Programa de Pós-Graduação em

Computação Aplicada

Mestrado Acadêmico

Felipe Chaves Rodrigues

Kairós: Modelo Preditivo Baseado em Históricos de Contextos para o Gerenciamento de Tempo em Projetos de Software

São Leopoldo, 2019

UNIVERSIDADE DO VALE DO RIO DOS SINOS
Escola Politécnica
Programa Interdisciplinar de Pós-Graduação em Computação Aplicada – PPGCA

FELIPE CHAVES RODRIGUES

KAIRÓS: MODELO PREDITIVO BASEADO EM HISTÓRICOS DE CONTEXTOS PARA O GERENCIAMENTO DE TEMPO EM PROJETOS DE SOFTWARE

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos - UNISINOS

Orientador: Prof. Dr. Jorge Luis Victória Barbosa

São Leopoldo

2019

R696k Rodrigues, Felipe Chaves.
Kairós : modelo preditivo baseado em históricos de contextos para o gerenciamento de tempo em projetos de software / Felipe Chaves Rodrigues – 2019.
[99] f. : il. ; 30 cm.

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, 2019.
"Orientador: Prof. Dr. Jorge Luis Victória Barbosa."

1. Modelo Kairós. 2. Projetos de software. 3. Gerenciamento de tempo. I. Título.

CDU 004

Dados Internacionais de Catalogação na Publicação (CIP)
(Bibliotecária: Amanda Schuster – CRB 10/2517)

UNISINOS

UNIVERSIDADE DO VALE DO RIO DOS SINOS
Escola Politécnica
Programa Interdisciplinar de Pós-Graduação em Computação Aplicada – PPGCA

FELIPE CHAVES RODRIGUES

KAIRÓS: MODELO PREDITIVO BASEADO EM HISTÓRICOS DE CONTEXTOS PARA O GERENCIAMENTO DE TEMPO EM PROJETOS DE SOFTWARE

Dissertação apresentada à Universidade do Vale do Rio dos Sinos – Unisinos, como requisito parcial para obtenção do título de Mestre em Computação Aplicada.

Aprovado em 28/03/2019

BANCA EXAMINADORA

Kleinner Silva Farias de Oliveira - UNISINOS

Nome do Componente da Banca Examinadora – Instituição a que pertence

Gustavo Pessin – Instituto Tecnologico Vale

Nome do Componente da Banca Examinadora – Instituição a que pertence

Prof. Dr. Jorge Luis Victória Barbosa (Orientador)

VISTO E PERMITIDA A IMPRESSÃO

São Leopoldo,

Prof. Dr.
Coordenador PPG em Computação Aplicada

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES) - Código de Financiamento 001

“However difficult life may seem, there is always something you can do and succeed at. It matters that you don't just give up.”

Stephen William Hawking, 2016.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API	Application Programming Interface
CSS	Cascading Style Sheets
EAP	Estrutura Analítica do Projeto
GRASP	Greedy Randomized Adaptive Search Procedure
HTML	HyperText Markup Language
IDE	Integrated Development Environment (Ambiente de Desenvolvimento Integrado)
PMBOK	Project Management Body of Knowledge
PMI	Project Management Institute
PRINCE2	Projects In Controlled Environments
RQ	Research Question (Questão de Pesquisa)
RUP	Rational Unified Process
UC	Use Case (Caso de Uso)
UML	Unified Modeling Language

SUMÁRIO

1 INTRODUÇÃO

Nas mais variadas áreas de negócio, o gerenciamento de projetos tem demonstrado sua importância para atingir os objetivos da empresa, seja ela privada ou governamental. De acordo com Mark Langley, presidente do *Project Management Institute* (PMI), a maior e mais reconhecida entidade na gestão de projetos, um projeto somente atinge o sucesso se entregar os benefícios que a empresa espera obter. Esta visão torna-se cada vez mais importante no contexto complexo, competitivo e em constante mudança em que o mercado atual se encontra (PMI, 2016).

A definição da palavra “projeto”, de acordo com o PMI, diz que este é um esforço temporário, com início e término definidos, para criar um produto, serviço ou resultado único (PMI, 2013). A conclusão de um projeto deve ocorrer quando este tiver seus objetivos atingidos, e para que os objetivos sejam atingidos dentro do esperado, o gerenciamento de tempo é aplicado desde a concepção até a conclusão.

Assim, dentre as inúmeras áreas da gestão de projetos, o gerenciamento de tempo se destaca por ter um papel essencial no resultado do trabalho. Por envolver diretamente os prazos, o custo e a qualidade são diretamente afetados por este ponto. O planejamento do cronograma do projeto é alvo de inúmeros estudos da literatura há muito tempo, e ainda hoje continua sendo bastante discutido (ABDEL-HAMID, MADNICK, 1983; JOSLIN, POOLE, 2005; XIAO et. al., 2010; LOPEZ-MARTIN, CHAVOYA, MEDA-CAMPANA, 2015; NIGAR, 2017), abordando diversas técnicas de predição para obter o melhor resultado, no menor tempo e custo.

Em um cenário de complexidade crescente, com equipes cada vez mais distribuídas, o uso de novas tecnologias para o auxílio na gestão de projetos tem se mostrado uma alternativa importante. Conceitos de computação móvel e ubíqua vêm sendo cada vez mais aplicados em diversas áreas (WEISER, 1991; SATYANARAYANAN, 2001; BARBOSA, 2015), e estudos mostram também a possibilidade de auxílio na gestão de projetos (BATISTA et. al., 2011; FILIPPETTO et. al., 2016; FILIPPETTO, LIMA, BARBOSA, 2017).

1.1 Motivação

Um cronograma de projeto é uma representação da sua estrutura analítica, organizada de maneira temporal, norteando a execução do projeto em todos os seus níveis, com datas estimadas de início e fim para cada uma das fases e suas tarefas. O cronograma deve ser elaborado considerando também todas as dependências entre as tarefas, bem como a disponibilidade de recursos para a execução. Visto que cada tarefa pode depender de uma ou várias outras tarefas, em algumas situações o atraso de apenas uma tarefa pode desencadear um atraso significativo no final do projeto, tanto pela própria dependência entre as tarefas, quanto pela indisponibilidade de algum recurso para a realização de outra tarefa.

Cada projeto, bem como suas fases e tarefas, tem características específicas. Muitas destas características se repetem entre eles, e podem ser identificadas, classificadas e quantificadas. Ao analisar o andamento de cada uma das tarefas que já ocorreram no próprio projeto ou em outros projetos da empresa, pode-se perceber a existência de padrões. Com a análise destes padrões torna-se viável a identificação de problemas, antes mesmo que cheguem a ocorrer, através de mecanismos de predição baseados nestes dados históricos.

Na área de desenvolvimento de software, existem fatores que tornam ainda mais complexa a execução e o controle do cronograma. Dada a intangibilidade do resultado do projeto, as alterações de escopo acabam ocorrendo com uma maior frequência do que em outras áreas de negócio. Além disso, a frequente necessidade de trabalhar com integrações com outros sistemas já existentes aumenta consideravelmente a complexidade destes projetos.

Além dos fatores de complexidade, as diferentes metodologias de desenvolvimento de software existentes trazem especificidades para o gerenciamento de tempo. Os métodos ágeis, amplamente utilizados atualmente, trazem em sua base a necessidade de realizar inúmeras entregas menores ao longo do projeto, e não uma única entrega ao final. Dessa forma, o software evolui de modo incremental, já trazendo o *feedback* dos usuários para o próximo ciclo evolutivo, o que também auxilia na priorização dos próximos itens a serem entregues.

Na computação ubíqua, diferentes áreas de estudo podem ser elencadas, como o uso de localização (HIGHTOWER, BORRIELLO, 2001), perfis (WAGNER, BARBOSA, BARBOSA, 2014) e contextos (DEY, SALBER, ABOWD, 2001). Visto que é uma área relativamente recente na computação, existem oportunidades de pesquisa em discussão, como o uso de históricos e predição de contextos (BARBOSA, 2015; ROSA, BARBOSA, BARCELOS, 2016). Ainda, a utilização de sistemas de recomendação é cada vez maior, apresentando as predições de maneira proativa, o que condiz com o conceito principal da computação ubíqua.

1.2 Definição do Problema

Para que os objetivos de um projeto sejam atingidos, a gestão de projetos atua desde o início do projeto, passando por planejamento, execução, monitoramento e controle, até o encerramento. Este processo se repete nos inúmeros grupos de conhecimento abordados, como escopo, qualidade, tempo, custos, entre outros (PMI, 2013).

Especialmente na gestão de tempo, o processo de controle do cronograma tem influência no sucesso do projeto, pois o andamento das tarefas é crucial para a conclusão do projeto no prazo. Neste sentido, uma análise da literatura sobre predições em cronogramas de projeto mostra a existência de inúmeros trabalhos atuando na fase de planejamento de cronogramas, mas poucas abordagens da fase de execução e controle.

Considerando a costumeira complexidade de projetos de software, cujos escopos costumam ser alterados inúmeras vezes durante o andamento do projeto, e que ainda podem ter equipes grandes e distribuídas ao redor do mundo, o controle da execução de todas as tarefas simultaneamente é uma tarefa difícil. Dessa forma, emerge uma necessidade de auxílio aos gestores, principalmente na identificação de problemas de maneira proativa. Neste cenário, surgem as seguintes questões de pesquisa:

- Como realizar predições sobre a possibilidade de atraso em uma tarefa durante sua execução?
- Como sugerir ações corretivas para tarefas que apresentam problemas?

1.3 Objetivos

Esta dissertação apresenta o modelo Kairós para predições no cronograma de tarefas, com base nos históricos de evolução das atividades e projetos, realizando recomendações para que o gestor do projeto saiba como atuar para resolver os possíveis problemas de maneira proativa. Para que seja possível atingir este objetivo, foram elencados os seguintes objetivos específicos:

- Analisar os conceitos relevantes relacionados aos temas de estudo, sendo eles gestão de tempo, metodologias de desenvolvimento de software, melhores práticas de gestão de projetos, computação ubíqua, sistemas de predição e recomendação;
- Caracterizar os trabalhos relacionados presentes na literatura, visando identificar as características já atendidas pelos estudos existentes e as necessidades ainda não supridas;
- Especificar um modelo computacional que suporte um sistema de recomendação na fase de execução e controle em cronogramas de projetos de software;
- Desenvolver um protótipo de aplicação baseado neste modelo, considerando as necessidades identificadas no estado da arte;
- Validar o modelo apresentado através do protótipo construído.

1.4 Metodologia

Para a elaboração deste estudo, inicialmente foram realizadas pesquisas com o objetivo de identificar ferramentas, técnicas, tecnologias e conceitos que auxiliassem

na predição de cronogramas de projeto. Percebeu-se a partir destas pesquisas que a literatura é bem esparsa neste tema, e que, apesar da variedade de abordagens, ainda existe espaço para novos estudos.

Assim, foi realizado um estudo de mapeamento sistemático, que tem como objetivo apresentar uma visão geral do cenário de pesquisa em uma determinada área, classificando e agrupando as evidências de maneira granular (PETERSEN et al., 2008; 2015). Este tipo de estudo permite identificar linhas de pesquisa pouco exploradas, auxiliando os pesquisadores a direcionarem o foco de futuras revisões sistemáticas ou estudos primários para estas áreas (KITCHENHAM, CHARTERS, 2007).

Considerando o resultado deste estudo, foi possível iniciar a definição de um modelo conceitual adequado às necessidades ainda não supridas. Além disso, algumas oportunidades de pesquisa foram encontradas em relação aos modelos de predição e recomendação, permitindo que novos algoritmos sejam aplicados aos problemas existentes.

Com a evolução deste modelo conceitual, foi possível iniciar o esboço da arquitetura do modelo, e assim identificar questões teóricas que necessitavam de maior aprofundamento. Novas pesquisas foram então realizadas, com o objetivo de identificar as melhores técnicas e tecnologias que poderiam ser utilizadas para completar a arquitetura do modelo. Assim, foi possível criar uma primeira versão da especificação do modelo. Mesmo que esta especificação venha a sofrer adaptações e melhorias, sua estrutura básica deverá ser mantida até o final da pesquisa.

Para a validação do modelo Kairós, foi elaborado um protótipo que suporta testes de desempenho, eficiência e eficácia. A avaliação do modelo se deu através de uma validação cruzada, em modelo *holdout*, utilizando dados reais de 24 meses de projetos em uma instituição do ramo de desenvolvimento de sistemas. Deste período, os primeiros 18 meses foram considerados como dados de treinamento, e os 6 meses restantes como dados de validação. Além disso, com este protótipo é possível identificar limitações do modelo, bem como outras possibilidades de evolução.

1.5 Organização

Esta dissertação está organizada em seis capítulos. O capítulo 2 apresenta a fundamentação teórica, trazendo conceitos e tecnologias que serão a base do desenvolvimento da pesquisa. Entre estes, destacam-se a gestão de projetos, com ênfase na área de gerenciamento de tempo e nos diferentes modelos de desenvolvimento de software, e a computação ubíqua, aliada aos conceitos de predição e recomendação. O capítulo 3 detalha o estudo de mapeamento sistemático realizado para identificar o estado da arte em modelos de predição para cronogramas de projeto de software. No capítulo 4, o modelo Kairós é apresentado, bem como suas características e sua arquitetura. Um protótipo para validação do modelo é abordado no capítulo 5, assim como a metodologia que foi utilizada neste processo de validação e a discussão dos resultados. Finalmente, o capítulo 6 traz as contribuições apresentadas pelo modelo, considerações finais e possibilidades de evolução do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo introduz os principais conceitos e tecnologias que são utilizados nesta proposta de dissertação. A organização deste texto consiste nas seguintes seções. A seção 2.1 apresenta um cenário geral da gestão de projetos, com ênfase no gerenciamento de tempo e em como as diferentes metodologias de desenvolvimento de software abordam este tema. A seção 2.2 aborda a computação ubíqua, seus princípios e aplicações, e como podem ser utilizadas para auxiliar na gestão de projetos. A seção 2.3 contempla sistemas de predição e recomendação, seus principais usos e as tecnologias relacionadas ao tema.

2.1 Gestão de tempo em projetos

A definição mais aceita do que é um projeto indica que este é um esforço temporário, realizado com o objetivo de criar um serviço, produto ou resultado único, sendo este tangível ou intangível (PMI, 2013). Já o conceito de um projeto sem falhas traz a ideia de que precisa ser concluído dentro do tempo e custo estimados, entregando por completo o objetivo definido para o trabalho (DINSMORE, CAVALIERI, 2011). Estas definições mostram a importância do tempo em um projeto, visto que ele precisa ter um início e término bem definidos, e somente é considerado como bem-sucedido caso estas datas sejam respeitadas.

A gestão de tempo atua nos processos necessários para o gerenciamento do término pontual do projeto, tendo o cronograma como a sua principal saída (PMI, 2013). O cronograma pode ser definido como uma lista de tarefas do projeto, com a previsão de início e fim da execução de cada uma destas (RAMZAN et. al., 2012). Cada tarefa representa o menor nível de pacote de trabalho existente, sendo uma atividade única com objetivo específico dentro do projeto.

O desenvolvimento do cronograma depende diretamente de alguns fatores como tamanho da equipe, escopo do projeto, complexidade das tecnologias envolvidas, entre outros inúmeros itens. Porém, a parte mais crucial do desenvolvimento de um cronograma é a EAP (Estrutura Analítica do Projeto), que

define os pacotes de trabalho que serão executados para atingir o objetivo do projeto, bem como as dependências que existem entre estes pacotes (JONES, 2009).

Com o aumento da complexidade dos projetos e ambientes em que estes estão inseridos, a criação de um cronograma consistente e executável torna-se um desafio com impacto no resultado do projeto (NIGAR, 2017). Além disso, o cronograma gerado na fase de planejamento pode sofrer modificações durante o andamento do projeto, o que torna imprescindível o monitoramento e o controle das tarefas ao longo de todo o período de execução.

Os projetos de desenvolvimento de software possuem características ainda mais marcantes no que se refere ao seu gerenciamento. Inúmeros fatores tornam este tipo de projeto ainda mais suscetível a mudanças e incertezas, como a intangibilidade do produto, a falta de técnicas assertivas para a estimativa dos pacotes de trabalho, e a frequente existência de equipes distribuídas trabalhando simultaneamente nas mesmas tarefas (KROLL, FRIBOIM, HEMMATI, 2017).

Apesar de grande parte da literatura sobre gerenciamento de projetos abordar métodos clássicos como PMBOK ou PRINCE2, o gerenciamento ágil de projetos tem se mostrado cada vez mais presente nas empresas. Tal formato de gestão aborda os mesmos processos e áreas de conhecimento dos demais métodos, com a principal variação no formato de planejamento a alocação das tarefas e recursos. Enquanto os métodos tradicionais dão maior importância para a organização e a documentação, os métodos ágeis pregam maior foco na capacidade de resposta rápida e na habilidade de lidar com situações inesperadas (KACZOROWSKA, 2015).

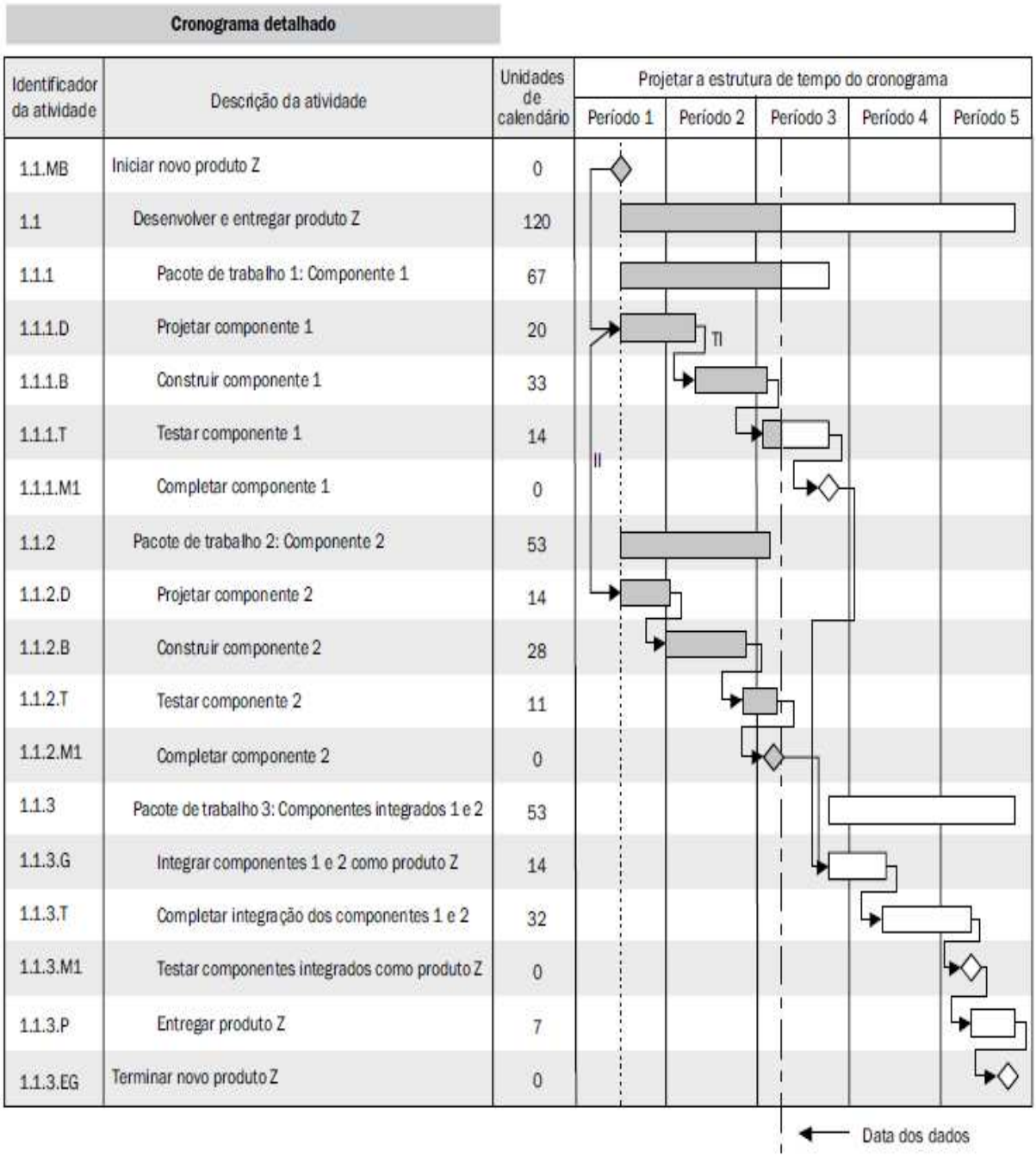
Em uma abordagem clássica, todas as tarefas do projeto são consideradas no cronograma planejado inicialmente. Tal modo visa uma única entrega com todas as atividades concluídas, conforme exemplo mostrado na Figura 1.

Já em uma abordagem ágil, todas as atividades fazem parte de uma listagem chamada *product backlog*. A partir desta, são geradas pequenas entregas que serão realizadas de maneira constante em pequenas janelas de tempo, cada uma destas possuindo um cronograma próprio (RAMIREZ-NORIEGA et. al., 2016), conforme mostra a Figura 2.

Cada uma dessas janelas, chamadas *sprints*, devem ser concluídas com a entrega de itens que agreguem valor ao projeto. Para a elaboração da lista de tarefas

de cada *sprint*, é realizada uma reunião chamada de *sprint planning*, que deve envolver todos os membros da equipe, com o intuito de estimar e priorizar todos os itens que serão desenvolvidos em uma iteração. Os itens de maior prioridade do *product backlog* devem ser alocados no sprint de acordo com a capacidade da equipe, gerando o *sprint backlog* – que corresponde a EAP de um cronograma de *sprint* (SOUSA et. al., 2015).

Figura 1 - Cronograma geral do projeto

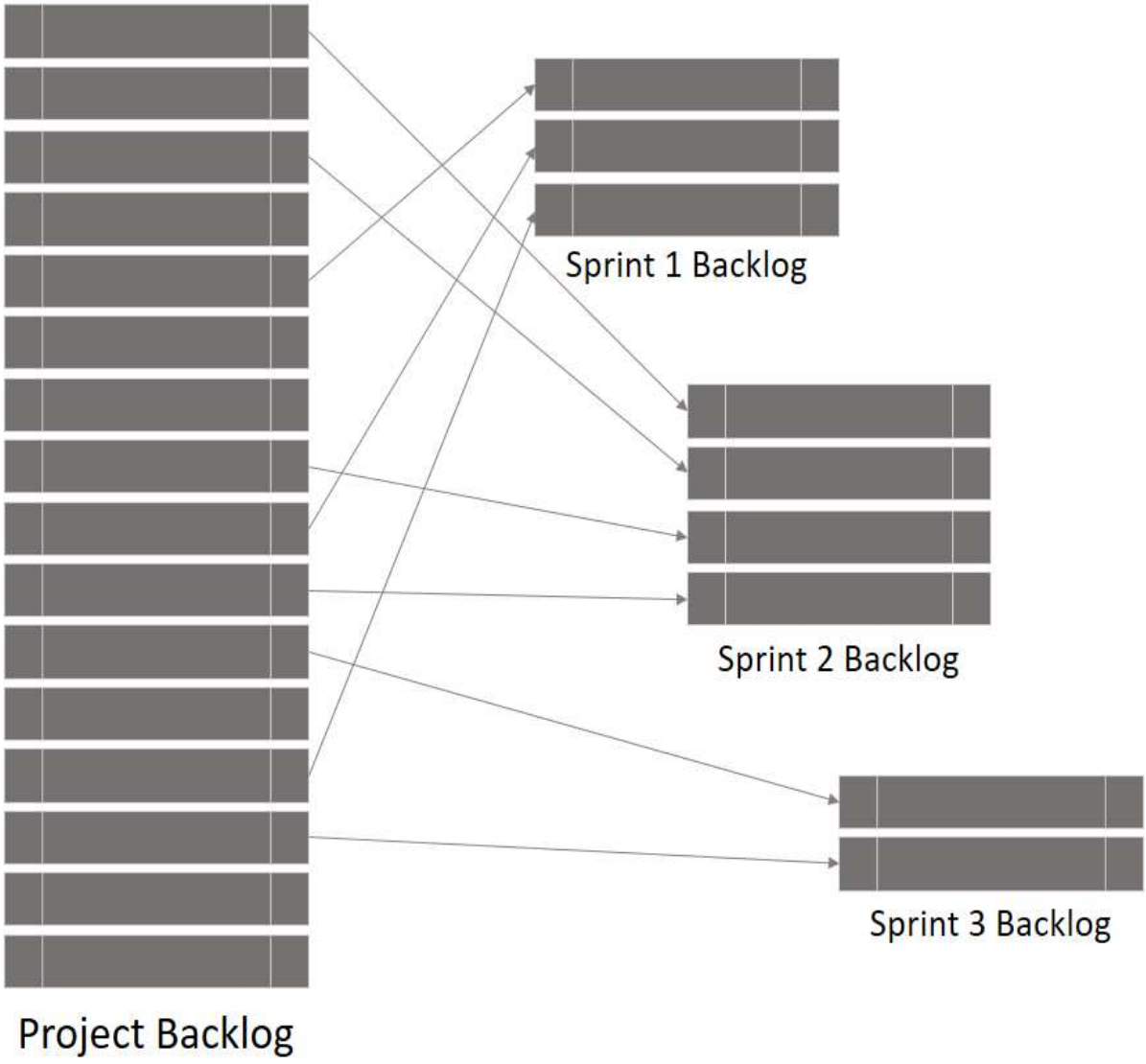


Cronograma detalhado

Identificador da atividade	Descrição da atividade	Unidades de calendário
1.1.MB	Iniciar novo produto Z	0
1.1	Desenvolver e entregar produto Z	120
1.1.1	Pacote de trabalho 1: Componente 1	67
1.1.1.D	Projetar componente 1	20
1.1.1.B	Construir componente 1	33
1.1.1.T	Testar componente 1	14
1.1.1.M1	Completar componente 1	0
1.1.2	Pacote de trabalho 2: Componente 2	53
1.1.2.D	Projetar componente 2	14
1.1.2.B	Construir componente 2	28
1.1.2.T	Testar componente 2	11
1.1.2.M1	Completar componente 2	0
1.1.3	Pacote de trabalho 3: Componentes integrados 1 e 2	53
1.1.3.G	Integrar componentes 1 e 2 como produto Z	14
1.1.3.T	Completar integração dos componentes 1 e 2	32
1.1.3.M1	Testar componentes integrados como produto Z	0
1.1.3.P	Entregar produto Z	7
1.1.3.EG	Terminar novo produto Z	0

Fonte: PMI, 2013

Figura 2 - *Project Backlog* e S*print Backlog*



Fonte: Elaborado pelo Autor

O uso de técnicas e ferramentas para o controle do cronograma se faz necessário em ambas as abordagens. O gerente de projetos precisa de informações atualizadas de cada tarefa, para que possa analisar os cenários encontrados e identificar as necessidades de mudança no cronograma.

2.2 Computação Ubíqua

O conceito de computação ubíqua foi introduzido por Mark Weiser há mais de duas décadas, com a previsão de que estaríamos cercados por inúmeros dispositivos e sensores, que deveriam interagir de maneira natural e transparente com os seres humanos (WEISER, 1991). Com a evolução das tecnologias ao longo do tempo, estes conceitos tornaram-se cada vez mais fortes. Em 2001, Satyanarayanan abordou novamente o tema, onde constatou que as tecnologias mais profundas são aquelas que não aparecem, fazendo parte do nosso dia-a-dia com tamanha importância que se tornam indistinguíveis do ambiente em que vivemos (SATYANARAYANAN, 2001).

