UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Contribuição ao Processo de Integração de Informações da Manufatura para Empresas de Pequeno e Médio Porte

Autor: Niederauer Mastelari
Orientador: Nivaldo Lemos Coppini

02/2004

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

M393c

Mastelari, Niederauer
Contribuição ao processo de integração de informações da manufatura para empresas de pequeno e médio porte / Niederauer.--Campinas, SP: [s.n.], 2004.

Orientador: Nivaldo Lemos Coppini.
Tese (Doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Tornos – Controle numérico. 2. Engenharia de software. 3. UML (Linguagem de modelagem padrão). 4. Sistemas de fabricação integrada por computador. I. Cappini, Nivaldo Lemos. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA
DEPARTAMENTO DE ENGENHARIA DE FABRICAÇÃO

Contribuição ao Processo de Integração de Informações da Manufatura para Empresas de Pequeno e Médio Porte

Autor: Niederauer Mastelari
Orientador: Nivaldo Lemos Coppini

Curso: Engenharia Mecânica
Área de Concentração: Materiais e Processos de Fabricação

Tese de doutorado apresentada à comissão de Pós Graduação da Faculdade de Engenharia Mecânica, como requisito para a obtenção do título de Doutor em Engenharia Mecânica.

Campinas, 2004
S.P . – Brasil

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA
DEPARTAMENTO DE ENGENHARIA FABRICAÇÃO

TESE DE DOUTORADO

Contribuição ao Processo de Integração de Informações da Manufatura para Empresas de Pequeno e Médio Porte

Autor: **Niederauer Mastelari**
Orientador: **Nivaldo Lemos Coppini**

_____
Prof. Dr. Nivaldo Lemos Coppini , Presidente
UNICAMP

_____
Prof. Dr. Oswaldo Agostinho
UNICAMP

_____
Prof. Dr. Olívio Novaski
UNICAMP

_____
Prof. Dr. Tamio Shimizu
USP

_____
Prof. Dr. -Ing. Klaus Schützer
UNIMEP

Campinas, 17 de fevereiro de 2004

DEDICATÓRIA:

Dedico este trabalho às minhas filhas Laís e Luíza, e à minha esposa Flávia.

AGRADECIMENTOS

Ao meu orientador por mais esta oportunidade.

Aos meu pais Esmeraldo Mastelari e Frumen Gonçalez Mastelari que sempre me incentivaram.

Aos profs. Batocchio e Olívio, pelo apoio, no papel de chefe do Departamento de Fabricação.

Ao CONDEF constituído pelos professores: Anselmo, Roseana, Paulo Lima, Eugênio, Batochio, Maria Helena, Olívio, pelo apoio.

Aos meus colegas de trabalho: Cristina, Vera, Rute, Aristides, Claudiomiro, Fábio Gatamorta pelo companheirismo.

E a todos os companheiros de trabalho da Faculdade de Engenharia Mecânica com quem tenho convivido nestes últimos dez anos.

Resumo

MASTELARI, Niederauer, *Contribuição ao Processo de Integração de Informações da Manufatura para Empresas de Pequeno e Médio Porte*, Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2004. 180 p. Tese (Doutorado)

As empresas podem ser compreendidas como sistemas complexos que competem por recursos limitados em um dado ambiente. Este ambiente é o espaço econômico, político e social na qual a organização opera. Conforme uma empresa cresce tende a diferenciação interna, desta forma as áreas de produção e materiais se diferenciam para que haja uma maior facilidade de coordenação das atividades relacionadas ao planejamento da produção e da gestão de materiais. No entanto, com isto observa-se maior dificuldade de coordenação entre estas funções. Os Sistemas de Informação se apresentam como um modo de resgatar a integração entre estas áreas. Porém, o desenvolvimento de software é uma atividade complexa que demanda pessoal especializado e tempo. Seus custos são altos e por ser uma atividade recente, suas técnicas e ferramental ainda estão em desenvolvimento, o que traz aos administradores que têm que tomar a decisão de desenvolver um software, muitas incertezas. Este trabalho busca abordar estes problemas propondo uma metodologia orientada a objetos para o desenvolvimento de software, e usando esta metodologia desenvolver um Sistema de Informação para edição de programas CN para centros de torneamento integrado com o estoque e compras, voltado para pequenas e médias empresas.

Palavras Chave:

- Processo de Torneamento, Comando Numérico, Sistemas de informação, Métodos Orientados a Objetos, Desenvolvimento de Sistemas,UML, Engenharia de Software.

Abstract

MASTELARI, Niederauer*, Contribution to Integration of Manufacturing Information for small and medium Enterprises*, Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2004. 180 p. Tese (Doutorado)

Companies can be understood as complex systems that compete for limited resources in a given environment. This environment is the economic, political and social space in which the organization operates. As a company grows it tends towards internal differentiation, and as a consequence production and materials areas also differentiate to make easier the coordination of activities related to production planning and the materials management. However, this very differentiation generates difficulties in coordinating these two tasks. Information System are presented as a way to regain the integration between these areas, but there is a downside to it: software development is a complex activity that demands specialized personnel and much time. Its costs are high and due to its newness, its techniques and tool rack are still under development, which brings many uncertainties to the administrators who must make decisions about software development. This work aims to approach these problems considering an object oriented methodology for software development, and to use this methodology to develop an Information System that edits turning NC programs integrated with supply and purchasing, suitable to small and medium enterprises.

Keywords: Turning Process, Numerical Control, Information Systems, Oriented Object Methods, Development of Information Systems, software enginnering, UML.

Índice

Lista de Figuras

Lista de Tabelas

Siglas

ADO: “Activex Data Objects”, Objetos para manipulação de dados Microsoft.

APT: “Automatically Programmed Tool”, Ferramenta Programada Automaticamente.

API: “Application Programming Interface”, Interface de programas aplicativos.

CAM: “Computer Aided Manufacturing”, Manufatura Auxiliada por Computador.

CAD: “Computer Aided Design”, Projeto Auxiliado por Computador.

CAPP: “Computer Aided Process Planning”, Planejamento do Processo Assistido por Computador.

CASE: “Computer Aided Software Engineering”, Engenharia de Software Auxiliada por Computador.

CIM: “Computer Integrated Manufacturing”, Manufatura Auxiliada por Computador.

CNC: Comando Numérico Computadorizado

DNC: “Direct/Distributed Numerical Control”, Controle Numérico Distribuído.

ERP: “Enterprise Resource Planning”, Planejamento dos Recursos Empresariais.

IGES: “Initial Graphics Exchange Specification” , Especificação Inicial para Transferência de Dados Gráficos.

LAN:”Local Area Network”, Rede Local de Computadores.

MDI: “Manual Data Input”, Entrada de Dados Manual.

MDAC: “Microsoft Data Access Components”, Componentes de acesso a dados Microsoft.

PDM: “Product Data Management”, Gerenciador de Dados do Produto.

SBDOO: Sistema de Banco de Dados Orientado a Objetos.

SI: Sistemas de Informação Computadorizados

SIG: Sistema de Informações Gerenciais

SFM: Sistema Flexível de Manufatura

SPT: Sistemas Processadores de Transações

SQL : “Structured Query Language”, Linguagem Estruturada de Consulta.

STEP: “Standard for the Exchange of Product Model Data”, Padrão para a Transferência de Dados de Modelagem de Produtos.

TG: Tecnologia de Grupo.

UML: “Unified Modeling Language”, Linguagem de Modelagem Unificada.

XP: “Extreme Programming”, Programação extrema.

1 Introdução

As empresas podem ser compreendidas como sistemas complexos que competem por recursos limitados em um dado ambiente. Este ambiente é o espaço econômico, político e social na qual a organização opera.

Conforme uma empresa cresce, tende à diferenciação, isto é à multiplicação e à elaboração de funções, o que lhe traz também multiplicação de papéis e diferenciação interna. Os padrões difusos e globais são substituídos por funções mais especializadas, hierarquizadas e altamente diferenciadas (Chiavenato,1995). Estas funções são subsistemas do sistema maior empresa. No entanto, não é possível cogitar de algum subsistema isolado e completamente independente, os subsistemas dependem de informações que provêm dos outros subsistemas.

Um sistema de informação baseado em computador, SI, é composto por hardware, software, banco de dados, telecomunicações, pessoas e procedimentos que estão configurados para coletar, manipular, armazenar e processar dados em informação (Stair,1998). Com o surgimento desta tecnologia foi possível cogitar um alto grau de integração entre os sistemas.

