UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

CAMPUS LUIZ MENEGHEL

GUSTAVO DOS SANTOS MORATO

GERENCIAMENTO DE ESTOQUE DE PEÇAS DE REPOSIÇÃO UTILIZANDO AS CADEIAS DE MARKOV

Bandeirantes

2012

GUSTAVO DOS SANTOS MORATO

GERENCIAMENTO DE ESTOQUE DE PEÇAS DE REPOSIÇÃO UTILIZANDO AS CADEIAS DE MARKOV

Trabalho de Conclusão de Curso, apresentado ao curso de Sistemas de Informação da Universidade Estadual do Norte do Paraná – UENP, como requisito parcial para a obtenção do título de Licenciatura e Bacharelado em SISTEMA DE INFORMAÇÃO.

Orientador: Ms. Christian James de Castro Bussmann.

Bandeirantes

2012

GUSTAVO DOS SANTOS MORATO

GERENCIAMENTO DE ESTOQUE DE PEÇAS DE REPOSIÇÃO UTILIZANDO AS CADEIAS DE MARKOV

_____

Christian James de Castro Bussmann.
Universidade Estadual do Norte do Paraná – UENP.

_____

Rodrigo Tomaz Pagno.
Universidade Estadual do Norte do Paraná – UENP.

_____

Glauco Carlos Silva
Universidade Estadual do Norte do Paraná – UENP.

Bandeirantes,___de_____de 2012

AGRADECIMENTOS

Aos meus amigos que estiveram comigo ao decorrer desta caminhada especialmente a Paola, Guilherme, Fabio que me auxiliaram durante o curso e sempre apoiaram.

Ao meu orientador por ter me instruído de forma a desenvolver este trabalho da melhor maneira possível pela sua dedicação e consideração e tempo dedicados para a produção do mesmo.

Por último não menos importante a empresa Moreira & Pontes que permitiu que fosse observada sua rotina de produção e regras de negócios servindo assim de modelo para aplicação da teoria levantada neste trabalho.

Aluno é como uma coluna de concreto quanto mais ferro melhor fica....

RESUMO

Neste trabalho é apresentada uma ferramenta para auxiliar a realização de estimativas de produtos em estoque de forma a ajuda no gerenciamento de estoque de produtos de baixa demanda que pode variar de acordo com o tempo, o que implica em grandes dificuldades para se escolher uma política de gerenciamento que seja capaz de gerir de forma satisfatória esta demanda. A fim de validar o trabalho é analisado um estudo de caso de uma na empresa a qual possui dificuldades para gerir seu estoque devido estas variâncias. Para solucionar este problema foi desenvolvida uma ferramenta na linguagem de programação C#(Sharp) que utiliza de uma abordagem matemática a qual está embasada nas Cadeias de Markov. Pois a mesma permite fazer estimativas futuras do estoque desde que se tenha ou saiba a quantidade atual de elementos armazenados, sendo assim, para resolução das propriedades markoviana, foi utilizado o método de Poisson a qual permite calcular a probabilidade de ocorrência de um evento em um contexto em uma determinada fatia de tempo onde, os resultados obtidos podem ser salvos em um arquivo XML para futuras analises ou até mesmo a persistência em uma base de dados. Em seguida, após a implementação e implantação da ferramenta foi realizada uma análise dos benefícios proporcionados por ela.

Palavras-chave: Ferramenta. Gerenciamento. Estoque. Cadeias de Markov. Poisson.

ABSTRACT

This work presents a tool to help make estimates of products in stock in order to help manage product inventory of low demand, which may vary with time, which implies great difficulties to choose a policy management that is able to manage this demand satisfactorily. In order to validate the work we analyze a case study in a company which has difficulties to manage their inventory because these variances. To solve this problem we developed a tool in the programming language C # (Sharp) which uses a mathematical approach, which is based on Markov chains. Because it allows to estimate future stock since it has or knows the current number of elements stored, so for solving the Markov properties, we used the Poisson method which allows to calculate the probability of occurrence of an event in a context in a particular slice of time where the results can be saved in an XML file for further analysis or even persistence in a database. Then, after the implementation and deployment of the tool was performed an analysis of the benefits provided by it.

Keywords: Tool. Management. Stock. Markov chains. Poisson.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1. INTRODUÇÃO

A gestão de peças de reposição, diferentemente dos outros tipos de estoque, tem ganhado uma maior atenção do setor empresarial.

> Estes itens representam um custo bastante alto para muitas empresas, frequentemente excedendo o lucro anual. Uma típica empresa de manufatura mantém entre $5 milhões e $15 milhões de dólares em peças de reposição com um custo médio de oportunidade de 20% a 40% do valor de estoque (Sandvig & Allaire, 1998 apud Silva, 2009).

Neste contexto uma boa prática para gerir este tipo de estoque se faz necessário, haja vista que a carências de métricas, modelos e ferramentas para atender esta demanda tornam mais difíceis estabelecer uma metodologia de gestão destes itens, nos quais apresentam características de itens *slow-moving*[1]*e* itens de demanda intermitente ou *errática*[2] característica esta que imprimem uma maior dificuldade na escolha de um modelo que seja capaz de gerir este tipo de estoque.

O mau controle pode implicar em um alto custo de armazenamento ou em um déficit de peças fazendo com que o acordo de nível de serviço com o cliente não cumprido satisfatoriamente.

Segundo Gomes & Wanke(2008) uma forma bastante corriqueira de se manter o cliente satisfeito é a assistência pós-venda através de uma rápida prestação de serviço, sendo assim necessária uma determinada quantidade de peças em estoque para que seja possível atender o cliente o mais rápido possível.

Basicamente, as duas principais abordagens que são utilizadas para desenvolver um modelo de gestão de estoque de peças de reposição: (2001 apud SILVA, 2009).

1. Classificação, e;

[1]*Slow-moving:* itens de baixo rotatividade no estoque segundo (SILVA, 2009).

[2]Itens de demanda intermitente ou errática: itens com uma grande variabilidade nos padrões de demanda (SILVA, 2009).

2. Modelos matemáticos Huiskonen,

O modelo de classificação visa estratificar cada item de acordo com suas características, tais como: níveis críticos; demanda de consumo, e; fornecimento.

O estereótipo matemático visa programação linear, programação dinâmica, cálculos complexos, simulação e outros para gerir este tipo de estoque. Atualmente esta abordagem tem sido mais utilizada do que a citada anteriormente.

Segundo SILVA (2009) a abordagem por modelos matemáticos normalmente compreendem em duas partes a serem definidas:

1. Modelo para a previsão de demanda, e;
2. Modelo para demanda durante o *lead time*[3].

Haja vista que os modelos matemáticos são mais aceitos, para a concepção deste trabalho foi abordado um modelo matemático para o auxílio da previsão de demanda e reposição utilizando as Cadeias de Markov.

Ainda com o XML (Extensible Markup Language) auxiliará na comunicação da aplicação que será desenvolvida em C# (Linguagem de programação) com a base de dados já existente.

1.1 JUSTIFICATIVA

Segundo Favaretto e Drohomeretski (2011):

> O bom controle dos estoques de uma empresa é uma atividade essencial para sua competitividade. A falta de material em estoque

[3]*Lead time*: Lead time é o tempo de processamento de um pedido, desde o momento que é colocado na empresa até o momento em que o produto é entregue ao cliente.

> pode fazer com que o nível de serviço seja comprometido e clientes deixem de ser atendidos. Por outro lado, excesso de material em estoque traz problemas de fluxo de caixa, espaço e perdas por obsolescência. (FAVARETTO, DROHOMERESKI, p.1, 2011).

A perspectiva apresentada pelos autores evidencia a necessidade, das empresas, de se ter um controlador de estoque que consiga gerir este tipo de situação.

Huiskonen (2001, *apud* Silva 2009*)* argumenta que com tantos requisitos essenciais relacionados a estes tipos de itens, é natural que a gestão das peças de reposição se torne uma importante área de pesquisa dentro do controle de estoque.

A partir das observações e hipóteses criadas na pesquisa o desenvolvimento de um *software* para previsões de demanda de estoque pode auxiliar o "gerente" não só na tomada de decisão, mas, também no auxílio ao controle de estoques. O que implica em um melhor gerenciamento em um todo. Isto acaba impactando diretamente nos custos, pois uma boa gestão de estoque pode diminuir o custo da empresa tanto em manutenção do mesmo quanto a armazenamento.

Para o desenvolvimento deste *software,* foi necessário estabelecer alguns critérios, métodos ou uma métrica para que possa fazer tal controle, podendo assim trazer benefícios às empresas.

Para entender melhor esta questão de controle de estoque, faz-se necessário inicialmente uma breve explanação sobre o mesmo.

Sendo assim, para o desenvolvimento deste trabalho a definição utilizada para este será a mesma apresentada por Favaretto e Drohomeretski (2011), na qual argumentam que:

> Um sistema de controle de estoque registra todas as movimentações de entrada e saída de materiais, assim como acompanha o saldo deste para que seja feita a decisão de pedir ou não mais material (FAVARETTO, DROHOMERESKI, p.1, 2011).

Com base nesta definição foi desenvolvido um *software* que pretende atender estas ponderações apresentadas pelos autores. Tendo base para o desenvolvimento, um conceito matemático denominado “Cadeias de Markov” [4], este conceito será abordado mais adiante no texto, como prévia o conceito seria o estudo de processos estocásticos em estados discretos.

Para o desenvolvimento deste *software* foi utilizado à linguagem de programação C#, pois apresenta algumas características que são significativas, o destaque inicial pelo fato de poder ser desenvolvida em uma situação de *desktop*, e com potencial para ser um sistema via *web*.

Ao se referir a um controlador de estoque, intuitivamente, do ponto de vista computacional, está se referindo a construção de um banco de dados e para tal.

> É essencial notar que entre sua aplicação e a base de dados existe um terceiro elemento (na verdade podem existir vários). O que queremos dizer é que sua aplicação não “fala” diretamente com nenhuma base de dados. Ela fala com alguém que leva o “recado” para a base e traz uma resposta. Esse mensageiro, em software, é conhecido como middleware[5] (Reis & Lima p.195, 2002).

Neste sistema a linguagem de ligação que será utilizada para fazer a comunicação entre os “entes” será o XML, pois segundo Anderson et. AL (2002 *apud* Oliveira, 2004), esta é uma linguagem universal de marcação e também padrão utilizada para comunicação WEB.

Com as ponderações anteriormente citadas espera-se desenvolver um sistema de controle de estoque que possa ser mais “leve” com respostas mais rápidas, do que os sistemas atuais.

[4] Recebe este nome em homenagem ao matemático Russo **Andrei Andreyevich Markov** (1856 – 1922)

[5] Middleware – termo inglês que designa alguma coisa que se encontra entre outras duas.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O desenvolvimento deste trabalho tem como objetivo principal desenvolver um modelo que seja capaz de "facilitar" [6] o gerenciamento baseado nas cadeias de Markov, pois o estoque de peças de reposição tem como característica um baixo giro e varia muito na demanda de acordo com o tempo baseado.

1.2.2 Objetivos Específicos

Para alcançar o objetivo geral, será necessário, fazer estudos específicos sobre:

- Conceitos e aplicações das Cadeias de Markov, linguagem C#, linguagem de marcação XML e teorias relacionadas à demanda e reposição de estoque;
- Analisar o funcionamento do negócio da empresa onde o software ira residir;
- Desenvolver uma ferramenta para auxiliar o gerenciamento de estoque e;
- Implantar e analisar os benefícios apresentados pela ferramenta.