Ao longo da última década, com a chegada da terceira geração da computação, os próprios conceitos de computação ubíqua passaram a tornar-se indistinguíveis do restante da computação (ABOWD, 2012). No entanto, a pesquisa relacionada a essa área passou a ser parte integrante de muitas áreas da computação, sendo abordada dentro de áreas como a computação móvel e distribuída (AMEYED, MIRAOUI, TADJ, 2015), sistemas de inteligência artificial (ROSA, BARBOSA, BARCELOS, 2016), computação aplicada à saúde (ESPOSITO et. al., 2017; VIANNA, BARBOSA, 2017; PITTOLI et. al., 2018), bem-estar (VIANNA, BARBOSA, PITTOLI, 2017), acessibilidade (BARBOSA et. al., 2018), educação (WAGNER, BARBOSA, BARBOSA, 2014; HEIDRICH et. al., 2018), comércio (BARBOSA et. al., 2016), entre outras (BARBOSA, 2015).

Um dos conceitos mais utilizados da computação ubíqua é o de aplicações sensíveis ao contexto. Um contexto computacional tem como objetivo representar quaisquer informações que possam ser usadas para caracterizar a situação de entidades que são consideradas importantes para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a própria aplicação. Assim, o contexto normalmente apresenta informações como localização, identidade, e o estado das entidades e grupos relevantes para a aplicação (DEY, SALBER, ABOWD, 2001).

Com o aumento do poder computacional, tornou-se possível o armazenamento em larga escala das informações de contexto nas aplicações. Os dados contextuais passaram a ser guardados de maneira histórica, provendo informações importantes para que as aplicações possam analisar o passado de uma ou mais entidades, criando

assim o conceito de históricos de contextos (HONG et. al., 2009). Diversos estudos vêm sendo conduzidos sobre esta área desde então, com aplicações em sistemas de ensino (RIGO, CAMBRUZZI, BARBOSA, 2015; ROSA et. al., 2015), tomada de decisão (PRABAKARAN, KANNADASAN, 2018), saúde (ANYA, TAWFIK, 2017; ABOROKBAH et. al., 2017), entre outras.

Para complementar o uso de contextos, que já abordam duas dimensões temporais (contextos atuais e históricos de contextos passados), alguns estudos passaram a considerar também o futuro dos contextos (KONIG et. al., 2011). Assim, novas técnicas computacionais foram desenvolvidas para a predição de contextos, utilizando como base as informações históricas e atuais de cada entidade, e possibilitando que as ferramentas ajam de maneira proativa no auxílio aos usuários (SIGG, HASELOFF, DAVID, 2010; SIGG et. al., 2011; BURBEY, MARTIN, 2012).

Enquanto o uso de históricos de contextos dependia principalmente de capacidade de armazenamento de dados, a predição de contextos necessita de poder de processamento para realizar suas tarefas. Diversas técnicas computacionais conhecidas podem ser empregadas para a realização das predições, como cadeias de Markov, redes bayesianas, redes neurais, entre outras (ROSA, BARBOSA, BARCELOS, 2016).

Ainda, outros métodos surgem como potenciais áreas de estudo, como as técnicas de *data mining* e a predição por similaridade semântica. Utilizando a técnica de *clustering* em mineração de dados, é possível agrupar elementos similares de maneira dinâmica, permitindo a classificação dos contextos e enquadramento de entidades nestas classes (AMEYED, MIRAOUI, TADJ, 2015). Já na técnica de similaridade semântica, são utilizados algoritmos de cruzamento que indicam o grau de similaridade de cada ação em seu contexto atual. Com isso, o sistema realiza comparações com outras ações em contextos passados e indica a previsão do contexto futuro (NAJAR, PINHEIRO, SOUVEYET, 2014).

2.3 Sistemas de recomendação

No início dos anos 90, com o aumento da capacidade computacional e o início da difusão da internet, o fluxo de informações aumentou a ponto de sobrecarregar os usuários e dificultar a tomada de decisões. Para auxiliar neste ponto, foram desenvolvidos modelos proativos que realizavam predições para estimar o quanto uma informação era realmente útil e interessante para determinado usuário, chamados modelos de recomendação (ADOMAVICIUS, TUZHILIN, 2005).

Com um aumento ainda mais significativo do fluxo de informações desde então, o assunto continuou sendo tema de inúmeros estudos. Desde os primeiros trabalhos, abordando filtros colaborativos (RESNICK et. al., 1994; HILL et. al., 1995), até as mais recentes aplicações com mecanismos de aprendizagem de máquina (WEI et. al., 2017), o tema tem evoluído através de pesquisa acadêmica e desenvolvimentos com objetivos comerciais, com uma ampla gama de aplicações (KONSTAN, RIEDL, 2012).

Os sistemas de recomendação têm como principal característica a sugestão proativa de algum item, produto, serviço ou ação, de acordo com o contexto em que se enquadra (ADOMAVICIUS, TUZHILIN, 2005). Os primeiros modelos abordavam apenas itens que possuíssem características e avaliações mais simplificadas, e trabalhavam com recomendações baseadas em similaridade e proximidade. Ou seja, se o usuário X gostou dos produtos A, B e C, e o usuário Y gostou dos produtos A e C, o sistema recomendaria o produto B para o usuário Y, visto que o usuário X tem gostos semelhantes ao dele.

Ao longo dos anos, novos modelos de recomendação foram criados, considerando variáveis mais complexas e trabalhando com novos algoritmos. Esta evolução facilitou a aplicação do conceito em áreas mais amplas do que recomendação de produtos. Diversas utilidades têm sido encontradas para estes algoritmos, como nos sistemas de aprendizado (SHISHEHCHI et. al., 2011), sugestões de pontos de interesse (WONG et. al., 2015), viagens (MAJID et. al., 2013), desenvolvimento de software (GASPARIC, MURPHY, RICCI, 2017), entre outros.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta um mapeamento sistemático sobre as pesquisas que envolvem predições e recomendações relacionadas ao gerenciamento de tempo em projetos de software. Neste mapeamento será apresentada uma visão geral das tecnologias ou modelos pesquisados na literatura, classificando os trabalhos desenvolvidos de acordo com critérios considerados relevantes para a resolução dos problemas identificados.

O estudo foi conduzido através da metodologia de mapeamento sistemático, que tem como objetivo analisar as publicações encontradas em uma área de conhecimento, classificá-las e agrupá-las de acordo com critérios pré-estabelecidos. Para tal, foram elencadas 4 questões de pesquisa, que serviram como base para a elaboração das *strings* de busca nas bases de dados. Foram encontrados 1893 artigos, que após triagem resultaram em 32 estudos. Sobre estes trabalhos, foi realizada uma análise com o intuito de apresentar o estado do cenário de pesquisa em relação a predição de cronogramas em projetos de software.

A execução do processo de mapeamento seguiu o processo descrito por Petersen et. al. (2015), onde as seguintes etapas principais foram definidas: a definição das questões de pesquisa e a busca pelos estudos na literatura, na seção 3.1; a seleção dos documentos encontrados na etapa anterior através do processo de triagem, mostrado na seção 3.2; a extração dos dados para realização do mapeamento, apresentada na seção 3.3; na seção 3.4, a análise dos artigos encontrados e os resultados obtidos; um comparativo entre os estudos selecionados e o modelo Kairós na seção 3.5; e por fim, a seção 3.6 apresenta conclusões que foram obtidas através do estudo.

3.1 Questões de pesquisa

As questões de pesquisa identificam na literatura os estudos realizados sobre predições e recomendações no gerenciamento de tempo em projetos. Com estas

questões, o estudo pretende identificar o estado atual deste cenário de pesquisa, as tendências e as oportunidades de estudo que não têm sido abordadas nos trabalhos encontrados. As questões de pesquisa que nortearam o estudo são mostradas na Tabela 1.

Tabela 1 - Questões de Pesquisa Elaboradas

Abordagem	Questão de Pesquisa
RQ1	Quais os tipos de estudos realizados nas áreas de predição e recomendação de cronogramas?
RQ2	Em quais etapas do ciclo de gerenciamento de projetos estão focados os estudos de predição e recomendação de cronogramas?
RQ3	Quais os modelos de projeto de software são abordados nos estudos de predição e recomendação de cronogramas?
RQ4	Quais as técnicas utilizadas para predição e recomendação de cronogramas são encontradas na literatura?

Fonte: Elaborado pelo Autor

Com base nestas questões de pesquisa, o objetivo seguinte é a definição das *strings* que serão usadas para pesquisa nas bases de dados. Para tal, foram elencadas palavras-chave que possibilitassem a localização de estudos relacionados, com os seguintes requisitos:

a) Os artigos devem estar relacionados ao gerenciamento de projetos;
b) Os artigos devem abordar cronogramas;
c) Os artigos devem apresentar alguma técnica, pesquisa ou modelo de predição e/ou recomendação;
d) Os artigos devem trabalhar com as especificidades de projetos de *software*.

Baseado nestes requisitos, as palavras-chave foram elaboradas, dando origem às *strings* para pesquisa nas bases de dados, que são apresentadas na Tabela 2.

Estas pesquisas foram executadas no mês de agosto de 2017 nas quatro bases de dados, através dos mecanismos de pesquisa avançada existente em cada uma delas, e foi obtida uma população inicial de 1893 artigos.

Tabela 2 - Bases de dados Pesquisadas e *Strings* de Pesquisa

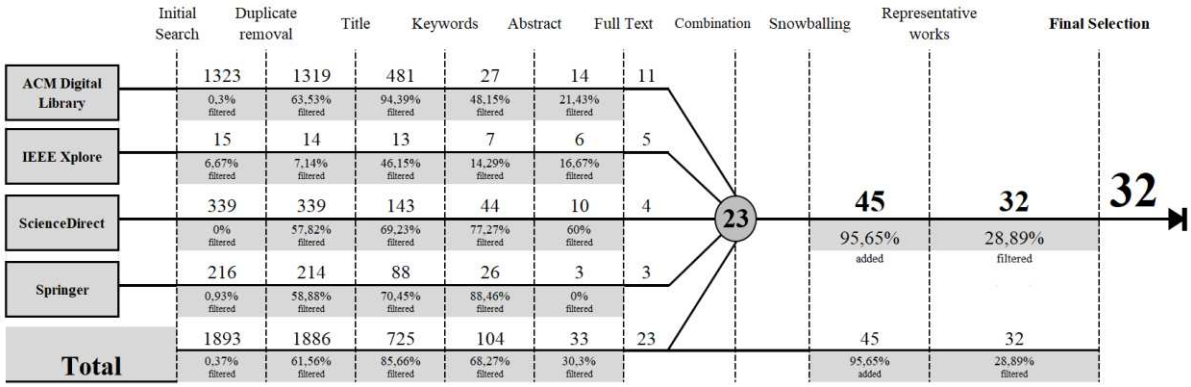
Base de Dados	***String* de Pesquisa**
ACM Digital Library	+ ("project management") + ("schedule") + ("prediction" "recommendation" "forecast") + ("software project")
IEEE Xplore	(("project management") AND ("schedule") AND ("prediction" OR "recommendation" OR "forecast") AND ("software project"))
Springer	("project management") AND ("schedule") AND ("prediction" OR "recommendation" OR "forecast") AND ("software project")
ScienceDirect	("project management") AND ("schedule") AND ("prediction" OR "recommendation" OR "forecast") AND ("software project")

Fonte: Elaborado pelo Autor

3.2 Critérios de Inclusão e Exclusão

Seguindo o modelo definido por Petersen et al. (2008; 2015), seguiu-se para a etapa de triagem dos dados obtidos, visando a seleção de artigos de maior relevância no que se refere às questões de pesquisa. Estes critérios foram executados em etapas, conforme mostra a Figura 3.

Figura 3 - Resultados das buscas e aplicação dos critérios de triagem



Fonte: Elaborado pelo Autor

Os artigos foram inicialmente agrupados por base de dados, com o objetivo de ter uma melhor visibilidade do impacto de cada um dos critérios de inclusão e exclusão aplicados. Foi utilizada para tal a ferramenta Mendeley, que possui funcionalidades de importação, agrupamento, identificação e classificação de artigos científicos de maneira simples e objetiva (VRKIĆ, 2014). Então, foram executadas as seguintes etapas do processo de triagem:

- Exclusão de artigos duplicados: uma pequena quantidade dos artigos encontrados estava disponível em mais de uma base de dados. Estes artigos foram identificados e marcados para exclusão, mantendo no estudo a base onde o artigo foi publicado primeiro. Nesta etapa, foram excluídos 4 artigos da ACM Digital Library, 1 artigo da IEEE Xplore e 2 artigos da Springer, totalizando uma redução de 0,37% no total de trabalhos;
- Filtro por título: os títulos dos trabalhos foram analisados, com o objetivo de encontrar artigos que não se relacionassem diretamente com as questões de pesquisa ou com a área dos estudos. Neste processo, 61,56% dos artigos foram filtrados, totalizando 725 artigos ainda enquadrados na pesquisa;

- *Keywording*: foram analisadas as palavras-chave disponibilizadas pelos autores, visando identificar apenas os artigos que abordassem a predição de cronogramas em gestão de projetos de software. Nesta etapa, percebeu-se uma grande tendência de estudos relacionados ao processo de estimativas das tarefas, ponto este que não se enquadra nos objetivos deste trabalho. Foram então removidos 85,66% dos artigos encontrados;
- Filtro por *abstract*: uma análise dos abstracts dos 104 artigos restantes foi realizada, onde foi possível identificar com maior precisão a área de atuação, o foco e a abordagem dos trabalhos encontrados. Assim, obteve-se um total de 33 artigos ainda enquadrados, demonstrando uma redução de 68,27% em relação à população anterior;
- Leitura completa: os trabalhos ainda enquadrados na pesquisa foram lidos por completo e classificados em relação às questões de pesquisa, removendo aqueles que não auxiliavam na resposta destas. 10 artigos foram excluídos por tratarem de cronogramas de projetos ou por não apresentar contribuição relacionada diretamente à predição ou recomendação;
- *Snowballing*: o processo de *snowballing* tem como meta adicionar estudos que não foram encontrados diretamente na pesquisa da base de dados, mas que se relacionam diretamente com o objetivo do trabalho, analisando as referências e as citações de artigos enquadrados na pesquisa (WOHLIN, 2014). Neste processo, foram analisados os títulos, *keywords* e abstracts dos 23 artigos presentes na busca até o momento, onde identificou-se 22 artigos adicionais para um processo de leitura completa;
- Trabalhos representativos: Foi realizada uma comparação entre os 45 artigos encontrados para identificar os que se tratavam de trabalhos únicos e representativos. Neste passo final, foram excluídos os artigos que se tratavam de incrementos de outros trabalhos já selecionados ou

que se tratavam de variações de um mesmo trabalho publicados em fontes distintas. Assim, a triagem foi finalizada com a exclusão de 28,89% dos artigos.

3.3 Extração dos dados

Os 32 estudos resultantes foram classificados com o objetivo primário de responder as questões de pesquisa identificadas na Tabela 1. Estes artigos são apresentados na Tabela 3, onde é possível identificar alguns dados referentes ao enquadramento de cada um destes com as questões de pesquisa, como o tipo de estudo, o modelo de gestão de projetos e as fases da gestão abordadas. A cada artigo foi atribuído um identificador, para facilitar posteriores referências na análise dos resultados.

Tabela 3 - Artigos selecionados através do mapeamento sistemático

Id	Title	In	Type	Year	Classification	Method	Phases
P1	RASP: Resource Allocator for Software Projects (BERTAZZONI, GIANNOTI, 1990)	3rd International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems	Conference	1990	Solution Purpose	Waterfall	Planning
P2	Predicting information technology project escalation: A neural network approach (ZHANG et. al., 2003)	European Journal of Operational Research	Journal	2003	Evaluation Research	Waterfall	Planning
P3	Genetic Optimization for Dynamic Project Control (HEGAZY, PETZOLD, 2003)	Journal of Construction Engineering and Management	Journal	2003	Model Purpose	Waterfall	Planning and Execution
P4	Making Resource Decisions for Software Projects (FENTON et. al., 2004)	26th International Conference on Software Engineering (ICSE 2004)	Conference	2004	Solution Purpose	Waterfall	Planning
P5	Agent-based simulation for software project planning (JOSLIN, POOLE, 2005)	2005 Winter Simulation Conference	Conference	2005	Evaluation Research	Waterfall	Planning and Execution

P6	Parametric Project Monitoring and Control: Performance-Based Progress Assessment and Prediction (HUNT, 2007)	2007 IEEE Aerospace Conference	Conference	2007	Evaluation of Existing Work	Waterfall	Execution
P7	Recommendation system for IT software project planning: A hybrid mining approach for the revised CBR algorithm (YANG, WANG, 2008)	5th International Conference Service Systems and Service Management	Conference	2008	Evaluation Research	Waterfall	Planning
P8	Time-line based model for software project scheduling with genetic algorithms (CHANG et. al., 2008)	Information and Software Technology	Journal	2008	Solution Purpose	Waterfall	Planning
P9	A genetic algorithm for the resource constrained multi-project scheduling problem (GONÇALVES, MENDES, RESENDE, 2008)	European Journal of Operational Research	Journal	2008	Evaluation Research	Waterfall	Planning
P10	A project scheduling problem with labour constraints and time-dependent activities requirements (DREZET, BILLAUT, 2008)	International Journal of Production Economics	Journal	2008	Evaluation Research	Waterfall	Planning
P11	Recommender system for software project planning one application of revised CBR algorithm (YANG, WANG, 2009)	Expert Systems with Applications	Journal	2009	Model Purpose	Waterfall	Planning
P12	New competitive results for the stochastic resource-constrained project scheduling problem: exploring the benefits of pre-processing (ASHTIANI, LEUS, ARYANEZHAD, 2009)	Journal of Scheduling	Journal	2009	Evaluation Research	Waterfall	Planning
P13	Resource-Constrained Project Scheduling for Timely Project Completion with Stochastic Activity Durations (BALLESTÍN, LEUS, 2009)	Production and Operations Management Society	Journal	2009	Evaluation Research	Waterfall	Planning
P14	Software project schedule variance prediction using Bayesian Network (WANG, WU, MA, 2010)	2010 IEEE International Conference on Advanced Management Science	Conference	2010	Solution Purpose	Waterfall	Planning and Execution
P15	Dynamic Resource Scheduling in Disruption-Prone Software Development Environments (XIAO et. al., 2010)	13th international conference on Fundamental Approaches to Software Engineering	Conference	2010	Evaluation Research	Waterfall	Execution
P16	Using Multi-objective Metaheuristics to Solve the Software Project Scheduling Problem (CHICANO et. al., 2011)	13th Annual Conference on Genetic and Evolutionary Computation (GECCO '11)	Conference	2011	Evaluation of Existing Work	Waterfall	Planning
P17	Theoretical and practical fundamentals for multi-objective optimisation in resource-constrained project scheduling problems (BALLESTÍN, BLANCO, 2011)	Computers & Operations Research	Journal	2011	Evaluation of Existing Work	Waterfall	Planning
P18	A Genetic Algorithm basic approach for software management project (KAROVA, AVRAMOVA, 2012)	International Conference on Computer Systems and Technologies - CompSysTech 2012	Conference	2012	Evaluation Research	Waterfall	Planning
P19	Decision making support in CMMI Project Planning Process Area using a hybrid simulation model (CRESPO, RUIZ, 2012)	2012 Winter Simulation Conference	Conference	2012	Evaluation Research	Waterfall	Planning and Execution

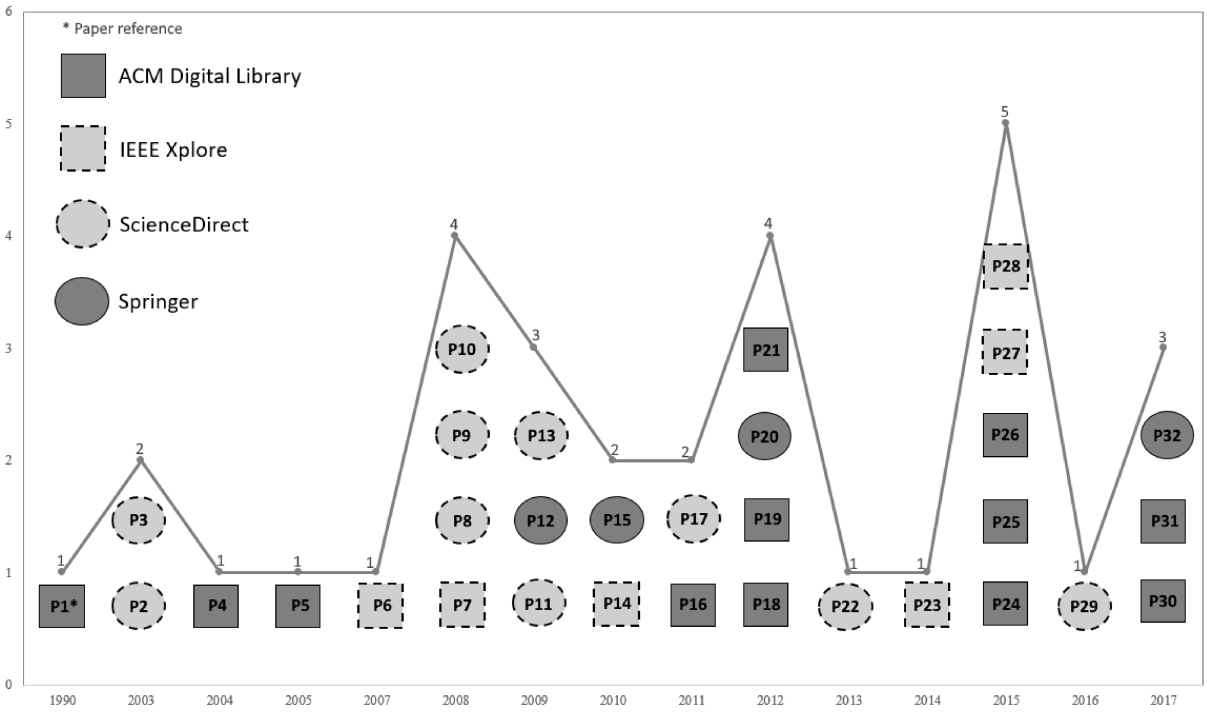
P20	Project scheduling conflict identification and resolution using genetic algorithms (RAMZAN et. al., 2012)	Telecommunication Systems	Journal	2012	Evaluation Research	Waterfall	Planning
P21	A Novel Multiobjective Formulation of the Robust Software Project Scheduling Problem (CHICANO et. al., 2012)	Proceedings of the 2012 European conference on Applications of Evolutionary Computation	Conference	2012	Evaluation Research	Waterfall	Planning
P22	Robust optimization for resource-constrained project scheduling with uncertain activity durations (ARTIGUES, LEUS, NOBIBON, 2013)	Flexible Services and Manufacturing Journal	Journal	2013	Evaluation Research	Waterfall	Planning
P23	Improved Evolutionary Algorithm Design for the Project Scheduling Problem Based on Runtime Analysis (MINKU, SUDHOLT, YAO, 2014)	IEEE Transactions on Software Engineering	Journal	2014	Model Purpose	Waterfall	Planning
P24	Ant colony algorithm based scheduling for handling software project delay (ZHANG et. al., 2015)	International Conference on Software and System Processes	Conference	2015	Evaluation Research	Waterfall	Execution
P25	Empirical Study of Multi-objective Ant Colony Optimization to Software Project Scheduling Problems (XIAO, GAO, HUANG, 2015)	2015 Annual Conference on Genetic and Evolutionary Computation	Conference	2015	Evaluation Research	Waterfall	Planning
P26	An Integrated Multi-Agent-Based Simulation Approach to Support Software Project Management (BAÍA, 2015)	37th International Conference on Software Engineering	Conference	2015	Model Purpose	Waterfall and Iterative	Planning and Execution
P27	Predicting delays in software projects using networked classification (CHOETKIERTIKUL, 2015)	30th IEEE/ACM International Conference on Automated Software Engineering	Conference	2015	Model Purpose	Waterfall	Execution
P28	A Fuzzy Logic Model for Predicting the Development Schedule of Software Projects (LOPEZ-MARTIN, CHAVOYA, MEDA-CAMPANA, 2015)	12th International Conference on Information Technology	Conference	2015	Evaluation Research	Waterfall	Planning
P29	Evolutionary multi-objective resource allocation and scheduling in the Chinese navigation satellite system project (XIONG et. al., 2016)	European Journal of Operational Research	Journal	2016	Model Purpose	Waterfall	Planning
P30	Model-based dynamic software project scheduling (NIGAR, 2017)	11th Joint Meeting on Foundations of Software Engineering	Conference	2017	Model Purpose	Own model	Planning
P31	An Empirical Study of Search-Based Task Scheduling in Global Software Development (KROLL, FRIBOIM, HEMMATI, 2017)	39th International Conference on Software Engineering: Software Engineering in Practice Track	Conference	2017	Solution Purpose	Distributed Waterfall	Planning
P32	Predicting the delay of issues with due dates in software projects (CHOETKIERTIKUL et. al., 2017)	Empirical Software Engineering	Journal	2017	Solution Purpose	Waterfall	Execution

Fonte: Elaborado pelo Autor

3.4 Análise dos dados

Dada a criticidade e a complexidade da gestão de tempo em projetos, a elaboração de cronogramas auxiliada por sistemas vem sendo tema de pesquisas desde a década de 80. Considerando abordagens preditivas em projetos de software, foi possível encontrar o primeiro artigo no ano de 1990 (BERTAZZONI, GIANNOTI, 1990). Os demais estudos estão distribuídos entre os anos de 2003 e 2017, com um pico no ano de 2015 onde 5 artigos foram publicados nesta área, conforme distribuição presente na Figura 4.

Figura 4 - Classificação dos estudos por ano de publicação

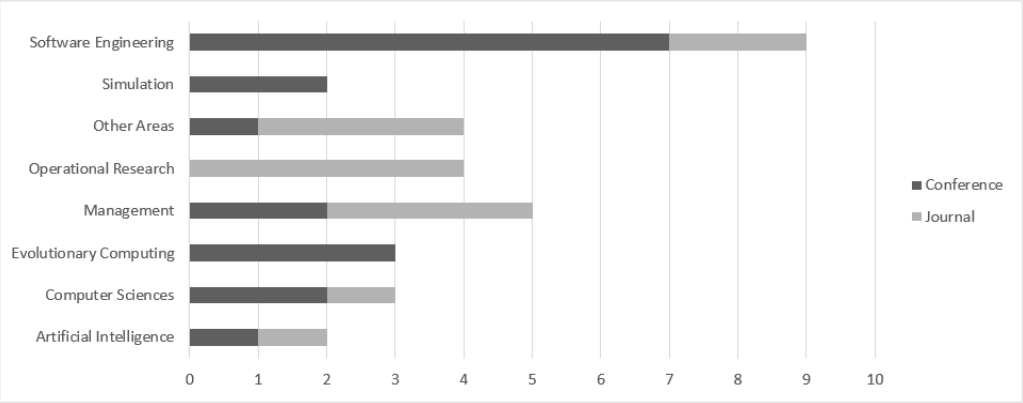


Fonte: Elaborado pelo Autor

Os 32 artigos analisados ficaram divididos entre publicações em conferências (18) e periódicos (14). Em relação a área de estudo dos periódicos e conferências, pode-se perceber que a maioria dos artigos são publicados no tema “engenharia de

software". A Figura 5 apresenta a distribuição dos trabalhos selecionados em razão do tipo e da área do periódico. Foram agrupados ainda como "outras áreas" alguns estudos publicados em revistas de tecnologia voltadas para engenharia aeroespacial (HUNT, 2007), telecomunicações (RAMZAN et. al., 2012), indústria (ARTIGUES, LEUS, NOBIBON, 2013) e economia (DREZET, BILLAUT, 2008).

