ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

ALECSANDRO BATISTA DE ALMEIDA

LUCAS ANDRADE PIMENTA DE FIGUEIREDO

Automação de bibliotecas empregando tecnologia RFID

São Paulo

2005

ALECSANDRO BATISTA DE ALMEIDA

LUCAS ANDRADE PIMENTA DE FIGUEIREDO

Nota final
8,8 (oito e vinte)

Automação de bibliotecas empregando tecnologia RFID

Trabalho de Formatura apresentado a Escola Politécnica da Universidade de São Paulo para obtenção de graduação em engenharia.

Área de concentração: Automação de Sistemas
Orientador: Prof. Dr. José Reinaldo Silva

São Paulo

2005

DEDICATÓRIA

A nossas famílias, com admiração e gratidão, por sua compreensão, presença e apoio ao longo do período de elaboração deste trabalho.

RESUMO

O presente trabalho tem por objetivo analisar e desenvolver uma aplicação da tecnologia RFID para otimização dos processos de empréstimo e devolução de livros em bibliotecas. RFID é uma tecnologia de identificação automática e captura de dados através de sinais de radio freqüência, onde um objeto recebe uma tag ou transponder que contém uma determinada informação que pode ser capturada pelo sistema através de leitores. Dessa forma é possível processar essa informação computacionalmente, armazenando a ação de empréstimo e devolução de livros em bancos de dados. O sistema é composto por um módulo, denominado middleware, responsável por receber, filtrar e enviar os dados provenientes dos leitores para bancos de dados, realizando dessa forma a integração dos hardwares envolvidos. A análise do processo, da performance e operabilidade do sistema é feita através de modelos em redes de Petri, possibilitando o confronto com tecnologias já empregas na solução de automação em bibliotecas.

ABSTRACT

The purpose of this work is to analyze and develop an application using the RFID technology for the optimization of all the processes concerning lending and returning books at libraries. RFID is a technology of automatic identification and capture of data through radio frequency signals, where an object receives a tag or transponder which has some information that can be captured by the system through readers. Therefore it is possible to process this information on a computer, storing the action of lending and returning on databases. The system is composed by a middleware, which is responsible for receiving, filtering and sending the data from the readers to higher level systems, enabling the integration of the involved hardwares. The analysis of the process performance and operation is made through models developed on Petri Nets, making it possible to compare with other technologies already in use at libraries.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

Antes de apresentarmos a tecnologia de identificação por rádio freqüência, RFID (Radio Frequency Identification), convém definirmos o termo auto-id. Auto-id (Automatic Identification) é o termo dado para a tecnologia de identificação automática de objetos. Empresas querem identificar itens, capturar informações sobre eles e, de alguma forma, armazenar essas informações em um computador.

Seguindo a tendência de automatização de praticamente todos os processos em uma empresa, a identificação automática de produtos já é uma realidade em todo o mundo. Os objetivos de todos os sistemas auto-id são: aumentar eficiência, reduzir erros de entrada de dados e reduzir o trabalho operacional que é requerido por um sistema não automatizado. É importante salientar que quando estamos falando de auto-id não estamos necessariamente nos referindo ao RFID. Existem diversas tecnologias para identificação automatizada. Dentre elas: código de barras, smart cards, reconhecimento de voz, tecnologias biométricas, reconhecimento óptico de caracteres, entre outras.

RFID é um termo genérico para tecnologias que usam ondas de rádio para identificar automaticamente objetos e existem diversos métodos para identificação de objetos utilizando RFID. O método mais empregado, e objeto desse trabalho, consiste na gravação em um microchip de um número serial que identifica unicamente um objeto. Uma antena é acoplada ao microchip (este conjunto é denominado transponder ou tag). A antena permite que o chip transmita o número de identificação para um leitor. O leitor, através de um conversor analógico digital, converte o sinal de rádio para ser transferido para um computador.

O código de barras tem sido o principal método de identificação automática nos últimos 25 anos. Devido ao seu baixo custo, facilidade de implantação e uso em larga escala no mundo todo, o código de barras ainda é um método com enorme potencial. No entanto, o

método de reconhecimento por código de barras exige que a etiqueta a ser reconhecida tenha contato visual com o leitor. Para aplicações como identificação do número de uma conta a pagar isso não chega a ser um problema. Já para casos em que se necessita agilidade e velocidade de reconhecimento, como um processo de manufatura automatizado, o uso do código de barras pode não ser a melhor solução. A velocidade de reconhecimento e o fato de não ser necessário que o objeto esteja em uma posição determinada para seu reconhecimento, são vantagens imediatas do uso do RFID, entre outras.

O uso da tecnologia RFID desencadeia uma rede que conecta objetos a computadores. Assim como a Internet, o objetivo é construir uma rede de proporções mundiais que conecte os mais diversos tipos de itens nas mais diversas cadeias de produção. Essa rede é conhecida como rede EPC (Eletronic Product Code). As vantagens de se interligar todos os produtos em uma cadeia de produção são inúmeras. A tecnologia RFID pode aumentar a capacidade de planejamento, reduzir perda e roubo de material, além de tornar o processo de produção mais eficiente e adaptável enquanto se garante a sua qualidade. Também é possível gerenciar estoque em tempo real, otimizando o processo de entrada e saída de materiais.

Nos primeiros capítulos deste trabalho faremos uma abordagem geral da tecnologia RFID, suas características técnicas, equipamentos e comparações com outras tecnologias. Em seguida apresentaremos uma aplicação desta tecnologia nos centros de distribuição, vislumbrando um futuro próximo. Por último será apresentada uma aplicação para automação de bibliotecas, que é o objetivo do presente trabalho.

2 TECNOLOGIA RFID

RFID é uma tecnologia que vem sendo usada há muitos anos na forma de transponders ativos para monitorar veículos, automatizar o pagamento de pedágios, controle de acesso em condomínios e muitas outras aplicações. Como serão detalhadas adiante, as tags ou etiquetas ativas utilizam uma fonte de energia para comunicar-se com o leitor. Com isso o sinal ganha potência, distância de leitura, confiabilidade e precisão, mas perde-se no tamanho, peso e mobilidade das etiquetas, já que estas devem estar ligadas a uma fonte de energia. Com o surgimento das etiquetas passivas as possibilidades de aplicação aumentaram significativamente. Nesse capítulo, assim como nesse trabalho, enfatizaremos a tecnologia das etiquetas passivas.

2.1 HISTÓRIA DO RFID

A tecnologia RFID desenvolveu-se juntamente com o desenvolvimento do radar, já que é baseada na combinação de transmissões de radio e radar. Os anos 50 foi a era de explosão das técnicas de RFID devido ao desenvolvimento do radio e radar, nos anos 30 e 40. As primeiras atividades comerciais iniciaram nos anos 60, com a fundação das companhias Sensormatic e Checkpoint. Essas companhias e outras como Knogo desenvolveram equipamentos de segurança com sistemas que usavam 1 bit para detectar a presença ou ausência de uma tag, usando tanto micro ondas como tecnologia indutiva. Esse foi o primeiro e maior uso do RFID.

Nos anos 70, desenvolvedores, inventores, companhias, instituições acadêmicas e mesmo laboratórios do governo dos Estados Unidos, trabalharam muito em RFID e realizaram avanços notáveis embora ainda não estivesse comercialmente disponível para

aplicações em transportes. Novas companhias surgiram, como a Identronix, Amtech, pois o potencial do RFID tornara-se óbvio.

Os anos 80 tornaram-se a década da total implementação da tecnologia RFID, com aplicações em varias partes do mundo. Os Estados Unidos tinha grande interesse em desenvolvimentos para transportes, controle de acesso de pessoas e controle de animais. Na Europa tinha-se grande interesse em sistemas para animais, para industrias, aplicações em negócios e pedágios em auto estradas.

Os anos 90 foram marcados pelo desenvolvimento do RFID em larga escala em sistemas de cobrança eletrônica. O primeiro sistema de cobrança eletrônica em auto-estradas foi instalado em Oklahoma em 1991, onde veículos podiam passar nos pontos de cobrança em altas velocidades. Tanto a tecnologia de micro ondas quanto a indutiva foi usada em sistemas de cobrança, controle de acesso e em inúmeras aplicações comerciais. Essas aplicações espalharam-se rapidamente para outros paises.

Com o crescente interesse em RFID em gerenciamento e a oportunidade do RFID substituir o já consagrado código de barras, muitas empresas foram entrando nesse emergente mercado. Embora o RFID já tenha se tornado parte da vida cotidiana, grandes avanços ainda são necessários, como por exemplo a nova alocação de espectro na banda de 5,9 GHz proposta pela Comissão Federal de Comunicação, que amplia a gama de aplicações embora os requisitos de equipamentos ainda não são atendidos.

2.2 FUNDAMENTOS TEÓRICOS

A figura 1 mostra o processo de comunicação entre o leitor e as etiquetas. Como se pode observar, o leitor transmite um sinal de interrogação para a etiqueta e ao mesmo tempo a etiqueta envia um sinal de resposta ao leitor.

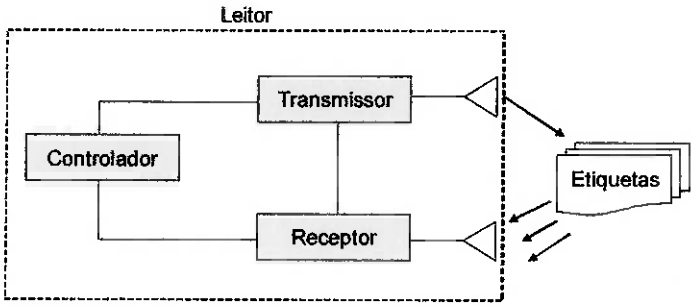

Figura 1 - Leitor e Etiquetas

No caso das etiquetas passivas, ocorre uma conversão da energia do sinal de interrogação para energia elétrica para gerar o sinal de resposta e, portanto, a resposta destas etiquetas é muito fraca. Vale lembrar que o leitor deve detectar e distinguir o sinal de resposta de cada etiqueta na presença de diversos outros ruídos e sinais que existem no ambiente. Geralmente, o leitor utiliza a mesma antena para enviar o sinal de interrogação e receber a resposta. Logo, o circuito do leitor normalmente inclui uma forma de distinguir sinais de interrogação e de resposta.

2.2.1 AS LEIS DE MAXWELL

As leis de Maxwell, no vácuo, são:

$$\oint E \, dA = \frac{Q}{\varepsilon_0} \tag{1}$$

$$\oint B \cdot dA = 0 \tag{2}$$

$$\oint E \cdot ds = -\frac{d\Phi_m}{dt} \tag{3}$$

$$\oint B \cdot ds = \mu_0 I + \mu_0 \varepsilon_0 \frac{d\Phi_e}{dt} \tag{4}$$

Lei da Gauss, equação 1, que afirma que o fluxo elétrico total que atravessa qualquer superfície fechada é igual à carga líquida que existe no interior da superfície, dividida por ε_0. Essa lei relaciona o campo elétrico à distribuição de carga, pois as linhas do campo elétrico iniciam nas cargas positivas e terminam nas cargas negativas.

A Lei de Gauss do Magnetismo, equação 2, diz que o fluxo magnético líquido através de qualquer superfície fechada é igual a zero, isto é, o número de linhas do campo magnético que entram em um volume fechado é igual ao número de linhas que saem desse volume. Isto quer dizer que es linhas de um campo magnético não podem principiar ou acabar em qualquer ponto.

Lei de Faraday da Indução, equação 3, descreve a relação entre um campo elétrico e um fluxo magnético variável. Essa lei afirma que a integral de linha de um campo elétrico, sobre qualquer curva fechada (que é igual à força eletro-motriz), é igual à taxa temporal da variação do fluxo magnético que atravessa a área de qualquer superfície limitada pela curva fechada. Uma conseqüência da lei de Faraday é a corrente induzida numa espira condutora colocada num campo magnético variável no tempo.

Lei de Ampère generalizada, equação 4, descreve uma relação entre campos magnéticos, campos elétricos e correntes, isto é, a integral de linha do campo magnético sobre qualquer curva fechada, é determinada pela soma da corrente de condução líquida que atravessa qualquer superfície limitada pela curva fechada com a taxa temporal de variação do fluxo elétrico que atravessa a mesma superfície limitada pela curva.

Uma maneira mais simples de entender a natureza dos campos eletromagnéticos é através da interpretação dos campos de origem e de vórtice dada por Helmholtz. Utilizando figuras de campo da Michael Faraday, podemos representar os campos de origem e vórtice como na figura 2.

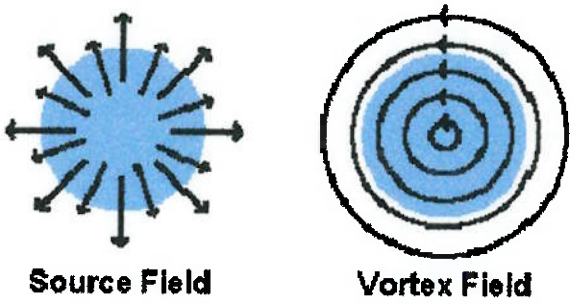

Figura 2 - Campos de origem e de vórtice

Helmholtz mostrou que campos vetoriais devem ser interpretados como a superposição de dois tipos de campos básicos, conhecidos como campos de origem e vórtice.

Na ilustração do campo de origem, vemos linhas de campo elétrico originando em uma região de carga positiva. Se essas linhas terminam em algum lugar, será em uma região de carga negativa. Essas linhas de campo nunca se interceptam. Na ilustração de campo de vórtices, nós vemos linhas de campo magnético ao redor de um fio onde circula corrente. Essas linhas estão sempre na forma de curvas fechadas e nunca têm pontos de origem e destino.

Sabemos, através das leis de Maxwell, que campos elétricos E podem ser do tipo de origem, de vórtices ou uma mistura dos dois. Cargas em condutores geram campos elétricos do tipo origem, enquanto densidade de fluxo magnético variável no tempo gera campos elétricos do tipo vórtice. Campos magnéticos são unicamente do tipo vórtice. Quando se tem corrente elétrica ou densidade de fluxo elétrico variável no tempo, surgem campos magnéticos.

Quando um campo eletromagnético se propaga por uma distância significativa da antena que o originou, o campo elétrico variável gera vórtices de campo magnético, e o campo magnético variável gera vórtices de campo elétrico. Isto faz com que a onda tenha um grande alcance.

2.2.2 PROPRIEDADES DOS CAMPOS ELETROMAGNÉTICOS

Considerando campos eletromagnéticos gerados por uma antena transmissora, é usual definir o conceito de campos próximos, que é o campo a uma distância de até, no máximo, $\lambda/2\pi$ m da antena e campos distantes que são considerados para distâncias maiores do que $\lambda/2\pi$ m.

O campo próximo é basicamente um campo de armazenamento de energia. Os campos nessa região são campos reativos e quase estáticos. Já o campo distante é um campo de propagação de energia. Em qualquer ponto do espaço existe um composto entre os dois tipos de campo, mas dependendo da distância da antena, um campo tende a ser dominante em relação ao outro. A diferença entre os campos podem ser melhor explicadas com um exemplo numérico: em um sinal de 13,56 MHz e λ=22 m a distância limite é dada por $\lambda/2\pi$=3,5 m ou em um sinal de 915 MHz e λ=328 mm a distância limite é dada por $\lambda/2\pi$=52 mm. Conclui-se que sistemas HF (High Frequency) operam nos campos próximos, enquanto sistemas UHF (Ultra High Frequency) operam nos campos distantes.

