metodologias descritas neste trabalho para a aquisição e uso de outros tipos de informação, tais como a previsão de custo de um projeto ou mesmo gestão automatizada de riscos deste.

REFERÊNCIAS

ACTIVECOLLAB. **Active Collab.** Disponível em: <https://activecollab.com>. Acesso em 09 set. 2015.

BALSERA, Joaquin V. set al. (2012). Data Mining Applied to the Improvement of Project Management. **Data Mining Applications in Engineering and Medicine**, Dordrecht, v.1, n.1, p. 59-74, ago. 2012.

BASILI, V. R., CALDIERA, G., ROMBACH, D. H. Goal Question Metric Paradigm. **Encyclopedia of Software Engineering**, New York, v. 2, n. 2, p. 529-532, jan. 2002.

FAYYAD, U., SHAPIRO, G. P., SMYTH, P. From Data Mining to Knowledge Discovery in DataBases. **AI Magazine**, Palo Alto, v. 17, n. 3, p. 37-54, 1996.

GOOGLE. **Angular**. Disponível em: <https://angularjs.org>. Acesso em 1 jan 2016.

MONGODB INC. **MongoDB**. Disponível em: <https://www.mongodb.com>. Acesso em 28 jan. 2017.

MOTA, M. **C4.5**. Disponível em: <https://github.com/miguelmota/C4.5>. Acesso em 11 fev. 2017.

NICHOLAS, J. M., STEYN, H. **Project Management for Engineering, Busineed and Technology**. 4. ed. New York: Routerledge, 2012.

NODE.JS FOUNDATION. **Express**. Disponível em: <http://expressjs.com/pt-br>. Acesso em 28 jan. 2017.

NODE.JS FOUNDATION. **Node**. Disponível em: <http://nodejs.org/en>. Acesso em 28 jan. 2017.

PRESSMAN, R. S. **Engenharia de Software**: uma abordagem profissional. 7. ed. Porto Alegre: AMGH, 2011.

PROJECT MANAGEMENT INSTITUTE. **A Guide to the Project Management Body of Knowledge**. 4. Ed. Newtown Square: Project Management Institute, 2008.

QUINLAN, J. R. Induction of Decision Trees. **Machine Learning**. Boston, v. 1, n. 1, p. 81-106, 1986.

QUINLAN, J. R. Learning Logical Definitions from Relations. **Machine Learning**. Boston, v. 5, n. 3, p. 239-266, 1990.

QUINLAN, J. R. **C4.5:** programs for machine learning. San Mateo: Morgan Kaufmann, 1994.

RUNRUN.IT. **Runrun.it**. Disponível em: <http://runrun.it/pt-BR>. Acesso em 9 set. 2015.

SCHWABER, K., SHUTHERLAND, J. **Guia do Scrum:** um guia definitivo para o scrum. Disponível em: < http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf >. Acesso em 9 set. 2015.

SOMMERVILLE, I. **Software Engineering**. 8. ed. Harlow: Pearson Education, 2007.

SYMEONIDIS, A. L., MITKAS, P. A. **Agent Intelligence Through Data Mining**. New York: Springer, 2005.

TWITTER. **Bootstrap**. Disponível em: <http://getbootstrap.com>. Acesso em 28 jan. 2017.

WITTEN, I. H., FRANK, E., HALL, M. A. **Data Mining:** practical machine learning tools and techniques. 3 ed. Burlington: Morgan Kaufmann, 2011.

WRIKE INC. **Wrike**. Disponível em: <https://www.wrike.com>. Acesso em 9 set. 2015.