Figura 5 - Distribuição da área de estudo dos periódicos e conferências

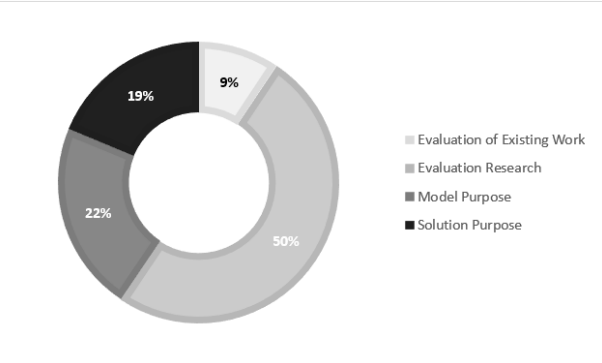


Fonte: Elaborado pelo Autor

3.4.1 RQ1: Quais os tipos de estudos realizados na pesquisa de predição e recomendação de cronogramas dentro do gerenciamento de tempo?

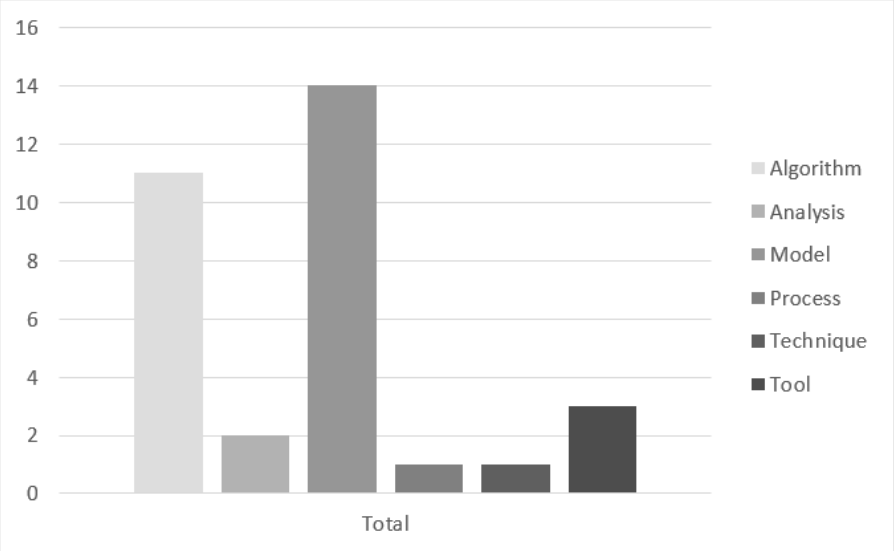
A primeira questão de pesquisa aborda os tipos de estudo que estão sendo realizados em relação a predição de cronogramas no gerenciamento de projetos, cuja distribuição pode ser visualizada através da Figura 6. Além do formato do estudo, foi também analisada a contribuição que cada um traz em sua pesquisa, conforme Figura 7.

Figura 6 - Classificação dos artigos de acordo com pesquisa proposta



Fonte: Elaborado pelo Autor

Figura 7 - Classificação dos artigos de acordo com a contribuição



Fonte: Elaborado pelo Autor

A maior parte dos trabalhos encontrados contribui com modelos que suportem a predição de cronogramas em projetos de software, onde alguns destes conseguiram apresentar implementações práticas que deram suporte à validação do modelo. Entretanto, 50% dos trabalhos foram classificados como “validação de pesquisa” – ou seja, trouxeram uma contribuição inovadora, porém com nível baixo de evidências que dessem suporte à validação do modelo em cenários reais. A Tabela 4 categoriza cada um dos estudos em relação aos critérios previamente apresentados, utilizando o identificador de cada artigo conforme apresentado na Tabela 3.

Tabela 4 - Artigos selecionados de acordo com a pesquisa apresentada

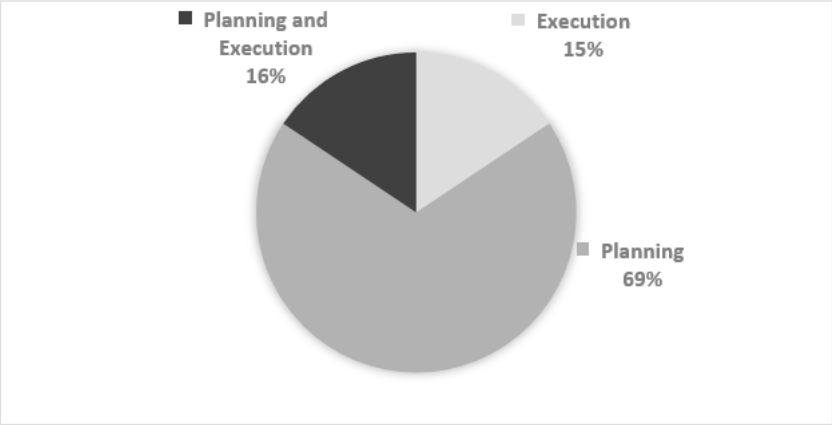
Classificação do Estudo	Artigos	Qtd
Validação de trabalhos existentes	P6, P16, P17	3
Proposta de modelo	P3, P11, P23, P26, P27, P29, P30	7
Proposta de solução	P1, P4, P8, P14, P31, P32	6
Validação de pesquisa	P2, P5, P7, P9, P10, P12, P13, P15, P18, P19, P20, P21, P22, P24, P25, P28	16

Fonte: Elaborado pelo Autor

3.4.2 RQ2: Em quais etapas do ciclo de gerenciamento de projetos estão focados os estudos de predição e recomendação de cronogramas?

O ciclo de gerenciamento de projetos, conforme descrito pelo PMI, possui 5 fases: abertura do projeto, planejamento, execução, monitoramento e controle, e encerramento do projeto. A questão 2 tem como objetivo identificar em quais destas fases estão os focos dos estudos de predições e recomendações de cronograma. Considerando que as fases de abertura e encerramento não se relacionam diretamente com o cronograma, e que as fases de execução e monitoramento e controle são complementares entre si, os estudos podem ser caracterizados como exclusivamente atuantes na fase de planejamento, na fase de execução/monitoramento/controle, ou em ambas, conforme demonstrado na Figura 8. Para maior facilidade de compreensão, a fase de execução, monitoramento e controle será nomeada apenas “execução” no restante deste trabalho.

Figura 8 - Classificação dos artigos por etapa do ciclo de gerenciamento



Fonte: Elaborado pelo Autor

Percebeu-se através desta análise o foco principal das publicações na fase de planejamento, totalizando 85% ao considerarmos todos os artigos que a abordam de modo exclusivo ou não, conforme demonstrado na Tabela 5. Em sua grande maioria, os trabalhos têm como objetivo a geração de um cronograma completo do projeto, com base em uma listagem de tarefas e uma listagem de recursos, abordando assim apenas o plano inicial do projeto.

Tabela 5 - Artigos categorizados pela etapa do ciclo de gerenciamento

Fases Abordadas nas Pesquisas	Artigos	Qtd
Planejamento	P1, P2, P4, P7, P8, P9, P10, P11, P12, P13, P16, P17, P18, P20, P21, P22, P23, P25, P28, P29, P30, P31	22
Execução	P6, P15, P24, P27, P32	5
Planejamento e Execução	P3, P5, P14, P19, P26	5

Fonte: Elaborado pelo Autor

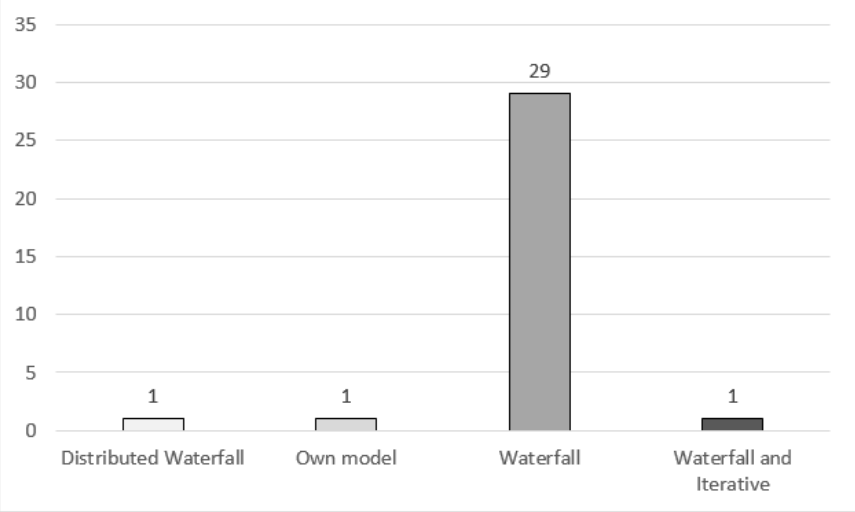
A fase de execução do projeto apresenta apenas 10 estudos relacionados. Nestes artigos, nota-se uma preocupação com o andamento do projeto e as incertezas que permeiam o desenvolvimento de software. A falta de assertividade nas estimativas e a constante necessidade de mudanças foram percebidos como temas recorrentes nestes artigos, pois não são consideradas na fase de planejamento. Assim, percebe-se uma necessidade estudos com o objetivo de acompanhar o

andamento dos projetos, para garantir que os cronogramas gerados na fase de planejamento sejam seguidos, bem como permitir a adaptação destes cronogramas aos imprevistos que podem ocorrer.

3.4.3 RQ3: Quais os modelos de projeto de software são abordados nos estudos de predição e recomendação de cronogramas?

A terceira questão de pesquisa teve como objetivo verificar quais são os modelos de projeto de software considerados nos estudos, de acordo com a Figura 9. Os principais modelos utilizados no mercado atualmente são o modelo Cascata, que aborda o projeto inteiro de uma só vez, com uma única entrega ao final do processo, e o modelo Iterativo-Incremental, utilizados nas metodologias ágeis, que prega a necessidade de realizar pequenas entregas visando melhor aceitação e avaliação dos stakeholders ao longo do projeto. Algumas empresas utilizam métodos diferentes, como o RUP (*Rational Unified Process*), enquanto outras desenvolvem sua própria metodologia para gestão.

Figura 9 - Classificação dos artigos por metodologias de gestão abordada



Fonte: Elaborado pelo Autor

Todos os trabalhos apresentados suportavam, de alguma maneira, modelos de desenvolvimento em cascata, conforme categorizado na Tabela 6. Porém, dos 32 artigos avaliados, apenas 1 deles trabalhava com abordagens iterativas e incrementais, através de métodos ágeis como SCRUM (BAIA, 2015). Percebe-se através desta análise uma grande oportunidade de pesquisa em relação a metodologias iterativas e incrementais, visto que vem sendo cada vez mais aplicadas no cenário atual do mercado.

Tabela 6 - Artigos classificados pela metodologia de gestão abordada

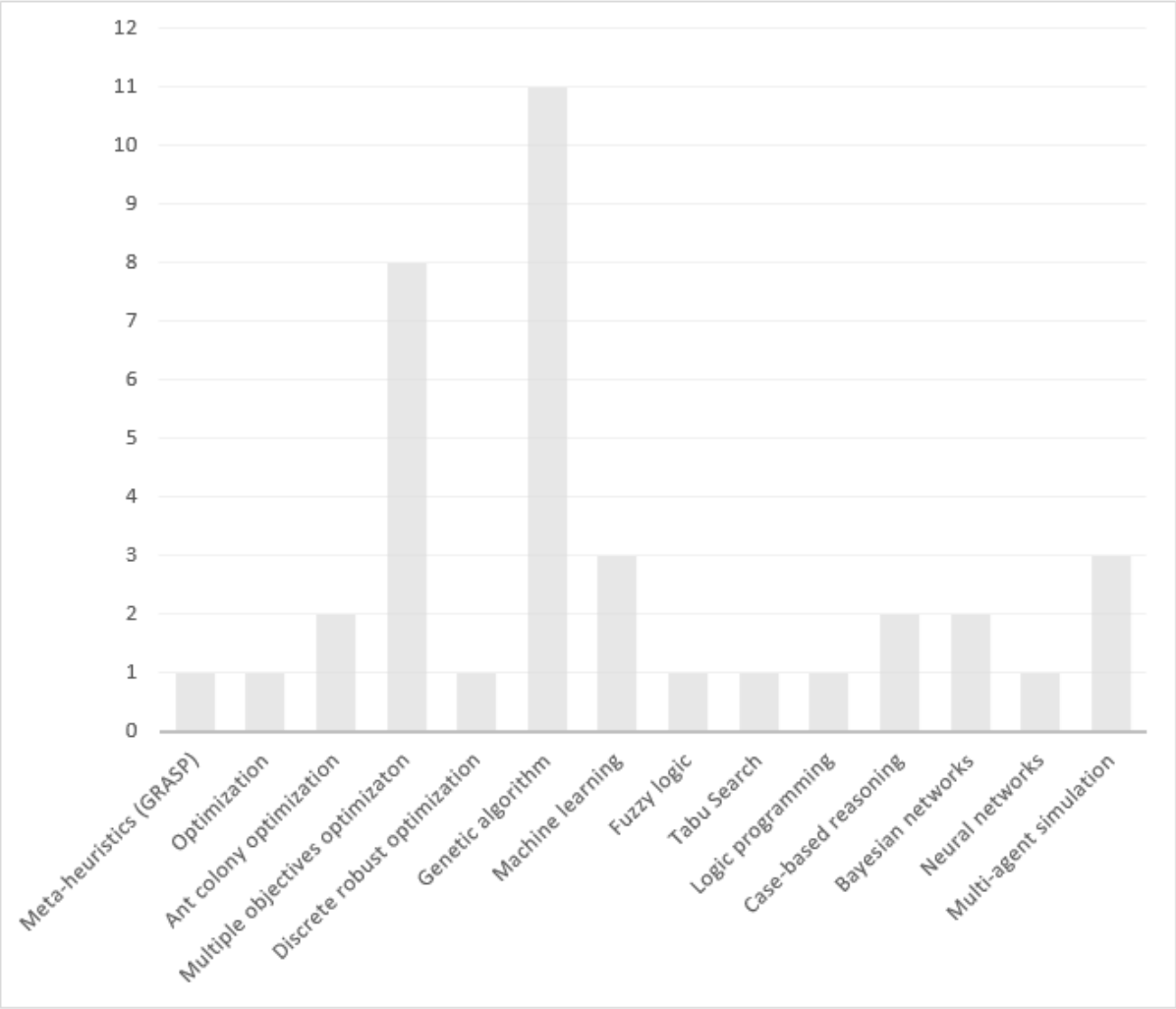
Metodologia	Artigos	Qtd
Cascata	P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16, P17, P18, P19, P20, P21, P22, P23, P24, P25, P27, P28, P29, P32	29
Cascata com equipes distribuídas	P31	1
Cascata e/ou iterativo-incremental	P26	1
Modelo próprio	P30	1

Fonte: Elaborado pelo Autor

3.4.4 RQ4: Quais as técnicas utilizadas para predição e recomendação de cronogramas encontradas na literatura?

A quarta e última questão de pesquisa tem como meta a identificação das tecnologias, métodos e/ou algoritmos que são utilizados no objetivo de prever ou recomendar informações referentes ao cronograma de projetos. Inúmeras técnicas foram encontradas nos trabalhos avaliados, conforme exibido na Figura 10. Considerando a complexidade do problema, percebe-se que muitos dos estudos utilizam mais de uma técnica ao tentar solucionar o problema, e por este motivo cada artigo pode apresentar mais de uma técnica relacionada nesta análise.

Figura 10 - Classificação dos artigos por técnicas aplicadas na predição



Fonte: Elaborado pelo Autor

Ao verificar os dados analisados, percebe-se uma tendência ao uso de algoritmos de otimização de múltiplos objetivos e também a algoritmos genéticos. Estes temas foram abordados em 8 e 10 estudos respectivamente, e foram utilizados em conjunto em 5 destes. Pode-se ainda verificar que nos últimos anos, novas técnicas foram utilizadas com resultados significativos, sendo elas a aprendizagem de máquina e a lógica Fuzzy. A Tabela 7 mostra a distribuição das técnicas utilizadas em cada artigo.

Tabela 7 - Artigos classificados pelo método ou técnica abordados

Metodologia	Artigos	Qtd

Algoritmo de meta-heuristica (GRASP)	P13	1
Algoritmo de otimização	P30	1
Algoritmo de otimização de colônia de formigas	P24, P25	2
Algoritmo de otimização de múltiplos objetivos	P9, P15, P16, P17, P20, P21, P25, P29	8
Algoritmo de otimização discreta robusta	P22	1
Algoritmos genéticos	P3, P8, P9, P12, P15, P18, P20, P21, P23, P29, P31	11
Aprendizagem de máquina	P14, P27, P32	3
Lógica fuzzy com conhecimento de especialista	P28	1
Pesquisa Tabu	P10	1
Programação lógica	P1	1
Raciocínio baseado em casos	P7, P11	2
Redes Bayesianas	P4, P14	2
Redes neurais	P2	1
Simulação multi-agentes	P5, P19, P26	3

Fonte: Elaborado pelo Autor

3.5 Trabalhos selecionados

Com base no estudo de mapeamento sistemático realizado, que resultou em uma análise com 32 trabalhos relacionados ao tema, foram aplicados novos critérios de triagem para a realização de um estudo comparativo entre o estado da arte e o modelo Kairós. Para esta nova seleção, 24 trabalhos que não apresentavam uma proposta de modelo ou que apresentavam estudos em fase preliminar, sem aprofundamento ou validação foram excluídos (BERTAZZONI, GIANNOTI, 1990; ZHANG et. al., 2003; HEGAZY, PETZOLD, 2003; JOSLIN, POOLE, 2005; HUNT, 2007; YANG, WANG, 2008; GONÇALVES, MENDES, RESENDE, 2008; DREZET, BILLAUT, 2008; YANG, WANG, 2009; ASHTIANI, LEUS, ARYANEZHAD, 2009; BALLESTÍN, LEUS, 2009; WANG, WU, MA, 2010; XIAO et. al., 2010; CHICANO et. al., 2011; BALLESTÍN, BLANCO, 2011; KAROVA, AVRAMOVA, 2012; CRESPO, RUIZ, 2012; RAMZAN et. al., 2012; CHICANO et. al., 2012; ARTIGUES, LEUS, NOBIBON, 2013; ZHANG et. al., 2015; XIAO, GAO, HUANG, 2015; NIGAR, 2017;

KROLL, FRIBOIM, HEMMATI, 2017). Entre os 8 estudos restantes, 2 destes foram unificados por fazerem parte do mesmo projeto (CHOETKIERTIKUL et. al., 2015; CHOETKIERTIKUL et. al., 2017), finalizando em 7 modelos para a realização do comparativo (FENTON et. al., 2004; CHANG et. al., 2008; MINKU, SUDHOLT, YAO, 2014; BAÍA, 2015; LOPEZ-MARTIN, CHAVOYA, MEDA-CAMPANA, 2015; XIONG et. al., 2016; CHOETKIERTIKUL et. al., 2017).

3.5.1 MODIST - *Making Resource Decisions for Software Projects*

O MODIST, modelo apresentado por Fenton et. al. (2004), tem o objetivo de sugerir a alocação de recursos em um cronograma de projeto utilizando múltiplas redes bayesianas. Cada rede bayesiana do modelo é responsável por um aspecto relevante do projeto, da tarefa ou do recurso em questão, como a qualidade da equipe, duração do projeto, complexidade do desenvolvimento necessário, entre outros. Através destas redes, o modelo permite o uso de inferências para prever o resultado de uma ou mais variáveis do modelo.

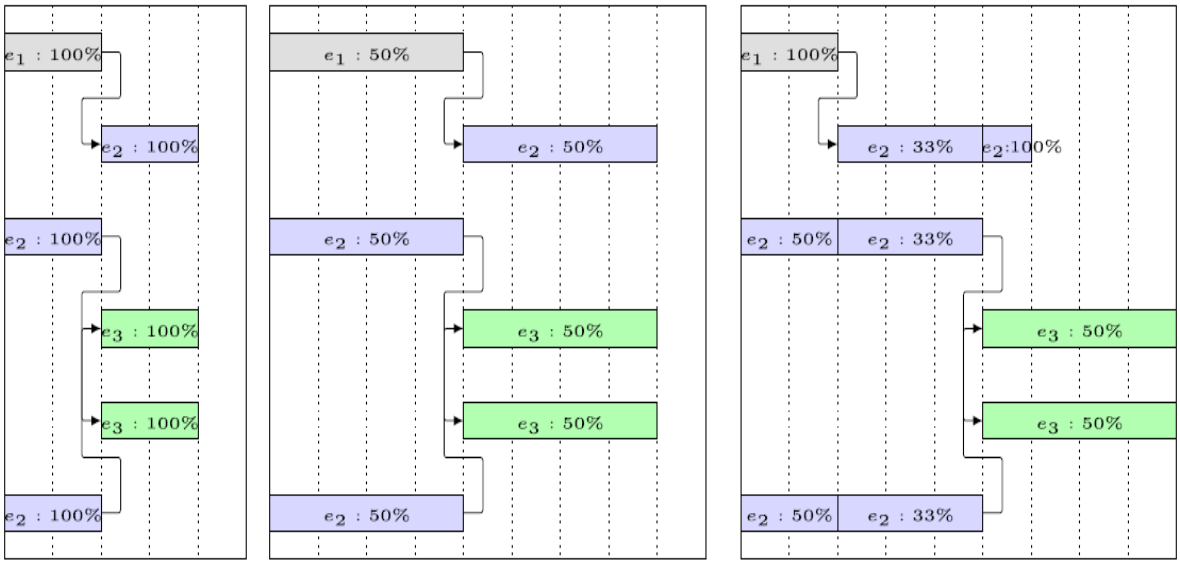
3.5.2 *Time-line based model for software project scheduling with genetic algorithms*

Este trabalho apresenta um modelo baseado em algoritmos genéticos de otimização, com foco na geração de um modelo de cronograma completo a partir de submodelos de tarefas, habilidades e funcionários, na fase de planejamento do projeto. Cada um destes submodelos possui suas próprias características, e evoluem dentro de suas próprias funções heurísticas para a composição de um modelo maior, que corresponde ao cronograma completo (CHANG et. al., 2008).

3.5.3 *Improved Evolutionary Algorithm Design for the Project Scheduling Problem Based on Runtime Analysis*

O modelo apresentado neste estudo utiliza algoritmos genéticos para a elaboração de um cronograma de projeto tradicional (MINKU, SUDHOLT, YAO, 2014). O algoritmo é executado em múltiplas etapas, iniciando pelo ajuste do tempo de dedicação diária de cada funcionário, permitindo que sejam configuradas horas extras em algumas tarefas que trariam maior benefício ao projeto. Com os resultados, aplica mutações e refaz as buscas, gerando novos cronogramas até que encontre o resultado ótimo, conforme demonstrado na Figura 11.

Figura 11 - Etapas da evolução do cronograma pelo algoritmo genético aplicado

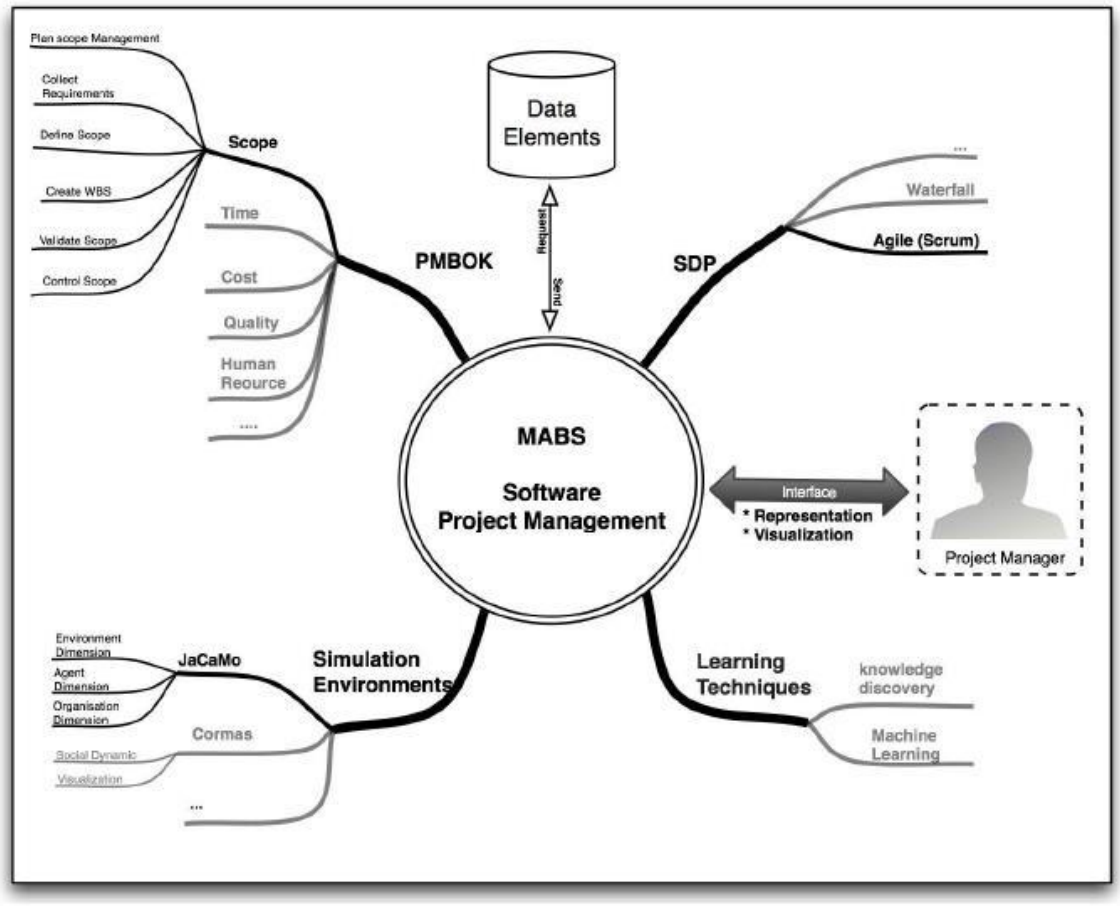


Fonte: MINKU, SUDHOLT, YAO, 2014

3.5.4 *An Integrated Multi-Agent-Based Simulation Approach to Support Software Project Management*

O objetivo do estudo apresentado por Baía (2015) é auxiliar o gestor do projeto durante todo o ciclo de vida, através de um modelo integrado e adaptável que alia técnicas de simulação e mecanismos de aprendizado, que pode ser visualizado na Figura 12. Além disso, um dos diferenciais deste estudo é o foco na área de desenvolvimento de software, abordando modelos de desenvolvimento tradicionais e ágeis.

Figura 12 - Modelo apresentado por Baía



Fonte: BAÍA, 2015.

