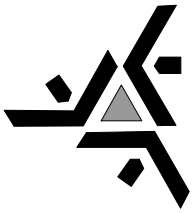

Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Engenharia de Produção

Definição de uma Especificação de Arquitetura de Software para o Domínio Educacional de Pesquisa Operacional

José Roberto Vasconcelos

TCC-EP-48-2010

Maringá - Paraná
Brasil

Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Engenharia de Produção

Especificação de Um Ambiente Computacional de Apoio ao Ensino de Pesquisa Operacional

José Roberto Vasconcelos

TCC-EP-48-2010

Trabalho de Conclusão de Curso apresentado como requisito de avaliação do Trabalho de Conclusão de Curso do curso de graduação em Engenharia de Produção na Universidade Estadual de Maringá – UEM.
Orientadora: Profª. Drª. Márcia M. Altimari Samed

Maringá - Paraná
2010

AGRADECIMENTOS

Primeiramente a Deus por ter me dado forças para chegar até aqui.

À minha mãe que me proporcionou a base para superar dificuldades, dando-me subsídios para a vida.

A todos os professores da Universidade Estadual de Maringá, em especial aos que contribuíram para minha formação em Engenharia.

Aos amigos do curso de Engenharia de Produção, e em especial da ênfase de Construção Civil, turma de 2006.

RESUMO

Este trabalho visa identificar e especificar componentes direcionados ao ensino de Pesquisa Operacional, por meio de um embasamento teórico sobre arquitetura de software e o modelo de referencia da IEEE/LTSA. Assim, o intuito e que este trabalho seja um guia para a construção de ambientes educacionais, tornando a dificuldade de identificar requisitos para especificação de ambientes mais amigáveis, em que o foco seja na formulação de conteúdos educacionais.

O trabalho proposto procurou fornecer conceitos relevantes para o desenvolvimento de ambientes baseado em componentes/objetos educacionais, em especial para um domínio específico. A identificação de componentes foi baseada em ferramentas existentes e na verificação de necessidades do ensino. O modelo proposto pode ser aplicado para especificar um método de ensino tradicional, ou com a interação de recursos computacional, que neste caso pode ser com a integração de ferramentas existentes, ou com o desenvolvimento de componentes específicos (dedicados) para o ambiente.

Palavras-chave: Arquitetura de Software, Componentes, Pesquisa Operacional, Ambiente Educacional.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ADL	*Advanced Distributed Learning*
AOC	Acoplamento entre Objetos das Classes
COM	*Component Object Model*
CORBA	*Common Object Request Broker Architeture*
FCM	Falta de Correção dos Métodos
GUI	*Graphic User Interface*
HTML	*HyperText Markup Language*
IDL	*Interface Definition Language*
IEC	*International Electrotechnical Commission*
IEEE	*Institute of Electrical and Electronics Engineers*
IMS	*Instructional Management System*
ISO	*International Organization for Standardization*
LOM	*Learning Objects Metadata*
LSM	*Learning Management System*
LTSA	*Learning Technology Systems Architecture*
LTSC	*Learning Technology Standard Committee*
MPC	Métodos Ponderados por Classe
ORB	*Object Request Broker*
PAH	Profundidade da Árvore de Herança
PHP	*Personal Home Page*
PL	Programação Linear
PO	Pesquisa Operacional
SOM	*System Object Model*
SQL	*Structured Query Language*
W3C	*World Wide Consortium*
WCOP	*Workshop on Component-Oriented Programming*
WWW	*World Wide Web (*ou simplesmente *Web*)
XML	*eXtensible Markup Language*

1 INTRODUÇÃO

As diferentes formas de conceber o ensino e a aprendizagem têm levado profissionais e pesquisadores da área educacional a constantes buscas de modos mais efetivos de promover a relação entre educador/professor, aluno e objeto de conhecimento. O processo de ensino e aprendizagem, seguindo as divergentes tendências pedagógicas, ora centrado no professor ou no aluno/aprendiz, ora nas técnicas ou conteúdos de ensino, tem sido receptivo às influências construtivistas, principalmente as fundamentadas a partir de teorias clássicas da forma de aprendizagem (Ribeiro, 2004).

Atualmente, a sala de aula não é mais a mesma, a tecnologia, principalmente a informática, passa a fazer parte do cotidiano dos alunos. Com isto o interesse no uso de computadores em aprendizagem e na educação tem motivado o desenvolvimento de ambientes educacionais, mas para isto tornar um caminho eficiente no processo de aprendizagem faz-se necessário a utilização de mecanismos que auxiliem no processo de elaboração de aplicações direcionadas ao domínio educacional. Uma abordagem, que procura minimizar esses problemas, é a utilização de arquiteturas de software para um domínio específico. A importância de arquitetura de software para aplicações educacionais foi reconhecida pelo *Institute of Electrical and Electronics Engineers* (IEEE), que propôs como parte de seus esforços de padronização, a definição de um modelo de referência *Learning Technology Systems Architecture* (LTSA) (IEEE P1484.1/D9, 2001).

No enfoque do aprendizado, a tríade dialética faz parte de um processo contínuo na construção do conhecimento: as estruturas atingidas no primeiro nível “intra” dão lugar, às análises do nível “inter” seguinte, e estas, por sua vez, à produção da estrutura “trans”. Os níveis de conhecimento correspondem sucessivamente às seguintes perguntas em relação ao objeto da aprendizagem: “o que é isso?”, “como funciona?” e “como se explica?”. Cada questionamento busca respostas que se pretende alcançar e desenvolver, podendo ocorrer através de atividades variadas propostas ao aluno. Mesmo cientes de que o raciocínio piagetiano da tríade dialética está relacionado com os processos de desenvolvimento da aprendizagem, hipoteticamente, busca-se associá-la às etapas da aprendizagem com o uso dos recursos tecnológicos nos processos educativos. Neste intuito, procura-se relacionar a tríade dialética às tecnologias distributiva, interativa e colaborativa, de acordo com as formas de

controle e o modo de aprendizagem privilegiado no processo educativo, como representado na Figura 1 (Ribeiro, 2004).

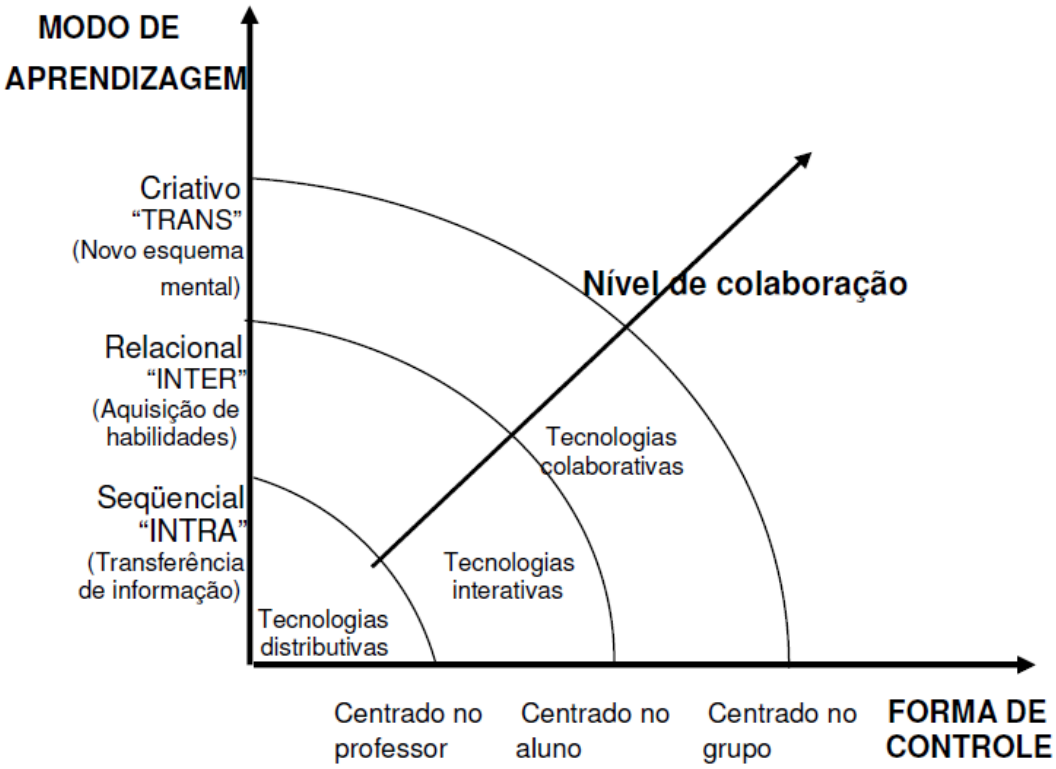

Figura 1 – Tecnologia, modo de aprendizado e formas de controle (Ribeiro, 2004).

A Pesquisa Operacional pode ser caracterizada conforme o modo de aprendizagem da Figura 1. Neste contexto, problemas de modelagem e otimização de sistemas são de grande importância para a formação do Engenheiro de Produção. Assim, o processo de aprendizagem do aluno requer a motivação para habilitá-lo a analisar e resolver problemas que poderão surgir na sua carreira profissional. Portanto, o uso de ferramentas computacionais no aprendizado poderá ajudar na tomada de decisões e modelagem de problemas e sistemas.

1.1 Justificativa

O entendimento de conceitos e técnicas de Pesquisa Operacional tem demonstrado ao longo do tempo que alunos possuem certa dificuldade para assimilar o aprendizado e com o atrativo de utilização de ferramentas computacionais o aprendizado pode tornar mais interessante e facilitado ao aluno. Assim, um ambiente integrado de ferramentas direcionadas ao ensino

pode estimular e aumentar a possibilidade de atrair a atenção do aprendiz, visando aumentar o rendimento e aproveitamento dos alunos.

O interesse para a área de modelagem matemática de problemas deve ser incentivada para a formação de Engenheiro de Produção, visto que o mesmo irá se deparar com problemas complexos e elevado grau de entendimento, assim um ambiente amigável e mais interativo para o ensino, irá propiciar um interesse maior na área de Pesquisa Operacional aplicada.

Assim, o educador deve conseguir criar um ambiente de ensino adequado a suas necessidades e também com determinado estímulo aos aprendizes/alunos. Além disto, alguns fatores motivadores para utilizar componentes na realização deste trabalho foram:

- Reutilização de projeto e implementação;
- Aumento de confiança (componentes testados anteriormente);
- Ganho de tempo e diminuição do custo de manutenção;
- Possibilita soluções padrões de forma mais rápida;
- Permite acrescentar novas funcionalidades a qualquer tempo ao adicionar novos componentes a uma solução já existente.

1.2 Definição e delimitação do problema

O trabalho proposto visa especificar um ambiente utilizando-se de técnicas para modelar arquitetura de software baseada em componentes/objetos. O escopo do trabalho irá basear-se na modelagem do ambiente (especificação) de maneira que possibilite o desenvolvimento de um ambiente de aprendizado, em que possam ser desenvolvidos e implementados novos componentes (objetos de aprendizagem) na plataforma do ambiente, bem como a integração de ferramentas já existentes. Portanto, este trabalho pretende ser um guia/roteiro para criação de ambientes integrados de ensino no domínio específico da Pesquisa Operacional.

1.3 Objetivo Geral

O objetivo principal é ter uma modelagem como guia para a construção, ou mesmo escolha, de um ambiente de aprendizado. Portanto a modelagem arquitetural visa o educador que irá construir um ambiente de aprendizado, direcionando o ambiente para seu enfoque pedagógico e estilo de ensino.

1.4 Objetivos Específicos

Especificar um modelo arquitetural de um ambiente computacional para ensino de Pesquisa Operacional, que consiste de uma plataforma de integração de objetos de aprendizagem (componentes), a serem desenvolvidos ou já existentes. Assim, o trabalho tentará facilitar o aprendizado de conteúdos de Pesquisa Operacional no curso de graduação em Engenharia de Produção da UEM.

Estudar os conteúdos de aprendizado em Pesquisa Operacional, bem como algumas ferramentas existentes, procurando identificar recursos e conteúdos nelas abordados. Criar um roteiro para que futuramente diferentes abordagens de aprendizados em Pesquisa Operacional possam ser enfocadas no âmbito de um ambiente computacional integrado.

Ter como base um modelo no desenvolvimento que possa contribuir para especificação e criação de novos ambientes para prendizado.

1.5 Método de Pesquisa

O trabalho será desenvolvido por meio da investigação cientifica, sendo realizado a pesquisa bibliográfica sobre o tema, bem como o levantamento das técnicas para especificação de software.

A primeira etapa do trabalho consistiu na elaboração da proposta de trabalho, em que se foi definido o tema e escopo do trabalho, isso foi realizado por meio de um levantamento de assuntos correlatos ao trabalho;

Em seguida foi feito um levantamento bibliográfico sobre assuntos correlatos de maneira ampla, nesta etapa conceitos e técnicas foram analisados procurando enriquecer de forma global o trabalho. A Arquitetura de Software foi explorada de maneira que o trabalho pudesse ser um guia para desenvolvimentos futuros. Estudos a respeito de arquitetura de software e identificação de objetos de aprendizagem/componentes gerais para o desenvolvimento de uma aplicação educacional.

Para o desenvolvimento foram realizados levantamentos de necessidades com professores da área e com material produzidos academicamente, com isto foi possível fazer uma especificação de uma arquitetura de software educacional para o ensino de Pesquisa Operacional, focando o ensino de Programação Linear. Portanto, foi possível realizar uma customização de um ambiente direcionado ao assunto. E por último um estudo de caso específico de mistura foi utilizado como estudo de casos, com o objetivo de identificar os componentes no modelo proposto.

1.6 Organização do Trabalho

Este trabalho apresenta quatro capítulos que incluem a introdução ao trabalho, mais três capítulos que estão estruturados da seguinte forma:

O capítulo 2 apresenta a fundamentação teórica que embasa todo o trabalho, com uma conceituação e definição de arquitetura de software, objetos de aprendizagem e uma breve contextualização de modelagem matemática relevantes para este trabalho.

No capítulo 3 é mostrado o desenvolvimento do trabalho, onde é apresentado o modelo padrão de arquitetura da IEEE/LTSA, juntamente com a proposta de especificação desse modelo (Arquitetura de Software Proposta). A proposta concentra-se na modelagem abstrata de componentes e subcomponentes com suas principais funcionalidades e serviços. Por fim, é apresentada a modelagem da Arquitetura Proposta para o domínio especifico de aplicação em Pesquisa Operacional (Programação Linear).

Finalmente, o capítulo 4 traz a conclusão do trabalho, ressaltando as contribuições e possíveis trabalhos futuros.

2 REVISÃO DA LITERATURA

Este capítulo apresenta a fundamentação teórica necessária como embasamento do trabalho. Para isto foram realizados levantamentos sobre os temas relevantes, contidos neste capítulo, e aceitos pela comunidade científica da área.

Essa fundamentação serve para melhor entender a proposta do trabalho, bem como de base para aprimoramento e complementações do trabalho proposto.

2.1 Arquitetura de Software

Arquitetura de Software surgiu como uma evolução natural das abstrações de projetos, na busca de novas formas de construir sistemas de software maiores e mais complexos (Shaw, 1996). A arquitetura de software busca formas de descrever organizações de sistemas existentes, com o objetivo de promover o reuso da experiência de projeto, possibilitando a escolha da forma de organização mais adequada para um sistema de software a ser desenvolvido.

As descrições de Arquitetura de Software podem ser classificadas como padrões arquitetônicos, padrões de construção de software, assim como os padrões de projeto (Gamma *et al*, 1995), porém apresenta um nível de abstração superior a estes (Buschmann, 1996).

Uma estrutura de classes, que constitui uma aplicação, um *framework* ou um componente, embute uma definição arquitetônica. No caso de uma aplicação produzida a partir de um único *framework*, que constitua um modelo de domínio, a arquitetura da aplicação está definida na estrutura do *framework*. Um *framework* que define uma arquitetura para um domínio de aplicações pode ser subdividido em *subframeworks* que corresponderiam aos componentes de software (Campos, 1997).

2.1.1 Conceitos e definições

A arquitetura de um sistema reflete o conjunto de decisões que devem ser tomadas, no projeto de um sistema computacional, determinando a organização e estrutura geral do sistema. Uma definição clássica de arquitetura diz que arquitetura de software define o que é o sistema em termos de componentes computacionais e os relacionamentos entre eles (Shaw, 1996).

Semelhante a esta definição tem-se que arquitetura de software são as estruturas que incluem componentes, suas propriedades externas e os relacionamentos entre eles, constituindo uma abstração do sistema. Esta abstração suprime detalhes de componentes que não afetam a forma como eles são usados ou como eles usam outros componentes, auxiliando o gerenciamento da complexidade.

