Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Uma arquitetura para tomada de decisão pedagógica com dados integrados

Társis Marinho de Souza

Campina Grande, Paraíba, Brasil

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Uma arquitetura para tomada de decisão pedagógica com dados integrados

Társis Marinho de Souza

Tese submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Doutor em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Modelos computacionais e cognitivos

Prof. Dr. Evandro de Barros Costa
(Orientador)

Campina Grande, Paraíba, Brasil

"UMA ARQUITETURA PARA TOMADA DE DECISÃO PEDAGÓGICA COM DADOS INTEGRADOS"

TÁRSIS MARINHO DE SOUZA

TESE APROVADA EM 16/11/2017

EVANDRO DE BARROS COSTA, Dr., UFAL
Orientador(a)

CLÁUDIO DE SOUZA BAPTISTA, PhD., UFCG
Examinador(a)

JOÃO ARTHUR BRUNET MONTEIRO, Dr., UFCG
Examinador(a)

ROBSON DO NASCIMENTO FIDALGO, Dr., UFPE
Examinador(a)

PATRICK HENRIQUE DA SILVA BRITO, Dr., UFAL
Examinador(a)

CAMPINA GRANDE - PB

S729a Souza, Társis Marinho de.
Uma arquitetura para tomada de decisão pedagógica com dados integrados / Társis Marinho de Souza. – Campina Grande, 2017.
129 f. : il. color.

Tese (Doutorado em Ciências da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2017.
"Orientação: Prof. Dr. Evandro de Barros Costa".
Referências.

1. Arquitetura de Software. 2. Arquitetura de Software - Tomada de Decisão Pedagógica. 3. Tomada de Decisão – Dados Integrados. I. Costa, Evandro de Barros. II. Título.

CDU 004.2(043)

FICHA CATALOGRÁFICA ELABORADA PELO BIBLIOTECÁRIA ITAPUANA SOARES DIAS CRB = 15/93

Resumo

Há uma crescente demanda por educação online e em sua realização ocupa papel de relevo os denominados ambientes virtuais de aprendizagem, ou em uma acepção mais geral as plataformas educacionais, responsáveis pelo provimento de recursos digitais e suporte principalmente a estudantes e professores. Os professores desempenham papéis de alta complexidade que vão da gerência dos cursos online ao suporte pedagógico oferecido aos estudantes, estes em turmas cada vez maiores, portanto, envolvendo tarefas sofisticadas de tomada de decisão. Para tanto, estes professores necessitam cada vez mais de informação de alta qualidade, permitindo efetividade em suas compreensões em detalhes do que ocorre com os estudantes interagindo no ambiente online, e assim impactando em suas tomadas de decisões. Neste ponto, ressalta-se que as interações dos estudantes em tais cursos geram grande quantidade e diversidade de dados, nesse caso, tem-se buscado extrair informações pedagogicamente relevantes, normalmente, usando-se técnicas de mineração de dados ou de *learning analytics*. No entanto, constata-se uma lacuna importante no suporte aos professores, dado que as plataformas educacionais atuais apresentam várias dificuldades relacionadas à complexidade em monitorar e avaliar os estudantes envolvidos nesses cursos, bem como a falta e/ou dificuldade em acessar informações relevantes que apoiem o processo de tomada de decisão pedagógica. Além disso, não se assume aqui que os professores tenham domínio das ferramentas técnicas para processar o grande volume de dados envolvidos. Assim, com vistas a um posterior atendimento efetivo às demandas do professor, nesta pesquisa focalizou-se uma problemática que diz respeito a aspectos de engenharia de software em educação, em um contexto de *business intelligence* educacional, buscando-se oferecer aos projetistas e desenvolvedores de softwares uma infraestrutura necessária para a construção de soluções que venham atender à demanda de informação dos professores, nos termos anteriormente mencionados. Nesta tese, propõe-se uma infraestrutura de software adequada para construção de soluções efetivas no provimento de informações relevantes no apoio à tomada de decisão pedagógica. Com isso, pretende-se auxiliar desenvolvedores de três formas complementares: (1) favorecer o projeto arquitetural do software, de modo a satisfazer requisitos de qualidade fundamentais para esse tipo de sistema, tais como escalabilidade e facilidade de evolução; (2) favorecer o reúso de código-fonte, aliado a padrões de projeto; (3) favorecer a integração

com ferramentas existentes, tais como *frameworks* e ambientes virtuais de aprendizagem. Os resultados alcançados mostram que a solução proposta, de fato, atende aos requisitos de qualidade pretendidos e facilita a integração com plataformas educacionais, além de favorecer o reuso de padrões de projeto e *frameworks* de desenvolvimento, alcançando uma taxa de reuso de cerca de 88%. Os atendimentos aos requisitos de qualidade foram avaliados em um experimento que adotando uma abordagem baseada em cenários de execução e utilizando uma base de dados real de um curso de sistemas de informação da modalidade a distância. Os cenários de execução utilizados são realistas e foram sugeridos em reuniões de *brainstorm* com especialistas e estudantes de mestrado.

Abstract

There is a growing demand for online education and in its accomplishment play a very important role the so-called virtual learning environments, or in a more general sense the educational platforms, responsible for the provision of digital resources and support mainly to students and teachers. Teachers play highly complex roles ranging from the management of online courses to the pedagogical support offered to students, these in larger and larger classes, thus involving sophisticated decision-making tasks. To do so, these teachers increasingly require high-quality information, allowing effectiveness in their understandings in detail of what academically happens to students interacting in the online environment, and thus impacting their understandings in pedagogical decision-making processes within online learning environments. At this point, it is important to emphasize that the interactions of students in such courses generate a large amount and diversity of data, in this case, allowing the extraction of pedagogically relevant information, usually using data mining techniques or learning analytics. However, there is a significant lack in the support of teachers, since the current educational platforms present several difficulties related to the complexity in monitoring and evaluating the students involved in these courses, as well as the lack and / or difficulty in accessing relevant information that support the pedagogical decision-making process. Moreover, it is not expected that teachers master the technical tools to process all the involved data. Thus, with a view to a subsequent effective attendance to the demands of the teacher, this research focused on a problem that concerns aspects of software engineering in education, in a context of educational business intelligence, seeking to offer designers and software developers, an infrastructure necessary for the construction of solutions that meet the information demand of teachers, under the terms mentioned above. In this thesis, we propose a suitable software infrastructure for the construction of effective solutions in the provision of relevant information for supporting pedagogical decision making. In this way, it is intended to help developers in three complementary ways: (1) to favor the architectural design of the software, in order to satisfy fundamental quality requirements for this type of system, such as scalability and ease of evolution; (2) favoring the reuse of source code, together with design patterns; (3) favor integration with existing tools such as frameworks and virtual learning environments. The obtained results show that the proposed solution does, in

fact, meet the desired quality requirements and facilitates integration with educational platforms, as well as favor the reuse of design patterns and development frameworks, achieving a reuse rate of about 88% . The attendances to the quality requirements were evaluated in an experiment that adopts an approach based on execution scenarios and using a real database of a course of distance information systems. The execution scenarios used are realistic and have been suggested in brainstorming meetings with experts and master's students.

Agradecimentos

Agradeço, primeiramente, a Deus por me oportunizar chegar até aqui e concluir mais esta etapa.

Aos meus pais, Juraci e Miriam, pela esforço, dedicação e amor dedicados durante toda minha vida para que eu pudesse chegar até aqui e sem os quais, sem dúvida, eu não seria nada.

A minha irmã, Talita (*In memoriam*), que tanto sonhou e esperou por esse momento, mas que, infelizmente, não está aqui hoje para compartilhar comigo essa alegria. Este momento também é seu Paita.

A minha família por todo o apoio dado durante todo o desenvolvimento deste trabalho.

Ao meu orientador, Prof. Dr. Evandro de Barros Costa, pela paciência, ensinamentos e todo o apoio a mim dedicados durante o desenvolvimento deste trabalho e da minha vida acadêmica. Ensinamentos que levarei para toda vida

Aos amigos que sempre apoiaram e tanto ajudaram para que eu pudesse concluir mais esta etapa tão importante. Em especial, aos amigos Bruna, Cristina, Clívia, Elvys e Ray.

Aos amigos e companheiros que ganhei no laboratório TIPS e que diariamente me apoiaram e contribuíram com este trabalho. Valeu, Pessoal.

A todos os companheiros do IFAL - *campus* Arapiraca pelo apoio.

Conteúdo

Lista de Símbolos

DM - *Data Mining* ou *Mineração de Dados*

EDM - *Educational Data Mining* ou *Mineração de Dados Educacional*

LAK - *Learning Analytics*

KDD - *Knowledge Discovery in Databases*

DW - *Data Warehouse*

ADW - *Active Data Warehouse* ou *Data Warehouse Ativo*

BI - *Business intelligence*

OLAP - *Online Analytical Processing*

ETL - *Extract, Transform, Load*

AVA - *Ambiente Virtual de Aprendizagem*

LMS - *Learning Management System*

MOOC - *Massive Open Online Course*

AAR - *Academic Alert Report* ou *Relatório de Alerta Acadêmico*

OASE - *Online Academic Support Environment* ou *Ambiente online de suporte acadêmico*

OAAI - *Open Academic Analytics Initiative*

AVA - *Ambiente Virtual de Aprendizagem*

ATAM - *Architecture Tradeoff Analysis Method*

BRMS - *Business Rules Management System*

DRL - *Drools Rule Language*

Lista de Figuras

Lista de Tabelas

Lista de Códigos Fonte

Capítulo 1

Introdução

Neste capítulo, apresenta-se a introdução dessa tese, iniciando-se com uma contextualização e motivação, seguida da problemática e questões de pesquisa, norteadoras dos propósitos deste trabalho. Em seguida, descrevem-se os objetivos pretendidos e os aspectos metodológicos gerais da pesquisa que guiaram a realização desse trabalho. Finalmente, ressaltou-se as principais contribuições do mesmo, além de esboçar-se uma delimitação da presente pesquisa, finalizando-se com com uma descrição da organização deste documento.

1.1 Contextualização e Motivação

É cada vez mais comum o uso de plataformas virtuais para apoiar a educação *on-line* (ZORRILLA, 2009), aqui, a educação *on-line* é entendida como uma forma de educação, seja na modalidade presencial ou a distância, que é entregue e administrada usando algum tipo de plataforma de aprendizado *on-line*[1], *Learning Management Systems* (LMS) ou Ambientes Virtuais de Aprendizagem (AVA). Essas plataformas, normalmente, oferecem uma grande variedades de canais e espaços para facilitar o compartilhamento de informações e a comunicação entre os seus participantes, possibilitando aos educadores compartilhar informações entre os estudantes, produzir conteúdo, preparar avaliações e testes, gerar e produzir discussões, gerenciar aulas a distância e possibilitar a aprendizagem colaborativa através de fóruns, *chats*, áreas de compartilhamento de arquivos, etc (ROMERO; VENTURA, 2007). Entre as principais plataformas, destacam-se: Blackboard, Moodle, Canvas,

[1]Plataformas de aprendizado *on-line* também são conhecidas como plataformas *e-learning*

Projeto Sakai, entre outros. O Moodle (MOODLE, 2017), por exemplo, é uma plataforma largamente adotada no Brasil. Segundo dados do Projeto Moodle [2], a plataforma vem sendo utilizada em mais de 200 países, com aproximadamente 10 milhões de cursos e 88 milhões de usuários.

A virtualização do ensino através dessas plataformas *e-Learning* possui diversas vantagens, dentre elas: popularização e disseminação da educação; acesso em qualquer hora ou lugar; uso de ferramentas colaborativas; suporte a diferentes estilos de aprendizagem; entre outras (ZORRILLA, 2009). Muitas iniciativas que implementam algum tipo de educação *on-line* tem surgido impulsionadas pela popularização do acesso à Internet, tais como Coursera[3], Udacity[4] e Khan Academy[5]. Particularmente, no Brasil, existe uma iniciativa bastante popular no contexto do governo federal, que envolve mais de 100 instituições de ensino, denominada Universidade Aberta do Brasil (UAB)[6]. A UAB oferece cursos e programas da educação superior através do ensino a distância.

Com a crescente disponibilidade de cursos apoiados por plataformas *e-Learning*, muitas dificuldades têm sido relatadas na literatura (ROMERO; VENTURA, 2007; ZORRILLA, 2009; LIÑÁN; PÉREZ, 2015). Em sua maioria, essas dificuldades estão relacionadas à complexidade em monitorar e avaliar os estudantes envolvidos nos cursos, bem como a falta e/ou dificuldade em acessar informações relevantes que apoiem o processo de tomada de decisão pedagógica. Entre as principais causas dessas dificuldades, pode-se destacar:

- **Ausência de contato direto com os estudantes** (ROMERO; VENTURA, 2007): o processo de tomada de decisão em salas de aula tradicionais envolve, dentre outros fatores, a observação do comportamento do estudante e acompanhamento da sua evolução histórica. Contudo, quando os estudantes estão em um ambiente eletrônico, este monitoramento informal não é possível.

- **Escassez de informações em ambientes *e-Learning*** (ZORRILLA, 2009; FALAKMASIR et al., 2010): Esses ambientes oferecerem aos professores o acesso a um conjunto específico de informações, tais como: resumo de acessos, datas da primeira e

[2]Dados retirados de https://moodle.net/stats/
[3]http://www.coursera.org/
[4]http://br.udacity.com/
[5]http://www.khanacademy.org/
[6]http://www.capes.gov.br/uab

última visita, o número de visitas em cada página e o tempo médio gasto por visita. Porém, há inúmeros relatos na literatura que apontam que tais informações não são suficientes para se ter uma boa análise do comportamento e da evolução de cada estudante.

- **Grande volume de dados gerados** (ZORRILLA, 2009): Além de poucas informações relevantes, dado o universo de estudantes, o volume de dados gerados é muito elevado, o que inviabiliza uma análise mais detalhada à procura, por exemplo, de correlações importantes entre dados que possam levar a achados mais específicos. Um exemplo disso, poderia ser a análise de variações dos valores no tempo. O que torna a análise dos dados ainda mais inviável é o fato deles serem provenientes de diferentes fontes de dados, tais como: arquivos de *logs*, interações em redes sociais, dados acadêmicos dos estudantes e dados socioeconômicos.

- **Heterogeneidade e quantidade de estudantes envolvidos nos cursos** (ZORRILLA, 2009): Quando o número de estudantes e a heterogeneidade das interações é alta, o instrutor tem sérias dificuldades para extrair e analisar informações úteis. No contexto da UAB, isso pode ser evidenciado através do Censo EAD BR (CENSO, 2015), divulgado anualmente pela Associação Brasileira de Educação a Distância (ABED). Em um comparativo entre os anos de 2012 e 2013, o censo apresentou um aumento de cerca de 52,2% no número de alunos EAD. Além disso, o censo apresentou também turmas da UAB com mais de 1000 alunos matriculados.

- **Desmotivação dos estudantes devido ao *feedback* tardio ou de baixa qualidade** (ZORRILLA, 2009): Quando o *feedback* é fornecido tardiamente ou de forma muito geral, os estudantes costumam apresentar descontentamento ou até mesmo dificuldades no aprendizado, podendo chegar à evasão do curso. Em plataformas de *e-Learning*, o tempo e qualidade do *feedback* são fortemente prejudicados pelo grande volume de dados e a consequente dificuldade de se identificar, de forma antecipada, estudantes que precisam de maior atenção por parte dos educadores.

Notadamente, muitas dessas questões dizem respeito ao pouco provimento de ferramentas de apoio adequadas aos tomadores de decisões pedagógicas, principalmente na tarefa

de obter informações na análise de dados das interações dos estudantes (ROMERO; VENTURA, 2017). Siemens et al. (2011) afirma que educadores não têm acesso a um conjunto de ferramentas integradas que permitam avaliações variadas e complexas do desempenho individual dos estudantes, a partir de uma análise comparativa entre diferentes grupos de estudantes.

1.2 Problemática Geral

A partir das questões mencionadas, técnicas de *Business Intelligence* (BI) aplicadas a bases de dados de plataformas *e-Learning* têm sido consideradas para auxiliar instrutores e outros *stakeholders* envolvidos gerando estatísticas, modelos de análises e a descoberta de padrões a partir do grande volume de dados armazenados (FALAKMASIR et al., 2010). Nesse contexto, áreas de pesquisas relacionadas a *Business Intelligence* (BI) para educação têm surgido com a proposta de entender e analisar o grande volume de dados gerado, provendo a professores e educadores informações relevantes para os auxiliar no processo de tomada de decisão pedagógica. Exemplos de áreas bem sucedidas nesse segmento são: Mineração de Dados Educacionais (EDM)[7] e *Learning Analytics* (LAK). Essas áreas têm enfrentado grandes desafios tanto na aplicação de suas técnicas quanto na construção de soluções específicas a fim de atender às demandas de professores e educadores. Diversos autores apontam desafios importantes nessas áreas (C.ROMERO; S.VENTURA, 2007; C.ROMERO; VENTURA, 2010; ROMERO; S.VENTURA, 2013; LIÑÁN; PÉREZ, 2015):

- **Dificuldade de se construir sistemas integrados**: Para haver a utilização efetiva de BI educacional, faz-se necessário integrar diferentes fontes de dados, tais como redes sociais, sistemas acadêmicos e plataformas de *e-Learning*. Porém, as ferramentas educacionais existentes não realizam tal integração e consolidação desses dados.

- **Dificuldade de se reutilizar soluções existentes**: As ferramentas atuais para apoio a BI educacional são normalmente construídas para apoiar necessidades específicas de um curso, não havendo soluções gerais que possam ser reutilizadas e integradas em diferentes ambientes educacionais.

[7] do inglês, *Educational Data Mining*

- **Integração com ambientes *e-Learning***: As ferramentas de BI educacional devem ser integradas a ambientes *e-Learning* como uma outra ferramenta de autoria. Assim, os *feedbacks* e resultados obtidos com a BI educacional podem ser aplicados diretamente no ambiente *e-Learning*.

- **Reuso de soluções existentes de descoberta de conhecimento em banco de dados**: Para reduzir o esforço de construção das ferramentas de BI educacional, são necessários projetos que priorizem o reuso de técnicas e algoritmos de descoberta de conhecimento em bancos de dados. Exemplos de tais técnicas são algoritmos de mineração de dados e ferramentas de *data warehouse*.

- **Automatização de ações pedagógicas**: Faltam ferramentas de BI educacional que permitam automatizações abrangentes e que possam ser integradas às plataformas de *e-Learning* mais populares a fim de possibilitar um acompanhamento personalizado em ambientes com altas taxas *estudante/instrutor*.

- **Personalização das ações pedagógicas**: Apesar da automação ser algo desejável na educação *on-line*, as ações pedagógicas do sistema devem ser flexíveis a ponto de serem definidas e configuradas pelos próprios educadores.

1.2.1 Questões de Pesquisa

Diante das questões supracitadas, é clara a necessidade de apoiar engenheiros de software durante o processo de desenvolvimento de soluções que atendam aos requisitos de professores e educadores envolvidos no processo de tomada de decisão em contextos educacionais. Embora as plataformas *e-Learning* disponibilizem informações, estas não são suficientes para a realização do processo de tomada de decisão (ZORRILLA, 2009). Além disso, com uma demanda crescente de alunos interagindo simultaneamente, instrutores têm tido ainda mais dificuldades em obter informações úteis para a tomada de decisão (ZORRILLA, 2009). Pode-se citar como exemplo do supracitado a UAB, em que um comparativo realizado entre os anos de 2012 e 2013, o Censo EAD BR apresentou um aumento de cerca de 52,2% no número de estudantes EAD. Este censo apresentou, ainda, turmas da UAB com mais de 1000 alunos matriculados. Como exemplo concreto, apresenta-se a Universidade Federal de Ala-

goas (UFAL) que chegou a oferecer cerca de 1000 vagas anuais em cursos de EAD através do Moodle, além de estender o acesso da plataforma educacional a todas as disciplinas de seus cursos presenciais. Dessa forma, apenas no contexto da UFAL, a solução proposta deve estar preparada para processar dados individuais de uma demanda de mais de 30.000 alunos (UFAL, 2017). Assim, atender à demanda crescente de alunos também se torna um desafio a ser alcançado.

Soluções de BI tradicionais e técnicas relacionadas têm sido aplicadas ao contexto educacional a fim de auxiliar os *stakeholders* no processo de tomada de decisão, integrando fontes heterogêneas, provendo uma visão ampla da plataforma educacional e gerando relatórios a partir de dados provenientes dos AVAs. Afinal, AVAs, por natureza, não se caracterizam como sistemas de apoio à decisão, portanto, eles não oferecem infraestrutura de BI necessária.

No entanto, a aplicação dessas soluções de BI tradicionais, tais como o *Pentaho* (PENTAHO, 2016), nitidamente não possuem as características necessárias que atendam aos requisitos do contexto educacional. Por exemplo, apesar do *Pentaho* oferecer ferramentas básicas de apoio ao BI, tais ferramentas necessitam de configurações manuais ao contexto educacional, que devem ser realizadas por um especialista em BI. Além disso, o *Pentaho*, a exemplo de outras ferramentas de BI, não oferece interface de integração com ambientes educacionais. Ou seja, os resultados obtidos a partir do processo de tomada de decisão precisam ser aplicados manualmente aos ambientes educacionais. Consequentemente, estas ferramentas não oferecem suporte à definição de ações pedagógicas automáticas, tais como, intervenções em ambientes educacionais.

A automação de ações pedagógicas se torna necessária devido à crescente demanda, volume de estudantes por turma em ambientes *on-line*, o que gera sérias dificuldades para se extrair e analisar informações úteis que possam ser analisadas individualmente pelo instrutor. Até mesmo soluções de BI projetadas especificamente para o contexto educacional atendem apenas parcialmente as necessidades e/ou desafios levantados. Desse modo, promover o desenvolvimento de soluções BI específicas que atendam a essas necessidades dos *stakeholders* em sua totalidade é fundamental ao contexto educacional, permitindo aos envolvidos o acesso a um completo conjunto de informações, proporcionando uma visão ampla do processo de aprendizagem dos estudantes e a realização de um acompanhamento indivi-

dualizado destes, entre outros benefícios.

A preocupação com requisitos de qualidade, tais como escalabilidade[8] e facilidade de integração com sistemas heterogêneos (BASS; CLEMENTS; KAZMAN, 2003a; BRITO et al., 2009), torna o desenvolvimento de sistemas educacionais uma tarefa complexa e que demanda a utilização de técnicas especializadas da engenharia de software. Nesse sentido, no desenvolvimento de sistemas complexos se faz necessário um investimento especial no projeto arquitetural do software (FRANCE; RUMPE, 2007). A arquitetura de software representa explicitamente a estrutura do sistema e é um dos primeiros artefatos que permitem a análise de atributos de qualidade do software (SHAW; GARLAN, 1996). Além disso, o desenvolvimento centrado na arquitetura de software promove o reuso em larga escala através do desenvolvimento modular do software ou de tecnologias conhecidas como arcabouços de software[9] (BEYER et al., 2008). Nesse sentido, a utilização de uma arquitetura de referência é fortemente encorajada, uma vez que tende a aumentar a qualidade geral do software através do reuso de decisões de projeto importantes para um domínio em particular (CLEMENTS et al., 2003).

Com base na problemática discutida na Seção 1.1, foram estabelecidas a seguir as questões de pesquisa que norteiam os objetivos deste trabalho:

- **QP 1.** Como oferecer a desenvolvedores de software apoio para o desenvolvimento de plataformas de *Business Intelligence* educacionais integradas a ambientes de *e-Learning*, com alta escalabilidade e que contemple automação de ações pedagógicas?

- **QP 2.** Qual a redução no custo de desenvolvimento, com base na porcentagem de reuso de software, conseguido com a utilização do *framework* proposto para plataformas de *Business Intelligence* educacionais integradas a ambientes de *e-learning*?

- **QP 3.** Como facilitar a adição e evolução de novas soluções de *Business Intelligence* educacionais de maneira simplificada, para descoberta de informações no contexto educacional?

[8] Capacidade do software evoluir para atender uma demanda crescente de acessos, requisições e volume de processamento simultâneo

[9] Do Inglês *software frameworks*

1.3 Objetivos

Considerando a problemática apresentada, o objetivo geral deste trabalho é oferecer aos engenheiros de softwares educacionais a infraestrutura necessária para a construção de soluções de *Business Intelligence* educacional. Pretende-se que as soluções construídas a partir da infraestrutura proposta possa auxiliar professores e educadores no processo de tomada de decisão educacional em ambientes educacionais utilizados no apoio à educação *on-line*.

1.3.1 Objetivos Específicos

Para atender ao objetivo geral apresentado, os seguintes objetivos específicos foram estabelecidos:

I **Propor uma arquitetura de referência para apoiar BI Educacional** - promover a produção de soluções de BI educacional que ofereçam a infraestrutura necessária para apoiar professores e educadores na realização do processo de tomada de decisão em ambientes educacionais, contemplando as características levantadas anteriormente. As principais características são: integração de dados provenientes de fontes heterogêneas, reúso de software para apoiar a execução de processos que envolvam a tomada de decisões pedagógicas e a personalização e automatização de ações pedagógicas.

II **Oferecer um arcabouço para realização da arquitetura em ambientes educacionais** - oferecer uma implementação para a arquitetura de referência proposta, através de um arcabouço de software, para promover o reuso de software e facilitar a integração entre bibliotecas existentes de mineração de dados e ambientes *e-Learning*, possibilitando a adição e evolução de novas soluções de maneira simplificada;

III **Atender a uma demanda crescente de processamento** - o arcabouço de software proposto deve ser escalável para atender a uma demanda crescente de processamento, tanto em termos de número de alunos por turma, quanto em termos de quantidade de turmas atendidas, preservando o desempenho do sistema;

IV **Possibilitar a automatização do processo de tomada de decisão** - Viabilizar a automatização de ações personalizadas pelo próprio educador, possibilitando um suporte proativo e individualizado aos estudantes;

1.4 Aspectos Metodológicos

Além do investimento das etapas iniciais que culminaram com a definição do tema, da problemática e da análise dos trabalhos relacionados, a metodologia utilizada para realização do objetivo desta tese pode ser resumida a seguir.

O projeto arquitetural envolveu decisões de projeto influenciadas por questões tanto objetivas, tais como métricas de atendimento aos requisitos de qualidade, quanto subjetivas, tais como a experiência do arquiteto. Por essa razão, o presente trabalho adotou uma abordagem quanti-qualitativa, que envolveu em sua condução as seguintes etapas: i) Revisão de literatura, a fim de evidenciar o problema e identificar as reais necessidades de professores/educadores e outros *stakeholders* envolvidos no processo de tomada de decisão em contextos educacionais, além de conhecer as características das soluções disponíveis; ii) Refinamento de um processo para definição de arquiteturas de referência, a partir da definição de um método quantitativo de refinamento arquitetural, que envolve monitoração de arquiteturas candidatas; iii) Elaboração do projeto arquitetural inicial da arquitetura de referência; iv) Instanciação e implementação da arquitetura concreta; e v) Avaliação e refinamento do projeto arquitetural inicial.

1.5 Contribuições Gerais e Delimitação da Pesquisa

Com base nos objetivos definidos, uma das contribuições principais deste trabalho para educação *online*, focada em engenharia de software para educação, consiste em uma arquitetura de referência escalável que serve como ponto de partida para a instanciação de arquiteturas concretas e suas implementações através de um arcabouço de software, com foco no desenvolvedor, que promove o reuso de software.

Entende-se que a infraestrutura necessária envolve funcionalidades providas por um conjunto de ferramentas integradas que ofereçam flexibilidade, escalabilidade e desempenho necessários a fim de favorecer que as soluções instanciadas a partir da arquitetura proposta possam ser utilizadas em situações reais dentro do contexto educacional.

Em síntese, as contribuições gerais desta tese, incluem: arquitetura de referência para a tomada de decisão pedagógica, implementação da arquitetura de referência, modelos de

regras avançadas, SEDAM-R (*framework* para mineração de dados), diferentes perspectivas arquiteturais e um método de refinamento para projetos de arquiteturas de referência.

Diante das especificidades e complexidade envolvidas no processo de descoberta de conhecimento, este trabalho não abordará as seguintes questões: i) seleção de algoritmos de mineração; ii) parametrização automática de algoritmos de mineração, sugerindo a utilização de valores padrões e/ou configurados *a priori* por um usuário administrador, com conhecimento técnico no domínio de mineração de dados; iii) avaliação da qualidade e da efetividade dos modelos de mineração construídos a partir da solução proposta. Entende-se que tais questões são demasiadamente complexas para serem tratadas e incluídas no escopo deste trabalho.

1.6 Organização do Trabalho

O restante desta tese está estruturado como segue. No Capítulo 2 é apresentado o referencial teórico utilizado como base para o desenvolvimento deste trabalho. No Capítulo 3 é apresentada a revisão de literatura, que destaca os principais trabalhos relacionados e os principais diferenciais da arquitetura de referência proposta diante do estado da arte. No Capítulo 4 é apresentado o refinamento realizado no processo adotado para o projeto e refinamento da arquitetura de referência proposta. No Capítulo 5 é apresentada a arquitetura de referência proposta em detalhes. No Capítulo 6 é detalhado o processo de instanciação da arquitetura, bem como os detalhes técnicos e decisões de projeto relativas à sua implementação, enfatizando os componentes da arquitetura. A avaliação e o processo de refinamento da arquitetura é detalhado no Capítulo 7. Por fim, no Capítulo 8 é apresentada a conclusão e direcionamentos para trabalhos futuros.

Capítulo 2

Referencial Teórico

Neste capítulo são apresentados, sucintamente, os principais tópicos que formam a base teórica e contribuem para o entendimento do trabalho proposto. São apresentados os conceitos relacionados à Arquitetura de Referência, Método ATAM, Mineração de Dados Educacionais, *Data warehouses* e *Data warehouses* ativos.

2.1 Arquitetura de Referência

Arquiteturas de referência são um tipo especial de arquiteturas de software que capturam a essência de um conjunto de sistemas de determinado domínio (NAKAGAWA; OQUENDO; MALDONADO, 2014). Segundo Nakagawa, Oquendo e Maldonado (2014), esta área vem ganhado evidência e tem como principal propósito prover um guia para o desenvolvimento, padronização e evolução de arquiteturas de sistemas.

A fim de estabelecer uma melhor compreensão a respeito do termo ”Arquitetura de Referência”, destacam-se duas outras definições bastante utilizadas na literatura. A primeira delas é apresentada por Angelov, Grefen e Greefhorst (2009), que estabelecem que uma arquitetura de software de referência é uma arquitetura genérica para uma classe de sistemas de informação que é utilizada como fundação para o projeto de arquiteturas concretas dessa classe. A segunda definição relevante é apresentada por Muller (2008 apud FEITOSA, 2013), que afirma que arquiteturas de referência podem ser utilizadas como guia para projetos de arquiteturas concretas ou como ferramenta de padronização, provendo interoperabilidade entre sistemas ou componentes de sistemas. Deste modo, uma arquitetura de referência é projetada

para abordar as funcionalidades e qualidades desejadas por todas as partes interessadas em seus contextos específicos (ANGELOV; TRIENEKENS; GREFEN, 2008), como destacado na Figura 2.1.

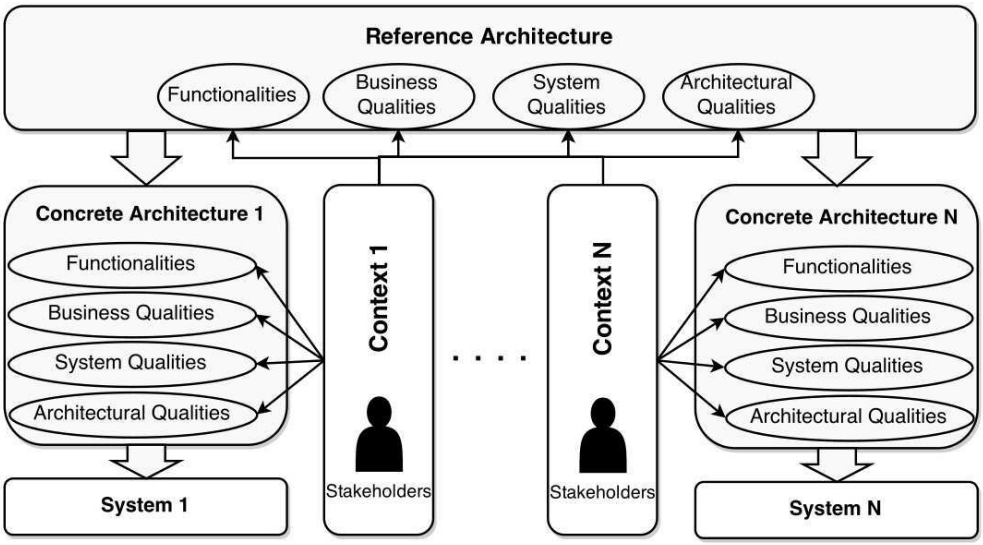


Figura 2.1: O papel dos *stakeholders* e os contextos para arquiteturas de referência e concretas

Fonte: Extraída de (ANGELOV; TRIENEKENS; GREFEN, 2008)

Modelo de Referência *versus* Arquitetura de Referência

Segundo Bass, Clements e Kazman (2003b), um modelo de referência é a divisão de funcionalidades juntamente ao fluxo de dados de como essas partes se relacionam. Esse modelo de referência envolve um padrão de decomposição de um problema em partes que cooperativamente resolvem esse problema. Um exemplo de modelo de referência é o OASIS (MACKENZIE et al., 2006). Já uma arquitetura de referência é um modelo de referência mapeado em elementos de software que cooperativamente implementam funcionalidades definidas nesses modelos (BASS; CLEMENTS; KAZMAN, 2003b).

Como representada na Figura 2.2, uma arquitetura de referência é criada com base em um ou mais modelos de referência existentes em conjunto com outros elementos, tais como conhecimento especializado de domínio, estilos arquiteturais e elementos de software (NAKAGAWA; OQUENDO; MALDONADO, 2014). Os autores destacam, ainda, que utilizando essa arquitetura como base, arquiteturas concretas podem ser instanciadas. Um exemplo

de arquitetura de referência é a AUTOSAR (LEITNER, 2007), (NAKAGAWA et al., 2011) voltada ao domínio automotivo.

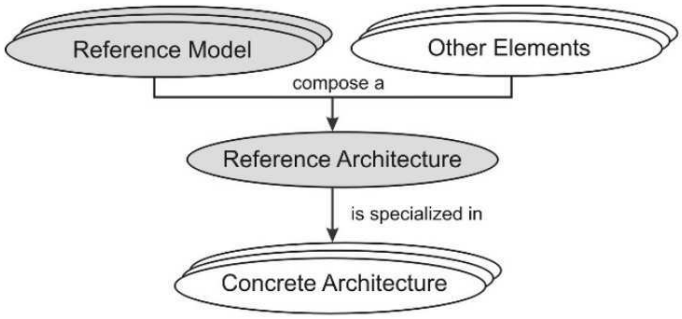


Figura 2.2: Relação entre modelo de referência, arquitetura de referência e arquitetura concreta

Fonte: Extraída de (NAKAGAWA; OQUENDO; MALDONADO, 2014)

Entre os processos de construção de arquiteturas de referência destaca-se o processo ProSA-RA, proposto por Nakagawa et al. (2009). O ProSA-RA sistematiza o *design*, a representação e a avaliação de arquiteturas de referência (NAKAGAWA et al., 2014). Esse processo foi utilizado como base para o estabelecimento da arquitetura de referência proposta neste trabalho. Porém, uma extensão foi realizada no tocante ao refinamento da arquitetura inicialmente projetada. O processo, incluindo o refinamento proposto, é apresentado no Capítulo 4.

2.2 O Método ATAM para Avaliação de Arquiteturas de Software

O ATAM[1] (KAZMAN; KLEIN; CLEMENTS, 2000) é um método de análise de arquiteturas, que foca na identificação e priorização dos requisitos não-funcionais do negócio relacionados aos atributos de qualidade desejados. Realizado a partir da definição dos requisitos não funcionais, esse método analisa diversos estilos arquiteturais a fim de atender os atributos de qualidade definidos. O método ATAM é apresentado na Figura 2.3 e suas nove atividades são detalhadas a seguir (SEI, 2017):

[1] do inglês *Architecture Tradeoff Analysis Method*

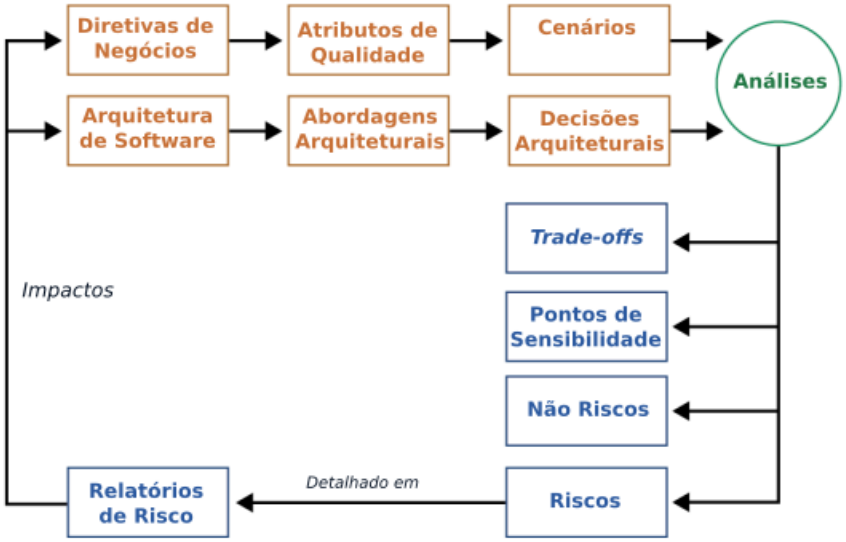


Figura 2.3: Etapas do ATAM
Fonte: Adaptada de (SEI, 2017)

1. **Apresentar o ATAM** - Nessa atividade, o método deve ser descrito para as partes interessadas, que tipicamente constituem um grupo formado pelo representante do cliente, o arquiteto de software, o testador, o gerente de projeto e o responsável pela fase de manutenção.

2. **Apresentar as diretivas do negócio** - Nesta etapa o gerente de projeto deve apresentar os objetivos pretendidos para o sistema. Com base nessa descrição, deve-se escolher a arquitetura que será proposta inicialmente.

3. **Apresentar a arquitetura** - Com base na descrição realizada pelo gerente, a arquitetura que será inicialmente proposta deve ser descrita pelo arquiteto, enfatizando como os objetivos apresentados serão materializados.