3.5.5 *A Fuzzy Logic Model for Predicting the Development Schedule of Software Projects*

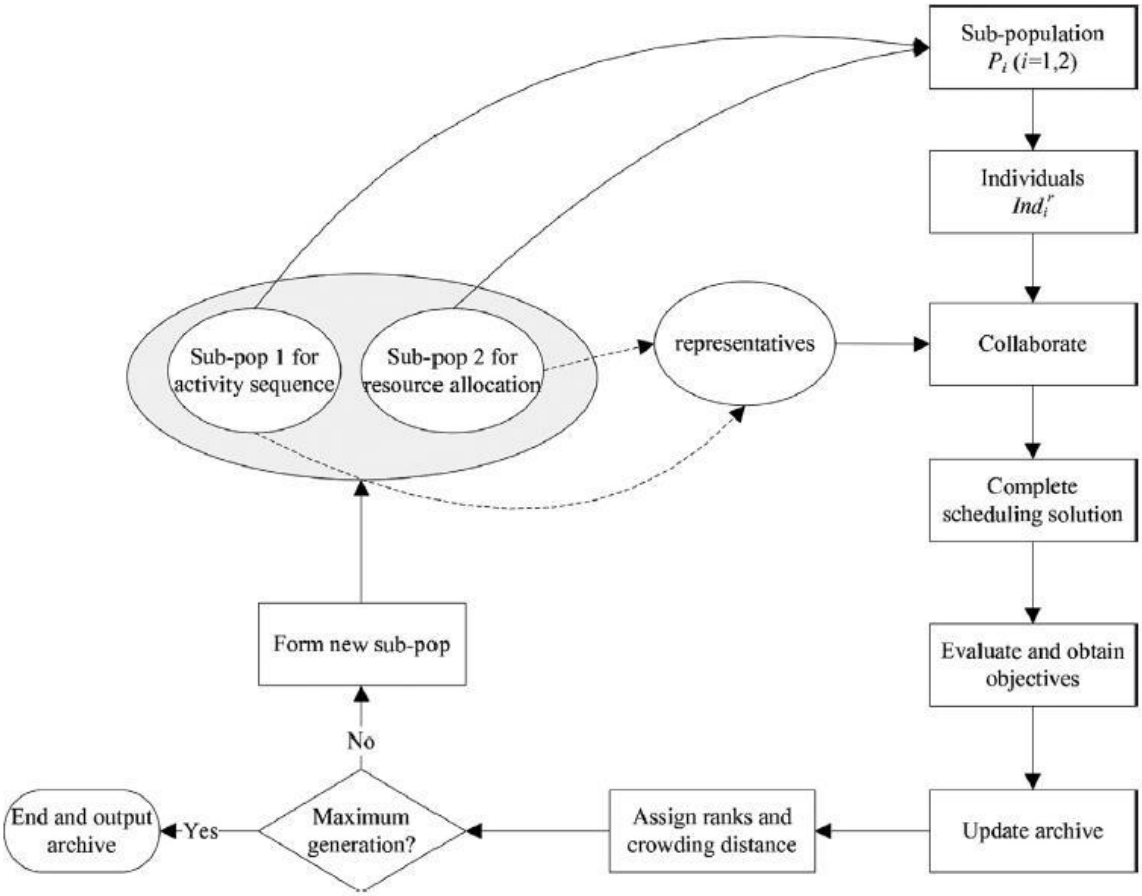
Neste trabalho, é proposta a criação de um modelo capaz de criar o cronograma de um projeto de software utilizando lógica *fuzzy*. Este tipo de algoritmo funciona através de uma série de regras do tipo “se-então” geradas a partir do conhecimento de um especialista, e é capaz de aplicar modelos matemáticos a estas regras para lidar com incerteza. No modelo apresentado, a principal variável é a estimativa por pontos de função, e a partir da correlação deste dado com a base utilizada para alimentar o modelo, é possível visualizar o cronograma do projeto (LOPEZ-MARTIN, CHAVOYA, MEDA-CAMPANA, 2015).

3.5.6 CCMOA - *Evolutionary multi-objective resource allocation and scheduling in the Chinese navigation satellite system project*

Este modelo foi construído para auxiliar na gestão de projetos de pesquisa em uma empresa chinesa, com foco em sistemas de navegação por satélite. O estudo apresenta o desenvolvimento de um modelo de múltiplos objetivos, considerando a alocação de recursos e o sequenciamento das tarefas do projeto, com objetivo de otimizar a minimização da janela de duração do projeto e maximização de balanceamento de recursos (XIONG et. al., 2016).

Para resolver estes problemas, foi utilizada uma abordagem com algoritmos cooperativos coevolucionários, que funcionam como múltiplos algoritmos genéticos em paralelo com objetivos específicos. Nesta aplicação, a população foi dividida em 2 subpopulações, onde uma delas corresponde às tarefas e a outra aos recursos disponíveis para alocação, conforme Figura 13.

Figura 13 - Visão geral do modelo CCMOA



Fonte: XIONG et. al., 2016

3.5.7 *Predicting the delay of issues with due dates in software projects*

O modelo elaborado por Choetkiertikul et. al. (2017) é uma evolução do modelo elaborado pelo mesmo grupo alguns anos antes (2015), e tem como objetivo a predição de atrasos em tarefas durante a fase de execução do projeto. O estudo apresenta a utilização de algoritmos de aprendizagem de máquina considerando 19 variáveis diferentes, extraídas através de uma análise de mais de 60.000 tarefas obtidas de projetos *open source* de grande porte como Apache, JIRA, JBoss, entre

outros. Ao realizar a predição, o modelo ainda é capaz de apontar a probabilidade percentual de ocorrência do problema.

3.5.8 Comparativo

Após a análise dos trabalhos relacionados, foi elaborado um comparativo entre estes modelos e o Kairós. A Tabela 8 resume os critérios utilizados para comparação, considerando a fase do ciclo de vida de projeto em que se aplica, para quais modelos de gestão de projetos está adaptado, quais as técnicas de predição utilizadas, se é capaz de realizar recomendações para solução do problema e se apresenta alguma proatividade ao ser utilizado.

Tabela 8 - Comparativo dos trabalhos relacionados

Referência	Fase de aplicação	Modelo de gestão	Técnica de predição	Recomendação	Proatividade
FENTON et. al., 2004	Planejamento	Tradicional	Redes Bayesianas	Não	Não
CHANG et. al., 2008	Planejamento	Tradicional	Algoritmos genéticos	Não	Não
MINKU, SUDHOLT, YAO, 2014	Planejamento	Tradicional	Algoritmos genéticos	Não	Não
BAIA, 2015	Planejamento e Execução	Tradicional e ágil	Simulação e aprendizado de máquina	Sim	Não
LOPEZ-MARTIN, CHAVOYA, MEDA-CAMPANA, 2015	Planejamento	Tradicional	Lógica Fuzzy com conhecimento de especialista	Não	Não
XIONG et. al., 2016	Planejamento	Tradicional	Algoritmo de otimização de multiplos objetivos e algoritmos genéticos	Não	Não
CHOETKIERTIKUL, 2017	Execução	Tradicional	Aprendizagem de máquina	Não	Sim
Modelo Kairós	Execução	Tradicional e ágil	Similaridade semântica e predição por históricos de contextos	Sim	Sim

Fonte: Elaborado pelo Autor

Foi possível perceber gaps de pesquisa: a realização de predições atualizadas durante o andamento do projeto; a predição de cronograma em modelos de desenvolvimento ágeis; a proatividade em realizar recomendações para os gestores durante a fase de monitoramento; e o uso de tecnologias com maior nível de inteligência para a realização das predições.

A maioria dos artigos identificados tem como foco a fase de planejamento do projeto, em detrimento da fase de execução. Visto que a fase de execução é onde ocorrem efetivamente os problemas, percebe-se uma oportunidade de pesquisa neste cenário.

Ainda, percebe-se também uma preferência por modelos de desenvolvimento em cascata, o que corrobora com a preocupação previamente exposta com a fase de planejamento. Em modelos ágeis, iterativos e incrementais, o planejamento ocorre em ciclos dentro do projeto, dando mais importância a execução do projeto em si e a qualidade do produto, do que aos contratos e necessidade de planejamento a longo-prazo dos projetos em cascata.

No que se refere ao modelo das ferramentas desenvolvidas para predição, não foram encontradas evidências de recomendações proativas sobre o cronograma, necessitando sempre uma intervenção ou consulta manual realizada pelo gestor. Em cenários como o da fase de execução do projeto, onde existe um alto fluxo de atualização das informações, os sistemas de recomendação se sobressaem, provendo dados atualizados que não apenas auxiliem na tomada de decisão, mas que também alertem para riscos e situações críticas em tempo real, antes que se tornem mais graves.

Ao tentar solucionar os problemas referentes aos cronogramas, diversas tecnologias e métodos foram consideradas, com destaque para algoritmos de otimização de múltiplos objetivos e algoritmos genéticos. Entretanto, ao analisar tecnicamente as soluções sugeridas, percebe-se que a maior parte delas aborda algoritmo simples de busca com heurísticas básicas, com um foco maior na performance do que na qualidade do resultado obtido. Considerando as tendências mais recentes de inteligência artificial que vem sendo aplicadas em diversas áreas, percebe-se que tecnologias como *deep learning*, predição por algoritmos de similaridade e predição por históricos de contextos ainda não foram aplicados a esta área.

3.6 Considerações

O gerenciamento de projetos é um tema que permanece no foco de pesquisa com o passar dos anos, dada a ampla aplicabilidade de seus conceitos nas mais distintas áreas. A gestão adequada do tempo dentro do projeto é um fator crítico para o sucesso ou a falha, e a complexidade de manter todos os membros da equipe dentro do planejamento é fato indiscutível. Nos projetos de desenvolvimento de software, este fator é ainda mais evidente dada a natureza dos resultados deste tipo de projeto, visto que os *stakeholders* costumam não deter total conhecimento do domínio nas fases de abertura e planejamento do projeto.

A execução deste estudo de mapeamento sistemático teve como principal objetivo a possibilidade de visualizar de maneira ampla e global os focos de pesquisa em relação às predições e recomendações de cronograma em projetos de software. Através das *strings* pesquisadas, baseadas nas questões de pesquisa elaboradas, foram identificados artigos que respondessem a estas questões.

Através da análise comparativa, foi possível perceber que o modelo Kairós possui um conjunto de características único em relação aos demais. Por ser adaptável a modelos tradicionais e ágeis de gestão, demonstra ser adaptável a uma maior variedade de cenários.

O modelo também se diferencia por ser aplicado na fase de execução, e com isso considera todas as alterações de curso que ocorrem durante o ciclo de vida do projeto, além de trabalhar com as informações em tempo real. Ainda, apresenta as informações de maneira proativa ao gestor do projeto, auxiliando nos cenários complexos onde existem grandes quantidades de informações que necessitam ser avaliadas simultaneamente.

Outro diferencial é o fato de que não apresenta apenas os possíveis problemas, mas traz recomendações para solucioná-los baseadas em melhores práticas de projeto e no conhecimento dos próprios gerentes. Por fim, aplica técnicas de análise de similaridade semântica e de predição por históricos de contextos, considerando dados reais de projetos da própria empresa em que está sendo aplicado, para gerar predições e recomendações mais próximas da realidade.

4 MODELO KAIRÓS

O modelo Kairós é descrito neste capítulo. A seção 4.1 apresenta uma visão geral do modelo proposto e suas funcionalidades. Na seção 4.2, serão apresentados os requisitos e alguns diagramas UML elaborados na análise do projeto. A arquitetura básica do modelo será demonstrada na seção 4.3, juntamente com alguns conceitos sobre as tecnologias que serão utilizadas. Os agentes integrantes do modelo serão apresentados na seção 4.4. Finalmente, a seção 4.5 traz considerações sobre o desenvolvimento do modelo e sua evolução.

4.1 Estruturação do modelo

Este estudo propõe o modelo Kairós para auxílio ao gerente de projetos no controle dos cronogramas. O batismo do modelo se deu em referência ao deus grego homônimo, que representava a natureza qualitativa do tempo, as oportunidades e os momentos especiais.

O Kairós atua de maneira proativa na fase de execução, recomendando ações corretivas ou pontos de atenção a cada modificação do contexto do projeto. Tais recomendações são realizadas utilizando técnicas de predição de contexto, analisando dados históricos do próprio projeto em andamento e de outros projetos da empresa em questão.

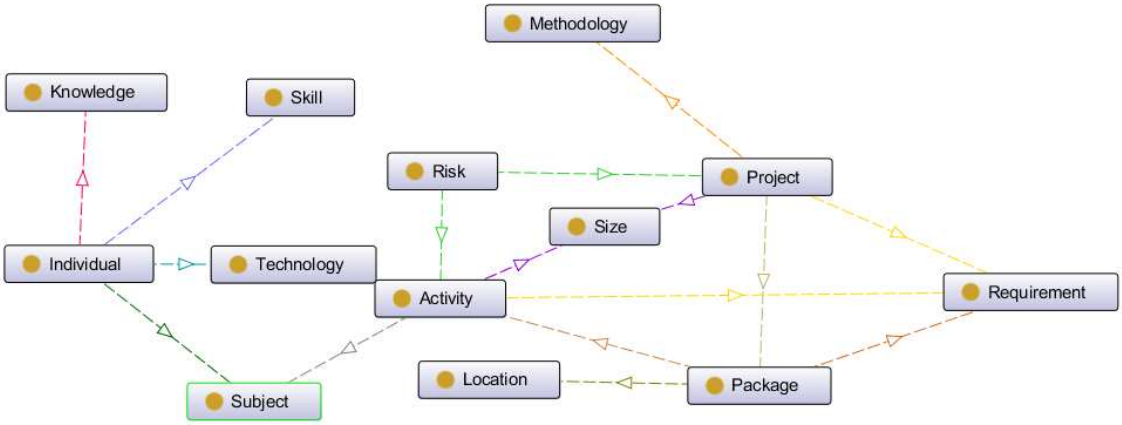
De acordo com Dey, Salber e Abowd (2001), um contexto computacional é uma representação das informações que caracterizam o estado de uma entidade relevante. No Kairós são armazenados dados contextuais de duas entidades, sendo elas a “tarefa” e o “projeto”. Visto que o Kairós tem como objetivo a realização de predições sobre o estado futuro das tarefas de um projeto, o contexto da tarefa será o ponto central do modelo. Já o contexto do projeto será usado principalmente para realizar a correlação entre os projetos, indicando o quanto um projeto é similar ao outro, e com isso permitindo dar maior ou menor importância para as tarefas deste projeto.

Com a intenção de atuar de maneira proativa, o modelo está preparado para atuar a cada alteração no contexto de tarefas e projetos. O contexto representa todas

as informações relacionadas ao estado da tarefa ou projeto em um determinado momento no tempo.

Para a representação das entidades do modelo, foi elaborada a ontologia mostrada na Figura 14. Nesta ontologia, foram definidas as principais informações relevantes no contexto do cronograma do projeto, considerando todos os itens que podem afetá-lo de alguma maneira. Além disso, também foram definidas as características críticas do projeto e da tarefa, bem como as relações entre estas. Posteriormente, estes dados serão utilizados pelos mecanismos de análise e predição.

Figura 14 – Ontologia representando o domínio do cronograma de projeto



Fonte: Elaborado pelo Autor

A tarefa, objeto principal do modelo, é representada pela classe *Activity*, que possui características como tamanho, pacote e localização. Cada atividade possui relação com as tecnologias utilizadas em seu desenvolvimento, aos requisitos que a tarefa representa e aos riscos envolvidos em sua execução. As tarefas ainda possuem um ou mais recursos responsáveis por seu desenvolvimento, representados através da classe *Individual*. Cada recurso tem uma série de habilidades, conhecimentos e experiência em tecnologias, o que indica se está apto ou não para desenvolver determinada tarefa. Ainda, o projeto também é representado através da classe

Project, que possui as mesmas relações que as tarefas, porém de maneira global – ou seja, representando todas as tarefas do projeto.

Qualquer alteração de dados como estado, prazo, estimativa, requisitos, etc., registro de andamento, ou até mesmo uma alteração no recurso definido para a tarefa, deverá disparar os gatilhos de contexto. A cada disparo, o modelo deverá armazenar o estado atual na base de históricos de contextos e iniciar os mecanismos de predição. Caso necessário, poderá recomendar que seja tomada alguma ação corretiva.

As ações corretivas sugeridas têm duas fontes principais. A fonte inicial é uma base de dados pré-estabelecida, que é preenchida com base em melhores práticas de projetos. Cada ação corretiva cadastrada nesta base estará vinculada a um ou mais tipos de problema, como atraso na conclusão da tarefa, impacto negativo na execução das tarefas dependentes, entre outros.

Durante o andamento do projeto, o próprio usuário poderá informar novas ações corretivas que atendam às suas necessidades. Ao receber uma notificação indicando que determinada tarefa possui um problema, o gestor poderá rejeitar as recomendações realizadas e informar qual será a ação corretiva que ele tomará neste caso. Dessa forma, será criada uma segunda fonte de recomendações, contendo aquelas sugeridas pelos próprios usuários.

Ainda, ao receber uma recomendação, independente da origem, o usuário poderá classificar as recomendações de acordo com a utilidade na solução do problema. Estas classificações serão avaliadas ao realizar novas recomendações, fazendo com que as consideradas mais úteis sejam mais recomendadas.

Com o objetivo de facilitar a aplicação do modelo de predição em cenários variados, o Kairós é adaptável aos principais formatos de gestão de projetos utilizados atualmente, sendo eles o formato clássico, abordado pelo PMBOK, e o formato ágil, abordado por metodologias como o SCRUM. Estes formatos possuem características conceituais distintas, mas ambos têm margem para aplicação de um modelo de predição sobre o controle do tempo na execução das tarefas do projeto. Para implementar a adaptabilidade a modelos de gestão tradicionais e ágeis, cada sprint é considerado um cronograma fechado. Dessa forma, todas as tarefas possuem também suas estimativas, previsões de início e fim, e assim se encaixam no modelo geral.

Para exemplificar o funcionamento do modelo, a Figura 15 mostra um cronograma de um projeto simples em andamento. Analisando o cronograma como um todo, percebe-se que não existem problemas na elaboração do mesmo, visto que todas as tarefas têm um início e fim definidos, o prazo de conclusão está dentro do limite do projeto, e os recursos não estão sobrecarregados.

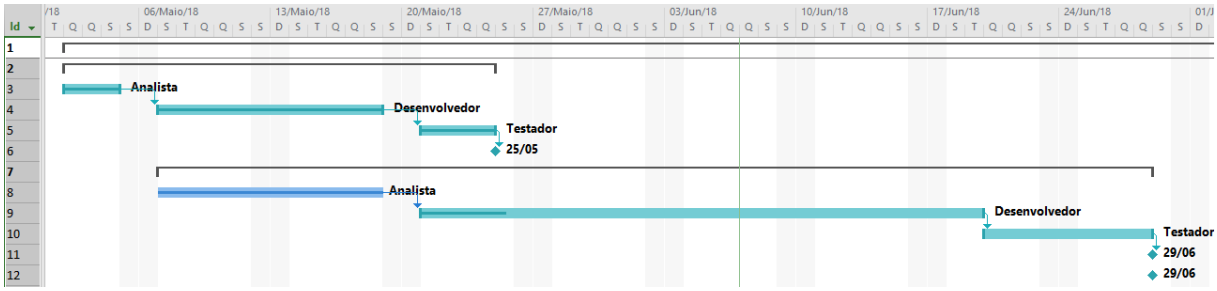
Figura 15 - Projeto em andamento

Id	Nome da Tarefa	Dura	Início	Término	Predecessoras	Nomes dos recursos
1	**Início do desenvolvimento**	**45 dias**	**Qua 02/05/18**	**Ter 03/07/18**		
2	**Requisito RF1**	**17 dias**	**Qua 02/05/18**	**Sex 25/05/18**		
3	Análise do requisito RF1	3 dias	Qua 02/05/18	Sex 04/05/18		Analista
4	Desenvolvimento do requisito RF1	10 dias	Seg 07/05/18	Sex 18/05/18	3	Desenvolvedor
5	Teste do requisito RF1	4 dias	Seg 21/05/18	Qui 24/05/18	4	Testador
6	Entrega do requisito RF1	0 dias	Sex 25/05/18	Sex 25/05/18	5	Implantação
7	**Requisito RF2**	**39 dias**	**Seg 07/05/18**	**Sex 29/06/18**		
8	Análise do requisito RF2	10 dias	Seg 07/05/18	Sex 18/05/18		Analista
9	Desenvolvimento do requisito RF2	22 dias	Seg 21/05/18	Ter 19/06/18	8	Desenvolvedor
10	Teste do requisito RF2	7 dias	Qua 20/06/18	Qui 28/06/18	9	Testador
11	Entrega do requisito RF2	0 dias	Sex 29/06/18	Sex 29/06/18	10	Implantação
12	Conclusão do projeto	0 dias	Sex 29/06/18	Sex 29/06/18		

Fonte: Elaborado pelo Autor

Ao analisar o diagrama de Gantt na Figura 16, que ilustra o andamento das tarefas do projeto em um eixo temporal, é possível perceber uma situação preocupante em relação à tarefa 9 – Desenvolvimento do requisito RF2. Considerando que o gráfico tenha sido analisado no dia 07 de junho, o andamento da tarefa está bastante atrasado em relação ao seu prazo estimado. A tarefa teve início em 21 de maio, com estimativa de 22 dias úteis para a conclusão, resultando na data de 19 de junho. Portanto, se no dia 07 de junho o percentual de conclusão era de aproximadamente 20%, a tarefa tem uma boa chance de não ser concluída no prazo, visto que o percentual esperado para o dia 7 seria de aproximadamente 60%.

Figura 16 - Diagrama de Gantt



Fonte: Elaborado pelo Autor

No momento em que o recurso “Desenvolvedor” registrar o andamento da tarefa em 20% nesta data, será disparada uma nova análise sobre esta tarefa. Verificando a base de históricos de contextos, o sistema poderá encontrar inúmeras tarefas para comparação. Estas comparações com os dados anteriormente armazenados na base consideram semelhanças entre os dados estáticos da tarefa que está sendo alterada, dados estáticos do projeto, e dado de evolução histórica da tarefa – ou seja, os históricos do contexto da tarefa.

Ao encontrar tarefas com características e históricos semelhantes, e que já tenham sido concluídas, é possível realizar uma predição por similaridade – ou seja, se duas tarefas têm características semelhantes e tiveram uma evolução semelhante ao longo do tempo, provavelmente terão também um final similar. O modelo utiliza esta técnica para então verificar o estado final das tarefas com maior similaridade, considerando se houve atrasos no prazo ou excesso nos custos.

Para auxiliar a tomada de decisão, o modelo segue uma estratégia de pesos entre as análises realizadas. Estes pesos podem ser configurados manualmente para o início do uso da ferramenta, e o próprio modelo poderá alimentar estes pesos com base na aceitação das recomendações realizadas.

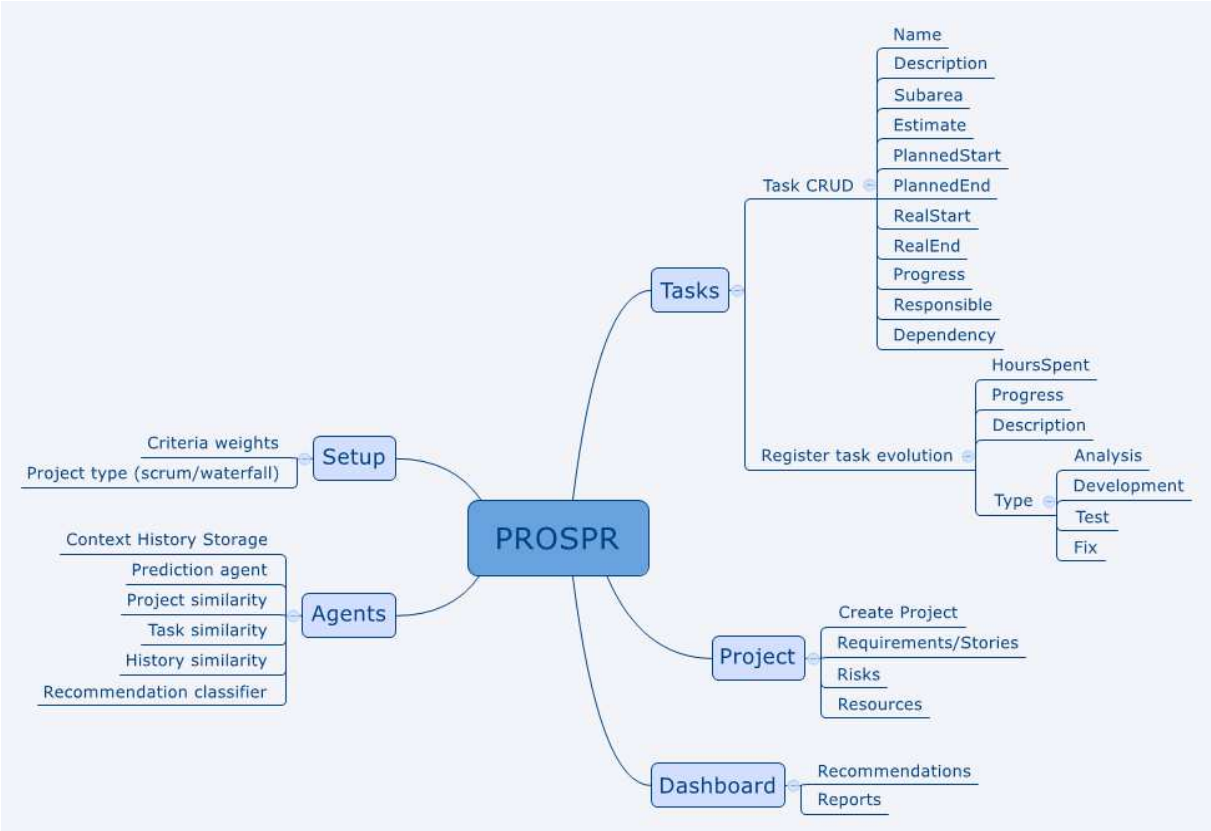
Supondo uma situação em que o Kairós encontre várias outras tarefas de diferentes projetos que apresentaram um histórico de andamento (ou seja, a evolução da tarefa no eixo temporal) semelhante, e todas tenham sido concluídas após o prazo. Neste caso, o peso desta variável poderá indicar que, mesmo que sejam tarefas com pouca semelhança em projetos variados, seja feita uma recomendação de possibilidade de atraso na tarefa.

Dessa forma, o modelo permite uma boa adaptação ao cenário da empresa, visto que a base de dados utilizada na análise é da própria empresa. Além disso, permite que a ferramenta seja aprimorada com o tempo, já que a base de dados para análise será cada vez mais completa e adequada ao uso.

4.2 Modelo conceitual

O modelo Kairós foi inicialmente definido através de um mapa mental, conforme Figura 17, contendo funcionalidades básicas e informações para o funcionamento. A partir deste mapa, foi possível detalhar as características para construção do modelo, com o levantamento de requisitos na seção 4.2.1, os casos de uso na seção 4.2.2 e alguns diagramas técnicos na seção 4.2.3.