No campo próximo, o campo elétrico e o campo magnético têm um grau de independência. É possível desenvolver antenas que geram, no campo próximo, campos elétricos grandes e campos magnéticos pequenos e antenas diferentes que geram campos magnéticos grandes e campos elétricos pequenos. A propagação de um campo elétrico E é facilmente interrompida por muitos materiais, por isso os sistemas HF procuram gerar mais campo magnético. Como conseqüência, as antenas das etiquetas para esses sistemas são projetadas para detectar campo magnético. Essas antenas normalmente têm a forma de uma espiral plana, pois através da Lei da Indução de Faraday, corrente elétrica é gerada quando esta espira está inserida em um campo magnético variável no tempo. Dessa forma, a tag recebe a maior quantidade de energia quando esta está ortogonal ao vetor campo magnético.

Este tipo de tag é conhecida como indutiva. Para interação com campos predominantemente elétricos, são utilizadas etiquetas capacitivas. Nesse trabalho daremos ênfase nas etiquetas indutivas, que são mais utilizadas.

2.2.3 ACOPLAMENTO INDUTIVO

Nessa topologia, um campo magnético de alta intensidade é gerado pelo circuito ressonante de saída, o leitor. Uma parcela desse campo magnético penetra no indutor (antena) do transponder. A tensão gerada no indutor do transponder é retificada e é utilizada para carregar um capacitor que auxiliará na alimentação dos circuitos internos. O capacitor e o indutor do transponder formam um circuito ressonante paralelo, cuja freqüência de ressonância corresponde à freqüência emitida pelo leitor. Dessa forma, a tensão no indutor do transponder será máxima em sua freqüência de ressonância.

A modulação com carga utiliza o transistor do tipo T para adicionar carga resistiva paralelamente ao circuito ressonador do transponder, sincronizando com os dados a serem transmitidos ao leitor. O chaveamento (acoplar e desacoplar a carga resistiva) afeta a tensão no indutor do elemento leitor, ocorrendo dessa maneira uma modulação em amplitude da freqüência fundamental de operação. Desta forma, o sinal de retorno no leitor é amplificado e demodulado para recuperação dos dados recebidos. A diferença de potencial (amplitude) gerada pela modulação por carga no elemento leitor pode ser algo da ordem de 10 mV, contra 100V de excitação (freqüência fundamental). Isto determina uma não desejada reação sinal-ruído, exigindo uma certa sofisticação do circuito demodulador.

2.2.4 TRANSFERÊNCIA DE DADOS

No uso de topologias de sistemas RFID, três tipos de modulação podem ser aplicadas: modulação em freqüência, modulação em amplitude ou modulação em fase que alteram as três principais características de uma onda eletromagnética. Devido à simplicidade de implementação e demodulação, a técnica mais utilizada é a modulação em amplitude.

A variação de amplitude está relacionada aos dois estados distintos, provenientes de um sinal digital binário. O fator de modulação m indica a porcentagem de redução da amplitude da portadora. As linhas harmônicas geradas por esse tipo de modulação são dependentes do fator de modulação m e da relação t/T.

2.3 COMPONENTES DO SISTEMA

Existem diferentes sistemas RFID desenvolvidos, sendo todos eles baseados em leitores e tags. Os leitores capturam a informação armazenada nas tags e são afetados por ambientes particulares enquanto as tags são afetadas pelos objetos no ambiente. A operabilidade entre leitores e ambiente e entre tags e objetos definem as especificações de projeto. A classificação de sistemas RFID é baseada nessas especificações.

Uma primeira classificação, referente a aspectos construtivos das tags, são tags do tipo chip e chipless. Tags do tipo chip são construídas com circuito integrado, enquanto tags chipless não. Outra classificação, referente ao grupo das tags com chip, são as tags passivas, semipassivas e ativas. Tags passivas não possuem fonte de energia interna, nem transmissores ativos, tags semipassiva possuem fonte de energia interna, mas não possuem transmissores ativos e tags ativas possuem tanto fonte de energia interna, quanto transmissores ativos. Ainda

uma outra classificação usada divide tags em read-only e read-write. Tags read-only podem ser lidas varias vezes, mas as informações nela contida podem ser gravadas apenas uma vez e tags read-write podem ser lidas varias vezes e permitem gravar e regravar informações varias vezes.

As tags chipless, embora possuam um custo de produção mais baixo em relação a tags com chip, permitem armazenar no máximo 24 bits. Sendo assim as tags do tipo chip possuem uma grande vantagem em sistemas que pretendam identificar unicamente todos os itens, pois permitem armazenar 96 bits ou mais. Na situação onde é necessário ler múltiplas tags no campo de leitura, somente usando tags com chip é possível conceber sistemas anti-colisão, dado sua alta capacidade de memória.

As tags ativas oferecem a melhor performance em relação às outras, pois podem ser lidas a quilômetros de distância, além de possuírem rápida comunicação e confiabilidade, porem a um alto custo. As tags semipassivas podem ser lidas a medias distâncias, por volta de dez metros, e possuem custos menores em relação as tags ativas. As tags passivas possuem os menores custos de produção porem só podem ser lidas a pequenas distâncias, bem inferiores a dez metros, e são sensivelmente afetadas pelo ambiente e os objetos nele presente.

Tags com chip podem ser read-only ou read-write. Tipicamente somente as tags passivas são read-only e tem um código de identificação gravado no momento da manufatura, possuindo uma memória tipo ROM (Read Only Memory) ou quando alocado em um objeto, possuindo memória tipo WORM (Write Only Read Many). As tags read-write possuem uma variedade de informações que podem ser gravadas e, por isso, oferece funcionalidades adicionais, porem a um alto custo. Por outro lado, as tag read-only se forem implementadas em um sistema de distribuição com um banco de dados bem projetado podem oferecer as mesmas funcionalidades que as tags read-write a custo bem inferior.

Os leitores são dispositivos simples que buscam tags no ambiente. Podem enviar ou receber informações das tags. Utilizam um sistema de codificação e decodificação que converte as ondas de rádio em uma forma que pode ser interpretada pelo computador. Os custos estão atrelados ao nível de funcionalidades que oferece, por exemplo, ler múltiplos itens ou efetuar leituras mesmo em movimento de alta velocidade da tag.

Os softwares são uma importante parte do sistema RFID, responsáveis pela usabilidade das informações lidas pelo leitor para integração da tecnologia RFID com qualquer outro sistema em cadeias de suprimentos, como WMS (Warehouse Management Systems), TMS (Transportation Management Systems), SEM (Event Management Systems), OMS (Order Management Systems), ERP (Enterprise Resource Planning), entre outros.

3 CUSTO BENEFÍCIO: RFID *x* CÓDIGO DE BARRAS

Um ponto importante a ser considerado na implementação da tecnologia RFID é um estudo de comparação com uma tecnologia largamente empregada denominada código de barras. Bem desenvolvida e há anos empregada comercialmente, com introdução no mercado brasileiro por volta dos anos 90, o código de barras é um símbolo composto de barras paralelas de larguras e espaçamentos variados com a finalidade de representar uma numeração, que viabiliza a captura automática dos dados por meio de leitura óptica nas operações automatizadas. Existem vários padrões de código de barras, como o código 39, o código ITF e o código IUPC e EAN. As etiquetas possuem um baixo custo e o processo de leitura é profundamente afetado pela qualidade da impressão e pela linha de visão do leitor. Possui baixo limite de armazenamento de dados, além de esses dados serem estáticos.

As tags do RFID não necessitam de uma linha de visão, possuindo um grande campo de leitura. Além de permitir mudanças nos dados armazenados, oferece mais informação sobre o produto ou conteúdo de um pacote e, em sua maior parte, não é afetada por condições físicas adversas de ambientes.

Entretanto, atualmente o RFID possui um custo mais alto do que a tecnologia de código de barras. É importante ressaltar que o RFID não precisa substituir totalmente o código de barras. O gráfico na figura 3 apresenta o custo versus performance da tecnologia RFID.

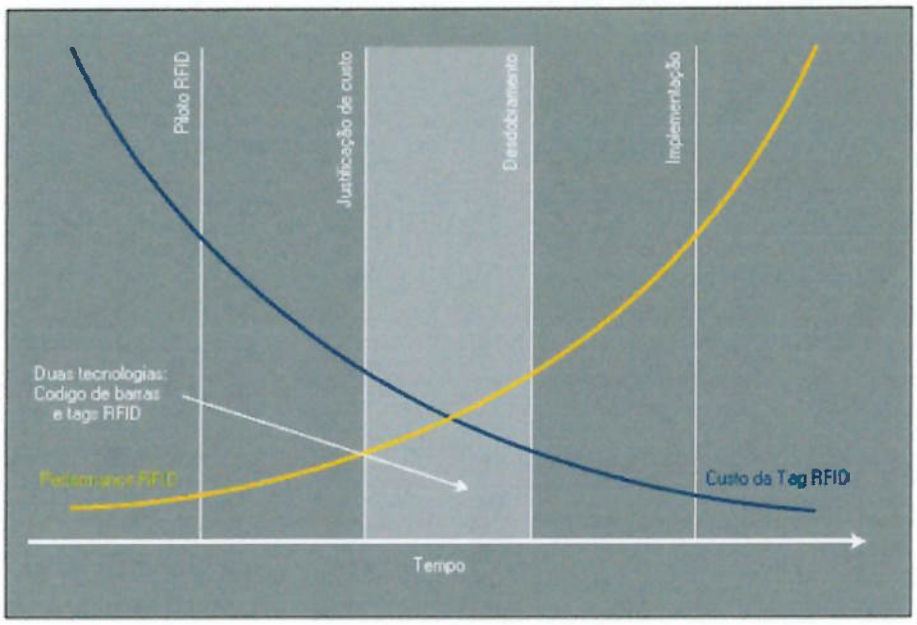

Figura 3 - Custo versus performance da tecnologia RFID

Essa curva ilustra uma típica curva de demanda muito conhecida em economia, onde identificamos um ponto no qual a performance supera o custo da tag. Nesse ponto a adoção do RFID começa a se justificar. Embora em armazéns e ambientes de distribuição ainda priorizem o custo da tag, o enfoque na relação entre custo e benefício é mais importante. Por exemplo, em muitas aplicações em manufatura, uma tag por dez dólares é insignificante quando comparado com custo do produto.

Essa tendência em priorizar o custo, restringe as aplicações da tecnologia RFID, com empresas justificando a viabilidade de implantação somente com tags por alguns centavos de dólar. Convém lembrar que essa situação não é diferente dos estágios iniciais da adoção do código de barras e conseqüentemente, com maior adoção do RFID o custo das tags irá diminuir, viabilizando a implantação em um número maior de aplicações.

4 REDE EPC

Há alguns anos atrás, uma industria de bens de consumo criou o UPC (Universal Product Code), permitindo que produtores e varejistas, entre outros, identifiquem automaticamente um grupo de produtos do mesmo tipo, ao longo da cadeia de suprimentos. Anos depois, essa industria desenvolveu o EPC (Eletronic Product Code), que é essencialmente uma segunda geração do UPC, usado em código de barras.

A diferença principal é que o EPC permite identificar unicamente cada item, mesmo que vários itens sejam exatamente idênticos, ou seja, um tipo de produto com mesmo UPC. Além disso, também permite identificar produtos não consumíveis, como pallets, caixas ou caminhões, que não possuem UPC.

Por uma questão de padronização, empresas como o Wal-Mart, entre outros varejistas, definiram um EPC com 96 bits hexadecimal que contém 24 caracteres, possuindo valores que vão de 0 a 9 e A a F. Esse conjunto de caracteres permite um número muito grande de combinações, em torno de centenas de bilhões, para cada companhia. Essa quantia se justifica se considerarmos o objetivo de identificar unicamente produtos, caixas, pallets, caminhões, containers, carros, gado ou qualquer outro objeto.

A idéia principal nessa padronização do EPC é que fornecedores podem, por exemplo, enviar para varejistas somente tags RFID com um número EPC e dessa forma os varejistas ou qualquer outro na cadeia de suprimentos podem usar esse EPC para acessar detalhes sobre o produto, incluindo da produção ou do pedido.

4.1 SAVANT

Para gerenciar os dados fornecidos pelo EPC, foi criada uma estrutura chamada Savant. Esse sistema consiste em uma estrutura de servidores distribuídos geograficamente, segundo uma hierarquia, localizados em armazéns, centros de distribuição locais ou regionais, e também em centros de dados.

O Savant é um encaminhador de dados que executa operações de captura de dados, monitoramento de dados e transmissão de dados. É constituído por três módulos: EMS (Event Management System), RIED (Realtime In-memory Event Database) e o TMS (Task Management System). Os Savants também são organizados hierarquicamente em estruturas de árvore, constituída de nós internos ou Internal Savants (IS) e nós de extremidade ou Edge Savants (ES), que simplificam a administração e melhora a performance do sistema.

O ES é um Savant que coleta em tempo real os dados do EPC, ou seja, todo o fluxo de dados entra no sistema através dos ES's. Tipicamente são alocados em lojas, armazéns, plantas de manufatura ou mesmo caminhões e continuamente capturam, monitoram e armazenam dados para reavê-los posteriormente. Este ES está diretamente conectado com os leitores de RFID. Os IS's coletam dados dos ES's, sendo seus descendentes e tipicamente estão alocados em uma região ou centro de dados e armazenam e processam os dados do EPC. A figura 4 ilustra uma árvore exemplo.

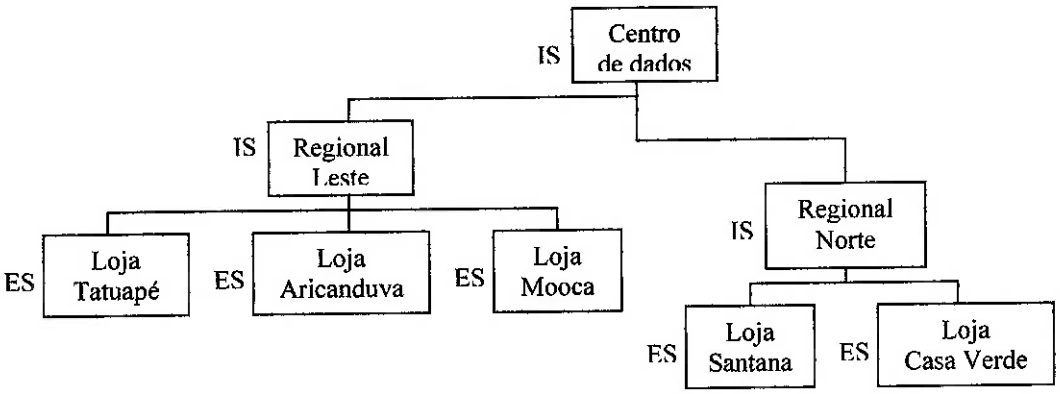

Figura 4 - Árvore exemplo

O EMS é implementado nos ES's para coletar eventos de leitura de tags. É responsável pelos adaptadores para serem lidos por vários tipos de leitores, coleta dos dados do EPC fornecidos pelos leitores em formato padrão, permite filtros de dados, suporta acessos de redes (HTTP, JMS, SOAP) para transmitir dados EPC para servidores remotos e armazena eventos para permitir acessos.

O RIED é um banco de dados que pode ser usado para armazenar informações de eventos pelos ES's, que mantém e organiza eventos enviados pelos leitores. O EMS oferece a estrutura para filtros e eventos de acesso. Um grande problema de bancos de dados é que não suportam mais que algumas centenas de transações por segundo, entretanto o RIED fornece a mesma interface que um banco de dados, mas com uma performance muito melhor.