1.3 METODOLOGIA DA PESQUISA

Inicialmente apresentaremos a metodologia que foi utilizada para o desenvolvimento do trabalho posteriormente será apresentada a metodologia

[6] Para este trabalho o termo "facilitar" será entendido como algo que se compreende sem esforço, claro e compreensivo – Dicionário Houais eletrônico.

do desenvolvimento, sendo caracterizada pela forma de construção da ferramenta.

A metodologia a ser abordada neste trabalho é o estudo de caso, pois segundo (Tsikriktsis, et al., 2002) este tem sido uma das mais poderosas ferramentas para desenvolvimento de novas teorias na área de Gestão Operacional. Pode ser utilizado para diferentes propósitos de pesquisa tais como: exploração; construção de teoria; teste de teoria, e; refinamento de teoria.

De acordo com Bell (1989 *apud* Dias 2000) o estudo de caso é particularmente apropriado para pesquisadores individuais, pois dá a oportunidade para que um aspecto de um problema seja estudado em profundidade dentro de um período de tempo limitado.

Deste modo a escolha do estudo de caso como método de pesquisa foi fundamental, pois desta forma é possível acompanhar o *workflow* de uma determinada empresa em um determinado período de tempo e em paralelo possibilita a formulação de hipóteses para a solução do problema.

Ainda outro fator deste tipo de abordagem é que possibilidade de se observar o fenômeno de acordo com a sua ocorrência, e sem as interferências significativas do pesquisador, e facilita o entendimento dos processos de uma forma lógica, Lee (1989 *apud* Dias 2000).

> A dedução controlada ou lógica não necessita de análise matemática para comprovar sua validade. A matemática é um subconjunto da lógica formal, e não o contrário. As deduções lógicas por meio de proposições verbais são tão válidas quanto aquelas derivadas de proposições matemáticas.

De acordo com Lee (1989 *apud* Dias 2000) a dedução lógica é tão valida quanto às proposições matemáticas, este fato é de uma importância fundamental para o desenvolvimento da ferramenta, pois para inferir previsões futuras a mesma precisará realizar uma simulação de como o fluxo de produção e consumo estaria em um determinado período.

Segundo Hartley (1994 *apud* Dias 2000) o estudo de caso consiste em uma investigação minuciosa de uma organização ou mais, visando uma análise do contexto e dos processos envolvidos no fenômeno em estudo. E afirma que o fenômeno não está isolado de seu contexto (como nas pesquisas de laboratório), já que o interesse do pesquisador é justamente essa relação entre o fenômeno e seu contexto.

Sendo assim, para o desenvolvimento de boas práticas para a gerência de estoque, não pode ser tratada isolada dos outros processos que ocorrem na organização, desta forma possibilita analisar também o impacto dos demais processos sobre o processo em estudo.

2. REFERENCIAL TEÓRICO

Dentro deste capítulo serão apresentados os conceitos que embasam o desenvolvimento deste trabalho.

2.1 SGML

O SGML (Standard Generalized Markup Language**)** foi à linguagem que proporcionou o desenvolvimento do XML (*Extensible Markup Language*) e HTML (*HyperText Markup Language*) ainda é utilizada em sistemas onde há a necessidades de grandes especificações.

O SGML tem por objetivo proporcionar a criação de sistemas portáveis que não fossem independentes de sistema operacionais e nem de formatos de arquivo e promover o intercâmbio de documentos e sua respectiva manipulação a marcação seria feita para descrever a estrutura do documento tais com outros atributos invés de como o mesmo deve ser processado.

Em outras palavras, a marcação deste tipo não restringe o documento a formatações ou estilos padrões. O SGML foi uma evolução significativa em sistema de marcação procedural tais como TEX[7] (techenical) criado por Knuth para a preparação de textos matemática.

Em meados de 1986 foi adotada por diversas empresas de porte médio com o intuito de padronizar o intercâmbio de informações algumas linguagens de marcação que tiveram a SGML como base: são *docbook* projetada para a marcação de documentos técnicos e a TEI (*Text Encoding Inative*) para a marcação de textos literários.

Alguns pontos negativos nesta linguagem é que ela é muito grande e complexa sua especificação tem centenas de páginas e o fato de ser obrigatório o uso de um formato de estilo muitas vezes acaba por inviabilizar a sua utilização

[7] TEX: foi criado no final dos anos 70, por Donald Knuth na Universidade de Stanford, com o objetivo de gerar textos com excelente representação gráfica.

2.2 Linguagem de marcação

Segundo Marchal (2000 *apud* Oliveira, 2004), desde o princípio do conhecimento da escrita já se tinha a preocupação da estruturação de documentos para a demonstração posteriormente. Mesmo não tendo a maioria dos recursos existente hoje em dia, a marcação do texto era feita de uma forma diferente, por exemplo, símbolos sinais gráficos ou qualquer coisa que pudesse representar aquele documento para o entendimento posterior.

Estereótipos bons são editores de texto modernos que com poucos cliques qualquer pessoa é capaz de formatar um texto, ou seja, fazer sua marcação de acordo com seu desejo. Um problema dos editores mais antigos e quem deseja utilizá-lo teria que saber uma serie de comando para fazer a marcação, por outro lado, os editores atuais já não são precisos saber os comandos em contra partida o usuário não sabe como uma determinada marcação é feita no texto realmente, pois ele não vê esta parte funcional.

Outro tipo de marcação pouco notada, mas muito utilizada que são os conteúdos que estão dispostos na internet, à marcação deste tipo de conteúdo é realizada por meio de marcações denominada ***tag.***

Geralmente estes arquivos são lidos e interpretados por navegadores de Internet, os quais interpretam as marcações e definem a forma como um texto, uma imagem e qualquer objeto são representados PROFFITT & ZUPAN (2001).

O HTML é uma linguagem que utiliza as tag para realizar a marcação no texto, porém ele tem suas tag definidas previamente em contra partida o XML é oferecida uma forma mais ampla e simples de marcação, para que o usuário possa definir suas próprias marcações, ou seja, suas *tags* MOULTIS, KIRK, (2000 *apud* Oliveira 2004).

2.3 HTML

Segundo Moultis e Kirk (2000 *apud* Oliveira 2004), apesar de simples e limitada o HTML ainda é uma linguagem muito utilizada na internet apesar de outras linguagens mais complexas também realizarem a mesmas tarefas. O

HTML é derivada da linguagem pioneira de marcação SGML (*Standard Generalized Markup Languagem*) e foi criada por Tim Berners Lee.

Segundo Mouniltise e Kirk (2000 *apud* Oliveira 2004) um fato que tornou o HTML mais forte foi à criação de programas externos que resolvem problemas que o HTML não conseguiria que foi posteriormente chamado de *plug-ins* o ponto fraco nesta questão são os acessos dos diferentes conteúdo da internet onde um tipo específico de *plug-in* é necessário para cada conteúdo que o HTML não consegue lidar tendo de instalar diversos *plug-ins,* o que acarreta em uma perda de desempenho.

2.4 Surgimento do XML

Por volta dos anos 60 surgiu uma grande necessidade de compartilhamentos de documentos os mesmos deveriam ser compartilhados de forma a não perder suas marcações e ser possível visualizá-los em qualquer computador.

Neste intuito a IBM (International Business Machines) começou a trabalhar no desenvolvimento de uma linguagem denominada GML (Generelized Markup Languagem), porém não foi muito utilizada entre os desenvolvedores, mas mesmo assim a IBM continuou a dar suporte a este projeto durante anos.

Em meados de 1986 a ISO (Internacional Organization for Standardization) sugeriu um novo modelo que a IBM tinha desenvolvido agora com o nome de SGML (Standard Generalized Markup Languagem) a qual foi adotada como uma padrão de marcação para os diferentes documentos do mundo todo.

Segundo Anderson (et. al 2000 *apud* Oliveira, 2004) por volta da década de 90 quando a internet se tornou mais popular a SGML foi o cenário perfeito para o favorecimento da utilização da SGML na WEB como linguagem padrão, toda via a mesma era muito complexa de se trabalhar o que acabava muitas vezes inviabilizando sua utilização. Com este problema inerente proporcionou o

surgimento do HTML que nada mais seria uma aplicação da SGM*L* e apenas para representação visual de documentos.

Com a facilidade e simplicidade que esta nova marcação oferecia muitos desenvolvedores acabaram por adotá-la como padrão de visualização de conteúdo.

Os grandes volumes de usuário que utilizavam a internet continuam a crescer e surgiu uma nova necessidade de se transmitir não só documentos, mas agora vídeos e áudios, sendo assim se fizeram necessário à criação de um sistema mais padronizado e que pudesse suprir esta necessidade.

Tendo este problema em mente empresas tais como a Microsoft e Netscape começaram a desenvolver seus próprios recursos para sanar estes problemas deu-se assim a criação dos *plug-ins* que eram programas externos que resolviam problemas que o HTML não supria ou não suportava.

Contudo, para cada conteúdo que o HTML não podia lidar seria necessário um *plug-in* diferente tendo assim uma quantidade considerável de *plug-ins* sendo instalados fazendo com que o sistema perdesse muito seu desempenho deixando o inviável.

Sendo necessário criar algo que fosse mais flexível e expansível e que ao mesmo tempo oferecesse a mesma capacidade que a SGML, porém de uma maneira mais simples e que também fosse compatível com o HTML, visto que j a aceitação do mesmo foi mundial e se encontrava já nos computadores.

Todos estes fatos levaram a W3C (World Wide Web Consortium) a desenvolver uma linguagem de marcação que teria por dever suprir todas estas necessidades, que no momento, não tinham uma solução aceitável. Em meados de 1998 foram lançadas nova especificações de uma linguagem que foi denominada de XML.

O objetivo primordial desta nova forma de marcação era ser uma ferramenta que tivesse a mesma capacidade da SGML de fácil manuseio e mais objetivo. Sendo assim, os desenvolvedores poderiam criar seus programas de maneira mais fácil e posteriormente toda e qualquer informação gerada poderia ser representada em documentos flexíveis e de simples entendimento para os programas desenvolvidos em XML. Ray (2000).

Segundo Daum e Metter (2002 apud Oliveira 2004) o XML é considerado uma ferramenta de padronização uma vez que o usuário pode

definir suas próprias marcações, podendo delimitar o texto da melhor forma que lhe convém, desta forma o usuário normalmente escolhe palavras-chave do texto para definir suas *tags* de estruturação do conteúdo.

Outro fator que contribuiu foi que o XML trabalha com mais de um conjunto de caracteres diferentemente das outras linguagens que normalmente se trabalha com o padrão americano ASCII.

Com a utilização de diferentes conjuntos o XML proporciona que os desenvolvedores criem suas aplicações ou suas marcações usando da sua língua nativa mesmo que contenha caracteres especiais como se pode citar o grego tais como €€, β que poderiam ser usados para definir as tags normalmente e no documento seriam apresentados naturalmente.

O XML também é utilizado para o armazenamento de informações, uma vez que este possua características para a estruturação dos dados de forma hierárquica de forma simples e de fácil manuseio.

2.5 XML

Como descrito anteriormente é uma meta linguagem de marcação criada a partir da SGML por não possuir *tags* pré-definidas o XML é dito meta linguagem para descrição de linguagem de marcação, sendo assim, as *tags* podem ser criadas livremente de acordo com as necessidades dos desenvolvedores.