Figura 17 - Mapa mental



Fonte: Elaborado pelo Autor

4.2.1 Levantamento de requisitos

Para descrever todas as funcionalidades esperadas no modelo, são considerados os seguintes requisitos funcionais:

i) Projetos
   (1) O modelo deve possibilitar a criação e alteração de projetos;
   (2) O modelo deve armazenar o histórico de alterações do projeto;

ii) Tarefas
   (1) O modelo deve possibilitar a criação e alteração de tarefas pelo usuário configurado como gerente do projeto;
   (2) O modelo deve permitir que os membros da equipe de projeto visualizem as todas as informações dos projetos em que está alocado;
   (3) O modelo deve permitir que os membros da equipe de projeto visualizem informações detalhadas de todas as tarefas dos projetos em que está alocado;
   (4) O modelo deve possibilitar o registro de andamento da tarefa pelo usuário responsável pela tarefa ou pelo gerente do projeto;
   (5) O registro de andamento da tarefa deve poder ser realizado de maneira simples, com o objetivo de aumentar a adoção do modelo pelos usuários;
   (6) O modelo deve armazenar o histórico de alterações e registros de todas as tarefas;

iii) Configurações
   (1) O modelo deve ser configurável em relação ao formato de desenvolvimento de cada projeto;
   (2) O modelo deve ser configurável em relação ao peso de cada uma das medidas de similaridade utilizadas para a predição;
   (3) O modelo deve ser configurável em relação a definição de sucesso de uma tarefa e projeto, permitindo pequenos desvios em relação ao planejamento, de acordo com parâmetros de prazo, custo e qualidade;

iv) Recomendações

(1) A cada alteração no projeto, o modelo deve verificar o contexto de todas as suas tarefas e analisar a necessidade de realizar novas predições;

(2) A cada alteração em uma tarefa, o modelo deve verificar o contexto da própria tarefa e de todas as outras tarefas que possuem alguma relação de dependência com a tarefa atual, analisando a necessidade de realizar novas predições;

(3) O modelo deverá considerar as melhores práticas de gerenciamento de projetos ao realizar as recomendações, de acordo com o formato de desenvolvimento de projeto utilizado;

(4) O modelo deverá informar os gestores do projeto sobre a recomendação em tempo real, através de um sistema de notificações;

(5) O modelo deverá permitir que o usuário solicite explicitamente recomendações para uma tarefa específica;

(6) O modelo deverá permitir que o usuário aceite ou rejeite as recomendações;

(7) O modelo deverá aprender com o uso das recomendações realizadas, dando maior peso para recomendações de maior aceitação;

(8) O modelo deverá manter um registro histórico de cada recomendação realizada para cada gestor, bem como o aceite ou não;

(9) Caso uma recomendação já tenha sido realizada e não aceita para uma determinada tarefa, o modelo deve ter a capacidade de identificar esta situação e não sugerir novamente a mesma ação;

(10) Caso uma recomendação seja aceita e aplicada em uma tarefa, e esta tarefa seja concluída com sucesso, esta recomendação deverá ser classificada com maior peso, para que em tarefas futuras que possuam histórico semelhante, seja possível utilizar a mesma recomendação;

(11) Caso ocorra algum erro ao executar o processo de recomendação, o usuário final deve ser informado, e deve ser ofertada a possibilidade de solicitar manualmente uma recomendação;

v) Relatórios

(1) O modelo deve exibir um gráfico de andamento dos projetos, contendo informações de *status*, prazo, custo e qualidade;

(2) O modelo deve permitir que este mesmo gráfico seja exibido no nível de projeto, considerando as informações referentes as tarefas;
(3) O modelo deve exibir um painel de controle sobre as recomendações, indicando as tarefas que possuem recomendação aceitas, pendentes de aceite e rejeitadas.

Enquanto os requisitos funcionais costumam apresentar funcionalidades do sistema, os requisitos não funcionais são aqueles que descrevem características, tecnologias e restrições no desenvolvimento de um projeto (LIU, 2016). Com este objetivo, foram elencados os seguintes requisitos não-funcionais:

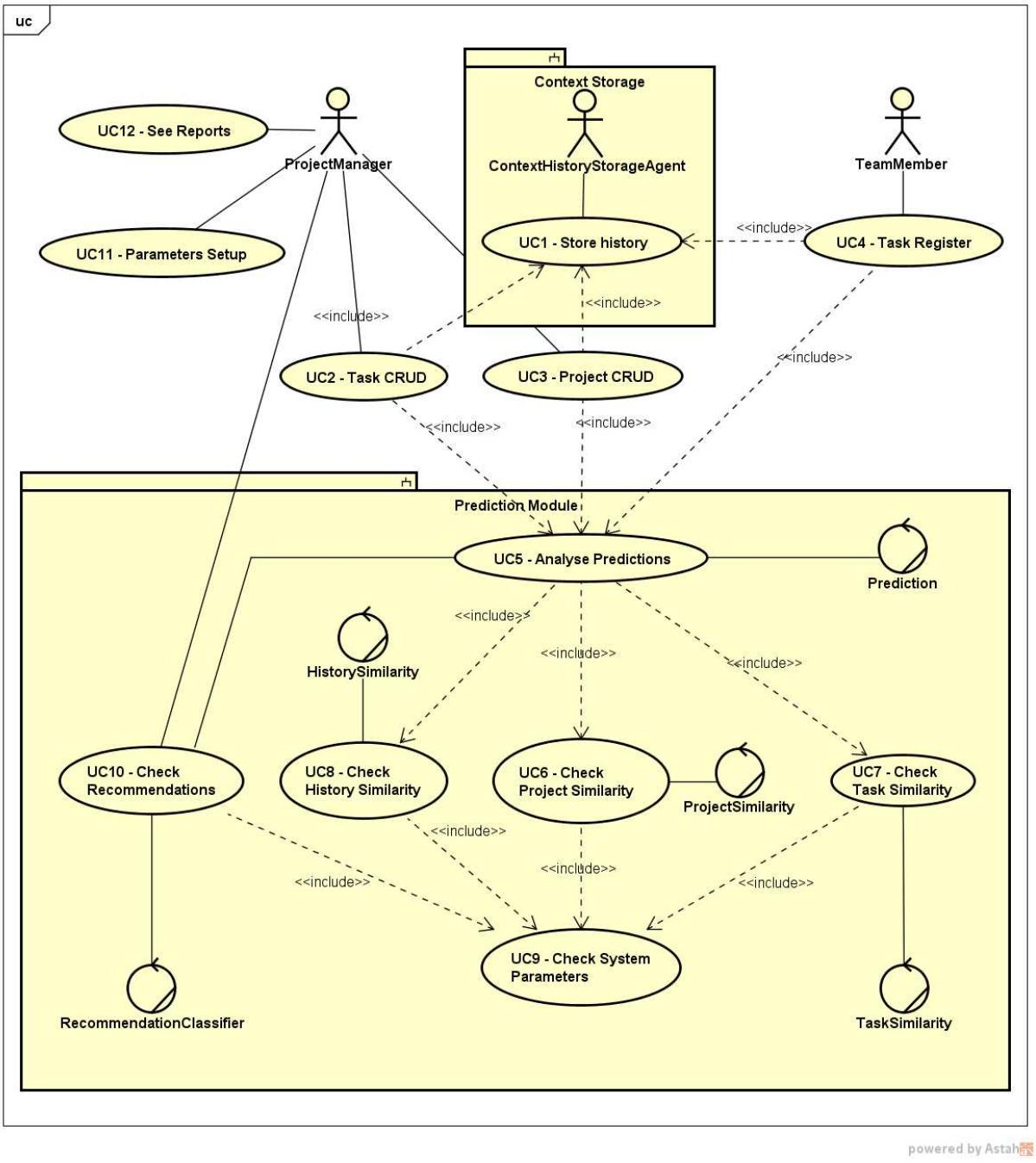
i) O modelo deverá ser acessível por computadores e dispositivos móveis;
ii) O modelo deverá ter funcionamento através da internet;
iii) As camadas de acesso a dados devem ser construídas considerando a possibilidade de evolução dos modelos de dados;
iv) A camada de acesso a dados utilizada pelos cadastros deve ser criada em uma base relacional;
v) O modelo deverá possui um agente específico para o armazenamento de históricos de contexto, trabalhando de forma independente do restante da ferramenta. As informações devem ser lidas da base relacional e gravadas em uma base não-relacional, visto que a massa de dados históricos tende a crescer rapidamente e de maneira não ordenada;
vi) O modelo não deve bloquear a interface do usuário enquanto realiza as análises;
vii) Os mecanismos de predição e recomendação devem ser construídos com limitadores de desempenho configuráveis, fazendo com que os algoritmos não permaneçam rodando indefinidamente;
viii) Considerando a natureza das funcionalidades, os algoritmos especialistas devem ser construídos em formato de múltiplos agentes, cada um destes com suas atribuições específicas;

ix) A comunicação entre a aplicação e os agentes deve se dar através de algum formato aberto, permitindo assim que cada agente seja desenvolvido utilizando linguagens e ambientes diferentes.

4.2.2 Casos de uso

Após o levantamento dos requisitos funcionais e não funcionais, o diagrama de casos de uso foi elaborado, conforme Figura 18. Considerando a possibilidade de criar atores que não sejam apenas usuários do sistema, mas também outras partes de um sistema, o diagrama apresenta como atores os usuários do modelo e também os agentes computacionais responsáveis pela execução de tarefas específicas em um ambiente distribuído.

Figura 18 - Diagrama de casos de uso



Fonte: Elaborado pelo Autor

Os atores que representam usuários do sistema são o *ProjectManager*, que representa o gestor do projeto, e o *TeamMember*, que engloba todos os demais papéis da equipe de projeto. De maneira resumida, o *ProjectManager* é responsável por

configurar os parâmetros de sistema (UC11), manter os cadastros de projetos (UC3) e tarefas (UC2), receber as recomendações (UC10) e visualizar os relatórios (UC12). O *TeamMember* tem o papel de executar as tarefas e manter seus registros de evolução atualizados (UC4), gerando os dados necessários para execução dos mecanismos de predição.

A notação de subsistemas presente no diagrama indica a existência de alguma parte específica do projeto, responsável por um conjunto de ações dentro de um mesmo contexto. Foram criados 2 subsistemas para o modelo, sendo eles o *ContextStorage* e o *Prediction Module*.

O módulo *ContextStorage* possui apenas o caso de uso UC1, que identifica a funcionalidade de armazenar os dados históricos de todas as entidades relevantes do sistema. Este caso de uso é acionado por todos os outros casos de uso que geram alterações nas entidades de projeto e tarefa. Dessa forma, ele é responsável por identificar as modificações realizadas em cada contexto, extrair os dados necessários para a execução posterior dos mecanismos de predição, e armazená-los em uma base de dados externa não-relacional, utilizada especificamente para o armazenamento de históricos de contexto.

O subsistema *Prediction Module* é responsável por todas as ações do fluxo de predição, sendo acionado pelos gatilhos de alteração de contexto. Os principais casos de uso deste subsistema são os seguintes:

- *UC5 – Analyse Predictions*: Este caso de uso representa as ações do agente chamado *Prediction*, que analisa as alterações de contexto e executa os mecanismos de predição. Ao ser executado, o agente deve acionar os demais agentes responsáveis pelos casos de uso UC6, UC7, UC8, que executam os algoritmos de similaridade de projeto, tarefa e histórico. Com o resultado destes casos de uso, o UC10 é ativado, para que realize as recomendações necessárias.
- *UC6 - Check Project Similarity*: Apresenta a funcionalidade de similaridade de projetos, analisando os dados do projeto atual em relação aos demais projetos existentes na base de dados, considerando os parâmetros previamente configurados pelo gerente para indicar o

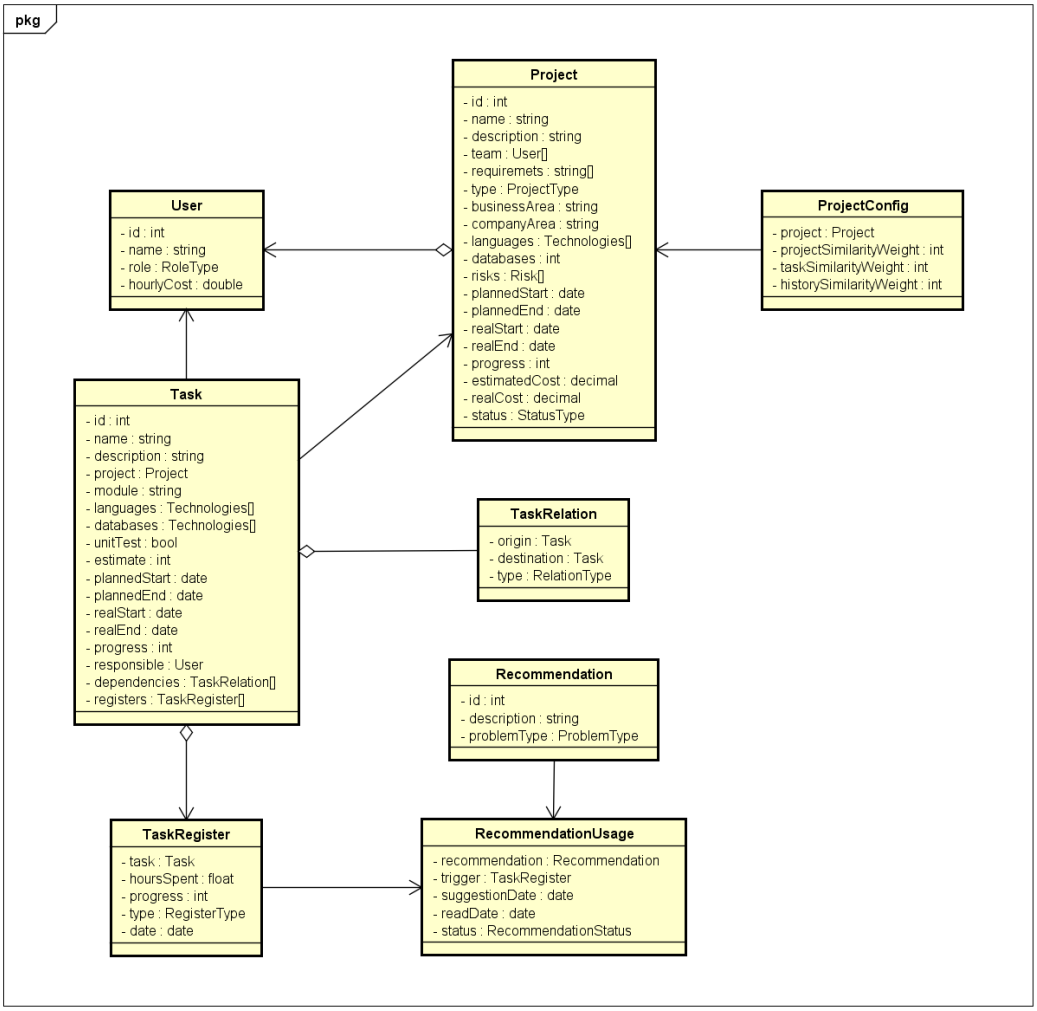
peso da similaridade pelo UC9. Ainda, caso a similaridade exista, deve retornar também o estado final do projeto, para que seja utilizado na predição.

- *UC7 - Check Task Similarity*: De modo semelhante ao UC6, este caso de uso traz a funcionalidade de similaridade de tarefas, analisando características da tarefa atual em relação às demais, também considerando os parâmetros de peso configurados no caso de uso UC9. Ao final da execução, deve listar as tarefas similares, de acordo com seus pesos, e os resultados de cada uma.
- *UC8 - Check History Similarity*: Analisa a similaridade do histórico de contextos da tarefa. Este algoritmo compara a evolução histórica de uma tarefa, considerando o consumo de horas, andamento do percentual de execução no eixo temporal, alterações de estado (como ser concluída e reaberta para correções), entre outros fatores. Considerando a complexidade do algoritmo, este é o algoritmo que tende a possuir a maior taxa de acertos em relação ao modelo de predição.
- *UC10 - Check Recommendations*: Este é o mecanismo que centraliza a realização de recomendações. É acionado pelo UC5, recebendo como parâmetros de entrada o retorno dos algoritmos de similaridade, contendo a listagem de projetos, tarefas e históricos mais similares ao da tarefa atual. Com estes dados, verifica os pesos de cada uma das variáveis nas configurações do sistema, e caso a chance de existirem problemas na tarefa seja alta, deverá verificar as recomendações configuradas. Para cada recomendação que se encaixe no problema da tarefa, deverá ser analisado o histórico de uso e aceitação dela em outras tarefas. As recomendações mais adequadas devem ser indicadas para o gestor do projeto em um ranking com percentuais de probabilidade de solução do problema, através de um sistema de notificações, com o objetivo de avisá-lo em tempo real para tomada de decisão imediata. A notificação deverá exibir uma descrição da situação e os dados que guiaram a previsão.

4.2.3 Diagramas

Para guiar o desenvolvimento do modelo, outros diagramas da UML foram desenvolvidos. Inicialmente, foram criadas classes conforme o diagrama conceitual de classes presente na Figura 19.

Figura 19 - Diagrama de Classes

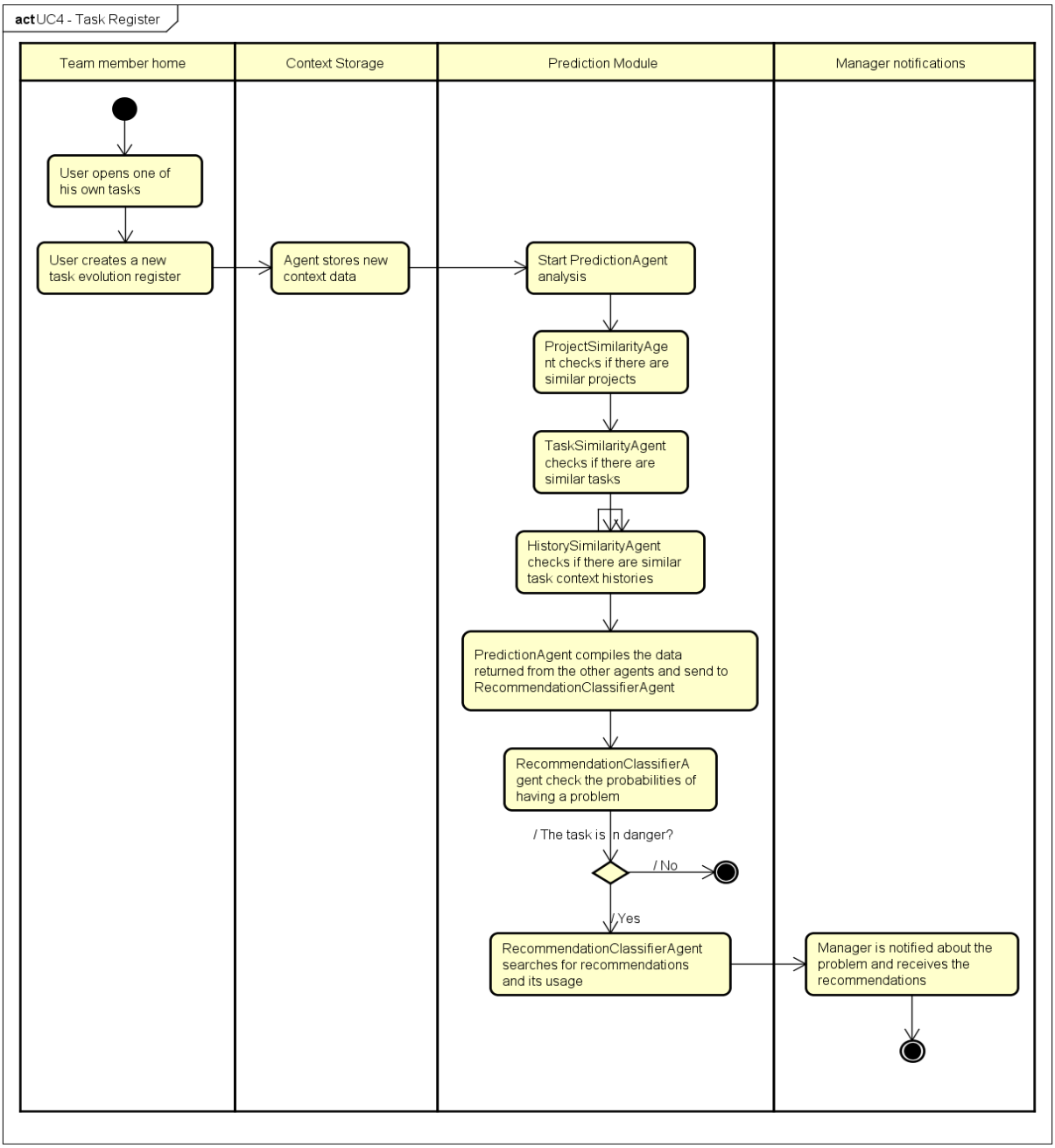


Fonte: Elaborado pelo Autor

Com o objetivo de demonstrar o fluxo de utilização do modelo, foi construído um diagrama de atividade. Este diagrama é apresentado na Figura 20, e apresenta o

fluxo de uso da funcionalidade de registro de horas de uma tarefa, identificada através do caso de uso UC4. Após o registro, o modelo armazena os dados do contexto atual da tarefa, insere na base de históricos, e inicia a execução dos mecanismos de predição. Caso seja encontrada alguma tarefa que possua alto nível de similaridade e que tenha passado por problemas de prazo, o modelo realiza uma recomendação de ação corretiva. Dada a natureza do modelo, este pode ser considerado o principal fluxo de utilização, visto que é a partir desta ação que são executados os algoritmos de análise para predição e recomendação.

Figura 20 - Diagrama de atividades

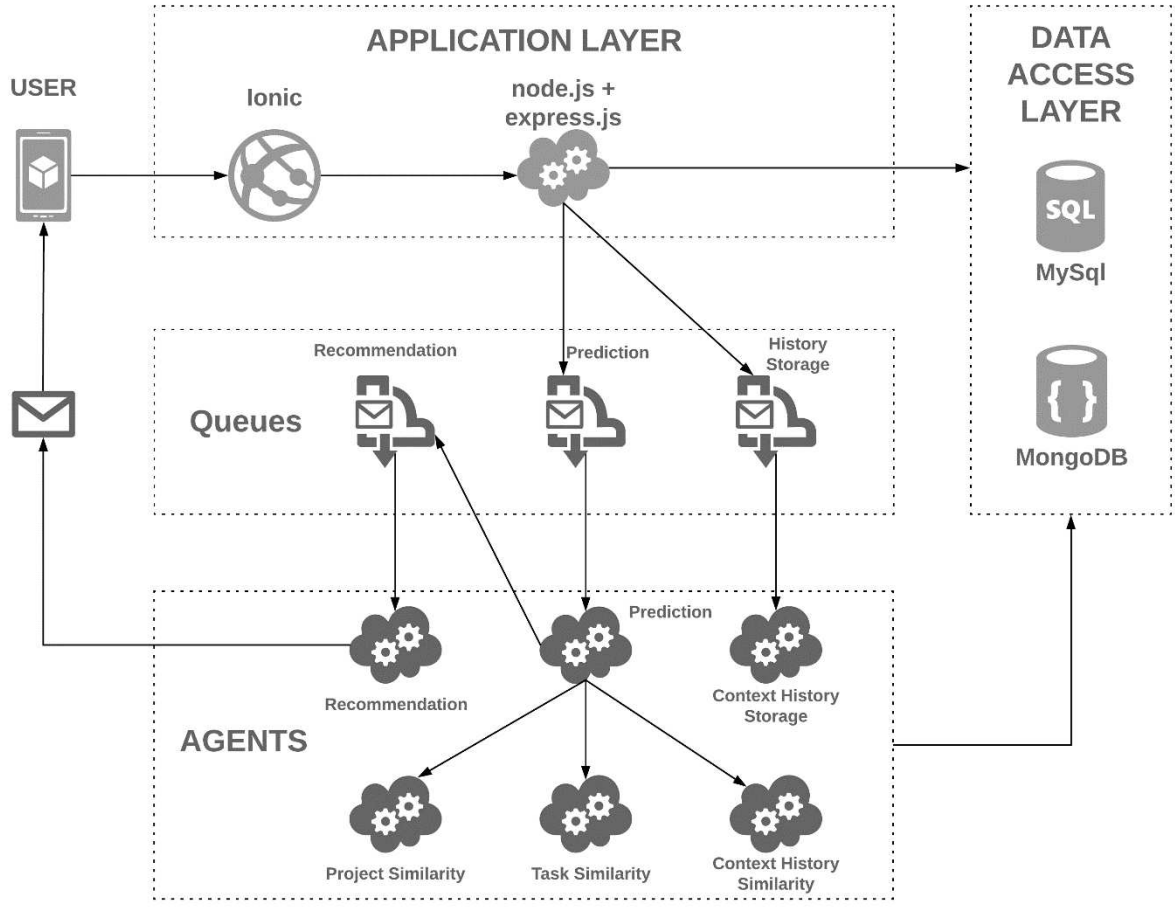


Fonte: Elaborado pelo Autor

4.3 Arquitetura

De acordo com os requisitos funcionais e não-funcionais, foram levantadas necessidades como a possibilidade de acesso por dispositivos móveis e a execução dos algoritmos de predição de modo desvinculado da interface da ferramenta, com o objetivo de não atrapalhar a experiência do usuário. Considerando estes fatores, a arquitetura do modelo Kairós foi construída em formato de uma aplicação web com o apoio de um subsistema de múltiplos agentes, onde cada agente é responsável pelas ações de uma parte do escopo da aplicação. A arquitetura da aplicação pode ser visualizada na Figura 21.

Figura 21 - Arquitetura do modelo Kairós



Fonte: Elaborado pelo Autor

Com o objetivo de demonstrar a validade do modelo, foi construída uma aplicação composta por uma interface de usuário e uma camada de serviços. A comunicação entre esta camada de aplicação e os agentes ocorre através do servidor de um serviço de enfileiramento de mensagens. Esta escolha foi feita para facilitar a integração entre as aplicações e os agentes, diminuindo o acoplamento entre os módulos, auxiliando na escalabilidade da aplicação, e desvinculando o processamento pesado da execução da aplicação principal. Além disso, ao trabalhar neste formato é possível desenvolver cada agente em uma linguagem de programação que se adeque melhor ao seu objetivo.

A camada de aplicação é responsável pela comunicação com os agentes *ContextHistoryStorage* (para o armazenamento de históricos de contexto) e *Prediction* (que centraliza o processo de predição e recomendação). Quando há a necessidade de acionamento dos agentes, a aplicação envia uma mensagem contendo os dados necessários para a fila de cada agente, e ao receber esta mensagem, o agente inicia seu processo de execução. O *Prediction*, por sua vez, é responsável por acionar os agentes utilizados no processo de predição (*ProjectSimilarity, TaskSimilarity* e *HistorySimilarity*) ao receber uma mensagem que indique a necessidade de verificação de predição. Dessa forma, compila os dados necessários para cada agente e faz o acionamento. Ao final dos processos de cada agente, consolida os resultados e envia uma mensagem para a fila do *RecommendationClassifier*, informando quais as melhores predições para o cenário atual. Ao ser acionado, o *RecommendationClassifier* verifica a necessidade de realizar uma recomendação com base no contexto recebido na mensagem, e caso seja necessário, busca e classifica as recomendações encontradas com base no histórico de uso e aceitação. Ao final do processo, registra a existência da recomendação e envia uma notificação para o gerente.

Os dados utilizados pela aplicação principal, como cadastros e recomendações, são armazenados em uma base de dados relacional. Visto que a estrutura que representa o contexto é complexa, podendo sofrer alterações ao longo do tempo, e o volume de dados possivelmente crescerá de maneira acelerada em

relação aos demais dados do sistema, os históricos de contextos são armazenados em uma base de dados não relacional. Estas características são bons indicadores do uso de um banco de dados deste tipo, pois permitem o armazenamento de documentos independentes do formato.

4.4 Agentes

De acordo com Weiss (1999), um agente é uma entidade computacional autônoma inteligente que interage em um ambiente específico, que possui características únicas e é responsável por resolver algum problema. Estes agentes costumam ter flexibilidade e certo nível de inteligência para resolver problemas, tomar decisões e aprender dentro do cenário em que se enquadra, podendo interagir com outros agentes e até mesmo seres humanos.

Considerando a complexidade dos problemas computacionais, estes agentes podem também possuir a capacidade de trabalharem de maneira integrada. Esta integração pode se dar de maneira competitiva, onde vários agentes tentam realizar a mesma tarefa e apenas alguns deles conseguem, ou colaborativa, onde compartilham seus conhecimentos e se comunicam para resolverem juntos o que um agente sozinho não conseguiria (WEISS, 1999).