4. **Identificar as possibilidades para a arquitetura** - O arquiteto deve identificar possíveis arquiteturas alternativas. Essa lista pode ser criada, por exemplo, com base nas arquiteturas semelhantes à arquitetura original.

5. **Gerar uma árvore de utilidade dos atributos de qualidade** - Os atributos de qualidade descritos são detalhados em requisitos não-funcionais. Na etapa seguinte, são definidos cenários para auxiliar a compreensão do que se espera desses requisitos. Por fim, esses requisitos são priorizados.

6. **Analisar as possíveis arquiteturas** - Após a definição das prioridades, uma lista de arquiteturas definidas será analisada. Por exemplo, comparar uma arquitetura que tem como prioridade a escalabilidade com outra que tem o desempenho como prioridade.

7. **Priorização dos cenários** - Com base nos cenários especificados na árvore de utilidade, os cenários devem ser refinados a partir das ideias dos atores envolvidos. Após a identificação dos cenários possíveis, são escolhidos cenários por meio de votação entre os envolvidos.

8. **Analisar as possíveis arquiteturas** Esta etapa envolve uma nova iteração da etapa 6, no entanto, agora devem ser gerados casos de testes a partir dos cenários priorizados anteriormente a fim de analisar a arquitetura. Após a execução desses casos de testes, podem ser identificados, por exemplo, riscos e *Tradeoffs* entre as arquiteturas analisadas.

9. **Apresentar resultados** Após a execução de todas as atividades, as informações mais importantes são sumarizadas e a partir dessas informações, é gerado um relatório com as principais estratégias para a arquitetura.

2.3 Mineração de Dados Educacionais

Mineração de Dados Educacionais ou *Educational Data Mining* (EDM) é uma disciplina baseada nos fundamentos da Mineração de Dados para explorar dados provenientes de ambientes educacionais com a finalidade de descobrir padrões descritivos e realizar predições que caracterizem o comportamento dos estudantes e suas realizações (PEÑA-AYALA, 2013).

Métodos em Mineração de Dados Educacionais

Entre os métodos adotados na EDM, destacam-se (ROMERO; S.VENTURA, 2013; C.ROMERO; VENTURA, 2010; BAKER et al., 2010; C.ROMERO; S.VENTURA, 2007; BAKER; INVENTADO, 2014): predição, agrupamento, *outlier detection*, mineração de texto, *distillation of data for human judgment*, descoberta com modelos,*process mining*, mineração de relações, entre outros. Os principais métodos são detalhados a seguir:

Distillation of data for human judgment - O objetivo desse método é representar os dados de maneiras inteligíveis, utilizando o resumo, visualização e interfaces interativas para destacar informações úteis e apoiar a tomada de decisões (ROMERO; S.VENTURA, 2013). De acordo com Romero e S.Ventura (2013), esse método tem sido utilizado, principalmente, para auxiliar professores e educadores no processo de visualização e acompanhamento de estudantes nos cursos.

Descoberta com Modelos - Na descoberta com modelos, o modelo de um fenômeno é desenvolvido por meio de *predição*, *agrupamento*, ou *engenharia de conhecimento*, em alguns casos (dentro da engenharia do conhecimento, o modelo é desenvolvido utilizando o raciocínio humano ao invés de métodos automatizados) (BAKER et al., 2010). Assim, segundo Baker et al. (2010), esses modelos podem então ser aplicados na realização de outras análises, tais como: predição, mineração de relações, entre outras aplicações.

Aplicação da Mineração de Dados Educacionais

No ciclo de aplicação da mineração de dados em contextos educacionais, apresentado na Figura 2.4, professores, educadores responsáveis projetam, planejam, constroem e mantêm sistemas educacionais enquanto estudantes acessam e interagem com esses sistemas, de modo que uma grande quantidade de dados é gerada e armazenada. Assim, diversas técnicas de mineração de dados podem ser utilizadas para descobrir informações úteis e aprimorar o processo de ensino-aprendizagem.

A EDM tem sido utilizada em: recomendação de conteúdo (NAGATA et al., 2009) (ZAIANE, 2002) (TANG; MCCALLA, 2003), recomendação de cursos (SACÍN et al., 2009) (RANKA; ANWAR; CHAE, 2010), análise de sentimento (KIM; CALVO, 2010), aperfeiçoamento de ambientes *e-Learning* (MACFADYEN; SORENSON, 2010) (ZAANE; ZAïANE, 2001) (TANG et al., 2000), desenvolvimento de modelos do estudante (KHODEIR et al., 2010) (TANG; MCCALLA, 2002), entre outras aplicações.

2.4 *Data Warehouse*

Um *data warehouse* é uma coleção de dados orientado a assunto, integrado, não-volátil e variante no tempo. Usado para o suporte ao gerenciamento de decisões, onde (INMON,

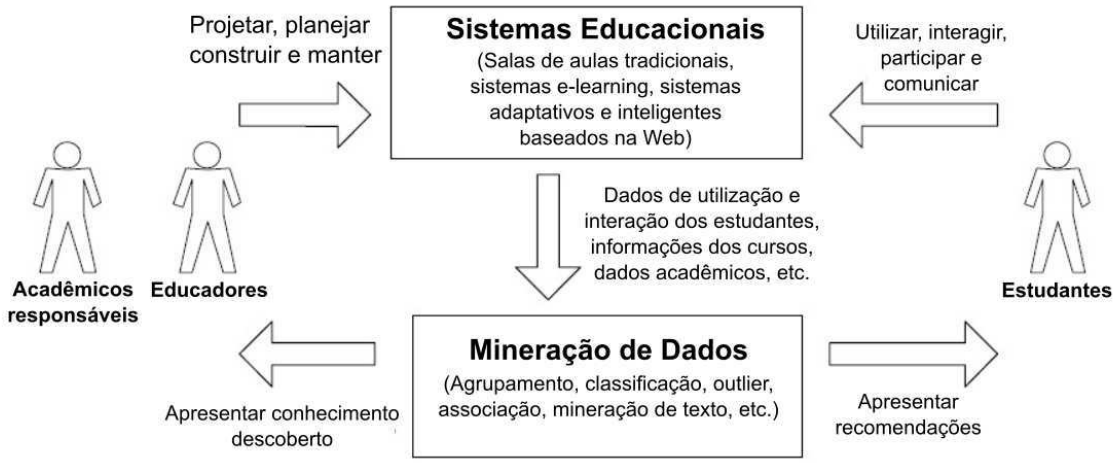


Figura 2.4: Ciclo de aplicação da mineração de dados educacional
Fonte: Adaptada de (ROMERO; VENTURA, 2007)

2002):

- **Orientado a assunto:** O *data warehouse* é orientado de acordo com as principais áreas temáticas da corporação, que foram definidos no modelo de dados corporativos de alto nível (INMON, 2002). Desse modo, cada empresa possui seu próprio conjunto de assuntos. Em uma companhia de seguros, por exemplo, as aplicações podem ser: saúde, automóveis, entre outros;

- **Integrado:** Várias fontes de dados diferentes alimentam um *data warehouse*. Para tal, os dados precisam ser convertidos, reformatados, resumidos, e assim por diante. O resultado é uma visão corporativa única dos dados (INMON, 2002);

- **Não-volátil:** Quando os dados são carregados no *data warehouse*, eles são carregados em um formato estático. Quando ocorrem alterações, um novo registro do momento é escrito. Assim, é mantido um histórico dos dados no *data warehouse* (INMON, 2002);

- **Variante no tempo:** A variação do tempo implica que toda unidade de dados no *data warehouse* é precisa a partir de algum momento no tempo (INMON, 2002);

Elmasri e Navathe (2015) destacam que *data warehouses* proporcionam acesso aos dados para análise complexa, descoberta de conhecimento, tomada de decisão e o suporte às demandas de alto desempenho por dados e informações de uma organização. O exemplo de

uma arquitetura típica de um *data warehouse* foi proposta por Kimball e Ross (2013) e é apresentada na Figura 2.5:

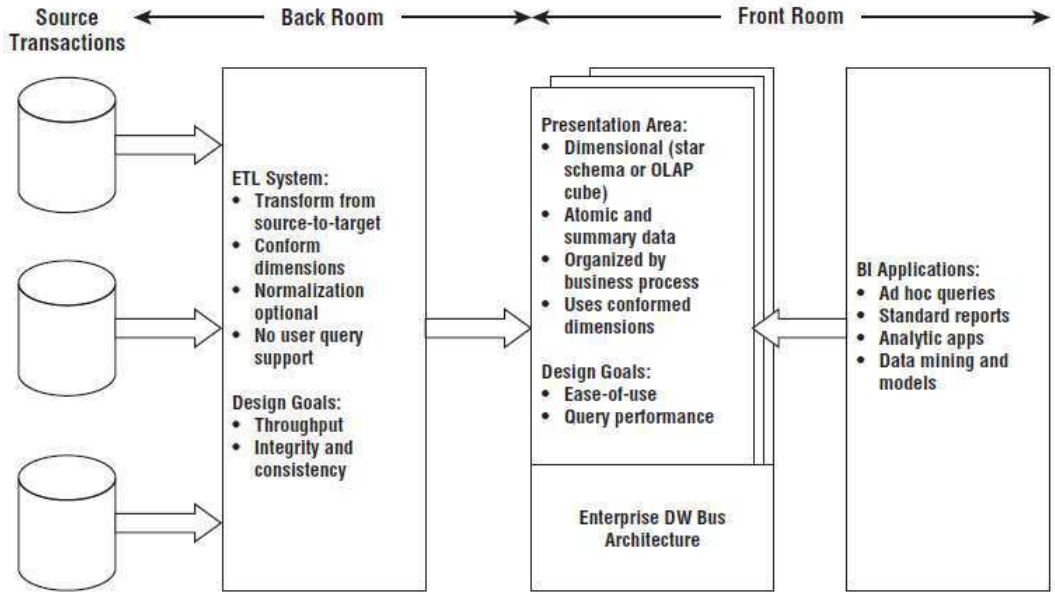


Figura 2.5: Arquitetura de um *data warehouse*
Fonte: Extraída de (KIMBALL; ROSS, 2013)

A arquitetura proposta por Kimball e Ross (2013) é composta por três componentes básicos: i) Processo de ETL; ii) Área de apresentação e suporte ao *Business Intelligence* (BI); e iii) Aplicações de *Business Intelligence*.

Processo de ETL - Kimball e Ross (2013) descrevem o processo de ETL como tudo que acontece entre as fontes de dados e a área de apresentação do DW/BI. O processo de ETL compreende as seguintes etapas (KIMBALL; ROSS, 2013):

- *Extract* - Ler e entender as diversas fontes de dados e carregar os dados necessários para o sistema de ETL para futuras manipulações;
- *Transformation* - Depois da extração, são realizadas as transformações, que buscam realizar a limpeza dos dados, como: corrigir erros, tratar elementos em falta, transformar os dados para um formato padrão;
- *Load System* - A etapa final é a estruturação física e carregamento dos dados em modelos dimensionais para a área de apresentação;

Presentation Area to Support Business Intelligence - A área de apresentação é onde os dados estão organizados, armazenados e disponíveis para a realização de consultas pelos usuários, geração de relatórios e outras aplicações analíticas de BI (KIMBALL; ROSS, 2013). Assim, segundo Kimball e Ross (2013), os dados podem ser apresentados, armazenados e acessados em esquemas dimensionais, sendo *star schemas* ou cubos *OLAP (On-Line Analytical Processing)*.

Business Intelligence Applications - Por fim, o componente *Business Intelligence Applications* é descrito por Kimball e Ross (2013) como uma aplicação de BI que pode ser uma simples ferramenta de consulta ou uma sofisticada aplicação de mineração de dados ou, ainda, uma aplicação de modelagem. Kimball e Ross (2013) destacam que ferramentas de consultas podem ser compreendidas e utilizadas de forma eficaz por apenas uma pequena percentagem de usuários de negócios. Asseveram, também, que a maioria dos usuários de negócios provavelmente acessará os dados por meio de aplicações dirigidas por parâmetros predefinidos e modelos que não requerem que os usuários construam consultas diretamente.

2.5 *Data Warehouse* Ativo

Data warehouses convencionais são passivos, seus usuários precisam realizar todo o processo de análise e tomada de decisão de modo manual (THALHAMMER; SCHREFL; MOHANIA, 2001). Muitas dessas decisões, no entanto, são recorrentes e bem definidas. Exemplos de decisões dessa natureza foram apresentados por Thalhammer, Schrefl e Mohania (2001): mudança de preços de produtos, retirada de produtos, conceder descontos especiais a clientes, entre outros. Por exemplo, sempre que ocorrem mudanças de preços em quaisquer produtos, os analistas precisam apresentar um novo relatório de vendas 7 dias após essas alterações. Assim, sempre que ocorrem mudanças de preços, o analista responsável precisa acessar e gerar esses novos relatórios. Com base nessas necessidades, surgiu o *active data warehouse* (SCHREFL; THALHAMMER, 2000), que tem como principal característica possibilitar a automatização de ações rotineiras, oferecendo assim uma capacidade reativa aos *data warehouses*.

Para automatizar tais ações, os *data warehouses* ativos se baseiam na ideia de regras provenientes dos bancos de dados ativos, as *ECA (EVENT-CONDITION-ACTION) rules*.

Segundo Thalhammer, Schrefl e Mohania (2001), *ECA rules* baseiam-se na ideia básica de que, quando um acontecimento especificado ocorrer (*o evento*), uma certa expressão booleana avalia se é verdadeiro (*a condição*), e em seguida, uma *ação* apropriada será realizada automaticamente, sem interação explícita do usuário ou requisição da aplicação. Assim, um sistema de banco de dados que suporta *ECA rules* é referido como um sistema de banco de dados ativo. Para os *data warehouses* ativos, as *ECA rules* foram estendidas e são chamadas de *analysis rules* ou regras de análise (SCHREFL; THALHAMMER, 2000).

Segundo Thalhammer, Schrefl e Mohania (2001), as principais diferenças entre as *ECA rules* e as *analysis rules* está na função da condição das regras. Enquanto as *ECA rules* possuem condicionais simples com expressões booleanas, as *analysis rules* utilizam um conjunto de condições para realizar análises multidimensionais que anteriormente eram realizadas manualmente por analistas.

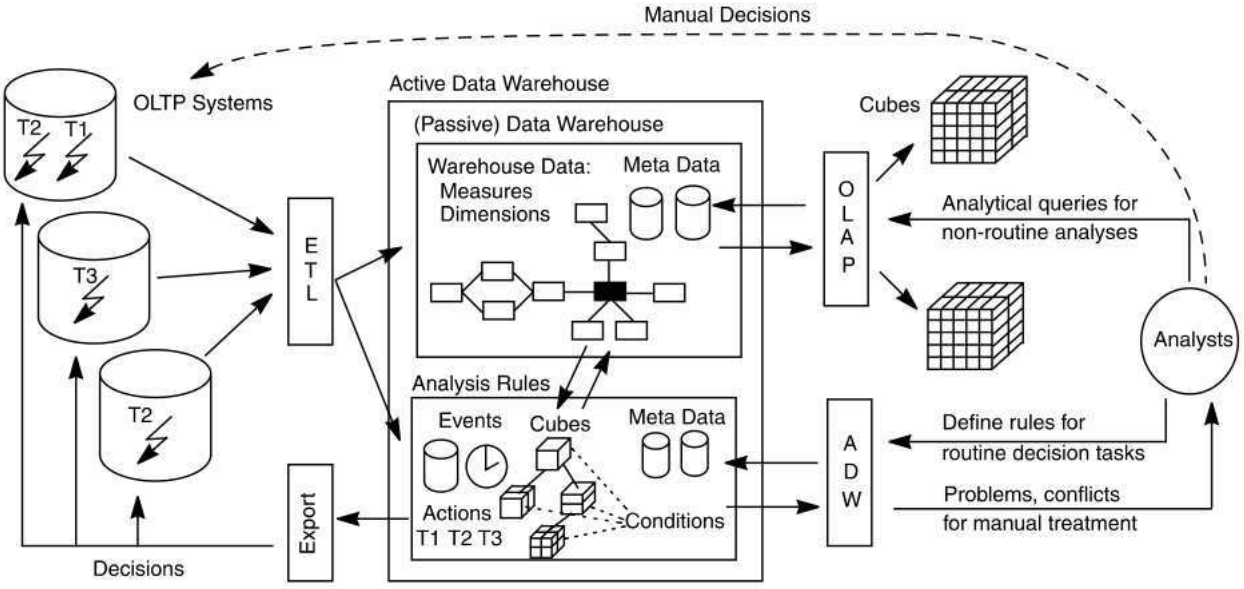


Figura 2.6: Arquitetura de um *data warehouse* ativo
Fonte: Extraída de (THALHAMMER; SCHREFL; MOHANIA, 2001)

Na arquitetura conceitual proposta por Thalhammer, Schrefl e Mohania (2001), apresentada na Figura 2.6, o *data warehouse* tradicional é utilizado para realização de análises OLAP (*On-Line Analytical Processing*) e para criar e manter as *analysis rules*. A arquitetura possui também um gerenciador que detecta e processa dados sobre os eventos, cubos e ações. Além disso, são geradas e exportadas decisões através de um mecanismo que invoca as transações diretamente nos sistemas OLTPs (*online transaction processing*). Por fim, ana-

listas podem acessar diferentes ferramentas do *data warehouse ativo* para definir e modificar *analysis rules*, eventos e cubos.

Entre as diversas abordagens relacionadas ao processamento ativo dos dados de um *data warehouse*, destacam-se duas categorias: i) Estão interessados no processamento de dados em tempo real; ii) Concentram-se na análise do processo de armazenamento de dados Hajlaoui e Hamdani (2014). Entre os exemplos para a primeira categoria, os autores apontam o trabalho proposto por Tho e Tjoa (2003) que definem um *framework* para *data warehousing* de latência zero, combinando a integração de dados contínua e regras ativas baseadas em *ECA rules* para minimizar o tempo de integração para o *data warehouse*. Já entre os exemplos da segunda categoria, os autores destacam o trabalho proposto Vasilecas e Smaizys (2006) onde cubos OLAP são gerados dinamicamente, selecionando novas dimensões e realizando detalhamentos de acordo com a situação de negócio avaliadas na análise dos dados de negócios atuais.

Capítulo 3

Trabalhos Relacionados

Neste capítulo são apresentados os trabalhos relacionados, frutos de uma revisão de literatura, cujo protocolo é apresentado no Apêndice A. São apresentados os detalhes das soluções existentes, bem como suas limitações enquanto soluções candidatas para atender as necessidades e desafios apresentados na introdução deste trabalho (Capítulo 1). Ao final, Seção 3.6, é apresentado um resumo das características consideradas essenciais a uma solução candidata ideal, a fim de responder de forma satisfatória às questões e desafios identificados. Esse resumo é apresentado na forma de uma tabela (Tabela 3.1), que também ressalta a relação entre os trabalhos relacionados e a solução proposta neste trabalho.

3.1 *A Decision Support System to Improve e-Learning Environments*

Zorrilla, García e Álvarez (2010) propõem um sistema de apoio à decisão para aperfeiçoamento de ambientes *e-Learning*. O sistema foi concebido com base em uma arquitetura modular e tem como objetivo ser genérico para que possa ser utilizado em diferentes plataformas *e-Learning*. Este sistema que estende uma arquitetura de *data warehouse* (ZORRILLA, 2009), proposta anteriormente por um dos autores, para gerar e armazenar modelos de mineração. Com isso, Zorrilla, García e Álvarez (2010) seu objetivo é escolher variáveis, especificar as tarefas de pré-processamento que necessitam, armazená-las adequadamente no *data warehouse*, determinar o algoritmo a ser usado de tal forma que o instrutor precise

apenas interpretar os resultados e tomar as ações educacionais que ele considere necessária, armazenando os modelos obtidos com a mineração a fim de calcular, incrementar e refinar padrões posteriormente.

A arquitetura desse sistema é apresentada na Figura 3.1.

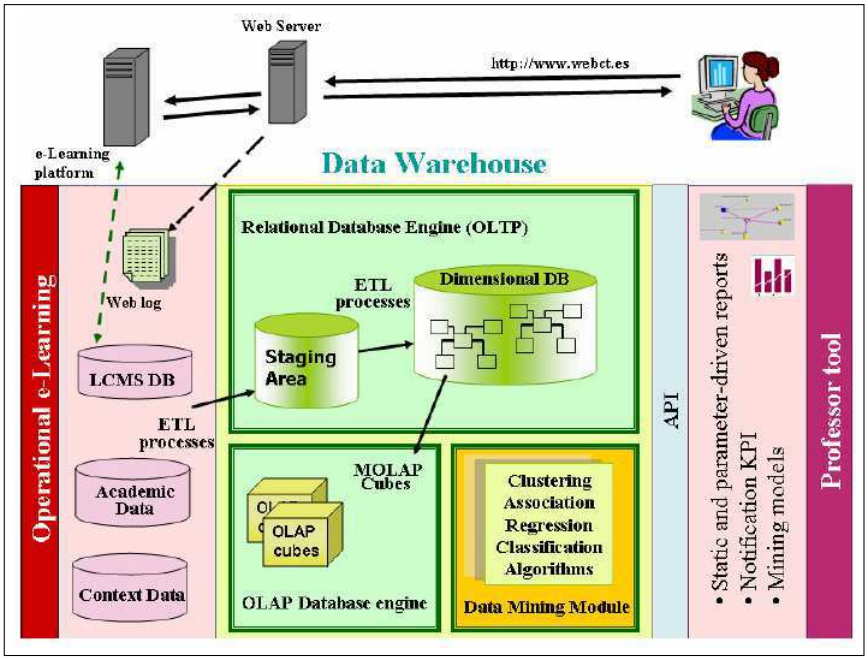


Figura 3.1: Arquitetura do sistema proposto
Fonte: Extraída de (ZORRILLA; GARCÍA; ÁLVAREZ, 2010)

A arquitetura do sistema proposto é composta de 3 módulos:

- Módulo para ler e reunir dados provenientes das plataformas *e-Learning*, para realizar as tarefas de pré-processamento relacionadas com a aplicação de algoritmos de mineração de dados e armazenamento desses em um *data warehouse* (Esse módulo reúne os processos de *ETL* e o *Data Staging Area*);
- Módulo que envolve os algoritmos de mineração de dados (Módulo de data mining);
- Interface amigável orientada à análise dos resultados.

A solução utiliza três aplicações de mineração: RapidMiner (MIERSWA et al., 2006), Weka (WITTEN et al., 2016) e Keel (ALCALA-FDEZ et al., 2009). Embora Zorrilla, García e Álvarez (2010) tenham utilizado as aplicações de mineração supracitadas, neste mesmo

trabalho eles descrevem suas pretensões para alterações na forma de utilização dessas tecnologias e seus respectivos módulos em uma versão futura de sua solução. A seguir são apresentadas estas alterações:

- A comunicação entre os módulos é feita por arquivos XML e, sempre que possível, é utilizado o padrão *Predictive Model Markup Language* (PMML). Além disso, a interface é projetada de acordo com padrões Web;

- Em seguida, Zorrilla, García e Álvarez (2010) projetam modelos que permitem visualizar as características dos estudantes e descrever seus padrões de aprendizagem. Após esse processo, é possível lidar com tarefas de classificação e predição, uma vez que determinaram quais são os parâmetros mais relevantes para essas tarefas;

- Após a realização desse processo, os padrões de comportamento descobertos são apresentados. Por exemplo, estudantes que não utilizam todos os recursos disponíveis em uma sessão. Segundo Zorrilla, García e Álvarez (2010), a partir dos padrões descobertos, instrutores podem detectar, por exemplo, conteúdos de páginas mal construídas, entre outros.

Além disso, Zorrilla, García e Álvarez (2010) propõe mais uma alteração na arquitetura apresentada anteriormente (ZORRILLA, 2009), a fim de incluir um módulo de descoberta de conhecimento, que era uma limitação da solução apresentada. No entanto, o foco do seu trabalho está em escolher variáveis que permitam responder questões relacionadas ao comportamento do estudante dentro de um ambiente *e-Learning*, mas, apesar de ter dado um passo importante quanto a essa questão, a solução proposta não é inteiramente realizada.

Apesar das vantagens alcançadas no uso do *data warehouse*, a solução proposta não é integrada a um ambiente educacional. Isso significa que o professor, após a realização das análises, terá que realizar as ações identificadas durante o processo de tomada de decisão manualmente, sem a possibilidade de aplicá-las diretamente aos ambientes educacionais e tampouco automatizar tais ações. Tal questão implica, no longo prazo, em uma sobrecarga do professor, pois, em futuras análises, caso os padrões identificados ocorram novamente, ele precisará repetir as ações definidas.

3.2 *Early Alert of Academically at Risk Students: An Open Source Analytics Initiative*

Este trabalho é fruto da *Open Academic Analytics Initiative* ou OAAI, que é um programa de concessão colaborativa plurianual destinado a investigar questões relacionadas à soluções e tecnologias de *learning analytics* para todo ensino superior. Nele, Jayaprakash et al. (2014) apresenta o processo e os desafios de coletar, organizar e minerar dados dos estudantes para predizer o seu risco acadêmico; os relatórios de resultados na execução da predição dos modelos construídos; avaliação da portabilidade desses modelos por meio de programas piloto e instituições parceiras; e os resultados das intervenções dos estudantes que foram classificados em situação de risco acadêmico. Para isto, é proposto um modelo preditivo.

A análise preditiva tem como objetivo detectar, relativamente cedo no semestre, os estudantes de graduação que apresentam dificuldades acadêmicas dentro do curso, por meio dos dados proveniente das interações destes. Jayaprakash et al. (2014) definiu duas classes para classificar os estudantes: em boa situação ou em situação de risco acadêmico. Para realização da classificação foram utilizadas quatro bases de dados: i) dados demográficos e de aptidão dos estudantes; ii) notas e dados dos cursos; iii) dados de interação dos estudantes dentro do SAKAI[1]; iv) contribuições parciais a nota final do estudante, coletados pelo SAKAI.

Para criação dos modelos, foram utilizados o Pentaho, Weka (WITTEN et al., 2016) e o SPSS. A ferramenta de integração de dados do Pentaho foi utilizada para automatizar o fornecimento de dados de entrada que alimenta a etapa de modelagem preditiva. É apresentado na Figura 3.2 a arquitetura do modelo de predição proposto.

O Modelo de regressão criado com os dados provenientes do *Marist College* foi aplicado aos dados piloto para identificar os estudantes que estavam potencialmente em risco de não completarem os seus cursos. Depois que o processo de predição foi executado, um Relatório de Alerta Acadêmico ou AAR foi produzido para cada instituição parceira, listando os estudantes que foram identificados nos dados em situação de risco acadêmico. O AAR gerado é colocado em lugar seguro, onde os instrutores podem acessar seus AARs correspondentes, a partir de um site específico, e recuperar a identidade de cada curso/estudante armazenado usando uma chave de encriptação. Na Figura 3.3 é apresentado o *workflow* de geração e

[1]SAKAI é um ambiente de aprendizagem de código aberto

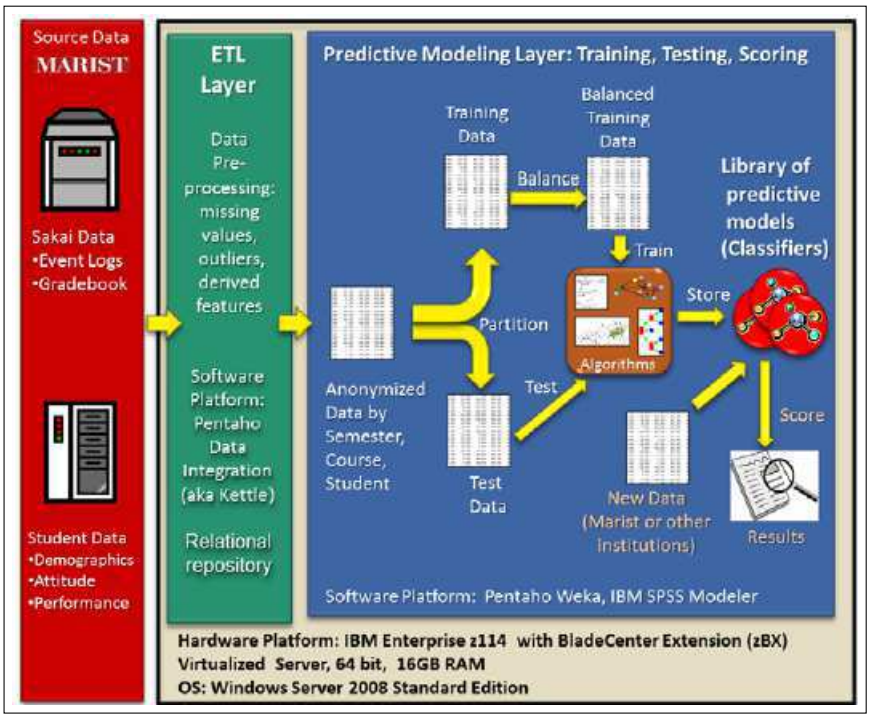


Figura 3.2: Arquitetura do modelo de predição
Fonte: Extraída de (JAYAPRAKASH et al., 2014)

distribuição dos AARs.

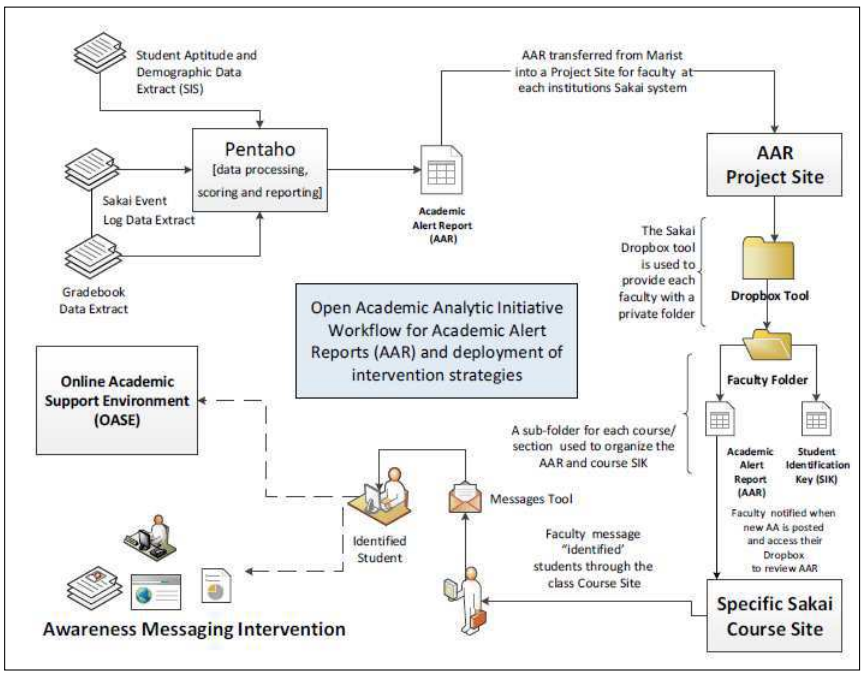


Figura 3.3: *Workflow* para geração e distribuição do Relatório de Alerta Acadêmico (AAR)
Fonte: Extraída de (JAYAPRAKASH et al., 2014)

Os estudantes que foram identificados em situação de risco acadêmico foram submeti-

dos a duas estratégias de intervenção diferentes: `Mensagem de sensibilização` e a participação em um `Ambiente online de suporte acadêmico` (OASE). Foram criados grupos de controle para avaliar o impacto de cada estratégia de intervenção. Estes grupos foram divididos em três seções. O mesmo instrutor cuidou das três seções, onde parte do grupo recebeu a intervenção por mensagem, outra parte a intervenção por OASE e outra parte do grupo que não recebeu intervenção alguma.

Após a apresentação detalhada dos resultados obtidos com a predição e a avaliação da eficácia das intervenções propostas, Jayaprakash et al. (2014) destaca suas contribuições. Dentre os resultados obtidos na pesquisa, os principais são destacados abaixo:

- A viabilidade de implementação de um protótipo de alerta precoce de código aberto para o ensino superior, e fornecimento um relato detalhado dos desafios e critérios de *design* utilizados na implementação de tal sistema;
- Estratégias relativamente simples, projetadas para alertar os estudantes de maneira antecipada no curso que eles possam estar em situação de risco acadêmico, podem gerar um impacto positivo sobre os resultados de aprendizagem desses alunos, tais como notas gerais do curso;
- Não exitem ganhos aparentes entre proporcionar aos estudantes um ambiente de apoio acadêmico e simplesmente deixar os alunos conscientes da sua situação; de risco acadêmico em potencial;

Jayaprakash et al. (2014) apresenta uma proposta bem construída e detalhada do processo de detecção e intervenção a estudantes classificados em situação de risco acadêmico. O processo envolve a construção do modelo, a classificação dos estudantes e a geração de relatórios de intervenção direcionados aos instrutores. Apesar disso, todo o processo é realizado de maneira manual, desde a construção do modelo até a intervenção junto aos estudantes. Algumas limitações dessa abordagem são: i) a necessidade da realização da classificação dos estudantes de maneira manual, mesmo após o modelo de classificação ter sido construído, ou seja, seria possível automatizar tal processo, não havendo a necessidade da intervenção de especialistas para a realização de novas classificações; ii) O processo de intervenção é inteiramente manual: o processo de intervenção manual tem um alto custo para ser executado inteiramente por instrutores e/ou professores. Um fato que torna esse processo ainda

mais custoso é o alto número de estudantes matriculados nos cursos apoiados por ambientes educacionais, principalmente nos MOOCs *(Massive Open Online Courses)*. Outro fato que deve ser levado em consideração é a necessidade de executar esse processo constantemente a fim de garantir um acompanhamento efetivo dos estudantes e detectar o mais cedo possível aqueles que se encontram em situação de risco acadêmico. Além disso, muitas decisões e ações identificadas a partir do processo de tomada de decisão podem ser recorrentes, de maneira que poderiam ser automatizadas.

3.3 Mineração de Dados Educacionais Para a Geração de Alertas em Ambientes Virtuais de Aprendizagem

Kampff (2009) propõe um sistema de alerta para o apoio ao acompanhamento e mediação docente. O sistema possui uma arquitetura híbrida baseada em dois tipos de alertas: alerta configurado pelo professor e alertas gerados pela mineração de dados educacionais. Os alertas são gerados por grupo de alunos para intervenção e apresentados ao professor.

A arquitetura do sistema é apresentada na Figura 3.4 e seus componentes são detalhados a seguir.

Interface do sistema de alerta

É o módulo em que o professor realiza a configuração dos alertas fixos, com base em indicadores, e onde ele recebe os alertas provenientes dos alertas fixos configurados e dos alertas obtidos por meio da mineração de dados educacionais.

Gerenciador de alertas fixos

Os alertas fixos são as questões que os professores desejam acompanhar de forma explicita, como: alerta para o fórum, por meio do qual o professor pode desejar ser notificado sobre o número de alunos com postagens inferior a um número estabelecido. É apresentado na Figura 3.5 a interface de configuração de alertas fixos.

No processo de configuração dos alertas fixos, o professor, por meio do módulo de configuração, pode definir o que deseja monitorar, o período de tempo que deseja ser notificado e quem será notificado, o professor e/ou os estudantes. Os alertas fixos são observados, pelo módulo de monitoramento, com base nas informações correntes dos estudantes por meio de consulta à base de dados do ambiente.

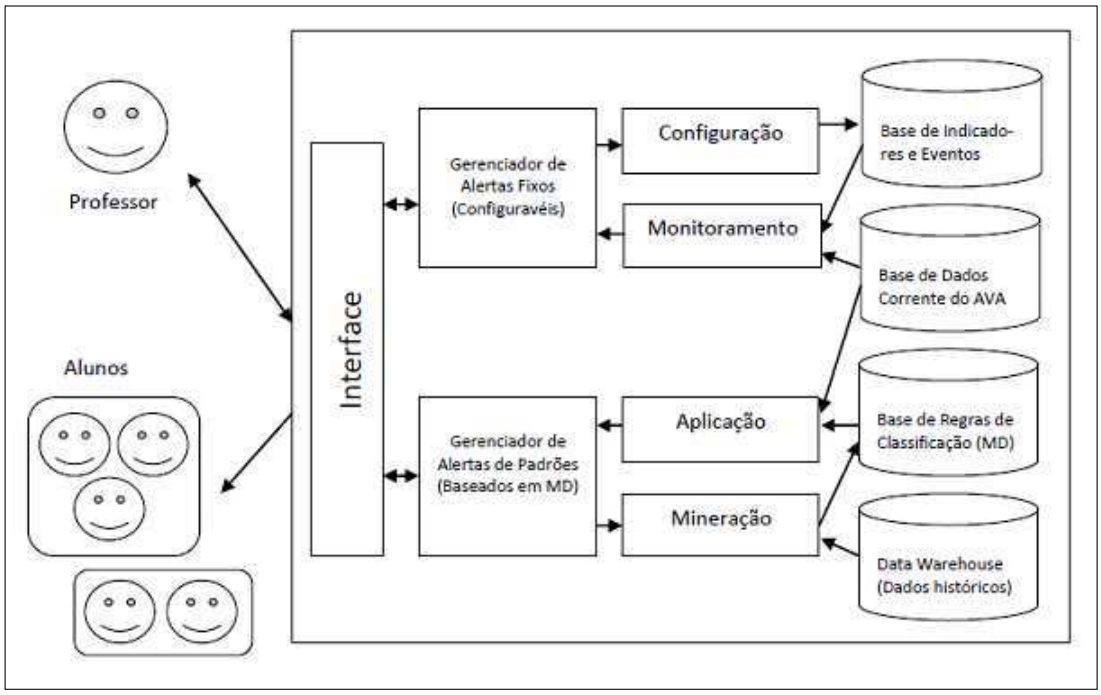


Figura 3.4: Arquitetura do sistema de alertas
Fonte: Extraída de (KAMPFF, 2009)

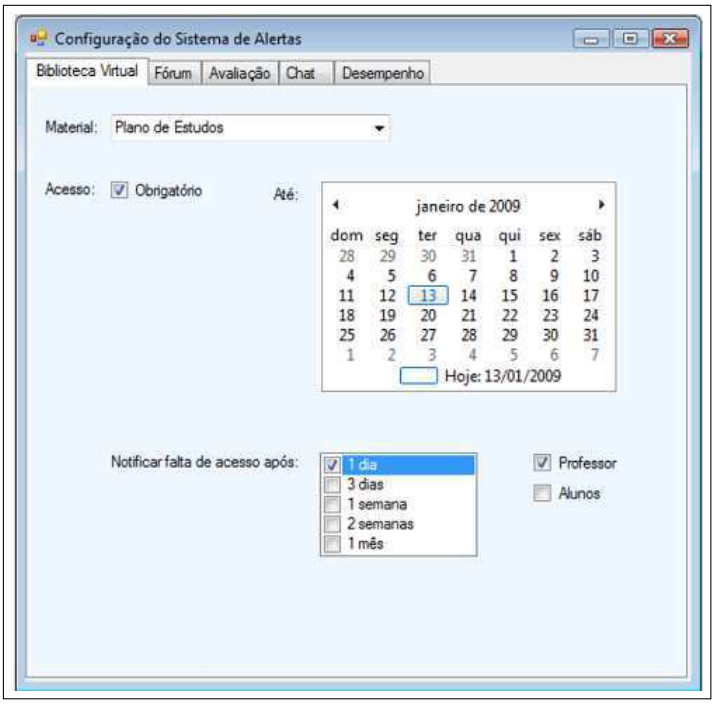


Figura 3.5: Configuração de alertas fixos
Fonte: Extraída de (KAMPFF, 2009)

Gerenciador de alertas de padrões

Os alertas de padrões são obtidos por meio do processo de mineração de dados educacionais e devem ser gerenciados automaticamente pelo sistema. O processo de geração desses alertas se inicia quando o módulo de mineração consulta o *data warehouse* com os dados históricos das disciplinas ou cursos e gera regras de classificação para um determinado período de tempo. O *data warehouse* é atualizado cada vez que o curso é iniciado.