Dentro deste contexto as áreas de produção e materiais se diferenciam para que haja uma maior facilidade de coordenação das atividades relacionadas ao planejamento da produção e da gestão de materiais. No entanto, com isto observa-se maior dificuldade de coordenação entre estas funções.

Os SI se apresentam como um modo de resgatar a integração entre estas áreas. Porém, os SI mais utilizados em cada área são de natureza diferente. Os sistemas de informação da área de desenvolvimento do produto e processos são típicos sistemas de trabalho do conhecimento (Laudon, 2001), e trabalham com informações de projeto, componentes e operações do item que está sendo projetado, o inter-relacionamento de componentes e versões antigas de projetos. Estas informações são difíceis de estruturar seguindo padrões similares, e o seu tamanho pode variar dinamicamente. A área de materiais, por sua vez, faz uso intensivo de Sistemas Processadores de Transações que utilizam informações bem estruturadas e estáveis e, os seus registros têm tamanhos fixos e normalmente pequenos, raramente mais longos que poucas centenas de bytes, organizados em bancos de dados relacionais.

A combinação destes sistemas, porém, é necessária, pois em muitas situações eles utilizam dados comuns. Por exemplo, a área de desenvolvimento de produtos necessita de dados sobre os materiais e ferramental tais como características, custos e disponibilidade. Por sua vez a área de materiais precisa de dados do produto para avaliar a necessidade dos insumos e assim poder gerenciá-los.

Este trabalho busca abranger este problema, propondo como solução o desenvolvimento de software que propicie a integração entre a área de produção e materiais. Este desenvolvimento retoma o trabalho de Mastelari (1996), de desenvolvimento de software voltado para pequenas e médias empresas, com enfoque na programação de peças de torneamento com interface baseada em linguagem técnica, porém, integrando este sistema a um banco de dados de ferramentas e materiais.

1.1 Objetivos

- Propor uma metodologia orientada a objetos para o desenvolvimento de software voltado para ambientes de fabricação, considerando a necessidade de integração entre os sistemas que atendem às diferentes áreas envolvidas no ciclo de vida de um produto.

- Usar a metodologia proposta para desenvolver um sistema de informação com um módulo para descrição de peças para centro de torneamento, integrando-o com o

banco de dados de da área de materiais de uma empresa hipotética de pequeno porte que trabalha na produção de peças com grande variedade, pequenos lotes e usando máquinas de Comando Numérico Computadorizado (CNC). O sistema deverá ter recursos para gerar o programa CN da peça, calcular percursos, tempos e custos. Deverá gerar solicitações de ferramentas e materiais, pedidos de compra e fornecer os orçamento das peças.

1.2 Justificativa do trabalho

Graças aos recentes desenvolvimentos havidos em materiais e geometrias para ferramentas de usinagem e em projetos e construção de máquinas-ferramenta, foi possível provocar drásticas reduções nos tempos e custos denominados diretos do processo. Desta forma, tempos e custos indiretos passaram a representar a grande parcela destes fatores na composição final da produção e composição dos custos industriais finais. Reduzir tais tempos e custos tornou-se, portanto, uma tarefa imposta pelas circunstâncias reinantes em qualquer indústria moderna. Tempos e custos indiretos relativamente altos são dispendiosos na realização de programas CN. Isto ocorre, pois os sistemas de programação genericamente chamados de CAM, ou são baseados em linguagens de programação de alto nível, ou são baseados em recursos gráficos. Os primeiros praticamente se encontram em desuso, ou sua utilização justifica-se quando e apenas se a manufatura encontra-se regida por um alto grau de integração (CIM). Os sistemas gráficos são válidos e práticos quando um mínimo de integração da manufatura existe com relação ao desenvolvimento do projeto (CAD/CAM). Para situações em que a manufatura ainda seja praticada de forma semi-automática, ou seja, a manufatura não é integrada, entretanto a fabricação de peças é essencialmente realizada em máquinas CNC, os sistemas de programação gráficos apresentam graus de dificuldades que demandam mão de obra qualificada, resultando desta forma, em tempos e custos relativamente altos como mencionados acima.

O objetivo deste trabalho é preencher a abrangência deste problema, apresentando uma contribuição ao processo de integração de informações da manufatura voltada para pequenas e médias empresas, nas quais parte das peças encomendadas por clientes internos ou externos são produzidas a partir de desenhos. Desta forma, o resultado do desenvolvimento apresentado neste trabalho é um Sistema de Informações constituído por um editor/simulador de programas CN

baseado em linguagem técnica mecânica e com interface interativa, e integrada a um banco de dados da área de materiais da empresa.

Para o desenvolvimento deste Sistema de Informações é proposta uma metodologia de desenvolvimento de software orientado a objetos a partir de um conjunto mínimo de ferramentas e atividades, adequada para pequenos grupos de desenvolvimento.

1.3 O Desenvolvimento

A orientação a objetos, apesar de não representar a solução final para os problemas de engenharia de software, encoraja a construção de sistemas confiáveis e robustos (Buzato,1998). A programação convencional esta focada em funções que processam os dados. Diferentemente, a orientação a objetos foca tipos de objetos que têm atributos e comportamento. Com a Análise Orientada a Objetos é mais natural se pensar os sistemas, o conceito de objeto utilizado no desenvolvimento é semelhante daquele utilizado na linguagem usual, e por isto a comunicação entre desenvolvedores, usuários e administradores é facilitada. Os objetos têm individualidade e por isto cada classe de objetos pode ser alterada de forma relativamente independente de outras classes. Isto faz com que seja mais fácil testar e modificar estas classes. A manutenção de sistemas orientados a objetos é mais fácil do que de sistemas convencionais. Além disto componentes de software orientados a objetos tem uma probabilidade maior de serem reutilizados. O processo de desenvolvimento de software Orientado a Objetos faz uso intensivo de classes de objetos já desenvolvidas e testadas.

Quanto mais tarde, em um ciclo de desenvolvimento, forem descobertos problemas, maiores serão os esforços e os custos para solucioná-los. Portanto, os trabalhos de coleta de requisitos, análise e projeto se tornam fundamentais para que um projeto se torne consistente e viável. Estas atividades relacionadas formam uma metodologia de desenvolvimento de software. Uma metodologia de desenvolvimento de software deve fornecer, também, diretrizes para a organização do desenvolvimento do software e um conjunto de ferramentas para que este possa ser construído. No ciclo de desenvolvimento o trabalho de modelagem busca a especificação, documentação, comunicação e visualização de artefatos. Apesar de não ser o objetivo principal de um projeto de software a documentação que envolve o processo de desenvolvimento disciplina e facilita o trabalho de desenvolvimento e sua posterior manutenção.

Diferentes metodologias orientadas a objetos utilizam diferentes simbologias. Cada simbologia possui seus próprios conceitos, terminologias, notações, regras e diagramas. Isto resulta numa grande confusão entre diferentes equipes dificultando a compreensão de projetos diferentes e a comunicação entre aqueles que querem criar modelos de qualidade que facilitam a construção e manutenção de sistemas cada vez mais eficazes. A idéia central da "Unified Modeling Language", UML, é, a utilização de uma só linguagem para o desenvolvimento de sistemas. Ela surgiu do esforço de Grady Booch, James Rumbaugh e Ivar Jacobson para unificar suas metodologias em uma só linguagem e, como resultado deste esforço surgiu a UML (Kobryn,1999).

O desenvolvimento busca atingir este escopo, partindo do estudo e compreensão do ambiente de uma pequena empresa que utiliza tecnologia CNC, passando pela administração de seus sistemas de informação e neste contexto propor o desenvolvimento de um sistema usando técnicas Orientadas a Objetos para o ambiente de manufatura. O sistema deverá ter como objetivos principais o auxílio na programação de peças em centros de torneamento, porém, integrado à área de materiais.

O desenvolvimento deste sistema é focado nas fases de levantamento dos requisitos, análise e projeto, implementação e testes. A metodologia utilizada é baseada no método ICONIX (Rosenberg e Scott, 2001) e é adequada às características do projeto: tamanho pequeno ou médio, requisitos bem definidos e estáveis, desenvolvimento em pequenas equipes; e por utilizar um conjunto restrito, porém robusto de diagramas, e ter diretrizes simples claras e objetivas. A modelagem do sistema é feita usando a Linguagem Unificada de Modelagem (UML).