Segundo Moultis e Kirk (2000 *apud* Oliveira 2004) o XML não oferece mecanismo para a apresentação de documentos ficando a cargo de uma especificação da linguagem como o XSL (*Extensible Style Languagem)* sendo responsáveis por definir o estilo do documento a ser apresentado e XLink que representa a ligações entre os diferentes documentos.

Pode-se dizer que o XML é um tipo de marcação simples, sua estrutura de dados é rica proporciona a troca e exibição de conteúdos da base de dados e pode ser usada para a troca de informação entre aplicações.

Em 1998 o XML tornou-se um padrão internacional, sendo assim, várias outras linguagens de marcação foi seguindo o seu padrão como, por exemplo,

algumas adaptações realizadas no HTML 4.01 para suportar o XML deram a origem do XHTML.

Ao contrário do SGML o XML não precisa de DTD (*Document Type Definition*), que é uma espécie de dicionário para a aplicação que pretende lidar com o arquivo XML em questão garantindo sua estrutura de visualização.

Outro fato importante que um analista XML pode verificar se sintaxe está correta e facilitar o entendimento para os diferentes desenvolvedores. Se a gramática pré-definida no DTD é seguida, o documento em questão é dito como sendo válido, a adoção de uma forma simples de aninhamento dos elementos de um documento garante que a mesmo esteja bem formatado.

2.6. C# (SHARP)

Antes de apresentar os conceitos básicos de C# será apresentada algumas teorias que fundamentam este conceito.

*2.*6.1 DOTNET (. Net)

DOTNET foi desenvolvida pela Microsoft em meados do ano 2000 inicialmente tinha uma visão de ser uma grande iniciativa onde o software passaria a ser comercializado na Internet como uma forma de serviço.

Utilizando o XML e o Soap (*Simple Object Access Protocol*) como protocolo padrão com o objetivo de aumentar a competitividade das empresas nos setores de T.I.

De acordo com (NETO, 2007). Net é um ambiente em tempo de execução que torna mais fácil o desenvolvimento de programas e com uma maior agilidade. O DOTNET é composto de duas partes o *Common Languagem Runtime* (CLR) e *Framework Class Library* (FCL).

O *Common Languagem Runtime* (CLR) é um ambiente de execução *(runtime*) que dá suporte para as linguagens de programação habilitadas no. Net tais com o C#, J#, Asp; e .Net. O *runtime* é o ambiente que proporciona a execução dos programas desenvolvidos na plataforma .Net

Framework Class Library (FCL) é uma biblioteca onde contém todas as classes utilizadas para o desenvolvimento tais como: a estrutura e a interface que estão dispostos vários serviços destacando o acesso a variados tipos de fontes de dados, configuração de segurança e manipulação de objeto.

O surgimento DotNet se deu pela necessidade de se criar programas baseados na estrutura de três camada para internet sendo assim necessário acessar dados de diversas fontes distintas.

2.6.2 C# (SHARP)

C# é uma linguagem de programação orientada a objetos criada pela Microsoft, faz parte da sua plataforma. Net. A companhia baseou C# na linguagem C++ e Java.

2.6.3 História da linguagem

A linguagem C# foi criada junto com a arquitetura .NET, embora existam várias outras linguagens que suportam essa tecnologia (como VB. NET, C++, J#), C# é considerada a linguagem símbolo DotNET. Pois foi criada praticamente do zero para funcionar na nova plataforma, sem preocupações de compatibilidade com código de legado e o compilador C# foi o primeiro a ser desenvolvido.

2.6.4 Características

De certa forma, a linguagem de programação que mais diretamente reflete a plataforma .NET é a linguagem C#, sobre a qual todos os programas .NET executam. C# está de tal forma ligado a esta plataforma que não existe o conceito de código não gerenciado (*unmanaged code*) em C#.

Suas estruturas de dados primitivas são objetos que correspondem a tipos em .NET. A desalocação automática de memória por *garbage colletor* além de várias de suas abstrações tais como: classes, interfaces, delegados, e, exceções são nada mais que a exposição explícita de recursos do ambiente. NET.

Quando comparada com C e C++, a linguagem é restrita e melhorada de várias formas incluindo:

- Ponteiros e aritmética sem checagem, só podem ser utilizados em uma modalidade especial chamada modo inseguro (*unsafe mode*). Normalmente os acessos a objetos são realizados por meio de referências seguras, as quais não podem ser invalidadas e normalmente as operações aritméticas são checadas evitando a sobrecarga (*overflow*).

  Objetos não são liberados explicitamente, mas por meio de um processo de coleta de lixo (*garbage collector*) quando não há referências aos mesmos, prevendo assim referências inválidas.
- Destrutores não existem, o equivalente mais próximo é a *interface Disposable*, que juntamente com a construção *using block* permitem que recursos alocados por um objeto sejam liberados prontamente. Também existem finalizadores, mas em Java sua execução não é imediata.
- No Java não é permitida herança múltipla, mas uma classe pode implementar várias interfaces abstratas. O objetivo principal é simplificar o desenvolvimento do ambiente de execução. As únicas conversões implícitas por *default* são conversões seguras, tais como ampliação de inteiros e conversões de um tipo derivado para um tipo base.

Não existem conversões implícitas entre inteiros e variáveis lógicas ou enumerações. Não existem ponteiros nulos (*void pointers*) (apesar de referências para *object* serem parecidas), qualquer conversão implícita definida pelo usuário deve ser marcada explicitamente, diferentemente dos construtores de cópia de C++.

Apesar de C# ser frequentemente tido como similar ao Java existe diferenças entre estas linguagens de programação. Algumas funcionalidades são implementadas de forma diferentes tais como:

- O Java não implementa propriedades, mas permite a utilização de métodos *get* e *set.*
- O Java não implementa o *goto* como estrutura de controle.
- O C# apesar de ser pouco usual implementa o *goto*.

- O Java utiliza comentários *javadoc*.
- O C# utiliza comentários baseados em XML.

Ao final desta explanação pode-se concluir que ambos possuem recursos similares, sendo possível o programador escolher a linguagem com a qual atenderá melhor a sua necessidade.

2.7 CADEIA DE MARKOV

Segundo (Boldrini, et. al, 1980) muitos dos eventos que ocorrem tanto na natureza como na sociedade moderna podem ser estudados, como sendo fenômenos que passam de um estado inicial, para uma sequência de estados, onde está transição pode ocorrer seguindo certa probabilidade, denominada probabilidade de transição. Ainda, segundo os autores, esta probabilidade depende apenas do estado em que o fenômeno se encontra e do estado a seguir, este tipo de processo é denominado processo de Markov e sua sequência de Cadeias de Markov.

As cadeias de Markov foram criadas por Andrei Andreyevich Markov (1856 – 1922) que é particularmente lembrado pelo estudo de cadeias que hoje levam seu nome. Este trabalho deu início à teoria de processos estocásticos.

As cadeias de Markov estão intrinsecamente ligadas aos processos estocásticos, segundo (Júnior, et al., 2012)

> Um processo estocástico tem a propriedade markoviana se os estados anteriores do processo são irrelevantes para a predição dos próximos estados, desde que o estado atual seja conhecido (Júnior, et al., 2012)

Segundo Silva (2009)

> Um processo estocástico é definido como uma coleção de variáveis aleatórias $\{X_t\}$ indexadas por um parâmetro t pertencente a um conjunto T. X_t representa uma característica mensurável de interesse no tempo t, por exemplo, o nível de estoque de um produto particular no final da semana t. (SILVA, 2009, p.13).

Os processos estocásticos utilizados neste projeto são denominados processos Markovianos. Um processo é dito Markoviano se o estado futuro

depender somente do estado atual. Ou ainda se o espaço de estado é discreto[8] pode-se dizer que o modelo passa a ser uma Cadeia de Markov, desde que em termos formais, siga a seguinte propriedade:

$$\Pr\{X_{n+1}=j \mid X_n=i, X_{n-1}=i_{n-1}, \ldots X_1=i_1, X_0=i_0\} = P\{X_{n+1}=j \mid X_n=i\} = p_{ij}$$

Equação I.

Sendo p_{ij} a matriz de transição de estados possíveis obtida por meio das equações de Kolmogorov.

> As equações de Kolmogorov - Chapman fornece o cálculo de probabilidades de transição em n passos. Estas equações mostram que as probabilidades de transição de n etapas podem ser obtidas, recursivamente, a partir das probabilidades de transição de uma etapa (Júnior, et al., 2012)

$$P_{ij}^{(n+m)} = \sum_{k=0}^{\infty} P_{i,k}^{(n)} P_{k,j}^{m} \text{ para todo } n,m \geq 0 \text{ e todo } i,j$$

Equação II.

Definição 2.7.1:

Um processo aleatório de Markov é um processo que pode assumir estado $a_1, a_2, \ldots, a_r$, de tal modo que a probabilidade de transição de um estado a_j para um estado a_i seja p_{ij} (um número que só depende de a_j e a_i.

A matriz das probabilidade de transição (matriz estocástica) é dada por:

$$T = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1r} \\ p_{21} & p_{22} & \cdots & p_{2r} \\ \vdots & \vdots & & \vdots \\ p_{r1} & p_{r2} & \cdots & p_{rr} \end{bmatrix}$$

Equação III.

(Observe que $p_{ij} \geq 0$, e que a soma de cada coluna deve ser 1.).

O *vetor* de probabilidades é aquele cuja i-ésima linha dá a probabilidade de ocorrência do estado a_i após *n* transações:

[8] Discreto: enumerável

$$\begin{bmatrix} p_1^n \\ \vdots \\ p_r^n \end{bmatrix}$$

Equação IV.

Seguindo o raciocínio do exemplo anterior nota-se que após *n* passos,

$$\begin{bmatrix} p_1^n \\ \vdots \\ p_r^n \end{bmatrix} = T^n . \begin{bmatrix} p_1^{(1)} \\ \vdots \\ p_r^{(1)} \end{bmatrix}$$

Equação V.

Previsão em longo prazo: para podermos fazer previsão em longo prazo, a matriz T deve cumprir certas condições. Possibilitando assim a definição a seguir.

Definição 2.7.2:
Uma matriz das probabilidade de transição é regular se alguma de suas potências tem todos os elementos não nulos.

A importância da matriz regular para as previsões em longo prazo é dada pelo teorema abaixo:

Teorema: Se a matriz T_{r*r} das probabilidades de transição é regular então:

i. As potências T^n aproximam-se de uma matriz P, no sentido de que cada elemento de T^n aproxima-se do elemento correspondente em P.
Todas as colunas de P são iguais, sendo dadas por um vetor-coluna
..... $\mathrm{V} = \begin{bmatrix} p_1 \\ \vdots \\ p_r \end{bmatrix}$
Com $p_1 > 0, p_2 > 0, \ldots, p_r > 0$.

Equação VI.

ii. Para qualquer vetor de probabilidade inicial

$$V_1 = \begin{bmatrix} p_1^{(1)} \\ \vdots \\ p_r^{(1)} \end{bmatrix}$$

Equação VII.

O vetor de probabilidade $T^n V_1$ aproxime-se de V (dado no item anterior).

iii. O vetor V é o único vetor que satisfaz V = TV.

Este teorema permite a compreensão de que, se a matriz das probabilidade de transição é regular, então é possível prever em longo prazo e esta não depende das probabilidade inicial V_1.