Os softwares no Savant realizam o gerenciamento e monitoramento de dados usando tarefas customizadas. Uma tarefa pode ser comparada a um processo em um sistema multi-tarefas. O TMS do Savant gerencia tarefas, da mesma forma que um sistema operacional gerencia processos, mas com alguns diferenciais como: interface externa para alocar tarefas, plataforma independente (Java) com bibliotecas uniformes, um sistema robusto de alocação de tarefas e capacidade de reiniciar tarefas em uma parada do Savant.

4.2 ONS

Para concretizar a visão de uma rede global de informações de produtos, é necessária uma arquitetura de rede especial. Ao passo que o EPC está armazenado em uma tag, os computadores necessitam de uma maneira de ligar esse EPC a informações sobre o produto associado. Essa é a função do ONS (Object Name Service), um serviço de rede automático similar ao DNS (Domain Name Service), que aponta computadores para sites na web.

Quando uma tag de RFID é lida por um leitor, o EPC é passado para o Savant. O Savant pode acessar uma ONS em rede local ou a Internet para achar onde a informação sobre o produto esta armazenada. O ONS aponta o Savant para o servidor onde um arquivo sobre o produto esta armazenado e dessa forma esse arquivo pode ser disponibilizado para o Savant, fornecendo a informação para uma empresa ou para aplicações em cadeia de suprimentos.

Um dos requisitos do ONS é que ira receber muito mais pesquisas do que o DNS. Portanto, é interessante que as companhias mantenham um ONS local, onde a informação é armazenada para rápido acesso. Dessa forma um computador em um manufatura armazena dados ONS dos fornecedores em sua própria rede, ao invés de procurar essa informação na Internet a cada recebimento na planta de montagem. O sistema também necessita de redundâncias, como por exemplo, no caso de um servidor de uma informação fora de serviço é preciso que o ONS seja capaz de apontar o Savant para outro servidor que tenha a mesma informação armazenada.

4.3 PML

O EPC identifica unicamente um produto, mas toda a informação funcional sobre um produto é escrita em uma linguagem padrão chamada PML (Physical Markup Language). Essa linguagem é baseada em XML (Extensible Markup Language) e por causa da necessidade de padronização universal para descrever todos os objetos físicos, processos e ambientes, o PML deverá suprir todos os segmentos industriais. Esse objetivo será alcançado com a introdução de uma linguagem simples e incentivo a adoção, como no caso do HTML (Hyper Text Markup Language), linguagem básica da web, que tem se tornado mais sofisticada desde sua introdução.

O PML introduz um método comum para descrever objetos físicos e é amplamente hierarquizado. Por exemplo, uma lata de refrigerante pode ser descrita como bebida carbonada, que deve estar na subcategoria de bebidas, que deve estar na categoria mais abrangente alimento. Nem todas as classificações são tão simples, por isso para aumentar a adoção do PML, padrões como o SI (Sistema Internacional de Pesos e Medidas) ou estudos do Instituto Nacional de Padrões e Tecnologia dos Estados Unidos são empregados.

Na sua concepção, o PML possui dois tipos de dados. Um tipo chamado de dados dinâmicos que se alteram constantemente como, por exemplo, temperatura no transporte, níveis de vibrações de uma máquina. Outro tipo de dados é chamado dados temporais, que não se altera constantemente, como o material que compõem o produto ou sua localização.

A aplicabilidade dessas informações disponíveis em arquivos PML é imensa, onde companhias podem ser capazes de usá-las de diferentes formas, por exemplo, iniciar uma promoção de um produto quando a data de validade esta próxima ou um cliente pode acompanhar a evolução da temperatura de um produto, acordada por um fornecedor, durante o transporte.

A figura 5 mostra um exemplo de uma aplicação em uma concessionária de veículos. Um detalhe interessante é que mesmo depois que o produto é vendido, componentes e história do material pode ser disponibilizado para pessoas autorizadas.

```
<pml><part label="car">
<system>VIN</system>
<epc>01.10A118.985BE1.0001984519
<code>JH4NA1152MT019519
</code>
</class>
<access>
<role>Maker</role>
<part label="engine">
<epc>01.58191.39485.000098EFB1
<part label="actuator unit">
```

Figura 5 - Exemplo de PML

4.4 ESTUDO DE CASO: CENTRO DE DISTRIBUIÇÃO

A tecnologia RFID é a maior promessa para melhorar os processos em um centro de distribuição. Utilizando código de barras, um funcionário deve fazer a leitura de cada produto ou pallet antes de realizar um movimento de entrada ou saída. Empregando RFID é possível aumentar significativamente a velocidade de entrada e saída de suprimentos no armazém. O leitor consegue ler quase que instantaneamente uma carga enquanto passa pelas docas de entrada do armazém. Além disso, não é necessário checar se existem erros como produtos a mais ou a menos em um determinado carregamento. Ainda mais, através do RFID é possível conectar-se ao sistema central indicando se o produto deve fazer o movimento de cross-docking. Cross-docking é um dos mais eficientes processos para movimentação de materiais através do centro de distribuição sem armazenamento, com início nas docas de recebimento.

Quando um produto é recebido e detectado pelo leitor, o sistema de controle verifica se este produto é necessário para completar alguma ordem em aberto. Se for, o produto é,

literalmente, movido através da doca (daí o termo cross-docking) até a doca de saída para que a ordem possa ser completada e posta no veículo. Se o produto não faz parte de uma ordem, ele é armazenado normalmente. Dessa forma, o RFID tornará a identificação de ordens em aberto muito mais rápidas e precisas em relação ao código de barras, pois ocorrerá assim que o produto é retirado do caminhão de entrega e não no chão das docas de entrada.

Outra grande vantagem que se tem utilizando o RFID em centros de distribuição ocorre na movimentação de materiais dentro do armazém. Como sabemos, para que ocorra a leitura de um objeto pelo sistema de código de barras, este deve estar posicionado corretamente em relação ao leitor. Essa necessidade é conhecida como face-up. Isso aumenta a demanda operacional e diminui a velocidade e precisão do fluxo de materiais dentro do armazém. Utilizando a tecnologia RFID perde-se a necessidade de um produto estar posicionado corretamente, praticamente eliminando erros de manipulação de objetos tanto por robôs quanto por operários. Os processos dentro de um centro de distribuição podem ser representados pela figura 6.

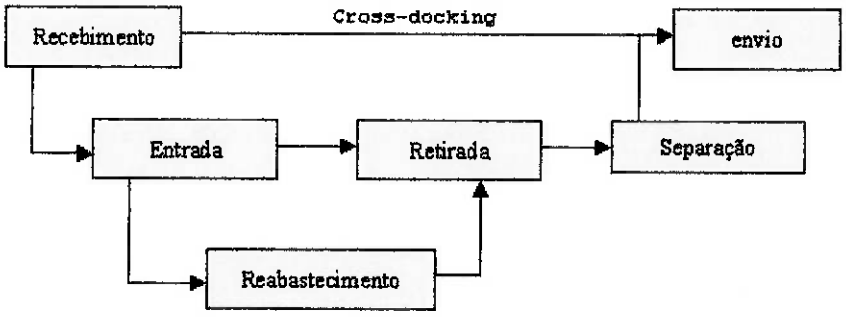

Figura 6 - Processos em um centro de distribuição

5 SISTEMAS EM BIBLIOTECAS

Atualmente, a maioria das grandes bibliotecas do Brasil utilizam a tecnologia de leitura de código de barras para efetuar o controle do acervo e executar os processos de registro de empréstimo e devolução de livros, eletronicamente. Esta tecnologia, apesar de largamente utilizada, apresenta algumas deficiências que podem ser corrigidas com a utilização da tecnologia RFID para o mesmo propósito.

Os processos de empréstimo e devolução são de vital importância dentro do sistema de informação de uma biblioteca. Tomando com base o Sistema de Bibliotecas da USP, será apresentada uma análise, dando especial enfoque à Biblioteca de Engenharia Mecânica, Naval e Oceânica da Escola Politécnica, da qual foram extraídos dados estatísticos para composição de um modelo de simulação computacional do seu funcionamento, objetivando, dessa forma, confrontarmos formalmente o sistema em atividade atualmente com o projeto proposto no próximo capítulo.

5.1 O SISTEMA DE BIBLIOTECAS DA USP

Desde a criação do Sistema Integrado de Bibliotecas da USP SIBi/USP, pela Resolução 2.226, de 8 de julho de 1981, a automação das informações bibliográficas dos acervos das bibliotecas do Sistema foi o objeto principal de sua ação, consolidada com a criação do Banco de Dados Bibliográficos da USP – DEDALUS.

O DEDALUS, criado em 1986 e regulamentado pela Portaria GR – 2722 de 16.11.94, é o instrumento oficial da Universidade que reúne os registros de informações bibliográficas da produção gerada na USP e dos acervos das bibliotecas, para recuperação e localização de livros e materiais especiais. Originalmente, foi instalado em computadores de grande porte

(Burroughs/Unisys), com programa desenvolvido na própria Universidade, pelo Centro de Computação Eletrônica, tendo recebido as implementações possíveis de software e hardware, que apresentavam limitações para o desenvolvimento pretendido.

Assim, em consonância com os avanços tecnológicos e com a política de informática da USP, através do projeto de modernização do SIBi/USP, criou-se as condições necessárias à implantação de uma nova rede de Serviços - SIBiNet, a partir do banco de dados DEDALUS. A figura 7 apresenta a nova configuração do SIBi/USP.

Figura 7 - Configuração do SIBiNet

Dentro desse projeto, foi realizado um processo para seleção e aquisição de um novo programa para armazenagem e recuperação das informações existentes no DEDALUS, com possibilidade de integração de funções como aquisição, empréstimo e catálogo Online para consulta dos usuários.

Como resultado, foi adquirido o software ALEPH (Automated Library Expandable Program) da empresa EX-LIBRIS, compatível com o ambiente cliente/servidor previsto, além do hardware apropriado para abrigar o software em questão. O ALEPH é um software integrado, próprio para o gerenciamento de bibliotecas e centros de documentação e informação, no qual cada rotina administrativa de biblioteca é gerenciada por um módulo específico. As operações Online ocorrem em tempo real, de modo que toda e qualquer alteração nos registros é efetuada imediatamente e a informação resultante se torna disponível a todas as operações subseqüentes à alteração.

O módulo catálogo Online permite recuperar registros bibliográficos de forma conversacional, utilizando o método de consulta a índices (SCAN/Browse – percurso seqüencial em listas ordenadas, a partir de um determinado item fornecido pelo usuário), ou a partir de estratégias de busca FIND, utilizando os recursos disponíveis no aplicativo, com o uso de caracteres maiúsculos ou minúsculos. Pode ser usado na modalidade texto ou nas modalidades gráficas: GUI (Graphical User Interface) e WWW (World Wide Web).

5.1.1 AVALIAÇÃO DAS BIBLIOTECAS DA USP

O SIBi/USP, através de seu Departamento Técnico, analisou os resultados da Avaliação Institucional da Universidade, em 2003, referente às Bibliotecas do Sistema.

Das 39 Bibliotecas que compõem o SIBi/USP, foram coletadas informações das 30 Bibliotecas de Unidades de Ensino e Pesquisa. Não foram avaliadas as 09 Bibliotecas, situadas nos Centros e Institutos de Pesquisa, Hospital e Museus.

A análise se baseou nos aspectos positivos e negativos da atuação das Bibliotecas, identificados nos textos coletados e categorizados por áreas e atividades que permeiam os

processos do SIBi/USP, e estabeleceu a relação entre aqueles pontos e as ações efetivadas em âmbito sistêmico. Para isso, foram utilizados:

- Documento da Avaliação Institucional da USP – item 1.1.3.1;
- Programas Integrados da USP, destinados ao SIBi/USP;
- Projetos Estratégicos do SIBi/USP – gestão 2001-2005.

Do total de 11 Bibliotecas analisadas na área de Ciências Exatas e Tecnologia, das quais 8 fazem parte da Escola Politécnica, observam-se como pontos negativos nas 3 primeiras posições: Necessidade de Ampliação da Área Física, com 71,4%, seguidos de Necessidade de Ampliação de Acervo e Necessidade de Ampliação/Atualização de Equipamentos de Informática, empatados com 42,9%, Automação e Necessidade de Equipamentos de Segurança empatados com 28,6%. A figura 8 mostra alguns resultados da pesquisa.

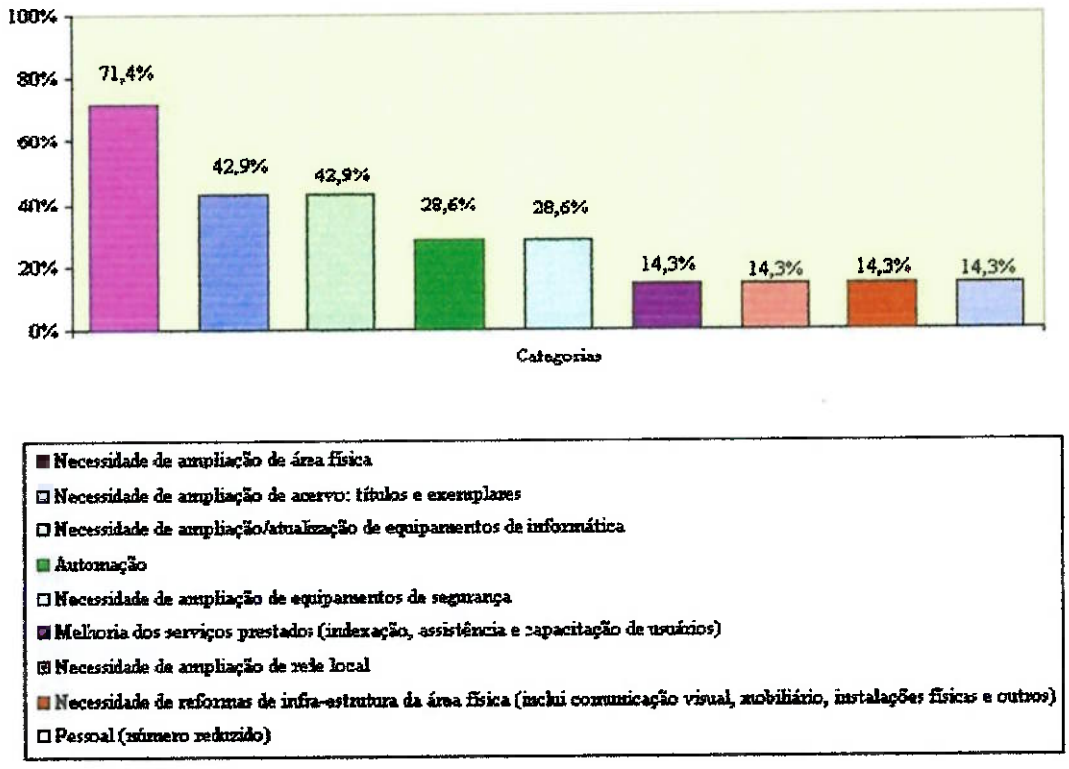

Figura 8 - Avaliação das bibliotecas

5.2 PROCESSOS NAS BIBLIOTECAS DA USP

A partir da avaliação das bibliotecas, o SIBi/USP constata que o processo de modernização precisa avançar, de forma a melhorar os serviços prestados. A necessidade de automação induz naturalmente a migração para novas tecnologias.