A codificação do protótipo é feita usando principalmente a linguagem Object Pascal do sistema Delphi da Borland. As tabelas do banco de dados de ferramentas são implementadas pelo software MS-Access assim como os aplicativos para o estoque e compras. O protótipo tem as principais funcionalidades apontadas no levantamento de requisitos. No desenvolvimento do sistema, um conjunto de classes é definido. Estas classes servem como base para futuros desenvolvimentos em contextos semelhantes ao deste trabalho.

1.4 Organização do Trabalho

O trabalho é organizado em seis capítulos, o primeiro, a Introdução, apresenta os objetivos do trabalho e como ele está organizado.

O segundo capítulo, "A Tecnologia CNC e sistemas computacionais associados", apresenta um estudo sobre a tecnologia CNC e seus impactos nas empresas. Apresenta também os principais sistemas que apoiam ou interagem com esta tecnologia.

O terceiro capítulo, "Sistemas de informação", aborda a Administração dos Sistemas de Informação e sua importância para as empresas enfatizando a estreita relação entre os SI e a estrutura organizacional das empresas, além disto, busca contextualizar o desenvolvimento de sistemas dentro da Administração de Sistemas de Informação.

O quarto capítulo, "Métodos Orientados a Objetos", aborda os principais conceitos da Orientação a Objetos e a sua importância para o desenvolvimento dos sistemas, apresenta também metodologias orientadas a objeto para o desenvolvimento de sistemas.

O quinto capítulo, "Metodologia de desenvolvimento", apresenta o projeto de uma metodologia de desenvolvimento de sistemas orientada a objetos apropriada para o desenvolvimento de sistemas de tamanho médio a serem realizados em pequenos grupos, tais como grupos de desenvolvimento das áreas de fabricação ou de áreas de tecnologia de informação de pequenas empresas.

O sexto capítulo, "Desenvolvimento do Sistema", apresenta as fases de desenvolvimento de um sistema de informação para edição de programas CN com geração automática de código, simulação do processo, geração de orçamentos e integrado ao estoque e a compras. Este desenvolvimento é feito usando a metodologia projetada e busca validá-la.

O sétimo capítulo, "Conclusões", fecha este trabalho com as conclusões obtidas neste trabalho.

2 A Tecnologia CNC e sistemas computacionais associados

Este capítulo faz um estudo do estado atual da tecnologia CNC e dos conceitos relacionados a ela. Os principais itens abordados são: A tecnologia CNC suas aplicações, vantagens e desvantagens e modos de programação. Além disto são apresentados de forma sucinta os sistemas baseados em computador que apóiam estes sistemas tais como: “Computer Aided Design”, CAD, “Computer Aided Manufacturing”, CAM, “Direct/Distributed Numerical Control”, DNC. Apresenta também os conceitos de Sistema Flexível de Manufatura SFM, e de “Computer Integrated Manufacturing”, CIM localizando os sistemas CNC e seus sistemas de apoio em relação a estes macro-sistemas.

2.1 A tecnologia CNC e suas aplicações

O termo automação programável aplica-se a sistemas nos quais os equipamentos são projetados com a capacidade de alterar os passos do processo e ou a sua seqüência de tal forma que diferentes estilos de produtos podem ser produzidos. Neste tipo de automação o processo é controlado por um programa, o qual é constituído por um conjunto de instruções que o equipamento pode ler e interpretar. Mudanças no processo são feitos alterando-se o programa (Groover, 1996).

Controle Numérico é uma forma de automação programável na qual certas funções das máquinas-ferramenta são controladas por um programa. O programa, formado por comandos e

números, define como uma peça particular pode ser feita. Se o projeto da peça ou o método de sua manufatura são alterados é necessário alterar-se a sua programação.

O advento e a adoção em larga escala de máquinas-ferramenta de comando numérico foi o mais significante desenvolvimento na manufatura nos últimos quarenta anos. Estas máquinas levaram a automação a um novo nível através de medidas automáticas do processo e o seu controle por realimentação de posicionamentos e da flexibilidade programável (Degarmo, 1999).

Nos primórdios desta tecnologia as máquinas de controle numérico não tinham computadores incorporados a elas. Com o barateamento dos microprocessadores, memórias e outros dispositivos da microeletrônica, estes recursos puderam ser incorporados transformando as máquinas de comando numérico em máquinas com Comando Numérico Computadorizado, CNC. Com isto novas funções puderam ser acrescentadas tais como: correção das trajetórias das ferramentas, correção do posicionamento das ferramentas devido a desgastes, edição de programas diretamente na máquina, capacidade de receber e enviar dados de diferentes fontes, armazenar um maior número de programas CN. Além disto tornou-se mais fácil o aprendizado da programação CN por parte dos seus operadores. Atualmente os comandos das máquinas são constituídos quase que exclusivamente por sistemas CNC.

A tecnologia CNC é mais adequada aos trabalhos que tenham as seguintes características gerais:

- Lotes pequenos e médios
- Geometria da peça complexa.
- Tolerâncias rígidas.
- Muitas operações precisam ser executadas para a realização da peça.
- Muito material precisa ser removido.
- Constantes mudanças de projeto.

- Peças com grande valor agregado, nas quais enganos no processamento seriam custosos.

- Peças que requerem 100% de inspeção.

As empresas investem em máquinas CNC para aumentar sua competitividade através de uma série de melhorias nos processos de produção, incluindo aumento de flexibilidade, melhoria da qualidade, tempos de ciclo reduzidos e a habilidade de produzir lotes pequenos de maneira econômica, (Simon, 2001).

Através dos programas CN é possível maior repetibilidade e qualidade na produção das peças. Dispositivos de fixação podem ser padronizados, tempos de preparação da máquina ("setup") podem ser reduzidos. Quando combinados com estratégias gerenciais e organizacionais tais como tecnologia de grupo e manufatura celular, máquinas programáveis podem prover uma tremenda melhoria na produtividade, (Degarmo, 1999).

A tecnologia de grupo é utilizada para a produção de peças em médias quantidades. Tecnologia de grupo, TG, é uma abordagem da manufatura na qual peças similares são identificadas e agrupadas para se obter melhorias no seu projeto e produção. A TG minimiza as desvantagens da produção em lotes e explora as similaridades entre as peças utilizando processos e ferramental similares para a sua produção. A TG pode ser implementada por meio manual ou com técnicas de automação. Com as mudanças no mercado solicitando novos e mais variados produtos em menor quantidade, a flexibilidade das operações de manufatura é altamente desejável. Quando automação é utilizada, o termo sistema de manufatura flexível, FMS, é freqüentemente utilizado.

As máquinas CNC também podem ser usadas como importante ponto para coleta de dados para sistemas de supervisão da manufatura. Para isto as máquinas CNC precisam ser conectadas através de redes locais que interligam os diversos equipamentos eletrônicos de uma determinada área. Por meio destes softwares de supervisão é possível se obter informações confiáveis sobre a produção: número de peças produzidas, tempos de ciclo e de parada, ritmo de produção. Isto pode ser conseguido com a simples aquisição de sinais de início, fim de ciclo e potência do motor, ou seja, o sistema de gestão pode avaliar quando as máquinas estão efetivamente

usinando, em manutenção ou ociosas (Meireles,2000). A coleta de dados nas máquinas CNC é possível com a inclusão de funções miscelâneas na programação CN que informem os estados das máquinas. Isto mostra a grande flexibilidade que os comandos numéricos oferecem.

A maior desvantagem das máquinas ferramentas CNC é o seu relativo alto custo inicial. Isto significa que a sua implantação deve ser bem justificada do ponto de vista econômico. Além disto, a programação e a manutenção das máquinas CNC demanda pessoal especializado, o que implica em custos indiretos altos. As máquinas ferramentas CNC são sistemas complexos, quebras podem ser custosas, por isto treinamento e manutenção preventiva são essenciais. Entretanto estas limitações são facilmente suplantadas pelas vantagens econômicas das máquinas CNC (Kalpakjian, 2001).

Simon (2001), mostra que o uso inadequado da tecnologia CNC pode não propiciar os níveis de produtividade e flexibilidade compatíveis com os esperados para aplicações dessa tecnologia. O estudo e otimização das atividades "elaboração do programa CN", "transferência do programa CN para a máquina" e "pré-ajustagem de ferramentas", é de fundamental importância para os trabalhos de melhoria de produtividade em ambiente CNC, pois o tempo gasto com estas atividades tem influência direta no tempo de ciclo dos produtos e não agrega valor aos mesmos.