2.1.1.1 O que o termo Arquitetura de Software significa?

Este campo emergiu como um foco explícito para pesquisa e desenvolvimento, não tendo assim uma definição aceita universalmente. Além disso, da observação do emprego do termo "Arquitetura de Software", conclui-se que é bastante usado em diferentes formas, frequentemente tornando-se difícil entender qual aspecto está sendo abordado. Dentre os vários usos estão: A Arquitetura de um Sistema Particular; Um Estilo de Arquitetura; O Estudo Geral de Arquitetura (Garlan, 1995).

2.1.1.2 O que é Arquitetura de Software?

O projeto arquitetural de um sistema de software se preocupa com a estrutura bruta e as maneiras em que essa estrutura conduz à satisfação das propriedades chaves do sistema. Na prática, um projeto arquitetural cumpre dois papéis preliminares. Primeiro, o projeto arquitetural fornece um nível de abstração do projeto em que os desenvolvedores do software podem raciocinar sobre o comportamento do sistema total, através de requisitos funcionais e não-funcionais. Abstraindo os detalhes da implementação, uma boa descrição arquitetural faz um sistema intelectualmente tratável, expondo as propriedades que são as mais cruciais a seu

sucesso. Desta maneira um projeto arquitetural é frequentemente o documento técnico chave usado para determinar se um novo sistema proposto poderá atender suas exigências mais críticas (Garlan, 1995).

Em segundo, um projeto arquitetural serve como "a consciência" para um sistema, como evolui com o tempo. Caracterizando as suposições cruciais do projeto do sistema, um bom projeto arquitetural orienta o processo de aprimoramento do sistema indicando que aspectos do sistema podem ser facilmente mudados sem comprometer a integridade do sistema. Assim, uma arquitetura bem documentada ajuda não somente no tempo do projeto, mas também durante todo o ciclo de vida de um sistema. Para satisfazer a seus múltiplos papéis durante todo o ciclo de vida do software, uma descrição arquitetural deve ser simples o bastante para permitir o entendimento e visão do sistema em nível abstrato. Praticamente, isto limita descrições arquiteturais a um nível do detalhe que possa caber em uma ou duas páginas. Consequentemente, para representar projetos arquiteturais, as descrições são geralmente hierárquicas: os elementos arquiteturais atômicos em um nível da abstração são descritos por uma arquitetura mais detalhada em um nível mais baixo (Garlan, 1995).

As descrições arquiteturais são relacionadas primeiramente com os seguintes pontos básicos:

- **Estrutura do sistema:** As descrições arquiteturais caracterizam a estrutura de um sistema em termos de elementos computacionais de alto nível e de suas interações, ou melhor, uma arquitetura se preocupa com a natureza de uma solução para um problema como uma configuração de interação de componentes. Não é especificamente sobre a representação das exigências (por exemplo, relacionamentos abstratos entre elementos de um domínio do problema), nem os detalhes das execuções (estruturas, por exemplo, dos algoritmos e de dados).

- **Abstrações importantes para a interação:** Interações entre os componentes arquiteturais frequentemente projetam como realizar conexões, refletindo um importante vocabulário para desenvolvedores de sistemas. Quando as interações puderem ser simples, como chamada de procedimentos ou variáveis dos dados compartilhadas, elas frequentemente são a representação de formas mais complexas de uma comunicação e da coordenação. Os exemplos incluem os tubos (*pipes*), interações cliente-servidor (com regras sobre a inicialização, finalização e a manipulação de exceções), acontecimento de conexões *broadcast* (com receptores múltiplos) e protocolos de acesso à base de dados (com protocolos para a requisição da transação).

- **Propriedades globais**: Desde que um uso principal de um projeto arquitetural deve considerar sobre o comportamento do sistema total, os projetos arquiteturais são tipicamente focados com o sistema inteiro. Os problemas orientados por uma arquitetura são geralmente em um nível mais global de sistema, tais como taxas e latências fim-a-fim (*end-to-end)* de dados, retorno de uma porção de um sistema à falha em outras peças, ou sistemática de propagação das mudanças quando uma porção de um sistema é modificada (como a mudança da plataforma em que o sistema funciona).

2.1.1.3 Porque Arquitetura de Software é importante?

Projeto Arquitetural de um sistema de software grande sempre desempenha um papel significante em determinar o sucesso do sistema. Escolher uma arquitetura inapropriada pode ter um efeito desastroso. O reconhecimento atual da importância da arquitetura de software emergiu para uma base mais disciplinada para projeto arquitetural, que tem o potencial para melhorar significativamente a habilidade de construir sistemas de software efetivos.

Especificamente, o uso desses princípios pode ter um impacto positivo sobre pelo menos cinco aspectos de desenvolvimento de software (Garlan, 1995):

Entendimento: Arquitetura de software simplifica a habilidade para compreender grandes sistemas e para representá-los em um nível de abstração no qual um projeto de alto nível pode ser entendido. Além disso, descrição arquitetural explora o alto nível de obstáculos sobre projeto de sistemas, bem como a razão para fazer escolhas arquiteturais específicas.

Reuso: Descrições arquiteturais apóiam reuso para múltiplos níveis. Alguns trabalhos sobre reuso geralmente focam sobre bibliotecas de componentes. Projetos arquiteturais apóiam tanto reuso de grandes componentes como também *frameworks* em que componentes podem ser integrados.

Evolução: A arquitetura de software pode expor as dimensões esperadas ao longo da evolução de um sistema. Tornando claras as barreiras de um sistema, fica fácil compreender melhor as ramificações de mudança e estimar melhor custos de ramificações. Além disso, descrições arquiteturais podem separar interesses da funcionalidade de um componente da maneira em que cada componente é conectado a outro componente. Isto permite mudar o

mecanismo de conexão para manusear os interesses evoluindo sobre o desempenho, interoperabilidade, prototipagem e reuso.

Análise: Descrições arquiteturais fornecem novas oportunidades para análises, incluindo formas de alto nível de checar a consistência de sistemas, adaptações para um estilo arquitetural, adaptações para qualidades de atributos e análise de domínio específico para arquiteturas que se adequam para estilos específicos.

Administração: Há uma forte base racional para se fazer uma realização de uma arquitetura de software viável, um ponto chave em um processo de desenvolvimento de software industrial. Alcançando este ponto devem-se especificar as exigências de capacidade operacionais iniciais de um sistema de software, suas dimensões de crescimento antecipado. A arquitetura de software é uma forte base, que demonstra que a arquitetura, se implementada, satisfaria as exigências iniciais do sistema e as direções antecipadas de crescimento. Se um produto de software for desenvolvido sem satisfazer essas condições, existe um risco significante que o sistema será também inadequado ou incapaz de acomodar a mudança.

A importância da arquitetura de software pode também ser em termos do amplo impacto sobre os condutores de mercado que é importante para negócios de software intensivo. Estes condutores afetam a forma de planejar seus projetos de software e a maneira de construir seus sistemas de software (Garlan, 1995).

2.1.2 Estilos Arquitetônicos

Uma arquitetura de software é definida a partir de um conjunto de artefatos de software e da forma como estes artefatos são interligados. Assim, um estilo arquitetônico define uma topologia de conexão específica, isto é, uma forma de interconectar os artefatos de uma arquitetura, o que restringe as possibilidades de interação entre os artefatos da topologia. A seguir, apresentam-se alguns estilos arquiteturais existentes (Shaw, 1996), (Buschmann, 1996), (Mendes, 2002)

Tubos e Filtros (*Pipes and Filters*): o estilo arquitetural de tubos e filtros considera a existência de uma rede pela qual fluem dados de uma extremidade (origem) a outra (destino). O fluxo de dados se dá através de tubos, que interligam entradas e saídas de filtros, e os dados

sofrem transformações quando processados nos filtros, que são artefatos de software com entradas por onde recebem fluxo de dados, e saídas, por onde disponibilizam fluxos de dados resultantes de sua atuação sobre os fluxos de entrada. A forma de interação característica deste estilo arquitetural é o fluxo de dados unidirecional, a Figura 2 representa um exemplo hipotético.

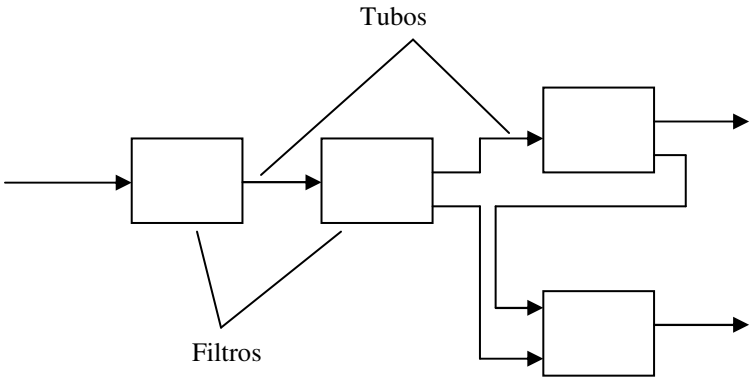

Figura 2 – Tubos e Filtros.

Sistemas em Camadas (*Layered Systems*): o estilo arquitetural de camadas estrutura um sistema num conjunto de camadas, onde cada uma delas agrupa um conjunto de tarefas num determinado nível de abstração. Neste estilo cada camada se conecta a outra, desde que disposta de forma adjacente e cada camada pode ter uma ou duas camadas adjacentes, como pode ser observado na Figura 3. A forma de interação entre camadas conectadas pode ser baseada em chamadas de procedimento ou em protocolos de comunicação mais complexos.

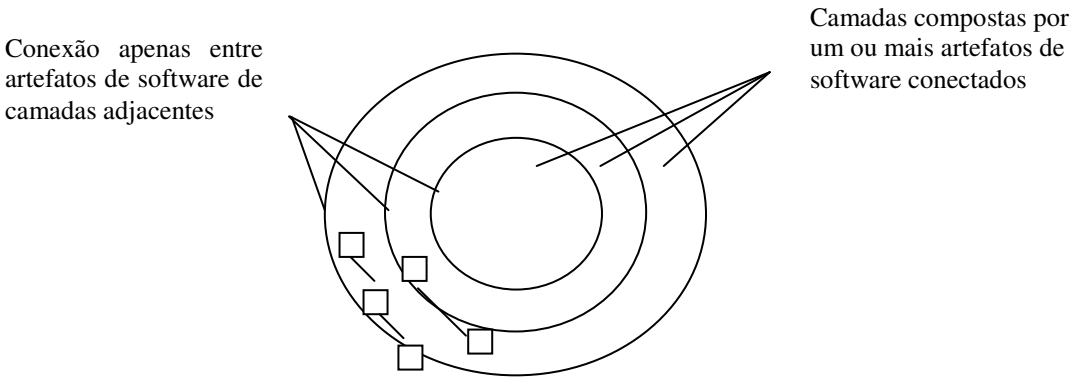

Figura 3 – Sistema em Camadas.

Sistemas Baseados em Invocação Implícita (*Implicit Invocation Systems*): Neste estilo, a comunicação é baseada na divulgação dos eventos. Componentes podem estar ou não habilitados para reagir a ocorrência de um evento. A reação consiste na execução de algum procedimento. O sistema dispõe de um mecanismo para tratamento de eventos, o qual encarrega-se de encaminhar os eventos aos componentes destinatários. Quando um componente divulga um evento, os componentes habilitados a reagirem àquele evento executam seus respectivos procedimentos.

Sistemas Baseados em Repositórios: neste estilo subsistemas independentes compartilham um repositório de dados, mas sem interagir diretamente entre si. A forma de interação entre subsistemas consiste na manipulação de informações compartilhadas através do repositório. Uma vantagem dos sistemas baseados em repositório é a sua flexibilidade, na medida que subsistemas compartilham um repositório, não tem dependências entre si. Assim, torna-se mais fácil alterar o sistema através de substituição, remoção, inclusão ou alteração de subsistemas sem que os demais sejam afetados. Este estilo está representado na Figura 4.

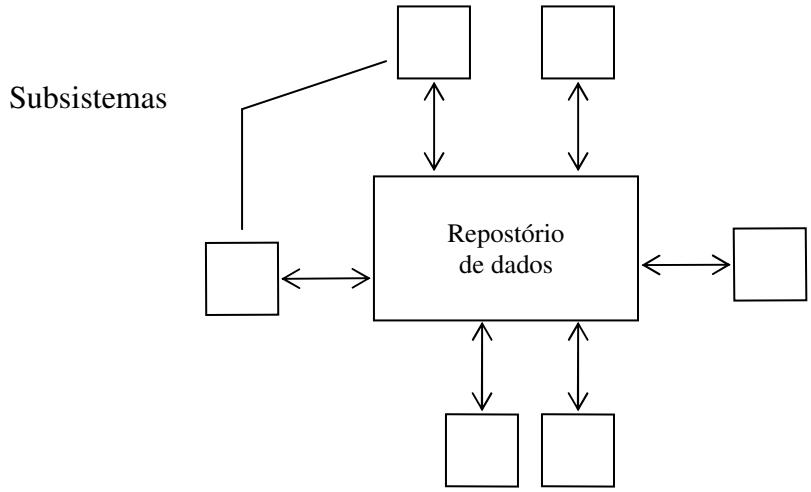

Figura 4 – Sistema Baseado em Repositório.

A dificuldade de descrever adequadamente estilos arquitetônicos a partir de mecanismos de descrição existentes está na falta da capacidade de associar uma semântica específica aos elementos de conexão. Os conectores, ao invés de serem tratados como elementos secundários de uma descrição, devem ser tratados como entidades de projeto com o mesmo grau de importância dos módulos que são interligados e, como tal, devem ser adequadamente especificados (Garlan, 1997).

2.2 Componentes de Software

A construção de aplicações utilizando o conceito de reutilização surgiu como uma alternativa para a dificuldade no desenvolvimento de software (Helton, 1998), (Pressman, 2005). Desenvolvedores precursores relatam que componentes sugerem um potencial considerável no desenvolvimento de grandes sistemas (Helton, 1998).

Uma das maiores vantagens na utilização de componentes de software é a criação de um mercado, que possibilita os desenvolvedores de software a adquirir os componentes e assim criar de maneira mais rápida as aplicações (Pfister, 1998).

Componentes comercializados são em sua maioria produtos padronizados, trazendo consigo todas as vantagens do processo de montagem de componentes permitindo realizar personalizações (Szyperski, 2002). O usuário final dos componentes possui então a alternativa de escolha dos componentes para elaboração do seu software (Wolfgang, 1997).

No evento WCOP 96 (*Workshop on Component-Oriented Programming*) definiu-se componente como uma unidade de composição com interfaces contratualmente especificadas e dependências de contexto explícitas. Componentes podem ser duplicados e estar sujeitos a composição com terceiros (Szyperski, 1996). Esta definição foi refinada no WCOP 97: o que torna alguma coisa um componente não é uma aplicação específica e nem uma tecnologia de implementação específica. Assim, qualquer dispositivo de software pode ser considerado um componente, desde que possua uma interface definida. Esta interface deve ser uma coleção de pontos de acesso a serviços, cada um com uma semântica estabelecida (Szyperski, 1997).

Para melhor compreender os componentes, torna-se interessante conhecer algumas de suas principais características (Szyperski, 2002) (Pfister, 1998):

- É uma unidade de desenvolvimento independente, deve estar separada de outros componentes, encapsulando desta forma sua característica, não deve ser instanciado parcialmente. Neste contexto, é uma unidade a parte que não pode esperar em ter acesso aos detalhes de construção de todos os componentes que o envolvem;
- É uma unidade de composição de uma terceira parte (reuso), precisa de especificações claras do que ele necessita e disponibiliza. Em outras palavras um componente precisa

encapsular sua implementação e interagir com seu ambiente através de interfaces bem definidas;

- Não tem estado persistente, é necessário que não possa ser distinguível das suas cópias ou dele próprio. Assim, um componente pode ser chamado de dentro de um sistema particular, ou seja, em um determinado processo pode vir a existir mais de uma cópia, ou instância, de um componente em particular. Desta forma, apesar de ser proveitoso perguntar se tem ou não um componente disponível, não é importante perguntar sobre o número disponível de cópias do componente;
- Pode ser uma simples e única classe e/ou uma coleção de classes, muitas vezes chamado de módulo.
- Um componente pode ser definido como uma unidade de software independente, que encapsula dentro de si seu projeto e implementação, e oferecer interfaces bem definidas para o meio externo.

2.2.1 Tipos de componentes

Componentes distinguem-se em cinco tipos (Szyperski, 2002):

- Componentes de *middlewar*e (um mediador que é acrescentado para juntar componentes, muitas vezes através de diferentes sistemas operacionais (Voas, 1998)), incluindo ORBs (*Object Request Broker*), monitores de processamento de transação (projetados para monitorar e muitas vezes modificar as funcionalidades de interação de um outro componente (Voas, 1998) e sistemas de gerenciamento de base de dados;
- Componentes lógicos, serviços de processamento de imagem, aperfeiçoamento de um serviço específico não visual;
- *Vertical component*s, pequenas aplicações servindo a um domínio específico;
- Componentes GUI (*Graphic User Interfac*e), disponibilizado como ferramenta para elaboração da interface do usuário; e,
- *Container component*s possibilitam uma visualização embutida (oculta) e dispõem de ambiente para componentes visuais.