Além disso o item (iv) indica como achar a probabilidade depois de um longo prazo. O processo utilizado para se encontrar o vetor final de probabilidade, usando o item (iv) corresponde à procura de autovetor associado ao autovalor de uma matriz T.

3. DESENVOLVIMENTO

3.1 Metodologia do Desenvolvimento

Este capítulo tende a facilitar o entendimento de como foi desenvolvido o trabalho, a figura 1 mostra em tópicos da fase de desenvolvimento:

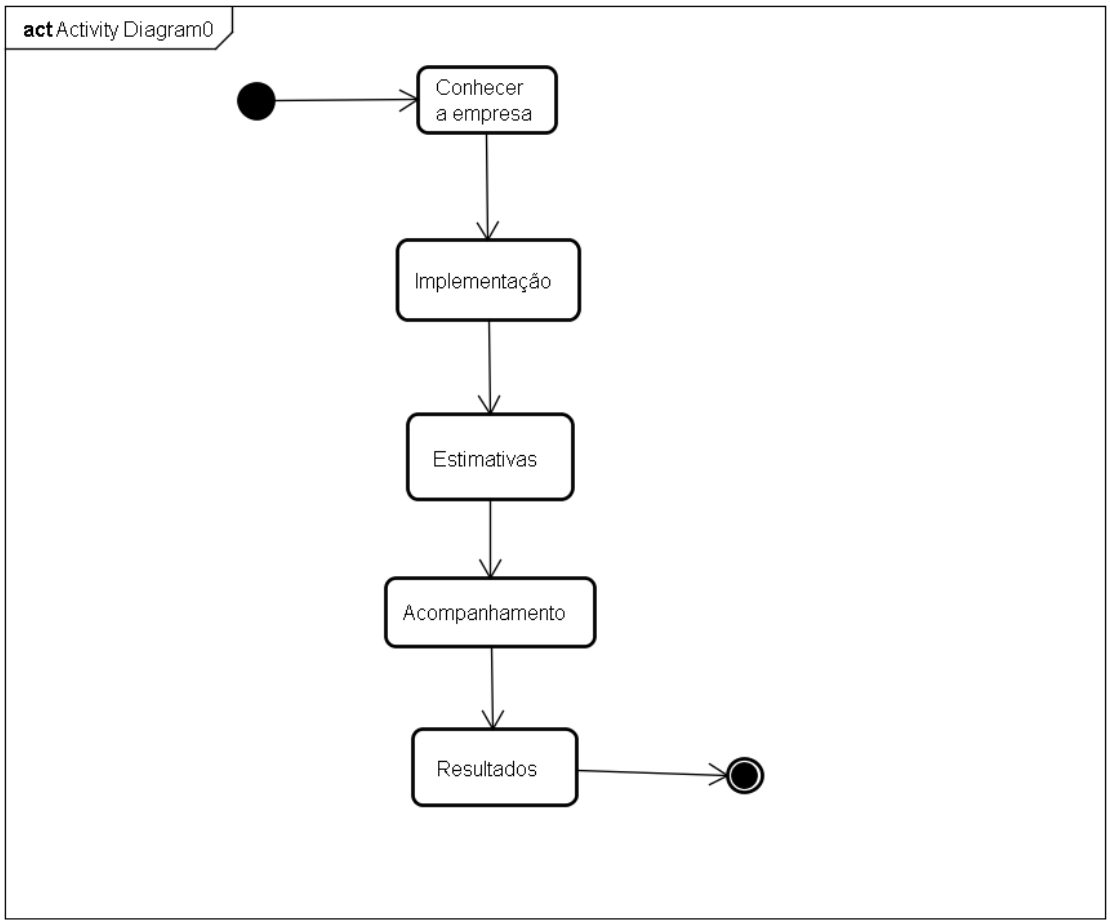

Figura 1: Diagrama de desenvolvimento

- **Conhecer a empresa**

  Esta fase inicial é para conhecer a empresa suas regras de negócio e como os processos realmente funcionam para que possam ser codificados.

- **Implementação**

  Esta é a fase de codificação da ferramenta onde todo conhecimento adquirido será utilizada para codificar da melhor maneira possível

- **Estimativas**

  Com a ferramenta já implementada e conhecendo as regras de negócio da empresa foram coletados alguns dados e foram feitas algumas estimativas
- **Acompanhamento**

  Nessa fase toda semana é anotado a quantidade de espumas, para que no final da sexta semana possa ser comparado com os resultados obtidos na fase de estimativa.
- **Resultados**

  Esta fase final do processo foi comparada as estimativas feita pela ferramenta, e as obtidas durante o período de acompanhamento, para analisar pontos positivos e negativos da ferramenta.

3.2 Desenvolvimento

Segundo Nogueira (2009) processos estocástico podem ser classificado como:

- **Processos de Estado Discreto:** A variável objeto pode assumir somente alguns valores discretos
- **Processos de Estado Contínuo:** A variável objeto pode assumir qualquer valor
- **Processos de Tempo Discreto:** *t* é uma variável que assume apenas valores discretos. O valor da variável objeto somente pode mudar apenas nestes pontos.
- **Processos de Tempo Contínuo:** *t* é uma variável contínua.

Desta forma sabendo que as Cadeias de Markov são processos estocástico, a escolha desta abordagem para gestão de estoque se dá pela característica deste tipo de estoque.

As regras de negócio da empresa delimita um conjunto de estados finitos o qual a matéria-prima pode assumir, e também delimita quando a matéria-prima deve muda de um estados para outro.

Para o presente estudo trabalhou-se com o fato de estoque de uma determinada empresa a qual trabalha com produtos de limpeza.

Para a produção dos seus insumos ocorra corretamente algumas regras de negócio[9] devem ser seguidas. Para o gerenciamento de estoque segue a seguinte regra de negócio:

- Número mínimo de peças em estoque é de 100 unidades
- Número máximo de peças em estoque é de 600 unidades

Quando o estoque chega ao mínimo é programada uma compra de matéria prima para que o estoque não fique zerado e a produção deve atender as demandas de pedidos de cada semana.

Atualmente sempre que um funcionário nota que as espumas estão acabando faz a contagem da quantidade que tem em estoque e repassa para a gerência com uma solicitação de compras

Neste contexto, destaca-se a necessidade para o desenvolvimento de uma ferramenta que calcula a probabilidade da demanda de acordo com o número de peças em estoque.

Para tal tarefa foi adotada as Cadeias de Markov, e a resolução da mesma deu-se pela distribuição de Poisson onde por definição tende a calcular a probabilidade de um acontecimento em um determinado contexto.

O que possibilita a criação de uma matriz de probabilidade denotada matriz de transição onde cada elemento representa a probabilidade de transição de um determinado nível de estoque para outro ou até mesmo permanecer no nível atual. Com a matriz montada a teoria de Kolmogorov possibilita as estimativas de demandas em longo prazo.

---

[9] Regras de negócio: “Regras do Negócio são declarações sobre a forma da empresa fazer negócio. Elas refletem políticas do negócio. Organizações têm políticas para satisfazer os objetivos do negócio, satisfazer clientes, fazer bom uso dos recursos, e obedecer às leis ou convenções gerais do negócio. LEITE & LEONARDI (1998) apud Dallavalle, et al.(2002).

3.3 FERRAMENTA

Tendo como base que a empresa já possui um sistema e que esta ferramenta seria um módulo deste sistema e que o mesmo seria responsável pelas funcionalidades de login e cadastro sendo a ferramenta responsável somente por calcular as probabilidade, o desenvolvimento da ferramenta foi baseado nos seguintes diagramadas:

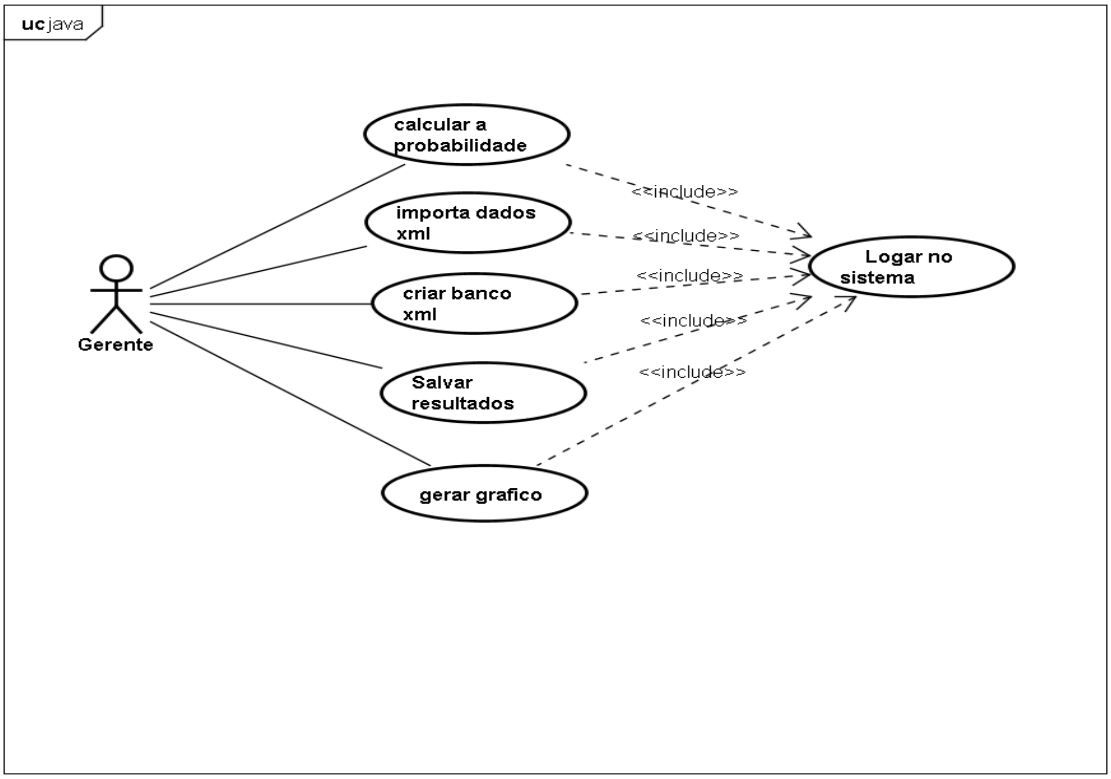

Figura 2: Diagrama de caso de uso.

Como apresentado na figura 2, as diferentes atividades da ferramenta só poderão ser acessadas pelo gerente ou outra pessoa com um nível de acesso superior ou igual, os gerentes são previamente cadastrados no sistema pelo chefe da organização.

Desta forma o primeiro contato com a ferramenta é necessário fazer o *login* no sistema da empresa para ter acesso às funcionalidades (esta tarefa é um pré-requisito), são permitidos usuários simultâneos utilizem a ferramenta, e o XML permite evitar a concorrência das tabelas da base de dados.

Caso o *login* esteja correto às funcionalidades da ferramenta serão ativadas, dentre elas, o gerente poderá criar uma base de dados em XML caso queira visualizar algum dado graficamente ou até mesmo importar uma base já existente.

Feito estes passos iniciais finalmente o usuário poderá fazer estimativas para o estoque que poderão ser visualizadas de forma numérica e gráfica, se necessário, os resultados obtidos poderão ser salvos.