O sistema proposto neste trabalho busca aumentar a eficiência do processo através de uma maior facilidade na elaboração e transferência dos programas CN.

2.1.1 A programação CN

Diferentemente das máquinas-ferramentas convencionais as máquinas CNC necessitam serem programadas, ou seja, a peça que será produzida deverá ser descrita através de um programa CN (Controle Numérico) que contém informações sobre a máquina-ferramenta, a geometria e disposição das ferramentas, trajetórias que estas ferramentas deverão realizar, as dimensões da peça bruta e acabada, e as informações tecnológicas tais como velocidade de corte, avanço e profundidade de corte.

Os programas CN variam muito em tamanho dependendo principalmente da complexidade da peça, e da utilização de ciclos de máquinas pré-definidos no comando numérico. Os programas de centros de usinagem tendem a serem maiores devido à possibilidade de usinar várias faces, ao maior número de ferramentas que podem ser armazenadas nos magazines, e por geralmente serem equipados com sistemas de troca de paletes, assim, pelo menos duas preparações diferentes podem ser processadas com um programa.

Para os centros de torneamento os fabricantes oferecem comandos elaborados para as operações mais complexas tais como: ciclos de desbaste, roscamento, furação. Quando se utilizam sistemas que automatizam a programação CN é interessante que os algoritmos que geram a programação façam uso destes ciclos.

A maior parte das máquinas CNC utilizam na sua programação instruções que seguem as normas DIN 66024 e DIN 66003. Estas instruções definem uma grande variedade de condições de percurso, coordenadas, parâmetros tecnológicos, e funções auxiliares. Este conjunto de instruções e regras forma uma linguagem de programação das máquinas de comando numérico, esta linguagem é comumente denominada de “código G”.

A realização do programa CN é uma questão bastante importante nos processos que utilizam tecnologia CNC, pois esta atividade deve atender a uma crescente exigência de redução de custos e elevada qualidade. Os programas CN podem ser realizados por um departamento de programação, serem feitos diretamente no chão de fábrica, ou ainda serem provenientes de clientes ou fornecedores externos.

Dependendo do grau de automatização podemos ter os seguintes principais modos de programação:

Programação manual:

A programação manual é o termo usado para descrever a preparação de um programa sem a utilização de recursos computacionais para determinar trajetórias da ferramenta, pontos de intersecção de perfis, avanços, velocidades (Gibbs em Simon, 2001).

Na programação manual faz-se primeiramente uma série de cálculos das relações dimensionais entre a ferramenta, a peça e a área de trabalho, com base nos desenhos da peça proveniente da engenharia. Então, um formulário é preparado detalhando as informações necessárias para a realização da particular operação. O programa da peça é preparado com base nestas informações.

A programação manual deve ser realizada por alguém que conhece o processo, a máquina onde a peça será produzida, seu comando numérico e sua capacidade de produção e é capaz de compreender, ler e alterar programas CN.

A programação manual é um trabalho que consome muito tempo e normalmente é tedioso. Freqüentemente, erros humanos grosseiros de cálculo e escrita podem ocorrer, e a depuração é uma tarefa difícil usando esta técnica. Conseqüentemente a programação manual é utilizada para aplicações simples, ponto a ponto, como operações de furação ou em trabalhos de torneamento e fresamento que envolvam poucas operações.

Programação Assistida por Computador

Na programação assistida por computador, os cálculos simples requeridos na programação manual são automatizados. Isso economiza tempo e resulta num programa mais preciso e mais eficiente (Lynch em Simon, 2001). Um dos modos de Programação Assistida por Computador envolve linguagens de programação. Uma linguagem de programação é um conjunto de regras, símbolos e instruções que facilitam a comunicação entre o ser humano e o computador.

No início da tecnologia CN muitas empresas desenvolveram linguagens próprias para auxiliar na tarefa de programar as máquinas CN, o que causava grande confusão. Isto levou ao desenvolvimento de uma linguagem computacional para o controle de máquinas ferramentas pelo “Massachussets Institute of Technology”, MIT , patrocinada pela Força Aérea Norte Americana. Esta linguagem denominada APT, “Automatically Programmed tools”, foi anuncia pelo MIT em 1959, inicialmente uma linguagem para ser usada em computadores de grande porte (“mainframes”).

Os programas em APT são pós-processados, de forma que o programa é convertido no conjunto de instruções para uma máquina específica, assim um mesmo programa pode ser utilizado em máquinas com diferentes controles numéricos.

O APT parte dos procedimentos para usinar o material no formato final da peça. O programador deve ser capaz de visualizar mentalmente a peça sendo usinada, o que requer grande experiência e treinamento para se tornar um especialista. Para muitas aplicações complexas as funções dos sistemas CAD/CAM se mostram limitadas para gerar o programa CN necessário. É estimado que de 5 a 10% das peças da indústria aeroespacial e de defesa nos Estados Unidos ainda requerem programação APT (Degarmo, 1999).

Programação MDI

Neste modo de programação MDI, "Manual Data Input", o programa é realizado diretamente por meio das interfaces do comando numérico na própria máquina-ferramenta. Portanto o programador utiliza o monitor e teclado do CNC para realizar a comunicação e entrada manual dos dados.

Este método é utilizado tanto pelos programadores como operadores e pelo pessoal que faz a manutenção dos equipamentos.

O modo MDI por ser simples não requer apoio de um grupo especializado de programação NC, MDI é um meio de pequenas empresas implementarem economicamente a tecnologia de controle numérico em suas operações (Groover, 1996).

Este modo de programação é também muito utilizado na correção de erros ou em pequenas alterações em programas prontos. Nos comandos numéricos mais recentes recursos gráficos foram introduzidos que auxiliam este modo de programação e tornam as tarefas de verificação de erro e de simulação do programa tarefas mais fáceis e eficazes.

Uma desvantagem deste modo de programação é a sua limitação quanto à complexidade do programa CN. Quanto maior e complexo o programa vai se tornando, mais necessária torna-se a utilização de sub-rotinas e de uma maior organização do programa para que se mantenha o controle sobre a ocorrência de pequenos erros.

Outra consideração importante, para a maior parte das máquinas CNC, em relação à programação via MDI é que é necessário que fiquem sem produzir enquanto é feita a programação, portanto se torna um modo muito caro de programação, caso a máquina esteja dedicada à produção.

Programação NC Via CAM:

Os sistemas gráficos CAM têm como uma de suas funções a geração de programas CN. A interação entre o programador e o sistema é um significante beneficio da programação assistida por sistemas CAM. Quando um sistema CAM é usado, o programador recebe um imediato retorno visual ao confirmar uma região a ser usinada, podendo verificar se o trabalho está de acordo com o projetado. Quando a peça esta sendo projetada, o programador pode ver exatamente como os movimentos serão executados por meio da simulação dos movimentos da ferramenta em relação à peça. Erros podem ser corrigidos imediatamente, diferentemente das linguagens de programação que fazem a depuração depois do programa ter sido escrito totalmente.

Nos sistemas CAD faz-se o projeto do produto e de suas partes. O resultado deste projeto, incluindo a definição geométrica de cada uma de suas peças, pode ser utilizado como ponto de partida pelo programador CN. Esta utilização do projeto como base da programação CN, economiza tempo valioso se comparado a se construir as peças a partir do desenho, usando uma linguagem de programação. Além disto, rotinas especiais são disponíveis nos sistemas CAD/CAM que automatizam a geração de percursos de ferramentas para diversas operações tais como fresamento em torno de uma peça, e certas operações ponto a ponto. Estas operações são denominadas de macros. O uso destas rotinas resulta em ganho significativo de tempo e esforço na programação (Groover, 1996).

Utilizando-se sistemas CAD/CAM pode-se projetar peças complexas. Eles oferecem recursos computacionais que se bem utilizados tornam o projeto mais rápido e preciso, estes sistemas propiciam uma maior facilidade para o programador verificar o programa. Além disto geram o programa CN para a máquina-ferramenta especificada de forma que diferentes máquinas podem ser utilizadas para realizar a peça projetada.

A grande desvantagem dos sistemas CAD/CAM é o seu alto custo, o software é caro, e o sistema necessita de estações de trabalho com alta capacidade de processamento e vídeos especiais de alta resolução. Além disto, para a utilização de seus recursos necessita de pessoal treinado e altamente especializado.