2.2.2 Compatibilidade de componentes

A compatibilidade entre os componentes determina quais servem para elaborar um novo componente ou uma aplicação (Garlan, 1997b), ou seja, é através da compatibilidade entre eles que se podem caracterizar dois componentes que possam se comunicar com o objetivo de formar uma arquitetura (Canal, 1998).

Na Figura 5 observa-se a compatibilidade entre os componentes C1 e C2. O componente C1 possui interfaces de saída C e D e de entrada A e B. O componente C2 possui interfaces de saída B e Y e de entrada C e X. A conexão entre os componentes C1 e C2 é estabelecida através da compatibilidade entre as interfaces B e C (Szyperski, 2002).

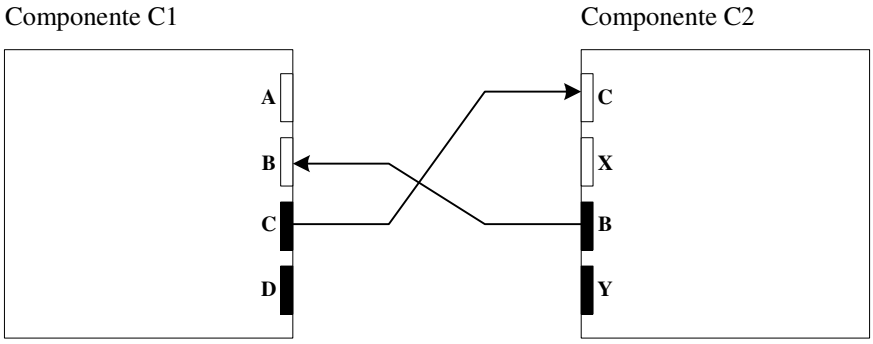

Figura 5 – Compatibilidade de Componentes.

A compatibilidade será encontrada ao compor duas especificações de componentes, através de análises de suas interações e especificações. Contudo este meio de descobrir a compatibilidade é muitas vezes complexo. A sugestão é que a compatibilidade entre os componentes possa ser encontrada através da especificação explícita de suas interfaces, seus papéis e as conexões entre os sistemas indicando a conduta do componente do lado de fora (Canal, 1998).

As especificações dos componentes, seus protocolos de comunicação e suas interfaces devem ser de fácil compreensão para auxiliarem na determinação da compatibilidade entre os componentes (Garlan, 1997b).

2.2.3 Composição de componentes

A possibilidade de composição de componentes permite ao desenvolvedor uma maior liberdade para preocupar-se com o domínio do problema e não com a implementação (Garlan, 1997). Cada componente é uma parte identificada separadamente composta de classes, métodos, sub-rotinas ou arquivos. Um grupo de componentes deve trabalhar em cooperação para fornecer uma particularidade de capacidade funcional ou de coerência lógica para resolução de seus serviços, assim esse grupo pode formar um componente de nível superior.

A composição de componentes pode ser definida como a capacidade do sistema de ser composto por combinações de componentes os quais elaboram sua arquitetura (Canal, 1998).

2.2.4 Conexão de Componentes

Um ponto principal da abordagem de desenvolvimento baseado em componentes é a forma de realizar a conexão de componentes. Levando-se em conta um ambiente de desenvolvimento em que todos os componentes tenham sido implementados em uma mesma linguagem de programação e possuam a mesma localização física, mecanismos da própria linguagem podem ser usados para que um componente mantenha referência de outro. Chamadas de procedimento dão acesso aos serviços de componentes.

A conexão de componentes ocorre desde que suas interfaces sejam compatíveis. Isto significa que, independentemente de homogeneidade de linguagem em que estão implementados, de localização física e de plataforma de execução, os serviços requeridos por um, estejam disponíveis no outro. A descrição de componentes deve permitir verificar esta compatibilidade, senão é impossível verificar se dois componentes podem ser conectados.

Pode ser citado como padrão de interconexão o *System Object Model* (SOM) da IBM, baseado em CORBA (*Common Object Request Broker Architeture*) e COM (*Component Object Model*) da Microsoft os quais usam IDL's (*Interface Definition Language*) e possibilitam conexão com independência de linguagem, plataforma de execução e localização física dos componentes (Pfister, 1998). *JavaBeans,* da *Sun Microsystems*, em que os componentes são

programados em Java, permite independência de plataforma de execução e localização física de componentes (Helton, 1998).

2.2.5 Adaptação de Componentes

Normalmente componentes precisam ser adaptados para se modelarem aos requisitos do sistema que serão acoplados, tendo em vista que dificilmente serão reusados da forma que foram desenvolvidos. Existem duas formas usualmente utilizadas para fazer esta adaptação, colagem (*glueing*) e empacotamento (*wrapping*) (Bosch, 1997).

O empacotamento, ao invés de modificar o componente, cria uma visão externa diferente para ele. Uma alternativa, para interconectar componentes originalmente incompatíveis consiste em criar um componente que intermedie a comunicação. Este componente intermediário é genericamente chamado de cola (*glue*) (Szyperski, 1997).

2.2.5.1 Empacotamento de componentes

O empacotamento tem por objetivo produzir uma interface original com vista a adaptá-la a requisitos específicos. A Figura 6 demonstra esta situação. Os componentes, C1 e C2**,** apresentam-se com interfaces incompatíveis, isto é, não podem ser conectados para operação conjunta. Na parte inferior da Figura 6 o componente C2 é inserido em uma estrutura de empacotamento com uma interface diferente da interface definida neste componente. O resultado é que a interface da estrutura de empacotamento é compatível com a interface do componente C1**,** possibilitando a integração de suas interfaces. Com isto, obtém-se a situação inicialmente impossível; a operação conjunta de componentes C1 e C2, com interfaces incompatíveis.

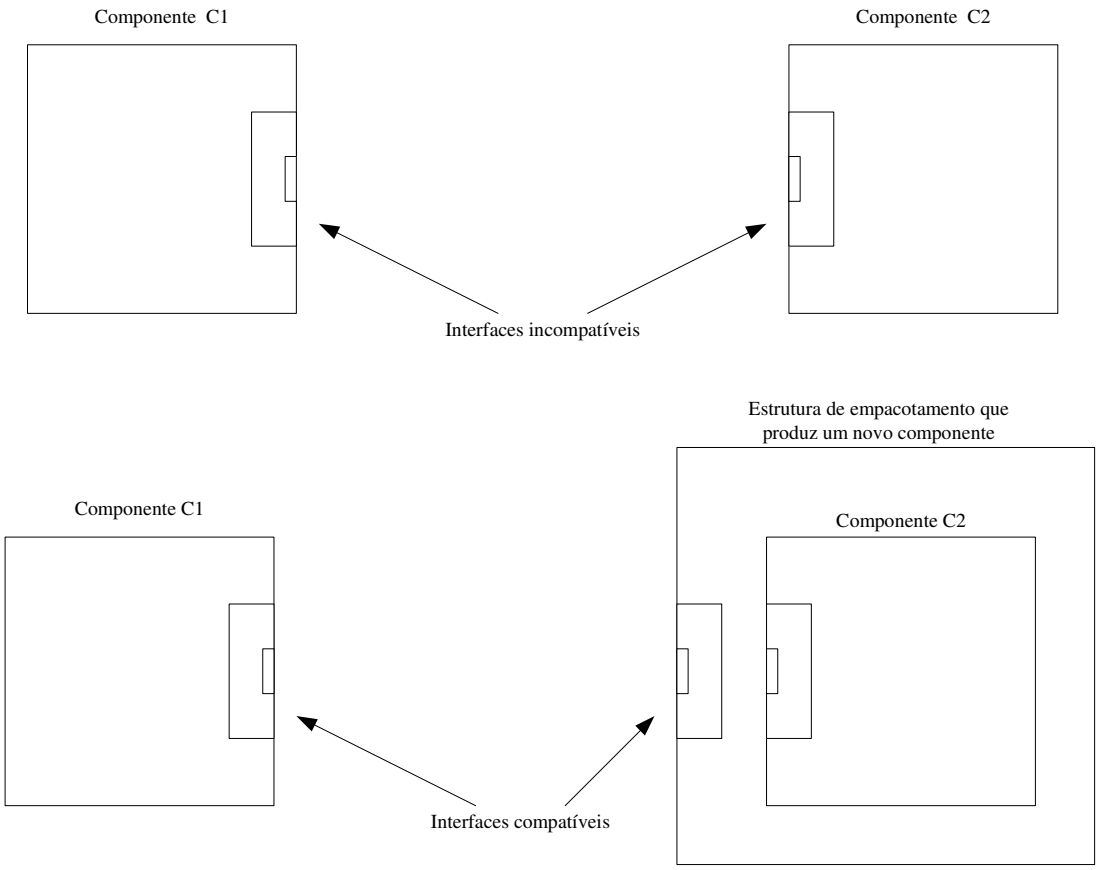

Figura 6 – Empacotamento de Componentes.

Este recurso pode ser usado com diferentes finalidades. Como em uma situação em que a única incompatibilidade seja a estrutura de interface, isto é, componentes com incompatibilidade de comportamento e funcionalidades e interfaces não conectáveis. Isto pode ser causado por assinaturas de métodos diferentes em um ambiente homogêneo (diferenças em nome de método, ordem de parâmetros, previsão de retorno) ou por heterogeneidade dos componentes (que pode incluir diferença de linguagem de programação, de plataforma de execução e de localização física). No caso de diferenças sintáticas nas assinaturas de métodos (invocados por um componente e supridos por outro). Em caso de problemas de heterogeneidade, isto é, a estrutura torna transparentes as incompatibilidades referentes a diferenças de linguagem de programação, plataforma de execução ou localização física.

Uma segunda finalidade de uma estrutura de empacotamento é alterar a funcionalidade original do componente empacotado. Neste caso, a estrutura poderia incluir novas funcionalidades, como também alterar o tratamento original a determinadas invocações de métodos.

2.2.5.2 Colagem de componentes

A colagem de componentes (*glueing*) representa outra forma de interconectar componentes, ou melhor, possibilitar a operação conjunta de componentes originalmente incompatíveis. Esta incompatibilidade pode estar associada à sintaxe das interfaces, heterogeneidade ou necessidade de extensões ou alterações funcionais. A diferença neste caso é que o tratamento dado ao problema é a inclusão de um novo elemento, a cola (*glue*) entre os componentes incompatíveis, possibilitando sua operação conjunta. A Figura 7 ilustra a compatibilização de componentes através de colagem.

O elemento cola da Figura 7 nada mais é do que um terceiro componente, em que a interface possibilita sua conexão aos componentes C1 e C2, cuja funcionalidade consiste em compatibilizar a operação conjunta destes componentes.

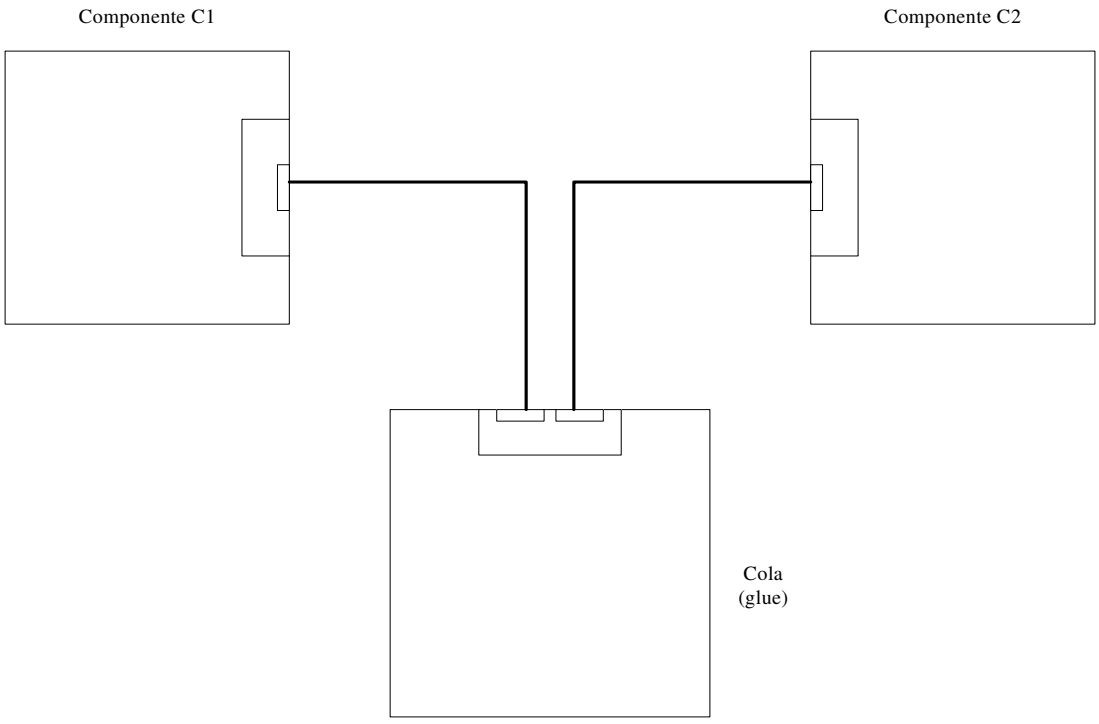

Figura 7 – Cola como um Terceiro Componente.

Observa-se um problema no uso da prática de colagem de componentes. Frequentemente a cola fica escondida em uma estrutura de programação complexa, que torna obscuro seu propósito. Um melhor entendimento da semântica da colagem de componentes pode ser obtido pelo uso de abstrações para modelar a cola no nível de projeto, isto é, modelá-la como um componente, como no exemplo citado na Figura 7.

2.2.6 Arquitetura de Componente de Software

A arquitetura de um componente de software descreve a composição dos componentes e os relacionamentos entre eles. Estes relacionamentos requerem considerações das interfaces dos componentes (Wolfgang, 1997).

Projetar um componente é muitas vezes mais complexo que montá-los (Wills, 1997). Classes e objetos são muitas vezes atribuídos a componentes de softwares específicos como parte de um projeto físico (Pressman, 2005).

Para desenvolver a arquitetura deve-se iniciar com a elaboração de um projeto das interfaces do componente e posteriormente sua implementação, sob este prisma, a arquitetura representa um conjunto de interfaces e suas conexões para serem preenchidas por componentes (Wolfgang, 1997).

A arquitetura de um sistema de software define o sistema em termos de componentes e suas interações (ou conexões) sobre outros componentes. A arquitetura deve mostrar também a relação entre as requisições do sistema e os componentes que a compõem, para tanto são analisadas propriedades como capacidade, comunicação, consistência e compatibilidade entre componentes (Garlan, 1997). Os componentes, em conjunto, devem se encaixar e serem facilmente especializados para uma variedade de produtos. A arquitetura de componentes deve ser eficiente e flexível. Esta tarefa exige bastante competência, experiência e responsabilidade do projetista do componente (Wills, 1997).

A arquitetura é um fator determinante no sucesso do reuso do componente e do sistema (Davis, 1997), pois ela possibilita a construção de sistemas com componentes reusáveis; e possibilita a adaptação de alterações tecnológicas.

2.2.6.1 Especificação da Arquitetura

A especificação da arquitetura da aplicação do software mostra a coleção dos componentes inter-relacionados, incluindo aspectos como derivação de componentes e arquiteturas usando mecanismos de herança, polimorfismo, verificação, análise de compatibilidade de sistemas componentizados (Canal, 1998).

É através da especificação da arquitetura do software que se pode alcançar um determinado nível de abstração, facilitando a descrição e compreensão de sistemas complexos e aumentar o reuso de componentes e arquiteturas (Canal, 1998).

O propósito da especificação da arquitetura do software é definir os componentes, considerando outros componentes para composição de aplicações. A estrutura é formada por diversos componentes e suas conexões e a composição destes requer moldar uma junção de composições com portas livres e funções que não podem ser classificados como componentes ou conectores (Canal, 1998).

Para uma melhor especificação da arquitetura são utilizadas muitas vezes técnicas formais, que especificam as características de abstrações das propriedades dos componentes e a dimensão de sua especialização, e o suporte de uma análise de sua arquitetura (Garlan, 1997).

2.2.6.2 Granularidade de Componente

O fator de granularidade nos componentes tem que ser observado no seu desenvolvimento. Muitos componentes de diferentes origens podem fazer parte ou de uma arquitetura de um componente ou de uma aplicação. Para que os componentes sejam desenvolvidos independentemente, sua granularidade e dependências devem ser controladas externamente de forma cuidadosa. Para esta finalidade, a tecnologia de orientação a objeto torna-se um bom caminho no desenvolvimento da tecnologia de componentes (Szyperski, 2002).