A fim de estudarmos as falhas no sistema, esta seção apresenta um modelo dos processos de empréstimo e devolução atualmente praticados pelo SIBi/USP, sob a ótica da simulação em Redes de Petri. O SIBi/USP regulamenta os procedimentos em manuais de serviços e pelo fato da Biblioteca da Engenharia Mecânica estar em conformidade com esses procedimentos, será a base para composição do modelo e também fonte dos dados estatísticos empregados.

5.2.1 SIMULAÇÃO E REDES DE PETRI

Simulação é, em geral, entendida como a imitação de uma operação ou de um processo do mundo real. A simulação envolve a geração de uma história artificial de um sistema para a análise de suas características operacionais.

O comportamento de um sistema pode ser estudado através de um modelo de simulação. Este modelo geralmente utiliza diversos parâmetros sobre a operação do sistema. Uma vez desenvolvido e validado, o modelo pode ser usado para investigar uma grande variedade de questões sobre o sistema. Mudanças no sistema podem ser simuladas a fim de prever seu impacto no seu desempenho. A simulação pode também ser usada para estudar sistemas ainda na fase de concepção, antes que sejam efetivamente implementados. Assim, a simulação pode ser usada como uma ferramenta para predizer os efeitos de uma mudança em

sistemas existentes e também como uma ferramenta de projeto para avaliar e validar o desempenho de novos sistemas.

Existem casos onde um modelo é baseado em formulações matemáticas. Este modelo é, em geral, desenvolvido através de equações integro-diferenciais, teoria de probabilidades, métodos algébricos, etc. Entretanto, muitos sistemas na vida real são tão complexos que seus modelos matemáticos são muito difíceis de serem formulados ou utilizados. Nestes casos, utilizam-se as técnicas de simulação para imitar o comportamento do sistema num certo intervalo de tempo.

Um modelo de simulação é um tipo particular de modelo matemático de um sistema e podem ser classificados em: modelos dinâmicos, que representam sistemas e como eles se comportam em função dos eventos passados e com o decorrer do tempo; modelos instantâneos, que representam sistemas e como eles se comportam em função apenas dos eventos atuais, isto é, não se considera os eventos passados; modelos determinísticos, que têm um conjunto conhecido de entradas, os quais resultarão em um único conjunto de saídas; modelos estocásticos, que possuem uma ou mais variáveis aleatórias como entrada que levam a saídas aleatórias, isto é, as saídas da simulação estocástica devem ser tratadas como estimativas estatísticas das características reais de um sistema; modelos discretos e contínuos, que são definidos de acordo com as mesmas considerações que definem se um sistema é discreto ou contínuo.

A simulação de sistemas a eventos discretos é própria para a análise de sistemas no qual o estado (discreto) das variáveis muda apenas com a ocorrência de eventos (considerados instantâneos). Os modelos de simulação são analisados por métodos numéricos ao invés de métodos analíticos, isto é, em vez de métodos analíticos que empregam o raciocínio dedutivo/matemático para resolver um modelo, consideram-se métodos numéricos que empregam procedimentos computacionais para executar os modelos matemáticos.

Dentro desse contexto de simulação a rede de Petri é uma ferramenta gráfica e matemática que se adapta bem a um grande número de aplicações em que as noções de eventos de evoluções simultâneas são importantes. O modelo de rede de Petri foi proposto por Carl Petri em 1962, na Universidade de Darmstadt, Alemanha. Entre as aplicações pode-se citar: avaliação de desempenho, análise e verificação formal em sistemas discretos, protocolos de comunicação, controle de oficinas de fabricação, concepção de software em tempo real e/ou distribuído, sistemas de informação, logística, entre outros.

Os elementos básicos que permitem a definição de uma rede de Petri são:

- Lugar (representado por um círculo): pode ser interpretado com uma condição, um estado parcial, uma espera, um procedimento, etc;
- Transição (representada por um retângulo): é associada a um evento que ocorre no sistema, seja ele instantâneo, temporal ou estocástico;
- Marcas (representada por um ponto num lugar): é um indicador significando que a condição associada ao lugar é verificada;
- Arcos (representado por setas): fazem a ligação entre dois estados, através de uma transição.

As linguagens de simulação e os pacotes de simulação discreta são ferramentas muito úteis para a simulação de sistemas discretos. Existem linguagens de programação de uso geral como Pascal, C, C++, etc, e linguagens específicas de simulação como GPSS, SIMAN V, SLAM II, e também pacotes de simulação orientada a objetos como MODSIM III e similares.

O HPSim é um software para simulação de redes de Petri e entre suas vantagens está a possibilidade do acompanhamento da evolução do estado da rede de uma forma gráfica, o que auxilia no desenvolvimento do modelo e na detecção de erros. Ele permite ainda a gravação do resultado da simulação e seu posterior tratamento em softwares como o Microsoft Excel, uma característica essencial para a análise de sistemas.

5.2.2 MODELO DA BIBLIOTECA DA ENGENHARIA MECÂNICA

O SIBiNet não possui um sistema automático de captura de dados para composição de estatísticas da movimentação do acervo. Dessa forma, cada unidade deve manter um serviço de coleta de dados manual, isto é, armazenando diariamente, em planilhas do Microsoft Excel, os dados da movimentação do acervo. A figura 9 mostra os dados da movimentação do acervo do mês de agosto de 2005, fornecidos pela Biblioteca da Engenharia Mecânica.

EPUSP - Serviço de Bibliotecas

Estatística Mensal - Movimentação do Acervo

Biblioteca: *EPMN* ***AGOSTO*** ***2005*** ***Dias Úteis*** ***23***

UTILIZAÇÃO DO ACERVO	USUÁRIOS		TOTAL Livros	Teses	Period.	Multimeios	Outros/TF	TOTAL GERAL	MÉDIA DIÁRIA
CONSULTAS	EP	Aluno Grad.	5 497	184	328	1	619	6.629	296,91
		Pós-Graduação	2 782	291	457	1	63	3.594	156,26
		Prof./Pesq.	304	88	196	0	12	590	25,65
		Funcionário	5	1	0	1	0	7	0,30
	USP	Aluno Grad.	23	0	2	0	2	27	1,17
		Pós-Graduação	15	3	12	0	0	30	1,30
		Prof./Pesq.	0	0	0	0	0	0	0,00
		Funcionário	0	0	0	0	0	0	0,00
	Outros		17	0	0	0	12	29	1,26
	Total Consultas		8.643	567	985	3	908	11.106	482,87
EMPRÉSTIMOS	EP	Aluno Grad.	859	58	13	2	81	1.013	44,04
		Pós-Graduação	318	57	25	0	19	419	18,22
		Prof./Pesq.	109	2	16	1	0	128	5,57
		Funcionário	8	0	2	0	0	8	0,35
	USP	Aluno Grad.	7	0	0	0	0	7	0,30
		Pós-Graduação	2	0	0	0	0	2	0,09
		Prof./Pesq.	0	0	0	0	0	0	0,00
		Funcionário	0	0	0	0	0	0	0,00
	EPBC		5	0	12	0	1	18	0,78
	Total Empréstimos		1.306	117	68	3	101	1.595	69,35
	EEBs	Solicitado pela EP	50	3	0	0	0	53	2,30
		Fornecido pela EP	57	3	1	0	3	64	2,78
	Total EEBs fornecidos		57	3	1	0	3	64	2,78
Movimentação do Acervo	Escola Politécnica		9 880	881	1.027	0	894	12.588	547,30
	Outros		126	6	27	0	18	177	7,70
	Total		10.006	887	1.054	6	1.012	12.765	555,00
INSCRIÇÕES	EP	Aluno Grad.	-	-	0	-	-	0	0,00
		Pós-Graduação	-	-	0	-	-	0	0,00
		Prof./Pesq.	-	-	0	-	-	0	0,00
		Funcionário	-	-	0	-	-	0	0,00
	USP	Aluno Grad.	-	-	0	-	-	0	0,00
		Pós-Graduação	-	-	0	-	-	0	0,00
		Prof./Pesq.	-	-	0	-	-	0	0,00
		Funcionário	-	-	0	-	-	0	0,00
	Outros		-	-	0	-	-	0	0,00
	TOTAL		-	-	0	-	-	0	0,00
Obras arquivadas			-	-	11.318	-	-	11.318	492,09
CATRACA			1ª SEMANA	2ª SEMANA	3ª SEMANA	4ª SEMANA	5ª SEMANA		
			714	758	759	867	765	3 863	168

Figura 9 - Movimentação do acervo – agosto/2005

Baseado nesses dados mensais, um modelo em Rede de Petri dos processos de empréstimo e devolução, do funcionamento diário da biblioteca, é apresentado na figura 10.

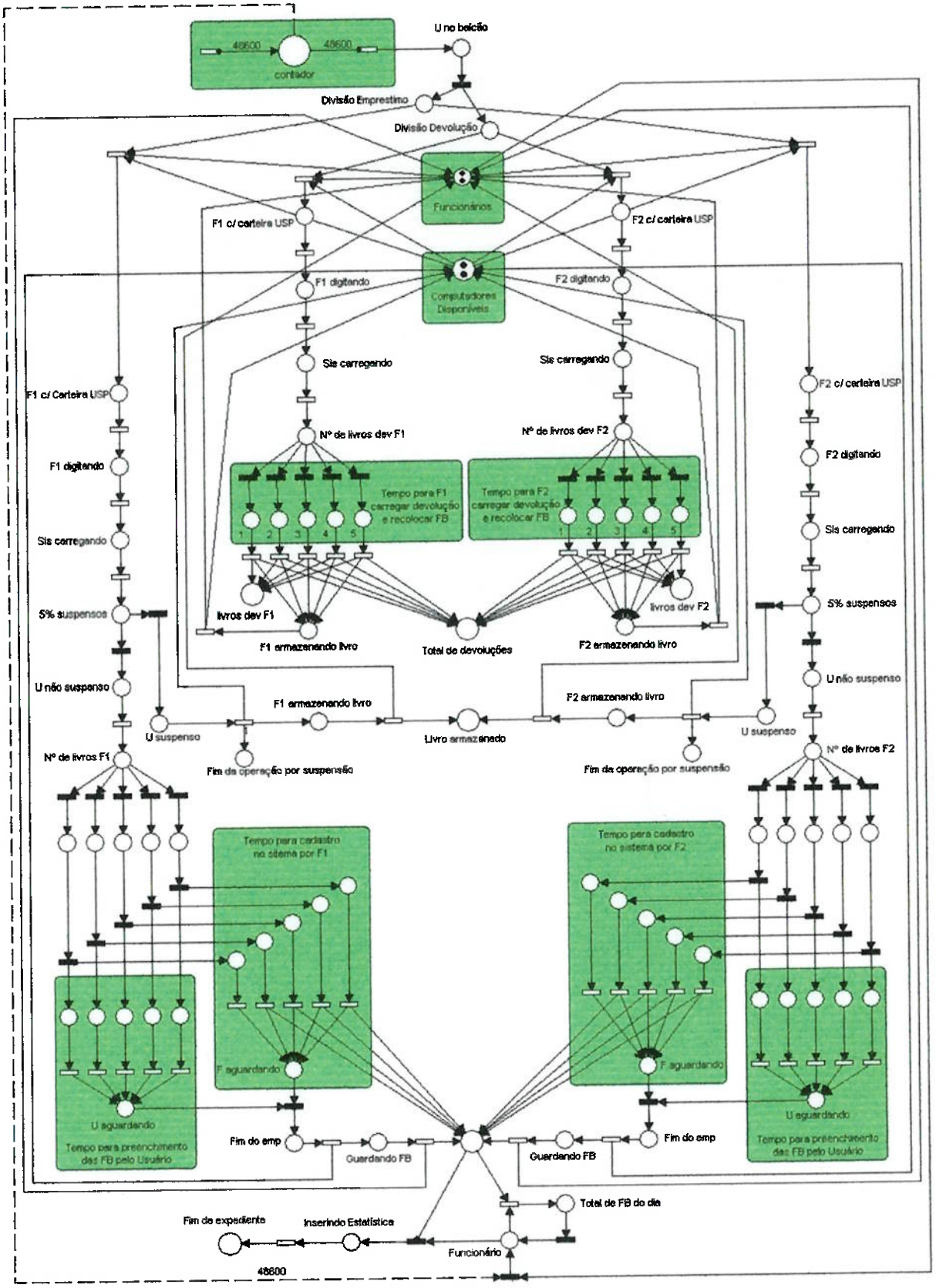

Figura 10 - Modelo do Sistema da Biblioteca da Engenharia Mecânica

Nesse modelo, o contador representa um marcador de tempo. Após o período de operação diária esse dispositivo inibe a chagada de novos usuários na biblioteca, representando seu fechamento. Para um balanço diário, assume-se que aproximadamente 50% dos usuários que chegam a biblioteca realizarão um empréstimo e os outros 50% realizarão uma devolução.

Essa Rede de Petri pode ser classificada como estocástica, já que os intervalos de duração de cada operação não são fixos, seguindo uma Distribuição Exponencial. Tal distribuição envolve probabilidades, ao longo do tempo num intervalo contínuo e se adere bem, por exemplo, ao modelo de falhas em equipamentos elétricos ou a chegada de clientes a uma loja. Essa função é implementada no software HPSim definindo-se a média de tempo λ, que é o parâmetro de entrada na definição de uma transição desse tipo. A figura 11 mostra a formulação e a curva característica dessa distribuição.

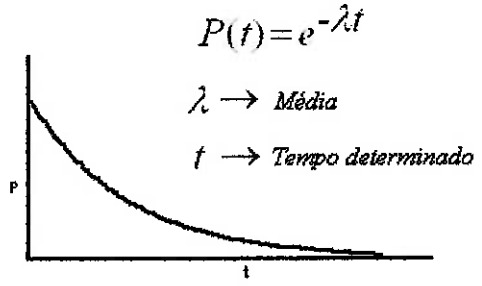

Figura 11 - Distribuição Exponencial

Utilizando estimativas fornecidas por funcionários, foi definido que em média 5% dos usuários apresentam alguma punição administrativa e não podem retirar livros. Quanto ao número de livros que podem ser retirados por empréstimo, que são cinco para alunos, foi utilizado uma proporção exponencial de base 4, indicando que a cada 4^x usuários retiram $5-x$ livros, com x variando entre 0 a 4.

O modelo apresenta dois blocos que possibilita a simulação de dois funcionários simultaneamente, isto é, os dois funcionários podem estar executando a mesma operação, durante um empréstimo ou devolução, ao mesmo tempo. Essa característica se deve ao fato de que na maior parte das bibliotecas encontramos dois funcionários no atendimento ao usuário. Isso porem não é um fator limitante do modelo, podendo ser simulado com qualquer número de funcionários ou de computadores, com a ressalva de que as operações com mais de 2 funcionários ocorrerem com defasagem de um step.

5.2.3 DESCRIÇÃO DO MODELO

O processo de empréstimo inicia-se com um usuário esperando atendimento no balcão de recepção da biblioteca. Havendo um funcionário e um computador disponível, o funcionário requisita a apresentação da carteira USP ao usuário e digita o número USP na tela de entrada de dados do sistema. Caso o usuário tenha alguma restrição e encontra-se sob punição administrativa, o processo encerra-se e o livro é armazenado. Se não há restrição, o modelo define o número de livros, segundo a proporção já discutida, e iniciam-se duas operações simultâneas. Uma de preenchimento das fichas brancas, que são cartões de controle com data e assinatura que ficam alojadas na capa do livro, pelo usuário e outra de inserção no sistema das informações de empréstimo pelo funcionário.