Programação automatizada

Neste modo de programação o sistema é capaz de tomar as decisões lógicas sobre como a peça deve ser usinada. Dado o modelo geométrico de uma peça que foi definido durante o projeto do produto, o sistema automatizado por computador deverá ter capacidades lógicas e de tomada de decisão suficientes para gerar o programa CN sem intervenção humana. A interação entre os sistemas CAD/CAM e o Planejamento do Processo Assistido por Computador é fundamental para o sucesso da técnica (Simon, 2001).

2.1.2 Comunicação, transferência e organização de programas CN

Como um programa pode ser reutilizado, é interessante o estabelecimento de um método para armazená-lo, organizá-lo e recuperá-lo. As máquinas atuais têm capacidade de armazenamento de um grande número de programas, porém, para uma melhor organização e por segurança, é necessário que exista uma cópia do programa fora da máquina.

De acordo com, Simon (2001), os principais sistemas não manuais de transferência de dados são:

- Gravadoras/Leitoras de fitas
- Leitura e gravação de fitas cassetes
- Sistemas transferências de dados portáteis alimentados por bateria, baseados em memória volátil.
- Dispositivos portáteis de leitura de disquetes
- Computadores portáteis (laptops, notebooks, Desktops).

Pode-se acrescentar a estes sistemas as redes locais de computadores, onde as máquinas CNC passam a ser um elemento, um nó, na rede fazendo a comunicação por meio de um protocolo de comunicação.

Um mecanismo muito comum de interligação e comunicação entre microcomputadores e as máquinas CNC são as portas de comunicação serial. A porta serial padrão RS232-C propicia um procedimento relativamente simples de comunicação entre os dispositivos de armazenagem (laptops, notebooks, desktops) e as máquinas ferramenta. Esta comunicação é indicada para pequenas distâncias. Esta comunicação, bidirecional, pode transferir, parâmetros, programas CN e dados da máquina para o sistema ou do sistema para a máquina. O sistema que será desenvolvido utilizará este modo de comunicação para transferir programas entre o sistema e as máquinas CNC.

2.2 Sistemas de apoio à produção: CAD/CAM , CAPP, DNC

Dentro destas empresas pode-se destacar os seguintes sistemas de apoio à produção:

• O CAD,"Computer Aided Design", envolve o uso de computadores para criar modelos de produtos. Estes sistemas gráficos interativos são uma poderosa ferramenta para uso em projeto, e modelagem geométrica. O CAD é o principal elemento da CIM "Computer Integrated Manufacturing". Segundo Degarmo (1999) as tarefas realizadas pelo CAD são :

- Modelagem geométrica
- Analise de engenharia
- Revisões de projeto e avaliação
- Desenho

• O sistema CAM, "Computer Aided Manufacturing", utiliza o computador para auxiliar no planejamento, administração, e controle dos processos de manufatura. A sua função primária é a preparação e remessa de programas de controle numérico para maquinas CNC, equipamentos de movimentação/manipulação de materiais e outros equipamentos de suporte. Os sistemas CAM

permitem que os dados sobre a geometria do produto, criados a partir do CAD, possam ser usados diretamente na criação de programas CN para máquinas CNC.

Por causa dos seus benefícios, os sistemas CAD são combinados com os sistemas CAM em sistemas CAD/CAM. A combinação permite a transferência das informações do estágio de projeto para o planejamento da manufatura, sem a necessidade de uma nova apresentação dos dados de geometria da peca. Os dados produzidos pelo CAD são processados pelo CAM gerando os programas e informações necessárias para as máquinas, para os equipamentos de manipulação de materiais e para os centros de teste e inspeção da qualidade do produto.

Nas operações de usinagem, os sistemas CAD/CAM podem simular os percursos das ferramentas para as diversas operações, de forma que possíveis problemas tais como colisões podem ser detectadas e a programação ser corrigida. Os sistemas CAD/CAM também podem ser capazes de codificar e classificar peças e partes em grupos que tem formas similares usando códigos alfanuméricos.

Devido à existência de uma grande variedade de sistemas CAD/CAM com diferentes características, e fornecidos por diferentes empresas, um modo de troca de informações entre estes diversos sistemas tornou-se um problema significativo. Um padrão de troca de informações para sistemas de CAD é o Initial Graphics Exchange Specification (IGES) um formato neutro que pode ser utilizado compatibilizar diferentes sistemas. Um outro padrão para troca de informações entre diferentes sistemas é o STEP (Standard for the Exchange of Product Model Data).

Para que o sistema de manufatura seja eficiente suas diversas atividades devem ser planejadas. Esta atividade é tradicionalmente feita por planejadores de processo. Esta tarefa quando feita manualmente demanda muito trabalho e tempo, e sua eficiência depende muito da experiência do planejador. O planejamento do processo esta relacionado aos métodos de seleção do modo de produção, escolha de ferramental, máquinas e acessórios; seqüenciamento das operações e montagem. O CAPP, “Computer Aided Process Planning”, realiza esta complexa tarefa visualizando a manufatura como um sistema integrado, então as operações individuais e os passos que envolvem a realização de cada parte são coordenados com outras. O CAPP é particularmente eficaz para pequenos volumes, alta variedade, e operações de montagem.

O CAPP é um importante adjunto do sistema CAD/CAM, ele faz a interligação entre o projeto e a manufatura. Três questões são necessárias para interligar a engenharia de projeto à de manufatura: O reconhecimento dos "features" de usinagem, a manipulação de informações técnicas e a implementação de uma interface neutra. Apesar das realizações na manipulação de informações de manufatura, o reconhecimento de "features" relevantes de formas complexas ainda permanece um desafio. Sem um sistema eficaz de reconhecimento de "features", não é possível se automatizar uma interface sem costura entre o projeto e a manufatura (Kang, Han, Moon,2003).

DNC é um modo pelo qual grupos de máquinas CNC são controladas numericamente, sendo um importante instrumento de automação do chão de fábrica (Zhang et. al.,2001). A definição original de DNC referia-se a Direct Numerical Control, que significava um pequeno grupo de máquinas conectadas a um computador central que transferia e gerenciava os arquivos CN. Com o desenvolvimento das redes de computadores e da tecnologia da comunicação, aliadas à necessidade de automação da produção, a demanda por sistemas controlados numericamente operando em rede tornou-se muito importante atualmente. Com o avanço da tecnologia da informação, funções de controle e monitoramento do processo foram acrescidas as funções iniciais do DNC, e seu significado passou a ser "Distributed Numerical control". Atualmente o conceito de DNC é diferente do seu significado original, ele foi bastante expandido e funções foram acrescentadas além de distribuir e fornecer programas CN para o sistema de produção o DNC é um modo de comunicação enviando também uma variedade de documentos de suporte, a comunicação é bidirecional, funções supervisão, de planejamento do processo e geração de programas CN foram também integradas ao sistema.

Atualmente os sistemas DNC e FMS estão inter-relacionados, o DNC tornou-se uma parte necessária do FMS. Os sistemas flexíveis de manufatura, FMS, se preocupam com a automatização do fluxo de materiais na área produtiva. O sistema DNC tornou-se gradualmente parte constituinte dos sistemas CIM, Computer Integrated Manufacturing, e é atualmente um método de integração de equipamentos e informação.

Black (1991), cita que é fundamental a reestruturação dos sistemas de manufatura antes de sua automação e informatização. O sistema de células interligadas fornece a estrutura apropriada para a informatização, permitindo a ligação deste sistema com as outras partes da empresa.

3 Sistemas de informação

Este capítulo tem o intuito de apresentar os conceitos básicos sobre Sistemas de Informação Computadorizados, SI, uma classificação destes sistemas, seus principais constituintes, as principais tecnologias, e as abordagens para o seu desenvolvimento.

3.1 Sistemas e Informação

Um sistema é um conjunto de elementos dinamicamente inter-relacionados desenvolvendo uma atividade ou função para atingir um ou mais objetivos ou propósitos (Chiavenato,1995).

Um sistema pode compor-se, sucessivamente, de subsistemas que se interelacionam entre si, compondo o sistema maior. O sistema de informação é um subsistema do sistema empresa, e é constituído por um conjunto de subsistemas de informação, por definição interdependentes. Assim pode-se pensar em subsistemas de orçamento, de contabilidade, como componentes do sistema de informação total da empresa (Bio,1991).