Quando é feito um projeto de um componente ou de uma aplicação que utiliza um componente, a granularidade do componente pode ser pré-determinada. A importância da granularidade deve então ser registrada. Componentes necessitam ser construídos com características de modularidade, onde são analisadas as propriedades do componente e as interfaces que devem ser construídas em conjunto (Szyperski, 2002), devendo assim, serem

projetados e desenvolvidos independente uns dos outros para então serem combinados pelo cliente (Pfister, 1998). Estas construções generalizadas de componentes são apenas válidas se podem ser bastante utilizadas em outras aplicações e a granularidade do componente influencia nisto (Szyperski, 2002).

Boas arquiteturas modulares possuem dependências explícitas e ajudam a reduzir e controlar estas dependências. Apenas quando a modularidade está bem estabelecida, é fácil de migrar parte de um sistema para componentes ao adotar padrões relevantes de interface (Szyperski, 2002).

As descrições de Arquitetura de Software podem ser classificadas como padrões arquitetônicos, padrões de construção de software, assim como os padrões de projeto, porém apresenta um nível de abstração superior a estes (Buschmann, 1996).

Uma estrutura de classes, que constitui uma aplicação, um *framework* ou um componente, embute uma definição arquitetônica. No caso de uma aplicação produzida a partir de um único *framework*, que constitua um modelo de domínio, a arquitetura da aplicação está definida na estrutura do *framework*. Um *framework* que define uma arquitetura para um domínio de aplicações pode ser subdividido em *subframeworks* que corresponderiam aos componentes de software (Campos, 1997), que no caso em questão pode ser visto como os objetos de aprendizagem software.

2.3 Objetos de Aprendizagem

Com o avanço da tecnologia, o computador tem ganhado grande destaque como instrumento para a construção e desenvolvimento de conceitos científicos. Surgem, então, novas configurações no processo de ensino e aprendizagem e o desenvolvimento de softwares dedicados à área educacional ganha destaque no âmbito acadêmico.

Com a difusão do computador e sua rápida expansão vêm proporcionando novas formas de comunicação e novos espaços de interatividade. E na educação, a Internet por meio da *World Wide Web (WWW* ou simplesmente *Web*) trouxe a possibilidade de desenvolvimento de portais que contêm objetos de aprendizagem para auxiliar no ensino e aprendizagem de conteúdos específicos como, por exemplo, no estudo de Pesquisa Operacional, e outras áreas,

em engenharia. Esses portais podem ser acessados na escola, no trabalho e em casa, se especializando ou concretizando o aprendizado.

2.3.1 Padrões, desenvolvimento e acesso

A definição de *Learning Objects,* segundo o IEEE/LTSC, refere-se a "qualquer entidade, digital ou não, que pode ser utilizada e reutilizada durante o processo de aprendizagem que utilize tecnologia. Tais objetos podem ter conteúdo hipermídia, conteúdo instrucional, outros objetos de aprendizagem e software de apoio" (IEEE/LTSC, 2010).

Esses objetos são elementos de uma nova metodologia de ensino e aprendizagem baseada no uso do computador e da Internet, fundamentados na orientação a objetos, valorizando a sua criação e reusabilidade para diversos contextos.

Os objetos de aprendizagem devem poder ser criados e utilizados em qualquer formato como, por exemplo: *applets Java* (JavaWorld, 2010); aplicativos em *Macromedia Flash* desenvolvidos em linguagem própria (*ActionScript*) (Ynemine, 2002); trechos de vídeo ou áudio em formatos diversos; e apresentações *PowerPoin*t. Em um senso amplo, qualquer conjunto de gráficos e imagens que, combinados com textos e mais algum elemento (hipertexto/hipermídia), possam causar uma reflexão no usuário podem ser considerado objetos de aprendizagem.

Como mencionado anteriormente, para se disponibilizar na *Web* esses objetos, tem-se a necessidade de construir um portal (ou repositório). Para a construção de um ambiente para o repositório de objetos de aprendizagem é necessário, em primeiro lugar, considerar a estrutura e funcionalidades desejadas para determinar o tipo de sistema operacional, as linguagens de programação e softwares de apoio, compondo uma base para o portal, devendo levar em consideração alguns aspectos técnicos (Scheer & Gama, 2004).

Um deles está no desenvolvimento e na identificação dos objetos de aprendizagem. O desenvolvimento destes objetos deve prever a possibilidade de seu reuso, a sua organização e uma classificação de metadados, armazenados em um sistema de gerenciamento de conteúdos ou de aprendizagem *(LCS/LMS*).

Os *metadados* (termo genérico usado para descrever dados que podem ser utilizados para identificar características comuns entre diferentes documentos) (Scheer & Gama, 2004) são padronizados, independentes de sistemas e têm como propósito facilitar a busca, avaliação, aquisição e uso dos objetos por estudantes ou instrutores.

Para auxiliar nesta tarefa, os esforços de padronização de metadados de objetos de aprendizagem são vários, como o *LOM (Learning Objects Metadata*) do *Learning Technology Standard Committee* do *Institute of Electrical and Electronic Engineers* (*IEEE/LTSC),* o *SCORM* da *Advanced Distributed Learning* (*ADL*), o *IMS-Metadata* do *Instructional Management System (IMS) Global Consortium* e a especificação da *Dublin Core Metadata Initiative* (Scheer & Gama, 2004).

O uso destes padrões visa atender as necessidades do desenvolvimento de um portal e podendo utilizar a linguagem XML (*eXtensible Markup Language)* que foi desenvolvida para descrever conteúdo de documentos, e projetada para ser utilizada na Internet conforme a definição de W3c - *World Wide Consortium* (W3C, 2010). Propicia a descrição e armazenamento de dados para os metadados que serão utilizados em um portal ou repositório.

O LOM do LTSC/IEEE, (LTSC/IEEE, 2004) é um padrão que segue os propósitos genéricos de metadados, e os objetos de aprendizagem desenvolvidos, organizados e armazenados neste padrão podem ser recuperados quando e como necessário. Outra característica deste padrão é a capacidade de reservar uma definição de blocos que podem ter referências para outros objetos e podem ser combinados sequencialmente para construir grandes unidades educacionais.

2.3.1.1 Outros aspectos técnicos

Para o desenvolvimento do portal e dos objetos, como também outras ferramentas educacionais para a comunidade acadêmica de acordo com as técnicas adequadas para o desenvolvimento de software e padrões, é necessária a determinação do tipo de sistema operacional, a estrutura, banco de dados e as linguagens de programação. Na sequência, as escolhas dizem respeito ao repositório do projeto OE3 (Objetos Educacionais) (Scheer *et* al, 2004).

Como sistema operacional, definiu-se o sistema Linux que, dentre tantas vantagens, apresenta a economia de recursos financeiros, posto que é plataforma de software livre e tem grande eficiência na conectividade em rede. E, como escolha possível, uma linguagem básica para o desenvolvimento do portal foi a PHP (*Personal Home Page*), por ser uma linguagem rápida e dinâmica voltada para a Internet, eficiente em gerar páginas HTML dinamicamente. A PHP possui acesso a diversos sistemas de banco de dados a partir da linguagem SQL (*Structured Query Language*) como, por exemplo, o MySQL que é capaz de processar e gerenciar uma grande quantidade de informações, suficiente para o conteúdo do portal pretendido. Este sistema é também gratuito e de código aberto, sendo um sistema de banco de dados "multiencadeado" (*multithread)*: o programa cria um encadeamento para cada cliente que estabelece uma conexão com o servidor. Não interfere na plataforma que o usuário utiliza e outra vantagem está exatamente na sua segurança e confiabilidade.

2.3.2 Características e classificação dos objetos de aprendizagem

2.3.2.1 Características fundamentais de objetos de aprendizagem

A avaliação da qualidade das informações e conteúdos disponibilizados na rede mundial de computadores, Internet, na forma de textos, apresentações, cursos, isto é, os objetos de aprendizagem, é uma questão que carece de modelos e padrões. Nesse sentido, poucos trabalhos procuram oferecer critérios para avaliar a qualidade dos objetos disponíveis nesta rede.

Há características específicas aos objetos de aprendizagem. Segundo Mendes (Mendes, 2004) os objetos de aprendizagem possuem as seguintes características:

- **reusabilidade**: reutilizável diversas vezes em diversos ambientes de aprendizagem;
- **adaptabilidade**: adaptável a qualquer ambiente de ensino;
- **granularidade**: conteúdo em pedaços, para facilitar sua reusabilidade;
- **acessibilidade**: acessível facilmente via Internet para ser usado em diversos locais;
- **durabilidade**: possibilidade de continuar a ser usado, independente da mudança de tecnologia;

• **interoperabilidade**: habilidade de operar através de uma variedade de *hardware,* sistemas operacionais e *browsers*, intercâmbio efetivo entre diferentes sistemas;

• **metadados** *('data about data'*): descrever as propriedades de um objeto, como: título, autor, data, assunto e etc.

2.3.2.2 Uma classificação para objetos de aprendizagem

González (2005) especifica uma classificação de objetos de aprendizagem para uso pedagógico, como segue:

Objetos de Instrução: são objetos destinados ao apoio da aprendizagem e são divididos em seis tipos distintos.

1. *Objetos de Lição*: combinam textos, imagens, filmes, vídeos, perguntas e exercícios para criar uma aprendizagem interativa.

2. *Objetos Workshop*: são eventos de aprendizagem que podem incluir apresentações, vídeo-conferência e ferramentas de colaboração em

geral.

3. *Objetos Seminários*: são seminários com uma comunicação síncrona com os aprendizes, com o uso de áudio, vídeo, intercâmbios de mensagens, etc.

4. *Objetos artigos*: correspondem a material de estudo, gráficos, tabelas, etc.

5. *Objetos White Papers*: são objetos baseados em textos que detalham tópicos completos.

6. *Objetos Caso de Estudo*: são objetos baseados em textos, que correspondem à análise em profundidade de uma implementação de um produto de software, experiências pedagógicas, etc.

• **Objetos de Colaboração**: são objetos para a comunicação em ambientes de aprendizagem colaborativa e se dividem em quatro tipos:

1. *Objetos Monitores de exercícios*: são objetos em que se produz intercâmbio entre aprendizes e um monitor guia.

2. *Objetos Chats*: são objetos que permitem os aprendizes a compartilhar experiências e conhecimentos. São intercâmbios de mensagens síncronas.

3. *Objetos Fórum*: são objetos que permitem intercâmbio de mensagens assíncronas.

4. *Objetos de Reuniões On-line*: são tipos de objetos que pode-se compartilhar desde documentos até computadores para trabalhos em grupo.

• **Objetos de Prática**: são objetos destinados a auto-aprendizagem, com uma alta interação, onde se distinguem oito tipos:

1. *Simulação Jogo de Roles*: este tipo de objeto permite ao aprendiz a construir e provar seu próprio conhecimento e habilidades inter atuando com a simulação de uma situação real. Trabalha com ambientes virtuais.

2. *Simulação de* Software: permite aos estudantes praticar tarefas completas com o uso de ambientes gráficos.

3. *Simulação de Hardware:* o uso de objetos de simulação de *hardware* que permite aos aprendizes obter conhecimentos de determinadas tarefas.

4. *Simulação de Códigos:* este tipo de objeto permite que o aprendiz aprenda técnicas completas da codificação de software.

5. *Simulação Conceitual:* ajudam os aprendizes a relacionar conceitos através de exercícios práticos.

6. *Simulação de Modelos de Negócios:* são objetos que permitem ao aprendiz controlar e manipular um conjunto de variáveis em uma companhia virtual para aprender a administrar uma situação real.

7. *Laboratórios On-line:* este tipo de objeto a aprendizagem de tópicos relativos a tecnologias de informação.

8. *Projetos de Investigação:* são objetos associados a atividades completas que impulsionam os aprendizes os comprometimentos através de exercícios com áreas bem específicas.

• **Objetos de Avaliação**: são objetos que têm a função de conhecer o nível de conhecimentos de um aprendiz. Divide-se em quatro tipos:

1. *Pré-avaliação*: são objetos que têm a função de verificar os conhecimentos dos aprendizes antes do processo de aprendizagem.

2. *Avaliação de Proficiência:* são objetos que servem para medir se o aprendiz assimilou determinados conhecimentos específicos para poder seguir adiante.

3. *Testes de Rendimentos:* este tipo de objeto possibilita medir a habilidade de um aprendiz em uma tarefa específica; normalmente este tipo de objeto se usa com objetos de simulação.

4. *Pré-teste de Certificação*: usado, geralmente, no final de um programa orientado a certificação e são usados em dois modos: estudo e certificação. Na modalidade de estudo é maximizada a aprendizagem entregando ao aprendiz uma lista dos erros cometidos, e na certificação é similar a um exame final.

2.3.3 Qualidade de um objeto de aprendizagem

Sabe-se pela literatura que a qualidade está relacionada diretamente à satisfação das necessidades implícitas dos clientes. Avaliar a qualidade de um objeto de aprendizagem requer definir o que avaliar e quando avaliar, isto é, avaliar um objeto ao longo do processo de construção e avaliar o produto pronto. Isto requer analisar alguns aspectos relevantes, como por exemplo:

- no aspecto manutenibilidade (pode ser consertado?);
- no aspecto da usabilidade (ele foi projetado para o usuário?);
- no aspecto portabilidade (é possível usá-lo em outra máquina?);
- no aspecto reusabilidade (é possível reutilizar parte do objeto?);
- no aspecto interoperabilidade (é possível compor uma interface com outro sistema?).

Para esta tarefa a literatura apresenta uma série de normas ISO/IEC para avaliar o software educacional desde a sua construção até requisitos para avaliar pacotes de softwares, possibilitando a sua aplicação aos objetos de aprendizagem. Entretanto, ao avaliar um objeto

de aprendizagem pode-se considerar as mesmas características que Rocha (2001) considera importantes em software educacional, que são:

• **Características pedagógicas**: conjunto de atributos que evidenciam a convivência e a viabilidade de utilização de software em situações educacionais. Inclui as seguintes sub-características:

1. *ambiente educacional*: identifica o ambiente e o modelo de aprendizagem que ele privilegia;

2. *pertinência ao programa curricular*: adequado ao contexto do conteúdo;

3. *aspectos didáticos*: facilidade de uso, motivacional, conteúdos claros e corretos, carga informacional e tratamento de erros.

• **Características ergonômicas**: conjunto de atributos que evidenciam a usabilidade do software. Inclui as seguintes sub-características:

1. *facilidade de aprendizagem e de memorização*;

2. *condução*: avalia os meios disponíveis para conduzir o usuário na interação com o computador como, por exemplo, presteza, localização, legibilidade e *feedback* imediato;

3. *afetividade*: avalia se existe relação afetuosa com o usuário;

4. *consistência*: avalia se a concepção da interface é considerada idêntica em contextos idênticos e diferentes em contextos distintos;

5. *significado dos códigos e denominações*: avalia a adequação entre o objeto e sua referência;

6. *gestão de erros*: avalia os mecanismos que permitem evitar ou reduzir a ocorrência de erros e, quando eles ocorrem, verifica os mecanismos que favorecem a sua correção.

• **Adaptabilidade**: conjunto de atributos que evidenciam a capacidade do software se adaptar às necessidades e preferências do usuário e ao ambiente educacional selecionado. Inclui as seguintes subcaracterísticas:

1. *Personalização***:** avalia a facilidade de uma personalização;

2. *Adequação ao ambiente*: avalia se o software é adequado ao modelo e aos objetivos educacionais pretendidos.

• **Documentação**: conjunto de atributos que evidenciam se a documentação para a instalação e uso do software está completa. Inclui as seguintes sub-características:

1. *mecanismo de Ajuda (Help on-line)*: avalia se existe ajuda;

2. *documentação do usuário*: avalia a facilidade de uso do sistema.

• **Portabilidade**: conjunto de atributos que evidenciam a adequação do software aos equipamentos do laboratório de informática. Inclui as seguintes sub-características:

1. *adequação tecnológica*: avalia a compatibilidade das tecnologias de software e hardware utilizadas com a do mercado;

2. *adequação aos recursos da instituição educacional*: avalia a compatibilidade de software e hardware usados na instituição.

• **Retorno do investimento**: conjunto de atributos que avalia o investimento na aquisição do software: inclui a sub-característica:

1. *preço e taxa de retorno*: avalia se o preço é compatível com suas características e se a taxa de retorno da utilização do software é compatível com o investimento.