O módulo de aplicação busca as regras de classificação pertinentes e as aplica nos dados dos estudantes que participam da turma que está em andamento, buscando classificar os que se encontram em risco acadêmico. Após a aplicação das regras, os professores serão alertados para tomar ações necessárias, como: enviar uma mensagem, reestruturar material e/ou alterar prazo de entrega.

O módulo de aplicação foi desenvolvido e validado para geração de alertas direcionados aos professores. Esse módulo tem como base a mineração de dados e pode salvar os nomes dos estudantes em risco ou enviar mensagens a eles. O módulo recebe como entrada um arquivo com as regras, que são obtidas a partir da mineração de dados, e uma planilha com os dados das turmas em curso. Após essa etapa, o módulo realiza a classificação como evadido, aprovado, reprovado ou sem acesso, como é apresentado na Figura 3.6. Após a classificação, o professor pode salvar as informações dos estudantes ou enviar um *e-mail* para todos os estudantes classificados.

A fim de realizar os experimentos para os alertas gerados e exemplos de intervenção, Kampff (2009) definiu dois indicadores: Entrega de atividades avaliativas e Regras de classificação. Para a entrega de atividades avaliativas, foi definido um alerta fixo a partir do prazo definido pelo professor. O alerta definido informa ao professor, com dois dias de antecedência do prazo para entrega de cada tarefa, quais os estudantes que ainda não haviam enviado a tarefa. Para as regras de classificação, foram definidos alertas de sistema a partir das regras obtidas pela mineração de dados para um semestre letivo com duração de 5 meses. Assim, ao final dos meses 2, 3 e 4, o professor foi notificado sobre os grupos de estudantes com tendência a aprovação, reprovação e evasão.

Para geração dessas regras de classificação provenientes da mineração de dados utilizadas no alerta de sistema, Kampff (2009) realiza o seguinte processo:

1. **Definição da amostra**: Foram considerados dados de 20 turmas divididas em dois se-

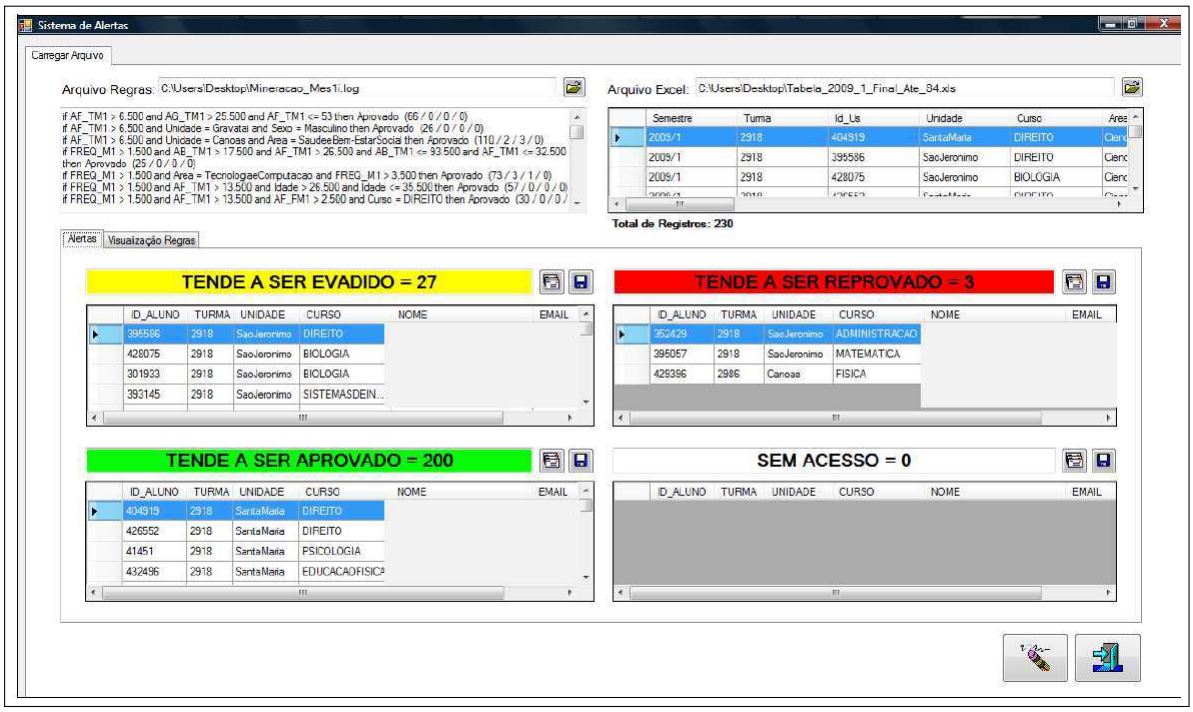


Figura 3.6: Classificação do sistema de alertas
Fonte: Extraída de (KAMPFF, 2009)

mestres letivos;

2. **Organização dos dados**: Compreendeu a fase de extração, limpeza, seleção e transformação dos dados obtidos. Entre os atributos dos estudantes utilizados, estão: Identificação do estudante, notas, sexo, idade, datas de acesso a plataforma, número de acessos as ferramentas, número de acesso a plataforma, número de acessos aos materiais, entre outros. Segundo Kampff (KAMPFF, 2009), ainda houve a necessidade de processar *logs* e realizar outras tarefas para consolidar todos os dados obtidos. Por fim, foram gerados mais alguns atributos. Desta forma, cada estudante foi representado por 158 atributos.

3. **Execução da mineração de dados**: Com a finalidade da geração de alertas, o processo de descoberta de conhecimento foi organizado em cinco momentos com o objetivo de gerar regras para cada mês do semestre letivo. Ou seja, ao final de cada mês foi executado o processo de mineração de dados. A mineração foi realizada utilizando *RapidMiner* e o algoritmo utilizado foi o *RuleLearner*.

4. **Alertas gerados pela mineração de dados**: Ao final dos meses 2, 3 e 4, os profes-

sores receberam alertas sobre os estudantes com tendência a aprovação, reprovação e a evasão. Para isso, o módulo de aplicação, apresentado anteriormente, recebeu como entrada uma planilha com os dados dos estudantes das turmas atuais e um arquivo de regras, obtidas por meio da mineração de dados, realizou a classificação e exibiu os alertas ao professor.

No entanto, a solução proposta por Kampff (2009) tem as seguintes limitações: i) O sistema de alertas não é integrado a uma plataforma educacional. Isso não permite que as intervenções possam ser aplicadas diretamente aos ambientes, como sugeriu Romero e Ventura (2007, 2010, 2013), e limita as decisões que podem ser tomadas pelo professor ao envio de mensagens; ii) A abordagem utilizada dificulta a extensão da solução, pois para realizar e gerar outros tipos de alertas utilizando as regras provenientes da mineração de dados, seria necessário a implementação de outro módulo de aplicação; iii) O processo de mineração de dados é completamente manual. É necessário a realização de todo o processo de KDD[2] de maneira manual, como foi destacado no processo de geração dos alertas de sistema, onde a geração de regras é realizada pela ferramenta *RapidMiner*, a consolidação é feita de forma independente para a geração de um arquivo com os dados dos estudantes e o módulo de aplicação recebe um arquivo com as regras e outro com novos dados para realizar a classificação. Deste modo, é necessário que um especialista repita todo o processo sempre que o professor desejar classificar novos estudantes. Consequentemente, o processo de acompanhamento dos mesmos pelo professor fica limitado; iv) As ferramentas utilizadas no processo de descoberta de conhecimento não são integradas, o que dificulta ainda mais a realização do processo por um usuário não especialista; v) Apesar de oferecer alertas aos professores, o processo de intervenção é restrito a uma ação e, consequentemente, sobrecarrega o professor por não possibilitar automatizar suas ações.

[2]do inglês *knowledge discovery in databases*

3.4 *Open Learning Analytics: An Integrated Modularized Platform*

Neste trabalho, Siemens et al. (2011) propõem o desenvolvimento de um conjunto de ferramentas integradas que auxiliem acadêmicos e organizações a avaliarem as atividades dos estudantes, determinar intervenções necessárias e aprimorar o avanço de oportunidades de aprendizagem.

Com esta proposta, os autores pretendem atender necessidades de educadores/professores, que, segundo Siemens et al. (2011), não possuem acesso a um conjunto de ferramentas integradas que permitam avaliações complexas e variadas da performance e comparações entre grupos de estudantes. Além disso, os estudantes não têm as informações necessárias sobre o seu desempenho. Assim, a solução proposta visa atender a cinco grupos de *stakeholders*: estudantes, educadores/professores, administradores, pesquisadores e analistas de dados.

A solução proposta por Siemens et al. (2011), apresentada na Figura 3.7, tem como base quatro ferramentas específicas e recursos: *learning analytics engine*; *adaptative content engine*; *intervention engine*: recomendações e suporte automatizados; e *dashboard*, relatórios e ferramentas de visualização.

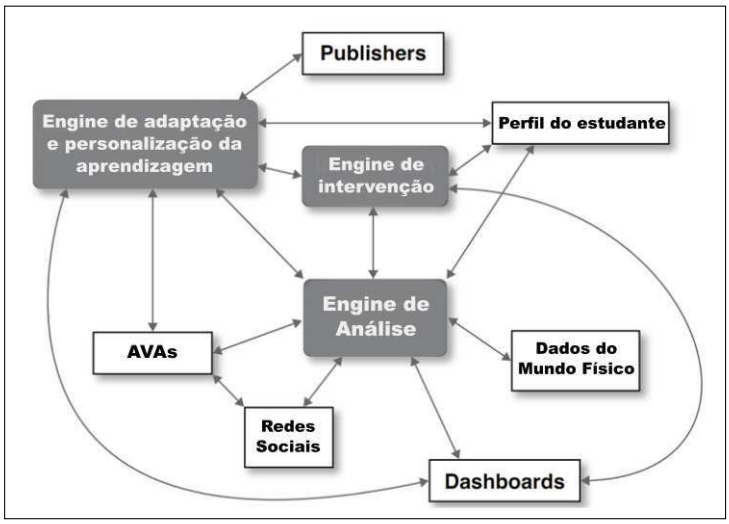


Figura 3.7: Sistema Integrado de *Learning Analytics*
Fonte: Adaptada de (SIEMENS et al., 2011)

O ***Engine de análise*** é o componente central do sistema de *learning analytics* e o res-

ponsável por identificar e processar os dados com base em vários módulos de análises. Segundo Siemens et al. (2011) esses módulos de análises devem ser desenvolvidos como *plugins* por pesquisadores e acoplados à solução, incorporando dados provenientes de diferentes fontes de dados, como: redes sociais, ambientes virtuais de aprendizagem (AVAs) ou *learning management system (LMS)*, entre outros, e aproveitando-se das melhores práticas das áreas de EDM e LAK.

O objetivo com ***Engine de adaptação e personalização da aprendizagem*** é incluir a adaptatividade do processo de aprendizado, *design* instrucional e conteúdo de aprendizagem (SIEMENS et al., 2011). Assim, esse mecanismo será o responsável por definir o perfil do estudante, conforme definido na plataforma educacional e das redes sociais, se assim o estudante permitir.

Engine de intervenção tem como principal função acompanhar o progresso do aluno e fornecerá vários serviços automatizados e intervenções de educadores usando modelos de previsão desenvolvidos no ***engine de análise*** (SIEMENS et al., 2011). Essas intervenções podem ser direcionadas à estudantes, educadores/professores e/ou tutores. Para os estudantes, elas podem ser recomendações de conteúdo, padrões de aprendizagem, enviar *e-mails*, lembrete sobre os cursos, entre outras.

Por fim, o mecanismo de ***dashboard* e visualização** deve, segundo os autores, apresentar os dados de modo visual por meio dos *dashboards* a fim de auxiliar na tomada de decisão no ensino e aprendizagem a partir de quatro visões: i) estudante; ii) educador/professor; iii) pesquisador e iv) institucional. Estudantes poderão, por exemplo, avaliar seu progresso em relação aos dos seus colega, entre outras atividades.

A plataforma proposta tem como principal característica o desenvolvimento modularizado e integrado, onde cada pesquisador ou grupo de pesquisadores podem desenvolver e acoplar suas próprias soluções a plataforma.

3.5 *Developing An Open Architecture For Learning Analytics*

Sclater, Berg e Webb (2015) propõem uma arquitetura de *learning analytics*, apresentada na Figura 3.8, com o objetivo de conceituar elementos básicos para sistemas de *learning*

analytics e servir como modelo para que organizações possam utilizá-la para auxiliar o desenvolvimento de suas próprias arquiteturas e sistemas. A solução proposta tem como característica principal a separação entre seus elementos arquiteturais. Deste modo, segundo os autores, diferentes fornecedores poderão propor soluções para áreas específicas de funcionalidade, gerando potencial para o ambiente onde os componentes podem ser conectados ou substituídos diante da necessidade de seus *stakeholders*. Entre os elementos chaves destacados, estão: *student app*, *learning records warehouse*, *alert and intervention system*, *learning analytics processor*.

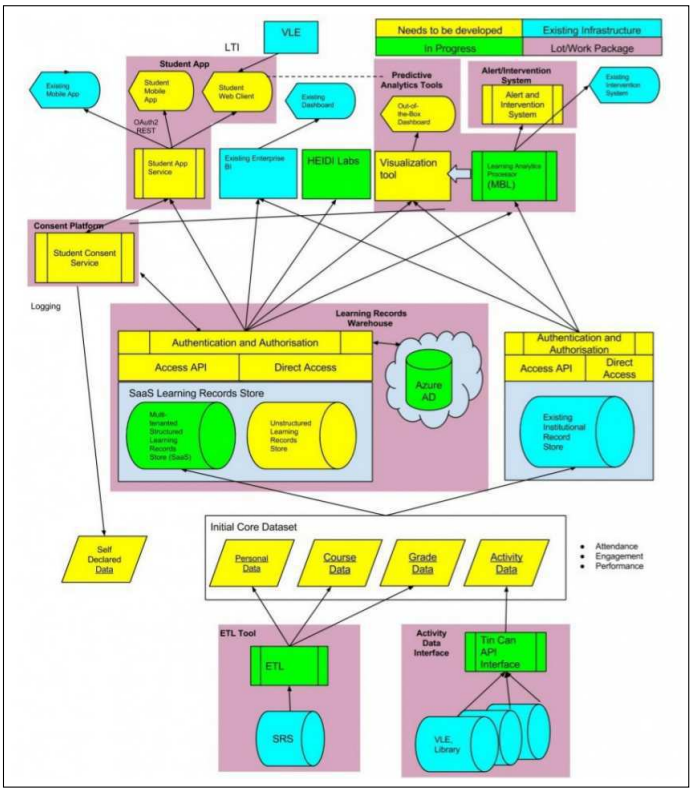


Figura 3.8: Arquitetura proposta
Fonte: Extraída de (SCLATER; BERG; WEBB, 2015)

Student app tem como função oferecer aos estudantes os dados de suas interações, que, de acordo com os autores, pode ser a chave para melhorar as chances do sucesso educacional. Esse componente pode incluir funcionalidades, como: medir engajamento, medir desempenho de avaliação e escolha de módulos por meio de recomendações.

Learning records warehouse é o repositório central que agrupará os dados das atividades dos estudantes, onde podem ser adicionados dados sobre os referidos e seus cursos. Esses dados podem ser obtidos através de um processo de *ETL* e através da *Experience API* (*xAPI*).

O componente *alert and intervention system* deverá permitir que as instituições envolvidas possam gerenciar intervenções junto aos discentes. Os autores preveem que esse componente deve funcionar de maneira semelhante aos sistemas de gerenciamento de relacionamentos com o cliente, mas com o foco no discente. Esse componente poderá auxiliar estudantes e professores/educadores durante o processo de ensino-aprendizagem.

Learning analytics processor é o componente responsável por utilizar técnicas de aprendizagem de máquina para, por exemplo, predizer o sucesso do estudante. Segundo Sclater, Berg e Webb (2015), o sistema deve produzir modelos preditivos baseados nos dados armazenados e validá-los em novos conjuntos de dados. É considerado pelos autores um componente tão crítico que recebeu a recomendação de ocupar uma camada separada.

Por fim, os autores destacam que uma ampla discussão pode influenciar e tem influenciado iniciativas, como a *Apereo Learning Analytics Initiative* [3], que integra um conjunto de peças interligadas para sistemas de *learning analytics* de código aberto. No entanto, os autores ponderam que um *framework* conceitual maduro apoiado por uma implementação de referência de ponta a ponta ainda não emergiu completamente. Além disso, as soluções existentes não contemplam de forma satisfatória características de flexibilidade para adaptação a diferentes domínios e soluções educacionais.

3.6 Comparação Entre as Soluções

Nesta seção é realizado uma comparativo entre os trabalhos relacionados apresentados e a solução proposta neste trabalho. Após a síntese dos trabalhos relacionados, são apresentadas as principais características para atender as necessidades e os desafios destacados na literatura. Estas informações são sumarizadas na Tabela 3.1.

Entre os trabalhos relacionados, o proposto por Siemens et al. (2011) não foi incluído na Tabela comparativa, pois este trabalho foi enquadrado como um modelo de referência e não como uma arquitetura de referência. Como apresentado na Seção 2.1, modelos de referência são utilizados como base para a construção de arquiteturas de referência. Portanto, esse trabalho é caracterizado como uma fonte de informação para o estabelecimento da arquitetura de referência proposta nesta Tese. Os demais trabalhos relacionados são representados na

[3] https://confluence.sakaiproject.org/display/LAI/Learning+Analytics+Initiative

Tabela 3.1 da seguinte maneira:

2. Sclater, Berg e Webb (2015)- *Developing an open architecture for learning analytics*
3. Zorrilla, García e Álvarez (2010) - *A Decision Support System to Improve e-Learning Environments*
4. Jayaprakash et al. (2014) - *Early alert of academically at risk students*
5. Kampff (2009) - Mineração de dados educacionais para a geração de alertas em AVAs

A fim de comparar os trabalhos relacionados, foram definidas 08 (oito) características com base na problemática estabelecida e nas necessidades/desafios apresentados nesta Tese. As características são: i) *Extensível* - visa avaliar se é possível estender a solução para atender requisitos não especificados inicialmente, ou seja, no momento de sua concepção; ii) *Integração com ambientes educacionais* - tem como objetivo avaliar se a solução em questão pode ser integrada a diferentes ambientes educacionais; iii) *Integração de fontes de dados heterogêneas* - verifica se a solução integra dados provenientes de diferentes fontes de dados educacionais; iv) *Descoberta de conhecimento* - visa verificar se a solução oferece suporte ao processo de descoberta de conhecimento; v) *Suporte a tomada de Decisão* - visa verificar se a solução oferece ferramentas de apoio à tomada de decisão; vi) *Personalização de ações* - tem como objetivo avaliar se a solução possibilita a realização de ações pedagógicas e se é possível personalizar as ações que serão executadas; vii) *automatização de ações* - visa avaliar se a solução proposta possibilita automatizar ações pedagógicas, de forma a aumentar a escalabilidade em relação ao tamanho das turmas; viii) *Variabilidade para tradeoffs arquiteturais* - tem como objetivo avaliar se a solução em questão oferece ao projetista a opção de escolher entre eventuais *tradeoffs* que possam existir no momento de instanciar a arquitetura de referência em uma arquitetura de software.

Após a avaliação das propostas, quando atendidas pelos trabalhos relacionados, as características foram avaliadas sob duas perspectivas complementares: modo de execução e perfil do executor. O modo de execução foi classificado em: **M** - Indica a necessidade de intervenção Manual e **SAT** - Indica a existência de apoio Automatizado. Já o perfil do executor foi classificado em: **ESP** - Indica a necessita de intervenção de um especialista em BI

Educacional e **NESP** - Indica que a configuração pode ser realizada por um não-especialista em BI Educacional. Os conceitos marcados com (*) indicam o atendimento parcial.

Tabela 3.1: Comparação entre os trabalhos relacionados

Características	2	3	4	5	Proposta
Extensível	M - ESP	M - ESP	M - ESP*	M - ESP*	M - ESP
Integração com Ambientes Educacionais	M - ESP				SAT - ESP
Integração de fontes heterogêneas de dados	SAT - ESP	SAT - ESP	SAT - ESP	M - ESP	SAT- ESP
Descoberta de Conhecimento	M - ESP*	M - NESP*	M - ESP*	M - ESP*	SAT - ESP
Suporte a Tomada de Decisão	M - ESP*	M - NESP	M - NESP*	M - NESP*	M - NESP
Personalização de ações	M - ESP				SAT - NESP
Automatização de Ações	SAT - ESP				SAT - NESP
Variabilidade para *trandeoffs* arquiteturais					M - ESP

Como pode ser observado na Tabela 3.1, há diferenças significativas entre os trabalhos existentes e a arquitetura de referência proposta. Dentre elas, vale destacar que a arquitetura de referência proposta neste trabalho é a única que considera a variabilidade relacionada à ponderação entre *tradeoffs* arquiteturais. Tal variabilidade possibilitará ao projetista da arquitetura de software a opção de priorização entre os requisitos de qualidade mais críticos em cada contexto específico. Outra característica marcante da arquitetura de referência proposta é o seu foco na automatização e personalização de ações pedagógicas, que pode ser configurada por qualquer usuário, sem a necessidade de ser um especialista em BI Educacional. Além disso, a arquitetura de referência proposta possibilita a integração com diferentes plataformas educacionais de maneira simplificada através de serviços Web, diferentemente de outras propostas que necessitam acesso ao código fonte e/ou grandes adaptações para realizar essa integração. Mais detalhes são apresentados nos Capítulos 5 e 7, que tratam, respectivamente, da apresentação da arquitetura de referência proposta e da avaliação e refinamento da dessa arquitetura de referência.

Capítulo 4

Processo de Estabelecimento e Refinamento de Arquitetura de Referência

É apresentada neste capítulo uma visão geral do processo para estabelecimento de arquitetura de referência adotado neste trabalho, denominado ProSA-RA (Seção 4.1), bem como, o seu refinamento, proposto no contexto desta tese, para a realização da etapa de avaliação e refinamento arquitetural (Seção 4.2). Para evitar redundâncias, a apresentação detalhada do processo de projeto arquitetural e do método de refinamento proposto é relatada no decorrer do trabalho, no âmbito da arquitetura de referência proposta. Tal apresentação foi divida em três capítulos. Inicialmente, no Capítulo 5, é apresentada a execução das etapas iniciais do processo ProSA-RA. Em seguida, No Capítulo 6, a arquitetura concreta é instanciada. O processo de instanciação envolveu o projeto da arquitetura concreta e a implementação dessa arquitetura utilizando componentes de software escritos em Java. Por fim, é apresentada no Capítulo 7 a etapa de avaliação e refinamento arquitetural realizadas, que representa a última fase do ProSA-RA, seguindo o método de avaliação e refinamento proposto na presente tese.

4.1 O Processo ProSA-RA

O processo adotado como base para o estabelecimento da arquitetura de referência para a tomada de decisão pedagógica proposta neste trabalho foi o ProSA-RA. Com sua primeira

versão apresentada em (NAKAGAWA et al., 2009), o ProSA-RA é um processo que sistematiza o *design*, a representação e a avaliação de arquiteturas de referências (NAKAGAWA et al., 2014). Esse processo tem sido experimentado no desenvolvimento de arquiteturas na academia e na indústria. Apresentado na Figura 4.1 e detalhado a seguir, o processo é composto por quatro etapas: i) Investigação das fontes de informação; ii) Análise arquitetural; iii) Síntese arquitetural; iv) Avaliação Arquitetural.

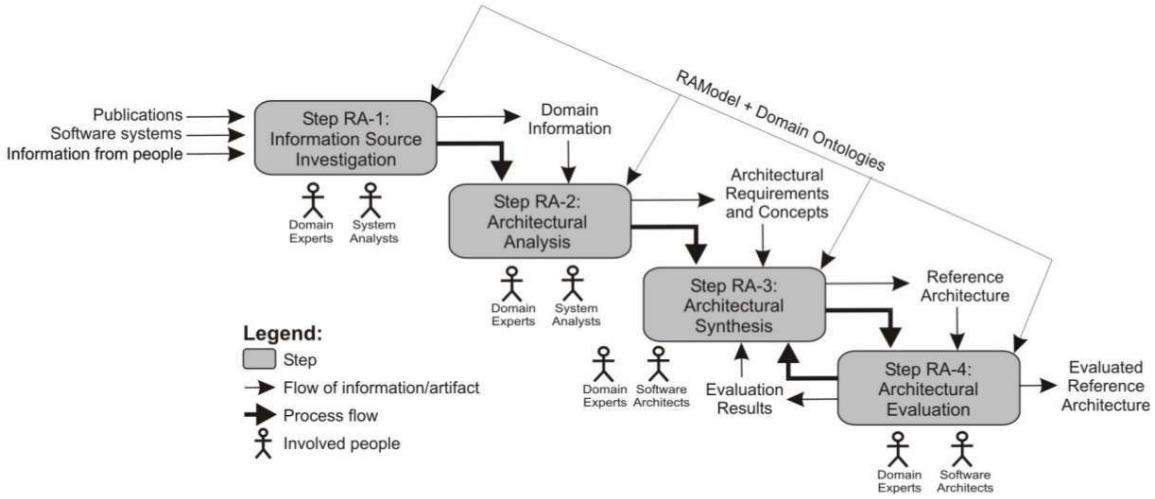


Figura 4.1: Estrutura do ProSA-RA
Fonte: Extraída de (NAKAGAWA et al., 2014)

A primeira etapa (RA-1) consiste no levantamento de informações relevantes para o estabelecimento da arquitetura de referência. Essas fontes devem prover informações sobre os processos e atividades que podem ser suportadas por sistemas do domínio alvo (NAKAGAWA et al., 2014). São consideradas importantes fontes de informação (NAKAGAWA et al., 2014): i) Pessoas - clientes, usuários, pesquisadores, desenvolvedores e outros indivíduos envoltos no domínio definido; ii) Sistemas de software - os principais sistemas relacionados ao domínio investigado. Também são investigados suas arquiteturas, capacidade de evolução, além de seus estilos e padrões arquiteturais; iii) Publicações - são investigadas publicações relacionadas ao domínio definido, tais como: livros, artigos, relatórios técnicos, entre outros; iv) Modelos de referência e Arquiteturas de referência - busca-se investigar outros modelos e/ou arquiteturas de referência no domínio escolhido ou em domínios correlatos; e v) Ontologias de domínio - são levantadas ontologias que representem terminologias do domínio especificado.

Na segunda etapa (RA-2), segundo Nakagawa et al. (2014), o objetivo do projetista é identificar os principais requisitos do sistema, requisitos da arquitetura de referência e os conceitos que devem ser considerados no processo de estabelecimento da arquitetura de referência com base nas informações levantadas na etapa RA-1.

Na terceira etapa (RA-3), o projeto arquitetural da arquitetura de referência é construído. Para tanto, estilos e padrões arquiteturais, bem como, uma combinação entre esses e outros estilos identificados na etapa RA-1 devem ser considerados (NAKAGAWA et al., 2014). Na apresentação do projeto arquitetural, diferentes visões arquiteturais podem ser adotadas. Para representar essas visões, diagramas UML podem ser utilizados.

Por fim, a última etapa (RA-4) envolve o processo de avaliação arquitetural. Nakagawa et al. (2014) propõe o uso de uma abordagem baseada em *checklist* tais como o FERA (*Framework for evaluation of Reference Architectures*) (SANTOS et al., 2013). Em resumo, essa abordagem utiliza um conjunto de questões a fim de identificar defeitos na documentação da arquitetura proposta. Adotar a abordagem *checklist*, no entanto, não é obrigatória. Nakagawa, Oquendo e Maldonado (2014) sugerem também que outras iniciativas podem ser utilizadas na avaliação de arquiteturas de referência, tais como: *Software Architecture Analisys Method* (SAAM) (GRAAF; DIJK; DEURSEN, 2005) e *Architecture Tradeoff Analisys Method* (ATAM) (KAZMAN; KLEIN; CLEMENTS, 2000).

Após a definição do processo de estabelecimento da arquitetura de referência, a etapa seguinte é definir quais métodos serão adotados na etapa de avaliação arquitetural. Desse modo, a Seção 4.2 apresenta uma visão geral do refinamento, proposto no contexto desta tese, para a execução da etapa RA-4.

4.2 Método Proposto para Avaliação e Refinamento Arquitetural

Para esta etapa, é previsto no ProSA-RA a utilização de diferentes métodos de avaliação arquitetural. Os autores, por sua vez, recomendam a adoção de uma abordagem de avaliação arquitetural baseada em *checklist*, tais como, o FERA (SANTOS et al., 2013). Apesar das vantagens oferecidas, como a possibilidade de execução através de questionários *online*, esses métodos de avaliação arquitetural dificultam a identificação de *trade-offs* e gargalos

arquiteturais durante o processo de avaliação. Diante das demandas geradas no domínio educacional - tais como: grande número de estudantes, alta demanda de requisições, grande volume de dados; a avaliação de atributos de qualidade como escalabilidade e desempenho - foi necessário investir esforço no refinamento de um método que, de fato, execute a arquitetura de referência para obter dados que possam auxiliar o projetista de maneira satisfatória, em cenários realistas de utilização.

Nesse contexto, o ATAM (KAZMAN; KLEIN; CLEMENTS, 2000), apresentado na Seção 2.2, é um método bem experimentado e bastante usual tanto na indústria quanto na academia, e pode ser utilizado para avaliar e aprimorar partes estratégicas da arquitetura de referência com o objetivo de melhorar os atributos de qualidade. A aplicação do ATAM na avaliação de arquiteturas de referência já foi experimentada por Gallagher (2000). Outro ponto fundamental para a adoção do ATAM é a possibilidade de identificação de *trade-offs* e gargalos arquiteturais a partir do processo de avaliação.

Apesar das vantagens apontadas, o ATAM necessita de valores aproximados de execução dos componentes arquiteturais como entrada para a realização do processo de avaliação. Essa característica aumenta a subjetividade da avaliação, uma vez que a torna uma espécie de ”simulação”da execução real do software. Em vez disso, optou-se por implementar um *framework* que realiza a arquitetura de referência proposta, a fim de instanciá-la em uma arquitetura de software real. Os dois motivos principais que justificam esse esforço adicional são: i) para que fosse possível obter dados mais precisos e realistas, próximos a cenários reais de utilização da arquitetura e seus componentes, o que é fundamental para a identificação de gargalos arquiteturais com maior precisão; e ii) apontar tecnologias e direcionamentos concretos para os principais componentes da arquitetura de referência proposta neste trabalho.

A visão geral do método proposto para avaliação e refinamento da arquitetura de referência é apresentado na Figura 4.2. Como pode ser observado, o método proposto é composto por 10 etapas. As 4 primeiras etapas, destacadas em rosa na Figura 4.2, seguem o método ATAM com a identificação dos atributos de qualidade, geração da árvore de utilidade, geração dos cenários para cada atributo e a execução da arquitetura para cada cenário definido. A partir da quinta etapa, o método se diferencia do ATAM. Na quinta e sexta etapas são avaliadas as métricas de cada cenário e sumarizados os resultados da avaliação, respec-

tivamente. Na Etapa 7 são identificados os impactos arquiteturais em busca dos gargalos arquiteturais. Nas Etapas 8 e 9 são sumarizadas as avaliações dos atributos de qualidade em busca dos *trade-offs*. Por fim, caso os atributos de qualidades não tenham sido atendidos apropriadamente, é executada a décima etapa, que consiste no refinamento arquitetural.

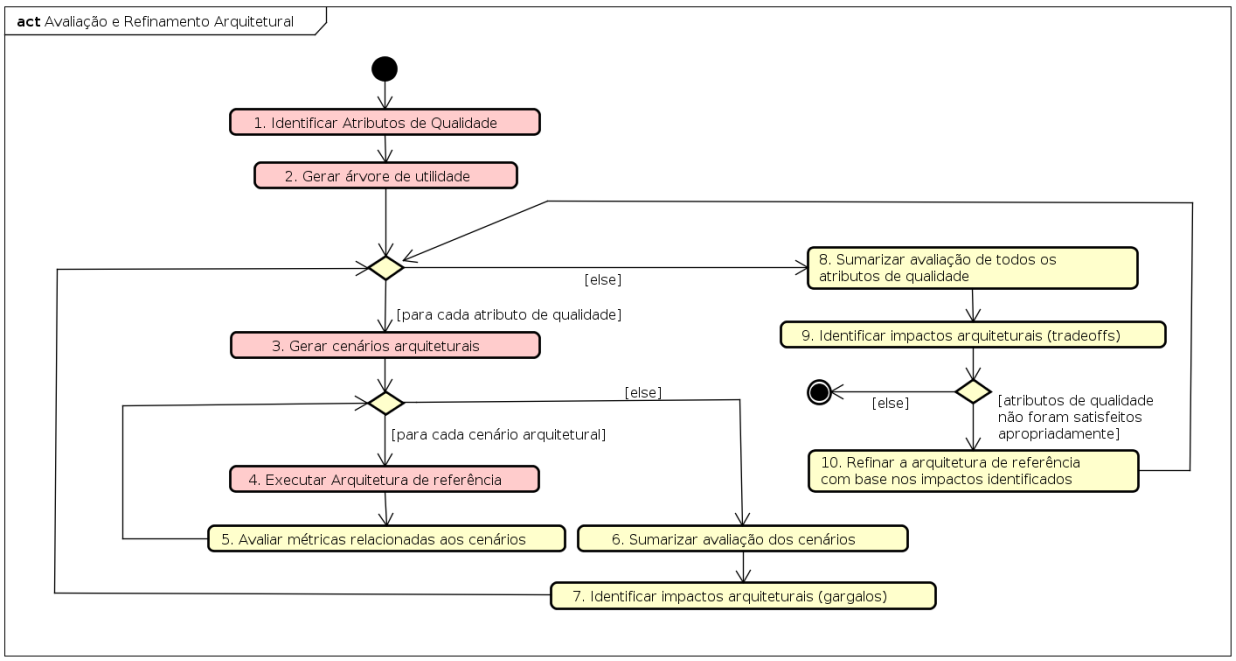


Figura 4.2: Etapa de Avaliação Quantitativa e Refinamento
Fonte: Elaborada pelo autor

Os detalhes do método proposto são apresentados no Capítulo 7, no contexto da avaliação e refinamento da arquitetura de referência proposta neste trabalho.

Capítulo 5

Projeto da Arquitetura de Referência

Neste capítulo é apresentado o processo de estabelecimento da versão inicial da arquitetura de referência, voltada à tomada de decisão em contextos educacionais. Com base no ProSA-RA, as duas primeiras atividades são apresentadas nas Seções 5.1 e 5.2, respectivamente. O projeto arquitetural, terceira etapa do ProSA-RA, da arquitetura de referência é apresentada através de cinco visões arquiteturais, de acordo com o modelo 4+1 (KRUCHTEN, 1995): visão lógica, visão de desenvolvimento, visão de processos, visão de implantação e a visão do usuário. Na Seção 5.3 são apresentadas as visões lógica e de processos da arquitetura de referência proposta, enquanto as demais visões são apresentadas no Capítulo 6, que abrange a instanciação e implementação da arquitetura de referência.

5.1 Identificação das Fontes de Informação

A fase inicial, proposta pelo ProSA-RA, consistiu em investigar as fontes de informação no domínio do processo de tomada de decisão pedagógica a fim de identificar os requisitos para estabelecer a arquitetura de referência. Assim, foram definidos dois grupos de fontes de informação: i) soluções para apoiar a tomada de decisões pedagógicas em ambientes educacionais; ii) arquiteturas e modelos de referência para a tomada de decisões pedagógicas.

Para o primeiro grupo, os objetivos foram levantar as principais soluções no contexto do apoio à tomada de decisões em ambientes educacionais e conhecer as principais técnicas/tecnologias aplicadas nessas soluções. Dessa maneira, foi realizada uma revisão de literatura, cujo protocolo é apresentado no Apêndice A. Tal revisão foi realizada com o obje-

tivo de identificar os trabalhos mais relevantes, as principais características/funcionalidades, as técnicas/tecnologias aplicadas no desenvolvimento dessas soluções, e os principais estilos e padrões arquiteturais adotados por essas soluções.

Como resultados do primeiro grupo, as principais técnicas e aplicações identificadas no domínio da tomada de decisão pedagógica foram: i) learning analytics (LAK); ii) mineração de dados educacionais (EDM); iii) estatística; iv) *social network analysis*; e v) visualização de informações. Entre os principais estilos e padrões arquiteturais, foram identificadas (SHAW; GARLAN, 1996; BUSCHMANN et al., 1996): i) arquitetura centrada em dados; ii) arquitetura orientada a serviços; iii) *pipes and filters*; iv) estilo baseado em regras; v) modelo cliente-servidor; vi) *Model-View-Controller* (MVC); e vii) arquitetura em camadas. Por fim, foram identificadas as principais características e funcionalidades oferecidas pelas soluções selecionadas: i) integração entre um ambiente educacional e a solução proposta; ii) interface simplificada; iii) Visualização de dados - *dashboards*, relatórios, entre outros; iv) intervenção junto aos estudantes, entre outros.