Os sistemas de informação coletam, armazenam e processam informações, que são utilizadas na tomada de decisões. Davenport (1998), ao discutir "o que é informação", apresenta que é difícil definir informação. Nos textos de Sistemas de Informação, muitas vezes, é feita uma distinção pouco precisa entre dados, informação e conhecimento. A tabela1 apresenta características úteis para estes o entendimento destes termos. '

Dados, Informação e conhecimento		
Dados	**Informação**	**Conhecimento**
Simples Observações sobre o estado do mundo	Dados dotados de relevância e propósito	Informação valiosa da mente humana inclui reflexão síntese contexto
o Facilmente estruturado	o Requer unidade de análise	o De difícil estruturação
o Facilmente obtido por máquinas	o Exige consenso em relação ao significado	o De difícil captura por máquinas
o Freqüentemente quantificado	o Exige necessariamente a mediação humana	o Freqüentemente tácito
o Facilmente transferível		o De difícil transferência

Tabela 1: Dados, Informação e Conhecimento. Fonte Davenport 1998

Os dados são mais fáceis de gerenciar, ou seja, é fácil de capturar, comunicar, e armazenar.

Pessoas transformam dados em informação, este processo exige análise e pode ser feito com diferentes abordagens. Estas abordagens podem ser questionadas por outras pessoas. Por serem mais complexas as informações também são mais difíceis de serem transportadas. As informações diminuem incertezas, e por isto são utilizadas nos processos decisórios. Conhecimento é a informação mais valiosa e, conseqüentemente, mais difícil de gerenciar. É valiosa precisamente porque alguém deu à informação um contexto, um significado, uma interpretação; alguém refletiu sobre o conhecimento, acrescentou a ele sua própria sabedoria, considerou suas implicações.

Os computadores são ótimos para lidar com dados, mas não são tão adequados para lidar com informações e, menos ainda, com o conhecimento.

3.2 Uma classificação dos sistemas de informação

Com o advento e popularização dos computadores e da tecnologia da informação os SI ganharam grande impulso o que trouxe grandes mudanças na estrutura das empresas e no modo delas operarem.

Dentro de uma organização existem diferentes interesses, especialidades e níveis hierárquicos que são servidos desta forma por diferentes tipos de SI. Laudon (2000), classifica os SI pela especialidade funcional e pelo nível organizacional a que servem Figura 3-1.

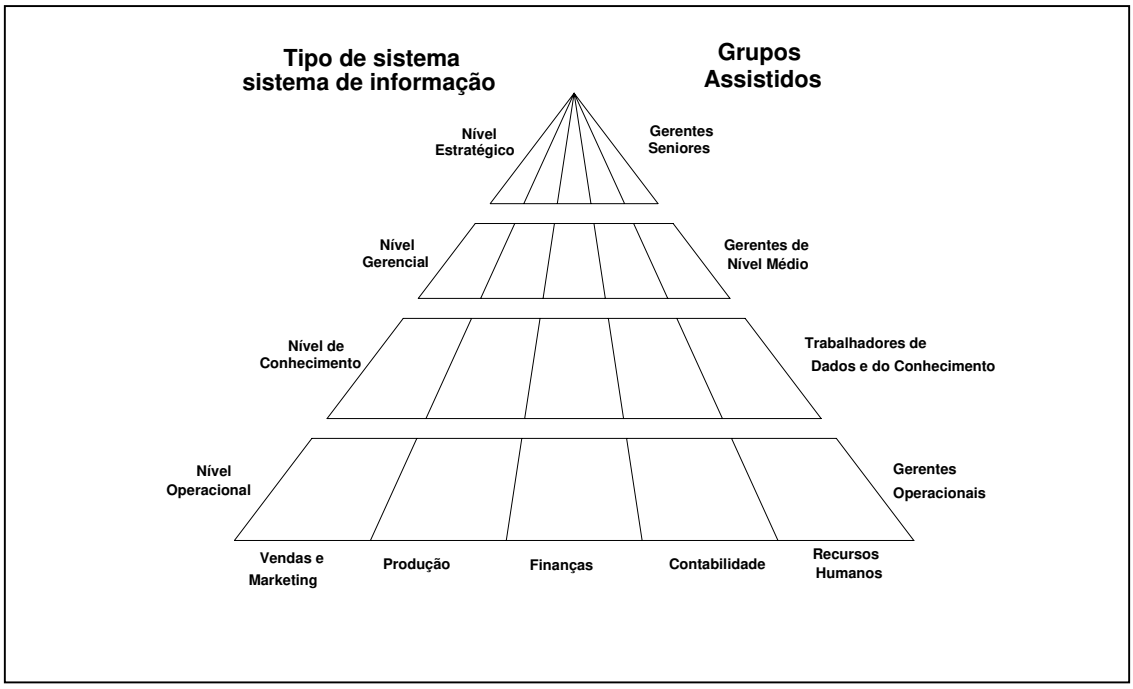

Figura 3-1 Tipos de Sistemas de Informação, Fonte:Laudon, 2001

Os sistemas do nível operacional dão suporte aos gerentes operacionais no acompanhamento das atividades e transações elementares da organização. O principal propósito dos sistemas desse nível é responder a perguntas rotineiras e localizar o fluxo de transações através da organização.

Os sistemas de nível de conhecimento dão suporte aos trabalhadores do conhecimento e de dados de uma organização. O seu propósito é ajudar a empresa a integrar novos conhecimentos no negócio e a controlar o fluxo de documentação.

Os sistemas de nível gerencial são projetados para servir ao monitoramento, ao controle, à tomada de decisão e às atividades administrativas dos gerentes médios.

Sistemas de nível estratégico ajudam a administração sênior a atacar e enfocar assuntos estratégicos e tendências de longo prazo, tanto na empresa como no ambiente externo. Sua principal preocupação é adequar as mudanças no ambiente externo com a capacidade organizacional existente.

As principais funções empresariais como vendas e marketing, produção, finanças, contabilidade e recursos humanos, são servidas por seus próprios sistemas de informação. Em grandes empresas, sub funções de cada uma destas principais funções também têm seus próprios sistemas de informação.

Diferentes organizações têm diferentes sistemas de informações para as mesmas áreas funcionais. Como duas empresas não possuem exatamente os mesmos objetivos, estruturas e interesses, os sistemas de informação demandam grande trabalho de organização e desenvolvimento para se ajustarem às características singulares de cada uma.

Ainda seguindo a classificação de Laudon, existem categorias específicas de sistemas que servem a cada nível organizacional. Desta forma existem os Sistemas de Suporte Executivo no nível estratégico, Sistemas de Informação Gerenciais, SIG, e Sistemas de Suporte à Decisão no nível gerencial, Sistemas de Trabalho de Conhecimento e Sistemas de Automação de Escritórios no nível de conhecimento, e Sistemas Processadores de Transações, SPT, no nível operacional.

Neste trabalho o maior interesse está nos sistemas de trabalho de conhecimento e sistemas processadores de transação voltados para a produção.

Os SPT operam transações, uma transação é qualquer troca de valor ou movimento de mercadorias que afete a lucratividade de uma organização ou seu ganho global, inclusive a realização de metas organizacionais. Os sistemas processadores de transações são fundamentais

para assegurar o movimento normal das operações comerciais, preservar o fluxo de caixa e a lucratividade e dar apoio ao sucesso da organização.

Podemos citar como exemplos de SPT, os sistemas de controle de estoques e ferramental para a produção. Os SPT de forma geral operam, de forma rápida, grande quantidade de dados tanto de entrada como de saída, com alto grau de repetição e através de processos simples. Os dados coletados através dos SPT são armazenados em um ou mais bancos de dados. Estes bancos de dados associados são de grande capacidade e normalmente relacionais armazenando as informações em tabelas. Estes sistemas necessitam de auditorias para assegurar que a alimentação dos dados, processamento, procedimentos de entrada e saída estejam corretos, precisos e válidos. Têm grande potencial de problemas relacionados à segurança. Seus serviços atendem a um grande número de usuários de forma que devem prover confiabilidade para evitar impactos negativos à organização devido à pane ou falhas de operação. Devem manter um alto grau de precisão e integridade em seus dados, e produzir documentos e relatórios confiáveis. Estes dados podem ser acessados por outros sistemas para processamento adicional ou manipulação, por exemplo, por sistemas de informação gerenciais, de tal forma que têm forte impacto sobre quase todos os outros sistemas de informação e processos de tomada de decisões na organização.