Para cada operação, existe um intervalo de tempo associado, que é função do número de livros que está sendo retirado. O empréstimo é finalizado e o funcionário libera o livro para o usuário.

Ao término do período de funcionamento diário da biblioteca, um funcionário ordena as fichas brancas alfabeticamente, a fim de organizar e facilitar a procura no processo devolução.

O processo de devolução também inicia-se com um usuário esperando atendimento no balcão de recepção da biblioteca. Novamente, havendo um funcionário e um computador disponível, o funcionário requisita o número USP ou obtém o sobrenome do autor diretamente do livro e digita na tela de entrada de dados do sistema, não sendo necessário a apresentação da carteira USP. O funcionário executa a operação de devolução no sistema, para cada livro. O processo é finalizado e o funcionário procura e recoloca a ficha branca de cada livro e o armazena.

5.2.4 RESULTADOS DA SIMULAÇÃO

O software HPSim possui uma ferramenta para gravar e exportar dados de uma simulação. Os dados são exportados para um arquivo de extensão *.csv* que é um formato tipo texto suportado pelo MS Excel. Como o modelo é relativamente complexo, possuindo muitos estados e transições, a simulação tem um custo computacional alto, de modo que cada simulação transcorreu em um intervalo de tempo de aproximadamente 15 minutos, gerando arquivos de 60 Mb. A figura 12 mostra uma parte dos dados de uma simulação.

	A	B	D	E	F	G	H	I	J
1	Simulation Data generated by HPSim Nov-11-2005 15 58 26								
2	Count/ Steps	Time/ ms	contador	U no balcão	Funcionários	F1 c/ Carteira USP	F1 digitando	Sis carregando	Nº de livros dev F1
3	1	1	0	0	2	0	0	0	0
4	2	1	1	0	2	0	0	0	0
5	3	2	1	0	2	0	0	0	0
6	4	2	2	0	2	0	0	0	0
7	5	3	2	0	2	0	0	0	0
8	6	3	3	0	2	0	0	0	0
9	7	4	3	0	2	0	0	0	0
10	8	4	4	0	2	0	0	0	0
11	9	5	4	0	2	0	0	0	0
12	10	5	5	0	2	0	0	0	0
13	11	6	5	0	2	0	0	0	0
14	12	6	6	0	2	0	0	0	0
15	13	7	6	0	2	0	0	0	0
16	14	7	7	0	2	0	0	0	0
17	15	8	7	0	2	0	0	0	0
18	16	8	8	0	2	0	0	0	0
19	17	9	8	0	2	0	0	0	0
20	18	9	9	0	2	0	0	0	0
21	19	10	9	0	2	0	0	0	0
22	20	10	10	0	2	0	0	0	0
23	21	11	10	0	2	0	0	0	0
24	22	11	11	0	2	0	0	0	0
25	23	12	11	0	2	0	0	0	0
26	24	12	12	0	2	0	0	0	0
27	25	13	12	0	2	0	0	0	0
28	26	13	13	0	2	0	0	0	0

Figura 12 - Dados de simulação

O modelo foi simulado para uma operação diária da biblioteca, constituído de um período de 13,5 horas ou 48600 segundos. Utilizando as ferramentas matemáticas e gráficas do MS Excel é possível extrairmos resultados convenientes. Por exemplo, a demanda de usuários para empréstimo e devolução segue uma Distribuição Exponencial, apresentada na figura 13.

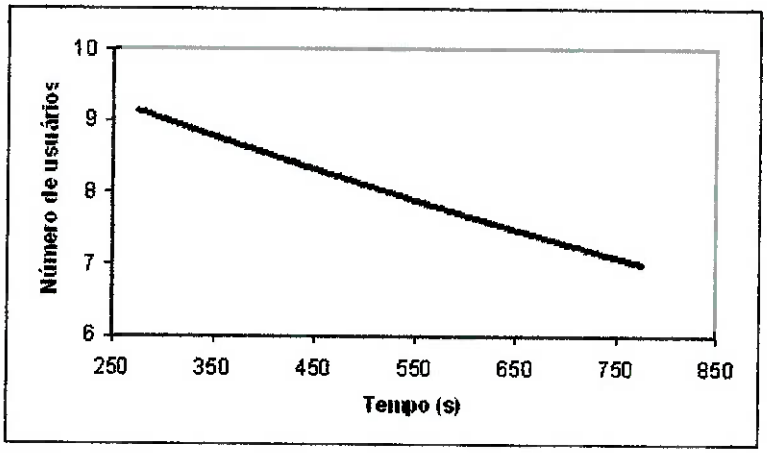

Figura 13 - Demanda de Usuários

Foram realizadas cinco simulações e os dados de desempenho estão apresentados na tabela 1. O número de empréstimos somado ao de devoluções supera o número de usuários no balcão, isso ocorre porque um único usuário pode retirar até cinco livros por empréstimo, segundo o modelo discutido no item 5.2.2.

Tabela 1 - Resultados da simulação de uma diária

Simulação	Usuários no balcão	Total de empréstimos	Total de devoluções	Tempo médio de Empréstimo/Devolução (s)	Tempo médio para ordenar Fichas Brancas (min)
1	94	75	73	37	17
2	98	85	61	36	19
3	94	73	75	37	18
4	98	85	61	36	19
5	94	75	73	37	17

Para conhecermos o desempenho do modelo frente um pico de demanda, isto é, um número muito grande de usuários para serem atendidos num dado instante do dia, o modelo foi simulado para um número infinito de usuário no balcão durante um intervalo de tempo de uma hora. Desse modo podemos identificar qual o número médio de usuários que o sistema comporta durante um pico de demanda. A tabela 2 apresenta os dados de cinco simulações de um pico de demanda. A análise dos dados dessa seção será apresentada adiante, dando um enfoque comparativo entre o sistema em operação modelado e o projeto para automação de bibliotecas que será proposto no próximo capítulo.

Tabela 2 - Resultados da simulação de um pico de demanda de usuários

Simulação	Usuários atendidos	Emprestimos	Devoluções	Tempo médio de Empréstimo/Devolução (s)
1	122	85	99	38
2	127	88	93	37
3	137	97	91	37
4	133	93	94	37
5	133	92	91	36

6 AUTOMAÇÃO DE BIBLIOTECAS

É cada vez mais iminente o uso da tecnologia de Rádio Freqüência nos diversos ambientes em que o gerenciamento eficiente de materiais se faz necessário. A tecnologia RFID tem como premissa básica ampliar significativamente a produtividade e o nível de serviço das bibliotecas. Marcar cada item da biblioteca com uma pequena etiqueta contendo um minúsculo chip eletrônico pode reduzir o tempo de checagem de entrada e saída, localizar itens fora do lugar e gerenciar melhor as estantes da biblioteca, entre outros benefícios.

Motivados pelo potencial dessa tecnologia, essa seção apresenta a estrutura de um projeto que implementa os processos de empréstimo e devolução em uma biblioteca.

6.1 SUN JAVA SYSTEM RFID SOFTWARE

O Sun Java System RFID é uma representação do Savant, com algumas diferenças, desenvolvida pela SUN Microsystems. O software consiste basicamente de dois módulos: o Event Manager e o Information Server. Este sistema, comumente denominado como middleware, é responsável pela filtragem, separação, e contagem de dados das etiquetas (dados de evento). Pode ser distribuído por diversos nós de computadores de forma que não exista nenhum ponto de falha.

O Event Manager ou gerenciador de eventos é projetado para processar as séries de etiquetas ou dados provenientes dos sensores que vem da rede de dispositivos de leitura. O Information Server ou servidor de informação é uma aplicação J2EE™ que serve como uma interface para captura e busca de dados relacionados aos eventos EPC's. Estes dados incluem observação de etiquetas provenientes do Event Manager, além de informações que mapeiam

EPC's para níveis mais altos. Seu objetivo é traduzir um conjunto de observações de baixo nível em informações de negócios de nível mais alto.

O Information Server roda sobre um Aplication Server e sustenta transportes de mensagem de HTTP e JMS, com aplicações externas usando o protocolo de XML para troca de mensagens. Um banco de dados JDB fornece os meios para armazenar e administrar dados de etiquetas e o mapeamento relacionado para códigos de produto e informações de produto. A figura 14 apresenta a arquitetura do sistema.

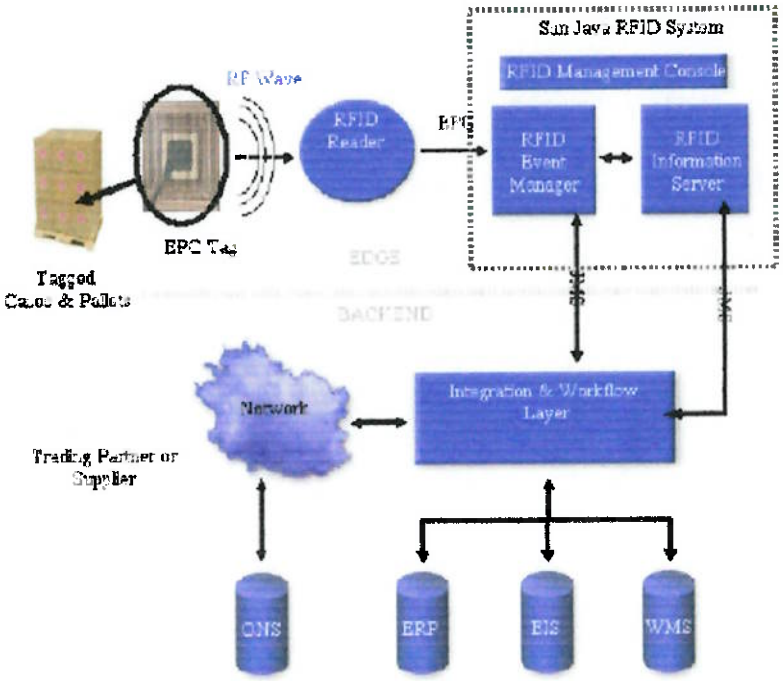

Figura 14 - Arquitetura Sun Java RFID System

Os eventos, capturados pelos leitores, podem ser filtrados e processados pelo Event Manager e em seguida fornecidos ao Information Server para que este possa disponibilizá-los para um banco de dados, que pode ainda relacionar essas informações com as informações de nível mais alto. Essa integração é feita através de API's Java fornecidos pelo sistema para que

seja feita a manipulação de dados necessária para, por exemplo, posterior envio a um ERP ou mesmo à rede ONS, que disponibiliza informações para toda a cadeia de suprimentos.

6.1.1 EVENT MANAGER

O Event Manager é um sistema de gerenciamento de eventos baseado no Jini. Ele facilita a captura, filtragem e um eventual armazenamento do EPC e os eventos gerados pelos leitores de RFID conectados a rede. Sua principal função é fazer a interface com leitores de RFID, reunião dos eventos de EPC, filtragem das informações redundantes, e alimentação dos eventos relevantes para o Information Server ou outro software. O Event Manager consiste em uma Estação de Controle com um ou mais Agentes de Execução, como apresentado na figura 15.

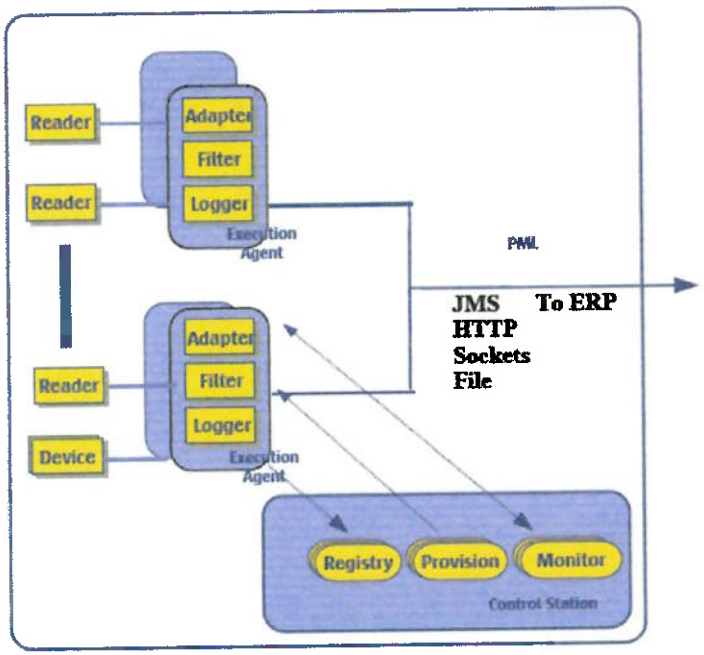

Figura 15 - Arquitetura do Event Manager

Os agentes de execução fazem a reunião e filtragem das informações provenientes dos leitores. Um computador tipicamente roda um Agente de Execução, que pode executar uma ou mais cargas de trabalho. Cada Agente de Execução é registrado no início com a Estação de Controle, que por sua vez atribui uma carga de trabalho para o Agente de Execução. Cada Agente de Execução é composto de um adaptador passando informações para um ou mais filtros ou conectores. Os filtros, por sua vez, podem passar informações para um ou mais conectores. Esta cadeia dos processos constitui uma federação de serviços, conhecidos como Business Processing Semantics (BPS), onde cada um dos serviços pode conter um ou mais componentes ligados juntos com eventos.

Um Leitor de RFID é um hardware que comunica com as etiquetas de RFID através de uma Freqüência de Rádio de microondas. Também se comunica com um adaptador por um protocolo proprietário. O adaptador é semelhante a um driver, isto é, um pedaço de código Java que especifica a interface do dispositivo de Leitor de RFID real.

O adaptador funciona recebendo o EPC da etiqueta de RFID e gerando um evento que inclui uma estampa de tempo e a fonte do evento. O evento é postado para um conjunto de ouvintes, como filtros ou conectores, que processam o evento, ilustrado na figura 16.

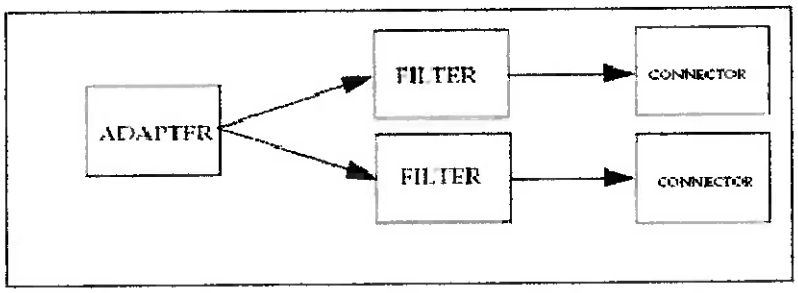

Figura 16 - Adaptador, filtros e conectores

Os filtros podem "suavizar" os dados, jogar fora eventos previamente descobertos ou rotear informações para outros filtros e conectores baseados em comparação de máscara. Os

dados filtrados são postados como um evento para definir ouvintes. Os ouvintes podem ser outros filtros ou conectores que servem como um link para aplicações terceiras que usam as informações de RFID. Esta coleção de adaptadores, filtros e conectores são conhecidos como um Papel.

A Estação de Controle mantém um registro de agentes de execução disponível, monitora sua condição, atribui cargas de trabalho, e fornece a eles as classes Java necessárias para executar a carga de trabalho. É possível especificar que uma carga de trabalho seja executada por um Agente de Execução com um nível dado de recursos e um conjunto dado de capacidades. Por exemplo, pode ser especificado que a carga de trabalho deve ser executada por um Host com 25% ou menos de uso de CPU e uma conexão 802.11 b.