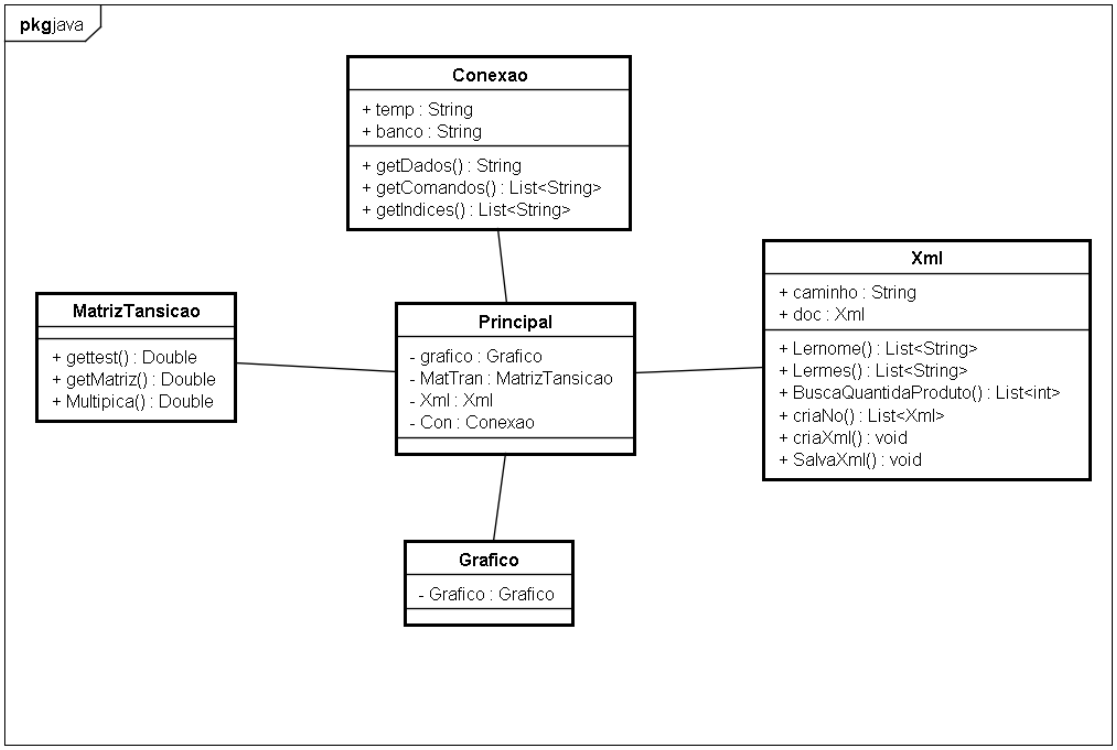

Figura 3: Diagrama de classe.

Segundo a figura 3 a classe principal contém uma instância da classe Poisson, Gráfico e XML.

A classe de MatrizTransicao possui o método “gettest()” que representa a média de ocorrência de um determinando evento em um determinado período de tempo, seguindo as regras de Poisson como mostra a equação viii.

$$f(k;\lambda) = \frac{e^{-\lambda}\lambda^{k}}{k!},$$

Equação VIII.

O método Multiplica() a partir da matriz de transição calcula as probabilidade de demandas futuras. De acordo com a seguinte definição de Kolmogorov:

> A matriz de transição “P” é uma matriz de transição de probabilidade de estado para um passo no tempo, ou seja, de t para t+1.

Em outras palavras de maneira bem simples as equações de Kolmogorov fornecem um método capaz de computar a matriz de transição em “n” passos no tempo ou seja de t para t+1, de t para t+2, de t para t+n de acordo com a seguinte equação matricial ix :

$$P_{ij}^{(n)} = \sum_{k=0}^{m} p_{ik}^{m}\, p_{kj}^{n-m}$$

Equação IX.

A classe gráficos é responsável por gerar gráficos de coordenadas cartesianas ou seja X e Y tanto das probabilidade de transição quanto dos elementos obtidos do banco de dados via arquivo XML.

A classe Conexao é responsável por realizar a comunicação com a base de dados com os três seguintes métodos:

1. getDados(): Responsável por retornar todos os itens cadastrados na tabela Produto.
2. getComandos(): Recebe por parâmetro um comando SQL (*Structured Query Language)*. E retorna uma lista de elementos.
3. getIndices(): Retorna o cabeçalho de cada tabela.

A classe XML que é responsável pela leitura do arquivo XML gerado do banco de dados ou também pode auxiliar na construção deste arquivo conectando no banco e gerando este arquivo (por motivos de estudo o SGBD (*Sistema de Gerenciamento de Banco de Dados*) utilizado será o Mysql) e também responsável por salvar os resultados obtidos.

Por fim a classe principal que contém uma instancia de cada classes. E responsável por executar todas as funcionalidades do sistema.

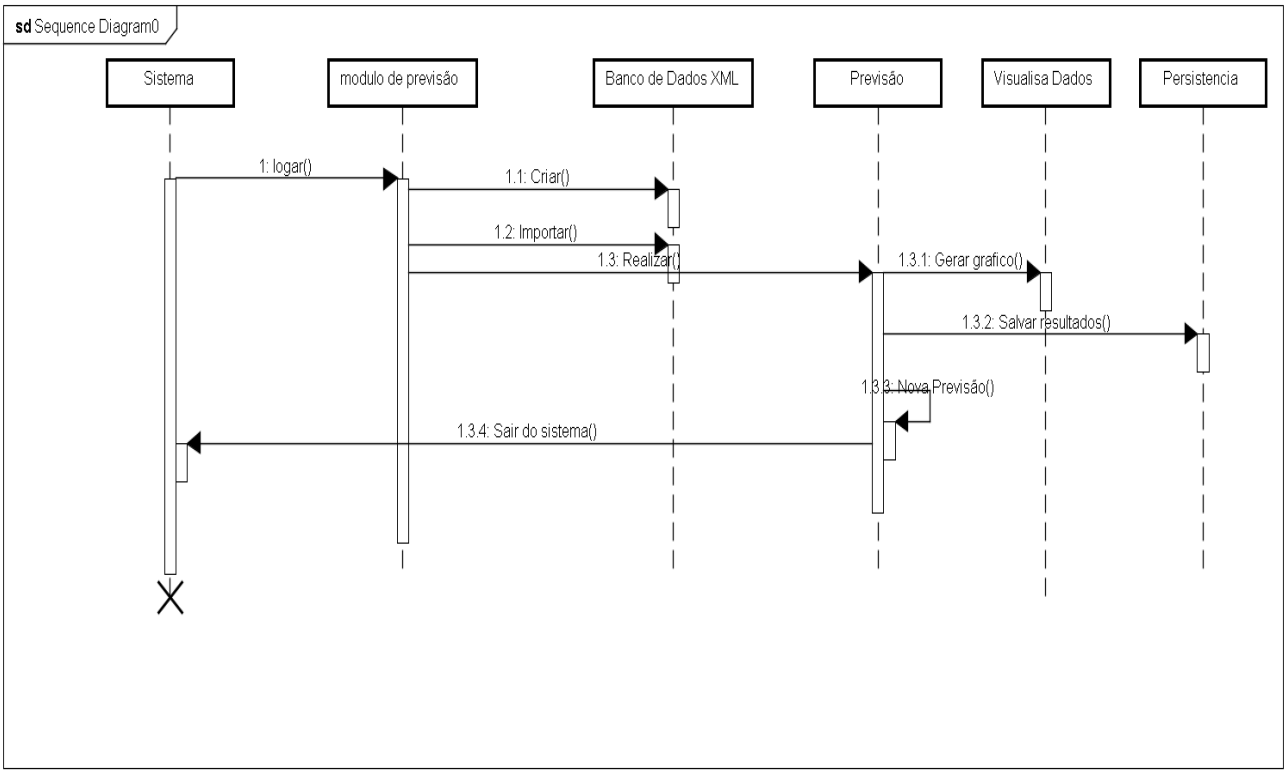

Figura 4: Diagrama de sequência.

No diagrama de sequência, figura 4, é mostrado a forma com que os processos ocorrem desde o início da atividade até o final do processo quando o módulo será desabilitado.

Desta forma seguindo o fluxo padrão o usurário loga no sistema (loga no sistema é um pré-requisito[10]) e criaria uma base de dados depois importar os dados para que a ferramenta possa trabalhar com diferentes dados vindos da base.

Em seguida é realizada as previsões que são estimativas feitas na forma de probabilidade dos dados que poderão ser salvas e gerar um gráfico, o usuário poderá realizar uma nova consulta ou finalizar o processo saindo do

[10] Pré-requisito: Condição indispensável e necessária antes de iniciar um processo.

sistema, como pode ser melhor visualizado no diagrama de atividades ilustrado na figura 4:

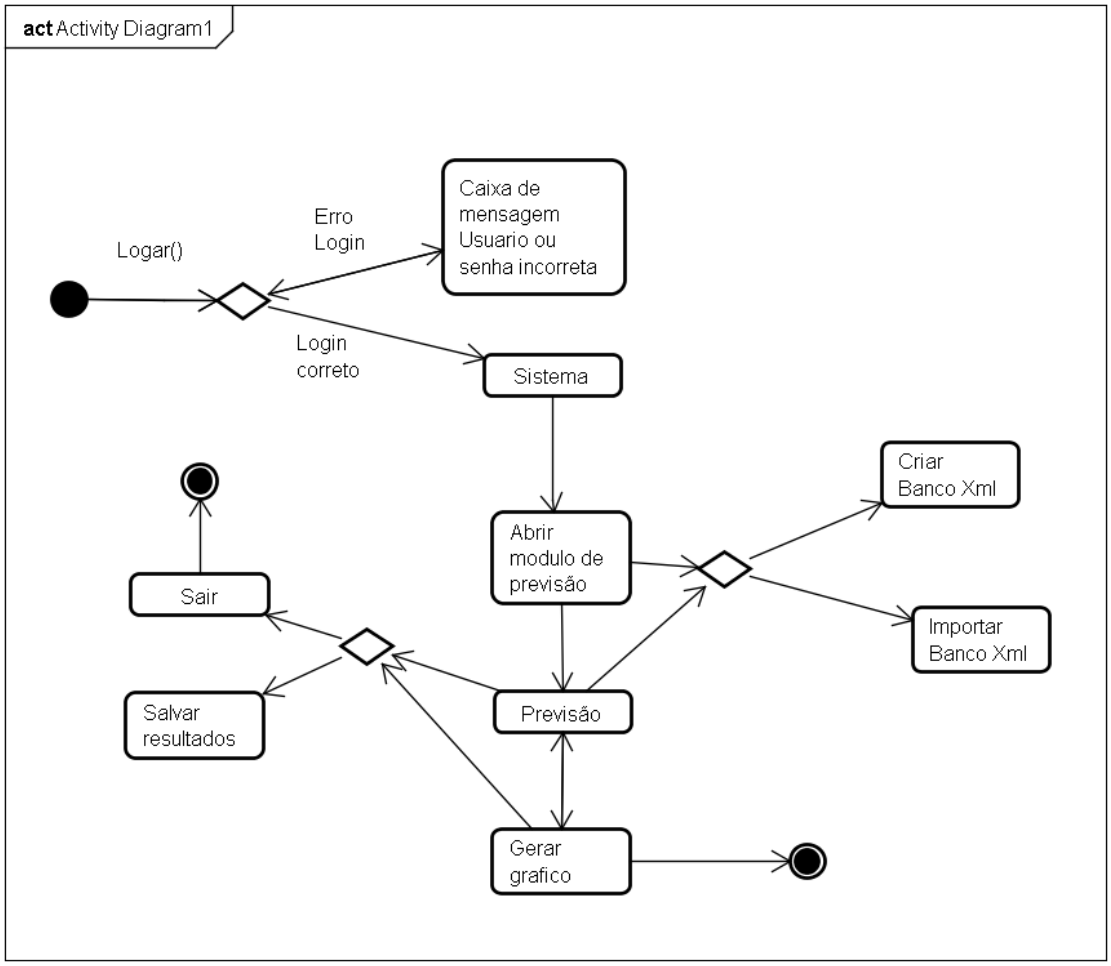

Figura 5: Diagrama de atividade.