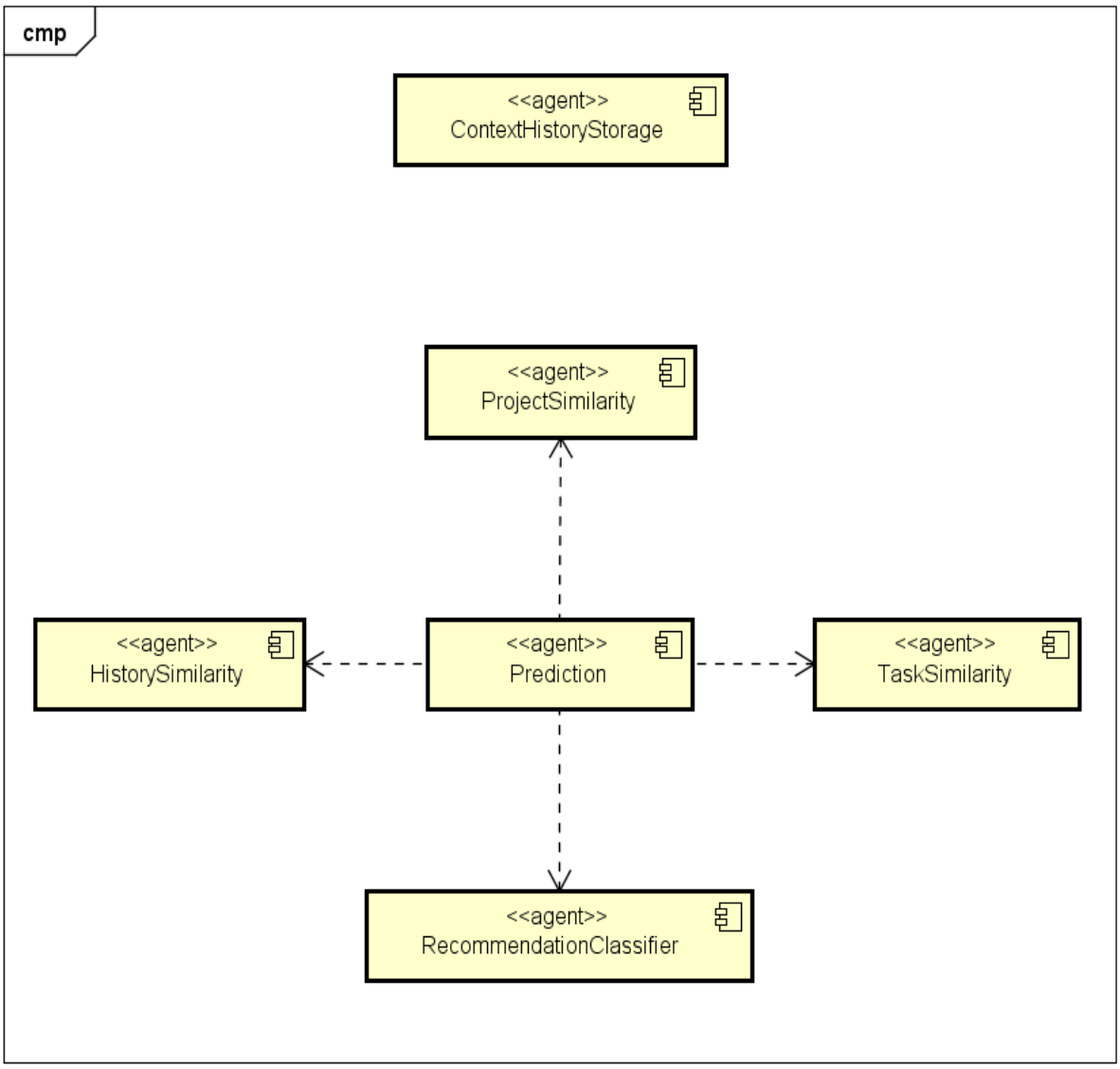
A inteligência e a autonomia são as principais características deste tipo de agente computacional. Neste cenário, Weiss (1999) explica que a inteligência indica uma orientação a metas pré-estabelecidas, através da execução de tarefas lógicas, enquanto a autonomia dos agentes é relacionada a capacidade de atuar sem a necessidade de intervenção humana, estando sempre em execução e em busca de seus objetivos.

No modelo Kairós, os agentes criados trabalham de maneira colaborativa em um sistema multiagentes, conforme Figura 22. Este tipo de arquitetura traz benefícios para modelos que possuem múltiplos algoritmos inteligentes, principalmente quando estes algoritmos podem atuar em ambientes independentes. Dessa forma, cada agente tem um escopo específico e é responsável por buscar um objetivo próprio e independente dos demais.

4.4.1 *ContextHistoryStorage*

Este agente é responsável pelo armazenamento do contexto de cada uma das entidades da aplicação a cada alteração nos dados de seu contexto, gerando uma base histórica que possui toda a evolução daquela entidade ao longo do tempo. Os dados são tratados e armazenados em uma base não-relacional com suporte a documentos em formato JSON, em estruturas semelhantes a tabelas chamadas “coleções”, conforme mostra a Figura 23. Cada histórico armazenado é vinculado a sua tarefa de origem através do campo identificador (*ProjectId* e *TaskId*), e marcado através da data e hora exata em que a alteração foi gerada. Assim, quando estes dados são acessados pelos demais agentes, é possível construir uma linha do tempo com a evolução dos contextos desta tarefa.

Figura 22 - Agentes do modelo Kairós



Fonte: Elaborado pelo Autor

Figura 23 - Histórico de contextos de uma tarefa em formato JSON

```
{
    Id : NumberLong(-8002985707495193368),
    Sk : NumberLong(32633961814037402l2),
    Ts : ISODate("2018-06-11T15:39:31.928Z"),
    TaskId : 5216,
    ProjectId : 873,
    Name : "Develop user authentication service",
    Description : "Create a restful service for users to
    login. This method receives an username and password,
    both as strings, and returns a string with the generated
    token. If the provided data is invalid or doesn't match
    any user in the database, the return will be an HTTP 401
    status (Unauthorized).",
    SystemModule : "security",
    ProgrammingLanguages: ["c#"],
    Databases: ["sqlserver"],
    UnitTest: true,
    Estimate : 16,
    PlannedStart: ISODate("2018-06-08T08:00:00.000Z"),
    PlannedEnd: ISODate("2018-06-11T18:00:00.000Z"),
    RealStart: ISODate("2018-06-08T08:54:32.754Z"),
    RealEnd: null,
    Progress: 50
    Responsible: "John Doe"
    Status: "Development"
    Dependencies: null
}
```

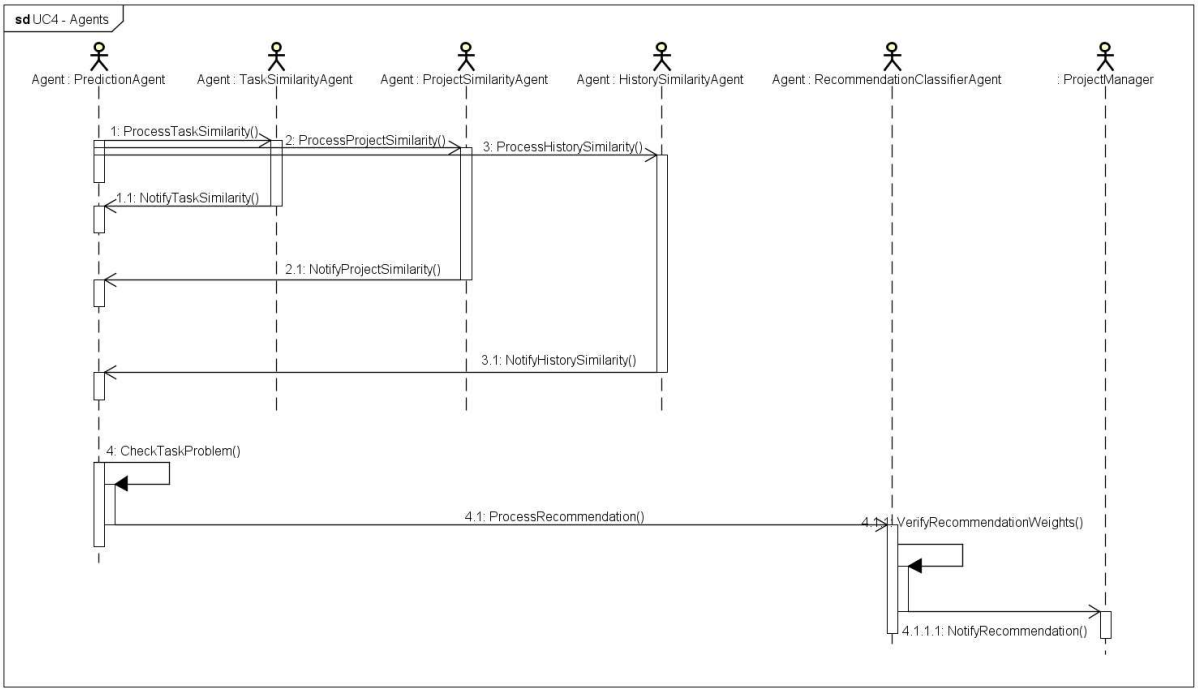
Fonte: Elaborado pelo Autor

4.4.2 *Prediction*

Este agente é responsável pela orquestração do processo completo de predição, conforme diagrama de atividades presente na Figura 24. É acionado ao receber uma mensagem enviada pela aplicação através do sistema de filas, contendo um contexto alterado, e a partir disso deve buscar os dados necessários e acionar os agentes responsáveis por identificar a similaridade com outras tarefas. Com o retorno das tarefas mais similares, é necessário verificar o estado final destas tarefas, para assim realizar a predição do estado final da tarefa atual, levando em consideração os pesos previamente configurados e as taxas de similaridade retornadas pelos agentes.

Caso a predição indique que a tarefa terminará fora do prazo estimado, o agente é responsável por acionar o *RecommendationClassifier*, para que verifique qual a melhor solução para o problema encontrado e envie a recomendação para o usuário.

Figura 24 - Diagrama representando a comunicação entre os agentes



Fonte: Elaborado pelo Autor

A predição é realizada através de analogia, considerando a similaridade da tarefa atual com demais tarefas existentes e já concluídas na base de dados. Para tal, são consideradas as características estáticas do projeto (agente *ProjectSimilarity*), da tarefa (agente *TaskSimilarity*), e o padrão evolutivo da tarefa, considerando as mudanças que ocorreram nesta ao longo de sua execução (agente *HistorySimilarity*).

Cada um destes agentes retornará uma lista com os itens mais similares, o grau de similaridade, a identificação do item e o estado final da tarefa vinculada, indicando o percentual de diferença no tempo de conclusão em relação ao prazo e a quantidade de horas estimados. A importância na similaridade retornada por cada um dos agentes

é indicada por um peso, que poderá ser configurado de acordo com as necessidades de cada cenário onde o modelo seja aplicado.

Para selecionar quais tarefas serão consideradas na predição, é aplicado um critério de classificação chamado *Multiple Criteria Weigthed Ranking* (ordenamento por múltiplos critérios com pesos), onde através destes pesos é possível obter uma classificação de importância. Assim, através de uma configuração previamente realizada que indica quantas tarefas devem ser consideradas para o peso – por exemplo, as 5 tarefas mais similares. Com estes dados, é realizado um cálculo que indica a probabilidade de atraso ($\hat{y}$), levando em consideração a similaridade de cada tarefa ($\tilde{x}_n$) e o percentual de atraso de cada uma destas (p_n), conforme equação abaixo:

$$\hat{y} = \frac{\sum(\widetilde{x_n} * p_n)}{\sum(\widetilde{x_n})}$$

$\hat{y}$ = predicted value
$\widetilde{x_n}$ = similarity level
p_n = delay percentage

O valor encontrado através do cálculo indica a probabilidade de atraso da tarefa atual. Para verificar a realidade de emitir um alerta, este valor é comparado com um parâmetro previamente configurado no modelo, que indica a agressividade da indicação. Através deste, é possível que o usuário indique qual o grau de probabilidade de atraso deverá desencadear as recomendações.

Se o valor encontrado estiver acima dos parâmetros, o agente *Prediction* deverá enviar uma mensagem para o agente *RecommendationClassifier* informando da necessidade de realizar uma recomendação para a tarefa atual. Nesta mensagem, são informados os dados identificadores da tarefa e a probabilidade de atraso encontrada.

4.4.3 *ProjectSimilarity* e *TaskSimilarity*

Estes agentes funcionaram de maneira semelhante, onde a única diferença são os dados que cada um trata. O *ProjectSimilarity* recebe os dados de um projeto, no formato mostrado na Figura 25, e é responsável por identificar outros projetos com

características similares ao projeto atual, enquanto o *TaskSimilarity* executa o mesmo procedimento no contexto da tarefa.

Figura 25 - Histórico de contexto de um projeto em formato JSON

```
{
    Id : NumberLong(-8002985707495193368),
    Sk : NumberLong(3263396181403740212),
    Ts : ISODate("2018-06-11T15:39:31.928Z"),
    ProjectId : 873,
    Name : "Flight Booking App",
    Description : "Creation of a new app for searching and booking flights, able to handle
    multiple customers. The app shall be developed using Ionic Framework, and its layout
    shall be customized by CSS, where each customer can have its own customizations.
    Communication with flight suppliers shall use iTravel's FlightSuite engine.",
    BusinessArea : "ecommerce",
    CompanyArea: "mobile",
    FunctionalRequirements: ["...", "..."]
    NonFunctionalRequirements: ["...", "..."]
    ProgrammingLanguages: ["c#", "javascript", "node", "ionic"],
    Databases: ["sqlserver", "mongodb"],
    PlannedRisks: [
        {
            RiskId: 51,
            Title: "Lack of experience with Ionic Framework",
            Category: "instruments",
            IsProblem: false,
            Probability: 7,
            Impact: 3,
            Review: true,
            ResponseType: "mitigate",
            Response: "Schedule a training with developers",
            Responsible:
            {
                ResourceId: 891,
                Name: "Joseph Doe",
                Role: "manager"
            }
        }
    ],
    Problems: [ ],
    Resources: [
        {
            ResourceId: 891,
            Name: "Joseph Doe",
                Role: "manager"
        },
        {
            ResourceId: 1852,
            Name: "John Doe",
            Role: "developer"
        }
    ],
    PlannedStart: ISODate("2018-05-02T08:00:00.000Z"),
    PlannedEnd: ISODate("2018-06-29T18:00:00.000Z"),
    RealStart: ISODate("2018-06-002T11:26:13.847Z"),
    RealEnd: null,
    Progress: 67,
    EstimatedCost: 67840.00,
    RealCost: 52416.00,
    Status: "development"
    Tasks: [...]
}
```

Fonte: Elaborado pelo Autor

O algoritmo usado para a realização destas comparações é o Coeficiente de Similaridade de Gower, inicialmente proposto pelo matemático J. C. Gower em 1971

e estendido por J. Podani em 1999 (GOWER, 1971; PODANI, 1999). Este tipo de comparação de similaridade se difere dos demais por trabalhar com estruturas de tipos e valores variados, sendo capaz de comparar uma estrutura complexa com valores inteiros, decimais e booleanos, sem a necessidade de padronizar a estrutura de dados em um único tipo de informação. Tal característica se faz necessária ao comparar estruturas como os dados de um projeto, onde temos campos como estimativa em horas, que pode variar muito de um projeto para outro, e o estado do projeto, que possui apenas algumas opções fixas de valor. Ainda, é possível atribuir um peso para cada uma das variáveis, o que também auxilia o cenário da comparação necessária. O algoritmo, implementado em linguagem R, consta na Figura 26.

Figura 26 – Trecho de código demonstrando o algoritmo de Gower

```
data = fetch(rs, n = -1)

library(cluster)
d <- daisy(data, metric = "gower", type =
     list(logratio = c(1, 5, 6, 7, 11, 12, 13),
        ordratio = c(2, 3, 4, 8, 9, 10)),
        weights = c(weight.name, weight.desc, weight.business, weight.area,
        weight.pl, weight.db, weight.funcReq, weight.nonFuncReq, weight.risk,
        weight.resources, weight.size, weight.duration, weight.estCost))

summary(d)
gower_mat <- as.matrix(d)
print(gower_mat)
clust <- pam(d, k=10, diss=T)
plot(clust)

```

Fonte: Elaborado pelo Autor

Além deste algoritmo, é também empregado o uso de técnicas de processamento de linguagem natural para que seja possível comparar campos textuais. Com isso, será possível analisar dados como a descrição do projeto e da tarefa, identificando assim as entidades com objetivos semelhantes e que são semanticamente similares. Cada campo textual será comparado através do algoritmo *Word Movers Distance*, que considera a semântica de cada frase ao realizar o cálculo de similaridade (KUSNER et. al., 2015), conforme Figura 27.

Figura 27 – Trecho de código demonstrando o cálculo da similaridade entre os nomes e descrições das tarefas usando WMD

```
from pyMongo import MongoClient
client = MongoClient('localhost', 27017)
db = client['database']
tasks  = db['tasks']


nameSimilarities = []
descSimilarities = []
for task in tasks.find({"$and": [{"status": 100}, {"id": {"$ne": baseTask.id}}]}):
  nameSim = word_vectors.wmdistance(baseTask.name.lower().split, task.name.lower().split())
  nameSimilarities.append([task.id, nameSim])
  descSim = word_vectors.wmdistance(baseTask.description.lower().split, task.description.lower().split())
  descSimilarities.append([task.id, descSim])

nameSimilarities.sort(key=lambda x:x[1])
descSimilarities.sort(key=lambda x:x[1])
```

Fonte: Elaborado pelo Autor

Para a execução do algoritmo de Gower, os dados serão previamente tratados através de uma função de normalização. Dessa forma, a estrutura representada na Figura 25 é transformada em estruturas conforme Figura 28 para projetos e Figura 29 para tarefas, fazendo com que seja possível efetuar comparações entre os dados, seguindo os seguintes passos:

- Exclusão de campos: os campos que não apresentam informação relevante são removidos do objeto, como os identificadores “Id”, “Sk”, “Ts” e “ProjectId”;
- Conversão de dados: campos booleanos são convertidos para números (0 e 1) e os campos textuais são removidos, deixando apenas seus indicadores de similaridade;
- Agregação: campos que apresentam múltiplos valores (como as listas de linguagens de programação, bancos de dados, requisitos funcionais, requisitos não-funcionais, riscos e recursos) são agregados através de média aritmética de seus valores numéricos.

Figura 28 – Dados de um projeto tratados para comparação


```
{
    Name : 0.785,
    Description : 0.560,
    BusinessArea : 1,
    CompanyArea: 0.682
    ProgrammingLanguages: 0.250,
    Databases: 1,
    FunctionalRequirements: 0.102,
    NonFunctionalRequirements: 0.067,
    PlannedRisks: 0,
    Resources: 0.650,
    Size: 680,
    Duration: 42,
    EstimatedCost: 67840.00
}
```


Fonte: Elaborado pelo Autor

Figura 29 – Dados de uma tarefa tratados para comparação

```
{
    Name : 0.569,
    Description : 0.142,
    SystemModule : 0.500,
    ProgrammingLanguages: 1,
    Databases: 0.750,
    UnitTest: 1,
    Estimate : 16,
    Responsible: 1
}
```

Fonte: Elaborado pelo Autor

4.4.4 *HistorySimilarity*

O *HistorySimilarity* é o agente responsável pela predição através de históricos de contextos. Para tal, são analisados os históricos de contextos de outras tarefas visando encontrar aquelas que tiveram uma evolução histórica similar. Assim, é possível verificar o estado final destas tarefas, e através disso prever o estado final da tarefa em andamento.

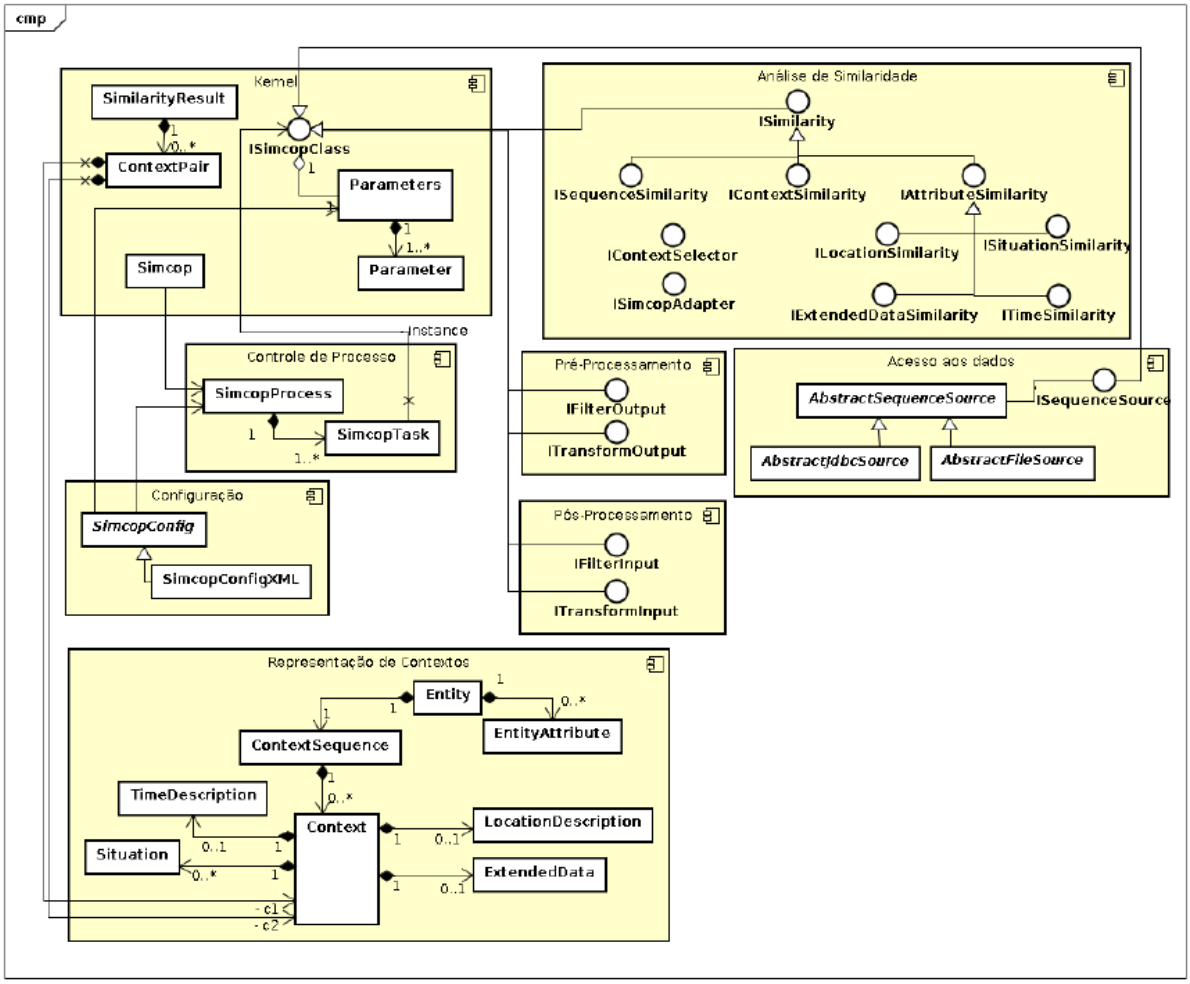
Neste cenário, o agente verifica todas as alterações realizadas em uma determinada tarefa, considerando horas executadas e percentual de conclusão, através dos seus registros de andamento. Com estas informações, o agente constrói uma estrutura de dados correspondente a uma linha do tempo de evolução da tarefa, que apresenta todos os estados pelos quais passou desde a sua criação e alocação no cronograma. Considerando que o objetivo da predição é verificar se esta tarefa em andamento sofrerá atrasos ou excesso nos custos, estas informações são de grande valia.

Desse modo, o mecanismo faz a comparação da evolução temporal da tarefa que disparou a predição com a evolução temporal de outras tarefas na base histórica que já estejam concluídas. Ao encontrar outra tarefa que teve uma evolução

semelhante até o momento, pode-se realizar uma predição por analogia, utilizando o estado final da tarefa encontrada como um possível estado final da tarefa original.

Para esta implementação, foi utilizado o *framework* SIMCOP, que dá suporte nativo a análise de similaridade entre sequências de contextos históricos de uma determinada entidade (WIEDEMANN, 2014). O SIMCOP possui inúmeras técnicas de análise de similaridade de contextos, e permite que o desenvolvedor que o utilizará decida qual delas utilizará, através de arquivos de configuração. A implementação foi realizada de acordo com a estrutura prevista pelo *framework*, de acordo com o diagrama de componentes na Figura 30.

Figura 30 - Diagrama de componentes do SIMCOP



Fonte: WIEDEMANN, 2014.

Internamente, o SIMCOP possui uma série de algoritmos para a comparação de sequências de contextos, como a distância LP, distância de Levenshtein, probabilidade contextual, *Dynamic Time Warp* (DTW) e *Merge* (WIEDEMANN, 2014). Ao executar a análise, é possível decidir qual destas será utilizada. Foram realizados testes, e o DTW foi o algoritmo que apresentou melhores resultados nos cenários analisados (BERNDT, CLIFFORD, 1994).

Ainda, o SIMCOP permite a parametrização de uma similaridade mínima para o retorno. Dessa forma, apenas os contextos mais similares são devolvidos, juntamente do grau de similaridade encontrado. Assim como os demais parâmetros, este também pode ser configurado pelo usuário no módulo de configurações do modelo.

O SIMCOP foi integrado no modelo através de uma camada de serviço, criada para abstrair as configurações necessárias e focando na execução dos algoritmos de comparação. Dessa forma, o serviço recebe uma tarefa como parâmetro, verifica o histórico desta em comparação com o histórico das demais tarefas, e retorna as tarefas que possuem o histórico mais similar, conforme Figura 31.

Figura 31 – Execução do serviço DTW dentro do SIMCOP

```
WSDynamicTimeWarp instance = new WSDynamicTimeWarp();
instance.setContextSimilarity(new SimpleAttributes());
instance.setParameter(FastDTWAdapter.PAR_SEARCH_RADIUS, "1");
//---

try {
    Date d1 = new Date();
    SimilarityResult sr = instance.getSimilarity(s1, s2);
    Date d2 = new Date();

    System.out.println("   TIME: " + (d2.getTime() - d1.getTime()) + "ms");
    assertNotNull("result null", sr);
    for (int i = 0; i < sr.size(); i++) {
        System.out.println("---[" + i + "]-------------------------");
        ContextPair cp = sr.getContextPairs().get(i);
        if (cp != null) {
            System.out.println(cp.toString());
        } else {
            System.out.println("null");
        }
        System.out.println(" ");
    }
    System.out.println("=========================");
    System.out.println("DTW Result: " + sr.getCalculatedValue());

} catch (Exception ex) {
    Logger.getLogger(WSDynamicTimeWarp.class.getName()).log(Level.SEVERE, null, ex);
}
```

Fonte: Elaborado pelo Autor

4.4.5 *RecommendationClassifier*

Após a realização das predições pelo agente *Prediction*, caso o resultado indique algum problema na tarefa, o agente *RecommendationClassifier* será acionado.

Este receberá os dados da tarefa em questão e os resultados das predições, para que a partir disso possa recomendar uma ou mais ações corretivas.