Além dos aspectos específicos levantados para a avaliação de um software educacional devem-se considerar alguns fatores adicionais quanto a qualidade de objetos de aprendizagem como, por exemplo: a propriedade de reusabilidade para um objeto de aprendizagem é muito importante, sendo que não existe nas referências pesquisadas modelos ou métodos que avaliem um objeto de aprendizagem quanto à possibilidade de reuso. Isto diferencia em parte a avaliação de softwares de outras categorias. Assim, a preocupação com a reusabilidade, granularidade, acessibilidade e a catalogação dos metadados são algumas das preocupações deste trabalho, que vêm ao encontro dos estudos de integração das partes ergonômica e pedagógica. Uma tentativa de avaliar um software é verificar as métricas deste software, isto é, verificar uma ampla variedade de medidas que Pressman divide em categorias (Pressman, 1995). A primeira divisão é composta por:

• métricas de produtividade: concentram-se na saída do processo de engenharia de software;

• métricas de qualidade: são métricas derivadas antes do software entrar no mercado. Uma medida da adequação ao uso do software;

• métricas técnicas: concentram-se nas características do software como, por exemplo, complexidade lógica e grau de modularidade.

E a segunda divisão é composta por:

• métricas orientadas ao tamanho: são medidas diretas do software e do processo por meio do qual ele é desenvolvido;

• métricas orientadas à função: são medidas indiretas do software e do processo por meio do qual ele é desenvolvido. Concentra-se na funcionalidade ou utilidade do programa. Os valores do domínio da informação são definidos como: número de entrada de usuários; número de saída de usuários; número de consulta de usuários; número de arquivos; número de interfaces externas;

• métricas orientadas a seres humanos: compilam informações sobre a maneira segundo a qual as pessoas desenvolvem software e percepções humanas sobre a efetividade das ferramentas e métodos.

Estas métricas são orientadas a medir a utilização real do software, mas poucos descrevem diretamente a reusabilidade de um objeto de aprendizagem. Chidamber & Kemerer (1994) desenvolveram métricas ditas ‘clássicas’ para projetos de sofware “orientados a objetos” (*Object Oriented Design*) que podem ser utilizadas para medir a reutilização de um objeto de aprendizagem. São elas:

• MPC - Métodos Ponderados por Classe: resulta na agregação da complexidade dos métodos de uma classe dada. Esta métrica poderia ser utilizada como medida de predição da reusabilidade da classe;

• PAH - Profundidade da Árvore de Herança: esta métrica está relacionada com a reusabilidade desde o ponto de vista que ‘classes que são mais profundas na árvore de herança são mais complexas. O conceito-chave de herança pode ser uma simples inclusão de etiqueta em um campo do metadado tendo assim um metadado mais detalhado;

• AOC - Acoplamento entre Objetos das Classes: esta métrica tem uma translação direta em términos de relações entre os objetos de aprendizagem, nos quais podem ser definidos baixo

nos marcos atuais dos metadados como LOM. Nesta métrica os objetos de aprendizagem têm como consequência grande nível de granularidade;

• FCM - Falta de Correção dos Métodos: esta é uma medida de carência e que seu alto grau de FCM resulta na dificuldade de reutilização dos objetos como também aumenta a dificuldade de compreensão. Para avaliar a utilidade desta métrica para objetos de aprendizagem, requer-se uma analogia com os atributos de uma classe.

Técnicas de avaliação de reusabilidade dos objetos de aprendizagem podem ser, ao menos em parte, automatizadas. Podem ser encontradas muitas analogias entre métricas clássicas de reusabilidade de software e as características próprias dos objetos de aprendizagem, especialmente na analogia da granularidade como uma forma de complexidade e as considerações sobre as dependências dos objetos de aprendizagem. Sendo uma das características mais importantes dos objetos de aprendizagem, considerada a idéia central do desenho moderno dos conteúdos digitais de aprendizagem, a reusabilidade dos objetos de aprendizagem é um conceito difícil de se caracterizar devido a sua natureza multidimensional e inclusive aspectos como o formato, os conteúdos e considerações sobre os metadados. A reusabilidade em diversos contextos educacionais requer um desenho cuidadoso dos conteúdos e seus registros com metadados associados de tal forma que sejam suficientemente consistentes e completos.

Para Cuadrado-Gallego (2005) a reusabilidade pode ser avaliada por pessoas considerando três aspectos inter-relacionados:

• a qualidade da separação entre o conteúdo e apresentação;

• a qualidade do registro de metadados, especialmente a facilidade de compreensão, claridade e precisão do contexto educacional no qual está inserido;

• o desenho das instruções para cada um dos contextos educacionais dirigidos.

E, para Sicília (2005), a reusabilidade tem outros aspectos:

• *técnico de formato*: implica que os materiais estão formatados de acordo com certas regras e convenções;

• *técnico de interpretação*: implica que os metadados utilizados tenham uma orientação a habilitar certas funcionalidades automatizadas conhecidas, de maneira precisa. LOM não é suficiente para esta área;

• *desenho instrucional*: de maneira que o desenho dos conteúdos e sua granularidade estão orientados a sua reutilização, pensando em possíveis transtornos de uso futuro.

A prática da criação de metadados também é um pré-requisito para elaboração de métricas de objetos de aprendizagem confiáveis. Quanto maior é o grau de detalhes dos metadados descritivos, maiores são as possibilidades efetivas de reuso. Entretanto, com um simples acesso à Internet, o usuário tem à sua disposição a possibilidade de uso e de reuso de um objeto de aprendizagem.

Tratando-se de objetos específicos para métodos numéricos, surgem novas características importantes como a da *qualidade da informação* que deve avaliar se os conteúdos são corretos, fidedignos e a carga informacional compatível com o tema e aspecto da *confiabilidade do objeto* onde se deve ter a preocupação com a correção dos cálculos com alto grau de exatidão.

Como o foco do trabalho é aplicações na área de otimização e programação linear (Pesquisa Operacional), a seguir será colocado sucintamente alguns conceitos e definições em modelagem matemática.

2.4 Modelagem Matemática

De acordo com Bassanezi (2002), "Modelagem Matemática consiste na arte de transformar problemas da realidade em problemas matemáticos e resolvê-los interpretando suas soluções na linguagem do mundo real". Esses problemas partem do interesse do próprio aluno e os conteúdos matemáticos abordados são gerados a partir do tema a ser problematizado. A busca de uma solução é feita por meio de um Modelo Matemático, que representa aquela situação, e uma consequente validação, que envolve a verificação da solução encontrada. Dessa forma, a Modelagem leva à aquisição de conhecimentos matemáticos e a sua consequente aplicação, o que não acontece no ensino "tradicional" em que o aluno não sabe onde aplicar os conhecimentos adquiridos e, por vezes, nem mesmo o próprio professor.

Num ambiente de aprendizagem com Modelagem Matemática, a situação do professor é de um mediador, orientador do trabalho e, o que é considerado de maior relevância, de aprendente. Nesse ambiente, o professor não sabe tudo, ele também aprenderá e crescerá a cada trabalho proposto. Essa é uma situação em que o docente terá de aprender a lidar, a

insegurança de não ter o controle em suas mãos; é importante estar aberto e valorizar os conhecimentos dos alunos, pois é nessa interação que se dará a aprendizagem.

2.4.1 Pesquisa Operacional

O termo Pesquisa Operacional surgiu na Inglaterra como *Operational Research*, nos Estados Unidos como "*Operations Research*", "Investigação Operacional" em Portugal e "*Investigación Operativa*" nos países hispânicos. Essa ferramenta foi utilizada pela primeira vez em 1938 para designar o estudo sistemático de problemas estratégicos decorrentes de operações militares realizadas na época.

Pode-se dizer de forma mais abrangente que Pesquisa Operacional é um enfoque científico e tecnológico sobre tomada de decisão, sendo chamada também de ciência e tecnologia de decisão. O componente científico relaciona-se a idéias e processos para articular e modelar problemas de decisão, determinando os objetivos do tomador de decisão e as restrições sob as quais de deve operar. Também está relacionado a métodos matemáticos para otimizar sistemas numéricos resultantes da utilização de dados no modelos. O componente tecnológico refere-se a ferramentas de software e hardware para coletar e comunicar dados, e organizar esses dados, usando-os para gerar e otimizar modelos e reportar resultados, ou seja, a pesquisa operacional está se tornando um importante elemento nas metodologias de tecnologia da informação (Arenales *et. al*, 2007).

2.4.2 Programação Linear

Segundo Puccini e Arenales (1972, 2007), os problemas de programação linear referem-se à distribuição eficiente de recursos limitados entre atividades competitivas, com a finalidade de atender a um determinado objetivo, por exemplo, maximização de lucros ou minimização de custos. Em se tratando de programação linear, esse objetivo será expresso por uma função linear, a qual dá-se o nome de função objetivo.

É claro que é necessário dizer quais atividades que consomem cada recurso, e em que proporção é feito esse consumo. Essa informação será fornecida por equações ou inequações lineares, uma para cada recurso. O conjunto dessas equações ou inequações lineares dá-se o nome de restrições do modelo.

A Equação (1) apresenta um modelo de maximização:

$$MAX\_Z = c_1x_1 + c_2x_2 + ... + c_nx_n \quad \textbf{(1)}$$

Sabendo-se que $x_1, x_2, ..., x_n$ devem satisfazer as seguintes restrições:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ...a_{1n}x_n &\le b_1 \\ a_{21}x_1 + a_{22}x_2 + ...a_{2n}x_n &\le b_2 \\ &..... \\ a_{m1}x_1 + a_{m2}x_2 + ...a_{mn}x_n &\le b_m \end{aligned} \quad \textbf{(2)}$$

e que

$$x_1 \ge 0;\ x_2 \ge 0;\ ...;\ x_n \ge 0; \quad \textbf{(3)}$$

Pode-se apresentar este modelo de forma mais compacta, ou seja:

Maximizar $Z = \sum_{j=1}^{n} c_j x_j$ sujeito às restrições $\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad (i = 1,2,3....,m)$

e

$$x_j \ge 0 \quad (j = 1,2,3....,n)$$

Este modelo pode ser associado a uma empresa que tem m recursos disponíveis para realização de n atividades. O b_i seria a quantidade de recursos i disponíveis para as n atividades, x_j seria o nível de produção da atividade j (x_j são as incógnitas do problema), c_j o lucro unitário do produto j, a_{ij} a quantidade do recurso i consumida na produção de uma unidade do produto j.

Verifica-se então, que a função objetivo a ser maximizada representa o lucro total da empresa nessas n atividades. As m restrições informam que o total gasto de recursos i nas n atividades tem de ser menor ou, no máximo, igual à quantidade b_i, disponível daquele recurso.

As restrições $x_j \geq 0$ eliminaram a possibilidade de níveis negativos para as diversas atividades.

Geralmente, existem inúmeras maneiras de distribuir os escassos recursos entre as diversas atividades, bastando para isso que essas distribuições sejam coerentes com as equações de consumo de cada recurso, ou seja, que elas satisfaçam as restrições do problema. Entretanto, deseja-se achar aquela distribuição que satisfaça as restrições do problema e que alcance o objetivo desejado, isto é, que maximize o lucro ou minimize os custos. Essa solução é chamada de solução ótima (Arenales *et. al*, 2007).

2.5 Considerações

Este capítulo faz uma breve descrição sobre arquitetura de software, ressaltando alguns pontos necessários para composição deste trabalho. Conceitos e definições são estabelecidos, descrevendo uma introdução sobre estilos arquiteturais. Foram abordados também conceitos e definições de componentes de software, fazendo uma descrição de formas de adaptação e conexão de componentes, entre outras. Este assunto fornece uma base concreta para a implementação e concretização de ambientes educacionais, valorizando a prática do reuso de software, que representa a chave para o aumento da produtividade e qualidade no desenvolvimento de ambientes integrados.

Neste capítulo se caracterizaram os objetos de aprendizagem sob a perspectiva de suas características, classificações e esforços de padronização. Mesmo podendo ser considerados como um tipo de software educacional, os objetos de aprendizagem têm um processo de desenvolvimento que preservam uma especificidade própria. O processo remete a questões sobre como as pessoas aprendem pelo tratamento dado aos conteúdos endereçados pelos objetos de aprendizagem. Finalizando este capítulo faz-se uma breve explanação sobre modelagem matemática mais comumente utilizada em Pesquisa Operacional, com o intuito de familiarizar com os termos da área.

3 DESENVOLVIMENTO

Este capítulo apresenta a proposta de desenvolvimento de uma arquitetura para software educacional baseada em componentes, embasada em alguns conceitos de reutilização, arquitetura e componentes de software, como a definição tradicional de arquitetura de software que especifica o sistema em termos de componentes computacionais e os seus relacionamentos (Vasconcelos, 2004).

No desenvolvimento desta arquitetura, primeiramente foi necessário encontrar um conjunto específico de informações que compõem a tecnologia de aprendizagem, e então, especificar o conjunto de componentes que a arquitetura suporta, as exigências funcionais que emergem e a especificação da forma de conexão utilizada para prover a interação entre componentes da arquitetura.

Inicialmente é apresentada a arquitetura básica do modelo IEEE/LTSA, seguido de sua especialização de maneira a torná-la menos abstrata. Para o domínio de Pesquisa Operacional (Programação Linear) serão realizados levantamentos sobre requisitos necessários para um ambiente direcionado a este domínio.

3.1 Componentes do Sistema IEEE/LTSA

A seguir, na Figura 8, é apresentado o modelo de referência IEEE / LTSA, que descreve os processos, os repositórios, e os fluxos do LTSA. Os processos são descritos em termos dos limites, das entradas, dos processos (funcionalidades), e das saídas. Os repositórios são descritos pelo tipo de informação armazenado, a busca, recuperação, e atualização de métodos. Os fluxos são descritos em termos de conectividade (de sentido único, em dois sentidos, de conexões dinâmicas, de conexões estáticas, entre outros) e o tipo de informação através do fluxo.

Os componentes de Arquitetura de Sistemas de Tecnologia de Aprendizado observados na Figura 8 são (IEEE P1484.1/D9, 2001):

- **Processos**: entidade de aprendiz, avaliação, técnico, entregador.

- **Repositório de dados:** registro de aprendiz, recursos de aprendizagem.
- **Fluxos:** especificando preferências, comportamento, avaliação das informações, informações do aprendiz, consulta, catalogo de informações, locador, contexto de aprendizagem, multimídia, contexto de interação.

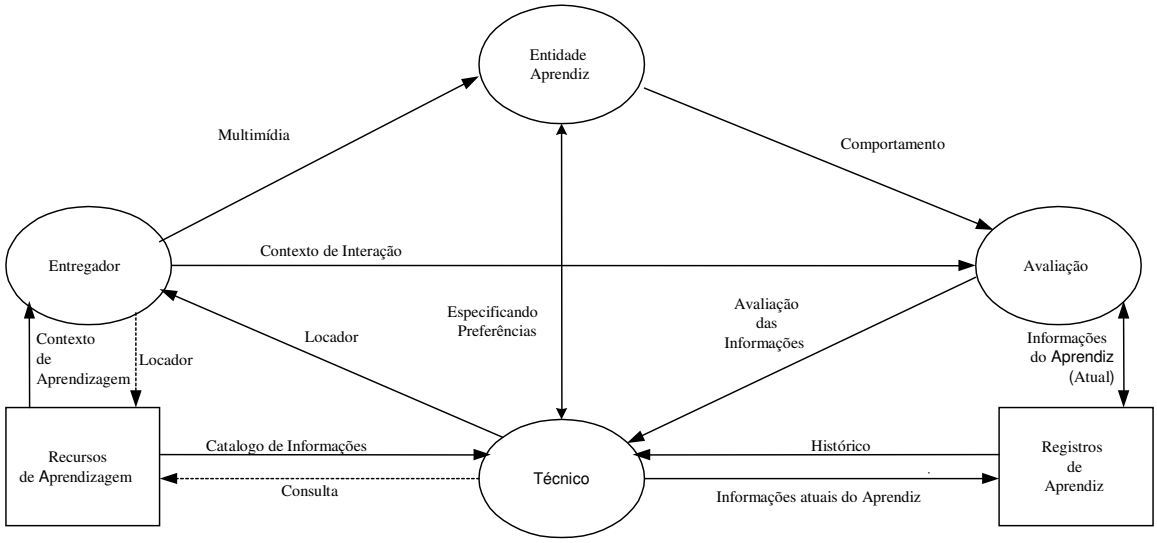

Figura 8 – Modelo IEEE/LTSA (IEEE P1484.1/D9, 2001).

Baseado no modelo IEEE/LTSA foi desenvolvido uma especificação para mesmo, de modo a facilitar a modelagem de aplicações de Pesquisa Operacional, entretanto deve-se destacar que este trabalho irá se concentrar na modelagem abstrata dos subcomponentes que serão identificados a partir do modelo de referência IEEE/LTSA, para compor esta arquitetura. A definição de interfaces será realizada de forma a identificar algumas das principais funcionalidades e serviços destes subcomponentes.

Como a especificação do Modelo IEEE/LTSA não o descaracteriza, as definiçõs dos seus componentes será realizada na próxima seção.