Na interface apresentada na figura 6 contém uma caixa de texto onde será mostrada a matriz de transição (por motivos de estudos será mantida, mas em uma implementação futura poderá ser retirada), e ao lado esquerdo um *chart* onde serão plotados os gráficos.

Na parte superior o menu “Arquivo” conta com as funcionalidades de abrir um documento XML ou salvar criar uma base de dados XML (ressaltando que a ferramenta seria um módulo de um sistema que já estaria ligado à base

de dados) e a opção de carregar os dados de um arquivo XML no gráfico que será mostrado nas figuras abaixo.

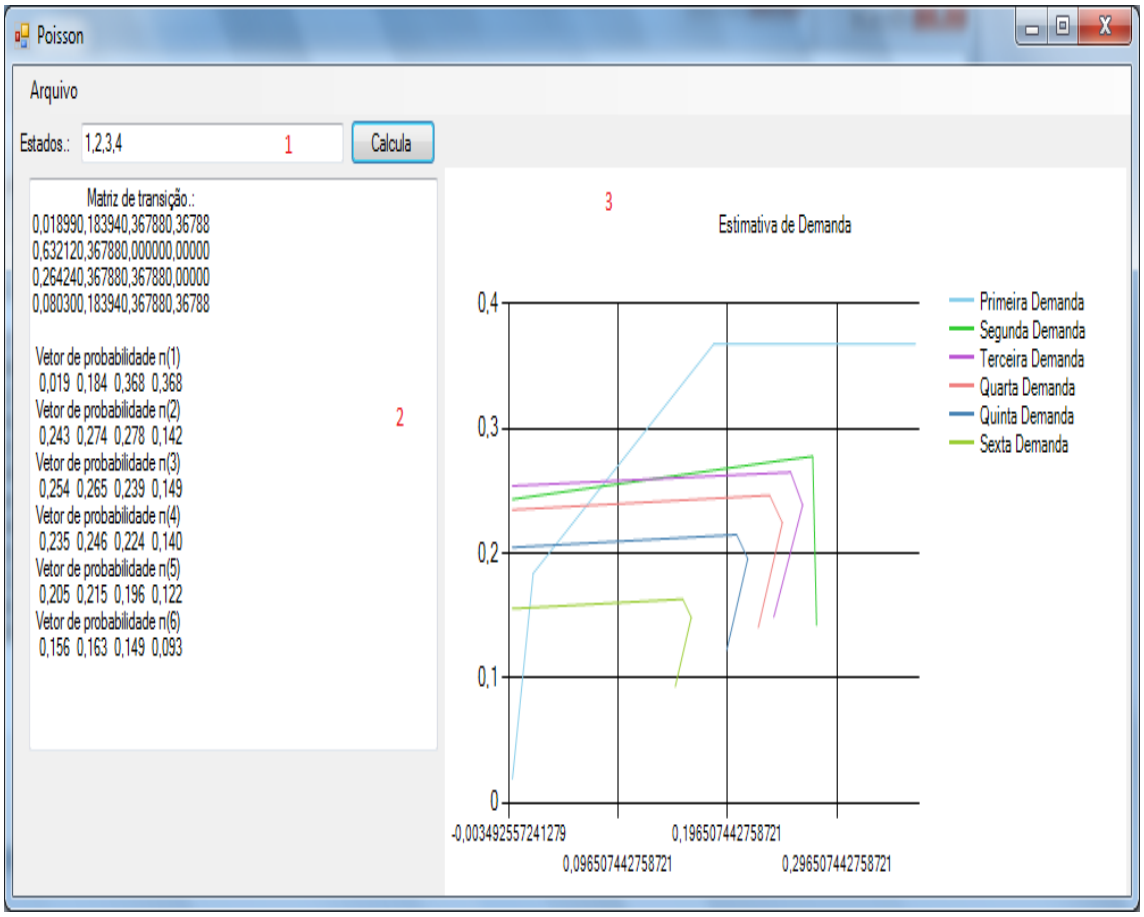

Figura 6: Tela principal da ferramenta.

1. Inserção de estados possíveis separado por vírgula.
2. Área onde será exibida a matriz de transição e os vetores de probabilidade.

3. Área onde será plotado o gráfico com as probabilidades de demanda.

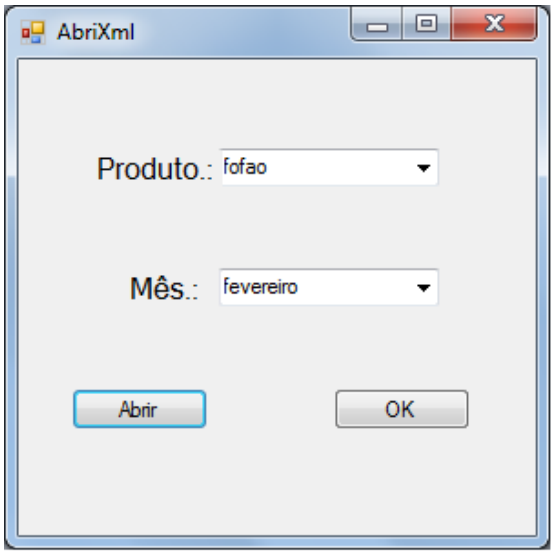

Figura 7: Tela de abertura de arquivo XML.

Depois de selecionado o arquivo XML a ser aberto é acionada a tela acima onde a *combobox* produto representa o nome do produto que se deseja ver a demanda e a *combobox* mês representa o mês da demanda do produto.

Caso a caixa de diálogo seja fechada o botão “Abrir” permite abri-la novamente ou caso o usuário queira selecionar um arquivo diferente, e por fim o botão “OK” que depois de selecionado o produto e o mês ele é responsável por acionar a execução da rotina de preenchimento do gráfico com as quantidades em estoque.

Considerando a rotina da empresa citada anteriormente, o valor de lambda será 100 e os estados possíveis serão 100, 200, 300, 400, 500 e 600 uma vez que o número máximo de espuma é 600 e o mínimo é 100.

Será desconsiderada a sazonalidade dos produtos e considerando que a empresa cumpre um *workflow* de 8 horas diárias e 5 dias durante a semana.

A empresa foi observada em um determinado período de tempo escolhido ao acaso para que não seja comprometida a integridade do referido

trabalho, foi utilizada a ferramenta para fazer uma previsão de demanda de seis semanas (considerando que o estoque de matéria prima está cheio).

Desta forma, os dados que a ferramenta gerou foram guardados para que fossem aferidos ao término das seis semanas que foi prevista, para tal tarefa foi feita a contagem do estoque de matéria prima de um determinado item ao término da semana (desconsiderando que um item pode ter uma demanda maior outros).

Ao final de seis semanas foi obtido o seguinte resultado da contagem de matéria-prima mostrado na tabela 1:

SEMANA	**QUANTIDADE EM ESTOQUE**
Primeira semana	510
Segunda semana	402
Terceira semana	293
Quarta semana	203
Quinta semana	129
Sexta semana	12

Tabela 1: Quantidade de espuma.

Na primeira e na segunda semana os resultados foram satisfatórios, pois, se mantiveram dentro dos parâmetros esperados, porém, na terceira semana, o nível esteve abaixo do esperado pois por questão da alta demanda de pedidos a empresa precisou sair do seu fluxo de trabalho normal ou padrão para conseguir atender ao pedidos, onde o jornada de trabalhos passou a ter duas horas a mais implicando em uma produção maior.

4.CONCLUSÃO

A ferramenta neste trabalho apresentada foi uma ferramenta para auxiliar o gerenciamento de estoque de produtos que possuem baixa rotatividade a ferramenta projetada com base nas regras de negócio praticadas na empresa na qual foi implantada e testada, a empresa que trabalha com a montagem de materiais de limpeza, e a ferramenta se mostrou capaz de fazer previsão em longo prazo, gerou os resultados apresentados na tabela abaixo.

Semanas	Quantidade esperada	Quantidades em estoque	Diferença
Primeira	600	510	90
Segunda	500	402	98
Terceira	400	293	107
Quarta	300	203	97
Quinta	200	129	71
Sexta	100	12	88

Tabela 2: Comparativo de quantidade de espuma esperada e obtida.

E ainda, auxiliou em outras questões, tais com a compra de matéria prima, pois esta não é fornecida na cidade onde está instalada a empresa, sempre que se faz um pedido existe a necessidade de realizar o transporte do mesmo, e tal situação é de responsabilidade da empresa que comprou o material.

Com as estimativas realizadas com o auxílio da ferramenta, a empresa passa a fazer compras agendadas e mais frequentes, pois tinha como fazer uma perspectiva de comprar e quanto comprar. Nesta perspectiva, houve uma redução com fretes, diminuindo os gastos da empresa. Bem como evitar a falta

de matéria prima na qual ocasiona na parada das atitudes de produção da empresa.

Com estes resultados, o que se espera é que haja um controle mais significativo do estoque, pois assim a empresa possa se manter competitiva no mercado. Com este controle acredita-se que possa diminuir significativamente o custo com reparo e manutenção do mesmo cooperando assim para um melhor rendimento.

Pois do contrário, um volume muito alto de matéria prima armazenada em estoque pode acarretar em prejuízo e em gastos**,** pois estes podem se deteriorar, bem como pode haver gastos com a manutenção do mesmo tais como: limpeza se for produto do gênero alimentício pode vencer dentre outros.

Implica quando na empresa acontece a falta de matéria-prima os funcionários recebem tarefas aleatórias pois não podem ficar sem produzir ou sem fazer nada.

Desta forma, o setor de produção apresentou-se mais padronizado e contínuo uma vez que a matéria-prima não faltará podendo assim ser melhor gerenciado.

As cadeias de Markov empregadas no trabalho foram um fator chave pois elas possibilitaram que a ferramenta fosse desenvolvida de forma genérica em outras palavras, de forma que possa ser empregado a mesma ferramenta em um setor diferente da empresas tal como:

- Setor de vendas: Pode ser utilizada para verificar as probabilidades de vendas e o desempenho de um vendedor.

Pois a ferramenta não foi desenvolvida de forma especifica para solucionar somente este problema.

Os dados gerados pelo sistema podem ser salvos em arquivo texto e ainda em XML caso os usuários queiram guardar os resultados para análise posterior em um SGBD

A utilização de uma base XML também foi um ganho pois, a ferramenta não tem que acessar o banco de dados a cada nova estimativa. Pois em empresas de grande porte a concorrência de acessos múltiplos às tabelas do banco de dados poderiam fazer com que a ferramenta trabalhe de forma mais lenta.

A resolução de Poisson também foi muito significativa para a geração das matrizes de transição já que foi trabalhada com cadeias de tempo contínuo e discreto.