Sistemas de Trabalho do Conhecimento suprem a necessidade de informação no nível de conhecimento da empresa. Em geral, trabalhadores do conhecimento são pessoas que possuem níveis universitários formais e que freqüentemente são membros de uma profissão reconhecida, como engenheiros, médicos, advogados e cientistas. Suas tarefas consistem principalmente em criar informação e conhecimentos novos e asseguram que esse novo conhecimento e especialidades técnicas sejam corretamente integrados ao negócio. Exemplos de Sistemas de Trabalho do Conhecimento são sistemas CAD, CAE, CAM, sistemas de robótica, monitoramento e gestão da manufatura (CIM).

3.3 Administração de sistemas de informação

A administração de Sistemas de informação é uma área relativamente recente, e que ainda passa por mudanças intensas e rápidas. É uma área que faz uso intensivo da tecnologia. Apesar deste viés tecnológico, o interesse da administração dos sistemas de informação é mais amplo, enfatiza o ambiente da informação em sua totalidade, levando em conta a cultura e a política

empresarial, estuda como as pessoas usam a informação e o que fazem com ela nos processos empresariais, buscando uma maior eficiência e eficácia na tarefa de informar, ou seja, produzir informação atualizada e na quantidade certa, adequada às pessoas que irão utilizá-la.

O campo de atuação do administrador de sistemas de informação é muito vasto, e abrange os níveis estratégicos, táticos e operacionais da empresa. Podemos citar como algumas de suas responsabilidades: articular e dirigir políticas relacionadas à informação, planejar, dirigir e avaliar os sistemas de informação, administrar o projeto, o desenvolvimento e a implantação de novos sistemas, prestar consultoria, desenvolver métodos de obtenção, troca, análise, divulgação e documentação de informações, eletrônicas ou não, internas ou externas à empresa.

3.4 Estruturas organizacionais e os Sistemas de informação

Uma empresa quando cresce e prospera, devido à diferenciação interna, assume uma estrutura organizacional. Assim, há forçosamente, o reagrupamento das atividades e o estabelecimento de novos níveis hierárquicos, com as conseqüentes redefinições de autoridade/responsabilidade. A estrutura organizacional cresce no sentido vertical em função dos limites da capacidade de supervisão em cada nível. Os sistemas, por sua vez, requerem fluxos de coleta dos dados que não se restringem às funções, e sim, permeiam as funções necessárias dentro da estrutura organizacional da empresa, ampliando-se no sentido horizontal.

Existe uma relação de interdependência entre os sistemas de informação e a estrutura organizacional. Se ambos estão inexoravelmente interligados, as alterações, as mudanças, o replanejamento de qualquer deles afetarão o outro (Bio, 1991).

Desta forma a implantação de sistemas de informação pode trazer impactos sobre a estrutura organizacional, tais como: levar a eliminação de funções, transferência de funções de uma área para outra, mudança da natureza e da forma de trabalho de uma função, outras funções podem ser criadas, pode surgir à necessidade de reagrupamento de funções existentes e de revisão de autoridade/responsabilidade. Outra conseqüência da implantação dos sistemas de informação é o menor crescimento da estrutura organizacional ao longo do tempo. Um exemplo disto é a técnica que utiliza os sistemas de informação como um elemento-chave para as empresas delegarem poder a seus empregados. Isto significa tornar os sistemas de informação disponíveis

aos empregados de níveis inferiores na hierarquia. Significa dar aos empregados, trabalhadores e gerentes mais poder, responsabilidade e autoridade para tomarem decisões, adotarem certas atitudes e terem mais controle sobre seus trabalhos. Esta técnica geralmente resulta em ações mais ágeis e uma rápida solução de problemas, podendo também reduzir custos e gerar produtos e serviços de melhor qualidade (Stair, 1998).

São relevantes, também, os reflexos dos problemas organizacionais sobre o desenvolvimento de sistemas. Muitas empresas defrontam-se com os mais variados tipos de problemas organizacionais, ou seja, agrupamento inadequado de funções, indefinições de autoridade/responsabilidade, funções executadas em duplicidade, estruturas produtivas inadequadas. Estes fatos devem ser levantados e considerados na análise dos sistemas ao planejar um sistema de informação, pois a simples introdução de SI será ineficaz em tratar destes problemas organizacionais. O agrupamento inadequado de funções resulta numa excessiva departamentalização, o reflexo desse fato se faz sentir por um fluxo de informações lento e confuso. Autoridade e responsabilidades vagas e difusas significam dificuldades para determinar o cumprimento de novos procedimentos, prazos, critérios, correções. As dificuldades normais de implantação podem tornar-se muito mais complexas pelo simples fato de que não conseguimos detectar quem são os responsáveis pela solução de um determinado problema, quem tem autoridade para resolvê-lo. O problema de se detectar funções em duplicidade, ao se desenvolver um sistema de informação, leva a uma reformulação organizacional profunda que deverá preceder este sistema. Estruturas produtivas inadequadas devem ser reformuladas para então os SI poderem ser implantados.

Buscando-se sintonizar as informações com as necessidades dos diferentes níveis de autoridade torna-se evidente a exigência de inter-relacionamento entre a estrutura organizacional e os sistemas de informação e a necessidade de integrar as decisões sobre a organização com as decisões sobre os sistemas de informação (Bio, 1991).

Um problema detectado com os SI ocorre quando se confronta o sentido vertical da organização com o sentido horizontal dos sistemas. Observa-se que os responsáveis pelas funções são facilmente identificáveis. Por exemplo, quem é o chefe do setor, o encarregado da seção. Já os dados que devem ser processados por um sistema normalmente se originam em vários pontos da estrutura. Isto faz com que não seja facilmente identificável o responsável pelo sistema. O que

várias empresas têm feito, e com sucesso, é determinar o usuário principal, aquele que se apóia diretamente no sistema, como seu dono funcional e administrador.

3.5 Sistemas de Informação na Manufatura

Com a ênfase no aumento da qualidade e da produtividade nos processos industriais, a manufatura tem recebido forte impacto dos grandes avanços da tecnologia da informação refletindo no uso crescente de sistemas computacionais em todos os seus níveis, desde o estratégico até o operacional.

Os sistemas de informação proporcionam a infraestrutura empresarial para integrar as operações de manufatura com seus processos de negócio relacionados. Eles são necessários para produzir a informação certa que indique que produtos fazer, quando fazê-los, e então fazê-los o mais rapidamente possível. Além disto, as operações de manufatura tem crescido globalmente e uma apropriada coordenação entre os negócios e as unidades de manufatura nas cadeias de agregação de valor global é necessária. Os sistemas de informação podem ajudar a proporcionar esta coordenação, Shaw (2000).

Os sistemas de informação são também de crucial importância para lidar com o fluxo de informações internas à organização principalmente para as operações que lidam com o fluxo e conversão de materiais por toda a empresa. Os SI são importantes para colher e processar informações sobre o plano estratégico e políticas corporativas tais como expansão da capacidade industrial, novas instalações, políticas de estoque e novos programas e parâmetros de qualidade. No âmbito operacional auxiliam no recebimento, inspeção, aceitação e expedição de todas as matérias primas e suprimentos, definição da quantidade necessária e o uso de matérias primas, dados de pessoal, folhas de pagamento, controle sobre cartões de ponto, levantamento e cálculo de custo, tabelas de tempo, dados de pedidos, planejamento das necessidades de matéria prima.

Os sistemas de informação também estão intimamente relacionados à integração da manufatura, integração significa que os processos de manufatura, operações, e seu gerenciamento são tratados como um sistema, procurando incrementar a produtividade, a qualidade dos produtos, a sua confiabilidade e reduzindo seus custos. Segundo Agostinho (1995), um sistema

de manufatura é, na sua essência, um sistema de informações. O seu nível de integração depende, essencialmente, da sinergia do fluxo de informações.

A manufatura integrada por computador, CIM, representa a tentativa de integrar todas as aplicações computacionais dentro de uma empresa de manufatura em um todo coerente. Ferneda (1999), apresenta a CIM como uma estratégia de manufatura que utiliza várias tecnologias auxiliadas por computador, para realizar a automação do sistema de manufatura como um todo. Ela consiste dos seguintes componentes básicos: máquinas-ferramenta e equipamentos relacionados, sistemas de manipulação de materiais, sistemas computacionais, trabalho humano. O Foco da CIM esta na informação como um elemento crucial para a ligação entre todas as facetas de uma empresa de manufatura. Na CIM, as funções de pesquisa e desenvolvimento, projeto, produção, montagem, inspeção e controle de qualidade são interligados. Conseqüentemente, a integração requer que relações quantitativas entre o projeto do produto, materiais, processos de manufatura, capacidades de equipamento e suas atividades relacionadas devem ser bem compreendidas. Os sistemas CIM também podem aumentar a eficiência através da coordenação de ações de várias unidades relacionadas à produção.