6.1.2 INFORMATION SERVER

O Information Server é uma aplicação J2EE™ que roda sobre um Aplication Server, onde todos os dados são armazenados em um banco de dados.

Aplicações externas fazem interface com o Information Server através da troca de mensagens XML, onde os pedidos e respostas são expressos em XML conforme um esquema padronizado. O Information Server sustenta transportes de mensagem de HTTP e JMS. O Software de RFID fornece uma biblioteca de cliente Java para acessar o Information Server. A arquitetura é apresentada na figura 17.

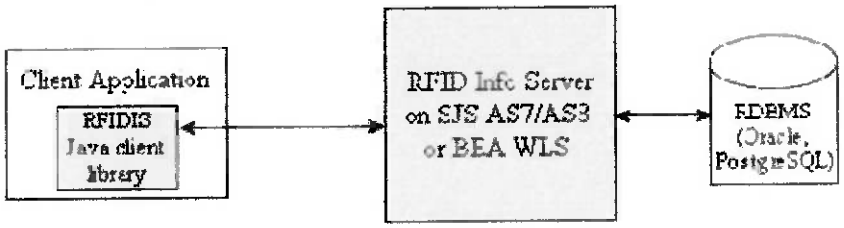

Figura 17 - Information Server

O Sun Java System RFID Software usa a API de JMS como um de seus métodos primários para comunicar com software de terceiros. A comunicação com o Information Server é sem estados e síncrona. Se HTTP é usado como transporte, o cliente usa HTTP POST para comunicar com o Information Server. Para implementar pedidos síncronos com API de JMS, o cliente usa um ID ou identificação de mensagem para relacionar pedidos com respostas. Mensagens de JMS (pedidos e respostas) são postadas para um tópico conhecido.

6.2 PROJETO DOS PROCESSOS DE EMPRÉSTIMO E DEVOLUÇÃO

Através da análise na demanda de melhorias no sistema de bibliotecas da USP, adotamos algumas hipóteses a fim de viabilizar um teste piloto de um módulo que compõem o sistema, de forma que os resultados encontrados podem ser diretamente absorvidos quando na implantação do sistema como um todo. Esse módulo, que é o foco desse trabalho, trata-se dos processos de empréstimo e devolução de livros. Para tanto, foram adotadas as seguintes hipóteses:

- Usuário é definido como todos os alunos, docentes e funcionários, pertencentes a qualquer unidade USP, portadores de cartão de identificação;
- Não há usuários com penalidades administrativas do tipo suspensão;
- Não há prazo definido para a devolução de livros.

Essas hipóteses foram adotadas, pois são de caráter administrativo, não fazendo parte do escopo desse trabalho, que é demonstrar as facilidades que o RFID pode oferecer. A figura 18 apresenta o fluxo dos processos de empréstimo e devolução de livros, que será implementado.

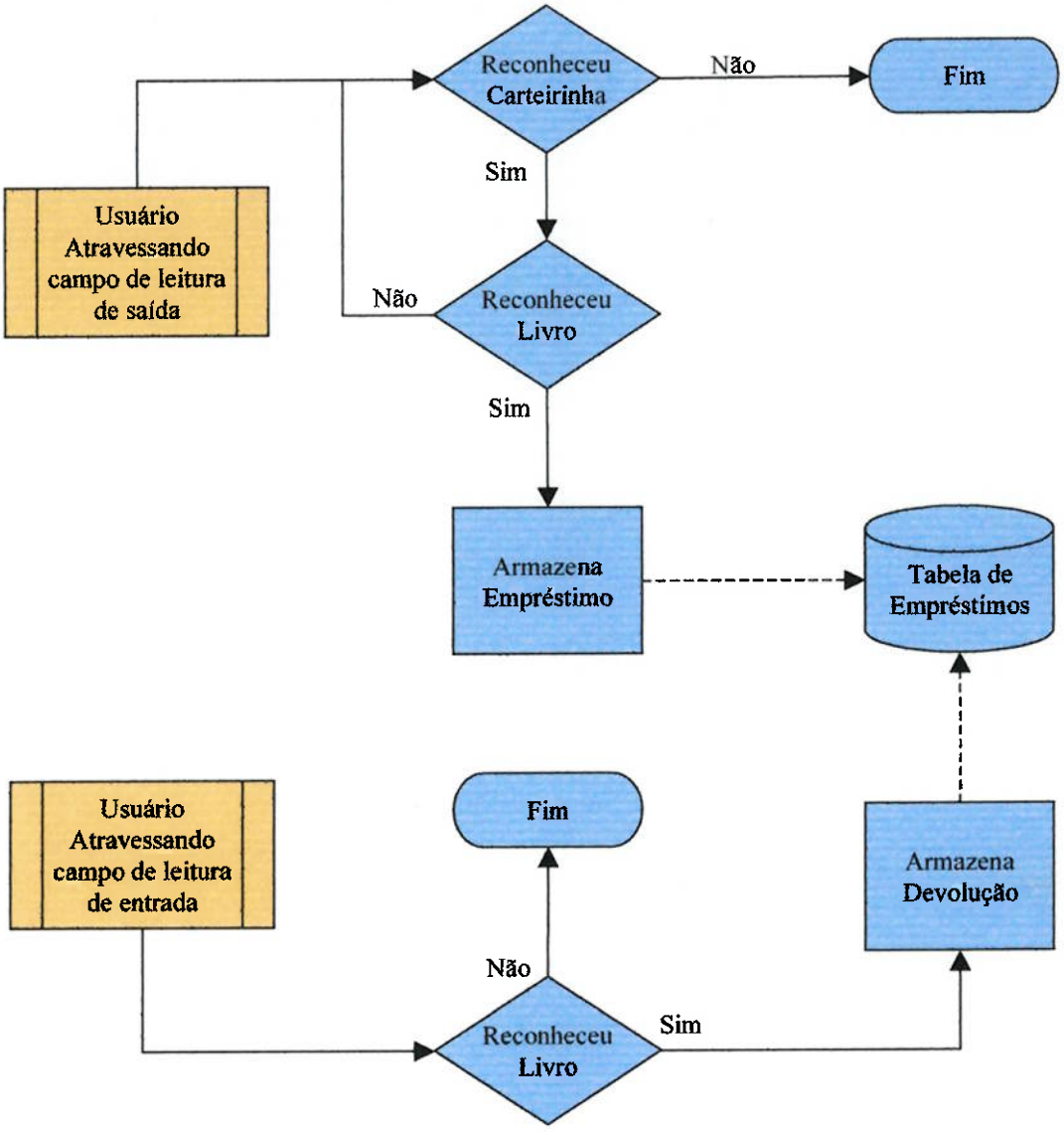

Figura 18 - Processos de empréstimo e devolução

Nesse projeto, a carteirinha utilizada como identificação de usuários tem acoplado uma etiqueta RFID, identificando unicamente cada usuário com um número EPC.

6.2.1 ARQUITETURA DO PROJETO

Na aplicação deste projeto, os dados, tratados pelo Event Manager, são enviados para uma aplicação através de um Socket. Quando o leitor é conectado ao Sun Java System RFID software, o Event Manager cria um Server Socket e começa a enviar os dados para um cliente que por sua vez pode tratá-los da forma desejada. A figura 19 apresenta a configuração do Event Manager.

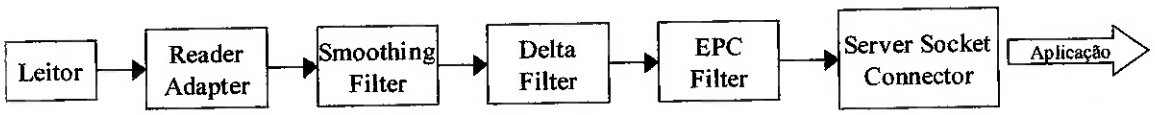

Figura 19 - Configuração do Event Manager do Projeto

A interface entre o leitor e o Sun Java System RFID é realizada pelo Reader Adapter. Cada leitor tem um Reader Adapter particular, que funciona semelhantemente a um driver utilizado para configurar hardwares em um sistema operacional. O Reader Adapter recebe os dados do leitor e os coloca em formato PML (Phisical Markup Language) com estampa de tempo.

O leitor RFID realiza ciclos de leitura, cabe ao usuário determinar o período em que os ciclos ocorrem. Nesta aplicação o ciclo de leitura ocorre a cada 500 ms, isto é, as etiquetas presentes no campo de leitura são reportadas a cada 500 ms, o que gera muitas informações redundantes. Dessa forma, os filtros em série são responsáveis em enviar para o Socket apenas as informações de interesse da aplicação.

A detecção das etiquetas dentro de um campo de leitura não é 100% eficiente, isto é, mesmo que a etiqueta esteja dentro do campo de leitura de um determinado leitor, a resposta não é detectada em todos os ciclos de leitura. Isto pode ocorrer por diversos motivos, dentre

eles a disposição e a distância da etiqueta com relação à antena e a colisão das leituras provenientes de outras etiquetas dentro do campo.

Para contornar este problema, é utilizado um filtro do tipo Smooth que considera a detecção de um evento apenas após um determinado número de ciclos. Assim, uma mudança de estado, como por exemplo, a saída de uma etiqueta do campo, só é reportada após n ciclos de leitura. Na aplicação deste projeto, o filtro está configurado para aguardar 5 ciclos, ou seja, n é igual a 5.

Como mencionado acima, o leitor envia informações da leitura a cada ciclo. Para a grande maioria das aplicações, apenas as mudanças de estado são importantes, isto é, quando as etiquetas saem e entram no campo de leitura.

O Delta Filter é responsável por filtrar as informações redundantes, disponibilizando em sua saída apenas os eventos TagsIn e TagsOut, isto é, as etiquetas que entram e saem do campo. É importante citar que o Delta Filter deve necessariamente estar conectado à saída de um Smoothing Filter para que os eventos TagsIn e TagsOut tenham consistência.

Na solução para automação dos processos de empréstimo e devolução de livros em uma biblioteca, os usuários, assim como os livros, são representados por um número EPC. A aplicação deve diferenciar uma etiqueta que representa um livro de uma que representa um usuário. Para isso é utilizado um EPC Filter que filtra os números EPC conforme uma máscara.

Na aplicação deste projeto, as etiquetas com número EPC acima de 1.1.110 representam os usuários enquanto as etiquetas com número EPC inferiores 1.1.109 representam os livros.

Os dados enviados pelo Server Socket são formatados conforme a linguagem PML que é muito semelhante à linguagem XML. Um trecho do código PML utilizado no projeto é apresentado na figura 20.

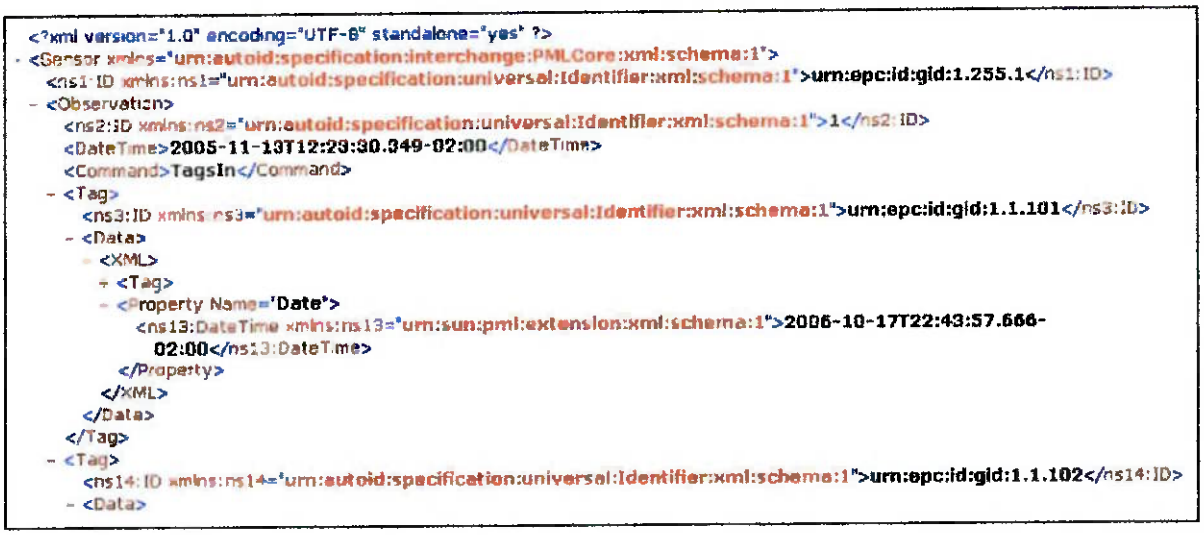

Figura 20 - Trecho de código PML utilizado no projeto

O evento representado pelo formato PML é caracterizado pela identificação do leitor, através de um número EPC, e uma observação. A observação é composta de um campo DateTime, que identifica o dia e a hora em que ela ocorreu, o campo Command, que identifica se o evento é uma entrada ou saída de etiquetas (TagsIn ou TagsOut) e o campo Tag, que identifica a etiqueta propriamente dita.

As etiquetas são identificadas pela expressão "urn:epc:id:gid:1.1.102" que, neste exemplo, identifica o número EPC 1.1.102 no formato GID.

O formato de dados do EPC padronizado consiste em um EPC (ou EPC Identifier) que identifica exclusivamente um objeto e pode incluir um filtro opcional para determinar a consistência das leituras. Para várias aplicações indústrias, a versão EPC 1.1 padrão especifica quaisquer dos esquemas de codificação seguinte:

- General Identifier (GID);
- Versão serial do EAN.UCC Global Trade Item Number (GTIN);
- EAN.UCC Serial Shipping Container Code (SSCC);
- EAN.UCC Global Location Number (GLN);

- EAN.UCC Global Returnable Asset Identifier (GRAI);
- EAN.UCC Global Individual Asset Identifier (GIAI).

Para qualquer tipo de sistema RFID, que identifica um número EPC, o esquema de codificação é identificado anteriormente ao número EPC apresentado.

O padrão EPC determina a definição de uma entidade pura, a identidade associada a uma entidade específica, sendo ela física ou lógica, independentemente do veículo de identificação utilizado, como uma etiqueta RFID ou um código de barras. Esta entidade é definida por uma URI (Uniform Resource Identifier). A URI é uma representação em string de caracteres utilizada para a troca de dados entre componentes de software. No exemplo do item anterior, "urn:epc:id:gid:1.1.102" representa a URI utilizada.

Os dados recebidos do Event Manager através de um socket são tratados por uma aplicação desenvolvida em Java que também grava as informações pertinentes em um banco de dados. As classes da aplicação que fazem essas tarefas estão representadas na figura 21.

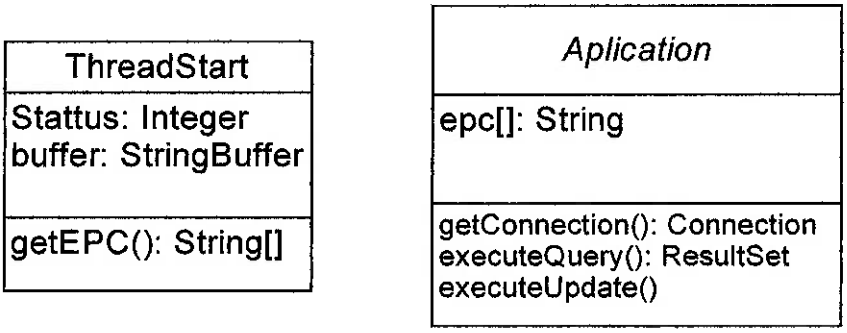

Figura 21 - Diagrama de Classes

A classe ThreadStart cria uma thread que por sua vez cria um conexão com o socket para receber os dados provenientes do Event Manager e guarda-los em um StringBuffer. A variável Status é utilizada para implementação de sincronia de gravação e leitura do buffer, isto é, apenas uma tarefa é executada por vez. O método getEPC() grava os dados do buffer

em uma String e o esvazia. Este método retorna um array de Strings contendo o tipo de evento (TagsIn ou TagsOut), a estampa de tempo, o número EPC do leitor e todos os números EPC's detectados no evento.