Ressaltando que os processos de estimativa poderiam demorar mais e provavelmente teriam o resultados semelhantes, desta forma a implementação da distribuição de Poisson foi menos complexa e seu processamento muito mais leve.

5.CONSIDERAÇÕES FINAIS

Um ponto a ser observado é quando o fluxo de trabalho é alterado por algum motivo, neste caso por questão de alta demanda de pedido ou em épocas de sazonalidade, a ferramenta deixa a desejar, pois sua previsão fica imprecisa, como pode ser observando na tabela 2. Neste caso, previsões baseadas em correlações podem auxiliar nas estimativas.

E este tipo de previsão tende a ter uma melhor desenvolvimento no caso da sazonalidade e quando o workflow da empresa for alterado por algum motivo, sendo assim, a previsão de demanda seguindo os conceitos das cadeias de Markov com a resolução de Poisson se dá em contextos mais específicos onde não tem muita mudança no processo.

Podendo-se citar no caso das montadoras que seguem a questão da linha de montagem onde o fluxo de trabalho é continuo em um determinado período de tempo e a chances de sofrer variâncias no processo de produção é muito pequeno.

A regressão linear seria o complemento do referido trabalho, pois ela supriu a necessidade de previsão em momentos que as cadeias de Markov não são tão eficazes uma implementação híbrida seja algo interessante de se analisar como possível implementação futura.

E outro aspecto que poderia ser implementado futuramente seria a possibilidade de adicionar ou remover um estado e possibilita o cálculo de número maior de demandas.

REFERÊNCIAS

ARCOS, Lucas Gonçalves de Oliveira. **Construção de um sistema de Blackboard para gestão de documentos usando XML Lucas.** 2004. 78 f. Dissertação (Graduação) - Curso de Sistema de Informação, Universidade Católica de Minas Gerais, Arcos, 2004.

BOLDRI, José Luiz et al. **Álgebra linear.** São Paulo: Harba Ltda, 1980.

COSTA, Carlos Alberto. A APLICAÇÃO DA LINGUAGEM DE MODELAGEM UNIFICADA (UML) PARA O SUPORTE AO PROJETO DE SISTEMAS COMPUTACIONAIS DENTRO DE UM MODELO DE REFERÊNCIA. **Gestão & Produção**, Caxias do Sul - Rs, n. , p.19-36, 12 abr. 2001.

DIAS, Claudia. **Estudo de caso: ideias importantes e referências.** Rio de Janeiro: Altabooks, 2000.

EDWIN, Lima; EUGÊNIO, Reis. **C# E .NET ? GUIA DO DESENVOLVEDOR.** Rio de Janeiro: Campus Ltda, 2002.

FAVARETTO, Fabio. **ANÁLISE DE PROBLEMAS NO CONTROLE DE ESTOQUE DECORRENTES DE ERROS NOS REGISTROS DE SAÍDA.** 2011. 39 f. Dissertação (Graduação) - Universidade Federal De Itajubá, Itajubá, 2011.

GOMES, Antonio Vinicius Pimpão; WANKE, Peter. Modelagem da gestão de estoques de peças de reposição através de cadeias de Markov. **Gestão & Produção**, São Carlos, n. , p.57-72, 12 jan. 2008.

SILVA, Gerson Luís Caetano Da. **MODELO DE ESTOQUE PARA PEÇAS DE REPOSIÇÃO SUJEITAS À DEMANDA INTERMITENTE E LEAD TIME ESTOCÁSTICO.** 2009. 69 f. Dissertação (Graduação) - Curso de Engenharia de Produção, Universidade Federal de Minas Gerais, Belo Horizonte, 2009.

SOBRAL, Eryka Fernanda Miranda; SILVA JÚNIOR, Luiz Honorato da. **O ICMS Socioambiental de Pernambuco : Uma avaliação dos componentes socioeconômicos da política a partir do processo de Markov.** Pernanbuco: Mcb Up, 2012.

VOSS, Chris; TSIKRIKTSIS, Nikos; FROHLICH, Mark. **Case research in operations management.** Londres: Mcb Up Limited, 2001.

ZUPPAN, Ann; PROFFITT, Brian. **Xhtml Desenvolvimento Web.** São Paulo: Makron Books, 2001.

NOGUEIRA, Fernando. **Modelagem e simulação - Cadeias de Markov.** Juiz de Fora: Seventh Edition, 2009.

ANEXOS

ANEXO 1- CÓDIGO FONTE

Classe Conexao.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using MySql.Data.MySqlClient;

namespace tela_Poison
{
   class Conexao
   {
      string temp;
      string Banco = "server=localhost;userid=root;password=35425783;database=test";

      public void set(string server, string userid, string password, string database)
      {
         Banco = "server=" + server + ";userid=" + userid + ";password=" + password + ";database=" +
database;
         Console.Write(Banco);
      }
      public String getDados()
      {

         MySqlConnection conexao = new MySqlConnection(Banco);
         conexao.Open();

         MySqlCommand cmd = new MySqlCommand();
         cmd.Connection = conexao;
         cmd.CommandText = "SELECT * from produtos";

         MySqlDataReader resut = cmd.ExecuteReader();

         while (resut.Read())
         {
            temp = resut.GetString(0);
         }
         conexao.Close();
         return temp;

      }



      public List<string> getComando(String comam)
      {
         List<string> temp1 = new List<string>();
         MySqlConnection conexao = new MySqlConnection(Banco);
         conexao.Open();
         MySqlCommand cmd = new MySqlCommand();
```

```
            cmd.Connection = conexao;
            cmd.CommandText = comam;

            MySqlDataReader resut = cmd.ExecuteReader();
            Console.WriteLine(resut.FieldCount + "numero de fields");

            while (resut.Read())
            {
                for (int i = 0; i < resut.FieldCount; i++)
                {


                    temp1.Add(resut.GetString(i));

                }

            }
            conexao.Close();
            return (temp1);
        }


        public List<string> getIndice(String comam)
        {
            List<string> temp1 = new List<string>();
            MySqlConnection conexao = new MySqlConnection(Banco);
            conexao.Open();
            MySqlCommand cmd = new MySqlCommand();
            cmd.Connection = conexao;
            cmd.CommandText = comam;

            MySqlDataReader resut = cmd.ExecuteReader();
            Console.WriteLine(resut.FieldCount + "numero de fields");

            while (resut.Read())
            {


                temp1.Add(resut.GetString(0));


            }
            conexao.Close();
            return (temp1);
        }


    }
}
```

Classe Grafico.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
```

```
namespace tela_Poison
{
   class Grafico
   {
      public Grafico(Chart grafico)
      {

         string[] seriesArray = { "Primeira Demanda", "Segunda Demanda", "Terceira Demanda",
"Quarta Demanda", "Quinta Demanda", "Sexta Demanda" };
         int[] pointsArray = { 2 };

         // Set Cor .
         grafico.Palette = ChartColorPalette.Pastel;
         // Set titulo.
         grafico.Titles.Add("Estimativa de Demanda");
         // tamanho do grafico
         grafico.SetBounds(380, 50, 600, 700);
         ChartArea t = new ChartArea();

         grafico.ChartAreas.Add(t);

      }
   }

}
```

Classe MatrizTransicao.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace tela_Poison
{
   class MatrizTransicao
   {

      public static double gettest(double n)
      {
         if (n <= -1)
         {
            return 0;
         }

         else
         {

            Console.WriteLine("numero.: " + n + "    " + ((Math.Pow(1, n) * Math.Pow(2.71828, -1)) /
MathNet.Numerics.SpecialFunctions.Factorial(Convert.ToInt16(n))));
            return ((Math.Pow(1, n) * Math.Pow(2.71828, -1)) /
MathNet.Numerics.SpecialFunctions.Factorial(Convert.ToInt16(n)));

         }

      }



      public double[,] getMatriz(double NumMax, double NumMin, double[] estados)
      {
         double[,] matt = new double[estados.Length, estados.Length];
```

```
for (int i = 0; i < estados.Length; i++)
{

    for (int j = 0; j < estados.Length; j++)
    {
        matt[i, j] = new double();

        if (i == j && i == 0)
        {
            int n = Convert.ToInt16(NumMax) - 1; //recebe total em estoque
            double probabilidade = 0;
            while (n >= 0)
            {
                probabilidade = probabilidade + gettest(n);

                n--;

            }

            matt[i, j] = 1 - probabilidade;

        }
        else if (i < j && i == 0)
        {
            matt[i, j] = gettest((Convert.ToInt16(NumMax)) - estados[j]);


        }
        else if (i != 0 && j == 0)
        {

            int n = Convert.ToInt16(estados[i] - estados[j]) - 1; //recebe total em estoque
            double probabilidade = 0;
            while (n >= 0)
            {
                probabilidade = probabilidade + gettest(n);

                n--;

            }

            matt[i, j] = 1 - probabilidade;


        }
        else if (i != 0 && i == j)
        {
            matt[i, j] = gettest(estados[i] - estados[j]);


        }
        else
            if (i != 0 && i < j)
            {
                matt[i, j] = gettest(estados[i] - estados[j]);


            }
            else if (i != 0 && i > j)
            {
                matt[i, j] = gettest(estados[i] - estados[j]);


            }
```

```
            }
        }
        return matt;
    }

    public double[,] multiplica(double[,] matriz)
    {
        double[,] mattA = matriz;
        double[,] mattB = matriz;
        double[,] mattResultado = new double[matriz.GetLength(0), matriz.GetLength(0)];

        double aux;
        int row, column, i;



        for (row = 0; row < mattA.GetLength(0); row++) // multiplicação das matrizes
        {
            for (column = 0; column < mattResultado.GetLength(0); column++)
            {
                aux = 0;
                for (i = 0; i < mattA.GetLength(0); i++)
                {
                    aux = aux + mattA[row, i] * mattB[i, column];

                }
                mattResultado[row, column] = aux;