3.2 Definição de Componentes

Os componentes da arquitetura proposta foram especificados com base nos seguintes conceitos:

- Segundo Szyperski (2002), o que torna algum artefato de software um componente não é uma aplicação e nem uma tecnologia de implementação específica. Assim, qualquer dispositivo de software pode ser considerado um componente, desde que possua uma interface definida. Esta interface deve ser uma coleção de pontos de acesso a serviços, cada um com uma semântica estabelecida.

- De acordo com Ólafsson (1996), além da interface provida, isto é o conjunto de métodos de componentes que podem ser invocados, uma descrição de interface do componente também deve estabelecer a interface requerida, isto é, os métodos que o componente invoca.

- Pfister (1998) diz que é de grande importância que a descrição da interface de um componente seja clara, completa e concisa, caso contrário pode provocar uma má interpretação do componente e assim este termina por não servir a necessidade. Métodos formais e semiformais podem auxiliar para especificar as interfaces com o objetivo de que estas não venham a estar ambíguas.

- Segundo Garlan (1997), a especificação das interfaces dos componentes (serviços providos e requeridos) e dos conectores (comunicação entre componentes) descrevem também o comportamento do componente.

- E para finalizar, é importante ressaltar mais um conceito definido por Szyperski. O primeiro passo na elaboração de um componente, é projetar as interfaces dos componentes que irão determinar quais são as funcionalidades necessárias que deverão ser implementadas no componente e de que forma essas se comunicarão (Szyperski, 2002).

A partir destes conceitos básicos e também do modelo de referência IEEE/LTSA é que a arquitetura proposta se fundamenta. A seguir, na Figura 9, é mostrada a especificação dos componentes da arquitetura proposta.

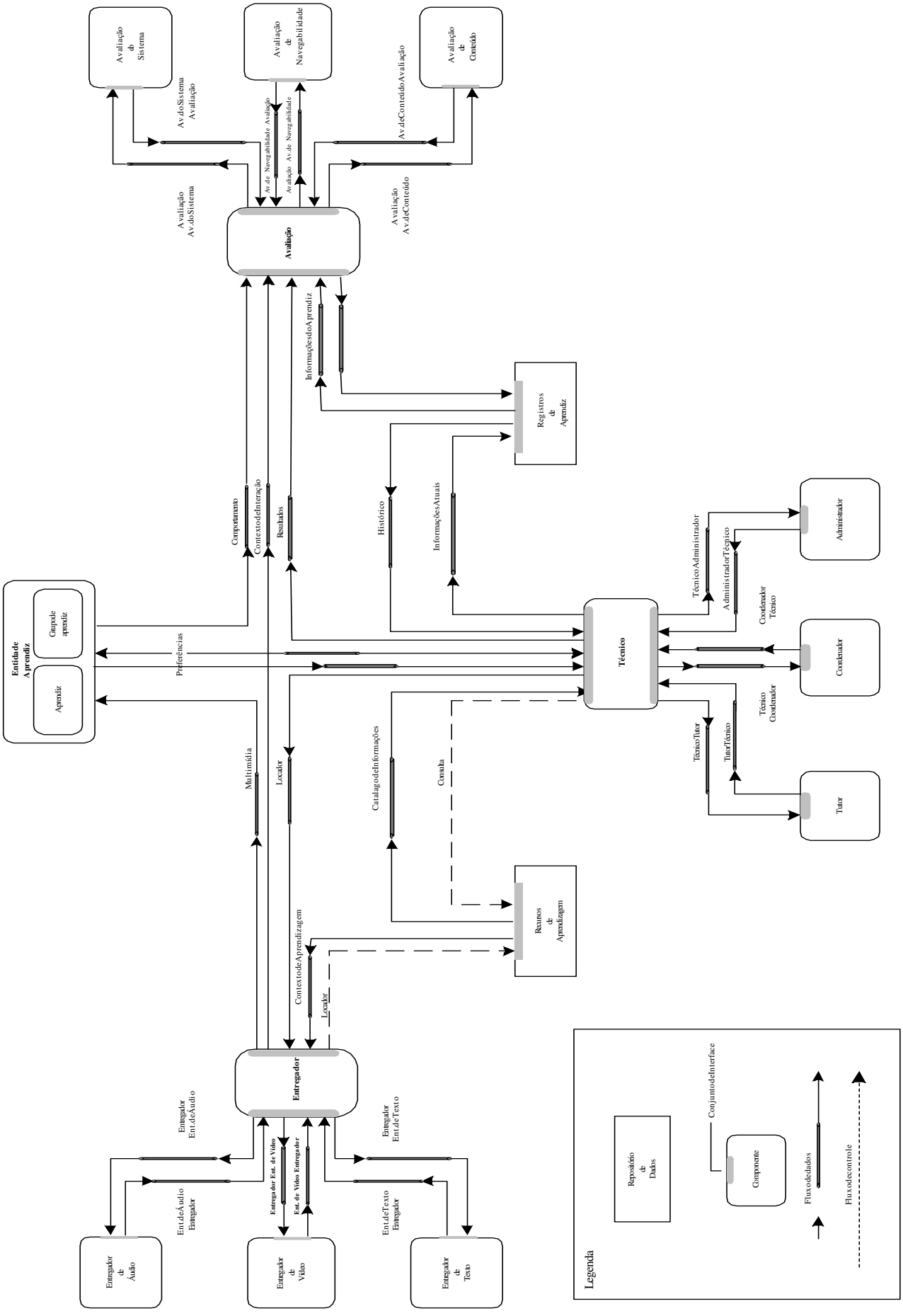

Figura 9 – Arquitetura Proposta.

3.2.1 Entidade Aprendiz

Para a entidade aprendiz não se encontrou a necessidade de uma divisão em subcomponentes, além dos já existentes. Esta entidade representa todas as entidades que interagirão com qualquer aplicação desenvolvida a partir da arquitetura proposta. A definição para esta entidade a trata como a abstração de uma pessoa como um aprendiz, ou melhor, esta entidade representa um único aprendiz, um grupo de aprendizes que aprendem colaborativamente, ou ainda, individualmente.

3.2.2 Técnico

O componente Técnico realiza a filtragem dos dados para os componentes Administrador, Tutor e Coordenador ou para outro componente que pode ser incorporado na arquitetura (um componente abstrato), ou melhor, realizará a distribuição de tarefas entre os componentes. Este componente receberá uma solicitação de serviço, identificando o serviço solicitado através dos parâmetros que foram passados, assim, o componente adequado será solicitado. Posteriormente receberá a resposta do serviço que foi requerido por ele, podendo realizar uma nova filtragem, para enviar a resposta ao componente que solicitou o serviço.

Serviços requeridos:

O Técnico poderá requerer alguns serviços que sejam necessários para suas funcionalidades, ou mesmo para tomadas de decisões. Quando o Técnico requer as preferências do aprendiz verifica informações relevantes para suas atitudes, que podem ser acrescidas de dados contidos no histórico, que está armazenado nos registros do aprendiz. Os resultados requeridos pelo Técnico podem ser fornecidos pelo catalogo de informações através dos recursos de aprendizado.

Serviços fornecidos:

As preferências podem ser enviadas à entidade aprendiz, sendo que isto ocorre em via dupla,

do técnico para o aprendiz e vice-versa. O Técnico submete consultas (fluxo de controle) aos recursos de aprendizagem, que retornam via catálogo de informação. Ele atualiza/altera informações recentes (informações atuais) nos registros do aprendiz. Também são informados dados ao entregador sobre condições/situações que sejam necessárias para este, e envia resultados ao processo avaliação para que o mesmo possa tomar algumas decisões.

O objetivo principal no refinamento (especialização) do que será apresentado, foi a possibilidade de agrupar funcionalidades e serviços do componente Técnico em subcomponentes (componentes) que tivessem funcionalidades bem determinadas e comum, caracterizando as partes específicas de um Técnico. Com isto espera-se que a incorporação destes novos elementos torne mais fácil a identificação de componentes reusáveis e, até mesmo, a implementação destes.

Cabe ressaltar aqui que a especificação destes subcomponenes não tornou o modelo direcionado, pois os mesmos continuam sendo descritos e especificados de maneira abstrata. A seguir, serão apresentados os serviços e funcionalidades dos subcomponentes do filtro Técnico.

Administrador

Conjunto de ações que facilitam a integração aprendiz / sistema, ou melhor, este é "responsável" pelo armazenamento e manutenção dos recursos de aprendizagem, pela validação de usuários e pelo monitoramento das atividades dos sistemas (componentes) em uso.

Serviços requeridos:

O Administrador, como os demais subcomponentes, requerer um subconjunto de serviços do técnico, os quais são importantes para sua funcionalidade. Neste caso, são importantes requer as preferências do aprendiz. Os resultados requeridos pelo técnico podem ser fornecidos do catálogo de informações dos recursos de aprendizado e dados do aprendiz para validação do mesmo.

Serviços fornecidos:

Administrar usuários é um conjunto de serviços que oferece funcionalidades para a gerência de usuários podendo aceitar usuários (autenticação de usuário); alterar, incluir ou excluir dados desses. Administrar recursos de aprendizagem compreende um conjunto de serviços que realizará a administração desses recursos podendo incluir, alterar, excluir os mesmos, e também poderá disponibilizar os recursos aceitando, ou não, a requisições de atores do sistema, tornado está parte direcionada principalmente para a autoria de materiais. Outra atividade é monitorar desempenho, conjunto de serviços que fornecerá suporte para o monitoramento de desempenho dos sistemas (ou componentes), isto é, a certificação do funcionamento dos sistemas de acordo com sua especificação. Através dos resultados de avaliação, o componente Técnico solicitará informações atuais das avaliações do sistema, assim o componente Administrador gerará visualizações das funcionalidades dos componentes em atividade.

Coordenador

Conjunto de ações responsável pelo planejamento, desenvolvimento e execução do sistema de aprendizagem (componentes). Como por exemplo, a negociação de preferências com os aprendizes e direcionamento / restrição de conteúdo proposto para um processo de aprendizagem.

Serviços requeridos:

Neste caso, o Coordenador necessita de todos os serviços do técnico que são: requerer preferência, histórico, consulta e resultados.

Serviços fornecidos:

O coordenador poderá validar preferências através de serviços que farão negociações de preferências com atores que não sejam o aprendiz, por exemplo, o professor/tutor, ou algum responsável pelo aprendiz. Para administrar conteúdo/tópicos (consulta) os serviços devem oferecer funcionalidades para direcionar e/ou restringir o conteúdo que pode ser oferecido em

um processo de aprendizagem (por exemplo, pode ser definido o plano de aprendizado da disciplina/conteúdo). Este componente oferecerá serviços de pesquisa de tópicos, validação de tópicos sugeridos por outros atores e mesmo a disponibilizarão destes tópicos, redigir planos e estratégias de ensino.

No gerenciamento dos registros de aprendiz alguns serviços oferecem funcionalidades para visualizar e atualizar dados do aprendiz referente ao seu grupo. Estes dados serão enviados através do fluxo de informações atuais. Para administrar avaliações o serviço está relacionado com informações relevantes ao coordenador, as quais podem ser dados dos alunos, ou dos Tutores.

Outras atitudes necessárias são administrar planos de aprendizado fornecendo mecanismos para a visualização e validação dos planos realizados pelo Tutor e monitorar desempenho com serviços que devem prover mecanismos para o monitoramento de atividades dos atores do sistema.

Tutor

Conjunto de ações que atuam na orientação do aprendiz podendo ser visto como um facilitador (mediador), monitorando as atividades do aprendiz e negociando preferências com o aprendiz, entre outros serviços.

Serviços requeridos:

Também faz necessário ao Tutor todos os serviços do técnico que são: requerer preferência, histórico, consulta e resultados.

Serviços fornecidos:

O Tutor deve validar preferências através de serviços que farão negociações de preferências com o aprendiz; juntamente com a administração de conteúdo (consulta) que fará pesquisa de conteúdo (consulta), inserção de conteúdo e validação de conteúdo (locador).

Deverá administrar registro de aprendiz com um conjunto de serviços que oferecem

funcionalidades para a alteração, inclusão e exclusão de informações referentes ao aprendiz, ou seja, dados que serão armezenados formando o perfil do aprendiz. Essas informações podem ser desde preferências até resultados de avaliações referentes ao processo de ensino de um grupo de aprendizes, como por exemplo, uma sala de aula, uma sala virtual, entre outros. Também deve administrar avaliações - referentes a informações do aprendiz, administrar planos de aprendizado - compor planos de ensino e monitorar desempenho – com serviços que tenham funcionalidades para realização de monitoramento do ator aprendiz, serviços que estão relacionados com as tarefas que o aprendiz realizará.

3.2.3 Avaliação

Analogamente ao componente Técnico, o componente Avaliação realizará a filtragem de dados para os componentes, Avaliação do Sistema, Avaliação de Navegabilidade e Avaliação de Conteúdo, ou para outro componente que pode ser incorporado na arquitetura (um componente abstrato), ou melhor, realizará a distribuição de tarefas entre os componentes. Este componente receberá uma solicitação de serviço, identificando o serviço solicitado através dos parâmetros que foram passados, assim, o componente adequado será solicitado. Posteriormente receberá a resposta do serviço que foi requerido por ele, podendo realizar uma nova filtragem, para enviar a resposta ao componente que solicitou o serviço.

Serviços**:**
As informações sobre atividades são indispensáveis para as ações do processo Avaliação, essas informações compreendem o comportamento do aprendiz, contexto de interações utilizadas. Também resultados como de avaliações, competências e habilidades, oriundas do técnico, são importantes para a avaliação direcionar suas atitudes. Requer informações anteriores úteis do aprendiz, que são acessadas nos registros do aprendiz. Vários dados são necessários para administrar informações de avaliação.

Para verificar os subcomponentes foram analisados os aspectos de possibilidade de adaptação do sistema, onde a avaliação do sistema permitiria que o sistema se auto-reorganizasse. Como é comum num sistema de aprendizagem a verificação do comportamento através das atitudes (passos) que o aprendiz realiza, a avaliação de navegabilidade consiste em acompanhar os

caminhos do aprendiz. E finalizando foi acrescentada a avaliação do conteúdo que será composta juntamente com o material, verificando as estratégias de ensino que se pretende enfocar na aplicação educacional.

Avaliação do Sistema

Este subcomponente tem a função de avaliar o grau de satisfação, motivação e as dificuldades no uso do ambiente, em suma, avaliar o desempenho de todo o sistema, permitindo assim, uma reorganização no formato do ensino / aprendizagem com o intuito de se adaptar as necessidades e preferências do aprendiz e do ambiente.

Serviços Requeridos:

Solicitar e/ou receber informações que os componentes estão recebendo e quais eles estão enviando. Com isso avalia-se o meio (fluxo) em que esses dados estão trafegando, e verifica se o comportando do componente está de acordo com suas especificações (para que possa ser ajustado e/ou adaptado se necessário).

O componente avaliação está recebendo informações da aplicação (outros componentes) através de fluxos ou agentes, assim temos que considerar a confiabilidade das informações que serão avaliadas.

Serviços Fornecidos:

Este componente viabilizará a avaliação da comunicação entre os componentes, assim ele pode verificar se a informação enviada por um componente está chegando num determinado destino.

Para avaliar serviços terá de viabilizar a avaliação dos serviços fornecidos pelos componentes da aplicação. Objetiva-se com esta avaliação aferir se os serviços que estão sendo solicitados pela aplicação (outros componentes) estão sendo realmente “processados” e/ou realizados por um determinado componente. Para isto, o componente vai solicitar / receber informações / notificações de quais os serviços foram solicitados e quais foram realizados pelo componente, assim ele pode aferir e produzir medidas. Estas medidas podem ser informações que auxiliarão no ajuste ou adaptação de componentes.

Avaliação de Navegabilidade

Espera-se com este componente registrar toda participação, navegação (ou caminho) do aprendiz para tentar alcançar seu objetivo de estudo. Com isso deve-se manter uma base que registre os itens visitados e o tempo gasto em cada tópico. Assim, poderá ser útil este acompanhamento para uma avaliação de desempenho do aprendiz.

Serviços requeridos:

Neste caso o processo irá solicitar informações/respostas do aprendiz (via comportamento) e dados necessários para interpretar as "respostas" do aprendiz (via contexto de interação), isto significa que será solicitado o comportamento do aprendiz.

Serviços Fornecidos:

Para se registrar itens visitados um conjunto de serviços irá capturar os eventos de interação do aprendiz com o ambiente através do comportamento. Para atualizar registros do aprendiz os serviços terão algumas atitudes associadas com esta função. Também deverá gerar resultados, através de um conjunto de serviços irá fornecer informações da trajetória do aprendiz no ambiente.