Estas recomendações tiveram como base inicial as melhores práticas de projeto, considerando ações como adicionar um recurso extra para auxiliar no desenvolvimento da tarefa, alterar o recurso responsável por alguma tarefa dependente cujo início será afetado pelo atraso na tarefa atual, entre outros. Ainda, no caso dos projetos com desenvolvimento através de métodos ágeis, outras estratégias também poderão ser adotadas, como a troca de um item do *sprint backlog* que não poderá ser entregue por algum outro item do *project backlog* que possa ser concluído no *sprint* atual.

Ainda, é realizado um processo de filtragem de recomendações inválidas, ou que podem ter um resultado negativo no processo. Por exemplo, a adição de recursos em uma tarefa apenas é recomendada caso existam recursos disponíveis e com as habilidades necessárias para a execução da tarefa, considerando conhecimentos em linguagens de programação e bancos de dados configurados na tarefa e na base de recursos.

Ao informar o gestor sobre as ações corretivas recomendadas, são exibidas opções para classificar a utilidade de cada recomendação, utilizando uma escala de 1 a 5, onde 1 significa que a recomendação não é nada adequada, e 5 indica que é muito útil. Estas classificações são armazenadas e consideradas ao realizar novas recomendações, onde a cada classificação ruim, a recomendação perde importância e passa a ser menos indicada.

Caso o usuário discorde do que foi sugerido, selecionando as opções 1 ou 2, é possível inserir manualmente uma descrição textual do que será realizado para corrigir o problema. Esta sugestão dada pelo usuário é cadastrada na base de recomendações, e em situações futuras será também considerada ao realizar novas recomendações.

4.5 Considerações

O modelo Kairós foi concebido com o objetivo de auxiliar as equipes a controlar a execução do projeto, atuando no registro de andamento das tarefas, e realizando predições para o gerente, visando evitar que problemas ocorram. Na presente abordagem, dentre todos os problemas que podem ocorrer, apenas o atraso nas tarefas é considerado. Outros tipos de problemas, como os relativos à gestão de pessoas, necessidade de contratação, ou problemas de qualidade na entrega, não fazem parte do escopo desse trabalho.

Nos projetos de desenvolvimento de software, cada empresa possui características em seus processos internos que podem ser diferentes de outras empresas. Dessa forma, justifica-se o uso de algoritmos de predição por similaridade, visto que este utiliza dados passados da própria empresa para a realização de predições.

Ainda, o uso deste tipo de algoritmo permite a configuração de pesos em cada uma das variáveis consideradas, o que dá ainda mais flexibilidade para o modelo. De acordo com a necessidade, qualquer variável pode ter seu peso reduzido ou aumentado para que a sua similaridade entre as tarefas seja mais decisiva ao realizar as predições. Assim, é possível, por exemplo, implantar o modelo em um cenário de empresa que não possui gestão de riscos, bastando configurar o peso da variável relativa aos riscos com o valor 0.

Da mesma forma, para a análise dos históricos de contextos, foi utilizado o framework SIMCOP. Este framework atua na comparação entre históricos, classificando-os por similaridade e indicando quais tem o maior grau de proximidade. Com isso, é possível identificar tarefas que tenham um tipo de evolução temporal semelhante, e assim utilizar o estado final das tarefas mais similares na predição.

5 ASPECTOS DE IMPLEMENTAÇÃO E AVALIAÇÃO

Para o processo de validação do modelo Kairós, foi construído um protótipo para dar suporte às funcionalidades do modelo, respeitando a arquitetura definida. Este protótipo foi desenvolvido como parte de um projeto macro, desenvolvido por alunos de mestrado e doutorado do Programa de Pós-Graduação em Computação Aplicada da Unisinos (PPGCA), no qual outros estudos para construção de ferramentas de auxílio ao gestor foram desenvolvidos.

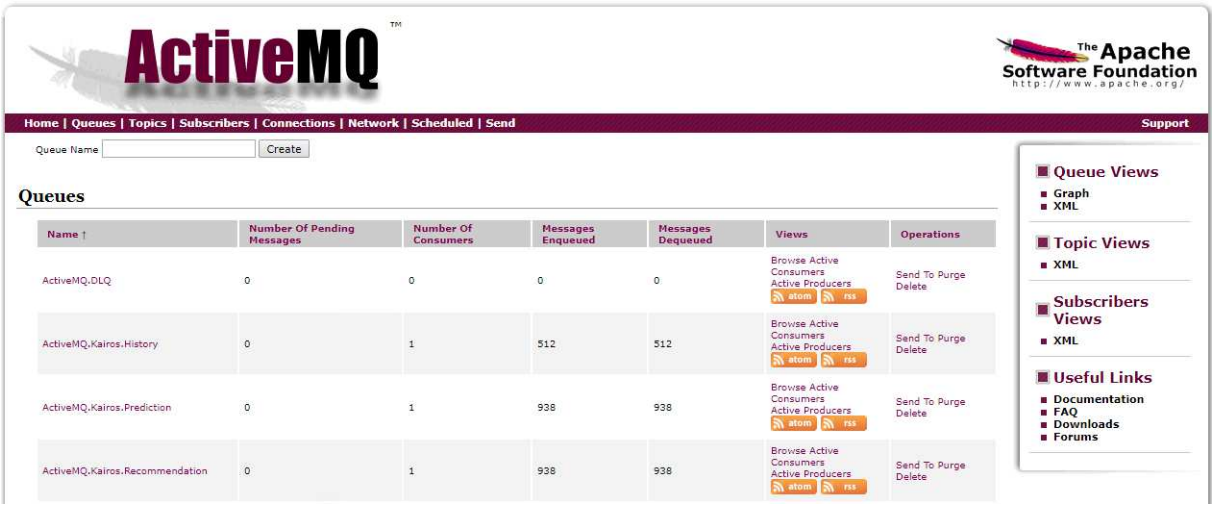
Estes estudos conjuntos têm o objetivo de construir um modelo completo para a gestão de projetos utilizando conceitos de computação ubíqua. Dessa forma, inúmeros outros trabalhos têm sido desenvolvidos dentro deste tema, abordando outras áreas da gestão tais como riscos, recursos e requisitos. A arquitetura deste modelo macro foi elaborada por todos os membros do grupo, de modo a adaptar-se a necessidade de todas as áreas de conhecimento envolvidas.

A implementação do protótipo será descrita na seção 5.1, e a metodologia utilizada para avaliar o modelo será apresentada na seção 5.2. A seção 5.3 apresenta um cenário de exemplo de uso do modelo através do protótipo. Finalmente, a seção 5.4 apresenta os resultados encontrados através da avaliação realizada.

5.1 Implementação do protótipo

Para a construção da aplicação que será acessada pelos usuários finais, a principal linguagem utilizada foi o JavaScript. O Node.js foi a escolha para o desenvolvimento dos serviços e APIs responsáveis pela execução da aplicação, comunicação com as camadas de dados e com os agentes. O *framework* Express.js foi também utilizado, trazendo inúmeras ferramentas úteis para a criação de aplicações e serviços web. Esta camada é responsável por acessar o banco de dados em um servidor MySQL e o serviço de filas através do Apache ActiveMQ, conforme Figura 31.

Figura 32 – Filas utilizadas para comunicação



Fonte: Elaborado pelo Autor

Quanto ao aplicativo que é acessado pelo cliente final, a implementação foi realizada com o uso do *framework* Ionic, que facilita a criação de aplicativos nativos para as principais plataformas de dispositivos móveis com um único código-fonte. Internamente, o Ionic encapsula outros *frameworks* como o AngularJS, criado pela Google para auxiliar no desenvolvimento de aplicativos em JavaScript, e o Apache Cordova, que tem o objetivo de utilizar códigos escritos em HTML5, JavaScript e CSS3 para a geração de aplicações nativas em Android, iOS, Windows e Mac OS X.

Os agentes foram desenvolvidos usando linguagens de programação e *frameworks* que facilitem o desenvolvimento, de acordo com a funcionalidade necessária. Por exemplo, os agentes responsáveis pela comparação de similaridade entre projetos e tarefas foram desenvolvidos em Python, uma linguagem de programação que possui muitas aplicações em inteligência artificial, similaridade e processamento de linguagem natural que são alguns dos algoritmos utilizados nestes agentes. Já o agente responsável pela comparação de históricos de contextos foi desenvolvido em JAVA, que é a linguagem mais utilizada nas implementações existentes para este tipo de algoritmo.

Utilizando estas ferramentas e tecnologias, uma versão do protótipo foi desenvolvida, contemplando funcionalidades básicas. As telas de cadastro de projetos e tarefas foram criadas contendo os campos definidos no modelo de dados. Após a criação e configuração das tarefas, elas ficam disponíveis para que seus usuários atribuídos possam registrar o andamento.

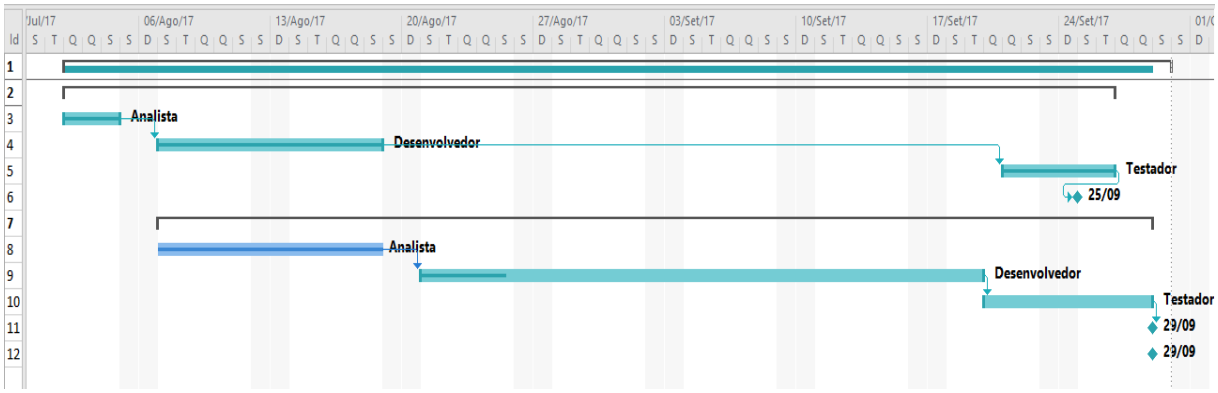
Ao registrar o andamento, o sistema percebe a alteração de um contexto e aciona o mecanismo de predição. Caso exista algum problema na tarefa e alguma recomendação seja realizada, o gestor deverá receber uma notificação com os dados do problema, e ao abrir esta notificação visualizará as recomendações. Caso não queira decidir no momento, o usuário também poderá dispensar a recomendação a acessá-la através da tela de entrada do sistema, onde as últimas recomendações ficam disponíveis para acesso a qualquer momento.

5.2 Metodologia de avaliação

O processo de avaliação tem como objetivo corroborar as contribuições do modelo e verificar a viabilidade de construção de um sistema computacional funcional seguindo suas especificações. Para tal, a avaliação do modelo foi realizada através de validação cruzada, utilizando dados de projetos reais de aproximadamente 24 meses de trabalho de uma empresa do ramo de desenvolvimento de sistemas, através da técnica de *holdout*, considerando 75% dos dados como parâmetro e 25% dos dados como teste (KOHAVI, 1995). Visto que o modelo necessita de dados de históricos de contexto, estes projetos foram importados para a base de dados do protótipo simulando o uso real das equipes de projeto durante o trabalho normal.

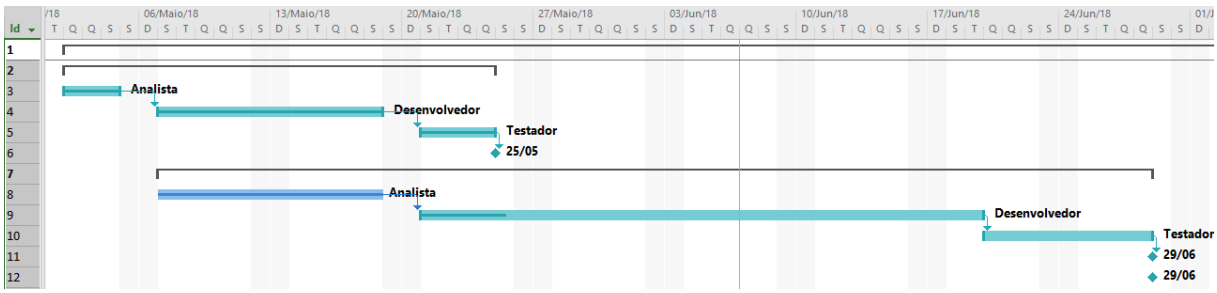
Todos os campos de data foram deslocados alguns meses para a frente no tempo, com o objetivo de fazer com que os projetos estejam de acordo com a data atual, e permitir o uso da data atual nos registros de andamento realizados pelos dispositivos na fase de teste. Ao analisar a Figura 33, percebe-se que o projeto foi executado nos meses de julho a setembro de 2017. Já a Figura 34, que apresenta o projeto após o deslocamento temporal, mostra as datas de execução entre abril e junho de 2018, sendo este o momento em que os testes do protótipo foram realizados.

Figura 33 – Projeto em suas datas originais



Fonte: Elaborado pelo Autor

Figura 34 – Projeto atualizado após deslocamento temporal



Fonte: Elaborado pelo Autor

As inserções de todos os dados foram feitas em ordem cronológica, simulando a execução real do projeto, através de um mecanismo de importação construído especificamente para esta finalidade, que armazenou os históricos de contextos normalmente, mas desconsiderou os mecanismos de predição. Todos os dados de

projetos e tarefas foram inseridos simultaneamente no início do processo, e após os registros de alteração e andamento foram inseridos um a um, simulando a passagem do tempo e a evolução real do projeto. Porém, para que fosse possível validar a eficiência do modelo, os dados foram criados desconsiderando os últimos 6 meses de andamento de todos os projetos. Assim, foi possível avaliar o comportamento simulado, realizado dentro do ambiente controlado do protótipo, em relação a evolução real do projeto na empresa de origem.

Durante a fase de testes, os 6 meses restantes foram inseridos manualmente, simulando também a atividade das equipes de desenvolvimento. No entanto, estas novas informações foram cadastradas através do próprio protótipo, acionando os mecanismos de predição e verificando as recomendações realizadas. Estas recomendações foram anotadas para posterior avaliação.

5.3 Cenário de exemplo

Para demonstrar a funcionalidade do protótipo, foi selecionada uma tarefa da massa de testes para estudo de caso. A tarefa "*Create payment page*" faz parte do projeto "*Cruise Suite*", tendo como objetivo a construção de uma tela de reserva e pagamento em um fluxo de *ecommerce* de cruzeiros marítimos. Esta tarefa possuía uma estimativa de 60 horas, com previsão de conclusão em 8 dias.

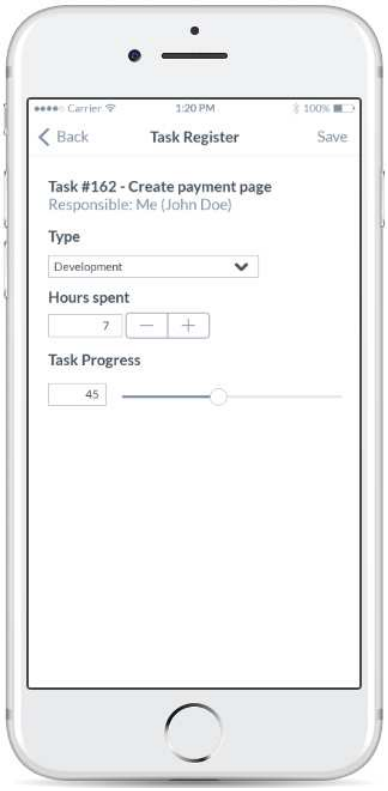
Através do protótipo, foi simulado o uso por um desenvolvedor, considerando os dados reais de execução da tarefa obtidos na base de dados original. Ao longo dos 4 primeiros dias, a tarefa recebeu os seguintes registros de andamento:

- Dia 1: 8 horas executadas no dia, progresso total de 10%;
- Dia 2: 8 horas executadas no dia, progresso total de 25%;
- Dia 3: 8 horas executadas no dia, progresso total de 30%;
- Dia 4: 8 horas executadas no dia, progresso total de 40%;

No quinto dia, o registro de andamento se deu conforme a Figura 35, no qual foram registradas mais 7 horas de trabalho e um percentual de progresso de 45%. Ao finalizar o preenchimento dos campos e clicar em "*Save*", os agentes de

armazenamento de histórico de contexto (*ContextHistoryStorage*) e verificação de predições (*Prediction*) foram acionados. Internamente, o agente *Prediction* acionou os agentes responsáveis pelas análises de similaridade entre os dados estáticos das tarefas e projetos (*ProjectSimilarity* e *TaskSimilarity*) e de históricos de contexto (*HistorySimilarity*).

Figura 35 – Tela de registro de andamento no 5º dia de execução



Fonte: Elaborado pelo Autor

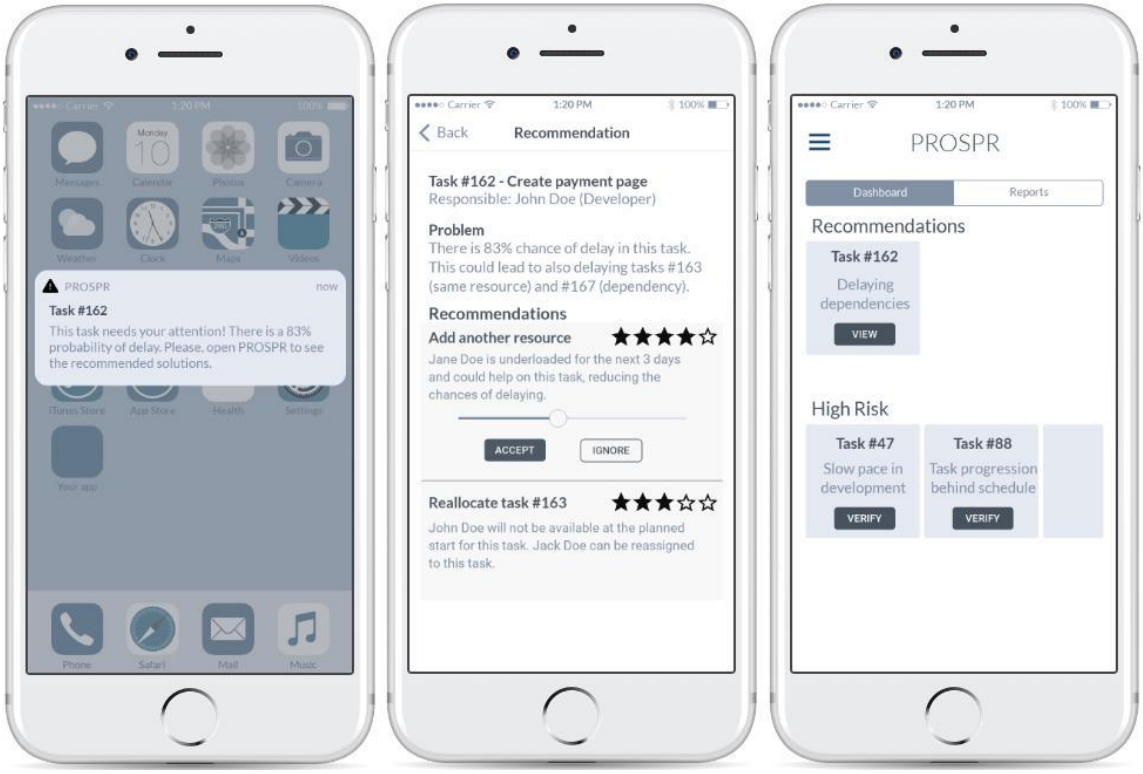
Analisando os dados da tarefa, é possível perceber que, de acordo com a estimativa de finalização em um prazo de 8 dias, no 5º dia a tarefa já deveria estar proporcionalmente mais evoluída. Internamente, os algoritmos de predição identificaram que entre as 5 tarefas mais semelhantes, 4 delas terminaram com atraso.

Através dos cálculos previamente apresentados, o modelo indicou uma probabilidade de atraso de 83%.

Ao identificar esta probabilidade, o agente responsável pela predição enviou uma mensagem contendo os dados da tarefa e o resultado obtido para o agente responsável pelas recomendações (*RecommendationClassifier*). Este agente verificou que existe um possível problema com a tarefa, e assim iniciou o processo de verificação de possíveis soluções.

Através da base de recomendações existentes, e da classificação destas obtidas através das avalições dos usuários, as principais recomendações são elencadas para exibição ao gestor. Ainda, o agente verifica possíveis impactos em outras tarefas e leva esta informação para o gestor, considerando a data de início da próxima tarefa do recurso alocado na tarefa atual e as tarefas que dependem desta para iniciar. Com o resultado destes algoritmos, o gestor recebe uma notificação, e ao clicar nesta visualizar a tela com os detalhes do problema encontrado, as recomendações para esta tarefa, e um dashboard com todas as recomendações realizadas, conforme Figura 36.

Figura 36 – Notificação, tela de recomendação e lista de recomendações



Fonte: Elaborado pelo Autor

Ao visualizar a tela de recomendação, o usuário tem acesso aos dados básicos da tarefa com problema, uma breve descrição do problema, detalhes de possíveis impactos negativos em outros pontos do projeto, e as recomendações em si. Cada recomendação traz consigo uma classificação prévia, indicando o quanto esta recomendação foi útil em situações anteriores. Com estas informações, o gestor pode aceitar ou ignorar cada uma das recomendações, bem como classificar o quanto ela está correta em relação ao problema.

Analisando a base de dados original, onde a tarefa em questão já teve seu andamento finalizado, percebe-se que a mesma foi concluída com 2 dias de atraso e 20% de excesso no custo em horas. Assim, é possível concluir que a predição foi realizada corretamente, indicando durante o andamento da tarefa que a mesma seria concluída com atraso.

5.4 Resultados

Para avaliar a assertividade de cada predição, foi utilizada uma matriz de confusão contendo os 4 resultados possíveis e os parâmetros de avaliação, conforme Figura 37. Foi considerado como parâmetro de acerto a indicação de conclusão da tarefa com atrasos em relação ao prazo definido no planejamento do cronograma.

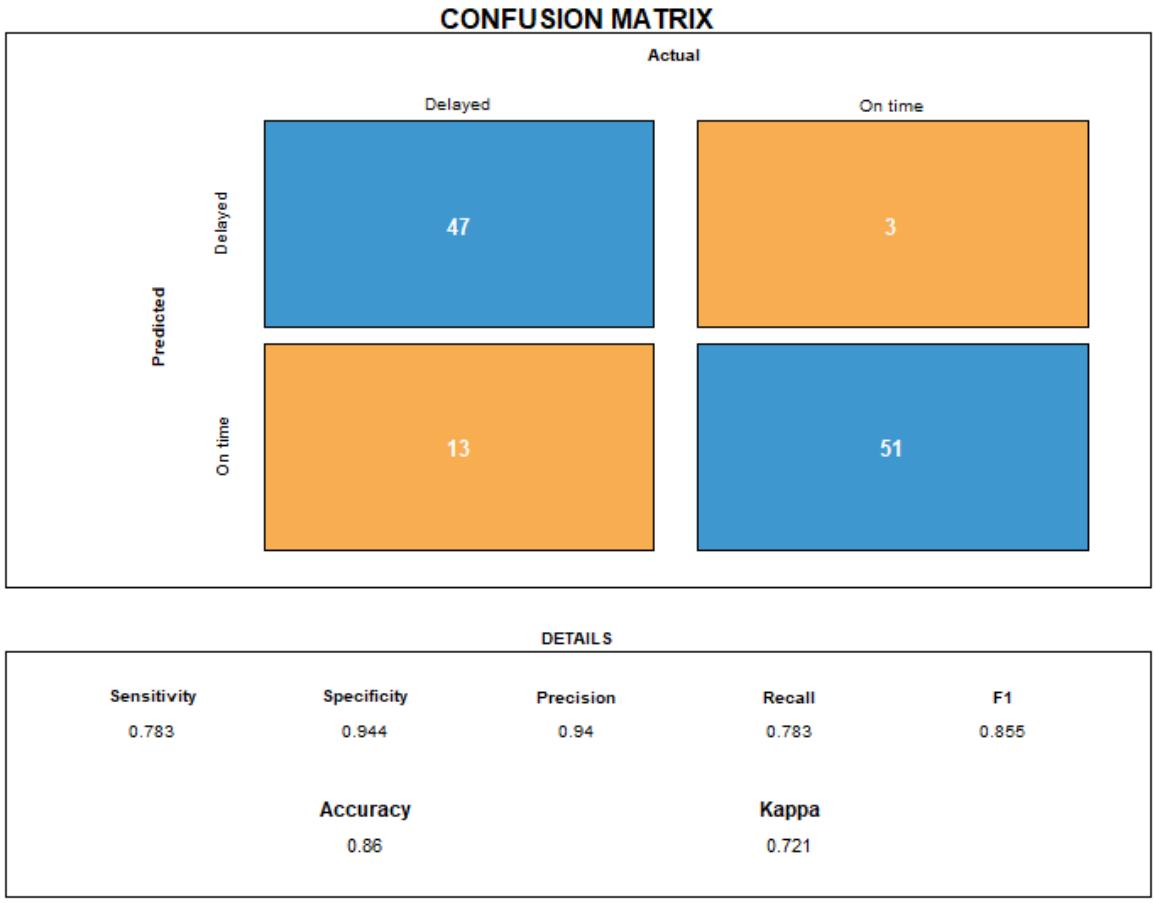
Figura 37 – Formato da avaliação

		Tarefa com atraso		
		Negativo	Positivo	
Predição de atraso	Negativo	Real Negativo (RN)	Falso Negativo (FN)	Valor preditivo negativo (NPV)
	Positivo	Falso Positivo (FP)	Real Positivo (TP)	Valor preditivo positivo (PPV)
		Especificidade	Sensibilidade	**Eficiência global**

Fonte: Elaborado pelo Autor

Com o objetivo de avaliar a eficiência das predições, 114 tarefas de 13 projetos diferentes foram selecionadas. As predições realizadas indicaram a probabilidade de cada tarefa ser concluída após a data prevista, e esta saída foi comparada com o estado real de conclusão da tarefa na base de dados original. A Figura 38 mostra os resultados estatísticos das predições realizadas.

Figura 38 – Resultado das estatísticas sobre a predição



Fonte: Elaborado pelo Autor

Analisando a matriz de confusão gerada pelas predições, onde as linhas mostram a predição e as colunas indicam o resultado real da tarefa, percebe-se que apenas 2 tarefas que terminaram efetivamente atrasadas não foram identificadas como tal pelo modelo, enquanto 45 foram identificadas corretamente. Nos testes realizados para avaliação, a sensibilidade do modelo - ou seja, a capacidade do modelo em indicar corretamente quando determinado evento efetivamente ocorreu - ficou acima de 78%. Já os dados de especificidade, que indicam a eficácia em identificar quando um evento não ocorre, se mostraram ainda mais altos, acima de 94%. Considerando estes resultados, percebe-se que o modelo teve uma precisão de

94%, e uma eficiência global de aproximadamente 86% na indicação de atraso das tarefas. Assim, percebe-se a validade do modelo proposto no cenário avaliado, visto que obteve um fator aceitável global de acerto nas predições.

Ao realizar as predições, também foram avaliadas questões técnicas como custo de processamento e tempo de execução. O ambiente utilizado para a realização dos testes foi um servidor local, com processador Core i7 de 2.3 GHz, 8 GB de memória RAM e disco SSD, onde todos os componentes do sistema foram executados (banco de dados relacional e não-relacional, serviço de filas, agentes e servidores de aplicação). Considerando a base de dados utilizada, com mais de 40 projetos e 450 tarefas, a cada execução dos mecanismos de predição, o protótipo levou em média 150 segundos para enviar as recomendações para o gestor. No pior caso encontrado, o tempo total de execução foi de 470 segundos.