A classe Aplication roda um loop infinito que chama o método getEPC toda vez que a variável Status indica a ocorrência de uma gravação no buffer. Os dados obtidos pelo método são comparados com os dados no banco de dados para determinar se o evento consiste em um empréstimo ou uma devolução. Caso seja um empréstimo, os dados são gravados na tabela de empréstimos. Caso seja uma devolução, o campo "data de devolução" é preenchido.

Uma terceira classe é utilizada para apresentar as tabelas do banco de dados para o usuário. Lembrando que essa classe é utilizada apenas para fins de demonstração e consulta, pois todos os processos envolvendo o empréstimo e a devolução são automáticos. Um diagrama de eventos da aplicação é apresentado na figura 22.

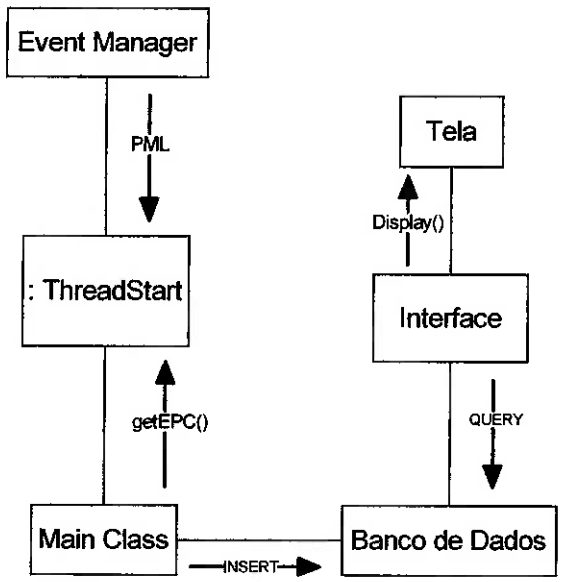

Figura 22 - Diagrama de Comunicação

Para o armazenamento dos dados referentes aos processos de empréstimo e devolução foi utilizado um banco de dados relacional PostGRESQL. A aplicação se comunica com o

banco através de um driver JDBC. As informações são armazenadas em três tabelas: "Usuários", "Livros" e "Empréstimos" conforme a figura 23. A tabela "Empréstimos" contém dois campos Data_E e Data_D que representam as datas de empréstimos e devoluções dos livros, respectivamente. O campo Data_D é utilizado como referência para a aplicação identificar se um determinado livro já foi devolvido ou não.

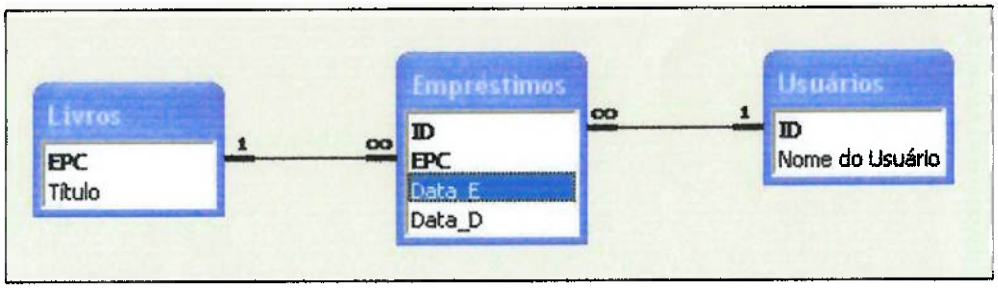

Figura 23 - Relacionamentos das tabelas

6.3 DESEMPENHO DO PROJETO

Sob o ponto de vista operacional, os parâmetros de simulação definidos para o modelo da Biblioteca da Engenharia Mecânica são os mesmos a serem adotados neste projeto, proporcionando, dessa forma, um conjunto de dados coerentes para uma comparação de desempenho.

O projeto foi simulado para uma operação diária da biblioteca, constituído de um período de 13,5 horas ou 48600 segundos. A demanda de usuários para empréstimo e devolução segue a mesma Distribuição Exponencial. O Processo de empréstimo inicia-se com um usuário atravessando o campo de leitura. Nesse momento, em algumas frações de segundo, o sistema realiza todo o processo de leitura, reconhecimento e armazenamento dos dados referentes a esse empréstimo, como apresentado anteriormente.

O Processo de devolução é ainda mais simples, constituído somente pelo usuário atravessando o campo de leitura, efetivando a devolução. A figura 24 apresenta o modelo construído para simular o projeto proposto.

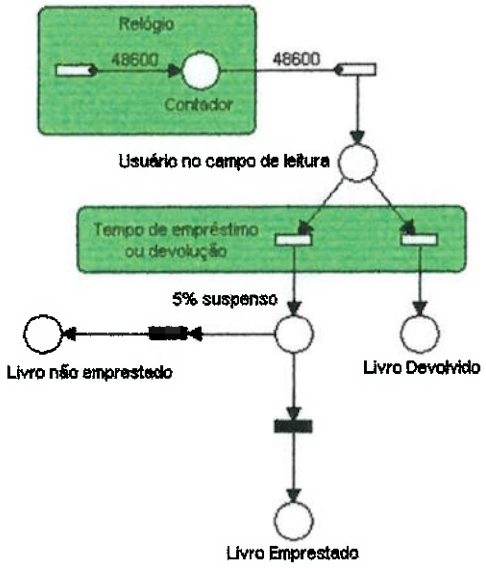

Figura 24 - Modelo do sistema do projeto

Foram realizadas também cinco simulações e os dados de desempenho estão apresentados na tabela 3. Novamente vale lembrar que o número de empréstimos somado ao de devoluções supera o número de usuários no balcão, porque um único usuário pode retirar até cinco livros por empréstimo, segundo o modelo discutido no item 5.2.2.

Tabela 3 - Resultados da simulação de uma diária do projeto

Simulação	Usuários no balcão	Total de empréstimos	Total de devoluções	Tempo médio de Empréstimo/Devolução (s)
1	95	75	72	1
2	97	83	79	1
3	95	74	78	1
4	97	81	75	1
5	97	75	76	1

A fim de conhecermos o desempenho do modelo frente um pico de demanda, o modelo também foi simulado para um número infinito de usuário no balcão durante um intervalo de tempo de uma hora. A tabela 4 apresenta os dados de cinco simulações de um pico de demanda.

Tabela 4 - Resultados de um pico de demanda de usuário no projeto

Simulação	Usuários atendidos	Emprestimos	Devoluções	Tempo médio de Empréstimo/Devolução (s)
1	3596	3057	3049	1
2	3569	3065	3025	1
3	3598	3038	3047	1
4	3587	3071	3053	1
5	3573	3062	3056	1

Os resultados da tabela 4 apresentam a enorme capacidade de atendimento de um sistema que utiliza a tecnologia RFID. A simulação realizada no capítulo 5 apresentou na simulação de pico de demanda, uma capacidade máxima de atendimento de aproximadamente 130 usuários por hora. Ressaltando que numa situação desse tipo, grandes filas são geradas no balcão de atendimento, resultando na insatisfação do usuário.

Na aplicação da tecnologia RFID a simulação apresentou como capacidade máxima de atendimento por volta de 3600 usuários por hora. Como o tempo médio de atendimento é de aproximadamente 1 segundo, praticamente não existe a formação de filas.

O confronto entre estes resultados explicita a grande vantagem da utilização do RFID em relação ao sistema atual em atividade. Ressaltando que a simulação do sistema atual considerou dois funcionários trabalhando simultaneamente, enquanto o sistema RFID não considera a necessidade de um funcionário no balcão de atendimento.

7 CONCLUSÃO

A utilização de código de barras para controle de acervo e de processos em bibliotecas apresenta resultados bastante satisfatórios principalmente em bibliotecas de pequeno e médio porte. O baixo custo de implementação e a relativa baixa movimentação de livros fazem do código de barras a tecnologia preferencial na grande maioria das bibliotecas do Brasil. No entanto, alguns fatores podem influenciar na busca de novas tecnologias para automação e controle de acervo em bibliotecas, especialmente as de grande porte, entre eles: baixa durabilidade das etiquetas com código de barras, difícil localização de livros perdidos dentro da biblioteca, necessidade de no mínimo um operador trabalhando no balcão de atendimento, entre outros.

A tecnologia RFID apresenta excelente desempenho na análise dos fatores mencionados acima. As etiquetas podem ser fixadas dentro das capas dos livros, dificultado a manipulação por um usuário hostil, o que aumenta consideravelmente a vida útil das mesmas. Livros perdidos, ou fora da estante correta, permanecem perdidos por até 1 ano a espera de um inventário na biblioteca, que ocorre somente uma vez por ano, dado o grau de dificuldade da realização e da demanda de aproximadamente 30 dias para revisão completa do acervo, efetuada de forma totalmente manual. Já o sistema RFID apresenta a versatilidade da utilização de leitores portáteis que viabilizam o inventário da biblioteca em poucas horas, além de ser utilizado para uma rápida localização de livros perdidos.

A análise do tempo de atendimento e performance do sistema foi avaliada neste trabalho através da modelagem em redes de Petri, que proporciona uma abordagem de excelente nível, pois os modelos construídos são estocásticos, isto é, utilizam ferramentas estatísticas para definição de eventos. Os resultados foram extremamente favoráveis à

utilização da tecnologia RFID. Além de não exigir a presença de um funcionário no balcão de atendimento para realizar empréstimos ou devoluções.

O projeto apresentado reduz os tempos de execução destas operações a quase zero, extinguindo as filas nos balcões. Outra característica é a versatilidade da incorporação de um sistema de segurança antifurto, já que um sistema de alarme pode soar quando um livro deixa o campo de leitura sem que uma carteira de usuário tenha sido detectada. Uma biblioteca que implementa um sistema deste tipo pode, facilmente, funcionar 24 horas por dia, necessitando apenas de um segurança, para garantir a boa fé de seus usuários, e de um operador para realizar a disposição dos livros nas estantes, como já é realizado atualmente.

BIBLIOGRAFIA

1. AUTO-ID CENTER. Technical Manual. **The Object Name Service**. Versão 0.5 (beta), 2002. Disponível em: < http://www.autoidlabs.org/>.

2. AUTO-ID CENTER. Technology Guide. **How the Auto-ID System Will Automate the Supply Chain**, 2002. Disponível em: < http://www.autoidlabs.org/>.

3. CARDOSO, Janette. VALETTE, Robert. **Redes de Petri.** Primeira Edição. Florianópolis: Ed. Da UFSC, 1997.

4. DEITEL, H. M., DEITEL, P.J.. **Java**: Como Programar. Sexta Edição. São Paulo: Prentice Hall, 2005.

5. FKI LOGISTEX. RFID for the Real Word: Challenges and Opportunities in the Warehouse and Distribution Center Environment, 2005. Disponível em: http://www.fkilogistex.com>.

6. MIYAGI, Paulo Eigi. **Controle programável**: fundamentos do controle de sistemas e eventos discretos. Primeira Edição. São Paulo: Edgard Blücher, 1996.

7. MULLER, Robert J. **Projeto de Banco de Dados**: Usando UML para modelagem de dados. Primeira Edição. São Paulo: Berkeley, 2002.

8. ORACLE. **List of Products A-Z**. Disponível em: <http://www.oracle.com/products/product_list.html>. Acessado em 25 de Julho de 2005.

9. RFID JOURNAL. **Auto-ID Savant specification 1.0**. Disponível em: <http://www.rfidjournal.com/whitepapers/downloads>. Acessado em 15 de Junho de 2005.

10. RFID JOURNAL. **The Savant Version 0.1 (alpha)**. Disponível em: <http://www.rfidjournal.com/whitepapers/downloads>. Acessado em 15 de Junho de 2005.

11. SCHARFELD, T. A. **An analysis of the fundamental Constraints on the Low cost Passive Radio Frequency Identification System Design.** Requirements for the Degree of Master of Science. Massachusetts Institute of Technology. Massachusetts, Agosto, 2001.

12. SERWAY, Raymond A. **Física 3**: para cientistas e engenheiros com física moderna. Terceira Edição. Rio de Janeiro: Livros Técnicos e Científicos, 1996.

13. SUN MICROSYSTEMS. **Administration Guide:** Sun Java Systems RFID Software 2.0, 2005. Versão 1. Disponível em: < http://docs.sun.com>.

14. SUN MICROSYSTEMS. **Developer's Guide:** Sun Java Systems RFID Software 2.0, 2005. Versão 1. Disponível em: < http://docs.sun.com>.

15. SUN MICROSYSTEMS. **Getting Started Guide**: Sun One Application Server 7, 2003. Versão 7. Disponível em: <http://docs.sun.com>.

16. SUN MICROSYSTEMS. **Installation Guide:** Sun Java Systems RFID Software 2.0, 2005. Versão 1. Disponível em: < http://docs.sun.com>.

17. SUN MICROSYSTEMS. **Sun Java Sustem RFID Software**. Disponível em: <http://www.sun.com/software/products/rfid/index.xml>. Acessado em 25 de Julho de 2005.

18. UNIVERSIDADE DE SÃO PAULO. Anuário Estatístico. Tabelas. Sistema integrado de bibliotecas. **Acervo das bibliotecas da USP Distribuídos por unidade,** 2004. Disponível em: < http://sistemas.usp.br/anuario/tabelas/T05_01.pdf >.

19. UNIVERSIDADE DE SÃO PAULO. Anuário Estatístico. Tabelas. Sistema integrado de bibliotecas. **Atendimento aos usuários (horário) pelas bibliotecas da USP,** 2004. Disponível em: <http://sistemas.usp.br/anuario/tabelas/T05_02.pdf>.

20. UNIVERSIDADE DE SÃO PAULO. Anuário Estatístico. Tabelas. Sistema integrado de bibliotecas. **Circulação do acervo das bibliotecas da USP**, 2004. Disponível em: <http://sistemas.usp.br/anuario/tabelas/T05_03.pdf>.

21. UNIVERSIDADE DE SÃO PAULO. Sistema Integrado de Bibliotecas. Departamento Técnico. **SIBiNet – Rede de Serviços do SIBi/USP**: características, usos e funções. São Paulo: SIBi/USP, 1999.

22. UNIVERSIDADE DE SÃO PAULO. Anuário Estatístico. Tabelas. Sistema integrado de bibliotecas. **Usuários das bibliotecas da USP, Distribuídos por Unidade**, 2004. Disponível em: <http://sistemas.usp.br/anuario/tabelas/T05_04.pdf>.

23. THE ASSOCIATION FOR AUTOMATIC IDENTIFICATION. Shrouds of Time. **The History of RFID**, 2001. Disponível em: <http://www.aimglobal.org>.

24. 3M. **Sistemas para bibliotecas**. Disponível em: <http://www.3m.com/intl/br/bibliotecas/>. Acessado em 25 de Julho de 2005.