Avaliação de Conteúdo

A *priori* esta avaliação deve estar em consonância com a proposta pedagógica da aplicação a ser modelada. Definida a metodologia educacional, este componente irá produzir métodos pré-definidos de avaliação, tais como, práticas de medida e teste.

Serviços requeridos:

Este processo irá solicitar informações / respostas do aprendiz (via comportamento) e dados necessários para interpretar as "respostas" do aprendiz (via contexto de interação).

Ele irá atualizar as informações do registro de aprendiz (via informações do aprendiz) por meio da avaliação.

Serviços Fornecidos:

Numa avaliação colaborativa é bastante comum que cada participante tenha de avaliar a participação dos seus iguais e a qualidade do produto resultante do trabalho destes. A avaliação de todo o processo requer serviços que permitam aos aprendizes constante observação das ações dos participantes do grupo e da realização de reuniões com a finalidade de discutir esta questão. É possível também a avaliação dos documentos criados pelos participantes do grupo, que representam o produto do seu trabalho.

A avaliação participativa irá medir, quantificar a participação dos indivíduos (entidade aprendiz) quanto à produção de materiais.

A avaliação tradicional consiste em medidas de determinação do conteúdo assimilado por um aprendiz, realizada em conformidade com o material produzido.

3.2.4 Entregador

Analogamente aos outros componentes já citados, o componente Entregador realizará a filtragem de dados para os componentes, Entregador de Áudio, Entregador de Vídeo e Entregador de Texto ou para um outro componente que possa ser incorporado na arquitetura (um componente abstrato) e realizará a distribuição de tarefas para os componentes. Este componente receberá uma solicitação de serviço, identificando o serviço solicitado através dos parâmetros que foram passados, assim, o componente adequado será solicitado. Posteriormente, receberá a resposta do serviço que foi requerido por ele, realizando uma nova filtragem, para enviar a resposta ao componente que invocou o serviço. Os três componentes identificados aqui realizam a mesma tarefa, a única diferença está no tipo de mídia que será transformado.

Serviços requeridos:

O Entregador deverá solicitar o conteúdo de aprendizagem, transferindo o controle da aplicação, utilizando o Locador, para os Recursos de Aprendizagem.

Serviços fornecidos:

Ele enviará recursos de aprendizagem que realizará a transmissão de dados para o Entregador de Áudio, Vídeo e/ou Texto, dependendo de qual realizou a solicitação, ou mesmo o componente Entregador pode ter solicitado um serviço e enviado os Recursos de Aprendizagem junto com esta solicitação.

Para enviar a apresentação deverá ser realizado o envio da apresentação para a multimídia, porém, esta apresentação será provida pelo entregador de Áudio, Vídeo ou Texto.
No envio do Contexto de Interação é necessário que o processo avaliação realize sua tarefa. Esses dados representam a forma como o Aprendriz interagiu com o sistema. O envio de consulta (via Locador) é um fluxo de controle.

A funcionalidade e serviços dos componentes identificados nesta seção possuem as mesmas caracteristicas, sendo diferenciadas na mídia em questão. Geralmente é comum ter um componente específico para apresentar uma mídia, o qual é ligado (*plugin*) a outros componentes. A seguir, serão apresentados os componentes identificados nesta seção.

Entregador de Áudio

Serviços requeridos:

Requer o Conteúdo de Aprendizagem que deverá solicitar o conteúdo de aprendizagem, através do Tubo Entregador_Ent.deAudio (fluxo).

Serviços Fornecidos**:**
Interpretar os Recursos de Áudio consiste num conjunto de serviços que fornece

funcionalidades para reconhecer e interpretar os Recursos de Aprendizagem que lhe foram enviados pelo Entregador. Após ter reconhecido e interpretado os recursos o componente Entregador de Áudio realiza o envio para o Entregador.

Gerar Apresentação de Áudio oferece funcionalidades para a geração da Apresentação de Áudio. Após o reconhecimento e interpretação dos Recursos de Aprendizagem, esses serão transformados em uma apresentação de áudio, esta transformação normalmente resulta em um "arquivo". O formato deste arquivo deve ser produzido levando em consideração qual o meio que esta mídia de áudio vai ser apresentada.

Para enviar a Apresentação de Áudio consiste em fornecer funcionalidades para a comunicação e envio da Apresentação de Áudio para o Entregador (através do Tubo Ent.deAudio_Entregador), que por sua vez encaminhará a seu destino através do componente Multimídia.

Entregador de Vídeo

Serviços requeridos:

Requerer o Conteúdo de Aprendizagem que deverá solicitar o conteúdo de aprendizagem, através do Tubo Entregador_Ent.deVideo (fluxo).

Serviços Fornecidos:

Interpretar os Recursos de Vídeo consiste num conjunto de serviços que fornece funcionalidades para reconhecer e interpretar os Recursos de Aprendizagem que lhe foram enviados pelo Entregador. Após ter reconhecido e interpretado os recursos o componente Entregador de Vídeo realiza o envio para o Entregador.

Gerar Apresentação de Vídeo oferece funcionalidades para a geração da Apresentação de Vídeo. Após o reconhecimento e interpretação dos Recursos de Aprendizagem, esses serão transformados em uma apresentação de vídeo, esta transformação normalmente resulta em um

“arquivo”. O formato deste arquivo deve ser produzido levando em consideração qual o meio que a mídia de vídeo vai ser apresentada.

Para enviar Apresentação de Vídeo consiste em fornecer funcionalidades para a comunicação e envio da Apresentação de Vídeo para o Entregador (através do Pipe Ent.deVideo_Entregador), que por sua vez encaminhará a seu destino através do componente Multimídia.

Entregador de Texto

Serviços requeridos:

Requerer o Conteúdo de Aprendizagem que deverá solicitar o conteúdo de aprendizagem, através do Tubo Entregador_Ent.deTexto (fluxo).

Serviços Fornecidos:
Interpretar os Recursos de Texto consiste num conjunto de serviços que fornece funcionalidades para reconhecer e interpretar os Recursos de Aprendizagem que lhe foram enviados pelo Entregador. Após ter reconhecido e interpretado os recursos o componente Entregador de Texto realiza o envio para o Entregador.

Gerar Apresentação de Texto oferece funcionalidades para a geração da Apresentação de Vídeo. Após o reconhecimento e interpretação dos Recursos de Aprendizagem, esses serão transformados em uma apresentação de texto, esta transformação normalmente resulta em um “arquivo”. O formato deste arquivo deve ser produzido levando em consideração qual o meio que a mídia de texto vai ser apresentada.

Para enviar Apresentação de Texto consiste em fornecer funcionalidades para a comunicação e envio da Apresentação de Texto para o Entregador (através do Tubo Ent.deTexto_Entregador), que por sua vez encaminhará a seu destino através do componente.

3.3 Especificação de um Ambiente de Ensino para Pesquisa Operacional

Para o levantamento dos componentes de um ambiente para o ensino de Pesquisa Operacional, mas especificamente Programação Linear, foram verificadas algumas ferramentas existentes (Leal *et al.*, 2010) e algumas abordagens com professores da área. Dentre as ferramentas verificadas destacam-se as que utilizam programação linear para resolução de problemas, sendo elas: Calc, Tora, MPL, LabPL, LINGO e GAMS; em Leal *et al* (2010) pode-se encontrara uma análise das mesmas de maneira simplificada.

Por meio desta análise foi possível identificar alguns requisitos necessários que se deseja um ambiente integrado de ensino, ou que o mesmo deva suportar para um bom conteúdo para aprendizagem.

A seguir os componentes/objetos identificados são descritos, também são analisados alguns serviços e resultados esperados pelos mesmos, esta identificação foi obtida a partir da Arquitetura Proposta (Figura 9).

APRENDIZ

A entidade aprendiz representa os alunos, individualmente ou em grupos, no ambiente de ensino, cujo objetivo é o aprendizado do conteúdo de Pesquisa Operacional, focando a Programação Linear. Esta entidade representa o foco do objeto de aprendizagem, em que as demais entidades interagirão com ela. Na figura 10 pode-se observar como na Arquitetura Proposta a entidade aprendiz é representada e sua comunicação com o componente técnico.

PROFESSOR/TUTOR

Este componente representa o Professor (pessoa física) ou mesmo sistema de tutoria inteligente. Em geral, corresponde ao professor para que se tenha a interação pessoal entre o

educador e o aprendiz. A figura 10 modela a representação do componente abstrato técnico com o professor e o monitor, nesse caso o técnico corresponderia a interface entre eles.

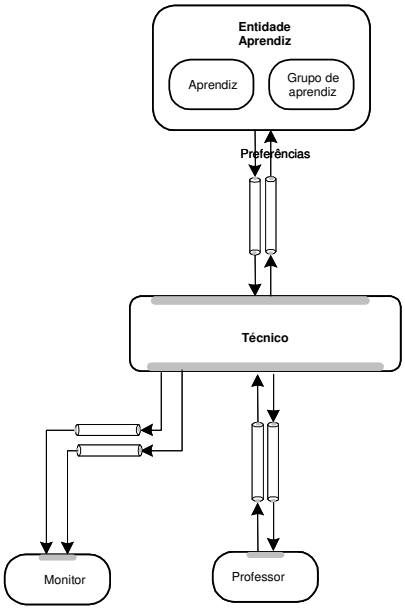

Figura 10 – Representação da entidade aprendiz e o componente técnico.

ENTREGADOR DE TEXTO

Este componente/objeto de aprendizagem deverá conter um Editor de Equações, o qual deverá ser claro e de fácil configuração, onde seja possível especificar (editar) as equações (e inequações) de maneira algébrica, conforme seção 2.4.2 (equações 1, 2 e 3); ou de maneira matricial, que é bastante familiar pelos alunos no ensino de Pesquisa Operacional.

Este componente deverá fornecer um conjunto de serviços para reconhecer e interpretar os dados de entrada (equações), verificando-os sintática e semanticamente. Assim, as informações que serão enviadas pelo Entregador para que o Recurso de Aprendizagem deverão estar adequadas de maneira a solucionar o problema modelado.

Os resultados gerados pelo Recurso de Aprendizagem deverão ser entregue ao Aprendiz de forma que a obter um Relatório dos resultados obtidos, para análise do aprendiz. Este

"Gerador de Relatório" poderá ser em forma textual ou mesmo gráfico que corresponderá aos serviços do Entregador de Texto.

Na figura 11 é modelado o componente entregador, no qual estão identificado editor, juntamente com o interpretador (linha pontilhada), que corresponderia a apresentação e verificação da entrada de equações.

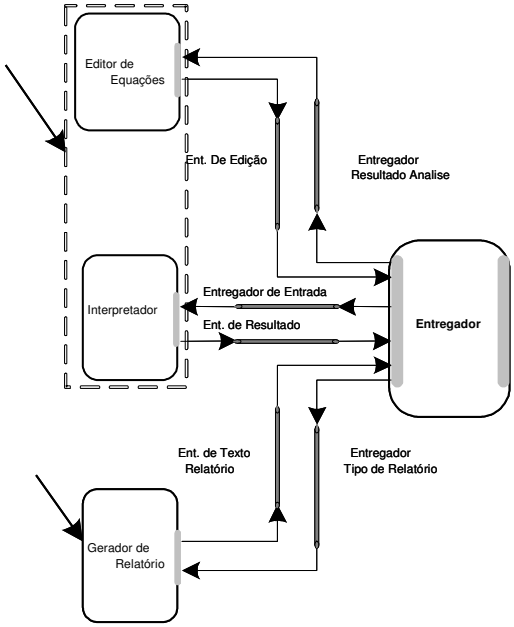

Figura 11 – Representação do componente entregador.

Para o relatório apresentado é interessante que o mesmo possa ser visualizado de maneira matricial (se o aluno desejar), constando o tempo de execução e a precisão. Como o enfoque aqui é computacional, é importante que possa obter o número e as iterações da solução, e não somente a solução, bem como em qual iteração obteve o resultado ótimo.

Neste caso, para enviar apresentação de resultados consiste em fornecer funcionalidades para a comunicação e envio da Apresentação de Texto para o Entregador, que por sua vez encaminhará a seu destino através do componente.

Assim, na Figura 11 o gerador de relatório é parte de visualização de resultados, que deverá ser possível a opção de configuração conforme preferência de utilização.

RECURSOS DE APRENDIZAGEM

Neste componente estão os recursos de utilização de ferramentas, tutorias e materiais de aprendizagem, no caso da Pesquisa Operacional (PL) é fundamental uma ferramenta para solucionar sistemas de equações e inequações, mais conhecida na comunidade científica (acadêmica) como *SOLVER*, que por meio do entregador receberá as informações necessárias para seu funcionamento. O entregador será um filtro de interação entre o Editor (entregador de texto) e o *Solver*, assim a entrada da ferramenta utilizada será ajustada conforme seus requisitos de entrada.

Um fluxo de dados proveniente dos recursos de aprendizado para o processo técnico que representa os resultados do mesmo e também pode armazená-los em repositório do aprendiz.

Para este recurso, verificou-se a necessidade de uma solução passo-a-passo para um melhor entendimento no aprendizado, assim a solução pode ser visualizada sobre a caminho escolhido pela chegar a solução.

Como um recurso de aprendizagem é interessante que exista uma ferramenta de ajuda para o conteúdo disponibilizado no ambiente, assim o aluno poderá tirar suas duvidas no próprio ambiente (limitações) e também informações sobre o ambiente e conteúdos disponíveis,

Outro recurso importante é uma ferramenta para comunicação entre o aprendiz e o professor/tutor (*chat* ou tutorial), com isto as dúvidas poderão ser esclarecidas, e servirão de suporte para uma avaliação do sistema.

REGISTRO DE APRENDIZ

Um repositório de dados com informações do aprendiz, como: desempenho, preferências, modelos de soluções adotados, versões dos modelos de problemas, entre outras informações.

Os Registros de Aprendiz podem armazenar e recuperar informações sobre o passado (por exemplo, registro de histórico de aprendizagem), mas deve também manter informações sobre o presente (por exemplo, validações atuais para suspender (versões de solução) e retomar sessões) e o futuro (por exemplo, objetivos pedagógicos e informações de professores).

AVALIAÇÃO

Estes componentes são para a obtenção de informações sobre atividades que são indispensáveis para as ações do processo, no caso se houver uma avaliação do aprendiz pelo ambiente, ou mesmo uma realização de verificações pelo sistema.

Avaliação, essas informações compreendem o comportamento do aprendiz, contexto de interações utilizadas. Também resultados como de avaliações, competências e habilidades, oriundas do técnico, são importantes para a avaliação direcionar suas atitudes. Requer informações anteriores úteis do aprendiz, que são acessadas nos registros do aprendiz. Vários dados são necessários para administrar informações de avaliação.

A **Avaliação de Conteúdo** deve estar em consonância com a proposta pedagógica da aplicação a ser modelada, portanto em alguns casos não se aplica. Definida a metodologia educacional, este componente irá produzir métodos pré-definidos de avaliação, tais como, práticas de medida e teste.

A **Avaliação do Sistema** tem a função de avaliar o grau de satisfação, motivação e as dificuldades no uso do ambiente, em suma avaliar o desempenho de todo o sistema, permitindo assim, uma reorganização no formato do ensino / aprendizagem com o intuito de se adaptar as necessidades e preferências do aprendiz e do ambiente.

A avaliação deve ser o processo de interação entre a entidade aprendiz e o ambiente de ensino, no caso da Pesquisa Operacional, este processo de avaliação deve ser analisado e verificada a melhor metodologia de avaliação.

3.4 Estudo de caso – Problema de Mistura

"Os problemas deste tipo consistem em combinar materiais (produtos/matérias-primas) obtidos na natureza (ou não para gerar novos materiais e produtos, com características convenientes. Estão entre os primeiros problemas de otimização linear implementados com sucesso na prática" (Arenales *et*. al, 2007).

Assim, o modelo proposto serve para vários tipos de soluções de problemas, com o intuito de deixar mais claro a abstração do modelo procurou identificar as necessidades do problema de mistura no modelo proposto de arquitetura. As figura 9, 10 e 11 são modelos em que é possível identificar claramente os componentes descritos para o ensino de Pesquisa Operacional, neste caso Programação Linear.

Neste contexto para um problema de mistura abstrato (genérico) pode-se identificar os seguintes atores: O **aprendiz**, que seria o aluno com o interesse de estudar o conteúdo proposto. O **professor**, auxiliado por monitores, como o componente que irá propiciar o estudo dos conteúdos necessários, bem como intermediário no processo de aprendizagem. A Figura 10 é o modelo da interação entre os componentes envolvidos, bem como o fluxo da informação necessária entre esses componentes.