6 CONSIDERAÇÕES FINAIS

Esta dissertação apresentou o modelo Kairós, criado com o objetivo de auxiliar de maneira proativa o gerente de projetos na identificação de problemas na execução das tarefas com impacto no cronograma. Este modelo foi construído utilizando conceitos de computação ubíqua e sistemas de múltiplos agentes, realizando predições e recomendações com base em contextos históricos das entidades relevantes.

O capítulo 2 mostrou a importância dos conceitos fundamentais para esta pesquisa, apresentando uma visão geral de cada área de conhecimento com ênfase no objetivo do trabalho. As áreas de gestão de tempo em projetos, computação ubíqua e sistemas de predição e recomendação foram abordadas, para que fosse possível embasar teoricamente o restante do trabalho.

No capítulo 3, foi apresentado um estudo de mapeamento sistemático realizado na área de predições em cronogramas de projetos. Através deste estudo, foi possível identificar as características do estado da arte desta área, bem como perceber oportunidades de pesquisa que auxiliaram no delineamento desta proposta.

As funcionalidades e requisitos do modelo Kairós foram apresentadas no capítulo 4, utilizando ainda diagramas UML para facilitar a compreensão da estrutura elaborada. Ainda neste capítulo foi apresentada a arquitetura e as tecnologias envolvidas na construção do modelo.

Finalmente, o capítulo 5 trouxe a implementação de um protótipo, com o objetivo de auxiliar no processo de validação do modelo criado, juntamente do modelo de dados e um esboço das principais telas. O método proposto para a avaliação do protótipo também foi apresentado, com a proposição de uma validação cruzada em *holdout*, bem como um cenário de exemplo de uso e os resultados encontrados pela avaliação.

6.1 Conclusões e trabalhos futuros

Os resultados obtidos através da avaliação do protótipo mostraram a viabilidade e a efetividade do modelo no cenário de exemplo analisado. Considerando o uso de dados reais, pode-se concluir que a aplicação do modelo em um cenário real poderia trazer ganhos, evitando inúmeros problemas nos cronogramas de projeto.

O estudo apresentou duas questões de pesquisas que guiaram seu desenvolvimento, e com a finalização do trabalho, foi possível responde-las de maneira satisfatória. Quanto à questão “Como realizar predições sobre a possibilidade de atraso em uma tarefa durante sua execução?”, pode-se afirmar que o uso das técnicas empregadas, como a análise de similaridades em históricos de contextos e a predição por analogia, são maneiras viáveis de realizar as predições sobre o estado final de uma tarefa.

Já em relação a segunda questão (“Como sugerir ações corretivas para tarefas que apresentam problemas?”), é possível afirmar que o uso de uma base de dados de melhores práticas de projeto, classificada e alimentada com avaliações e sugestões dos usuários, foi uma solução satisfatória. Ainda, o uso de recomendações – ou seja, a exibição de soluções de modo proativo – fez com que o mecanismo de sugestões tivesse uma maior possibilidade de aceitação pelo gestor.

Como trabalhos futuros, é possível destacar a ampliação do modelo de dados utilizado como referência para as predições, considerando um número maior de variáveis, para que as predições tenham melhor assertividade. Ainda, é viável a implementação de algoritmos de aprendizagem de máquina em uma versão futura, visando obter melhores resultados nas análises e, por consequência, predições mais corretas.

Ainda, pode ser realizada a implementação de outras percepções de problemas na tarefa além do atraso, como falhas de qualidade ou estimativas incoerentes. Finalmente, o mecanismo de recomendações pode evoluir de modo a reconhecer padrões de solução aplicado em outras tarefas que possuíam probabilidade de atraso, mas que receberam alguma correção para que fossem finalizadas dentro do prazo.

6.2 Contribuições

O modelo Kairós diferencia-se das demais propostas por contribuir com o processo de execução e controle das tarefas, e não apenas com o planejamento do cronograma no início do projeto. Conforme o estudo mostrado no capítulo 3, aproximadamente 85% dos trabalhos encontrados tem foco na fase de planejamento. Porém, mesmo com a otimização do cronograma inicial, a existência de imprevistos ou desvios pode impactar em todo o restante do cronograma, evidenciando a importância de um modelo que acompanhe todo o ciclo de vida do projeto.

Outra contribuição trazida pelo modelo é a utilização de históricos de contextos nos mecanismos de predição, técnica ainda não abordada na área de cronogramas de projetos. No entanto, esta técnica é amplamente utilizada em outras áreas, como modelos aplicados à saúde, acessibilidade, educação, comércio, entre outros (BARBOSA, 2015). Além disso, o uso de dados históricos para a avaliação da situação atual já é uma técnica amplamente utilizada pelos gerentes de projetos, porém de forma manual. Ao utilizar o modelo aqui proposto, os algoritmos consideram todo o histórico de maneira automatizada, não dependendo assim da experiência do gerente.

O Kairós ainda contribui ao especializar-se em projetos de desenvolvimento de software, que tem características bastante específicas que os diferenciam das demais áreas de negócio, como a intangibilidade do produto, as recorrentes alterações de escopo e a execução do projeto por equipes geograficamente distribuídas. A abordagem de múltiplos modelos de desenvolvimento como SCRUM e PMBOK também pode ser considerada uma contribuição, visto que a maior parte da literatura da área aborda apenas os métodos tradicionais, baseados em desenvolvimento em cascata.

A natureza proativa do modelo, trazida pelo uso de recomendações, é um dos diferenciais apresentados. Os modelos existentes para a predição de cronogramas costumam atuar de modo dependente de uma ação específica do usuário, que deve solicitar uma predição para que a mesma seja feita. Em um cenário complexo e acelerado como o atual, a necessidade de visualizar e solicitar recomendações de modo manual faz com que se percam inúmeras chances de prever os problemas antes que aconteçam.

REFERÊNCIAS

ABDEL-HAMID, T. K., and MADNICK, S. E., “The dynamics of software project scheduling” in Communications of the ACM, 1983, 26(5), 340–346. https://doi.org/10.1145/69586.358135

ABOROKBAH, M. M., AL-MUTAIRI, S., SANGAIAH, A. K., and SAMUEL, O. W. “Adaptive context aware decision computing paradigm for intensive health care delivery in smart cities—A case analysis”. Sustainable Cities and Society. 2018. https://doi.org/10.1016/j.scs.2017.09.004

ABOWD. G. D. 2012. “What next, ubicomp?: celebrating an intellectual disappearing act”. In Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12). ACM, New York, NY, USA, 31-40. DOI=http://dx.doi.org/10.1145/2370216.2370222

ADOMAVICIUS, G., TUZHILIN, A. “Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions”. IEEE Transactions on Knowledge and Data Engineering, Vol. 17, NO. 6, JUNE 2005.

AMEYED, D., MIRAOUI, M. and TADJ, C. “A Survey of Prediction Approach in Pervasive Computing,” International Journal of Scientific & Engineering Research, vol.6, no.5, pp. 1–11, 2015.

ANYA, O. and TAWFIK, H. “Designing for practice-based context-awareness in ubiquitous e-health environments”, Computers & Electrical Engineering, Vol. 61, 2017, Pages 312-326, ISSN 0045-7906, https://doi.org/10.1016/j.compeleceng.2016.08.012.

ARTIGUES, C., LEUS, R., and NOBIBON, F. T., "Robust optimization for resource-constrained project scheduling with uncertain activity durations" in Flexible Services and Manufacturing Journal, 2013, vol. 25, no. 1–2, pp. 175–205.

ASHTIANI, B., LEUS, R., and ARYANEZHAD, M. B., "New competitive results for the stochastic resource-constrained project scheduling problem: Exploring the benefits of pre-processing" in Journal of Scheduling, 2011, vol. 14, no. 2, pp. 157–171.

BAIA, D. M., "An Integrated Multi-Agent-Based Simulation Approach to Support Software Project Management" in Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on, 2015, vol. 2, pp. 911–914.

BALLESTÍN, F., and BLANCO, R., "Theoretical and practical fundamentals for multi-objective optimisation in resource-constrained project scheduling problems" in Computers & Operations Research, 2011, vol. 38, no. 1, pp. 51–62.

BALLESTÍN, F., and LEUS, R., "Resource-constrained project scheduling for timely project completion with stochastic activity durations" in Production and Operations Management, 2009, vol. 18, no. 4, pp. 459–474.

BARBOSA, J. L. V. “Ubiquitous Computing: Applications and Research Opportunities” (Invited Talk). VI IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Madurai, Índia, p.1–8, 2015. Disponível em: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7435625

BARBOSA, J. L. V., MARTINS, C. J., FRANCO, L. K., BARBOSA, D. N. F. TrailTrade: A model for trail-aware commerce support. Computers In Industry , v. 80, p. 43-53, 2016.

BARBOSA, J. L. V., TAVARES, J. E. R., CARDOSO, I. G., MOTA, B., MARTINI, B. G. TrailCare: an Indoor and Outdoor Context-aware System to Assist Wheelchair Users. International Journal Of Human-Computer Studies , v. 116, p. 1-14, 2018.

BATISTA, M. H. E., TAVARES, J. E. R., BARCELOS,G., FILIPPETTO, A. and BARBOSA, J. “Chronos: A model for Ubiquitous Project”. In: IADIS International Conference on Applied Computing, Rio de Janeiro. v. 1. p. 1-5, 2011.

BERNDT, D. J., CLIFFORD, J. “Using dynamic time warping to find patterns in time series”. KDD workshop, pp. 359–370. Seattle, WA, 1994.

BERTAZZONI, C. and GIANNOTTI, F., "RASP: A Resource Allocator for Software Projects" in Proceedings of the 3rd International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, 1990, Volume 2, pp. 628–637.

BURBEY, I. and MARTIN, T. L. “A Survey on Predicting Personal Mobility”. International Journal of Pervasive Computing and Communications. 2012. [Online]. 8(1), pp. 5–22. Disponível em: http://dx.doi.org/10.1108/17427371211221063

CHANG, C. K., JIANG, H., DI, Y., ZHU, D., and GE, Y., "Time-line based model for software project scheduling with genetic algorithms" in Information and Software Technology, 2008, vol. 50, no. 11, pp. 1142–1154.

CHICANO, F., CERVANTES, A., LUNA, F., and RECIO, G., "A novel multiobjective formulation of the robust software project scheduling problem" in Lecture Notes in Computer Science, 2012, vol. 7248 LNCS, pp. 497–507.

CHICANO, F., LUNA, F., NEBRO, A., and ALBA, E., "Using Multi-objective Metaheuristics to Solve the Software Project Scheduling Problem" in Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO ’11), 2011, pp. 1915–1922.

CHOETKIERTIKUL, M., DAM, H. K., TRAN, T., and GHOSE, A., "Predicting delays in software projects using networked classification" in Proceedings - 2015 30th

IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, 2015, pp. 353–364.

CHOETKIERTIKUL, M., DAM, H. K., TRAN, T., and GHOSE, A., "Predicting the delay of issues with due dates in software projects" in Empirical Software Engineering, 2017, vol. 22, no. 3, pp. 1223–1263.

CRESPO, D., and RUIZ, M., "Decision making support in CMMI Project Planning Process Area using a hybrid simulation model" in Computer Languages, 2012, p. 399:1--399:12.

DEY, A., SALBER, D. and ABOWD, G. “A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware application”. Human-Computer Interaction. [Online]. 2001. 16(2), pp. 97–166. Disponível em: http://dx.doi.org/10.1207/S15327051HCI16234_02

DINSMORE, P. C. and CAVALIERI, A. Como se Tornar um Profissional em Gerenciamento de Projetos, 4 ed. Rio de Janeiro, Brasil: Qualitymark, 2011.

DREZET, L. E., and BILLAUT, J. C., "A project scheduling problem with labour constraints and time-dependent activities requirements" in International Journal of Production Economics, 2008, vol. 112, no. 1, pp. 217–225.

ESPOSITO, M., MINUTOLO, A., MEGNA, R., FORASTIERE, M., MAGLIULO, M., and DE PIETRO, G. “A smart mobile, self-configuring, context-aware architecture for personal health monitoring”. Engineering Applications of Artificial Intelligence, 2018, 67, 136–156. https://doi.org/10.1016/j.engappai.2017.09.019

FENTON, N., MARSH, W., NEIL, M., CATES, P., FOREY, S., and TAILOR, M., "Making resource decisions for software projects" in Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on, 2004, pp. 397–406.

FILIPPETTO, A. S., BARBOSA, J. L. V., FRANCISCO, R. and KLEIN, A. Z. “A Project Management Model based on an Activity Theory Ontology”. In: XLII Latin American Computing Conference, 2016, Valparaíso, Chile.

FILIPPETTO, A. S., LIMA, R. K., BARBOSA, J. L. V. “Lean Risk - Gestão de Riscos em Times Distribuídos”. Universo PM, v. 02, p. 07-13, 2017.

GASPARIC, M., MURPHY, G. C. and RICCI, F. “A context model for IDE-based recommendation systems”. Journal of Systems and Software, Volume 128, 2017, Pages 200-219, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2016.09.012.

GONÇALVES, J. F., MENDES, J. J. M., and RESENDE, M. G. C., "A genetic algorithm for the resource constrained multi-project scheduling problem" European Journal of Operational Research, vol. 189, no. 3, pp. 1171–1190, 2008.

GOWER, J. C. “A General Coefficient of Similarity and Some of Its Properties”. Biometrics, 1971, 27(4), 857. https://doi.org/10.2307/2528823

HEGAZY, T., and PETZOLD, K., "Genetic optimization for dynamic project control" Journal of Construction Engineering and Management, vol. 129, no. 4, pp. 396–404, 2003.

HEIDRICH, L., BARBOSA, J. L. V., CAMBRUZZI, W. L., RIGO, S. J., MARTINS, M. G., SANTOS,R. Diagnosis of Learner Dropout Based on Learning Styles for Online Distance Learning. Telematics and Informatics , v. 1, p. 1-30, 2018.

HIGHTOWER, J. and BORRIELLO, G "Location Systems for Ubiquitous Computing". Computer. 2001. [Online]. 34(8), pp. 57–66. Disponível em: http://dx.doi.org/10.1109/2.940014

HILL, W., STEAD, L., ROSENSTEIN, M. and FURNAS, G. "Recommending and evaluating choices in a virtual community of use". In Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '95 (pp. 194–201). 1995. https://doi.org/10.1145/223904.223929

HONG, J., SUH, E. H., KIM, J. and KIM, S. "Context-aware system for proactive personalized service based on context history". Expert Systems with Applications. 2009. [Online]. 36(4), pp. 7448–7457. Disponível em: http://dx.doi.org/10.1016/j.eswa.2008.09.002

HUNT, B., "Parametric Project Monitoring and Control: Performance-Based Progress Assessment and Prediction" in Aerospace Conference, 2007 IEEE, 2007, pp. 1–12.

JONES, E. F. "Scheduling 101—the basic of best practices". Paper presented at PMI® Global Congress 2009 -North America, Orlando, FL. 2009. Newtown Square, PA: Project Management Institute.

JOSLIN, D. and POOLE, W., "Agent-based simulation for software project planning" in Proceedings of the 2005 Winter Simulation Conference, 2005, pp. 1059–1066.

KACZOROWSKA, A. "Traditional and Agile Project Management in Public Sector and ICT". In: 2015 Federated Conference on Computer Science and Information Systems (FedCSIS), Lodz, 2015, pp. 1521-1531. doi: 10.15439/2015F279. Disponível em: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7321626&isnumber=7321412

KAROVA, M., and AVRAMOVA, N., "A Genetic Algorithm basic approach for software management project" in ACM International Conference Proceeding Series, 2012, pp. 103–110.

KITCHENHAM, B. and CHARTERS, S., "Guidelines for Performing Systematic Literature Reviews in Software Engineering", Tech. rep., Technical report, EBSE Technical Report EBSE-2007-01, 2007.

KOHAVI, R. "A study of cross-validation and bootstrap for accuracy estimation and model selection". In: IJCAI'95 Proceedings of the 14th international joint conference

on Artificial intelligence - Volume 2, Canada, 1995, pp. 1137-1143. Disponível em: https://dlnext.acm.org/doi/abs/10.5555/1643031.1643047

KONIG, I., VOIGTMANN, C., KLEIN, B. N. and DAVID, K. “Enhancing alignment based context prediction by using multiple context sources: experiment and analysis”. In: Proceedings of the 7th International and Interdisciplinary Conference on Modeling and Using Context, Karlsruhe, Germany, 2011, pp. 159-172. [Online]. Disponível em: http://dx.doi.org/10.1007/978-3-642-24279-3_18

KONSTAN J. A. and RIEDL J. “Recommender systems: from algorithms to user experience”. User Model User-Adap Inter. 2012. 22:101–123. DOI 10.1007/s11257-011-9112-x.

KROLL, J., FRIBOIM, S. and HEMMATI, H., "An Empirical Study of Search-Based Task Scheduling in Global Software Development" in 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track, 2017, pp. 183–192.

KUSNER, M. J., SUN, Y., KOLKIN, N. I., WEINBERGER, K. Q., “From word embeddings to document distances” in ICML’15 Proceedings of the 32nd International Conference on International Conference on Machine Learning, 2015, 37, 957-966.

LOPEZ-MARTIN, C., CHAVOYA, A. and MEDA-CAMPANA, M. E., “A Fuzzy Logic Model for Predicting the Development Schedule of Software Projects” in 12th International Conference on Information Technology - New Generations, 2015, 415–420. https://doi.org/10.1109/ITNG.2015.73

LIU, C. L. CDNFRE: Conflict detector in non-functional requirement evolution based on ontologies. Computer Standards and Interfaces, 47, 62–76. 2016. https://doi.org/10.1016/j.csi.2016.03.002

MAJID, A., CHEN, L., CHEN, G., MIRZA, H. T., HUSSAIN, I. and WOODWARD, J. “A context-aware personalized travel recommendation system based on geotagged social media data mining”. International Journal of Geographical Information Science, 27(4), 662–684. 2013. https://doi.org/10.1080/13658816.2012.696649

MINKU, L. L., SUDHOLT, D. and YAO, X. "Improved Evolutionary Algorithm Design for the Project Scheduling Problem Based on Runtime Analysis". In IEEE Transactions on Software Engineering, 2014, vol. 40, no. 1, pp. 83–102.

NAJAR, S. PINHEIRO, M. K. and SOUVEYET, C. "A new approach for service discovery and prediction on Pervasive Information System," Procedia computer science, vol. 32, pp. 421-428, 2014.

NIGAR, N. “Model-based dynamic software project scheduling” in Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering - ESEC/FSE, 2017, pp. 1042–1045, New York, NY, USA: ACM. https://doi.org/10.1145/3106237.3119879

PETERSEN, K., FELDT, R., MUJTABA, S. and MATTSSON, M., "Systematic mapping studies in software engineering", in: 12th International Conference on Evaluation and Assessment in Software Engineering, 2008, vol. 17, p. 1.

PETERSEN, K., VAKKALANKA, S. and KUZNIARZ, L., "Guidelines for conducting systematic mapping studies in software engineering: An update", In Information and Software Technology, 2015, Volume 64, Pages 1-18, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2015.03.007.

PITTOLI, F., VIANNA, H. D., BARBOSA, J. L. V., BUTZEN, E., GAEDKE, M., DA COSTA, J. S. D., DOS SANTOS, R. B. S. An Intelligent System for Prognosis of Noncommunicable Diseases Risk Factors. Telematics and Informatics , v. 1, p. 1-34, 2018.

PMI (Project Management Institute). “A Guide to the Project Management Body of Knowledge (PMBOK)”, 2013, 5th ed. Newtown Square, PA, EUA.

PMI (Project Management Institute). “Delivering Value – Focus on Benefits during Project Exeuction”, in Pulse of the Profession: In-Depth Report, 2016, Newtown Square, PA, EUA.

PODANI, J. “Extending Gower’s general coefficient of similarity to ordinal characters”. Taxon, 1999, 48(2), 331–340. https://doi.org/10.2307/1224438

PRABAKARAN, N. and KANNADASAN, R. “Contexts enabled Decision Making using sensors to perceive pervasive environment”. Procedia Computer Science, Volume 132, 2018, Pages 477-485, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2018.05.145.

RAMIREZ-NORIEGA, A., JUAREZ-RAMIREZ, R., NAVARRO, R. and LOPEZ-MARTINEZ, J. “Using Bayesian Networks to Obtain the Task's Parameters for Schedule Planning in Scrum”. 4th International Conference in Software Engineering Research and Innovation (CONISOFT), Puebla, 2016, pp. 167-174. doi: 0.1109/CONISOFT.2016.33 Disponível em: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7477927&isnumber=7477893

RAMZAN, M., JAFFAR, A., IQBAL, A., ANWAR, S., RAUF, A. and SHAHID, A. A. “Project scheduling conflict identification and resolution using genetic algorithms (GA),” in Telecommunication Systems, 2012, vol. 51, no. 2–3, pp. 167–175.

RESNICK, P., IAKOVOU, N., SUSHAK, N., BERGSTROM, P. and RIEDL, J. “GroupLens: An Open Architecture for Collaborative Filtering of Netnews”. Proc. 1994 Computer Supported Cooperative Work Conf. 1994.

RIGO, S. J., CAMBRUZZI, W. L. and BARBOSA, J. L. V. “Dropout Prediction and Reduction in Distance Education Courses with the Learning Analytics Multitrail Approach,” Journal of Universal Computer Science, vol. 21, no. 1, pp. 23–47, 2015.

ROSA, J. H., BARBOSA, J. L. V. and BARCELOS, G. O. “ORACON: An Adaptive Model For Context Prediction”. Expert Systems with Applications. 2016. [Online]. 45(1), pp. 56–70. Disponível em: http://dx.doi.org/10.1016/j.eswa.2015.09.016

ROSA, J. H., BARBOSA, J. L. V., KICH, M. R. and BRITO, L. K. “A Multi-Temporal Context-aware System for Competences Management”. International Journal of Artificial Intelligence in Education. 2015. [Online]. 25(4), pp. 455–492. Disponível em: http://dx.doi.org/10.1007/s40593-015-0047-y

SATYANARAYANAN, M. “Pervasive Computing: Vision and Challenges”, In: IEEE Personal Communications. Pág. 10-17. 2001. [Online]. 8(4), pp. 10–17. Disponível em: http://dx.doi.org/10.1109/98.943998.

SHISHEHCHI, S. BANIHASHEM, S. Y. ZIN, N. A. M. and NOAH, S. A. M. "Review of personalized recommendation techniques for learners in e-learning systems," 2011 International Conference on Semantic Technology and Information Retrieval, Putrajaya, 2011, pp. 277-281. doi: 10.1109/STAIR.2011.5995802

SIGG, S., GORDON, D., VON ZENGEN, G., BEIGL M., HASELOFF, S. and DAVID, K. “Investigation of context prediction accuracy for different context abstraction levels”. IEEE Transactions on Mobile Computing. 2011. [Online]. 11(6), pp. 1047–1059. Disponível em: http://dx.doi.org/10.1109/TMC.2011.170

SIGG, S., HASELOFF, S. and DAVID, K. An alignment approach for context prediction tasks in ubicomp environments. IEEE Pervasive Computing. 2010. [Online]. 9(4), pp. 90–97. Disponível em: http://dx.doi.org/10.1109/MPRV.2010.23

SOUSA, H. T., SILVA, T. S., SANTOS, P. S. J., CALHAU, R. F. and COSTA, M. B., "Automated Support for Scrum Projects Sprint Planning". 2015. Proceedings of the XI Brazilian Symposium on Information Systems (SBSI 2015). 62.

VIANNA, H. D., BARBOSA, J. L. V. In Search of Computer-aided Social Support in Non-communicable Diseases Care. Telematics and Informatics , v. 34, p. 1419-1432, 2017.

VIANNA, H. D., BARBOSA, J. L. V., PITTOLI, F. In the Pursuit of Hygge Software. IEEE Software , v. 34, p. 48-52, 2017.

VRKIĆ, D., "Are they a perfect match? Analysis of usage of author suggested keywords, IEEE terms and social tags," 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, 2014, pp. 732-737. doi: 10.1109/MIPRO.2014.6859662

WAGNER, A., BARBOSA, J. L. V. and BARBOSA, D. N. F. “A Model for Profile Management Applied to Ubiquitous Learning Environments”. Expert Systems with Applications. [Online]. 41(4), pp. 2023–2034. 2014. Disponível em: http://dx.doi.org/10.1016/j.eswa.2013.08.098

WANG, X., WU, C. and MA, L., "Software project schedule variance prediction using Bayesian Network" 2010 IEEE International Conference on Advanced Management Science, pp. 26–30, Jul. 2010.

WEI, J., HE, J., CHEN, K., ZHOU, Y., and TANG, Z. Collaborative filtering and deep learning based recommendation system for cold start items. Expert Systems with Applications, 69, 1339–1351. 2017. https://doi.org/10.1016/j.eswa.2016.09.040

WEISER, M. “The computer for the 21st century.” In Scientific American, pp. 94-104, Setembro. 1991. [Online]. 265(3), pp. 94–104. Disponível em: http://dx.doi.org/10.1145/329124.329126.

WEISS, G. “Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence”. 1999. Cambridge MA: MIT Press. https://doi.org/10.1007/s10709-010-9480-x

WIEDEMANN, T. SIMCOP: “Um Framework para Análise de Similaridade em Sequências de Contextos” 125 f.: il.; 30 cm. Dissertação (mestrado)—Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, São Leopoldo, 2014.

WOHLIN, C., “Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering”, Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. 2014. London, England, BC, United Kingdom. http://dx.doi.org/10.1145/2601248.2601268

WONG, F., LEE, S., WONG, Q. and LEE, S. “Points of Interest Recommendation Based on Context-aware,” International Journal of Hybrid Information Technology, vol. 8, no. 3, pp. 55–62, 2015.

XIAO, J., GAO, M., and HUANG, M., "Empirical Study of Multi-objective Ant Colony Optimization to Software Project Scheduling Problems" in Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, 2015, pp. 759–766.

XIAO, J., OSTERWEIL, L. J., WANG, Q., and LI, M., “Dynamic Resource Scheduling in Disruption-Prone Software Development Environments” in 13th International Conference on Fundamental Approaches to Software Engineering, 2010, 6013, 107–122. https://doi.org/10.1007/978-3-642-12029-9_8

XIONG, J, LEUS, R., YANG, Z., and ABBASS, H. A., "Evolutionary multi-objective resource allocation and scheduling in the Chinese navigation satellite system project" Eur. J. Oper. Res., Vol. 251, no. 2, pp. 662–675, 2016.

YANG, H. L., and Wang, C. S., "Recommendation system for IT software project planning: A hybrid mining approach for the revised CBR algorithm" in 5th International Conference Service Systems and Service Management - Exploring Service Dynamics with Science and Innovative Technology, ICSSSM’08, 2008, pp. 1–5.

YANG, H. L., and WANG, C. S., "Recommender system for software project planning one application of revised CBR algorithm" Expert Systems with Applications, vol. 36, no. 5, pp. 8938–8945, 2009.

ZHANG, G. P., KEIL, M., RAI, A., and MANN, J., "Predicting information technology project escalation: A neural network approach" European Journal of Operational Research, vol. 146, no. 1, pp. 115–129, 2003.

ZHANG, M. A. B. W., YANG, Y., XIAO, J., and LIU, X., "Ant colony algorithm-based scheduling for handling software project delay" in ICSSP Conf Abstract, 2015, pp. 52–56.