Na busca por arquiteturas de referência para a tomada de decisão, nenhuma foi encontrada com as mesmas características e abrangência da arquitetura proposta. No entanto, foram encontradas arquiteturas de referência em domínios correlatos dentro do contexto educacional, tais como os trabalhos relacionados apresentados no Capítulo 3. Além desses trabalhos, outros menos relacionados também foram considerados como fonte de informação, uma vez que possuem características complementares que podem derivar requisitos relevantes. Os trabalhos adicionais considerados apresentam modelos de referência que auxiliaram durante o processo de estabelecimento da arquitetura proposta nesta tese. Os modelos adicionais identificados foram: (SIEMENS et al., 2011), (PALANIVEL; KUPPUSWAMI, 2011), (BAKHARIA et al., 2016) e (CHATTI; MUSLIM; SCHROEDER, 2017).

5.2 Estabelecimento dos Requisitos Arquiteturais

A segunda atividade consistiu na realização da identificação e análise baseada nos resultados obtidos na atividade anterior. Os requisitos identificados são apresentados a seguir:

- Facilitar o desenvolvimento de sistemas de apoio à tomada de decisão em ambientes educacionais;

- Executar o processo de descoberta de conhecimento através de uma interface simplificada que pode ser usada por usuários não-especialistas;
- Oferecer uma interface para análise de dados através de *dashboards*, relatórios, visualização, entre outras;
- Apoiar a tomada de decisão em plataformas educacionais com baixa complexidade para a adição, modificação e exclusão de funcionalidades;
- Permitir a adoção de diferentes algoritmos de serviços, tais como mineração de dados, visualização de dados, entre outros;
- Facilitar o processo de adição, modificação e exclusão de algoritmos na solução, tais como mineração de dados, visualização de dados, entre outros;
- Executar ações/intervenções diretamente em ambientes educacionais. Tais ações podem ser manuais ou automatizadas;
- As ações automatizadas devem ser configuradas pelos usuários usando uma interface simplificada;
- Integrar dados provenientes de diferentes fontes de dados educacionais;
- Facilitar a substituição de componentes através da redução do acoplamento entre eles;
- Ser capaz de atender a um número crescente de requisições simultâneas de acordo com a demanda de professores/educadores e estudantes;

5.3 Consolidação da Arquitetura de Referência

Esta seção apresenta os principais artefatos gerados durante o projeto da versão inicial da arquitetura de referência proposta. Tais artefatos consistem na visão lógica (Seção 5.3.2) e visão de processos (Seção 5.3.3). Para a geração dos artefatos, foram consideradas decisões de projeto importantes, que são apresentadas na Seção 5.3.1.

5.3.1 Análise das Decisões Arquiteturais

Nesta Seção, são discutidas as principais decisões de projeto tomadas durante o desenvolvimento da arquitetura de referência. A discussão inicial aborda as motivações que levaram à escolha de um *Data Warehouse* para o contexto educacional. Em seguida, são apresentadas as questões que levaram à evolução deste *Data Warehouse* para um *Data Warehouse* Ativo direcionado ao contexto educacional.

Opção por um *Data Warehouse* A primeira decisão de projeto tomada foi o desenvolvimento do projeto de um *Data Warehouse* (DW) educacional. O *DW* Educacional visa superar importantes obstáculos e desafios encontrados por professores e demais *stakeholders* durante o processo de tomada de decisão em contextos educacionais apontados nesse trabalho. Zorrilla (2009) exemplifica um desses obstáculos: A autora conclui que apesar das diversas informações disponibilizadas por ambientes educacionais, elas não são suficientes para a realização do processo de tomada de decisão por parte dos atores envolvidos. Ela destaca o exemplo de alguns instrutores, que encontram sérias dificuldades em extrair informações úteis quando há um grande número de estudantes e a diversidade das interações são altas.

Diante das necessidades, obstáculos e requisitos da arquitetura apontados neste trabalho, a opção por uma ferramenta de análise de dados que atenda as diferentes necessidades e objetivos dos professores e demais *stakeholders* envolvidos, além de permitir sua integração e reutilização em diferentes ambientes educacionais, justifica a escolha de um *Data warehouse* educacional.

Zorrilla (2009) também aponta algumas razões para a adoção de um *data warehouse* com o enfoque educacional:

1. **Diferentes fontes de dados** - Os dados gerenciados (provenientes de cursos virtuais, arquivos de *logs* de ambientes *e-Learning*, dados demográficos e acadêmicos dos estudantes, informação de ingresso / matrículas, e mais) são, usualmente, distribuídos em diferentes fontes de dados e, muitas vezes, a mesma informação é armazenada em formatos heterogêneos e codificação.

2. **Dados não orientados à decisão** - É necessário extrair e transformar dados que in-

teressem a professores e educadores a partir de diferentes fontes de dados a fim de identificar os indicadores-chave que possam ser utilizados para medir o desempenho dos estudantes e, depois disso, carregá-los em um banco de dados específico para a tomada de decisões, por exemplo: os dados das páginas solicitadas por cada aluno estão, geralmente, em um arquivo de *log* que precisa ser processado antes da extração dessas informações.

3. **Execução de consultas complexas** - trabalhar com um banco de dados que segue o modelo dimensional possibilita a execução de consultas complexas. Nesta perspectiva, a construção de cubos OLAP é facilitada.

4. **Extensibilidade** - o *data warehouse* é facilmente expansível e, portanto, também pode fornecer, no futuro, soluções para as necessidades de outros atores envoltos nos ambientes educacionais.

Por fim, Zorrilla (2009) corrobora que um *data warehouse* é, por definição, um sistema de informação específico para a tomada de decisões, orientado pelas necessidades dos usuários, alimentado por diferentes fontes de dados, construído e apresentado usando uma perspectiva simples que atende aos requisitos da aplicação.

Opção por um *Data Warehouse* Ativo - Apesar das vantagens obtidas com a opção de um *data warehouse* educacional - tais como: ser expansível para atender às diferentes necessidades de professores e educadores, fornecer informações de qualidade, integrar fontes heterogêneas, entre outras - esta opção não atende todas as questões levantadas e que orientam este trabalho. Assim, optou-se pela evolução do *data warehouse* educacional para um *data warehouse* educacional ativo. As justificativas que nortearam esta evolução são apresentadas abaixo.

É importante para o processo de ensino-aprendizagem que os professores realizem o acompanhamento e a avaliação dos estudantes em tempo real (PEÑA-AYALA, 2014) ou próximo a isso, dentro ou fora dos ambientes educacionais. No entanto, devido a questões como o grande número de estudantes (ZORRILLA, 2009), a alta diversidade destes e suas interações (ZORRILLA, 2009) (LIÑÁN; PÉREZ, 2015), (PEÑA-AYALA, 2014) e altas taxas na relação estudantes-instrutores (LIÑÁN; PÉREZ, 2015), esse monitoramento dentro

dos ambientes educacionais torna-se uma tarefa complexa demais e dispendiosa para professores e os demais atores envolvidos.

Nesse contexto, Jayaprakash et al. (2014) ressaltam que com o advento das técnicas *Learning Analytics* e o uso de sistemas de alertas automáticos, é possível, por exemplo, fornecer *feedbacks* automatizados, permitindo a execução de uma intervenção antecipada junto a estudantes em situação de risco acadêmico para que tenham tempo suficiente e possam mudar seus comportamentos e estratégias. Como solução para essas questões, um sistema com capacidades reativas e com ações que possam ser automatizadas é necessário para esse monitoramento contínuo e personalidado dos estudantes, como apontado por (PEÑA-AYALA, 2014), (LIÑÁN; PÉREZ, 2015) e (JAYAPRAKASH et al., 2014).

Em vista das questões discutidas acima, a opção pela evolução do *data warehouse* educacional para um *data warehouse* educacional ativo apoia-se em sua principal característica: a capacidade reativa (SCHREFL; THALHAMMER, 2000). Diversos autores enfatizam a relevância dessa capacidade reativa no mundo dos negócios: (HAJLAOUI; HAMDANI, 2014), (BOUATTOUR et al., 2009) e (BOSE; HUI, 2013).

Hajlaoui e Hamdani (2014) evidenciam que a tomada de decisões pode automatizar rotinas e tarefas de decisão através de um *data warehouses* ativo. De acordo com Bouattour et al. (2009), em um ambiente competitivo, as empresas precisam de um sistema de decisão reativo que possa automatizar relatórios de desempenho, realizar análise de manutenção de maneira dinâmica e aperfeiçoar o controle de usuário sobre o processamento analítico com a finalidade de equipar *data warehouses* e cubos OLAP com essas capacidades. Como exemplo de soluções baseadas em *data warehouses* ativo, *Teradata* oferece o *Teradata Active Enterprise Data Warehouse* (TERADATA, 2017), que tem como objetivo oferecer inteligência em tempo real para colocar dados vitais nas mãos dos tomadores de decisão da linha de frente (TERADATA, 2015).

Um *data warehouse* ativo agrega vantagens, como: sistema de decisão reativa para automatizar a produção de relatórios, análises dinâmicas, carga de dados em tempo real, tomada de decisão efetiva, entre outros. Dessa forma, entende-se que a adaptação de um *data warehouse* ativo ao contexto educacional apresenta-se como uma solução para as questões levantadas. Essa solução deve possibilitar que professores e educadores conduzam uma avaliação contínua de seus estudantes através de intervenções automatizadas em ambien-

tes educacionais, permitindo a automatização de ações em situações identificadas durante o processo de tomada de decisão. Nesse contexto, um professor pode ao identificar, durante o processo de tomada de decisão, que estudantes que participam pouco dos fóruns tendem a abandonar as disciplinas as quais estão matriculados. Desse modo, pode-se configurar o *data warehouse* ativo, inserindo uma nova regra, para receber um *e-mail* de alerta sobre esses estudantes e/ou enviar uma mensagem aos que possuem participação abaixo da média nos fóruns.

5.3.2 Visão Lógica da Arquitetura

É apresentada nesta Seção a visão lógica da arquitetura de referência para a tomada de decisão em contextos educacionais. O desenvolvimento do projeto arquitetural, que gerou a visão lógica apresentada na Figura 5.1, levou em consideração as informações obtidas nas etapas anteriores, tais como: estilos e padrões arquiteturais, funcionalidades, requisitos identificados, entre outras. Como pode ser observado na Figura 5.1, a arquitetura de referência proposta adota uma arquitetura heterogênea envolvendo diferentes estilos arquiteturais, entre eles: centrado em dados (repositório), orientado a serviços, *pipes and filters* e baseado em regras. A arquitetura centrada em dados foi adotada a fim de facilitar a integração e comunicação entre os módulos que possam ser desenvolvidos em linguagens diferentes e que não interajam diretamente entre si, através de serviços Web. Nesse caso, a integração se dá pelo compartilhamento de dados. A arquitetura orientada a serviços foi adotada para facilitar a integração entre ferramentas de descoberta de conhecimento e/ou algoritmos de mineração e ambientes *e-Learning*, como o Moodle, além de facilitar a atuação nos ambientes fruto da tomada de decisão. O estilo *pipes and filters* foi adotado para explicitar a execução dos estágios sequenciais do processo de descoberta de conhecimento: pré-processamento, processamento e pós-processamento. Variações de algoritmos em cada um desses estágios são possíveis graças à adoção de diferentes versões de filtros, que são selecionadas automaticamente de acordo com as necessidades dos usuários. O estilo baseado em regras foi adotado a fim de permitir gatilhos envolvendo ações em ambientes *e-Learning*, com base em informações descobertas no *data warehouse*. Assim, essas regras favorecem a característica ativa do *data warehouse*, tornando o processo de tomada de decisão mais rápido e automatizado. O estilo arquitetural baseado em regras também possibilita que o comportamento ativo

do sistema seja definido e refinado por especialistas no domínio de ensino (educadores), sem exigir conhecimento de desenvolvimento de sistemas. Para isso, basta disponibilizar um editor de regras com interface amigável.

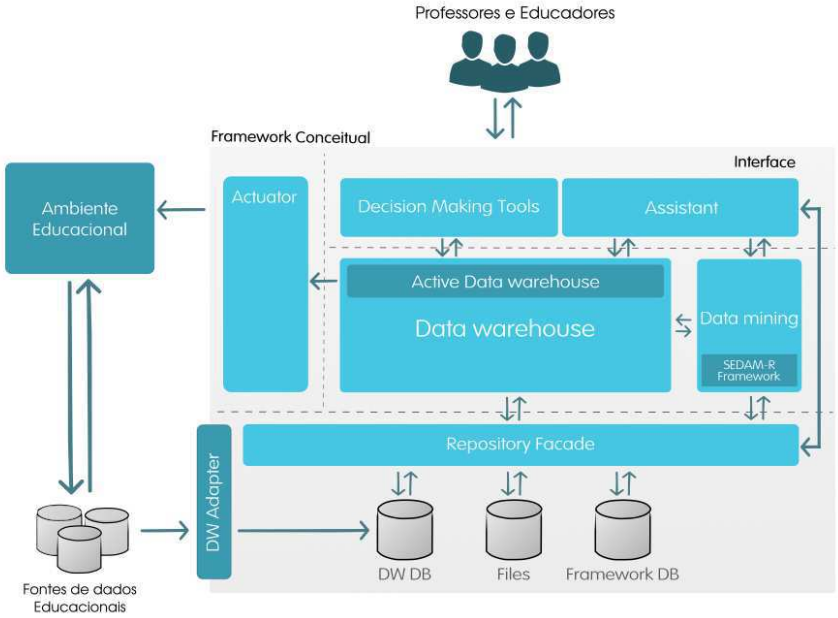


Figura 5.1: Versão Inicial da Arquitetura de Referência Proposta
Fonte: Elaborada pelo autor

Nas seções a seguir, os componentes da arquitetura de referência são detalhados, apresentando suas principais características, funções, e os estilos e padrões arquiteturais adotados por cada um.

Data Warehouse

O *DW DB* ou *DW Database* representa a base de dados do *DW* em si. Ele foi apresentado separadamente apenas para explicitar o estilo arquitetural centrado em dados, pois a integração entre a arquitetura proposta e os ambientes educacionais é centrada em dados.

Data Warehouse Adapter

O componente ***Data Warehouse Adapter*** tem a função de importar e adaptar os dados das diferentes fontes de dados que envolvem o contexto educacional, como: dados dos ambientes *e-Learning*, dados de *logs* gerados pelos ambientes, dados de sistemas acadêmicos, entre outras fontes de dados utilizadas. Os dados adaptados são armazenados de forma integrada

no ***DW Database***, que é acessado pelo componente *Data Warehouse*. Para isso, os dados são adaptados para o modelo de referência proposto para este trabalho, utilizando ferramentas como o Pentaho (PENTAHO, 2016) e *scripts* específicos. O componente *adapter* também pode adotar outras tecnologias a fim de aumentar ainda mais as possibilidades de integração. Assim, esse componente pode ser construído para adotar, por exemplo, a *Experience API* (xAPI)[1].

Active Data Warehouse

O ***Active Data Warehouse*** tem como função principal estender o modelo do *Data Warehouse* tradicional a fim de prover capacidade reativa. Este componente possibilita configurar ações automáticas através de regras previamente cadastradas. Tais regras podem utilizar condições decorrentes de informações descobertas na base de dados integrada e ativadas automaticamente a partir de uma atualização dos dados. Exemplos de possíveis ações configuradas por professores e educadores são: i) realizar intervenções junto ao estudante dentro do ambiente educacional, como: enviar mensagens, criar novos fóruns, criar grupos de estudantes, entre outras; ii) produzir e enviar relatórios periódicos aos atores envolvidos e/ou quando houver alguma alteração importante nos dados; entre outras ações.

Este componente tem uma arquitetura baseada em regras e utiliza essas regras para estabelecer quando e em quais condições as ações definidas devem ser executadas. Essas regras, bem como as ações definidas, são configuráveis por professores e educadores por meio de um assistente, representado na arquitetura como o elemento ***Assistant***.

As regras são representadas pela tríade: *Event*, *Condition* e *Action*, como ilustrado na Figura 5.2. O *evento* determina a frequência de execução da regra, enquanto a *condição* determina a verificação prévia a ser satisfeita para que a *ação* estabelecida seja executada. A condição da regra é formada por expressões lógicas que podem utilizar dados oferecidos diretamente por um dos bancos de dados integrados, tais como: número de postagem nos fóruns, número de acesso aos fóruns, número de materiais disponíveis, entre outras. Os atributos utilizados como critérios são resultados da sumarização de dados provenientes de ferramentas disponíveis no ambiente *Moodle*, que foram mapeados e incluídos no *data warehouse*, como destacado em Seções anteriores.

[1] https://experienceapi.com/overview

Figura 5.2: Modelo de Regras Simples
Fonte: Elaborada pelo autor

De forma complementar, um novo modelo de regra é proposto nesta Tese, onde as condições de uma regra podem utilizar informações não presentes nos bancos de dados, como ilustrado na Figura 5.3. As informações não presentes no banco de dados referem-se a informações que podem ser descobertas a partir deles, utilizando técnicas como a mineração de dados. Esse modelo estende as *analysis rules* inicialmente proposta por Thalhammer, Schrefl e Mohania (2001), que utilizam um conjunto de condições para realizar análises multidimensionais que anteriormente eram realizadas manualmente por analistas. No modelo proposto neste trabalho, as condições das regras envolvem a aplicação de modelos de mineração sobre novos dados. Para cada novo modelo, a execução desse processo envolveria: i) recuperação dos dados associados ao modelo - quais métricas utilizadas para a construção do modelo e as quais grupos de dados se destina o modelo (grupos de estudantes e/ou disciplina, por exemplo); ii) frequência de sumarização dos dados - se os dados do modelo foram sumarizados por semana, a aplicação do modelo deve seguir a mesma sumarização; iii) aplicação do modelo de mineração e a recuperação da classificação atribuída. Deste modo, após a realização de todo esse processo e com o resultado da aplicação realizada, as ações correspondentes podem ser disparadas. Por exemplo, uma regra poderia ter como condicional a motivação do estudante, e caso ele seja classificado como desmotivado, uma mensagem seria enviada ao mesmo. Além disso, as condições podem utilizar informações descobertas em dois momentos diferentes a fim de compará-las. Por exemplo, na segunda semana de curso, alguns estudantes que foram classificados como motivados, mas na semana seguinte, foram classificados como desmotivados. A mudança de comportamento também pode ser utilizada com condicional de regra, assim, uma regra poderia ser especificada da seguinte maneira: se um estudante que foi classificado como motivado em uma semana, mas, na semana seguinte, foi classificado como desmotivado, então uma mensagem pode ser

enviada a este estudante, caso o professor/educador deseje.

Para criação dos modelos de mineração, professores e educadores podem utilizar o assistente específico para esse fim, representado pelo componente ***Assistant***, para execução do processo de descoberta de conhecimento. O ***Assistant*** oferece uma interface baseada em objetivos que possibilita a execução de algoritmos de mineração disponíveis no componente **Data Mining** e o armazenamento dos modelos construídos. Dessa maneira, professores e demais *stakeholders* podem, posteriormente, utilizá-los como condicionais de regras. Um professor pode, por exemplo, executar o processo de mineração de dados e descobrir informações relacionadas a probabilidade de evasão, desse modo, ele pode salvar o modelo de mineração construído e, em seguida, criar uma regra que gere um relatório para ele contendo os estudantes que tenham alta probabilidade de evasão. Outras informações poderiam ser descobertas, tais como: índice de motivação, probabilidade de aprovação, entre outras, e, consequentemente, utilizadas em condicionais de regras.

Figura 5.3: Modelo de Regras Avançadas
Fonte: Elaborada pelo autor

Em síntese, sempre que acontecem alterações no ***DW Database***, um gatilho é disparado. Ao perceber que aconteceram alterações, o componente de monitoramento de notificações verificará quais alterações foram realizadas e se essas alterações são suficientes para acionar o componente de execução de regras. A partir deste ponto, se as condições forem satisfeitas, os novos dados relacionados àquele conjunto de regras são carregados e as regras, agendadas para execução, de acordo com a frequência do seu *evento*. No entanto, o ***Active Data Warehouse*** não executa diretamente as ações definidas nas regras como acontece em um modelo de ***Active Data Warehouse*** tradicional. ***Active Data Warehouse*** proposto aciona o componente ***Actuator***, que é o responsável pela execução das ações definidas diretamente dentro dos ambientes educacionais. Essa opção foi feita com a finalidade de aumentar a coesão do sistema, pois, dessa maneira, o ***Actuator*** terá o foco no domínio educacional, enquanto

Active Data Warehouse será independente de domínio, potencializando sua utilização em diferentes domínios.

Actuator

Esse componente tem a função de executar as ações definidas em regras diretamente dentro dos ambientes educacionais. Para isso, o atuador é focado em um domínio, ou seja, é considerado um ponto de extensão (*hot spot*) da arquitetura de referência proposta. Dessa forma, é necessário construir um atuador para cada ambiente educacional integrado. Com cada atuador focado em um domínio especifico, a solução proposta tem uma clara separação de responsabilidades, onde o ***Active Data Warehouse*** verifica as regras e, se necessário, aciona o ***Actuator***, que pode acionar recursos da plataforma educacional. Assim, a arquitetura proposta torna-se mais flexível e possibilita que os demais componentes da arquitetura sejam reutilizados na íntegra em diferentes AVAs, alcançando uma maior coesão e um baixo acoplamento.

A integração entre os atuadores e os ambientes educacionais é realizada através de serviços Web, onde o atuador executa os serviços disponibilizados pelos ambientes. As vantagens no uso serviços Web incluem: simples integração, baixo acoplamento, escalabilidade, portabilidade, disponibilidade, entre outros.

Componente de *Data Mining*

Esse componente fornece os algoritmos de mineração de dados utilizados pela arquitetura de referência proposta para a realização do processo de descoberta de conhecimento. São disponibilizadas duas fontes de algoritmos de mineração. A primeira inclui os algoritmos implementados pela própria solução. A segunda inclui os algoritmos disponibilizados por um *framework* específico para mineração de dados educacionais baseado em serviços semânticos, denominado *SEDAM-R* (MARINHO et al., 2017), que também está integrado à arquitetura de referência proposta.

SEDAM-R *Framework*

Inicialmente proposto por Marinho et al. (2010), o ***SEDAM Framework*** passou por um processo de reengenharia, envolvendo o desenvolvimento e implementação de um novo pro-

jeto arquitetural. Entre as principais mudanças, destaca-se a adoção de um novo modelo semântico, que oferece suporte a serviços Rest e simplifica o processo de construção de serviços Web semânticos. Além disso, adotou-se uma abordagem centrada em dados a fim de gerenciar as descrições semânticas dos serviços e tornar o processo de descoberta, composição e invocação mais ágil e preciso. O novo framework, fruto da reengenharia do SEDAM, foi denominado SEDAM-R (MARINHO et al., 2017).

O SEDAM-R é responsável pela criação dos processos personalizados de descoberta do conhecimento utilizando algoritmos de mineração de dados. Por utilizar serviços semânticos, ele possibilita a configuração do processo sem que o usuário precise conhecer detalhes técnicos dos algoritmos utilizados; toda a configuração acontece de forma semi-automática, a partir dos objetivos indicados pelo usuário. As vantagens que justificam a adoção do SEDAM-R incluem: extensão facilitada para adição de novos serviços de mineração, integração entre sistemas simplificada, flexibilidade, entre outras. O componente *SEDAM-R Framework* é utilizado pelo componente *Assistant* durante a configuração dos processos de descoberta de conhecimento que podem ser definidos por professores e educadores.

Devido à complexidade deste componente, sua arquitetura interna foi abstraída na Figura 5.1 e é apresentado na Figura 5.4. A arquitetura do SEDAM-R adota o estilo arquitetural orientado a serviços, com influência do estilo arquitetural *pipes and filters*, onde a sequência de execução dos serviços é definida pelo componente *Service Manager*. Nesse modelo, cada algoritmo e/ou ferramenta de mineração, pré-processamento ou pós-processamento é encapsulada em um serviço Web.

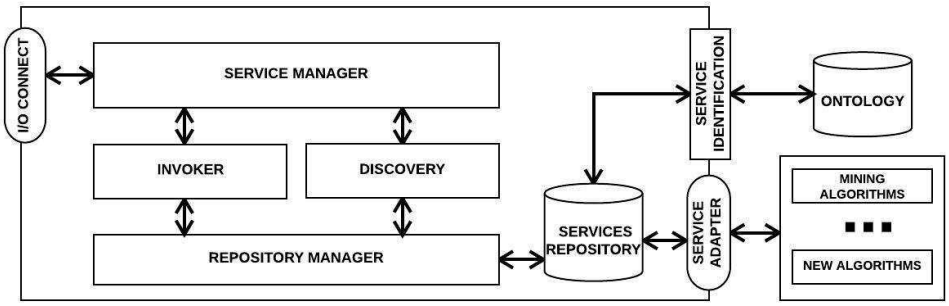


Figura 5.4: Arquitetura do *Framework* (SEDAM-R)
Fonte: Elaborada pelo autor

O SEDAM-R adota serviços semânticos (MCILRAITH; SON; ZENG, 2001) para possi-

bilitar que o processo de descoberta, composição e invocação dos serviços seja realizado de maneira automática e dinâmica. Para tal, o usuário deve informar os parâmetros de entrada e saída desejados, assim o sistema é capaz de realizar todo o processo automaticamente, incluindo a descoberta e execução da sequência correta dos serviços, seguindo o estilo arquitetural *pipes and filters*, onde a saída de um serviço é utilizada como entrada para a execução do serviço seguinte. Todo o processo de localização e orquestração dos serviços é realizado pelo componente **Service Manager**, que é um serviço Web tradicional e pode ser executado pelos demais componentes da arquitetura de referência apresentada na Figura 5.1. Além disso, a adoção de serviços semânticos possibilita que novos serviços possam ser adicionados sem a necessidade de nenhum tipo de adaptação ou reconfiguração tal como ocorre em padrões como o BPEL (ANDREWS et al., 2003). Isso facilita a adição de novos algoritmos de pré-processamento, de mineração de dados ou de pós-processamento.

O modelo semântico adotado para descrever os serviços Web foi proposto por Barros (2016). Nesse modelo, a ontologia que descreve o serviço é escrita em uma adaptação da linguagem *OWL-S*, visando a representação semântica de serviços Web implementados com o paradigma REST. Para a construção desta ontologia de serviços, um subconjunto de descrições OWL-S foi criado a fim de prover compatibilidade com essa linguagem e adicionar o suporte de características que não estão presentes na linguagem *OWL-S*. Na Figura 5.5 são apresentadas as entidades que representam parte da ontologia de serviço com seus respectivos atributos e relacionamentos. A classe de serviço representa os serviços Web. Essa classe tem como atributo um rótulo, descrição e relacionamentos. Para cada serviço, é definido um conjunto de parâmetros, que podem incluir entradas e saídas.

Nessa classe de serviços é definido um *schema* para os parâmetros de entrada e outro para os parâmetros de saída. Esses *schemas* definem o formato que os parâmetros de entrada, *input_schema*, devem ter antes de serem enviados aos serviços e como os parâmetros de saída, *output_schema*, dos serviços serão devolvidos. Esses *schemas* seguem uma definição baseada em JSON *Schema*. Com a finalidade de simplificar o processo de manipulação das descrições semânticas dos serviços, adotou-se uma arquitetura centrada em dados. Nesse formato, ao iniciar o sistema, todas as ontologias são carregadas, onde são extraídas as principais características dos serviços, bem como os *schemas* de entrada e saída. Em seguida, são criados os arquivos no formato JSON *schema* com suas respectivas descrições semânticas e todos os

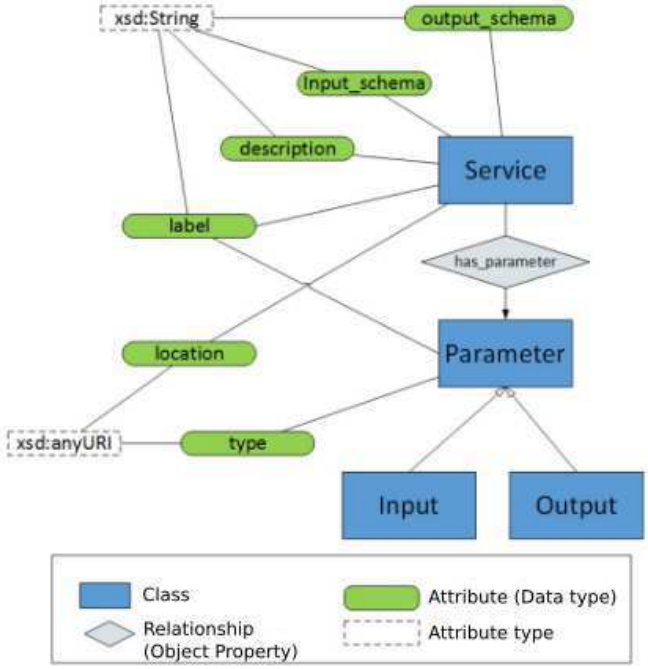


Figura 5.5: Representação gráfica da ontologia de serviços
Fonte: Extraída de (BARROS, 2016)

dados relacionados aos serviços são carregados em um base de dados *NoSQL*. Desse modo, é possível manipular os serviços e suas descrições semânticas sem a necessidade de acessar as ontologias diretamente, rotina que tornava o processo de descoberta e composição demasiadamente complexo e custoso. Assim sendo, é possível executar o processo de descoberta, composição e invocação de maneira mais simples, rápida e precisa.

Assistant

Direcionada a professores e educadores, a arquitetura proposta oferece uma interface simplificada e baseada em objetivos para o auxilio no processo de tomada de decisão pedagógica em ambientes educacionais. O *Assistant*, como é chamado, será um sistema Web que reúne importantes funcionalidades aos atores envolvidos nesse processo, tais como: i) prover uma interface simplificada para a realização do processo de descoberta de conhecimento por usuários não-especialistas; ii) oferecer uma interface para o gerenciamento de regras do *data warehouse* ativo, ou seja, possibilitar a automatização de ações; iii) oferecer a possibilidade de atuação diretamente dentro dos ambientes educacionais por meio de ações manuais, como: envio de mensagens, postagens nos fóruns, criação de fóruns, entre outras ações

possíveis dentro dos ambientes educacionais; entre outras.

O fluxo de execução do processo de descoberta de conhecimento apoiado pelo *Assistant* é apresentado na Figura 5.6, através de um diagrama de atividades. As atividades do diagrama representam a sequência de ações apoiadas pelo assistente e que são executadas em sequência. As atividades escuras são executadas automaticamente pelo assistente, enquanto as atividades claras necessitam de alguma intervenção do professor/educador.

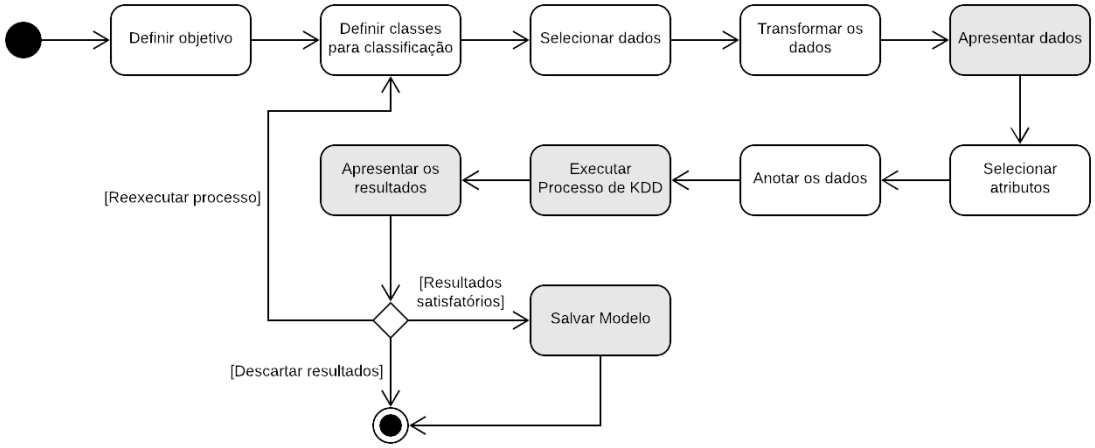


Figura 5.6: Processo para assistência à descoberta de conhecimento
Fonte: Elaborada pelo autor

No final do processo, dados com uma amostra das informações descobertas são exibidos aos professores/educadores, que decidirão se os resultados são satisfatórios. Se forem satisfatórios, professores/educadores podem utilizar o modelo de mineração criado como parte do condicional de novas regras. Caso contrário, eles podem encerrar e descartar os resultados ou executar o processo novamente.

A interface de gerenciamento de regras permite que professores e os demais *stakeholders* envolvidos possam gerenciar as regras do *Data warehouse* Ativo, seguindo os modelos de regras apresentados na Seção 5.3.2. Desse modo, professores e/ou educadores podem definir regras e automatizar ações a serem executadas diretamente dentro dos ambientes educacionais de maneira reativa. A definição de regras envolve também a definição de outros critérios, tais como: a quem se destina as regras, podendo ser um grupo, uma turma e/ou disciplina; a frequência de sumarização dos dados para esses conjunto de regras e qual o volume de mudanças necessárias para disparar a execução desse conjunto de regras.

Por fim, professores e educadores podem realizar a execução de ações manuais. Essas ações incluem as ações implementadas pela arquitetura proposta e que também podem ser automatizadas através das regras. A possibilidade de executar essas ações manualmente é importante, pois, possibilita uma atuação rápida dos atores envolvidos sem sair do ambiente de tomada de decisão ao detectar uma situação relevante.

Ferramentas de Tomada de Decisão

Professores e/ou educadores também terão acesso a ferramentas gerenciais tradicionais. Inicialmente, os atores envoltos podem realizar consultas complexas, visualização de dados e *dashboards*, gerar e ter acesso a relatórios personalizados, entre outras funcionalidades. O objetivo é agregar essas funcionalidades e oferecer ainda mais informações aos professores e educadores durante o processo de tomada de decisão pedagógica. Como ponto de partida, essas funcionalidades estarão disponíveis por meio da personalização da interface de ferramentas tradicionais e já disponíveis no mercado, como o Pentaho. Com isso, professores e educadores podem, por exemplo, gerar relatórios das suas disciplinas de acordo com indicadores das interações dos estudantes, como: número de acessos a plataforma, número de acesso aos fóruns, número de postagens, entre outras; visualizar esses indicadores em gráficos, os quais fornecem uma visão geral das interações dos estudantes; realizar consultas envolvendo diversos indicadores, como realizar uma consulta para visualizar o comportamento do discente ao longo do tempo nos fóruns da disciplina; entre outras ações. A arquitetura também está preparada para receber uma interface personalizada, que inclua *dashboards* próprios, bem como, outras funcionalidades, que podem ser integradas ao DW e/ou DW Ativo.

5.3.3 Visão de Processo

A visão de processo de uma arquitetura de software visa representar aspectos dinâmicos de alguns cenários arquiteturais que são considerados estratégicos para o domínio do sistema (KRUCHTEN, 1995). No contexto da arquitetura de referência proposta, é apresentado na Figura 5.7 um diagrama de sequência UML representando o comportamento arquitetural no momento em que regras com condições avançadas (com mineração de dados) são executadas.

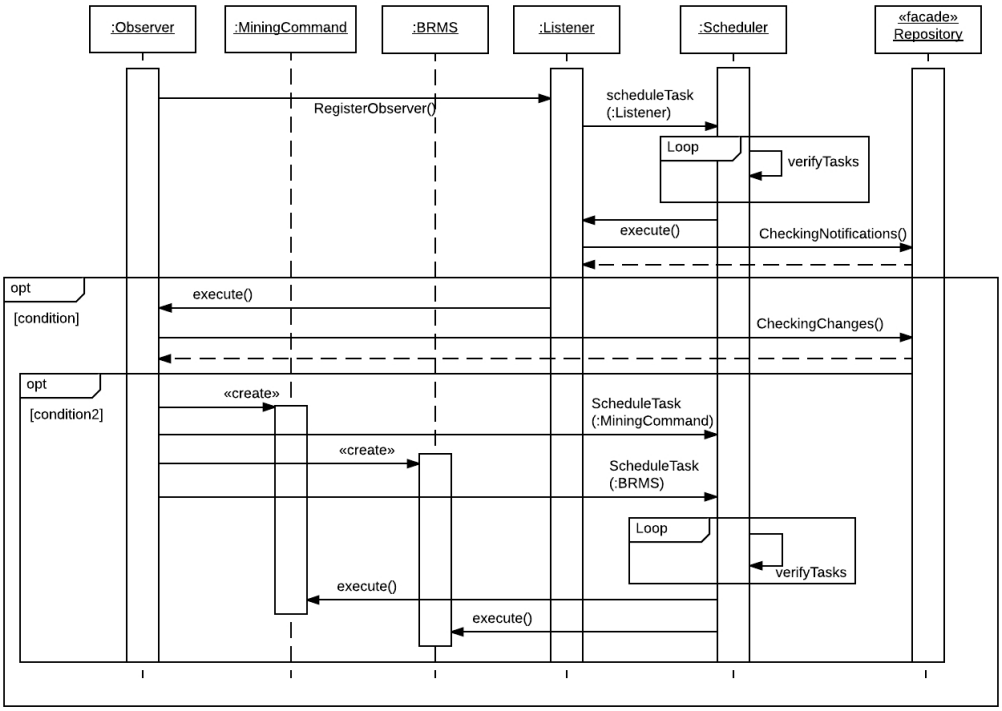


Figura 5.7: Diagrama de Sequência para execução de regras com condições avançadas
Fonte: Elaborada pelo autor

O mecanismo ativo do *data warehouse* é inicializado com instanciação das classes *Observer* e *Listener*. Seguindo o comportamento do padrão de projeto, o objeto *Observer* registra-se no *Listener* que, por sua vez, realiza o seu agendamento no *Scheduler*. O *Scheduler* realiza uma verificação periódica e executa os objetos agendados de acordo com a periodicidade determinada em cada agendamento. Assim, *Listener* verifica a tabela de notificações dentro do *Data warehouse*, que é atualizada através de uma *trigger* presente no *data warehouse*. Ao perceber que mudanças ocorreram no *data warehouse*, o *Listener* notifica todos os objetos *Observer* registrados. Cada *Observer*, por sua vez, reúne os conjuntos de regras armazenados, verifica os dados relacionados a cada conjunto de regras e consulta o *data warehouse* para averiguar se ocorreram alterações significativas nos conjuntos de dados correspondentes. Como cada conjunto de regras é relacionado a um conjunto de dados, seu monitoramento pode ser relacionado a um grupo de estudantes, disciplina e/ou a um determinado curso. Além disso, é estabelecido para cada conjunto de regras um volume de alterações necessária para o disparo das regras. Desta forma, o *Observer* verifica se os dados alterados correspondem àquele conjunto de regras e se a condição *volume de alterações* foi satisfeita. Se ambas

as condições foram satisfeitas, o *Observer* verifica se existem modelos de mineração associados ao conjunto de regras. Se a resposta for positiva, o *Observer* agenda a execução do modelo de mineração sobre os novos dados através do *MiningCommand*. Em seguida, cria um novo *BRMS* (*engine de regras*) e agenda sua execução de acordo com a frequência de sumarização de dados estabelecidas para aquele conjunto de regras. O *Scheduler*, mais uma vez, realiza suas verificações e, no momento agendado, executa o *MiningCommand*. Por fim, o *BRMS* recupera o conjunto de regras correspondente, os dados correspondentes, de acordo com a frequência de sumarização dos dados pré-estabelecida para este conjunto de regras, e executa as regras sobre esses dados, disparando as ações associadas às regras que tiveram suas condições satisfeitas.