            }
        }
        return mattResultado;
    }
  }
}
```

Classe XML.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Xml;
using System.IO;



namespace tela_Poison
{
   class Xlm
   {


      string caminho = "Artigo_XML.xml";
      XmlDocument doc = new XmlDocument();

      public List<string> Lernome(string cam)
      {
         string sCaminhoDoArquivo = cam;
         List<string> lista = new List<string>();
```

```
    XmlDocument xmlDoc = new XmlDocument();
    xmlDoc.Load(sCaminhoDoArquivo); //Carregando o arquivo

    //Pegando elemento pelo nome da TAG
    XmlNodeList xnList = xmlDoc.GetElementsByTagName("produtos");


    //Usando foreach para imprimir na tela
    foreach (XmlNode xn in xnList)
    {
        string sNome = xn["nome"].InnerText;

        if (!lista.Contains(sNome))
        {
            lista.Add(sNome);
        }



    }
    return lista;
}



public List<string> Lermes(string cam)
{

    string sCaminhoDoArquivo = cam;
    List<string> lista = new List<string>();

    XmlDocument xmlDoc = new XmlDocument();
    xmlDoc.Load(sCaminhoDoArquivo); //Carregando o arquivo

    //Pegando elemento pelo nome da TAG
    XmlNodeList xnList = xmlDoc.GetElementsByTagName("produtos");


    //Usando foreach para imprimir na tela
    foreach (XmlNode xn in xnList)
    {

        string sMes = xn["mes"].InnerText;
        if (!lista.Contains(sMes))
        {
            lista.Add(sMes);
        }
    }
    return lista;
}



public List<int> BuscaQuantidadeProduto(string cam,string mes, string nome)
{

    string sCaminhoDoArquivo = cam;
    List<int> lista = new List<int>();

    XmlDocument xmlDoc = new XmlDocument();
    xmlDoc.Load(sCaminhoDoArquivo); //Carregando o arquivo

    //Pegando elemento pelo nome da TAG
    XmlNodeList xnList = xmlDoc.GetElementsByTagName("produtos");
```

```
    //Usando foreach para imprimir na tela
    foreach (XmlNode xn in xnList)
    {
        string sNome = xn["nome"].InnerText;
        string smes = xn["mes"].InnerText;
        if (sNome == nome && smes == mes)
        {
            string squantidade = xn["quantidade"].InnerText;
            lista.Add(Convert.ToInt16(squantidade));
        }

    }
    return lista;
}


public List<XmlNode> criaNo(List<string> indices)
{
    List<XmlNode> indice = new List<XmlNode>();
    for (int i = 0; i < indices.Count; i++)
    {
        XmlNode temp = doc.CreateElement(indices[i]);
        indice.Add(temp);

    }
    return indice;
}

public void criaXml(string tabela, List<string> indices, List<string> elementos)
{

    List<XmlNode> indice = new List<XmlNode>();

    doc.Load(caminho);



    XmlNode linha = doc.CreateElement(tabela);

    XmlNode linha2 = doc.CreateElement("produto");

    int j = 0;

    for (int i = 0; i < elementos.Count; i++)
    {
        indice = criaNo(indices);

        if (j ==5 && i > 0)

        {

            doc.SelectSingleNode("/empresa").AppendChild(linha);
             linha = doc.CreateElement(tabela);
           j = 0;


        }

        indice[j].InnerText = elementos[i];

        Console.WriteLine(indice[j].InnerText);
```

```
            linha.AppendChild(indice[j]);


            j++;


        }

        doc.Save(caminho);


    }




public void SalvaXml(string raiz, List<string> indices, List<string> elementos,string caminho)
    {


        List<XmlNode> indice = new List<XmlNode>();
        if (File.Exists(caminho))
        {
            doc.Load(caminho);
        }
        else
        {
            XmlDocument doc1 = new XmlDocument();
            doc = doc1;
            XmlNode rai = doc.CreateElement("Estimativa");
             doc.AppendChild(rai);
            doc.Save(caminho);
            doc.Load(caminho);
        }

        XmlNode linha = doc.CreateElement(raiz);



        int j = 0;

        for (int i = 0; i <elementos.Count; i++)
        {

            indice = criaNo(indices);


            indice[j].InnerText = elementos[i];

            Console.WriteLine(indice[j].InnerText);

            linha.AppendChild(indice[j]);

            if (j == elementos.Count-1)
            {
                indice[j].InnerText = elementos[i];


                doc.SelectSingleNode("/Estimativa").AppendChild(linha);
                  linha = doc.CreateElement(raiz);
                j = 0;
```

```
            }


            j++;


        }

        doc.Save(caminho);

    }
  }

}
```

Classe Principal.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;

using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;

namespace tela_Poison
{
   public partial class Form1 : Form
   {
      string[] seriesArray = { "Primeira Demanda", "Segunda Demanda", "Terceira Demanda", "Quarta
Demanda", "Quinta Demanda", "Sexta Demanda" };
      AbriXml novo = new AbriXml();
      double[,] matriz; double[,] matriztemp;
      Xlm cria = new Xlm();
      public Form1()
      {
         InitializeComponent();

         Grafico grafico = new Grafico(this.grafico);

      }

      private void button1_Click(object sender, EventArgs e)
      {




         string[] temp = (estadoTX.Text.Split(','));
         double[] ste = new double[temp.Length];
         for (int i = 0; i < temp.Length; i++)
         {
            ste[i] = Convert.ToDouble(temp[i]);

         }

         MatrizTransicao mt = new MatrizTransicao();
         matriz = mt.getMatriz(ste[ste.Length - 1], ste[0], ste);
```

```
            matriztemp = matriz;
            areaTX.Text = "\tMatriz de transição.:\n";
            grafico.Series.Clear();
            for (int i = 0; i < matriz.GetLength(0); i++)
            {
                areaTX.Text = areaTX.Text + System.Environment.NewLine;

                for (int j = 0; j < matriz.GetLength(0); j++)
                {
                    areaTX.Text = areaTX.Text + matriz[i, j].ToString("0.00000");
                }
            }

            areaTX.Text = areaTX.Text + System.Environment.NewLine;

            areaTX.Text = areaTX.Text + (System.Environment.NewLine + "\n Vetor de probabilidade
π(1)\r" + System.Environment.NewLine);

            Series series = this.grafico.Series.Add(seriesArray[0]);
            series.ChartType = SeriesChartType.FastLine;

            for (int i = 0; i < temp.Length; i++)
            {
                areaTX.Text = areaTX.Text + ("  " + matriz[0, i].ToString("0.000"));
                if (i == 0)
                {
                    series.Points.AddXY(0, matriz[0, i]);
                }
                else { series.Points.AddXY(matriz[0, i - 1], matriz[0, i]); }
            }


            matriz = mt.multiplica(matriz);

            Series series1 = this.grafico.Series.Add(seriesArray[1]);

            series1.ChartType = SeriesChartType.FastLine;

            areaTX.Text = areaTX.Text + (System.Environment.NewLine + "\n Vetor de probabilidade
π(2)\r" + System.Environment.NewLine);
            for (int i = 0; i < temp.Length; i++)
            {
                areaTX.Text = areaTX.Text + ("  " + matriz[0, i].ToString("0.000"));
                if (i == 0)
                {
                    series1.Points.AddXY(0, matriz[0, i]);

                }
                else { series1.Points.AddXY(matriz[0, i - 1], matriz[0, i]); }
            }

            matriz = mt.multiplica(matriz);

            Series series2 = this.grafico.Series.Add(seriesArray[2]);

            series2.ChartType = SeriesChartType.FastLine;

            areaTX.Text = areaTX.Text + (System.Environment.NewLine + "\n Vetor de probabilidade
π(3)\r" + System.Environment.NewLine);
            for (int i = 0; i < temp.Length; i++)
            {
                areaTX.Text = areaTX.Text + ("  " + matriz[0, i].ToString("0.000"));
                if (i == 0)
                {
```

```
            series2.Points.AddXY(0, matriz[0, i]);

        }
        else { series2.Points.AddXY(matriz[0, i - 1], matriz[0, i]); }
    }

    matriz = mt.multiplica(matriz);

    Series series3 = this.grafico.Series.Add(seriesArray[3]);

    series3.ChartType = SeriesChartType.FastLine;


    areaTX.Text = areaTX.Text + (System.Environment.NewLine + "\n Vetor de probabilidade
π(4)\r" + System.Environment.NewLine);
    for (int i = 0; i < temp.Length; i++)
    {
        areaTX.Text = areaTX.Text + ("  " + matriz[0, i].ToString("0.000"));
        if (i == 0)
        {
            series3.Points.AddXY(0, matriz[0, i]);

        }
        else { series3.Points.AddXY(matriz[0, i - 1], matriz[0, i]); }
    }

    matriz = mt.multiplica(matriz);

    Series series4 = this.grafico.Series.Add(seriesArray[4]);

    series4.ChartType = SeriesChartType.FastLine;


    areaTX.Text = areaTX.Text + (System.Environment.NewLine + "\n Vetor de probabilidade
π(5)\r" + System.Environment.NewLine);
    for (int i = 0; i < temp.Length; i++)
    {
        areaTX.Text = areaTX.Text + ("  " + matriz[0, i].ToString("0.000"));
        if (i == 0)
        {
            series4.Points.AddXY(0, matriz[0, i]);


        }
        else { series4.Points.AddXY(matriz[0, i - 1], matriz[0, i]); }
    }

    matriz = mt.multiplica(matriz);

    Series series5 = this.grafico.Series.Add(seriesArray[5]);

    series5.ChartType = SeriesChartType.FastLine;


    areaTX.Text = areaTX.Text + (System.Environment.NewLine + "\n Vetor de probabilidade
π(6)\r" + System.Environment.NewLine);
    for (int i = 0; i < temp.Length; i++)
    {
        areaTX.Text = areaTX.Text + ("  " + matriz[0, i].ToString("0.000"));
        if (i == 0)
        {
            series5.Points.AddXY(0, matriz[0, i]);

        }
```

```
            else { series5.Points.AddXY(matriz[0, i - 1], matriz[0, i]); }
        }
    }

    private void criaXmlToolStripMenuItem_Click(object sender, EventArgs e)
    {

        Conexao com = new Conexao();
        List<string> banco = new List<string>();
        List<string> tabela = new List<string>();
        List<string> elementos = new List<string>();



        banco = com.getIndice("show tables");

        for (int i = 0; i < banco.Count; i++)
        {
            tabela = com.getIndice("describe " + banco[i]);


            elementos = com.getComando("Select * From " + banco[i]);

            cria.criaXml(banco[i], tabela, elementos);

        }

        MessageBox.Show("Banco xml criado com sucesso", "Banco XML",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
    }

    private void abriXmlToolStripMenuItem_Click(object sender, EventArgs e)
    {


        novo = new AbriXml();
        novo.Show();


    }

    private void carregaGraficoToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.areaTX.Text = "";
        this.estadoTX.Text = "";
        List<int> quantidade = new List<int>();
        int[] valores = new int[6];

        quantidade = novo.getQuantidade();

        Series[] series = new Series[6];
        int j = 0;
        foreach (int val in quantidade)
        {
            valores[j] = val; Console.WriteLine(val);
            j++;
        }
        grafico.Series.Clear();

        for (int i = 0; i < valores.Length; i++)
        {
            if (valores[i] > 0)
            {
                series[i] = this.grafico.Series.Add(seriesArray[i]);
```

```
                series[i].ChartType = SeriesChartType.Bar;

                series[i].Points.Add(valores[i]);
            }

        }
    }

    private void salvarToolStripMenuItem_Click(object sender, EventArgs e)
    {
        String caminho = "";
        SaveFileDialog openFileDialog1 = new SaveFileDialog();
        openFileDialog1.Filter = "xml files (*.xml)|*.xml|All Files (*.*)|*.*;";
        DialogResult dr = openFileDialog1.ShowDialog();

        if (dr == System.Windows.Forms.DialogResult.OK)
        {
            foreach (String cam in openFileDialog1.FileNames)
            {
                caminho = cam;
            }



            List<string> valores = new List<string>();
            List<string> indices = new List<string>();

            indices.Add("primeira");
            indices.Add("segunda");
            indices.Add("terceira");
            indices.Add("quarta");
            indices.Add("quinta");
            indices.Add("sexta");
            indices.Add("data");
            //Apresentação da data com Data e Hora atual
            string sDataHoraAtual = System.DateTime.Now.ToString("dd/MM/yyyy hh:mm:ss");
            String temp = "";
            MatrizTransicao MT = new MatrizTransicao();
            for (int i = 0; i < 7; i++)
            {


                for (int j = 0; j < 5; j++)
                {
                    temp = temp + ";" + matriztemp[0, j];
                }
                matriztemp = MT.multiplica(matriztemp);
                valores.Add(temp);
                temp = "";
            }
            valores.RemoveAt(4);
            valores.Add(sDataHoraAtual);
            cria.SalvaXml("Previsao", indices, valores, caminho);

        }
    }
  }
}
```