Ainda como foi apresentado na seção 2.4.2, será necessário para a Equação (1, 2 e 3) um editor que possa representar as equações (inequações) abaixo:

$$MAX\_Z = c_1x_1 + c_2x_2 + ... + c_nx_n \quad \textbf{(1)}$$

Sabendo-se que $x_1, x_2, ..., x_n$ devem satisfazer as seguintes restrições:

$$\begin{gathered} a_{11}x_1 + a_{12}x_2 + ...a_{1n}x_n \le b_1 \\ a_{21}x_1 + a_{22}x_2 + ...a_{2n}x_n \le b_2 \\ ..... \\ a_{m1}x_1 + a_{m2}x_2 + ...a_{mn}x_n \le b_m \end{gathered} \quad \textbf{(2)}$$

e que

$$x_1 \ge 0;\ x_2 \ge 0;\ ...;\ x_n \ge 0; \quad \textbf{(3)}$$

No caso de mistura, de forma geral, é necessário que se tenha um editor para equações (ou inequações) como é identificado na Figura 11, no componente entregador. Neste caso mais específico pode-se optar por editor matricial, como muitas ferramentas já utilizam essa representação, e também como na maioria dos casos se limitam a questões operacionais de tamanho de matriz e precisão computacional. Assim no modelo proposto o editor de texto seria um editor que por meio da modelagem matricial realizasse a representação.
Após isto, esse editor deveria verificar se modelagem matricial estivesse de acordo com o padrão de entrada de um *solver* (solucionador de equações), assim existe a necessidade de um interpretador para essa verificação, em geral as ferramenta já disponibilizam esse componente (interpretador).

No caso do *solver* que é um recurso de aprendizagem, este é um aplicativo que estará disponível para a utilização, em que o entregador através da interface irá comunicar-se com o recurso disponível.

Até esta etapa foi possível identificar que o modelo proposto está servindo de referência para um problema de aprendizado que o professor irá verificar as necessidades que o mesmo precisa, identificando os requisitos que precisa para ensinar um determinado conteúdo (problema de mistura). Por meio do modelo proposto e dos requisitos identificados ira formando um ambiente de aprendizado, com ferramentas computacionais existentes, ou por meio de um modelo tradicional de aprendizado, ou combinando ambos.

Com essas identificações de necessidades, pode-se ter a necessidade de verificação de resultados, em geral os professores querem a visualização de parte da solução ou de como foi realizado a solução. Assim, um gerador de relatórios e importante como verifica-se na Figura 11, esse gerador deverá ser apresentado ao aprendiz de forma que o mesmo consiga verificar o processo de solução da ferramenta.

Quanto a avaliação de aprendizagem, esta pode ser realizada de forma tradicional, integrando o ambiente proposto aqui com uma forma mais convencional de avaliação. Ou mesmo por processos informatizados de avaliação, como teste prontos e avaliação por navegabilidade de conteúdos.

Desta maneira, a identificação e o uso do modelo pode ser melhor compreendido quando de deseja criar (montar) um ambiente de como ensinar determinado conteúdo aos alunos.

Nota-se que um ambiente integrado para o ensino de Pesquisa Operacional deve suportar, ou mesmo possuir um componente (objetos), a integração com gerenciadores de base de dados de maneira que seja possível utilizar modelos já existentes e testados, e também modelos experimentais definidos pelo professor, que poderá definir métodos diferenciados (ou clássicos) para soluções de problemas de Pesquisa Operacional.

Outra característica que o ambiente deva suportar é a possibilidade de utilização na Web, fazendo com o aprendizado possa ser continuado pelo aluno em locais e horários diferenciados, assim o ambiente deve ser conseguir salvar as configurações existentes (atuais) para um possível carregamento futuro da situação guardada. Importante também que o ambiente seja configurável (customizado) de maneira que o acesso não fique restrito a poucas configurações de máquinas.

3.4 Considerações

Este capítulo mostrou a arquitetura do modelo de IEEE / LTSA em termos de processos, repositórios e fluxos de dados, fornecendo uma estrutura para compreender e modelar a especificação da Arquitetura Proposta baseada nos conceitos e definições presentes neste trabalho.

Por fim, foi realizado um levantamento para um domínio específico (Pesquisa Operacional), para identificação e composição de componentes/objetos educacionais, os quais servirão como base para a modelagem e desenvolvimento de ambientes educacionais direcionados para o ensino de Programação Linear. Um modelo de como identificar os componentes no modelo proposto foi feito de maneira verificar seu uso.

4 CONCLUSÃO

O trabalho foi desenvolvido por meio da investigação científica, sendo realizada uma pesquisa bibliográfica sobre o tema, bem como o levantamento de técnicas para especificação de software, que contribuem para o desenvolvimento integrado de ambientes educacionais, que foram fundamentais para o desenvolvimento deste trabalho.

Primeiramente, foram realizados estudos sobre os recursos computacionais como apoio ao ensino (LMS – *Learning Management System*), seguido de estudo de métodos para especificação de software (arquitetura). Com este embasamento teórico será realizada a identificação de objetos de aprendizagem direcionados para a Pesquisa Operacional.

Uma breve descrição sobre arquitetura de software, ressaltando alguns pontos que são necessários para a composição deste trabalho. Conceitos e definições são estabelecidos, descrevendo uma introdução sobre estilos arquiteturais.

Conceitos e definições adicionais que são fornecidos ressaltam a importância do reuso na análise, projeto e código de um artefato de software; fatores importantes no desenvolvimento de software. Também, conceitos e definições de componentes de software, fazendo uma descrição de formas de adaptação e conexão de componentes, entre outras. Este assunto fornece uma base concreta para a modelagem da arquitetura proposta.

Os objetos de aprendizagem à luz de suas características, classificações e esforços de padronização, podem ser considerados como um tipo de software educacional, os objetos de aprendizagem têm um processo de desenvolvimento que guarda uma especificidade própria. O processo remete a questões sobre como as pessoas aprendem pelo tratamento dado aos conteúdos endereçados pelos objetos de aprendizagem.

A modelagem matemática é a representação do mundo real em um ambiente controlado. Conforme a quantidade de restrições for adicionada à modelagem, maior será a exigência de

processamento, aumentando a complexidade da solução. Deve ser feita uma análise das restrições do problema para se encontrar as restrições críticas (aquelas que realmente influenciam no resultado final) para se diminuir essa complexidade.

Atualmente, a modelagem matemática pode ser estruturada através de softwares os quais possuem um poder de processamento forte o que possibilita modelar problemas de forma específica, com mais detalhes e obter respostas mais confiáveis com menor tempo possível.

O trabalho realizado procura ser um guia para a especificação e o desenvolvimento de um ambiente integrado de ensino, focando o ensino de Pesquisa Operacional (PL). O educador quando necessita de modelar e detalhar um ambiente de auxilio ao ensino muitas vezes tem certa dificuldade para identificar o que necessário para o mesmo. Assim, este trabalho procura ajudar nesta dificuldade, deixando o educador mais focado em elaborar e desenvolver o conteúdo que deseja ensinar.

Portanto, este trabalho procurou identificar alguns componentes/objetos educacionais utilizados num ambiente para ensino de Pesquisa Operacional, verificando a análise de ferramentas existentes (Leal *et al.*, 2010), em que pode-se identificar os objetivos comuns entre elas, bem como a carência de alguns recursos.

Como o levantamento teórico e o embasamento de técnicas para modelagem e especificação da arquitetura de sistemas, o trabalho pretende ser um facilitador na concepção e desenvolvimento de ambientes de ensino, assim criando mecanismos para que a criação de um ambiente possa ser realizada de maneira mais rápida, com o reuso de componentes e como um guia auxiliador. Por isto, vários assuntos foram abordados de maneira que no desenvolvimento e integração de ambientes os desenvolvedores (e implementadores) tenha como base este trabalho.

Tentando deixar mais claro o uso do modelo foi realizado um pequeno estudo de casos de maneira a verificar como a identificação de requisitos de aprendizagem e o modelo podem ser utilizados, e como pode ser realizado a interação entre os componentes.

Como trabalho futuro seria interessante que alguns ambientes já existentes fossem modelados utilizando o roteiro proposto, fazendo deles como sendo alguns estudos de casos. Outra sugestão poderia ser o desenvolvimento de um ambiente, em que fossem utilizadas as ferramentas (componentes) disponíveis para a integração em um ambiente novo de ensino, porém com ferramentas já desenvolvida e *free*. Por último, seria a integração com ferramentas existentes e o desenvolvimento e implementação de componentes que interagissem com as ferramentas disponíveis, fazendo isto utilizando o trabalho aqui proposto, bem como o embasamento cientifico deste trabalho.

REFERÊNCIAS

ARENALES, Marcos; ARMENTANO, Vinícius; MORABITO, Reinaldo; YANASSE, Horacio. Pesquisa Operacional: Para Cursos de Engenharia. 2. ed. Rio de Janeiro: Elsevier Editora Ltda, 2007. 519 p.

BITTENCOURT, Dr. Evandro. Pesquisa Operacional - Administração de Empresas. Joinville: Univille, 2003. 29 p.

BOSCH, Jan. **Adapting object-oriented components**. International Workshop on Component-oriented Programming, (WCOP), 1997.

BUSCHMANN, Frank. et al. **A System of Patterns**, Wiley, 1996.

CAMPOS, Marcelo R. **Compreensão visual de *frameworks* através da introspeção de exemplos**.Porto Alegre: UFRGS/II/CPGCC, Mar. 1997. Tese de Doutorado.

CANAL, Carlos; PIMENTEL, Ernesto; TROYA, José M. **On the composition and extension of software components**. Málaga, Spain. Disponível por WWW em: http://www.cs.iastate.edu/~leavens/FoCBS/canal.html.

CHIDAMBER, S. and KEMERER, C. (1994). **A Metrics Suite for Object Oriented Design**. IEEE Transactions on Software Engineering, 20(6). Disponível em: <http://portal.acm.org/citation.cfm?id=631131> Acesso em: 15 de maio de 2010.

COAD, Peter. **Object-Oriented Patterns**. Communications of the ACM, V.35, nº 9, p. 152-159, setembro 1992.

CUADRADO- GALLEGO,J.J. *Adaptación de las métricas de reusabilidad de la Ingeniería Del* Software *a los learning objects. RED.* ***Revista de Educación a Distancia***, número monográfico II. Maio de 2005. Disponível em: <http://www.um.es/ead/red/M4.> Acesso em: 03 de maio de 2010.

FAYAD, Mohamed; SCHMIDT, Douglas. **Object-Oriented Application Frameworks.** Communications of the ACM, New York, v. 40, n. 10, p. 32-38, Oct. 1997.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. **Design Patterns: Elements of Reusable Object-Oriented Softwar**e. Reading, MA : Addison-Wesley, 1995.

GARLAN, David; PERRY, Dewayne**. Introduction to the Special Issue on Software Architecture**. IEEE Transactions on Software Engineering, April 1995.

GARLAN, David; ALLEN, Robert. **A Formal Basis for Architectural Connection. ACM Transactions on Software Engineering and Methodology**. v.6, n.3, p.213-249, July 1997.

GARLAN, David; SHAW, Mary; WING, Jeannette; **Composable Software Systems**. **Software Engineering Note**s. v.22, n.5, p.111-114, Sep1997.

HELTON, David. **The Impact of Large-Scale Component and Framework Application Development on Business**s. Proceedings...WCOP98. Disponível por WWW em: http://www.abo.fi/ ~Wolfgang.Weck/WCOP/98/Papers/. (28/11/1998).

IEEE P1484.1/D9 – Institute of Electrical and Electronics Engineers - Learning Technology Standards Committee, IEEE Computer Society**. Draft Standard for Learning Technology—Learning Technology Systems Architecture (LTSA)**. IEEE P1484.1/D9, 2001-11-30.

IEEE Learning Technology Standards Committee (IEEE/LTSC). '*IEEE Standard for earning Object Metadata'*. Disponível em: <http://ltsc.ieee.org/wg12/> Acesso em: 11 de marco de 2010.

JAVA World. ***Make Java Fast***. Disponível em: <http://www.javaworld.com> Acesso em: 19 de junho de 2010.

JOHNSON, Ralph E.; Foote B. **Designing Reusable Classes**. Journal of Object Oriented Programming – JOOP, 1(2):22-35, Junho/Julho 1988.

LEAL, G. C. L.; SAMED, M. . A; FENERICH, F.C. **Análise de Ferramentas para Resolução de Problemas de Otimização Linear**, Departamento de Engenharia de Produção, Universidade Estadual de Maringá, 2010.

MENDES, Antonio. **Arquitetura de Software**. Editora Campus, 2002.

MENDES, R. M.; SOUZA, V.I.; CAREGNATO, S. E. **A propriedade intelectual na elaboração de objetos de aprendizagem.** Disponível em: http://www.cinform.ufba.br/v_anais/artigos/rozimaramendes.html> Acesso em: 11 de julho de 2010.

ÓLAFSSON, Asgeir.; DOUG, Bryan. **On the need for "required interfaces" of components**. In Special Issues in Object-Oriented Programming, Workshop of the ECOOP, 1996.

PFISTER, Cuno. **A Case Study using BlackBox Components**.Disponível por WWW em: http://www.oberon.ch/blackbox.html (1998).Acesso em: 15 junho de 2010.

PRESSMAN, R. S. **Software *engineering***: *a practitioner's approach*, 3. ed. New York: Ed. McGraw-Hill, 1995.

RIBEIRO, Antônio Mendes; COELHO, Maria L. **O Uso das Novas Tecnologias e as Formas de Aprendizagem : Análise de uma Experiência. DCC/UFMG, 2004.**

PUCCINI, Abelardo de Lima. Introdução à Programação Linear. Rio de Janeiro: Livros Técnicos e Científicos Editora S.A., 1972. 252 p.

ROCHA, A.R.C., MALDONADO,J.C.; WEBER,K.C., **Qualidade de Software: Teoria e Prática**. São Paulo: Ed. Prentice Hall, 2001.

SCHEER, S.; GAMA, C.L.G. Developing learning objects for a structural engineering educational network. In: INTERNATIONAL CONFERENCE ON COMPUTING IN CIVIL AND BUILDING ENGINEERING, X ICCCBE., 2004, Weimar. **Anais…** Weimar. 2004. X-Proceedings. Weimar : Bauhaus-Universität Weimar e ICCCBE, v. 1. p. 1-10.

SHAW, Mary; GARLAN, David. **Software Architecture – perspective on an emerging discipline**. Upper Saddle River: Prentice Hall, 1996.

SZYPERSKI, Clemens. *et. al.* **Sumary of the Second International WorkShop on Component-Oriented Programing** In: INTERNATIONAL WORKSHOP ON COMPONED-ORIENTED PROGRAMING, (WCOP), 1., 1996, Linz, Austria. Proceedings Linz: [s.n.], 1996.

SZYPERSKI, Clemens. *et. al.* **Sumary of the First International WorkShop on Component-Oriented Programing** In: INTERNATIONAL WORKSHOP ON COMPONED-ORIENTED PROGRAMING, (WCOP), 2., 1997, Jyaskyla, Finland. Proceedings Jyaskyla.: [s.n.], 1997.

SZYPERSKI, Clemens. **Component Software Beyond Object-Oriented Programmin**g. Editora Addison-Wesley & ACM Press, Segunda Edição, New York, 2002.

SICILIA, M.A. Reusabilidad y reutilización de objetos didácticos: mitos, realidades y posibilidaddes. RED. **Revista de Educación a Distancia,** número monográfico II. Fevereiro de 2005. Disponível em: <http//www.um.es/ead/red/M2> Acesso em: 03 de maio de 2010.

VASCONCELOS, José. R.; RICARTE, Ivan. L. M.; MARCHI, Gecén. D.; GATTO, Rafael A. **Uma abordagem de Arquitetura Estilizada para Software Educacional.** X Congreso Argentino de Ciências de la Computación. III Workshop de Tecnologia Informática Aplicada en Educación, p. 1032-1043, outubro, 2004.

VOAS, Jeffrey. **Maintaining Component-Based Systems. IEEE Software.** [s.l.], p. 22-27, July/Aug 1998.

YNIMINE, S. **Flash MX**. Florianópolis: Ed. Visual Books, 2002.

World Wide WEB Consortium – (W3C). **Extensible Markup Language** *(XML)*.Disponível em: <http://www.w3.org/xml/> . Acesso em: 20 de setembro de 2010.

WILLS, Alan C. **Component Based Development: Principles and Prerequisite**s. Disponível por WWW em: http://www.trireme.com/catalysis/cbd97ppt.pdf. (09/09/1998).

WOLFGANG, Weck; BOSCH, Jan; SZYPERSKI, Clemens. **Summary of the Second International Workshop on Component-Oriented Programing(WCOP'97**). Disponível por WWW em: http://www.abo.fi/~Wolfgang.Weck/WCOP/97/ Summary.html. 09 de junho de 1997 (09/09/1998).

Universidade Estadual de Maringá
Departamento de Engenharia de Produção
Av. Colombo 5790, Maringá-PR CEP 87020-900
Tel: (044) 3011-4196/3011-5833 Fax: (044) 3011-4196