5.3.4 Representação dos Pontos de Variação

A fim de representar os pontos de variação da arquitetura de referência que foram previstos em tempo de projeto, é apresentada na Figura 5.8 o seu modelo de características, de acordo com a notação apresentada por (BOSCH, 2001) e descrita na legenda.

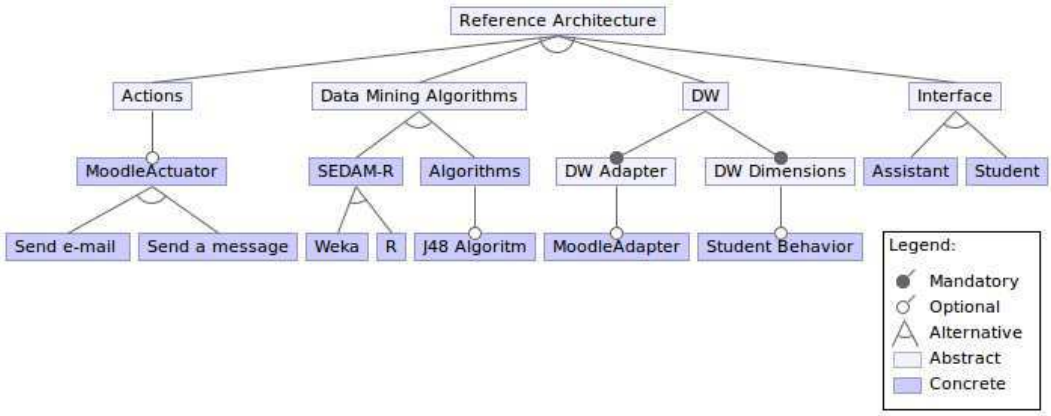


Figura 5.8: Modelo de Características Explicitando os Pontos de Variação da Arquitetura de Referência Proposta

O modelo é composto por 4 (quatro) características opcionais: *actions*, *data mining algorithms*, *DW*, *interface*. Cada característica opcional é composta por outras características, que podem ser alternativas ou obrigatórias. Portanto, após a definição das características opcionais que integrarão a instância da arquitetura concreta, uma ou mais características alternativas devem ser escolhidas. Para a característica *DW*, há mais duas características

obrigatórias: *DW adapter* - responsável pela integração e carga do *DW* definido, e a característica *DW dimensions* - que definirá quais dimensões e *tabelas fato* estarão disponíveis no *DW*.

As características opcionais e alternativas determinam os pontos de variação da arquitetura, isto é, as escolhas que o projetista tem à disposição durante a instanciação da arquitetura de referência em uma arquitetura de software específica.

Capítulo 6

Instanciação e Implementação da Arquitetura Concreta

Neste capítulo são apresentadas as visões complementares ao modelo 4+1, apresentado no Capítulo anterior. Dessa maneira, são apresentadas as visões de desenvolvimento (Seção 6.1), implantação (Seção 6.2) e a visão do usuário (Seção 6.3). Nesta tese adotou-se uma notação alternativa para o modelo do usuário. Ao invés de utilizar os tradicionais diagramas de casos de uso UML, optou-se por representar a visão do usuário utilizando protótipos de telas da interface gráfica. Tal alteração foi realizada por considerar os protótipos de tela mais representativos do ponto de vista do usuário do que diagramas de casos de uso UML.

Como pode ser observado na Seção 6.1, a implementação da arquitetura de referência foi realizada através de um *framework* que possui as variabilidades apresentadas na Seção 5.3.4. A linguagem de programação adotada foi a linguagem Java, seguindo a modularização de componentes apresentada na visão lógica da arquitetura (Seção 5.3.2, Figura 5.1).

6.1 Visão de Desenvolvimento e Implementação

A visão de desenvolvimento da arquitetura de referência é apresentada na Figura 6.1 usando um diagrama de componentes UML. Nas seções a seguir são apresentados os componentes arquiteturais, incluindo as decisões de projeto, padrões de projeto adotados durante o projeto orientado a objetos e os detalhes de implementação de cada componente.

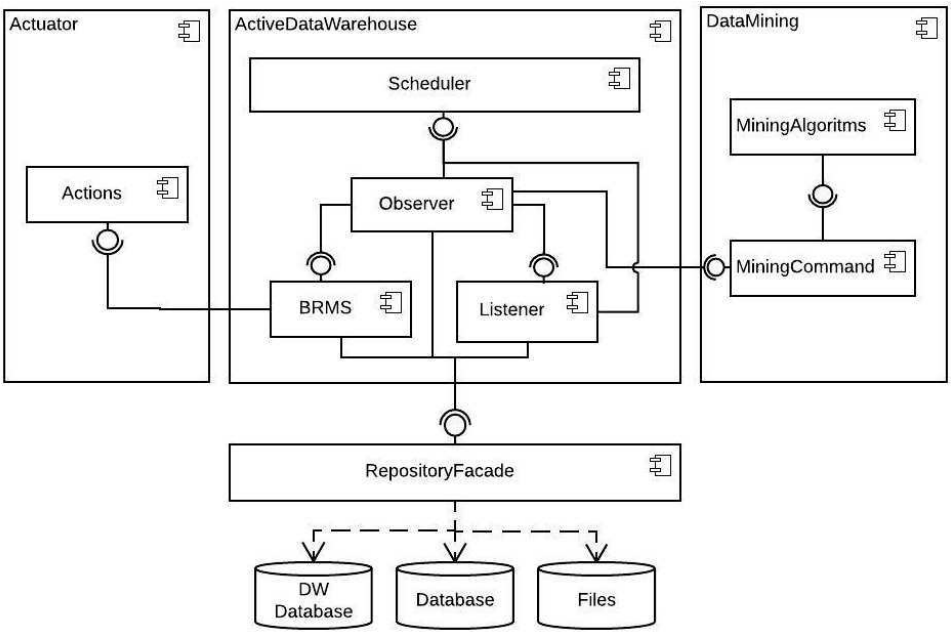


Figura 6.1: Diagrama de Componentes
Fonte: Elaborada pelo autor

6.1.1 *Repository Facade*

O componente repositório tem a função de gerenciar o acesso a três bases de dados diferentes: i) **DW Database** - é o *data warehouse* proposto e pode conter dados sumarizados provenientes de diferentes fontes educacionais, tais como: ambientes educacionais, sistemas acadêmicos, entre outros; ii) **Database** - base de dados utilizada para armazenar dados relacionados aos conjuntos de regras, critérios de monitoramento das regras, dados relacionados ao modelos de mineração, resultados dos modelos de mineração, entre outros; iii) **Files** - É a base de arquivos que mantém os *templates* de regras, conjunto de regras e modelos de mineração armazenados.

No projeto de implementação do repositório foi adotado o padrão Facade em conjunto com o padrão Singleton com o objetivo de oferecer uma interface única e simplificada a camada de acesso aos dados. Além disso, todas as classes de acesso aos dados são orientadas a interface, proporcionando um baixo acoplamento entre a fachada e as classes de acesso aos dados.

6.1.2 *Scheduler*

Parte integrante do *data warehouse* ativo, esse componente tem como função agendar, gerenciar e executar diversas tarefas, sendo parte fundamental da arquitetura proposta, pois permite gerenciar tarefas complexas que podem sobrecarregar os servidores envolvidos, como a aplicação baseada em regras e servidores de dados. Assim, a fim de evitar uma sobrecarga sobre esses servidores, as tarefas são agendadas para serem executadas quando a demanda sobre esses servidores for menor. Entre as tarefas que podem ser agendadas, estão: verificação de alterações nos dados no *data warehouse*, execução de modelos mineração sobre novos dados, execução dos conjuntos de regras, entre outras.

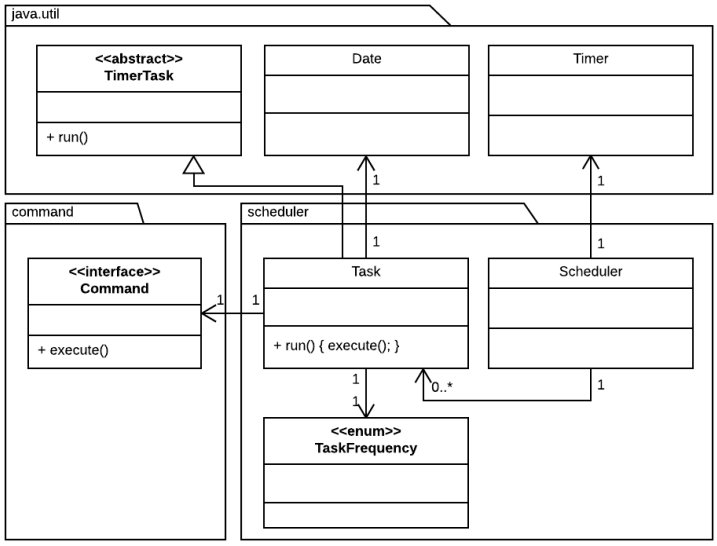


Figura 6.2: Diagrama de Classes - Componente *Scheduler*
Fonte: Elaborada pelo autor

Na implementação deste componente foram utilizados as classes *Timer* e *TimerTask* do Java para implementação das classes *Scheduler* e *Task*, respectivamente. O padrão Command foi adotado na implementação de diversas classes da solução com o objetivo de possibilitar que a classe *Scheduler*, responsável pelos agendamentos e suas execuções, possa executar quaisquer objetos independente do seu tipo, além de promover o baixo acoplamento entre as classes.

Código Fonte 6.1: Parte da Classe *Scheduler*

```
1 public class Scheduler {
```

```
2
3       private static Scheduler scheduler;
4       private Timer timer;
5       private List<Task> tasks;
6
7       public static Scheduler getInstance() {
8           if (scheduler == null) {
9               scheduler = new Scheduler();
10          }
11          return scheduler;
12      }
13
14      private Scheduler() {
15          timer = new Timer();
16          tasks = new ArrayList<>();
17      }
18
19      public void scheduleTask(Task task) {
20          if (!tasks.contains(task)) {
21              tasks.add(task);
22              includeTaskOnScheduler(task);
23          }
24      }
25
26      private void includeTaskOnScheduler(Task task) {
27          switch (task.getFrequency()) {
28              case MINUTELY:
29                  timer.schedule(task, getDate(), TimeUnit.MINUTES.toMillis
                        (1));
30                  break;
31              case HOURLY:
32                  timer.schedule(task, getDate(), TimeUnit.HOURS.toMillis
                        (1));
33                  break;
34              case DAILY:
35                  timer.schedule(task, getDate(), TimeUnit.DAYS.toMillis(1)
                        );
```

```
36                  break;
37              case WEEKLY:
38                  timer.schedule(task, getDate(), TimeUnit.DAYS.toMillis(7)
                        );
39                  break;
40              case MONTHLY:
41                  timer.schedule(task, getDate(), TimeUnit.DAYS.toMillis
                        (30));
42                  break;
43              case END_OF_WEEK:
44                  timer.schedule(task, getLastDayoftheWeek());
45                  break;
46              case END_OF_MONTH:
47                  timer.schedule(task, getLastDayoftheMonth());
48                  break;
49              case SPECIFIC_DATE:
50                  timer.schedule(task, task.getDate());
51                  break;
52          }
53      }
```

6.1.3 *Data Warehouse*

Como ponto de partida, é proposto um *data warehouse* (DW) de referência baseado no ambiente *Moodle* (MOODLE, 2017). Como mencionado anteriormente, o *Moodle* foi escolhido pois se destaca entre as plataformas mais utilizadas no apoio ao ensino à distancia, sendo utilizado em mais de 200 países, com aproximadamente 10 milhões de cursos e 88 milhões de usuários[1], incluindo grande parte das instituições envolvidas no programa brasileiro Universidade Aberta do Brasil (UAB). O modelo de referência construído tem como principal objetivo avaliar e acompanhar o comportamento do estudante dentro do ambiente educacional e engloba a maior parte das ferramentas utilizadas pelos dicentes dentro do *Moodle*, tais como: fórum, envio de tarefas, blog, wiki, entre outras.

O *data warehouse* (DW) adota um modelo estrela, onde a *tabela fato* representa o com-

[1]Dados retirados de https://moodle.net/stats/

portamento do estudante dentro da plataforma e as medidas sumarizam suas interações com as principais ferramentas do *Moodle*. O modelo proposto possui 5 dimensões: Estudante, Grupo, Curso, Disciplina e Tempo. Através desse modelo, que reúne dados das interações dos estudantes com as principais ferramentas da plataforma, é possível acompanhar e avaliar o comportamento dos estudantes dentro do ambiente educacional.

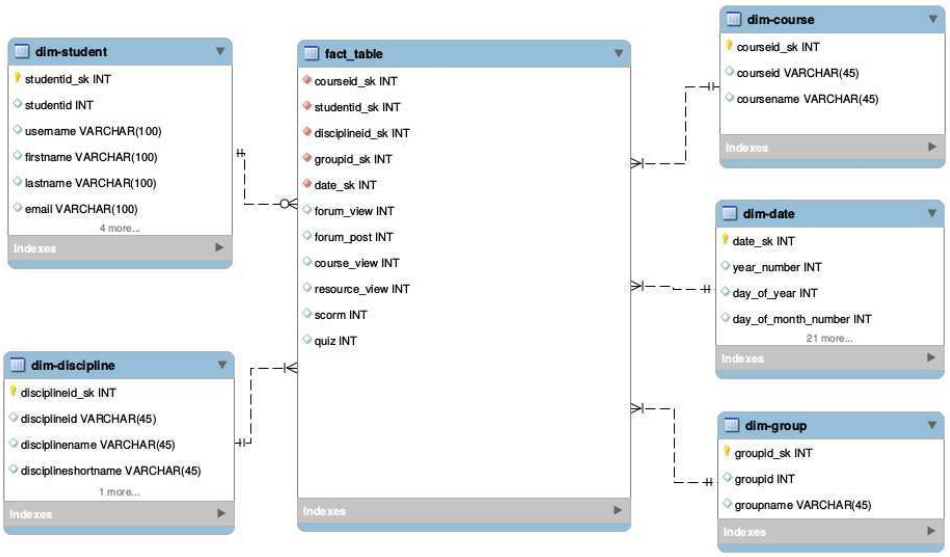


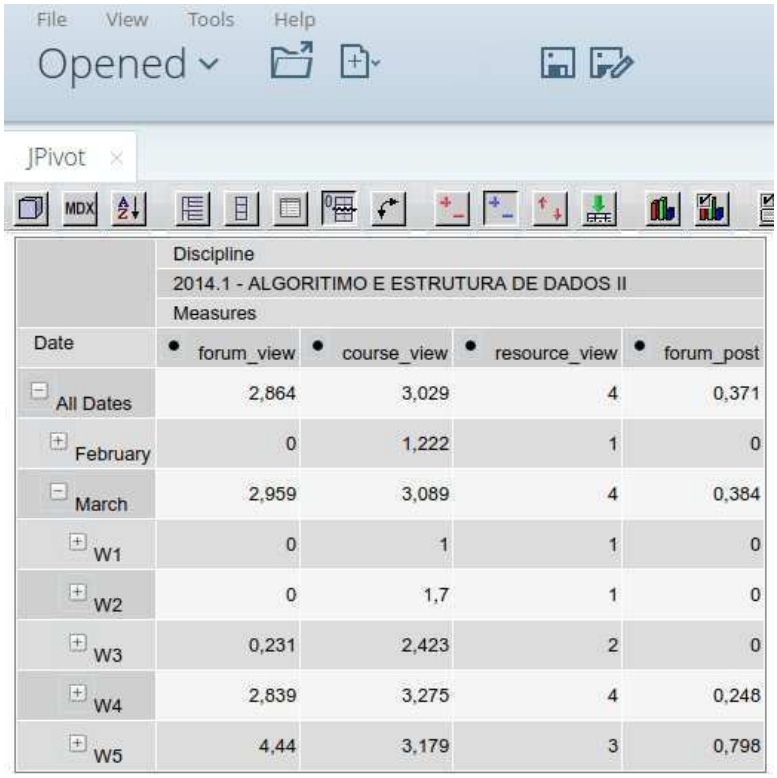
Figura 6.3: Modelo Entidade-Relacionamento do *Data Warehouse*
Fonte: Elaborada pelo autor

Na construção do DW foram utilizados dados reais obtidos a partir do Moodle. O processo de ETL foi realizado através da ferramenta Pentaho (PENTAHO, 2016). Após a carga inicial, a *tabela fato* possui aproximadamente de 167000 tuplas. Entre as dimensões, a dimensão *student* possui 2134 tuplas; a dimensão *groups*, 643 tuplas; a dimensão *discipline*, 180 tuplas; a dimensão *course*, 12 tuplas; e a dimensão *date*, 40000 tuplas.

A partir do DW construído, foi desenvolvido um cubo OLAP, apresentado na Figura 6.4, e a definição para a utilização da ferramenta de consultas complexas, apresentada na Figura 6.5. Foram definidas através do Pentaho com o objetivo de integrar essas ferramentas gerenciais à plataforma.

6.1.4 *Active Data Warehouse*

O componente ativo do *Data Warehouse* é responsável por fornecer a capacidade reativa para a arquitetura de referência proposta. Este componente é composto por duas partes: módulo



	Discipline			
	2014.1 - ALGORITIMO E ESTRUTURA DE DADOS II			
	Measures			
Date	forum_view	course_view	resource_view	forum_post
All Dates	2,864	3,029	4	0,371
February	0	1,222	1	0
March	2,959	3,089	4	0,384
W1	0	1	1	0
W2	0	1,7	1	0
W3	0,231	2,423	2	0
W4	2,839	3,275	4	0,248
W5	4,44	3,179	3	0,798

Figura 6.4: Cubo OLAP
Fonte: Elaborada pelo autor

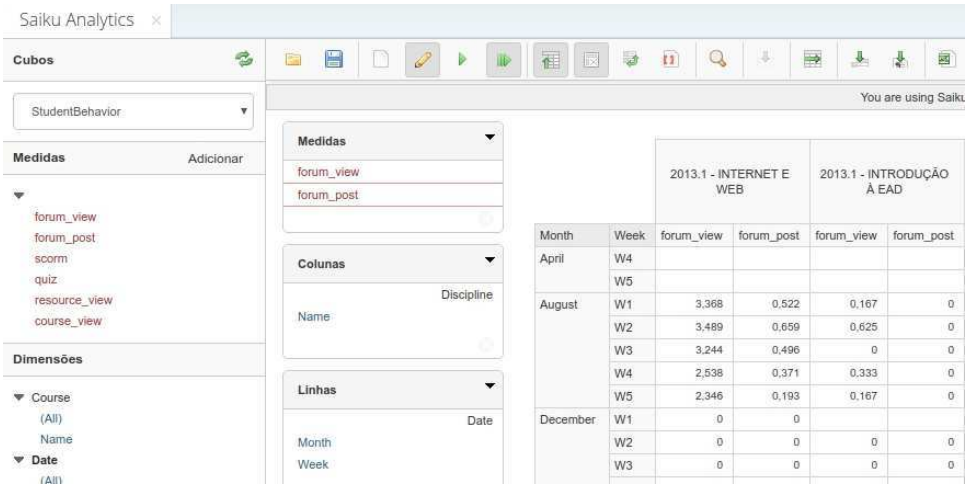


Figura 6.5: Ferramenta de Consultas Complexas
Fonte: Elaborada pelo autor

de notificação e módulo de monitoramento.

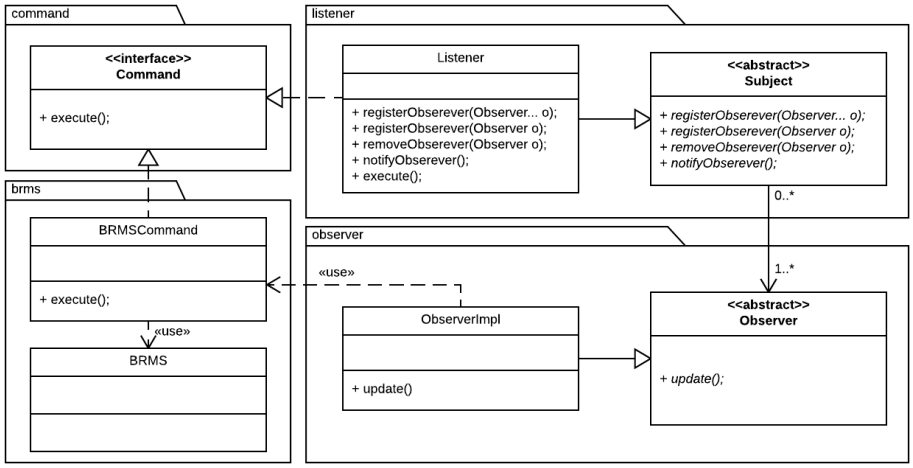


Figura 6.6: Diagrama de Classes - Componente Ativo do *Data warehouse*
Fonte: Elaborada pelo autor

O *módulo de notificação* consiste em um mecanismo ativo integrado com o *data warehouse*, que inclui uma tabela de notificação e um gatilho de banco de dados. Este mecanismo é criado automaticamente pela arquitetura de referência proposta no momento da sua inicialização, caso ainda não exista. A tabela de notificação é composta por um *ID* e a data em que o gatilho de notificação foi disparado, bem como o próprio gatilho, que é composto pela função que realiza uma nova inserção nessa tabela no caso de novas cargas no *data warehouse*.

O *módulo de monitoramento* é responsável por analisar periodicamente a tabela de notificação, verificando os tipos de alterações ocorridas nos dados. Ao analisar essas alterações, ele verificará se aconteceram alterações suficientes que justifique a execução das regras registradas sobre os novos dados. Tais regras podem ainda estar associadas a modelos de mineração, o que desencadearia a execução desses modelos antes da execução das regras de fato, como explicitado na Seção 5.3.3. Essas responsabilidades são divididas entre as classes *Subject* e *Observer*. Desse modo, enquanto uma instância de *Listener* monitora periodicamente a tabela de notificações e notifica os *Observadores* associados, a classe *Observer* assume o restante das responsabilidades, seguindo o padrão de projeto Observer. Vale ressaltar que cada *Observer* representa uma unidade de execução. Essa unidade de execução pode representar uma disciplina, um curso ou até um ambiente educacional, dependendo das

necessidades e requisitos de cada arquitetura concreta instanciada. A unidade de execução estará associada a um ou mais conjuntos de regras e sua separação deve ser motivada, principalmente, pela quantidade de conjuntos de regras que ele precise gerenciar.

Com o objetivo de gerenciar e executar as regras, foi adotada a *engine* de regras *Drools* (BROWNE, 2009) através de sua API. O Drools oferece uma API que permite o gerenciamento de regras, definidas através da DRL (*Drools Rule Language*), e construção de uma *engine* de regras baseado em inferência. Baseado na tríade evento-condição-ação, o *Drools* possibilita a execução de ações, podendo essas ações serem chamadas externas a métodos escritos em Java. No entanto, o processo de gerenciamento dessas regras é complexo e dispendioso, pois no *Drools*, uma vez construída a *engine* de regras, não é possível alterar as regras definidas. No contexto desse trabalho, é fundamental que as regras sejam criadas e/ou editadas em tempo de execução. Desse modo, foi necessário um conjunto de classes a fim de facilitar e permitir a manipulação de regras no formato *DRL*, linguagem para definição de regras do próprio *Drools*, para tornar possível a manipulação dessas regras em tempo de execução.

Nesse âmbito, foram implementadas as classes *BRMS* e *BRMSCommand*, apresentada na Listagem 6.2, que adota a API do *Drools* (BROWNE, 2009) para: i) manipular e criar os conjuntos de regras através da *Drools Rule Language* (DRL); ii) construir a *engine* de regras; e iii) executar os conjuntos de regras sobre os dados. Com isso, foi possível desenvolver um mecanismo reativo que verifica as mudanças ocorridas no *data warehouse*, gerencia e executa os modelos de mineração, avalia e executa as regras definidas por professores e educadores sobre os dados e, se necessário, executa as ações pré-definidas de forma automatizada diretamente dentro dos ambientes educacionais.

Código Fonte 6.2: Classe *BRMSCommand*

```
1 import br.framework.command.Command;
2 import br.framework.configuration.FilePath;
3 import br.framework.domain.RuleFile;
4 import br.framework.domain.Student;
5 import br.framework.infrastructure.facade.RepositoryFacade;
6 import br.framework.activedw.util.RuleUtil;
7 import java.util.HashMap;
8 import java.util.List;
```

```
9
10 public class BRMSCommand implements Command {
11
12     private RuleFile ruleFile;
13
14     public BRMSCommand(RuleFile ruleFile) {
15         this.ruleFile = ruleFile;
16     }
17
18     @Override
19     public void execute() {
20         try {
21             /* Obter DRL */
22             String drl = RuleUtil.buildDrlFileFromTemplate(FilePath.
                   RULE_TEMPLATE, ruleFile.getRules());
23             BRMS brms = BRMS.createFromDrl(drl);
24
25             /* Obter os dados  */
26             List<Student> data = RepositoryFacade.getInstance().
                   getFactStudentDataForFireRules(ruleFile);
27
28             /* Add globals  */
29             HashMap<String, Object> globals = new HashMap<>();
30             globals.put("command", ruleFile.getGLOBAL());
31
32             /* Execute */
33             brms.execute(data, globals);
34
35         } catch (Exception e) {
36             e.printStackTrace();
37         }
38     }
39
40 }
```

No desenvolvimento desse mecanismo, foram adotados dois padrões de projeto (GAMMA, 1995): o padrão Observer foi adotado para promover a característica em tempo

real da reação do sistema (parte ativa). O padrão Command foi adotado para garantir que as instâncias da classe *Listener* possam ser gerenciadas e executadas pelo componente *Scheduler*.

Para permitir que o mecanismo de regras do *Drools* processasse regras com condições avançadas, um novo componente foi desenvolvido para realizar esse processo. Este componente consiste em um módulo para a execução automática de modelos de mineração. Para realizar esse processo, o componente usa os modelos de mineração salvos, bem como, uma tabela de classificação para cada modelo de mineração existente. A tabela de classificação contém os dados dos estudantes e os rótulos de classificação recebidos; Por exemplo, se o professor criar uma regra que deve enviar uma mensagem para um aluno que tenha sua condição classificada pelo modelo de mineração como *desmotivada*, o sistema deve executar as seguintes etapas: recuperar o modelo de mineração, executar o modelo sobre os novos dados, armazenar a classificação atribuída para cada estudante na tabela de classificação correspondente e utilizar o rótulo armazenado nesta tabela como condição da regra. É importante enfatizar que todo esse processo ocorre de forma transparente e automática.

6.1.5 *Actuator*

O componente atuador tem a função de executar as ações definidas nas regras dentro dos ambientes educacionais. Com o intuito de separar responsabilidades e garantir um atuador focado no domínio ou em um ambiente educacional, foram adotados os seguintes padrões de projeto: Factory Method e Command.

A razão para adotar esses padrões foi a necessidade de executar diferentes ações em diferentes ambientes, mas sem conhecer a priori quais ações poderiam ser realizadas nem em qual ambiente elas seriam executadas. Desse modo, através do Factory Method, é possível instanciar o objeto que executa uma determinada ação sem conhecê-lo previamente, tornando a solução mais flexível. Além disso, com o objetivo de estabelecer um ponto de acesso único para executar as ações, o padrão *Command* foi adotado. Esse padrão fornece uma interface única de execução, o que possibilita a execução das ações sem conhecer os tipos instanciados. Assim, é necessário apenas um objeto global, *CommandExecutor*, que sabe como executar todas as ações dentro de um determinado arquivo de regras. Além do que, é necessário instanciar apenas um objeto para todo o conjunto de regra ao invés de instanciar um objeto

de ação para cada regra, ganhando em desempenho no momento da execução da *engine* de regras do *Drools*.

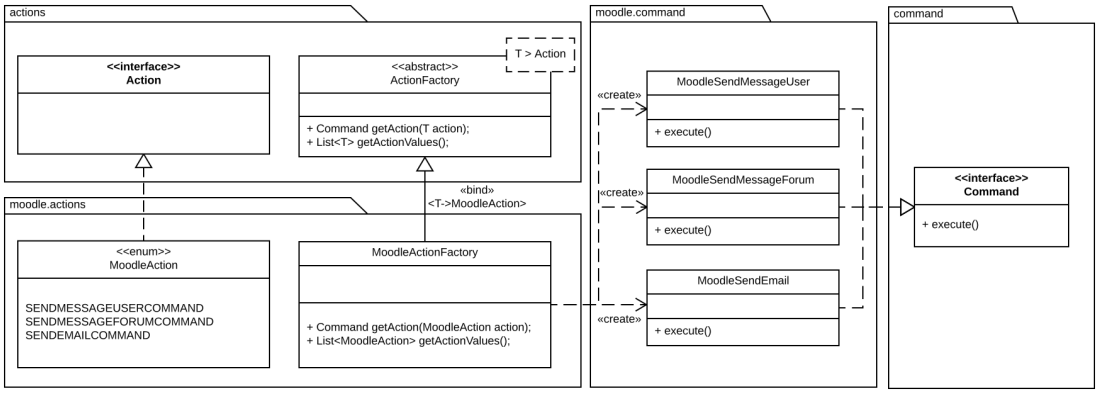


Figura 6.7: Diagrama de Classes - Componente Atuador
Fonte: Elaborada pelo autor

Este componente tem como classes principais a classe abstrata *ActionFactory* e interface *Action*. A classe *ActionFactory* adota o padrão Factory Method a fim de possibilitar a criação de diferentes fábricas de ações, ou seja, uma fábrica para cada ambiente educacional. A interface *Action* representa as ações que podem ser executadas por um determinado ambiente.

ActionFactory é uma classe genérica tendo um *generic type* T que implementa a interface *Action*. Esta decisão de projeto foi tomada para garantir que cada *ActionFactory* criada esteja associada a uma classe *Enum* de ações suportadas para essa fábrica. Assim, é possível garantir que uma fábrica só receberá parâmetros com ações válidas no momento de sua invocação. Por exemplo, a classe *MoodleActionFactory*, apresentada Listagem 6.3, representa a fábrica de ações para o ambiente Moodle e a classe *Enum MoodleAction* as ações suportadas pelo Moodle. Dessa forma, é possível restringir a criação de ações que são suportadas e estão configuradas para o ambiente Moodle.

Código Fonte 6.3: Classe *MoodleActionFactory*

```
1 import br.framework.actuator.actions.ActionFactory;
2 import br.framework.command.Command;
3 import br.framework.actuator.moodle.command.MoodleSendMessageUser;
4 import br.framework.actuator.moodle.command.MoodleSendMessageForum;
5 import java.util.Arrays;
6 import java.util.List;
```

```
7
8  public class MoodleActionFactory extends ActionFactory<MoodleAction> {
9
10     public MoodleActionFactory(Object... arguments) {
11         super(arguments);
12     }
13
14     public MoodleActionFactory() {
15         super();
16     }
17
18     @Override
19     public Command getAction(MoodleAction action) {
20         switch (action) {
21             case SENDMESSAGEUSERCOMMAND:
22                 return new MoodleSendMessageUser(super.getArguments());
23
24             case SENDMESSAGEFORUMCOMMAND:
25                 return new MoodleSendMessageForum(super.getArguments());
26         }
27
28         return null;
29     }
30
31     @Override
32     public List<MoodleAction> getActionValues() {
33         return Arrays.asList(MoodleAction.values());
34     }
35 }
```

Para incluir novas ações, para um novo ambiente educacional, são necessários implementar 3 novas classes: i) Uma *Enum* com as possíveis ações de um determinado ambiente que implementa da interface *Action*; ii) Uma *Factory*, que estende de *ActionFactory*, responsável por criar ações para aquele ambiente; e iii) A classe que executará a ação de fato. Assim, essa classe deve implementar a classe *Command* e sobrecarregar o método *Execute()*. Para executar uma ação em um ambiente que ofereça serviços Rest, por exemplo, o desenvolve-

dor deve, neste método, implementar um cliente para consumir esse serviço. No entanto, o desenvolvedor pode, nesse método, realizar invocação quaisquer outra API ou serviço Web, promovendo a flexibilidade de executar diversas ações diferentes.

6.1.6 Componente de *Data Mining*

Este componente é composto por serviços de mineração de dados providos pela solução. Esses serviços podem ser implementados com algoritmos de mineração de ferramentas como o Weka (WITTEN et al., 2016) ou integrados à arquitetura de referência proposta através do *framework* de mineração de dados baseado em serviços Web semânticos, chamado SEDAM-R (MARINHO et al., 2017). O componente foi divido em duas partes: o módulo de algoritmos de mineração e o módulo de execução de mineração.

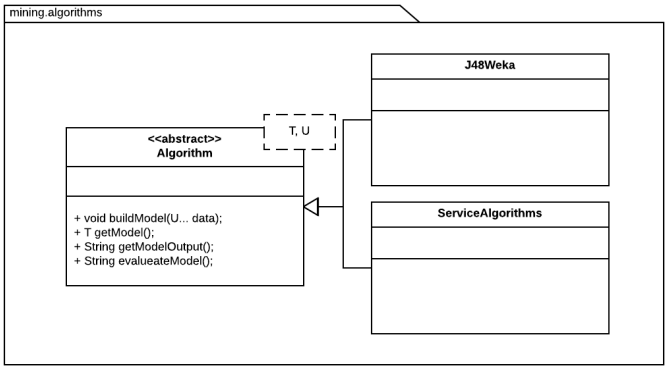


Figura 6.8: Diagrama de Classes - Algoritmos de mineração
Fonte: Elaborada pelo autor

O *módulo de algoritmos de mineração* reúne os algoritmos implementados nessa abordagem com a finalidade de construir e avaliar modelos de mineração. Para tal, os algoritmos implementados devem estender a classe abstrata *Algorithm*, que é uma classe genérica. Os parâmetros de tipos estabelecidos para esta classe definem o tipo do modelo do algoritmo de mineração e o tipo de entrada de dados que cada algoritmo recebe, possibilitando um código genérico e que elimina a necessidade de realizar conversões de tipo, *Casting*. Outra maneira de adicionar novos algoritmos ao *framework* proposto é através da integração com o *SEDAM-R framework* (MARINHO et al., 2017), conforme apresentado na Seção 5.3.2. A integração acontece através da classe *ServiceAlgorithms*, que também estende de *Algorithm*,

e, adicionalmente, fornece a infraestrutura necessária para acessar os serviços disponibilizados pelo SEDAM-R. Nesse módulo foi implementado o algoritmo J48 através da extensão da classe *Algorithm*. Além disso, foi realizada a integração com o SEDAM-R, que tem entre seus serviços diversos algoritmos de mineração. Desse modo, é possível executar algoritmos de mineração tanto através da arquitetura quanto através do SEDAM-R.

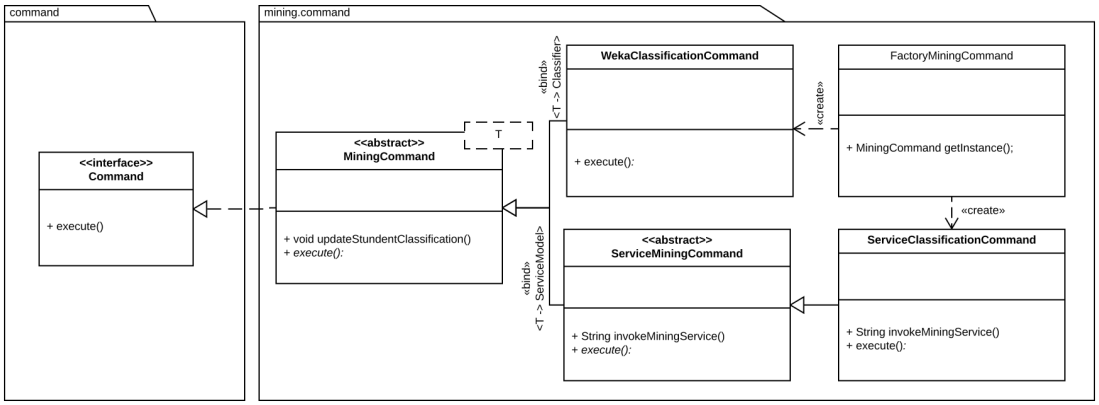


Figura 6.9: Diagrama de Classes - Módulo de execução
Fonte: Elaborada pelo autor

O *módulo de execução de mineração* é responsável pela execução dos modelos mineração, construídos em etapas anteriores, sobre novos conjuntos de dados. O componente precisa recuperar o modelo de mineração armazenado e executá-lo sobre os novos dados. Contudo, o tipo do modelo de mineração recuperado é desconhecido e, consequentemente, não se sabe como executá-lo. Assim, é necessário conhecer o modelo de mineração previamente para executá-los de maneira apropriada. Nesse módulo foram adotados dois padrões de projeto, Factory Method e o Command.

O padrão Command foi adotado para definir uma interface única de execução que encapsula as especificidades de execução de cada família de modelos de mineração, mas que possuem uma mesma interface pública de execução. Por esta razão a classe *MiningCommand* foi criada. Essa classe representa de maneira abstrata a família de classes destinadas a realizar a execução de modelos de mineração sobre novos dados. Para executar famílias de modelos de mineração mais específicos, foram implementadas as seguintes subclasses: i) *WekaClassificationCommand* - realiza a execução da família de modelos construídos a partir dos algoritmos de classificação do Weka, apresentada na Listagem 6.4; e ii) *ServiceMiningCommand* - realiza a execução da família de modelos construídos a partir de algoritmos do

SEDAM-R. No entanto, ao se recuperar um modelo salvo, não se conhece o seu tipo, pois os modelos são gerenciados de maneira genérica pela arquitetura proposta. Essa decisão foi tomada com o objetivo de tornar a arquitetura proposta mais flexível e permitir a manipulação de quaisquer modelos de mineração independente do seu tipo.