APÊNDICE A – CLASSE THREADSTART

```
import java.net.*;
import java.io.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class ThreadStart extends Thread {
    public int status = 0;//cria variÃ¡vel status para sincronia
    public int indice = 0;
    private StringBuffer buffer;//cria o buffer

    //Contrutor de ThreadStart
    public ThreadStart(StringBuffer sb) {
        buffer = sb;
    }

    public String[] getEPC(){
        int j = 2;
        status=0;//dados enviados
        String resposta;
        String epc[] = new String[15];
        epc[0]="Buffer vazio"; //se o buffer estiver vazio
        Pattern expression = Pattern.compile("urn:epc:id:gid:.......?");//cria
Pattern para extraÃ§Ã£o dos nÃºmeros EPC do PML
        Pattern command = Pattern.compile("<Command>......");
        if(buffer.length()==0) //se o buffer estiver vazio
        return (epc);
        else
        {//busca os numeros EPC e os armazenam em um array
            String dados;
            indice = buffer.indexOf("</Sensor>")+9;
            dados = buffer.substring(0,indice);
            buffer.delete(0,indice);
            epc[0]=dados.substring(dados.indexOf("Tags"),dados.indexOf("Tags")+7);

epc[1]=dados.substring(dados.indexOf("<DateTime>"),dados.indexOf("<DateTime>")+29);
            Matcher matcher = expression.matcher(dados);
            while(matcher.find()){
            epc[j]=matcher.group();
            j++;
            }
            return (epc);
        }
    }

    public void run(){ //thread

        try{
        Socket s = new Socket("localhost", 9001);//cria o socket
        BufferedInputStream bis = new BufferedInputStream(s.getInputStream());
        System.out.println("Socket Connected!");
        while(true){//grava dados no buffer e informa que o buffer estÃ¡ carregado
            char ch = (char)bis.read();
            buffer.append(ch);
            status = 1;

        }

    }
    catch(Exception e){
        System.out.println(e.toString());
    }
}
}
```

APÊNDICE B – CLASSE TESTE

```
import java.sql.*;
public class teste {

    static final String JDBC_DRIVER = "org.postgresql.Driver"; //determina driver
JDBC
    static final String DATABASE_URL = "jdbc:postgresql://localhost/biblioteca";
//determina url do JDBC

    public static void main(String args[]){
        Connection connection = null; //Inicializa connection e statement
        Statement statement = null;
        try{
            Class.forName(JDBC_DRIVER); //Conecta ao BD
            connection = DriverManager.getConnection(DATABASE_URL, "postgres",
"postgres");
            statement = connection.createStatement();


        StringBuffer dw = new StringBuffer(100); //Construtor do buffer
        ThreadStart ts = new ThreadStart(dw); // Construtor da ThreadStart
        ts.start(); //Inicia a Thread

        while(true){  //Inicia loop infinito
        while(ts.status==0){ //aguarda envio de dados do leitor
        }
        String epc[] = ts.getEPC(); //busca os dados e os guarda em um array
        String aluno = null; //inicia a variÃ¡vel que identifica dono da
carteirinha
        if(epc[0].equals("TagsIn<")){ //Se Ã© um evento de entrada de etiquetas
        for(int i=0; i<=12;i++){
        ResultSet resultSet = statement.executeQuery("SELECT * FROM alunos WHERE id
= '"+epc[i]+"'");//procura o aluno no banco de dados
        if(resultSet.next())
           aluno=epc[i];

        }//fim do for

        for(int i=3; i<=11;i++){
         ResultSet resultSet1 = statement.executeQuery(" SELECT * FROM emprestimos
WHERE data_d='aberto' and epc='"+epc[i]+"'");//determina se Ã© emprestimo ou
devoluÃ§Ã£o


         if(resultSet1.next()){
        statement.executeUpdate("UPDATE emprestimos SET data_d='"+epc[1]+"' WHERE
epc='"+epc[i]+"'"); //Se for devoluÃ§Ã£o armazena data de devoluÃ§Ã£o
        System.out.printf("\n%s\n", "Etiquetas lidas!");}
        else{
           statement.executeUpdate("INSERT INTO emprestimos (id, epc, data_e,
data_d) " + " VALUES ('"+aluno+"','"+epc[i]+"','"+epc[1]+"', 'aberto')");//Se for
emprÃ©stimo armazena dados
        System.out.printf("\n%s\n", "Etiquetas lidas!");}
        }
        }

        }
        }  // fim do try
      catch (SQLException sqlException)
      {
         sqlException.printStackTrace();
         System.exit( 1 );
      } // fim do catch
      catch (ClassNotFoundException classNotFound)
      {
```

```
            classNotFound.printStackTrace();
            System.exit( 1 );
         } // fim do catch
         finally // assegura que a instruÃ$Ã£o e conexÃ£o sÃ£o fechadas adequadamente
         {
            try
            {
               statement.close();
               connection.close();
            } // fim do try
            catch ( Exception exception )
            {
               exception.printStackTrace();
               System.exit( 1 );
            } // fim do catch
         } // fim do finally

      }
      }
```

APÊNDICE C – CLASSE RESULTS

```
import java.sql.Connection;
import java.sql.Statement;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import javax.swing.table.AbstractTableModel;


public class Results extends AbstractTableModel
{
   private Connection connection;
   private Statement statement;
   private ResultSet resultSet;
   private ResultSetMetaData metaData;
   private int numberOfRows;

   // monitora o status da conexÃ£o de banco de dados
   private boolean connectedToDatabase = false;

   // construtor inicializa Results e obtÃ©m seu objeto de metadados;
   // determina nÃºmero de linhas
   public Results( String driver, String url,
      String username, String password, String query )
      throws SQLException, ClassNotFoundException
   {
      // carrega classe de driver do banco de dados
      Class.forName( driver );

      // conecta-se ao banco de dados
      connection = DriverManager.getConnection( url, username, password );

      // cria Statement para consultar banco de dados
      statement = connection.createStatement(
         ResultSet.TYPE_SCROLL_INSENSITIVE,
         ResultSet.CONCUR_READ_ONLY );

      // atualiza status de conexÃ£o de banco de dados
      connectedToDatabase = true;

      // configura consulta e a executa
      setQuery( query );
   } // fim do construtor ResultSetTableModel

   // obtÃ©m a classe que representa o tipo de coluna
   public Class getColumnClass( int column )throws IllegalStateException
   {
      // assegura que o banco de dados conexÃ£o estÃ¡ disponÃvel
      if ( !connectedToDatabase )
         throw new IllegalStateException( "Not Connected to Database" );

      // determina a classe Java de coluna
      try
      {
         String className = metaData.getColumnClassName( column + 1 );

         // retorna objeto Class que representa className
         return Class.forName( className );
      } // fim do try
      catch ( Exception exception )
      {
         exception.printStackTrace();
      } // fim do catch
```

```
      return Object.class; // se ocorrerem os problemas acima, assume tipo Object
   } // fim do mÃ©todo getColumnClass

   // obtÃ©m nÃºmero de colunas em ResultSet
   public int getColumnCount()throws IllegalStateException
   {
      // assegura que o banco de dados conexÃ£o estÃ¡ disponÃvel
      if ( !connectedToDatabase )
         throw new IllegalStateException( "Not Connected to Database" );

      // determina nÃºmero de colunas
      try
      {
         return metaData.getColumnCount();
      } // fim do try
      catch ( SQLException sqlException )
      {
         sqlException.printStackTrace();
      } // fim do catch

      return 0; // se ocorrerem os problemas acima, retorna 0 para o nÃºmero de
colunas
   } // fim do mÃ©todo getColumnCount

   // obtÃ©m nome de uma coluna particular em ResultSet
   public String getColumnName( int column )throws IllegalStateException
   {
      // assegura que o banco de dados conexÃ£o estÃ¡ disponÃvel
      if ( !connectedToDatabase )
         throw new IllegalStateException( "Not Connected to Database" );

      // determina o nome de coluna
      try
      {
         return metaData.getColumnName( column + 1 );
      } // fim do try
      catch ( SQLException sqlException )
      {
         sqlException.printStackTrace();
      } // fim do catch

      return ""; // se ocorrerem problemas, retorna string vazia para nome de
coluna
   } // fim do mÃ©todo getColumnName

   // retorna nÃºmero de linhas em Results
   public int getRowCount()throws IllegalStateException
   {
      // assegura que o banco de dados conexÃ£o estÃ¡ disponÃvel
      if ( !connectedToDatabase )
         throw new IllegalStateException( "Not Connected to Database" );

      return numberOfRows;
   } // fim do mÃ©todo getRowCount

   // obtÃ©m valor na linha e coluna particular
   public Object getValueAt( int row, int column )
      throws IllegalStateException
   {
      // assegura que o banco de dados conexÃ£o estÃ¡ disponÃvel
      if ( !connectedToDatabase )
         throw new IllegalStateException( "Not Connected to Database" );

      // obtÃ©m um valor na linha e coluna da Results especificada
      try
      {
         resultSet.absolute( row + 1 );
         return resultSet.getObject( column + 1 );
```

```
      } // fim do try
      catch ( SQLException sqlException )
      {
         sqlException.printStackTrace();
      } // fim do catch

      return ""; // se ocorrerem problemas, retorna objeto string vazio
   } // fim do mÃ©todo getValueAt

   // configura nova string de consulta de banco de dados
   public void setQuery( String query )
      throws SQLException, IllegalStateException
   {
      // assegura que o banco de dados conexÃ£o estÃ¡ disponÃvel
      if ( !connectedToDatabase )
         throw new IllegalStateException( "Not Connected to Database" );

      // especifica consulta e a executa
      resultSet = statement.executeQuery( query );

      // obtÃ©m metadados para Results
      metaData = resultSet.getMetaData();

      // determina o nÃºmero de linhas em Results
      resultSet.last();                   // move para a Ãºltima linha
      numberOfRows = resultSet.getRow();  // obtÃ©m nÃºmero de linha

      // notifica a JTable de que modelo foi alterado
      fireTableStructureChanged();
   } // fim do mÃ©todo setQuery

   // fecha Statement e Connection
   public void disconnectFromDatabase()
   {
      if ( !connectedToDatabase )
         return;

      // fecha Statement e Connection
      try
      {
         statement.close();
         connection.close();
      } // fim do try
      catch ( SQLException sqlException )
      {
         sqlException.printStackTrace();
      } // fim do catch
      finally  // atualiza status de conexÃ£o de banco de dados
      {
         connectedToDatabase = false;
      } // fim do finally
   } // fim do mÃ©todo disconnectFromDatabase
} // fim da classe Results
```

APÊNDICE D – CLASSE INTERFACE

```
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.sql.SQLException;
import javax.swing.JFrame;
import javax.swing.JTextArea;
import javax.swing.JScrollPane;
import javax.swing.ScrollPaneConstants;
import javax.swing.JTable;
import javax.swing.JOptionPane;
import javax.swing.JButton;
import javax.swing.Box;

public class Interface extends JFrame {

   // driver JDBC e URL de banco de dados
   static final String JDBC_DRIVER = "org.postgresql.Driver";
   static final String DATABASE_URL = "jdbc:postgresql://localhost/biblioteca";
   static final String USERNAME= "postgres";
   static final String PASSWORD= "postgres";

   // consulta padrÃ£o seleciona todas as linhas de tabela emprestimos
   static final String DEFAULT_QUERY = "SELECT * FROM emprestimos";

   private Results tableModel;
   private JTextArea queryArea;

   // cria a Interface e GUI
   public Interface()
   {
      super( "Displaying Query Results" );

      // cria a Interface e exibe tabela de banco de dados
      try
      {
         // cria o TableModel para resultados da consulta SELECT * FROM emprestimos
         tableModel = new Results( JDBC_DRIVER, DATABASE_URL,
            USERNAME, PASSWORD, DEFAULT_QUERY );

         // configura JTextArea em que o usuÃ¡rio digita consultas
         queryArea = new JTextArea( DEFAULT_QUERY, 3, 100 );
         queryArea.setWrapStyleWord( true );
         queryArea.setLineWrap( true );

         JScrollPane scrollPane = new JScrollPane( queryArea,
            ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED,
            ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER );

         // configura o JButton para enviar consulta
         JButton submitButton = new JButton( "Submit Query" );

         // cria o Box para gerenciar o posicionamento da queryArea e do
         // submitButton na GUI
         Box box = Box.createHorizontalBox();
         box.add( scrollPane );
         box.add( submitButton );

         // cria o delegado JTable para tableModel
         JTable resultTable = new JTable( tableModel );

         // posiciona os componentes GUI no painel de conteÃºdo
         add( box, BorderLayout.NORTH );
```

```
      add( new JScrollPane( resultTable ), BorderLayout.CENTER );

      // cria evento ouvinte para submitButton
      submitButton.addActionListener(

         new ActionListener()
         {
            // passa consulta para modelo de tabela
            public void actionPerformed( ActionEvent event )
            {
               // realiza uma nova consulta
               try
               {
                  tableModel.setQuery( queryArea.getText() );
               } // fim do try
               catch ( SQLException sqlException )
               {
                  JOptionPane.showMessageDialog( null,
                     sqlException.getMessage(), "Database error",
                     JOptionPane.ERROR_MESSAGE );

                  // tenta recuperar a partir da consulta de usuÃ¡rio invÃ¡lida
                  // executando consulta padrÃ£o
                  try
                  {
                     tableModel.setQuery( DEFAULT_QUERY );
                     queryArea.setText( DEFAULT_QUERY );
                  } // fim do try
                  catch ( SQLException sqlException2 )
                  {
                     JOptionPane.showMessageDialog( null,
                        sqlException2.getMessage(), "Database error",
                        JOptionPane.ERROR_MESSAGE );

                     // assegura que a conexÃ£o de banco de dados estÃ¡ fechada
                     tableModel.disconnectFromDatabase();

                     System.exit( 1 ); // termina o aplicativo
                  } // fim do catch interno
               } // fim do catch externo
            } // fim do mÃ©todo actionPerformed
         } // fim da classe ActionListener interna
      ); // fim da chamada para addActionListener

      setSize( 500, 250 ); // configura o tamanho da janela
      setVisible( true ); // exibe a janela
   } // fim do try
   catch ( ClassNotFoundException classNotFound )
   {
      JOptionPane.showMessageDialog( null,
         "MySQL driver not found", "Driver not found",
         JOptionPane.ERROR_MESSAGE );

      System.exit( 1 ); // termina o aplicativo
   } // fim do catch
   catch ( SQLException sqlException )
   {
      JOptionPane.showMessageDialog( null, sqlException.getMessage(),
         "Database error", JOptionPane.ERROR_MESSAGE );

      // assegura que a conexÃ£o de banco de dados estÃ¡ fechada
      tableModel.disconnectFromDatabase();

      System.exit( 1 );   // termina o aplicativo
   } // fim do catch

   // dispÃµe da janela quando o usuÃ¡rio fecha o aplicativo (isso sobrescreve
   // o padrÃ£o de HIDE_ON_CLOSE)
```

```
      setDefaultCloseOperation( DISPOSE_ON_CLOSE );

      // assegura que a conexÃ£o de banco de dados Ã© fechada quando usuÃ¡rio fecha
o aplicativo
      addWindowListener(

         new WindowAdapter()
         {
            // desconecta-se do banco de dados e sai quando a janela for fechada
            public void windowClosed( WindowEvent event )
            {
               tableModel.disconnectFromDatabase();
               System.exit( 0 );
            } // fim do mÃ©todo windowClosed
         } // fim da classe WindowAdapter interna
      ); // fim da chamada a addWindowListener
   } // fim do construtor DisplayQueryResults

   // executa o aplicativo
   public static void main( String args[] )
   {
      new Interface();
   } // fim de main
}
```