Código Fonte 6.4: Classe *WekaClassificationCommand*

```
1  import br.framework.domain.MiningModel;
2  import br.framework.domain.MonitoringCriteria;
3  import br.framework.domain.Student;
4  import br.framework.infrastructure.facade.RepositoryFacade;
5  import br.framework.mining.util.MiningUtil;
6  import java.util.List;
7  import java.util.Map;
8  import weka.classifiers.Classifier;
9  import weka.core.Instance;
10 import weka.core.Instances;
11
12 public class WekaClassificationCommand extends MiningCommand<Classifier>
       {
13
14     public WekaClassificationCommand(MiningModel<Classifier> miningModel,
           Map<MonitoringCriteria, String> monitoringCriteria) {
15         super(miningModel, monitoringCriteria);
16     }
17
18     @Override
19     public void execute() {
20         try {
21
22             /* Obter os dados para classificacao */
23             List<Student> students = MiningUtil.
                   retrieveDataForClassification(miningModel,
                   monitoringCriteria);
24
25             /* Preparar dados */
26             Instances instances = MiningUtil.prepareDataForClassification
                   (miningModel, students);
```

```
27
28              /* Executar classificacao */
29              for (Instance instance : instances) {
30                  int label = (int) miningModel.getModel().classifyInstance
                        (instance);
31                  String labelName = (String) miningModel.getLabels().get(
                        label);
32
33                  for (Student student : students) {
34                      if (MiningUtil.compareStudentInstance(student,
                            instance)) {
35                          student.addCharacteristic(miningModel.
                                getTableName(), labelName);
36                      }
37                  }
38
39              }
40
41              /* Atualizar classificacao dos Estudantes */
42              RepositoryFacade.getInstance().
                    saveOrUpdateStudentClassifications(students, miningModel.
                    getTableName());
43
44          } catch (Exception e) {
45              e.printStackTrace();
46          }
47      }
48
49  }
```

Isto posto, é preciso conhecer o tipo do modelo de mineração recuperado e atribuí-lo a subclasse de *MiningCommand* correspondente. Para solucionar esse problema, foi adotado o padrão Factory Method. A *Factory* desenvolvida recebe o modelo de mineração e os parâmetros sobre os dados que devem ser utilizados na execução. Dessa maneira, de acordo com tipo de modelo de mineração recebido, um objeto de uma das subclasses correspondente de *MiningCommand* é instanciado. Com isso, é possível executar o modelo sem conhecer

previamente o seu tipo, ou seja, é possível estender a arquitetura de referência proposta para criar, armazenar e executar quaisquer tipos de modelos de mineração implementados, mesmo àqueles que não foram previstos inicialmente.

6.2 Visão de Implantação

A visão de implantação da arquitetura de referência proposta tem a finalidade de detalhar a infraestrutura envolvida no seu processo de implantação, uma vez que a tomada de decisões no domínio educacional tem características que podem afetar atributos de qualidade importantes. As principais características consideradas são: a tomada de decisão envolve um grande volume de dados e uma alta demanda de processamento para realização da análise de dados e o processo de descoberta de conhecimento. Assim, a fim de representar a visão de implantação, é apresentado na Figura 6.10 o diagrama de implantação utilizando a notação UML.

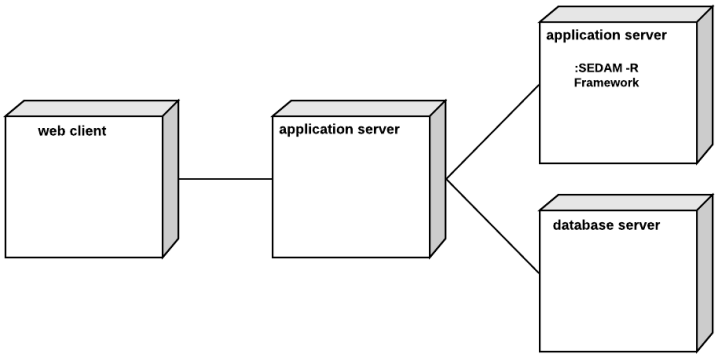


Figura 6.10: Diagrama de Implantação
Fonte: Elaborada pelo autor

Na visão representada na Figura 6.10, o *core* da aplicação pode ser implantado no Servidor de aplicação principal, que realiza o gerenciamento e comunicação entre componentes e servidores envolvidos. O *SEDAM-R framework* pode ser implantado em um servidor de aplicação secundário por realizar o processo de descoberta, composição e invocação dos serviços de mineração de dados, ou seja, envolve todo o processo de descoberta de conhecimento, o que demanda poder de processamento dos servidores dada a grande quantidade de dados envolvida. As base de dados podem ser implantadas em um servidor dedicado, pois

têm o potencial de envolver um grande volume de dados e uma alta demanda de acesso a esses dados para realização do processo de descoberta de conhecimento e análise dos dados. Por fim, a interface cliente implantada acessa serviços oferecidos pela aplicação através do servidor de aplicação principal.

6.3 Visão do Usuário

Nesta seção é apresentada a visão do usuário da arquitetura de referência proposta através de uma notação alternativa, onde protótipos de telas da interface foram considerados para representar o ponto de vista do usuário em substituição aos diagramas UML de casos de uso. Assim, são expostos abaixo algumas interfaces do componente *Assistente*.

Na Figura 6.11 é apresentada a tela relativa à funcionalidade de configuração do conjunto de regras. Como pode ser observado, a configuração para cada conjunto de regra envolve a definição da frequência dos dados, que representa a frequência de sumarização dos mesmos que serão apresentados aos *stakeholders*, e a frequência de aplicação do conjunto de regras sobre esses dados, nesse caso, mensalmente. Para cada frequência de aplicação escolhida, há um dia específico definido no sistema. Por exemplo, para frequência de aplicação mensal, as regras serão aplicadas no último dia de cada mês. Para frequência semanal, o último dia da semana. A opção volume de mudanças representa o volume mínimo de alterações nos dados necessário para que as regras sejam disparadas, dessa maneira, o usuário pode definir que um determinado conjunto de regras só será aplicado se grandes mudanças ocorrerem nos dados. Por fim, são definidos sobre quais dados o conjunto de regras será aplicado. Com isso, o usuário pode definir o curso, a disciplina e quais grupos/turmas serão monitoradas para aplicação dessas regras.

Após definição das configurações gerais do conjunto de regras, Figura 6.11, é apresentada uma outra interface na qual os usuários podem escolher quais modelos de mineração eles desejam utilizar para definir novas regras. Em seguida, é apresentada a interface, representada na Figura 6.12, que auxilia os atores envolvidos na definição e gerenciamento das regras. Essa interface permite que o usuário defina uma nova regra, configurando o nome da regra, as condições, onde o usuário pode manipular condicionais que formam a condição geral da regra. Por último, o usuário pode definir quais serão as ações dessa regra específica,

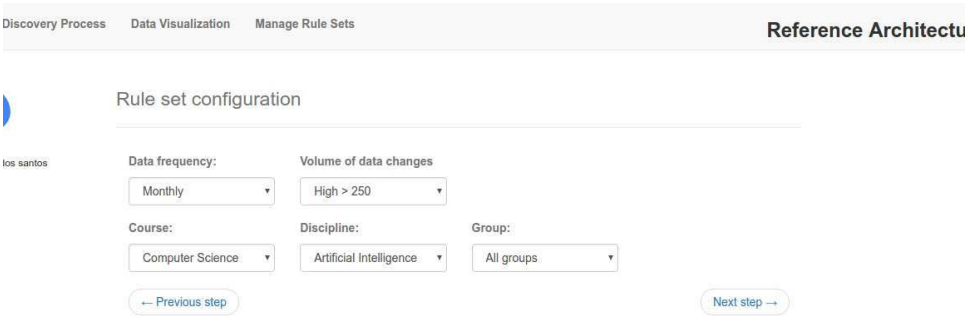


Figura 6.11: Configuração do Conjunto de Regras
Fonte: Elaborada pelo autor

onde ele pode incluir mais de uma ação por regra, se desejar. A construção da regra ainda passa por uma fase de finalização, como a inclusão da mensagem que será enviada para o estudante.

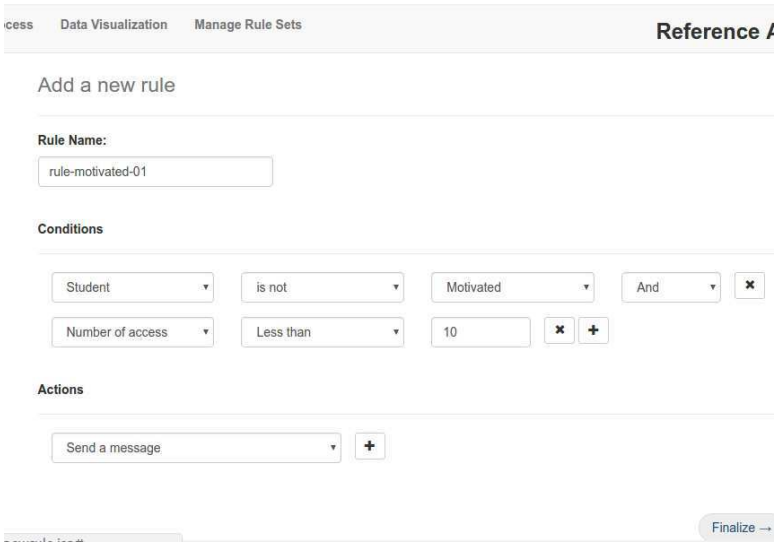


Figura 6.12: Definição de uma nova regra
Fonte: Elaborada pelo autor

Capítulo 7

Avaliação e Refinamento Arquitetural

A infraestrutura para a tomada de decisão em ambientes educacionais proposta nesta tese tem como um de seus pilares a arquitetura de referência apresentada nos Capítulos 5 e 6. Como parte do processo metodológico para aumentar a qualidade da arquitetura, buscou-se avaliar e refinar a arquitetura com critérios práticos qualitativos e quantitativos. Para isso, foi definido um método sistemático, apresentado na Seção 4.2, cuja execução é apresentada no presente capítulo. Para realizar essa avaliação, o método proposto faz uso do *Architecture Tradeoff Analysis Method* (ATAM), que avalia a arquitetura de software à luz dos atributos de qualidade desejados (KAZMAN; KLEIN; CLEMENTS, 2000). De forma geral, os refinamentos realizados tiveram o objetivo de aprimorar o cumprimento dos atributos de qualidade e gerar evidências de que as arquiteturas concretas instanciadas a partir da arquitetura de referência proposta atendam às necessidades e demandas em cenários reais de utilização. As seções a seguir detalham como foram executadas as etapas de avaliação da arquitetura proposta, bem como, o respectivo processo de refinamento.

Baseado no ATAM (KAZMAN; KLEIN; CLEMENTS, 2000), a primeira etapa da avaliação consistiu em recapitular a arquitetura de referência inicial e em levantar os objetivos do sistema, seus atributos de qualidade (Seções 7.1 e 7.2). Após essa apresentação, iniciou-se a execução do método de avaliação e refinamento proposto, apresentado na Seção 4.2. Como primeiro passo do método proposto, os atributos de qualidade foram organizados e priorizados em uma árvore de utilidade (Seção 7.3). Com a informação das prioridades a serem dadas a cada atributo de qualidade, a Seção 7.4 expõe o design do experimento, de acordo com o método *Goal-Question-Metric* (GQM). Em seguida, fo-

ram definidos cenários de execução para avaliar cada atributo de qualidade selecionado (Seção 7.5). Por fim, esses cenários foram executados (Seção 7.6) e seus resultados analisados (Seção 7.7). Desse modo, os gargalos arquiteturais de cada atributo de qualidade são identificados (Seção 7.8). Para a identificação dos *trade-offs* arquiteturais, se faz necessário sumarizar os resultados obtidos para todos os atributos de qualidade e avaliar a sua significância estatística dos resultados de cada um dos refinamentos realizados na arquitetura de referência (Seção 7.9). Analisando os resultados dos atributos de qualidade, são identificados os *trade-offs* arquiteturais (Seção 7.10), que por sua vez, são utilizados como guias no processo de refinamento da arquitetura de referência (Seção 7.11).

7.1 Apresentação das Diretivas de Negócios

A arquitetura proposta busca oferecer aos desenvolvedores de softwares educacionais a infraestrutura necessária para construção de ferramentas de BI Educacional, que possam auxiliar professores e educadores no processo de tomada de decisão pedagógica em ambientes educacionais utilizados no apoio à educação *on-line*. Tal infraestrutura deve considerar a integração de dados provenientes de diferentes fontes, a fim de executar ações personalizadas pelos próprios professores e educadores. Algumas das principais características da arquitetura proposta são:

- Adaptação a diferentes ambientes de educação *on-line*;
- Integração de fontes de dados heterogêneas;
- Reuso de software e de soluções e algoritmos para descoberta de conhecimento em bancos de dados no contexto educacional;
- Automatização de ações pedagógicas definidas pelo próprio professor/educador;
- Atender a uma demanda crescente de acessos simultâneos, tanto em número de disciplinas e cursos, quanto em número de alunos;

7.2 Arquitetura de Referência Inicial

A partir das características desejadas para a infraestrutura, foi projetada uma primeira versão da arquitetura de referência, conforme apresentado no Capítulo 5 (Figura 5.1). Para facilitar a leitura da tese, a mesma figura é reapresentada na Figura 7.1; em seguida, nas Seções 7.3 a 7.11, é apresentada a execução do processo de avaliação e refinamento dessa arquitetura, de acordo com as atividades do método proposto, apresentado na Seção 4.2.

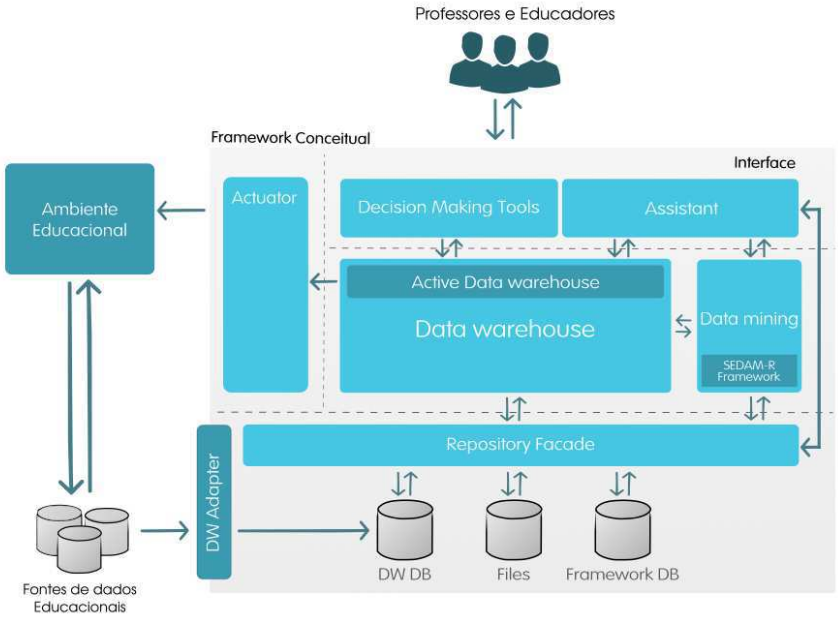


Figura 7.1: Versão Inicial da Arquitetura de Referência Proposta (reapresentação da Figura 5.1)

Fonte: Elaborada pelo autor

7.3 Identificação dos Atributos de Qualidade e Geração da Árvore de Utilidade (Atividades 1 e 2)

As primeiras atividades do método visam definir os atributos de qualidade da solução proposta e as versões iniciais dos cenários de avaliação estruturados em uma árvore de utilidade. Para a definição da lista de atributos de qualidade com as respectivas prioridades foi realizado um *brainstorm* envolvendo alguns *stakeholders*: potenciais usuários (professores volutuários), desenvolvedores (alunos de mestrado) e arquiteto de software (pesquisadores

da área). O *brainstorm* foi conduzido em três etapas; (1) apresentação da contextualização e problemática e de um resumo dos trabalhos relacionados apresentados no Capítulo 3; (2) discussão sobre os requisitos de qualidade necessários no contexto atual da educação *online*; (3) priorização dos requisitos de qualidade identificados. O moderador do *brainstorm* foi o próprio pesquisador, autor da tese. Com exceção dos desenvolvedores, os demais voluntários são todos doutores, com pelo menos cinco anos de experiência na docência do ensino superior envolvendo plataformas de educação *online*.

Como resultado, os principais atributos de qualidades em ordem de prioridade são:

1. **Escalabilidade**, que consiste em aumentar, de acordo com a demanda do sistema, o número de unidades (e.g., disciplina, curso, universidade) que podem ser atendidos simultaneamente pelo sistema;

2. **Eficiência de desempenho**, que consiste no menor tempo para processamento de cada unidade (e.g., disciplina, curso, universidade);

3. **Interoperabilidade**, que consiste na capacidade para integração da arquitetura proposta a diferentes tecnologias, tais como ambientes de educação *on-line* e bases de dados;

4. **Manutenibilidade**, que consiste na facilidade para se realizar atividades de manutenção, sejam elas corretivas ou evolutivas; e

5. **Usabilidade**, que consiste na facilidade de uso da interface oferecida pela arquitetura proposta.

Dentre os atributos identificados, escalabilidade e desempenho foram definidos como prioritários para uma avaliação quantitativa. Apesar de sua relevância, o atributo de usabilidade foi considerado de baixo impacto arquitetural, já que depende principalmente do projeto da interface. Devido ao escopo, os demais atributos devem ser avaliados posteriormente. Tal análise é apresentada durante a discussão dos resultados, apresentada na Seção 7.12.

Para a condução da avaliação detalhada (quantitativa) da arquitetura de referência projetada, foi elaborada uma árvore de utilidade (Figura 7.2), que considera os dois atributos de qualidade principais: escalabilidade e desempenho.

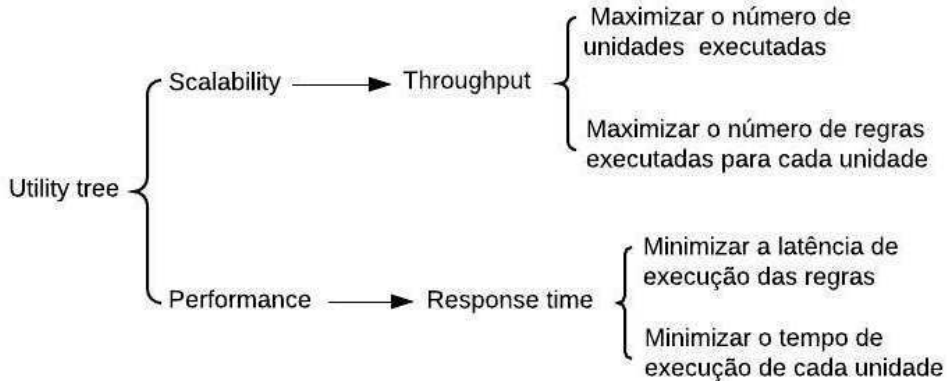


Figura 7.2: Árvore de Utilidade
Fonte: Elaborada pelo autor

Como pode ser observado na Figura 7.2, em relação ao atributo de escalabilidade, vale ressaltar que ele é avaliado a partir da variação do *throughput*. São considerados dois pontos de vista de *throughput*: (1) número de unidades de execução atendidas simultaneamente, isto é, que podem ser executadas em paralelo; e (2) número de regras que podem ser executadas por segundo. Em relação ao atributo de desempenho, foi considerado o tempo de resposta também sob duas perspectivas: (1) menor latência de execução das regras (tempo de execução de uma regra); e (2) o tempo para finalizar cada unidade de execução, tais como: disciplina, curso, departamento, universidade, etc.

Outro ponto fundamental que motivou a priorização dos atributos de escalabilidade e desempenho, é o *tradeoff* natural que acontece entre eles. Foi anseio dos projetistas entender melhor a repercussão desse *tradeoff* na arquitetura. Na arquitetura inicial, apresentada no Capítulo 5 e reapresentado na Figura 7.1, o modelo de execução das requisições adotado foi o de agendamento. Basicamente, a arquitetura de referência proposta agenda a maioria das requisições que executa, tais como execução de regras, dos modelos de mineração associados as regras, entre outros. Esse modelo foi adotado para evitar a sobrecarga sobre os servidores de aplicação diante da demanda das plataformas educacionais e o grande volume de dados gerados por elas, uma vez que a execução das requisições acontecem em horários alternativos onde a demanda sobre esses servidores é reduzida. Dessa maneira, ao final do dia, semana ou mês, dependendo da frequência de sumarização dos dados definidos pelos usuários, a arquitetura proposta executa todas as requisições agendadas em intervalos de 200 ms, ou seja, esse modelo pode levar a execução de um grande volume de requisições. Portanto, espera-se um alto *throughput* aliado a um bom desempenho, de modo que a arquitetura

de referência proposta seja viável. Porém, mesmo sendo tendo potencial de viabilidade, a expectativa dessa etapa é que com a identificação dos *tradeoffs* seja possível identificar melhorias e aperfeiçoá-las.

7.4 Design do experimento

O objetivo desse experimento foi definido com base no *Template GQM (Goal, Question, Metric)* (BASILI, 1992). Assim, *o experimento tem como objetivo* avaliar diferentes versões de arquiteturas de software *com a intenção de* compará-las *a respeito de* seu desempenho e escalabilidade *do ponto de vista de* desenvolvedores de ferramentas de tomada de decisão educacional *no contexto de* um ambiente educacional, a exemplo do Moodle. Com base no objetivo, as seguintes questões de pesquisa foram definidas:

- **QP 1:** A arquitetura alternativa tem um melhor desempenho do que a arquitetura original?
- **QP 2:** A arquitetura alternativa possui uma maior escalabilidade do que a arquitetura original?

Além da análise de satisfação dos requisitos de qualidade, um objetivo secundário desejado com o experimento é avaliar a redução de esforço de desenvolvimento proporcionado pela infraestrutura desenvolvida. Tal avaliação deve considerar a taxa de reúso de software alcançada com a adoção do *framework* que realiza a arquitetura de referência proposta, apresentado no Capítulo 6. Esse objetivo gerou mais uma questão de pesquisa, apresentada a seguir:

- **QP 3:** A adoção da arquitetura de referência proposta proporcionou reúso de software de forma significativa?

7.4.1 Planejamento do experimento

O experimento foi executado no contexto de um ambiente educacional específico e possui um caráter comparativo. Foram utilizados dados reais provenientes de um ambiente educacional, mas de modo completamente independente deste ambiente. Foram utilizados dados de

uma disciplina do curso de Sistema da Informação com cerca de 130 estudantes que foram carregados no DW construído e apresentado na Seção 6.1.3. As hipóteses definidas para cada um dos dois atributos de qualidade considerados foram:

1. **Desempenho:**

   Hipótese Nula, H_0: Não há diferença de desempenho (*Des*) entre a arquitetura alternativa (*ArqAlt*) e a arquitetura original (*ArqOrg*). H_0: $Des(ArqAlt) = Des(ArqOrg)$

   Hipótese Alternativa, H_1: Há diferença de desempenho (*Des*) entre a arquitetura alternativa (*ArqAlt*) e a arquitetura original (*ArqOrg*). H_1:$Des(ArqAlt) \neq Des(ArqOrg)$

2. **Escalabilidade**

   Hipótese Nula, H_0: Não há diferença na escalabilidade (*Esc*) entre a arquitetura alternativa (*ArqAlt*) e a arquitetura original (*ArqOrg*). H_0: $Esc(ArqAlt) = Esc(ArqOrg)$

   Hipótese Alternativa, H_1: Há diferença na escalabilidade (*Esc*) entre a arquitetura alternativa (*ArqAlt*) e a arquitetura original (*ArqOrg*). H_1: $Esc(ArqAlt) \neq Esc(ArqOrg)$

No tocante ao reuso de software proporcionado pela arquitetura de referência, as hipóteses definidas foram as seguintes:

1. **Reuso de Software:**

   Hipótese Nula, H_0: O índice de reuso de software não foi considerado significativo, sendo menor que 50% do código-fonte

   Hipótese Alternativa, H_1: O índice de reuso de software foi considerado significativo, sendo maior ou igual a 50% do código-fonte.

As métricas para o atributo de escalabilidade foram avaliadas sob dois pontos de vista diferentes: (1) número de unidades atendidas simultaneamente, isto é, que podem ser executadas em paralelo; e (2) número de regras que podem ser executadas por segundo. Em relação ao atributo de desempenho, foi considerado o tempo de resposta também sob duas perspectivas: (1) latência de execução das regras (tempo de resposta); e (2) o tempo para finalizar cada unidade de execução, tais como disciplina, curso, departamento, universidade,

etc. A métrica adotada para calcular o índice de reuso de código foi a proporção de linhas de código consideradas gerais e específicas.

O número de unidades atendidas simultaneamente deve ser fruto de execução em paralelo. Acredita-se que quanto maior o número de unidades de execução em paralelo, menor será o tempo de espera em fila, para que sua unidade de execução tenha o processamento ao menos iniciado. Já o número de regras que podem ser executadas por segundo demanda a vasão de execução do ponto de vista da unidade mínima de execução.

Em relação à latência de execução das regras, essa métrica representa o tempo de espera entre o recebimento da requisição e a finalização da execução das regras. Essa métrica foi considerada importante por representar uma medida de desempenho adequado para avaliar a execução nos recursos computacionais disponíveis. Já o tempo para finalizar cada unidade de execução representa o início da execução da primeira regra e o tempo de finalização da última regra da unidade de execução.

No tocante ao índice de reuso de código, as linhas de código foram classificadas como gerais quando provenientes de *frozen-spots* do framework desenvolvido. Já as linhas de código consideradas específicas são as implementadas fruto de adaptações do *framework* (*hot spots*), como por exemplo, adequação a uma plataforma de educação *on-line* específica.

7.4.2 Execução dos Experimentos

A realização dos experimentos envolveu a execução da arquitetura implementada como um todo, desde a recuperação dos dados até a execução das ações definidas nas regras, seguindo o fluxo definido no diagrama de sequência apresentado na Figura 5.7. Para cada um dos cenários definidos foram realizadas 30 execuções. Para cada cenário, foram mensuradas as métricas definidas para os atributos de escalabilidade e desempenho. O número de execuções foi definido a partir dos resultados de testes preliminares. Durante esses testes foram mensuradas cada uma das métricas definidas anteriormente, e, a partir delas, calculado o número de execuções com base no tempo de resposta para um intervalo de confiança de 95%.

Em cada execução foram disparadas múltiplas *threads* a fim de simular a execução de múltiplas unidades de execução, que são responsáveis por executar os conjuntos de regras disponíveis. As *threads* foram disparadas em intervalos de 200 ms. O tempo de 200 ms foi adotado na instanciação da arquitetura de referência como o intervalo de tempo ideal entre

as execuções das unidades de execução agendadas previamente pelo sistema. Esse valor foi definido após a execução de diversos testes preliminares, onde foram avaliadas execuções com diferentes intervalos de tempo entre as unidades. Os testes variaram com intervalos entre 0 ms e 1000 ms, incrementando 100 ms em cada teste preliminar. Após a realização desses testes, percebeu-se que o menor intervalo de tempo que possibilitava uma execução estável do sistema foi de 200 ms. Vale ressaltar que esse intervalo está diretamente ligado à infraestrutura de *hardware* utilizada nos experimentos e que diferentes infraestruturas podem possibilitar uma redução ou aumento desse intervalo de tempo.

Para a execução, foram adotadas duas configurações de ambientes de execução. A escolha dos ambientes se deu de acordo com os objetivos de cada cenário. Os dois ambientes de execução definidos foram: local e distribuído.

- **Ambiente local:** nesse ambiente as execuções aconteceram em um único computador com a seguinte configuração: IDE Netbeans e o Java 8 em um notebook com processador Intel Core i7-3520M (3ª geração) com 6 GB de memória RAM e sistema operacional Ubuntu 16.04;

- **Ambiente distribuído:** esse ambiente foi configurado utilizando quatro computadores em uma rede local de computadores conectados via interface RJ45 com velocidade 10/100 MBs. O computador utilizado no ambiente local representou o cliente, outros dois computadores foram configurados como servidores de aplicação e um outro computador foi configurado como servidor de dados. Entre os servidores de aplicação, ambos configurados com IDE Netbeans e o Java 8, e equipados com processadores Intel Core i5, um com 4GB de memória RAM e outro com 8GB de memória RAM. O servidor de banco de dados teve configurado o PostgreSQL e era equipado com processador Intel Core2 Duo e 8GB de memória RAM, todos com sistema operacional Ubuntu 16.04.

7.5 Gerar Cenários Arquiteturais (Atividade 3)

Seguindo o método de avaliação e refinamento arquitetural proposto neste trabalho, Seção 4.2, a atividade atual consiste na identificação de cenários de execução arquiteturais

que reflitam execuções reais do sistema.

Com a opção pelos atributos de escalabilidade e desempenho, os cenários abstratos idealizados na árvore de utilidade apresentada na Figura 7.2 passaram por um refinamento através de uma reunião de *brainstorm* com *stakeholders* para criação de cenários de execução. Após esse processo, foram definidos 02 cenários de execução:

Cenário 1: Com os agendamentos para execução das regras para o final de semana, é esperado um grande volume de ações acumuladas, que devem ser executadas o mais rápido possível. Considerando que os agendamentos são realizados de maneira sequencial com um intervalo de 200 milissegundos entre eles, atingindo um pico de 10 execuções por segundo.

Cenário 2: Com o número de execuções de regras demandadas pelos agendamentos, em um pico de execuções com um intervalo de 200 milissegundos, atingindo 10 requisições por segundo, as regras devem ser executadas com latência máxima de 15 segundos para a execução da primeira regra.

Para as etapas acima mencionadas, podem ser necessárias a realização de diversas iterações. Mas para facilitar a análise dos resultados, as próximas seções apresentam as informações de todas as iterações realizadas, em ordem cronológica. Por fim, a Seção 7.11 apresenta uma discussão detalhada das iterações executadas durante o projeto arquitetural.

Ao todo, foram executadas três iterações. As iterações seguiram o fluxo de execução definido no método de avaliação e refinamento proposto na Seção 4.2, iniciando na etapa de execução dos cenários arquiteturais até a etapa de refinamento arquitetural.

7.6 Executar Cenários Arquiteturais (Atividade 4)

Na primeira iteração, as execuções ocorreram no ambiente local. Nesse ambiente foram executadas duas versões da arquitetura, a versão original da arquitetura proposta e a versão alternativa (com distribuição no componente ativo), ou seja, simulando dois servidores de aplicação e o servidor de banco de dados. Em ambos os casos cada unidade de execução estava associada a apenas 1 conjunto de regras. Para realizar o monitoramento na execução dos cenários, foram utilizados temporizadores definidos no código-fonte e o *JConsole*, disponibilizado pela maquina virtual do Java.

Para a segunda iteração, as execuções foram realizadas no ambiente distribuído. Nessa

iteração foi executada a versão alternativa (com distribuição no componente ativo) sob duas perspectivas: um conjunto de regras por unidade de execução e 100 conjuntos de regras por unidade de execução. Assim como no cenário anterior, foram utilizados temporizadores definidos no código-fonte e o *JConsole*, disponibilizado pela máquina virtual do Java.

Para a terceira e última iteração, as execuções foram realizadas também no ambiente distribuído, mas utilizando uma nova versão da arquitetura. A versão alternativa da arquitetura passou por um refinamento após a iteração anterior. O refinamento consistiu em alterar a implementação do distribuidor de carga arquitetural. Assim, a arquitetura passou a não mais distribuir unidades de execução entre os servidores de aplicação, como nos cenários anteriores, mas em distribuir um conjunto de regras para cada servidor de aplicação. Desse modo, o componente que realiza o balanceamento de carga passou a recuperar as unidades de execução e as regras associadas a eles, e distribuir essas regras entre os servidores. Assim como na iteração anterior, cada unidade de execução também estava associada a 100 conjuntos de regras com o objetivo de comparar a nova versão da arquitetura com a versão alternativa, executada na iteração anterior.

7.7 Avaliar Métricas Relacionadas aos Cenários e Sumarizar a Avaliação de Todos os Cenários(Atividades 5 e 6)

Na primeira iteração, os resultados obtidos mostraram que para a métrica de desempenho, onde cada unidade de execução tem apenas um conjunto de regras associada, a latência de execução da primeira regra a ser disparada se manteve semelhante em ambas as versões da arquitetura, com tempo médio variando entre aproximadamente 40 e 50 segundos. Para as métricas de *threads* por segundo e regras por segundo, ambas as versões mantiveram médias equivalentes, apesar de uma pequena variação. Em média, as versões da arquitetura alcançaram um *throughput* de 3 *threads* por segundo. No entanto, na métrica de regras executadas por segundo, houve uma pequena variação onde a versão original alcançou uma média de 0,5 regras por segundo, enquanto a versão alternativa alcançou, em média, o *throughput* de 1,5 regras por segundo.

Na segunda iteração, a arquitetura alternativa foi executada com uma regra para cada unidade de execução e com 100 regras para cada unidade de execução, a fim de avaliar o

comportamento do sistema, bem como, mensurar as métricas definidas em um ambiente distribuído, ou seja, mais próximo de um cenário real de utilização. A execução da arquitetura com uma regra por unidade de execução apresentou redução da latência de execução da primeira regra; que caiu de 40 segundos, na iteração anterior, para em média 10 segundos. Na execução da arquitetura alternativa no ambiente distribuído com 100 regras por unidade de execução, os resultados mostraram que a latência de execução da primeira regra permaneceu o mesmo (10s), suportando bem o aumento no volume de regras por unidade de execução. Mas ao analisar os resultados, percebeu-se que o mesmo já era esperado, uma vez que o distribuidor de carga repassa uma unidade de execução para cada servidor de aplicação; dessa forma, a execução das 100 regras ocorre sequencialmente. Percebemos com isso, que apesar do distribuidor de carga favorecer o atendimento de um maior número de unidades de execução em paralelo, o tempo de execução de cada unidade continua sendo a soma do tempo de execução das regras, definidas pelo professor/educador para cada uma, que nos cenários, foram 100 regras. O *throughput* médio alcançou máxima de 26,66 regras por segundo e média de aproximadamente 10 regras por segundo. Vale ressaltar ainda que essa arquitetura permitiu que diferentes unidades de execução fossem atendidas simultaneamente, na mesma proporção do número de servidores de aplicação disponíveis para execuções.

Com o objetivo de tentar reduzir o tempo para processar cada unidade de execução, na terceira e última iteração, a arquitetura alternativa, utilizada na Iteração 2, foi refinada. Nesse refinamento, o único ponto ajustado foi o comportamento do distribuidor de carga. Ao invés de distribuir uma unidade de execução para cada servidor, o novo distribuidor de carga passou a carregar as unidades de execução de forma sequencial e distribuir as suas regras para os servidores de aplicação. Os resultados da execução da arquitetura alternativa no ambiente distribuído executados na iteração anterior foram comparados com a nova versão refinada da arquitetura. Apesar de ter sido registrado um crescimento considerável no número de *threads* abertas, foi constatado que esse fenômeno aconteceu devido à menor granularidade da distribuição, que passou a distribuir as regras individualmente. Outra constatação observada, foi o fato do número de regras processadas por segundo ter se mantido inalterado. Sendo assim, dependendo do ponto de vista, *throughput* da arquitetura permaneceu o mesmo. Porém, se observarmos sob o ponto de vista do número de unidades de execução atendidas simultaneamente, a nova arquitetura registrou redução considerável do *throughput*, uma vez que

antes cada um dos dois servidores de aplicação executava uma unidade. O maior ganho constatado com a arquitetura refatorada foi em relação ao tempo de execução de cada unidade de execução, contando a partir do momento que a execução foi iniciada. Com o paralelismo na execução das regras de uma mesma unidade de execução, o tempo de execução de cada uma pôde ser reduzido pela metade, já que o cenário considerou a existência de dois servidores de aplicação.

7.8 Identificação dos Impactos Arquiteturais (*bottlenecks*) (Atividade 7)

Os resultados obtidos na primeira iteração ressaltam que, de fato, a execução de regras se mostraram um gargalo arquitetural. Essa restrição se deve tanto ao grande volume de processamento demandado para execução das regras, quanto à forma como as regras são executadas pelo *framework Drools*. Dessa forma, a distribuição proposta para a evolução da arquitetura se mostrou bem sucedida e necessária em cenários com grande demanda de execução.

Na segunda iteração, apesar do crescimento no *throughput* do processamento de unidades de execução no ambiente distribuído, a execução de cada unidade ainda é sequencial. Isso implica em um gargalo de desempenho. Além disso, percebeu-se uma alta demanda de requisições ao servidor de banco de dados, podendo este tornar-se um gargalo arquitetural. Um primeiro passo para reduzir a demanda do SGBD foi o refinamento realizado na Iteração 3, que consistiu na execução de uma unidade de execução de cada vez, paralelizando as suas regras internas. Com isso, espera-se se beneficiar de artifícios existentes em alguns SGBDs, tais como cache de dados recentes, já que as regras de uma mesma unidade de execução faz referência aos dados dos mesmos grupos de estudantes.

Ao avaliar os resultados da terceira iteração, percebeu-se um aumento no desempenho de processamento de cada unidade de execução. Porém, o *throughout* de processamento de unidades de execução é sequencial. Diante disso, mesmo considerando uma mesma unidade de execução, as regras podem demandar acesso a diferentes dados do banco de dados, o que também pode acarretar um grande número de consultas e congestionamento do servidor de banco de dados. Na discussão final (Seção 7.12) são apresentados possíveis refinamentos para contornar tais gargalos, caso passem a ocorrer.

Tanto na Iteração 2, quanto na Iteração 3, foi constatado que dependendo do número de requisições realizadas aos servidores de aplicação, eles provocavam a parada do servidor de banco de dados. A falha no servidor de dados foi sobrecarregado devido ao grande número de requisições recebidas para acesso aos dados das turmas, fazendo com que as requisições parassem de responder. Por consequência, as regras paravam de ser executadas, continuando a execução após o reinício do servidor de dados. Os cenários de execução foram repetidos aumentando o tempo de espera entre a criação das *threads*, que originalmente era de 200ms, para 1000ms. Nessa nova configuração, a arquitetura estabilizou-se e o servidor de dados não mais falhou. Após análise dos resultados, chegou-se à conclusão de que, se a demanda por desempenho e *throughput* for crítica, o que envolveria um maior paralelismo da execução e a abertura de *threads* num tempo mais curto, a arquitetura de referência proposta precisaria ser executada utilizando SGBDs distribuídos, que é uma tecnologia presente atualmente nos SGBDs comerciais e até mesmo em alguns SGBDs gratuitos.

7.9 Sumarizar Avaliação de Todos os Atributos de Qualidade (Atividade 8)

Nesta seção são sumarizados os resultados obtidos na execuções dos cenários arquiteturais. Os resultados estão apresentados a partir das métricas apresentadas no planejamento do experimento (Seção 7.4), de acordo com o método GQM.

Iteração 01

A Tabela 7.1 apresenta os valores mínimos e máximos, a média, mediana e desvio padrão do tempo de execução de uma regra (tempo de resposta). Para o cálculo do tempo de execução de cada unidade, como as regras de cada unidade são executadas em sequência, a fórmula utilizada foi a seguinte: $Tempo_{Total} = TempoMédio_{Regra}x100$. O valor totalizado foi $4560s$ ou $76min$ para a arquitetura original e $5270s$ ou $87,83min$ para a arquitetura alternativa. A Tabela 7.2 apresenta os mesmos valores para o número de regras executadas por segundo (*throughput*). A Figura 7.3 apresenta os *box plots* das duas métricas avaliadas.

Os histogramas das métricas de tempo de resposta e número de regras executadas por

Tabela 7.1: Resultado da métrica - Tempo de resposta

	Mínimo	1º Qt	Mediana	Média	Desvio Padrão	3º Qt	Máximo
Arq. Original	34,0	41,0	44,0	45,6	8,219	47,5	71,0
Arq. Alternativa	45,0	49,0	52,0	52,70	5,553	55,75	66,0

Tabela 7.2: Resultado da métrica - Regras por segundo

	Mínimo	1º Qt	Mediana	Média	Desvio Padrão	3º Qt	Máximo
Arq. Original	0,1500	0,3782	0,5270	0,525	0,2204	0,6275	1,2260
Arq. Alternativa	0,600	1,321	1,550	1,617	0,5585	1,929	3,000

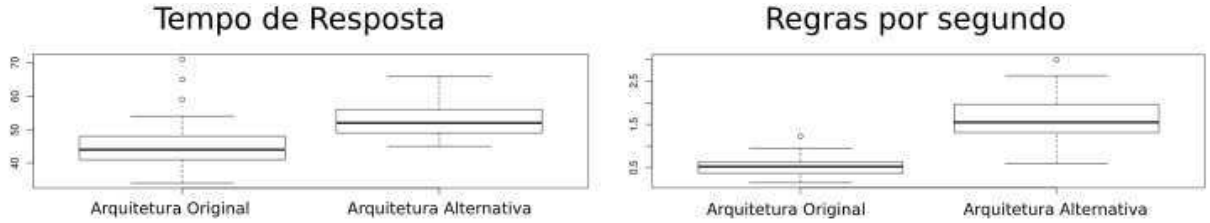


Figura 7.3: Boxplot -Iteração 01
Fonte: Elaborada pelo autor

segundo são apresentados na Figura 7.4. Como pode ser observado, os histogramas para a métrica *tempo de resposta* indicam uma distribuição não-normal, que foi confirmado pelo teste *Shapiro-Wilk* (SHAPIRO; WILK, 1965) com *p-value* de 0,024 para a arquitetura original e 0,00268 para a arquitetura alternativa, ou seja, resultados com *p-value* abaixo do limiar de 0,05. Para métrica *regras por segundo*, os histogramas apresentados indicam uma distribuição normal, que foram confirmados pelo teste *Shapiro-Wilk* (SHAPIRO; WILK, 1965) com *p-value* de 0,068 para a arquitetura original e 0,639 para a arquitetura alternativa.

A fim de avaliar a métrica *tempo de resposta* foi aplicado o teste de *Wilcoxon* (WILCOXON, 1945) e o teste de *Kolmogorov-Smirnov* (LILLIEFORS, 1967) para a métrica de *regras por segundo*. Os resultados obtidos apresentaram diferenças significantes estatisticamente, com *p-value* de 1,393e-05 para o tempo de resposta e 3,273e-10 para a métrica *regras por segundo*. No entanto, os resultados apontam que na métrica *tempo de resposta*, a arquitetura original teve um melhor desempenho. Para a métrica de *regras por segundo*, a arquitetura alternativa apresentou uma vazão significativamente maior, ou seja, maior número de regras executadas por segundo. O aumento no tempo de execução se deve, principalmente, pelo atraso decorrente do tráfego na rede para a comunicação entre os componentes, quando ana-

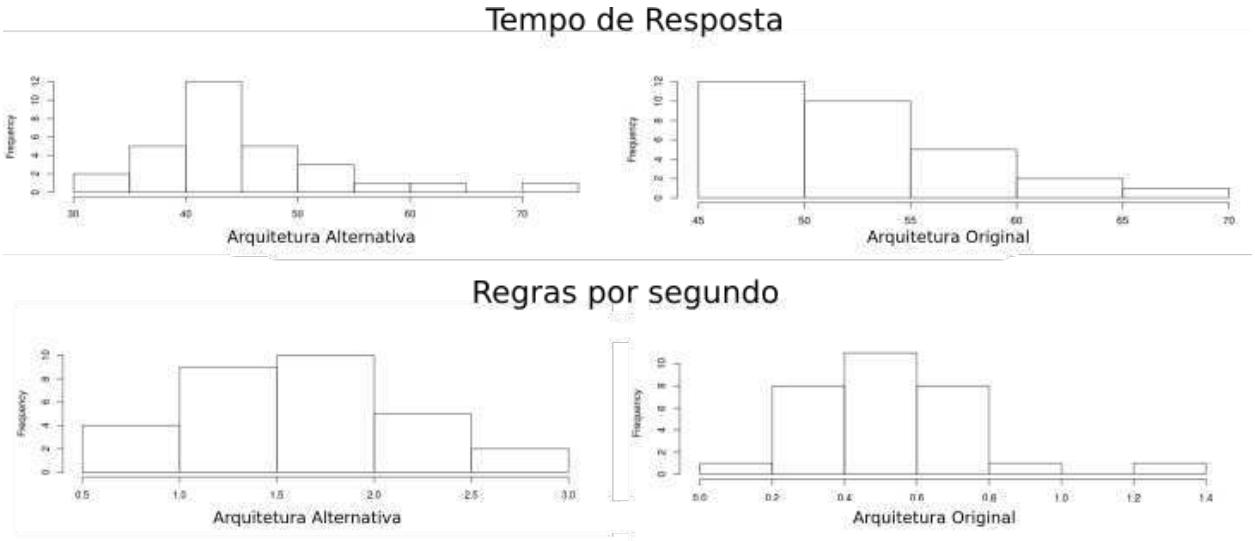


Figura 7.4: Histograma - Iteração 01
Fonte: Elaborada pelo autor

lisamos um cenário de execução sequencial. Contudo, como foi confirmado em uma iteração posterior de refinamento arquitetural (Iteração 3), o tempo de execução de cada unidade de execução tende a reduzir consideravelmente em cenários que permitam a execução das regras em paralelo.

Iteração 02

Em relação à Iteração 2, a Tabela 7.3 apresenta os valores mínimos e máximos, a média, mediana e desvio padrão dos resultados obtidos para as métricas tempo de resposta. O valor do tempo total para se processar cada unidade de execução foi de $991s$ ou $16,52min$ para a arquitetura alternativa, com a execução de uma única regra, e $946,7s$ ou $15,78min$ para a arquitetura alternativa, com a execução de 100 regras. Essa pequena diferença se justifica, uma vez que a execução da primeira regra costuma ser ligeiramente mais lenta, devido às atividades de inicialização das variáveis e serviços. A Tabela 7.4 apresenta os mesmos dados para o número de regras executadas por segundo. A Figura 7.5 apresenta os *box plots* das duas métricas avaliadas.

Tabela 7.3: Resultados da métrica - *Tempo de resposta*

	Min.	1º Qt	Mediana	Média	Desv. Padrão	3º Qt	Max.
Arq. Alt. (01 Regra)	8,000	9,000	9,500	9,910	1,983	9,950	19,500
Arq. Alt. (100 Regras)	8,000	9,000	9,000	9,467	1,620	10,000	16,000

Tabela 7.4: Resultados da métrica - *Regras por Segundo*

	Min.	1º Qt	Mediana	Média	Desv. Padrão	3º Qt	Max.
Arq. Alt. (01 Regra)	3,795	4,5000	4,667	4,774	0,7011	4,828	8,100
Arq. Alt. (100 Regras)	6,333	8,157	9,062	9,998	3,7397	10,431	26,666

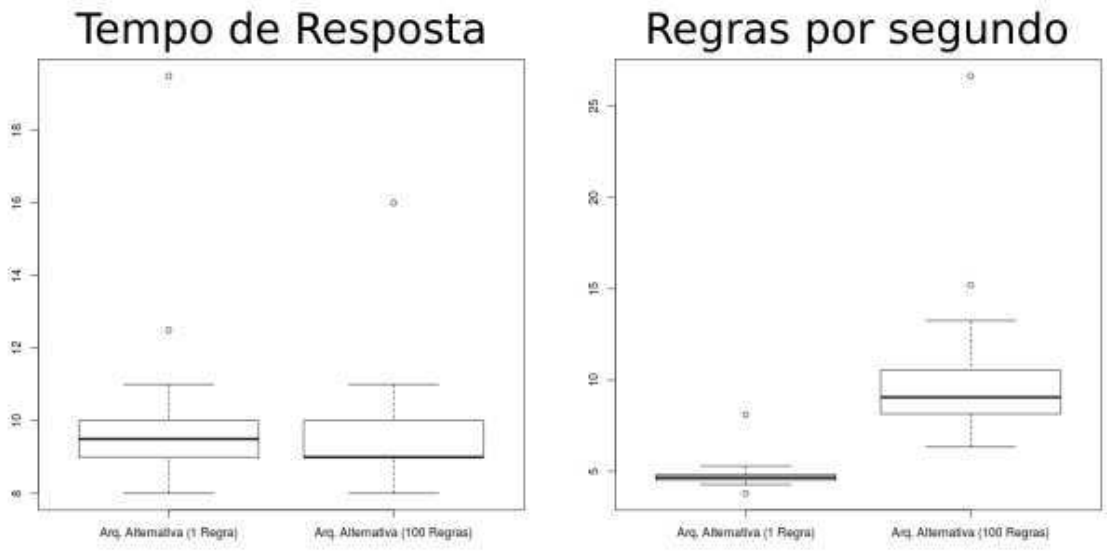


Figura 7.5: Boxplot -Iteração 02
Fonte: Elaborada pelo autor

Os histogramas das métricas tempo de resposta e número de regras executadas por segundo, da segunda iteração, são apresentados na Figura 7.6. Os histogramas para a métrica *tempo de resposta* indicam uma distribuição não-normal, que foi confirmado pelo teste *Shapiro-Wilk* (SHAPIRO; WILK, 1965) com *p-value* de 6.603e-09 para a arquitetura alternativa (01 Regra) e 6.327e-07 para a arquitetura alternativa (100 Regras). Para a métrica *regras por segundo*, os histogramas apresentados na Figura 7.7 indicam uma distribuição não-normal, que foi confirmado pelo teste *Shapiro-Wilk* (SHAPIRO; WILK, 1965) com *p-value* de 4.214e-08 para a arquitetura alternativa (01 Regra) e 5.43e-07 para a arquitetura alternativa (100 Regras).

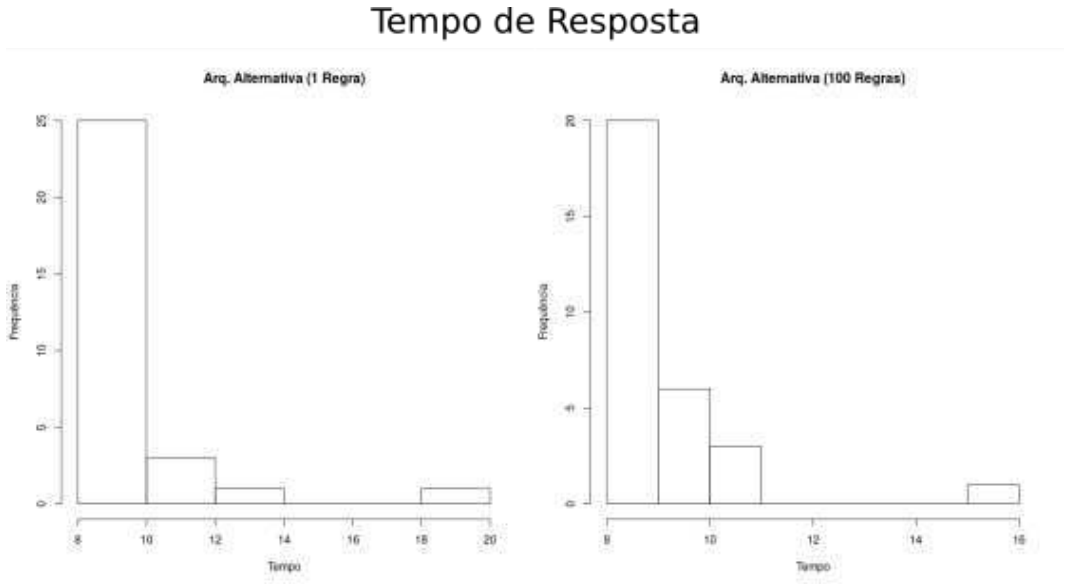


Figura 7.6: Histograma - Tempo de resposta - Iteração 02
Fonte: Elaborada pelo autor

Comparando as duas versões da arquitetura de referência, foi aplicado o teste de *Wilcoxon* (WILCOXON, 1945), com o intuito de avaliar se houve diferenças estatisticamente significantes entre os dois atributos de qualidade avaliados. Na avaliação da métrica *tempo de resposta*, o resultado obtido mostrou que não houve diferença significante estatisticamente, com *p-value* de 0,08736. Para a métrica de *regras por segundo*, o *Wilcoxon* (WILCOXON, 1945) apresentou como resultado um *p-value* de 8.96e-11, mostrando que há diferença significante estatisticamente. Assim, os resultados mostram que o desempenho se manteve o mesmo, com um aumento considerável na vazão, devido ao maior número de regras executadas.

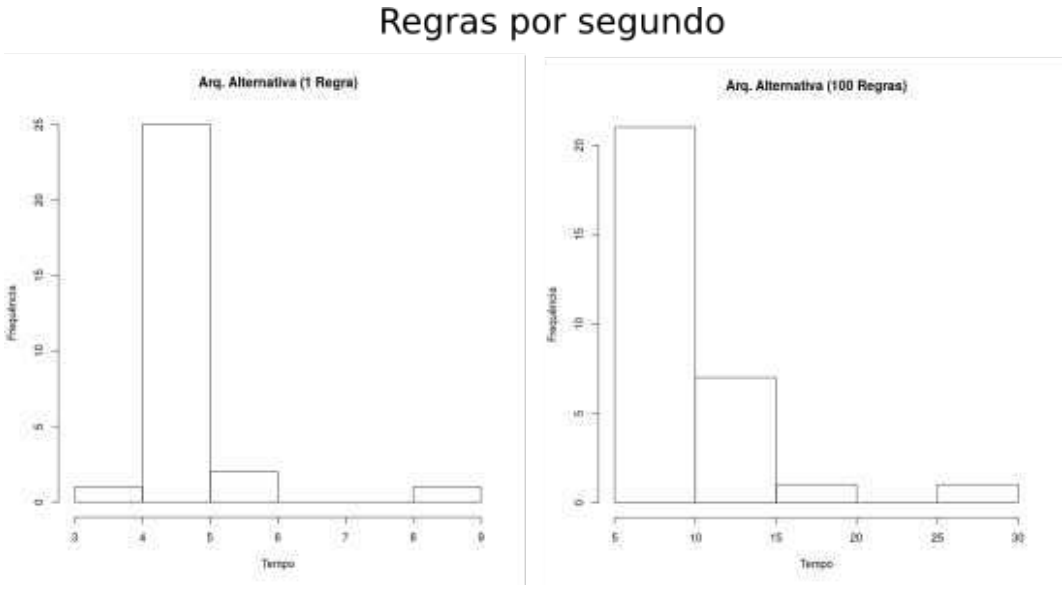


Figura 7.7: Histograma - Regras por segundo - Iteração 02
Fonte: Elaborada pelo autor

Iteração 3

Os resultados obtidos na terceira iteração foram sumarizados nas Tabelas 7.5 e 7.6, onde foram representados os valores mínimos e máximos, a média, mediana e desvio padrão. Na Figura 7.9, são apresentados os *box plots* das métricas avaliadas. O valor do tempo total para se processar cada unidade de execução foi de $946,7s$ ou $15,78min$ para a arquitetura alternativa da iteração anterior (Alternativa 1). Para a versão alternativa da arquitetura, nesta iteração, como houve uma preocupação com a paralelização de execução das regras de cada unidade de execução, a fórmula para calcular o tempo total da unidade teve de ser ajustado para: $Tempo_{Total} = TempoMédio_{Regra}x100/Quantidade_{Servidores}$; sendo assim, o valor total foi de $525s$ ou $8,75min$ para a nova versão da arquitetura alternativa (Alternativa 2).

Tabela 7.5: Resultados da Métrica - *Tempo de Resposta*

	Min.	1º Qt	Mediana	Média	Desv. Padrão	3º Qt	Max.
Arq. Alternativa 1	8,000	9,000	9,000	9,467	1,62	10,000	16,000
Arq. Alternativa 2	8,500	10,000	10,500	10,720	1,70	11,000	17,000

Tabela 7.6: Resultados da Métrica - *Regras por Segundo*

	Min.	1º Qt	Mediana	Média	Desv. Padrão	3º Qt	Max.
Arq. Alternativa 1	6,333	8,157	9,062	9,998	3,7397	10,431	26,666
Arq. Alternativa 2	6,874	8,348	9,222	9,768	1,9529	10,933	14,313

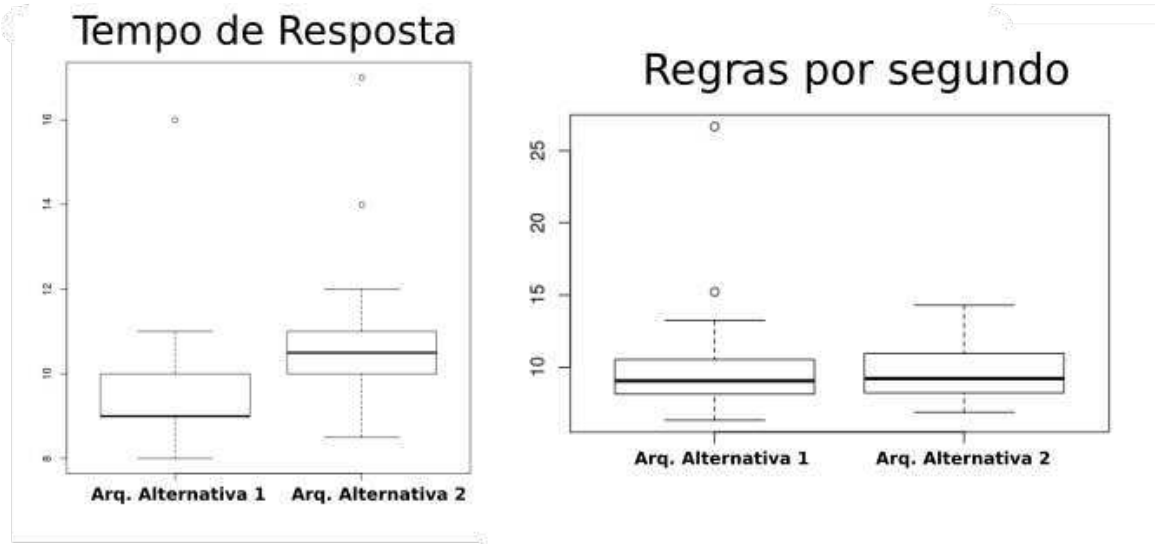


Figura 7.8: Boxplot -Iteração 03
Fonte: Elaborada pelo autor

Os histogramas das métricas tempo de resposta e regras por segundo da terceira iteração são apresentados a seguir. As métricas *tempo de resposta*, apresentadas na Figura 7.9, indicam uma distribuição não-normal, que foi confirmado pelo teste *Shapiro-Wilk* (SHAPIRO; WILK, 1965) com *p-value* de 6.327e-07 para arquitetura alternativa 1 e 7,083e-05 para arquitetural alternativa 2. Para métrica *regras por segundo*, os histogramas apresentados na Figura 7.10 indicam uma distribuição não-normal, que foi confirmado pelo teste *Shapiro-Wilk* (SHAPIRO; WILK, 1965) com *p-value* de 5.43e-07 para arquitetura alternativa 1 e 0,04276 para arquitetural alternativa 2.

Para ambas as métricas, foi aplicado o teste de *Wilcoxon* (WILCOXON, 1945) para comparar as duas execuções. O resultado obtido mostrou que não houve diferença significante estatisticamente com *p-value* de 0,08736 para a métrica *tempo de resposta*. Para métrica de *regras por segundo*, o *Wilcoxon* (WILCOXON, 1945) apresentou como resultado um *p-value* de 0.6735, mostrando que não há diferença significante estatisticamente.

Nota-se que apesar do atraso decorrente dos algoritmos de escalonamento e do tráfego na rede, o paralelismo das regras da Iteração 3 reduziu consideravelmente o tempo de execução total, para cada unidade de execução. Esse tempo é reduzido proporcionalmente ao número de servidores disponíveis. Como no experimento foram utilizados dois servidores, o tempo de execução de cada unidade de execução foi reduzido praticamente na metade.

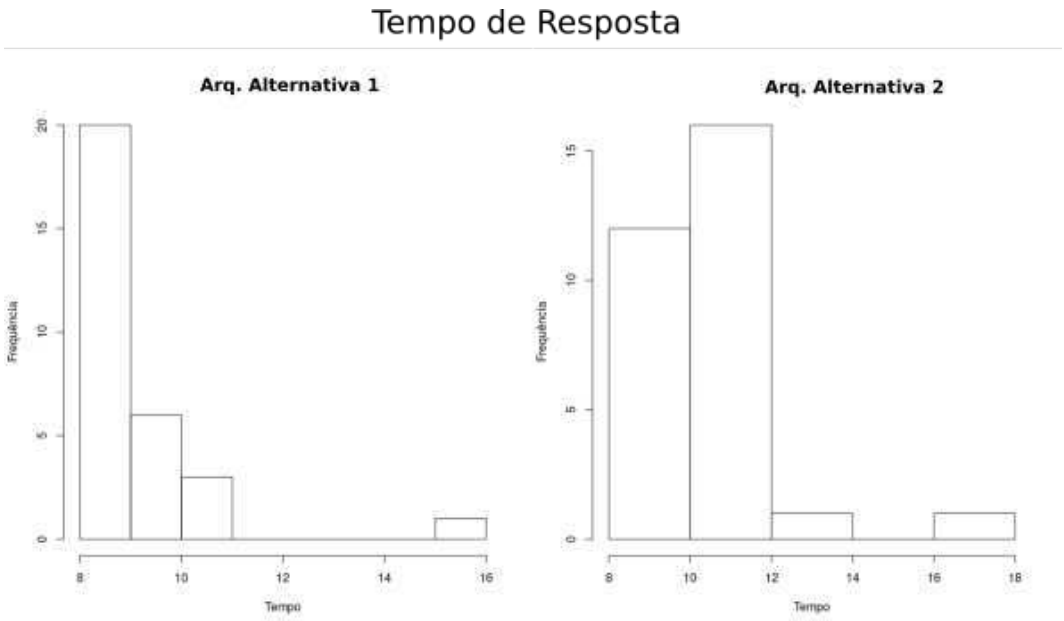


Figura 7.9: Histograma - Tempo de resposta - Iteração 03
Fonte: Elaborada pelo autor

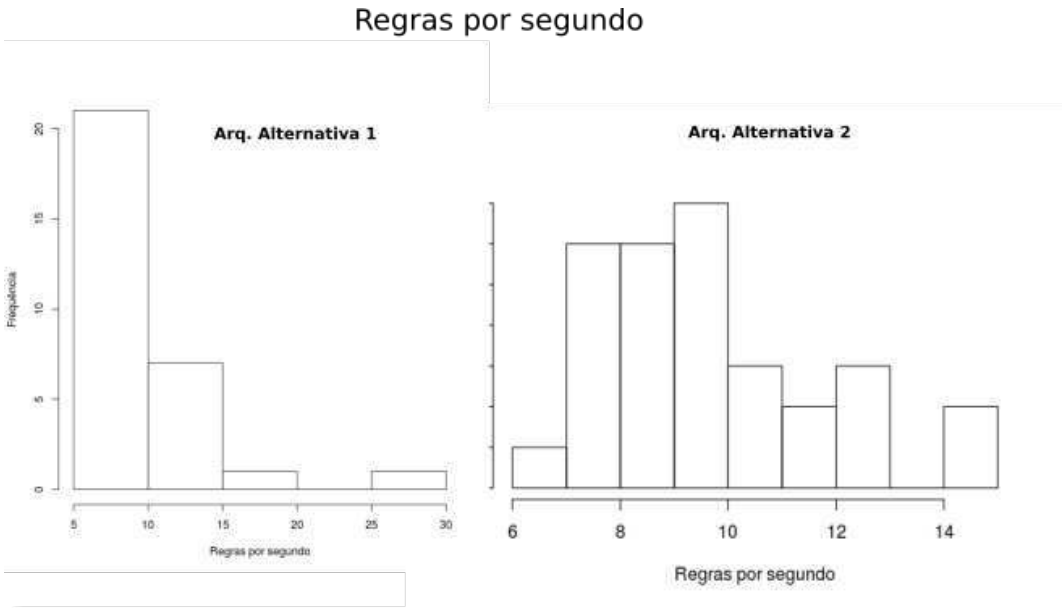


Figura 7.10: Histograma - Regras por Segundo - Iteração 03
Fonte: Elaborada pelo autor

Reuso de Software

O projeto de implementação envolveu aproximadamente 90 classes e 5700 linhas de código, divididos em cerca de 40 pacotes. Para estender a solução proposta é necessário a criação ou alteração em cerca de 12 classes diferentes e aproximadamente 650 linhas de código, gerando um reuso de aproximadamente 88% do *framework* implementado.

7.10 Identificação dos *trade-offs* (Atividade 9)

O principal *trade-off* identificado envolve a relação entre o *throughput* de unidades de execução atendidas simultaneamente e o desempenho individual para se processar cada unidade de execução. Tal *throughput* pôde ser constatado claramente na execução das Iterações 2 e 3. Sob a perspectiva arquitetural avaliada na segunda iteração, o crescimento do *throughput* de processamento de unidades de execução implica na diminuição no desempenho do processamento individual das unidade de execução, já que o processamento de cada unidade de execução ocorre de forma sequencial. Na perspectiva executada na terceira iteração, o aumento no desempenho no processamento individual das unidades de execução impacta na redução do *throughout* de processamento de unidades de execução, que passam a ser executadas de forma sequencial.

Por considerar que as duas opções de configuração arquitetural podem ser uteis, de acordo com as necessidades, preferências do usuário e disponibilidade de recursos, a arquitetura de referência apresenta as duas possibilidades como sendo variabilidades do distribuidor de carga. Dessa forma, o usuário pode optar por uma ou outra opção.

7.11 Refinamento da arquitetura de referência (Atividade 10)

A avaliação baseada em cenários permitiu a identificação de possíveis gargalos arquiteturais e *trade-offs* durante as iterações. Como resultado, a arquitetura proposta passou por um processo de refinamento, evoluindo a arquitetura de referência a cada iteração. Como a arquitetura inicial foi apresentada em detalhes no Capítulo 5, as seções a seguir apresentam

cada uma das duas adaptações geradas nessa arquitetura. Dadas as similaridades das duas versões refinadas da arquitetura de referência, os dois refinamentos são apresentados separadamente, sob o ponto de vista de diferentes visões arquiteturais. A Seção 7.11.1 apresenta a arquitetura alternativa, que considera a distribuição de unidades de execução e favorece o *throughput* de unidades executadas simultaneamente (Iteração 2). A Seção 7.11.2 apresenta a versão distribuída que prioriza desempenho na execução de cada unidade de execução, paralelizando a execução de regras. Como pode ser observado, a principal distinção existente entre as duas versões da arquitetura de referência está na visão de processos, que representa aspectos dinâmicos do comportamento da arquitetura. Nesse trabalho, utilizamos a notação de diagramas de sequência UML para a representação da visão de processos.

7.11.1 Arquitetura com Distribuição de Unidades de Execução (Iteração 2)

Além da arquitetura inicial, foi projetada uma versão alternativa da arquitetura a fim de comparar as diferentes abordagens arquiteturais. A hipótese que norteou o projeto da versão alternativa foi a possibilidade da arquitetura original não conseguir atender à alta demanda de processamento decorrente das requisições sobre o *Data Warehouse* (DW) ativo, que envolve execuções das regras de ações pedagógicas, de gatilhos do DW, e o processamento de dados para execução das regras e dos modelos de mineração. A arquitetura alternativa projetada, apresentada na Figura 7.11, considera a distribuição do componente *data warehouse* ativo, responsável pelo processamento das regras. Dessa forma, esse potencial gargalo poderia ser distribuído proporcionalmente, de acordo com o aumento na demanda de requisições.

Visão de Desenvolvimento

Na **Visão de Desenvolvimento** da arquitetura alternativa é apresentado no diagrama UML de componentes, apresentado na Figura 7.15, destacando os componentes envolvidos que foram reestruturados e refinados.

O refinamento arquitetural realizado incluiu a adição de um conector responsável pelo gerenciamento de carga do componente ativo. Esse conector recebe as requisições e realiza a distribuição de unidades de execução entre os servidores de aplicação disponíveis, gerando,

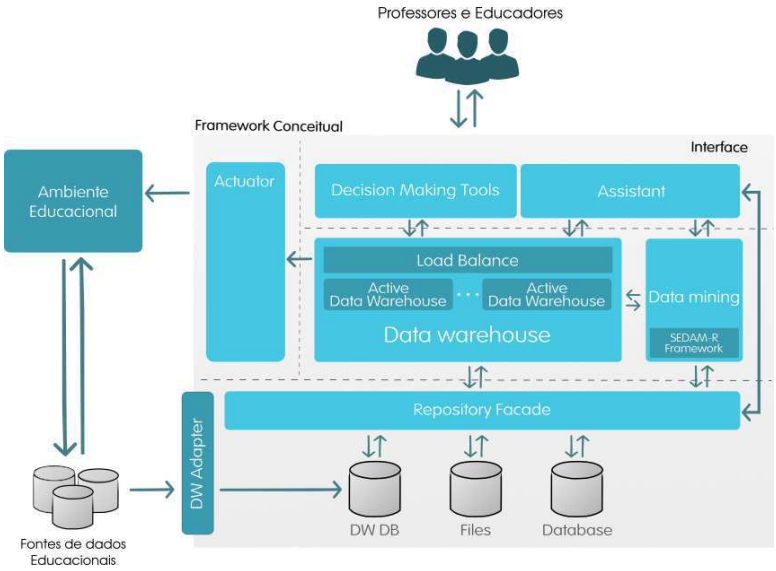


Figura 7.11: Arquitetura Alternativa (Com Distribuição do DW Ativo)
Fonte: Elaborada pelo autor

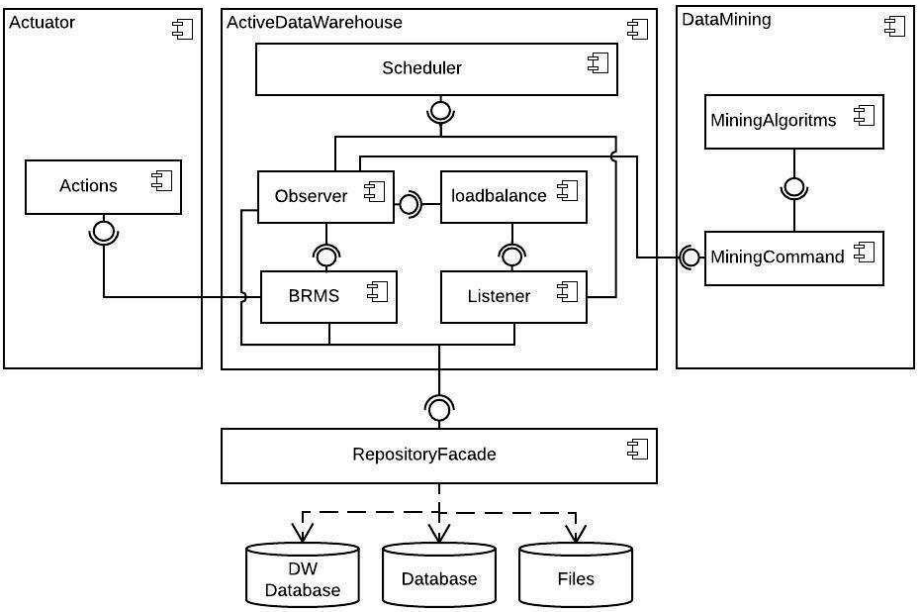


Figura 7.12: Diagrama de Componente
Fonte: Elaborada pelo autor

com isso, um aumento *throughput* de processamento de unidades de execução.

Visão de Processo Alternativa

Para ilustrar o funcionamento da arquitetura distribuída que prioriza a execução paralela de unidades de execução, a Figura 5.7 apresenta um diagrama de sequência UML representando o comportamento arquitetural quando regras com condições avançadas (com mineração de dados) são executadas. A diferença está no conector que realiza o balanceamento de carga das unidades de execução (*:LoadBalance*). Assim sendo, quando o :Listener dispara a execução das unidades de execução (*:Observer*), o conector distribuirá as unidades entre os servidores de aplicações disponíveis.

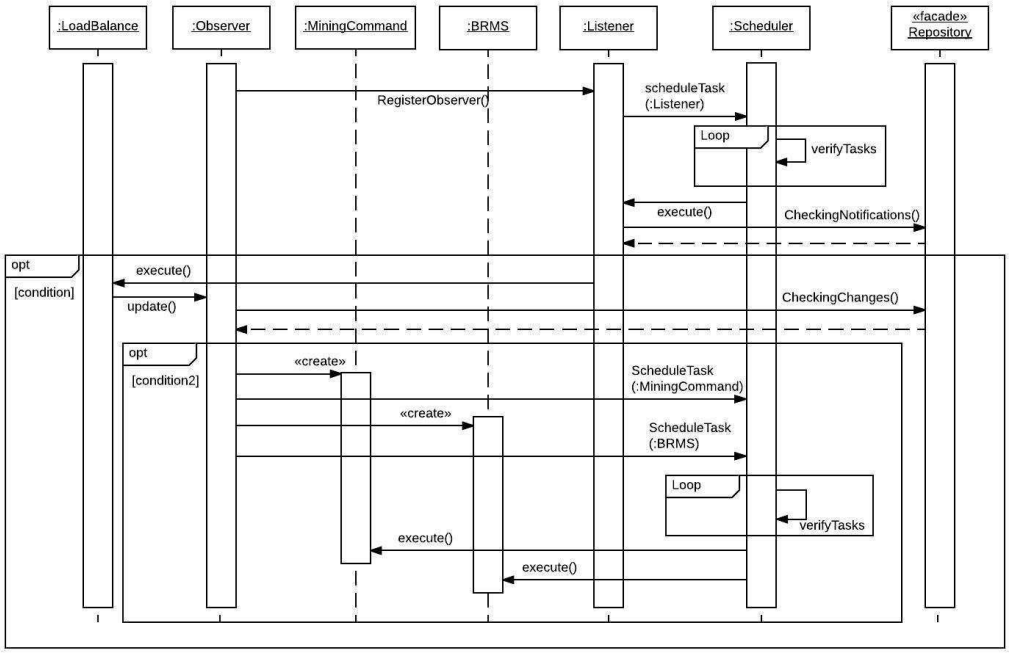


Figura 7.13: Diagrama de Sequência para execução de regras com condições avançadas
Fonte: Elaborada pelo autor

Visão de Implantação

Com base nos possíveis gargalos identificados e a fim de ilustrar a nova infraestrutura, é apresentado na Figura 7.14 o diagrama de implantação UML. Na perspectiva distribuída da arquitetura, a implantação do componente ativo do *data warehouse* pode ser realizada em

dois ou mais servidores de aplicação, adotando um componente ativo distribuído, e a adição de um conector para realizar o balanceamento de carga sobre esse componente.

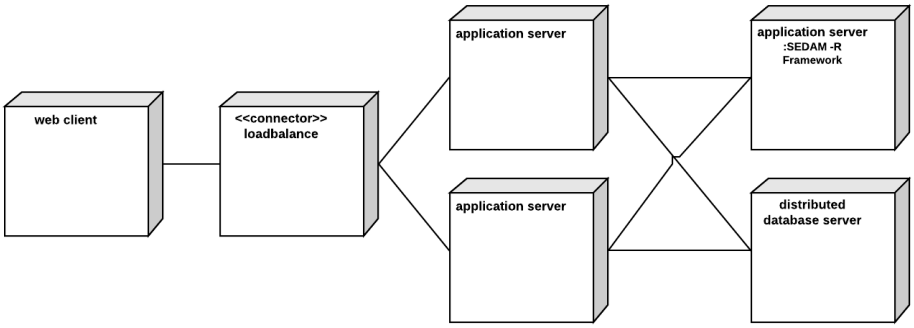


Figura 7.14: Visão de Implantação
Fonte: Elaborada pelo autor

7.11.2 Arquitetura com Distribuição de Regras de Execução (Iteração 3)

Após a segunda iteração, os resultados mostram que houve um aumento do *throughput* do processamento de unidades de execução no ambiente de execução distribuído. Porém, o desempenho de processamento de cada unidade de execução individualmente ainda foi sequencial. Ou seja, cada unidade de execução é responsável por uma determinada quantidade de conjuntos de regras. Vale lembrar que cada unidade de execução pode representar uma disciplina, um curso ou até mesmo um ambiente educacional, dependendo de como o projetista instancia a arquitetura de referência. Na arquitetura da Iteração 2, os conjuntos de regras associados a cada unidade de execução são executados um após o outro, de forma sequencial.

Nesse contexto, a arquitetura proposta passou por um novo processo de refinamento com o objetivo de ofertar uma versão alternativa do distribuidor de carga da versão distribuída da arquitetura de referência. Como as visões lógica (Figura 7.11) e de implantação (Figura 7.14) da arquitetura permaneceram inalteradas, as diferenças implementadas no refinamento são apresentadas através das seguintes visões arquiteturais: visão de desenvolvimento e visão de processos.

Visão de Desenvolvimento

São explicitados no diagrama de componentes UML, representado na Figura 7.15, os componentes envolvidos no processo de refinamento e reestruturação realizados com o objetivo de melhorar o desempenho individual de cada unidade de execução.

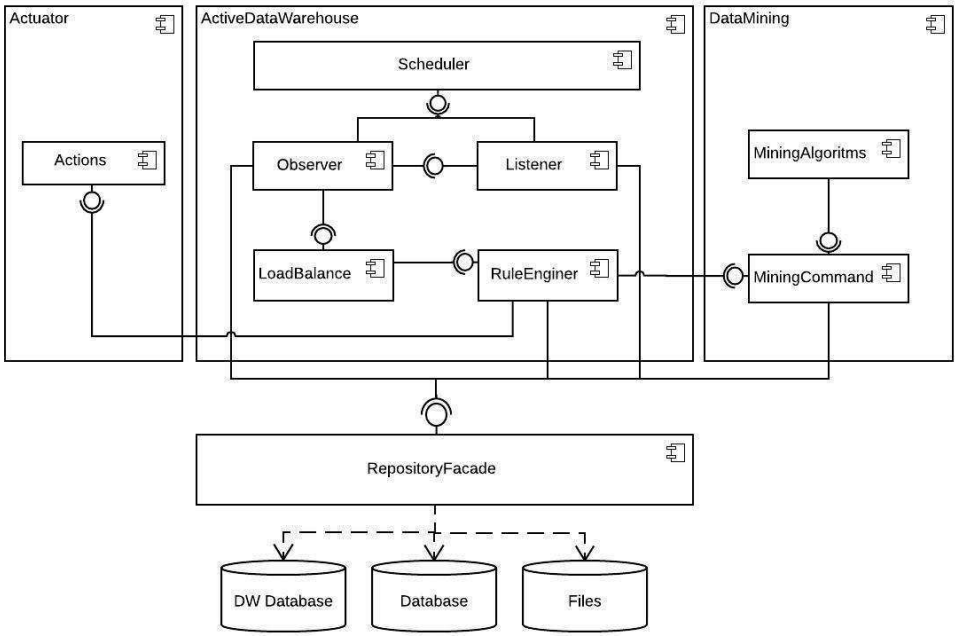


Figura 7.15: Diagrama de Componente
Fonte: Elaborada pelo autor

Em relação ao componente ativo do *data warehouse*, a maior mudança aconteceu no componente *BRMS*, que deixou de existir, dando lugar ao componente *RuleEngine*. O novo componente tem o papel de controlar toda a execução das regras, inclusive o acionamento de ações engatilhadas pelas regras, decorrentes da satisfação de alguma condição pré-definida pelo educador. A outra mudança foi interna ao componente *LoadBalance*, que passou a distribuir regras de execução, ao invés de unidades de execução aos servidores de aplicação.

Visão de Processo

Diante do refinamento arquitetural realizado, a Figura 7.16 apresenta o novo comportamento arquitetural quando regras com condições avançadas (com mineração de dados) são executadas.

Ao ser notificado pelo componente *Listener* que ocorreram mudanças nos dados, **condition**, o *Observer* agenda a sua própria execução para um momento posterior. Ao ser exe-

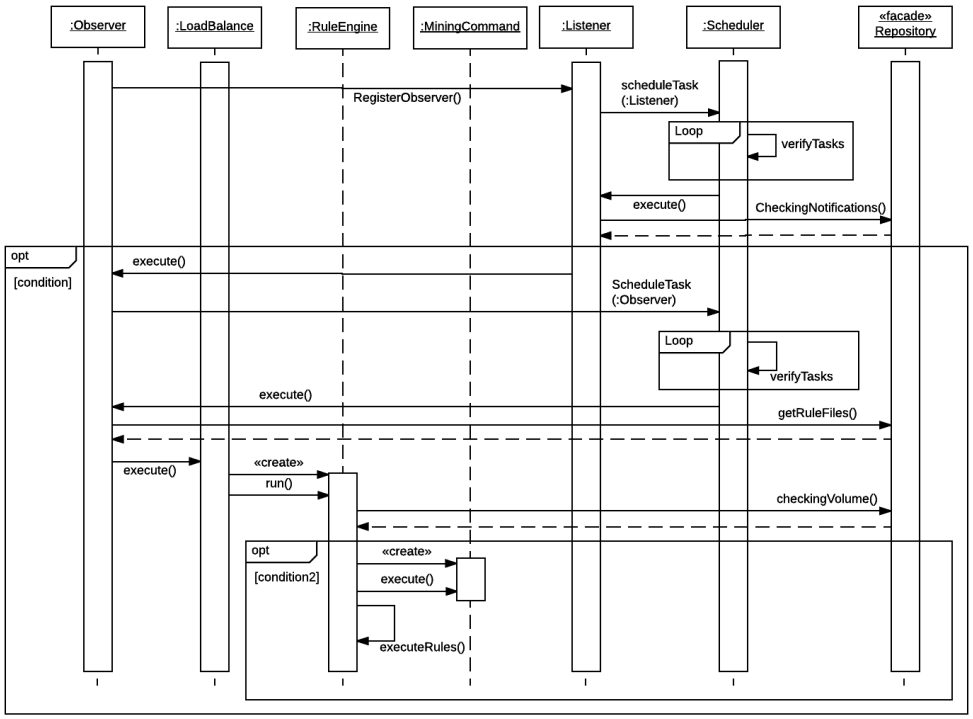


Figura 7.16: Diagrama de sequência da execução de regras com condicionais complexos
Fonte: Elaborada pelo autor

cutado, após o agendamento, o *Observer*, assumindo sua nova função, recupera todos os conjuntos de regras armazenados e aciona o conector *Load Balance*. Esse conector distribui as requisições recebidas entre os servidores disponíveis, enviando um conjunto de regras para o servidor. A cada requisição para execução de um novo conjunto de regras, uma nova instancia do *RuleEngine* é criada para receber esse conjunto de regras.

O novo componente *RuleEngine*, por sua vez, é responsável por gerenciar todo o processo de execução de regras. Assim, a cada requisição recebida, *RuleEngine* verificará se o volume de alterações que ocorreram nos dados são suficientes para o disparo daquele conjunto de regras e caso essa condição seja satisfeita (**condition2**), o componente inicia o processo de execução das regras. Primeiro, o componente recupera os dados relacionados ao conjunto de regras; se necessário, executa os modelos de mineração sobre esses dados; e por fim, aciona a *engine* de regra, passando os dados necessários e as regras que devem ser executadas. Tais evoluções promoveram melhorias no desempenho de cada unidade de execução, preservando a escalabilidade do número de regras executadas num determinado intervalode tempo; requisitos fundamentais à arquitetura de referência proposta para sua instanciação em ambientes

reais de execução.

7.12 Discussão dos Resultados e Limitações

O principal destaque para os resultados alcançados foi o conjunto de decisões de projeto que foram tomadas no decorrer dos refinamentos arquiteturais, optando por oferecer um modelo com um ponto de variação localizado no *LoadBalance*. Dessa forma, são oferecidas duas opções de perspectivas arquiteturais, permitindo ao projetista optar por: (1) uma perspectiva onde a variação arquitetural favorece um maior *throughput* no processamento simultâneo de unidades de execução; ou (2) uma perspectiva com maior desempenho de processamento de unidades de execução individualmente. Desta forma, a responsabilidade da decisão final é atribuída ao arquiteto de software, que pode optar pela perspectiva mais adequada às suas necessidades, aumentando ainda mais a flexibilidade da arquitetura de referência proposta.

Durante a execução dos cenários das Iterações 2 e 3, percebeu-se também um grande número de requisições ao servidor de banco de dados. Foi constatado que o servidor de banco de dados e a camada de acesso aos dados podem vir a se tornar um gargalo arquitetural. Afinal, ambos lidam com uma grande quantidade de requisições e uma grande massa de dados. Além disso, acredita-se que em cenários de alta demanda, como nos casos dos MOOCs, ocorrerá uma maior demanda por escalabilidade nas execuções, a fim de atender um crescente aumento de unidades (e consequentemente de regras) de execução, que demandarão ainda mais de todos os servidores envolvidos. Um dos pontos cruciais a serem observados é a quantidade de dados que ambientes MOOCs manipulam, uma massa consideravelmente maior que as de ambientes educacionais tradicionais. Diante dos potenciais gargalos arquiteturais identificados, a arquitetura de referência propõe um novo refinamento a fim de contorná-los.

Nesse âmbito, a mudança deve ocorrer no componente *Repository Facade* (Figura 6.1). Em casos de utilização com MOOCs, a arquitetura de referência recomenda que esse componente inclua recursos de memória cache temporária com a finalidade de reduzir a demanda por consultas repetitivas vindas de diferentes servidores de aplicação. Além disso, a arquitetura de referência recomenda a utilização de SGBDs distribuídos, que é uma tecnologia presente atualmente nos SGBDs comerciais e até mesmo em alguns SGBDs gratuitos.

Outra decisão de projeto que visou a redução da sobrecarga sobre o SGDB já foi implementada nas duas versões distribuídas da arquitetura de referência. Tal decisão consistiu na refatoração da camada de acesso aos dados, pertencente ao componente *Repository Facade*. Entre as principais alterações realizadas, pode-se destacar o aprimoramento dos comandos SQLs executados, que incluiu a adoção do padrão em *Batch* para atualizar os estudantes classificados após a execução dos modelos de mineração de dados salvos sobre os novos conjuntos de dados. Além disso, também foi adotado um *Pool* de conexões JDBC, a fim de gerenciar a grande quantidade de requisições recebidas pelos SGBDs e controlar o número de conexões simultâneas, implementando uma variação do padrão de projeto *Singleton* que fixa o número máximo de conexões e contém uma fila de espera para as demandas recebidas. Como consequência, a interrupção no acesso ao banco de dados deixou de acontecer, ocorrendo em contrapartida um tempo de espera potencialmente maior para a liberação dos recursos, a depender da demanda de requisições. O número máximo de conexões simultâneas é outro ponto de variação da arquitetura de referência e deve ser definido pelo projetista, de acordo com a capacidade dos recursos de hardware disponíveis e a disponibilidade de bancos de dados distribuídos.

Dentre as limitações e ameaças à avaliação executada, destaca-se a dificuldade em experimentar a arquitetura concreta em um ambiente real, envolvendo um ambiente educacional que possibilitasse acompanhar os discentes durante uma ou mais disciplinas. Outra limitação está relacionada ao fato dos experimentos terem sido executados em um ambiente controlado e executados pelo próprio autor. Hardware também pode ser apontado como limitação, pois não se conseguiu acesso a um servidor, onde as simulações pudessem ocorrer mais próximo de um cenário real; sendo os experimentos executados em PCs convencionais e com configurações modestas, quando comparadas a servidores.

7.13 Revisitando as Questões de Pesquisa

A fim de responder as questões de pesquisas que nortearam esta Tese, estas serão reapresentadas a seguir, bem como, as discussões acerca de cada uma delas.

- Como oferecer a desenvolvedores de software apoio para o desenvolvimento de plataformas de *Business Intelligence* educacionais integradas a ambientes de *e-learning*,

com alta escalabilidade e que contemple automação de ações pedagógicas?

  - Esse apoio foi oferecido em duas perspectivas distintas: (1) do ponto de vista conceitual, através da disponibilidade de uma arquitetura de referência, que permite reutilizar soluções de projeto para atender os requisitos de qualidade necessários ao domínio; e (2) do ponto de vista técnico, através da disponibilidade de um arcabouço de software baseado em componentes que realiza a arquitetura de referência e preserva seus pontos de variação, facilitando a rastreabilidade entre as decisões de projeto arquiteturais e as suas respectivas realizações em código-fonte.

- Qual a redução no custo de desenvolvimento, com base na porcentagem de reuso de software, conseguido com a utilização do arcabouço proposto para plataformas de *Business Intelligence* educacionais integradas a ambientes de *e-learning*?

  - Como apresentado na Seção 7.9, a utilização da arquitetura de referência, agregada ao arcabouço proposto, proporcionou um índice de reuso de aproximadamente 88% de código. Em outras palavras, cerca de 88% do arcabouço desenvolvido reflete partes fixas da arquitetura de referência (*frozen spots*), enquanto apenas 12% foi considerado específico de tecnologia (*hot spots*). Exemplos de ajustes realizados foram: código de integração ao ambiente de educação *online* adotado no experimento, código de integração ao SGBD utilizado, integração às ferramentas de BI utilizadas e a definição de regras específicas feitas para o curso alvo do experimento.

- Como facilitar a adição e evolução de novas soluções de *Business Intelligence* educacionais de maneira simplificada, para descoberta de informações no contexto educacional?

  - A flexibilidade no tocante à evolução do ambiente de BI educacional foi efetivada de duas formas complementares: (1) a existência de um ponto de variação que possibilita a integração com diferentes ferramentas de BI, tais como Pentaho e o *framework* SEDAM-R, tornam a arquitetura de referência mais flexível a diferentes tecnologias; e (2) a adoção de serviços semânticos, por parte do *framework*

SEDAM-R, que facilita a adição e utilização de novos serviços de mineração, pré-processamento ou pós-processamento de dados, sem a necessidade sequer de reinicialização do servidor.

Capítulo 8

Conclusões

Neste trabalho foi proposta uma arquitetura de referência realizada por um arcabouço de software com vistas a oferecer aos projetistas e desenvolvedores de sistemas educacionais uma infraestrutura apropriada para construção de soluções de *Business Intelligence*, focalizando nos processos de obtenção de informação e seu uso no apoio à tomada de decisões, orientadas por dados, em contextos educacionais. A solução desenvolvida apoia o desenvolvimento de três formas complementares: (1) favorece o projeto arquitetural do software, de modo a satisfazer requisitos de qualidade fundamentais para esse tipo de sistema, tais como escalabilidade e facilidade de evolução; (2) favorece o reúso de código-fonte, aliado a padrões de projeto; (3) favorece a integração com ferramentas existentes, tais como *frameworks* e ambientes virtuais de aprendizagem.

Tendo em conta os resultados obtidos nesta tese, pode-se afirmar que as questões de pesquisa e os associados objetivos específicos foram respondidos do seguinte modo. Com respeito à questão de pesquisa 1 e os objetivos I, III e IV considera-se que todos foram respondidos satisfatoriamente através da disponibilidade de uma arquitetura de referência e da disponibilidade de um arcabouço de software baseado em componentes que realizam a arquitetura de referência e preservam seus pontos de variação, conforme detalhado na Seção 7.13. Com relação às questões de pesquisa 2 e 3 e os objetivos I e II, esses também foram atendidos satisfatoriamente. No tocante a redução no custo de desenvolvimento, foi atendida através da utilização da arquitetura de referência, agregada ao arcabouço proposto que proporcionou um índice de reuso de aproximadamente 88% de código. Já em relação à evolução do ambiente de BI educacional, foi efetivada de duas formas complementares:

através da existência de um ponto de variação que possibilita a integração com distintas ferramentas de BI e da adoção de serviços semânticos, por parte do *framework* SEDAM-R, conforme explicitado na Seção 7.13.

8.1 Contribuições

As principais contribuições desta Tese são apontadas a seguir:

- **Arquitetura de Referência para a tomada de decisão em contextos educacionais** - essa relevante contribuição consistiu em uma arquitetura de referência realizada por um arcabouço de software, que serve como ponto de partida para a instanciação de arquiteturas de software e sistemas que reúnam a infraestrutura necessária para que professores e os demais *stakeholders* envolvidos possam realizar o processo de tomada de decisão em contextos educacionais.

- **Implementação da Arquitetura de Referência** - essa contribuição pode ser classificada como uma contribuição técnica que favoreceu diretamente o detalhamento e refinamento da arquitetura de referência mencionada anteriormente. No contexto desse refinamento, foi possível identificar padrões de projetos e ferramentas existentes que puderam ser integradas à arquitetura, tais como, o *framework* Drools (BROWNE, 2009), a ferramenta de BI Pentaho (PENTAHO, 2016), os algoritmos de mineração de dados do Weka (WITTEN et al., 2016) e o ambiente virtual de aprendizagem Moodle (MOODLE, 2017). Além disso, destaca-se a construção de um conjunto de classes definidas que permite a manipulação de regras do Drools em tempo de execução.

- **Modelos de regras avançadas** - é proposto nesta tese um novo modelo de regras para o *data warehouse* ativo. Esse modelo possibilita a utilização de informações não presentes nos bancos de dados, mas que possam ser descobertas a partir deles, utilizando técnicas como a mineração de dados. Esse modelo estende as *analysis rules*, inicialmente proposta por Thalhammer, Schrefl e Mohania (2001), que utilizam um conjunto de condições para realizar análises multidimensionais que anteriormente eram realizadas manualmente por analistas. No modelo proposto nesta tese, as condições das regras utilizam o resultado da aplicação de modelos de mineração sobre novos dados.

Deste modo, antes de analisar se os condicionais de uma regra foram satisfeitos, o DW ativo realiza os procedimentos necessários à aplicação de modelos de mineração, recupera o seu resultado e verifica se as condições das regras foram satisfeitas, antes de executar as devidas ações.

- **SEDAM-R** (MARINHO et al., 2017) - O *framework* para mineração de dados apresentado nesta tese, que é acoplado à arquitetura de referência proposta, é uma contribuição chave para as necessidades e demandas da área da mineração de dados educacionais apontadas na literatura por diversos autores (C.ROMERO; S.VENTURA, 2007; C.ROMERO; VENTURA, 2010; ROMERO; S.VENTURA, 2013). As vantagens que justificam a adoção do *SEDAM-R* incluem: extensão facilitada para adição de novos serviços de mineração, integração entre sistemas simplificada, flexibilidade, entre outras. Além disso, através do SEDAM-R é possível configurar o processo de descoberta de conhecimento sem que o usuário precise conhecer detalhes técnicos dos algoritmos utilizados; uma vez que toda a configuração acontece de forma semi-automática, a partir dos objetivos indicados pelo professor/educador.

- **Diferentes perspectivas arquiteturais** - A arquitetura de referência proposta oferece ao projetista um modelo único com duas opções de perspectivas arquiteturais. Na primeira perspectiva, a variação arquitetural favorece um *throughput* maior no processamento de unidades de execução. Na segunda, uma perspectiva que favorece um maior desempenho de processamento de unidades de execução para cenários de alta demanda. Desta forma, o projetista pode optar pela perspectiva adequada às suas necessidades, tornando a arquitetura de referência proposta ainda mais flexível.

- **Método de refinamento para projetos de arquiteturas de referência** - Baseado no ATAM, esse método constitui uma contribuição importante no processo de avaliação e refinamento arquitetural com foco na detecção de gargalos arquiteturais e *trade-offs*, guiando projetistas durante todo esse processo.

As realizações desta tese constituem um passo relevante e uma contribuição significativa no caminho de uma solução plena para um *Business intelligence* educacional e, consequentemente, trazem contribuições para a área de Informática na Educação, numa perspectiva

interdisciplinar envolvendo principalmente aspectos de engenharia de software, além de aspectos de banco de dados e inteligência artificial.

8.2 Limitações e Trabalhos Futuros

Apesar da solução proposta nesta tese apresentar avanços importantes, não é ainda uma solução definitiva para BI Educacional. É necessário oferecer uma interface simplificada para que o professor/educador possa realizar o processo de tomada de decisão de modo integrado dentro dessas plataformas. Além disso, é desejável que o professor/educador possa realizar o processo de descoberta de conhecimento sem a necessidade de escolher algoritmos de mineração ou definir seus parâmetros.

Como trabalhos futuros, pretende-se avaliar outros atributos de qualidade, tais como manutenibilidade e interoperabilidade, entre outros. Também é prevista a adição de novas funcionalidades e a evolução da solução para incluir novos algoritmos de mineração. Além disso, pretende-se avaliar a arquitetura de referência em um cenário real de utilização, a fim de envolver diretamente professores e os demais atores envoltos. Por fim, pretende-se avaliar a qualidade e efetividade dos modelos de mineração e das intervenções realizadas.

Bibliografia

ALCALA-FDEZ, J. et al. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, Springer, v. 13, n. 3, p. 307–318, 2009.

ANDREWS, T. et al. *Business process execution language for web services*. [S.l.]: version, 2003.

ANGELOV, S.; GREFEN, P.; GREEFHORST, D. A classification of software reference architectures: Analyzing their success and effectiveness. In: IEEE. *Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on*. [S.l.], 2009. p. 141–150.

ANGELOV, S.; TRIENEKENS, J.; GREFEN, P. Towards a method for the evaluation of reference architectures: Experiences from a case. *Software architecture*, Springer, p. 225–240, 2008.

BAKER, R. et al. Data mining for education. *International encyclopedia of education*, Oxford, UK: Elsevier, v. 7, n. 3, p. 112–118, 2010.

BAKER, R. S.; INVENTADO, P. S. Educational data mining and learning analytics. In: ____. *Learning Analytics: From Research to Practice*. New York, NY: Springer New York, 2014. p. 61–75. ISBN 978-1-4614-3305-7. Disponível em: ⟨https://doi.org/10.1007/978-1-4614-3305-7_4⟩.

BAKHARIA, A. et al. A conceptual framework linking learning design with learning analytics. In: ACM. *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*. [S.l.], 2016. p. 329–338.

BARROS, H. J. S. *Um Modelo Semântico para Compartilhamento de Recursos Educacionais*. Tese (Doutorado) — Federal University of Campina Grande, 2016.

BASILI, V. R. *Software modeling and measurement: the Goal/Question/Metric paradigm*. [S.l.], 1992.

BASS, L.; CLEMENTS, P.; KAZMAN, R. *Software Architecture in Practice*. Second edition. [S.l.]: Addison Wesley, 2003. ISBN 0-321-15495-9.

BASS, L.; CLEMENTS, P.; KAZMAN, R. *Software Architecture in Practice*. [S.l.]: Addison-Wesley Professional, 2003.

BEYER, H.-J. et al. Introducing architecture-centric reuse into a small development organization. In: ____. *High Confidence Software Reuse in Large Systems: 10th International Conference on Software Reuse, ICSR 2008, Beijing, China, May 25-29, 2008 Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 1–13. ISBN 978-3-540-68073-4. Disponível em: ⟨https://doi.org/10.1007/978-3-540-68073-4_1⟩.

BOSCH, J. Software product lines: Organizational alternatives. In: *Proceedings of the 23rd International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2001. (ICSE '01), p. 91–100. ISBN 0-7695-1050-7. Disponível em: ⟨http://dl.acm.org/citation.cfm?id=381473.381482⟩.

BOSE, I.; HUI, B. A case based analysis of active data warehousing projects using the logical framework method. *IT Professional*, IEEE, n. 1, p. 1, 2013.

BOUATTOUR, S. et al. A framework for active data warehouses. In: *The 2008 International Arab Conference on Information Technology (ACIT'2008), Hammamet, Tunisie*. [S.l.: s.n.], 2009.

BRITO, P. H. S. et al. Architecting fault tolerance with exception handling: Verification and validation. *J. Comput. Sci. Technol.*, v. 24, n. 2, p. 212–237, 2009. Disponível em: ⟨https://doi.org/10.1007/s11390-009-9219-2⟩.

BROWNE, P. *JBoss Drools business rules*. [S.l.]: Packt Publishing Ltd, 2009.

BUSCHMANN, F. et al. *Pattern-Oriented Software Architecture - Volume 1: A System of Patterns*. [S.l.]: Wiley Publishing, 1996. ISBN 0471958697, 9780471958697.

CENSO, E. Br 2015- analytic report of distance learning in brazil 2015. 2015.

CHATTI, M. A.; MUSLIM, A.; SCHROEDER, U. Toward an open learning analytics ecosystem. In: *Big Data and Learning Analytics in Higher Education*. [S.l.]: Springer, 2017. p. 195–219.

CLEMENTS, P. et al. *Documenting Software Architectures: Views and Beyond*. [S.l.]: Addison-Wesley, 2003.

C.ROMERO; S.VENTURA. Educational data mining: A survey from 1995 to 2005. *Expert systems with applications*, Elsevier, v. 33, n. 1, p. 135–146, 2007.

C.ROMERO; VENTURA, S. Educational data mining: a review of the state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, IEEE, v. 40, n. 6, p. 601–618, 2010.

ELMASRI, R.; NAVATHE, S. B. *Fundamentals of database systems*. [S.l.]: Pearson, 2015.

FALAKMASIR, M. H. et al. Business intelligence in e-learning:(case study on the iran university of science and technology dataset). In: IEEE. *Software Engineering and Data Mining (SEDM), 2010 2nd International Conference on*. [S.l.], 2010. p. 473–477.

FEITOSA, D. *Simus-uma arquitetura de referência para sistemas multirrobóticos de serviço*. Tese (Doutorado) — Universidade de São Paulo, 2013.

FRANCE, R.; RUMPE, B. Model-driven development of complex software: A research roadmap. In: *2007 Future of Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2007. (FOSE '07), p. 37–54. ISBN 0-7695-2829-5. Disponível em: ⟨http://dx.doi.org/10.1109/FOSE.2007.14⟩.

GALLAGHER, B. P. *Using the architecture tradeoff analysis methodsm to evaluate a reference architecture: a case study*. [S.l.], 2000.

GAMMA, E. *Design patterns: elements of reusable object-oriented software*. [S.l.]: Pearson Education India, 1995.

GRAAF, B.; DIJK, H. V.; DEURSEN, A. V. Evaluating an embedded software reference architecture-industrial experience report. In: IEEE. *Software Maintenance and Reengineering, 2005. CSMR 2005. Ninth European Conference on*. [S.l.], 2005. p. 354–363.

HAJLAOUI, J. E.; HAMDANI, N. Active data warehouse: Review, challenges and issues. In: IEEE. *Computer Applications & Research (WSCAR), 2014 World Symposium on*. [S.l.], 2014. p. 1–6.

INMON, W. H. *Building the data warehouse*. [S.l.]: John wiley & sons, 2002.

JAYAPRAKASH, S. M. et al. Early alert of academically at-risk students: An open source analytics initiative. *Journal of Learning Analytics*, v. 1, n. 1, p. 6–47, 2014.

KAMPFF, A. J. C. Mineração de dados educacionais para geração de alertas em ambientes virtuais de aprendizagem como apoio à prática docente. 2009.

KAZMAN, R.; KLEIN, M.; CLEMENTS, P. *ATAM: Method for architecture evaluation*. [S.l.], 2000.

KHODEIR, N. et al. Inferring the differential student model in a probabilistic domain using abduction inference in bayesian networks. In: *In Educational Data Mining*. [S.l.: s.n.], 2010. p. 299–300.

KIM, S. M.; CALVO, R. A. Sentiment analysis in student experiences of learning. In: *In Educational Data Mining*. [S.l.: s.n.], 2010. p. 111–120.

KIMBALL, R.; ROSS, M. *The data warehouse toolkit: the complete guide to dimensional modeling*. [S.l.]: John Wiley & Sons, 2013.

KRUCHTEN, P. The 4+1 view model of architecture. *IEEE Softw.*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 12, n. 6, p. 42–50, nov. 1995. ISSN 0740-7459. Disponível em: ⟨http://dx.doi.org/10.1109/52.469759⟩.

LEITNER, F. Automotive open system architecture. In: *Seminar Software Engineering for Automotive Systems*. [S.l.: s.n.], 2007.

LILLIEFORS, H. W. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American statistical Association*, Taylor & Francis Group, v. 62, n. 318, p. 399–402, 1967.

LIÑÁN, L. C.; PÉREZ, Á. A. J. Educational data mining and learning analytics: differences, similarities, and time evolution. *RUSC. Universities and Knowledge Society Journal*, v. 12, n. 3, p. 98–112, 2015.

MACFADYEN, L.; SORENSON, P. Using lims (the learner interaction monitoring system) to track online learner engagement and evaluate course design. In: *In Educational Data Mining*. [S.l.: s.n.], 2010. p. 301–302.

MACKENZIE, C. et al. *OASIS Reference Model for Service Oriented Architecture 1.0. Committee Specification 1*. 2006.

MARINHO, T. et al. An ontology-based software framework to provide educational data mining. In: ACM. *Proceedings of the 2010 ACM Symposium on Applied Computing*. [S.l.], 2010. p. 1433–1437.

MARINHO, T. et al. An approach for automatic discovery, composition and invocation of semantics web services for data mining. In: *16th Mexican Conference (International) on Artificial Intelligence Proceedings*. [S.l.: s.n.], 2017.

MCILRAITH, S. A.; SON, T. C.; ZENG, H. Semantic web services. *IEEE intelligent systems*, IEEE, n. 2, p. 46–53, 2001.

MIERSWA, I. et al. Yale: Rapid prototyping for complex data mining tasks. In: ACM. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2006. p. 935–940.

MOODLE. *Moodle project*. 2017. ⟨https://moodle.org/⟩. Accessed: 2017-02-05.

MULLER, G. A reference architecture primer. *Eindhoven Univ. of Techn., Eindhoven, White paper*, 2008.

NAGATA, R. et al. Edu-mining for book recommendation for pupils. In: *In Educational Data Mining*. [S.l.: s.n.], 2009. p. 91–100.

NAKAGAWA, E. Y. et al. An aspect-oriented reference architecture for software engineering environments. *Journal of Systems and Software*, Elsevier, v. 84, n. 10, p. 1670–1684, 2011.

NAKAGAWA, E. Y. et al. Consolidating a process for the design, representation, and evaluation of reference architectures. In: IEEE. *Software Architecture (WICSA), 2014 IEEE/IFIP Conference on*. [S.l.], 2014. p. 143–152.

NAKAGAWA, E. Y. et al. Towards a process to design aspect-oriented reference architectures. In: *Proceedings of the XXXV Latin American Informatics Conference (CLEI 2009)*. [S.l.: s.n.], 2009. p. 1–10.

NAKAGAWA, E. Y.; OQUENDO, F.; MALDONADO, J. C. Reference architectures. *Software Architecture 1*, Wiley Online Library, p. 55–82, 2014.

PALANIVEL, K.; KUPPUSWAMI, S. Service-oriented reference architecture for personalized e-learning systems (sorapes). *International Journal of Computer Applications*, International Journal of Computer Applications, 244 5 th Avenue,# 1526, New York, NY 10001, USA India, v. 24, n. 5, p. 35–44, 2011.

PEÑA-AYALA, A. *Educational Data Mining: Applications and Trends*. [S.l.]: Springer, 2013. v. 524.

PEÑA-AYALA, A. Educational data mining: A survey and a data mining-based analysis of recent works. *Expert systems with applications*, Elsevier, v. 41, n. 4, p. 1432–1462, 2014.

PENTAHO. *Pentaho: Data Integration, Business Analytics and Big Data*. 2016. ⟨http://www.pentaho.com⟩. Accessed: 2016-08-14.

RANKA, A.; ANWAR, F.; CHAE, H. S. Pundit: Intelligent recommender of courses. In: *In Educational Data Mining*. [S.l.: s.n.], 2010. p. 339–340.

ROMERO, C.; S.VENTURA. Data mining in education. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Wiley Online Library, v. 3, n. 1, p. 12–27, 2013.

ROMERO, C.; VENTURA, S. Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications*, v. 33, n. 1, p. 135 – 146, 2007. ISSN 0957-4174. Disponível em: ⟨http://www.sciencedirect.com/science/article/B6V03-4JW10WR-2/2/7df5551b2738c1402738eece307e6c61⟩.

ROMERO, C.; VENTURA, S. Educational data science in massive open online courses. *WILEY INTERDISCIPLINARY REVIEWS-DATA MINING AND KNOWLEDGE DISCOVERY*, WILEY PERIODICALS, INC ONE MONTGOMERY ST, SUITE 1200, SAN FRANCISCO, CA 94104 USA, v. 7, n. 1, 2017.

SACÍN, C. V. et al. Recommendation in higher education using data mining techniques. In: *In Educational Data Mining*. [S.l.: s.n.], 2009. p. 191–199.

SANTOS, J. F. M. et al. A checklist for evaluation of reference architectures of embedded systems (s). In: *SEKE*. [S.l.: s.n.], 2013. v. 13, p. 1–4.

SCHREFL, M.; THALHAMMER, T. On making data warehouses active. In: *Data Warehousing and Knowledge Discovery*. [S.l.]: Springer, 2000. p. 34–46.

SCLATER, N.; BERG, A.; WEBB, M. Developing an open architecture for learning analytics. *EUNIS Journal of Higher Education*, EUNIS, 2015.

SEI, S. E. I. *Architecture Tradeoff Analysis Method*. 2017. ⟨htt://www.sei.cmu.edu/architecture/tools/evaluate/atam.cfm⟩. Accessed: 2016-12-14.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). *Biometrika*, JSTOR, v. 52, n. 3/4, p. 591–611, 1965.

SHAW, M.; GARLAN, D. *Software Architecture: Perspectives on an Emerging Discipline*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996. ISBN 0-13-182957-2.

SIEMENS, G. et al. Open learning analytics: an integrated & modularized platform. 2011.

TANG, C. et al. Personalized courseware construction based on web data mining. In: *WISE '00: Proceedings of the First International Conference on Web Information Systems Engineering (WISE'00)-Volume 2*. Washington, DC, USA: IEEE Computer Society, 2000. p. 2204. ISBN 0-7695-0577-5-2.

TANG, T.; MCCALLA, G. Smart recommendation for an evolving e-learning system. *Workshop on Technologies for Electronic Documents for Supporting Learning, International Conference on Artificial Intelligence in Education*, A, p. 699–710, 20-24 July 2003.

TANG, T. Y.; MCCALLA, G. Student modeling for a web-based learning environment: a data mining approach. In: *Eighteenth national conference on Artificial intelligence*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 2002. p. 967–968. ISBN 0-262-51129-0.

TERADATA. Teradata active enterprise data warehouse 6800. ⟨http://assets.teradata.com/resourceCenter/downloads/Datasheets/EB7080.pdf⟩. 2015.

TERADATA. *Teradata Active Enterprise Data Warehouse*. 2017. ⟨http://www.teradata.com/products-and-services/enterprise-data-warehouse/⟩. Accessed: 2017-02-01.

THALHAMMER, T.; SCHREFL, M.; MOHANIA, M. Active data warehouses: complementing olap with analysis rules. *Data & Knowledge Engineering*, Elsevier, v. 39, n. 3, p. 241–269, 2001.

THO, M. N.; TJOA, A. M. Zero-latency data warehousing for heterogeneous data sources and continuous data streams. In: *5th International Conference on Information Integrationand Web-based Applications Services*. [S.l.: s.n.], 2003. p. 55–64.

UFAL, U. F. de A. *UFAL em Números*. 2017. ⟨http://www.ufal.edu.br/transparencia/ufal-em-numeros⟩. Accessed: 2017-12-20.

VASILECAS, O.; SMAIZYS, A. Business rule based data analysis for decision support and automation. In: CITESEER. *International Conference on Computer Systems and Technologies*. [S.l.], 2006. v. 2.

WILCOXON, F. Individual comparisons by ranking methods. *Biometrics bulletin*, JSTOR, v. 1, n. 6, p. 80–83, 1945.

WITTEN, I. H. et al. *Data Mining: Practical machine learning tools and techniques*. [S.l.]: Morgan Kaufmann, 2016.

ZAANE, E. O.; ZAïANE, O. R. Web usage mining for a better web-based learning. *A*, A, p. 60–64, 2001.

ZAIANE, O. R. Building a recommender agent for e-learning systems. *A*, IEEE Computer Society, Washington, DC, USA, A, p. 55, 2002.

ZORRILLA, M.; GARCÍA, D.; ÁLVAREZ, E. A decision support system to improve e-learning environments. In: *Proceedings of the 2010 EDBT/ICDT Workshops*. New York, NY, USA: ACM, 2010. (EDBT '10), p. 11:1–11:8. ISBN 978-1-60558-990-9. Disponível em: ⟨http://doi.acm.org/10.1145/1754239.1754252⟩.

ZORRILLA, M. E. Data warehouse technology for e-learning. In: *Methods and Supporting Technologies for Data Analysis*. [S.l.]: Springer, 2009. p. 1–20.

Apêndice A

Protocolo de Revisão de Literatura

A.1 Objetivo

O objetivo desta revisão foi caracterizar estudos primários relacionados ao uso de técnicas/ferramentas no apoio a tomada de decisões pedagógicas em ambientes educacionais.

A.2 Questões de Pesquisa

Com base no objetivo, foram definidas as seguintes questões de pesquisa que nortearam essa revisão, são elas:

- QP1: Quais as principais soluções utilizadas no apoio a tomada de decisões pedagógicas em ambientes educacionais?
- QP2: Quais técnicas/tecnologias são empregadas no desenvolvimento de soluções para tomada de decisões pedagógicas em ambientes educacionais?

A.3 Estratégias de busca

Inicialmente, foi realizada uma investigação dos principais autores e coautores das áreas de pesquisa envolvidas a fim de identificar os principais termos chaves. Seguidamente, foram definidos os termos chaves para a busca, seus sinônimos, as bases de dados utilizadas e os

parâmetros para busca manual. Os termos definidos foram: *Learning Management Systems* e *pedagogical decision-making*. Na construção da *String* de busca, sinônimos das palavras chaves identificadas foram combinados com o objetivo de tornar a *String* de busca mais abrangente.

Na execução da busca foram levantados, preferencialmente, trabalhos na língua inglesa por ser um idioma, normalmente, adotado internacionalmente para área foco desta revisão. No entanto, trabalhos relevantes na língua portuguesa também foram levados em consideração.

Por fim, foi realizada uma busca manual com base nos trabalhos relacionados presentes nas referencias dos trabalhos identificados, investigação dos principais autores e coautores das áreas de pesquisa envolvidas, e identificados a partir dos trabalhos levantados.

A.4 Expressão Geral de Busca

Com base nas definições da etapa anterior, a *String* geral de busca definida foi:

("e-learning"OR "Learning Management Systems") AND ("pedagogical decision-making"OR "pedagogical decision making"OR "Educational Decision Support System"OR "educational decision-making")

Em seguida, foram definidas as estratégias de busca específicas:

- **ACM Digital Library e IEEEXplore:** *("e-learning"OR "LMS"OR "Learning Management Systems") AND ("pedagogical decision-making"OR "pedagogical decision making")*
- **Scorpus:** *TITLE-ABS-KEY ((("e-learning"OR "LMS"OR "Learning Management Systems") AND ("pedagogical decision-making"OR "decision-making"OR "decision making"OR "pedagogical decision making"))) AND PUBYEAR > 1980 AND PUBYEAR < 2018 AND (LIMIT-TO (SUBJAREA , "COMP"))*

A.5 Critérios de inclusão e exclusão

Abaixo são apresentados os critérios de inclusão e exclusão adotados na execução dessa revisão:

Critérios de Inclusão (CI)

- Trabalhos que apresentam projeto ou desenvolvimento de ferramentas destinadas ao apoio a tomada de decisões pedagógicas em ambientes educacionais;
- Trabalhos que apresentam aplicações que podem ser utilizadas no apoio a tomada de decisões pedagógicas em ambientes educacionais;

Critérios de Exclusão (CE)

- Trabalhos com soluções ou propostas que não sejam direcionadas ao contexto educacional;
- Trabalhos com soluções ou propostas que sejam direcionadas a plataforma *Mobile*;
- Trabalhos ou projetos de desenvolvimento de ambientes educacionais;
- Trabalhos que apresentam estudos preliminares e resumos em conferencias;
- Trabalhos duplicados;