A CIM é uma metodologia e um objetivo e não uma montagem de computadores e equipamentos. A CIM deve conter um extensivo banco de dados sobre os diversos aspectos técnicos e de negócio de operação que é compartilhado por todo o sistema de manufatura. A CIM pode ser proibitiva em termos de custo particularmente para pequenas e médias empresas

A tecnologia para implementar a CIM atualmente é bem compreendida e prontamente disponível, porém pode demandar grandes investimentos. Apesar de não ser uma panacéia, a CIM representa uma potente força de mudança dentro da indústria. Conviver no ambiente competitivo atual, sem ela, pode representar uma considerável desvantagem. Em geral as empresas de pequeno e médio porte deveriam adotar a CIM a menos que fosse financeiramente impraticável. Para tanto, elas precisam ter uma clara e coerente estratégia e um número de alianças estratégicas compatíveis e complementares (Vassel,1999).

Os benefícios da CIM são: Melhor resposta à demanda por menores ciclos de vida de produto, mudanças de mercado e competição global; maior ênfase na qualidade e uniformidade dos produtos implementados por um melhor controle do processo; melhor uso de materiais,

maquinaria e pessoal; redução de materiais em processo; melhor controle da produção, planejamento e gerenciamento das operações de manufatura. Estas melhorias implicam em um incremento de produtividade e menor custo do produto.

Um eficiente sistema CIM normalmente requer uma base de dados centralizada o qual é compartilhada por toda a organização. Esta base de dados geralmente contém: Dados de produto tais como: Formas das peças, dimensões e especificações. Dados de gerenciamento de atributos tais como: Proprietários, versões e componentes. Dados de produção tais como: Os processos de manufatura envolvidos para a produção das peças e produtos. Dados operacionais tais como: Cronogramas, lotes, requerimentos de montagem. Dados de recursos tais como: Capital disponível, máquinas, equipamentos, ferramental, pessoal. Estas bases de dados são alimentadas por sistemas supervisórios que adquirem dados por meio de vários sensores nas máquinas e equipamentos usados na produção. Com isto é possível obter informações em tempo real do número de peças produzidas por unidade de tempo, problemas na produção, manutenções, dados de inspeção e outros detalhes que sejam pertinentes. Estes dados são importantes para análises estatísticas, relatórios e prognósticos de demanda.

A CIM é a base conceitual para a integração de aplicações e do fluxo de informações do projeto de produtos, planejamento do processo, planejamento da produção e dos processos de manufatura. A meta do CIM é agrupar todos estes aspectos da produção. Os projetos dos produtos são realizados nos sistemas CAD que são ferramentas poderosas para o projeto e a modelagem geométrica de produtos e componentes. No sistema CAD, o projetista pode desenhar o produto e seus componentes mais facilmente e interativamente. Assim, tendo um contínuo retorno por meio do vídeo de resolução gráfica, ele pode facilmente considerar projetos alternativos ou mesmo modificar desenhos rapidamente para encontrar soluções que atendam aos requisitos desejados. O projetista pode realizar análises de engenharia para identificar potenciais problemas. A velocidade e precisão de tais análises superam àquelas disponíveis nos métodos tradicionais. Além de dados de projeto, como geometria e dimensões, são produzidas outras informações como listas de materiais, especificações, e instruções de manufatura. O trabalho desenvolvido pelo CAD é então processado pelo CAM gerando os dados e instruções necessárias para operar e controlar as máquinas de produção, os equipamentos para a manipulação de materiais e equipamentos automatizados de inspeção para o controle de qualidade. Nas operações

de usinagem, uma importante função do CAM é descrever os percursos das ferramentas nas várias operações, tais como os programas CN para centros de torneamento e centros de usinagem. Estes percursos são gerados automaticamente e podem ser modificados pelos programadores das máquinas para otimizar os percursos das ferramentas. Os engenheiros podem verificar visualmente através de simulações estes percursos para evitar possíveis colisões com fixadores ou outros dispositivos.

Enquanto a informação geométrica é criada na atividade de projeto as informações de manufatura dizem respeito ao planejamento do processo planejamento da produção e às operações da planta. Dado a geometria de uma peça, o Computer Aided Process Planning (CAPP), a ponte entre o CAD e o CAM, gera um conjunto seqüenciado de instruções para produzir a peça especificada. Para fazer isto o CAPP tem que extrair as informações de manufatura tais como as formas elementares para a usinagem, os "features" , as especificações de precisão incluindo a rugosidade superficial , as tolerâncias dimensionais e geométricas a fim de selecionar os processos necessários e determinar as condições de operação. Apesar do grande esforço já realizado para interligar o projeto com o planejamento do processo, o compartilhamento de informações de manufatura ainda permanece um gargalo (Kang, Han, Moon,2003).

O CAPP é particularmente efetivo para produções em pequeno volume, alta variedade que requerem usinagem, fundição e operações de montagem. Há dois tipos de sistemas CAPP, os variantes e os generativos. No modelo variante, o sistema contém um plano de processo padrão para a peça a ser manufaturada. A pesquisa é então feita no sistema baseada na forma da peça e nas suas características de manufatura. O plano padrão é então recuperado apresentado e revisto. O plano de processo contém informações sobre tipos de ferramentas e máquinas a serem utilizadas, seqüência das operações de manufatura, dados tecnológicos dos processos, o tempo necessário para cada seqüência. Se um plano de produção não existe para a peça em questão um semelhante é adaptado ou um novo é feito e armazenado no sistema.

No sistema generativo um plano de processo é gerado automaticamente com base nos mesmos procedimentos lógicos que seriam seguidos por um planejador de processo para uma peça particular. Este sistema é complexo porque deve conter conhecimento detalhado sobre a forma da peça e suas dimensões, capacidades do processo, seleção de métodos de manufatura,

maquinaria, ferramental e seqüenciamento de operações que deverão ser realizadas. Estes sistemas computacionais são conhecidos como sistemas especialistas.

O sistema generativo é capaz de criar um novo plano ao invés de ter que modificar um já existente.Embora este sistema seja menos comum que o sistema variante ele apresenta maior flexibilidade e consistência para a realização de planejamento de processos de novas peças e qualidade maior devido à capacidade de decisão lógica.

Os sistemas CAPP apresentam melhor padronização do processo, com melhoria desta tarefa, menores tempos para a sua realização, menor custo, e incremento na consistência da qualidade do produto e sua confiabilidade.

Outro tipo de sistemas de informações, muito utilizados nas empresas industriais são os sistemas de gestão denominados ERP (Enterprise Resource Planning). Eles originaram-se dos sistemas de gerenciamento das necessidades de materiais (MRP). A partir da necessidade de produtos finais, os sistemas MRP calculam quanto e quando produzir e comprar as diversas matérias primas (Zancul et. Al., 1999). Esses novos sistemas, ERP, integraram aos módulos de produção módulos que ultrapassam os limites da manufatura tais como: gerenciamento de recursos humanos, vendas e distribuição, finanças, controladoria, e outros, capazes de suportar as necessidades de informação de diversos processos de negócio. Os sistemas ERP são compostos por uma base de dados única e por módulos que suportam diversas atividades dos processos de negócio das empresas. Estes módulos assistem os responsáveis por estas atividades a planejar, monitorar e controlar os negócios da empresa. Os sistemas ERP por utilizarem uma base de dados única propiciam a integração entre os diversos elementos do sistema de informação e evitam redundância e inconsistência nos dados. Correa (1998), sugere que erros freqüentes na implantação explicam porque elas custam tão caro e o grande número de insucessos neste processo. Apresenta também que a implantação de sistemas ERP deve ser gerenciada por pessoas que entendam de mudança organizacional e de negócio, deve ser feita por pessoal interno a própria empresa, possivelmente facilitado em situações pontuais por capacitação externa, deve contar com o comprometimento da alta direção, deve ter uma visão clara e compartilhada de situação futura.

3.6 Tecnologia da Informação

Com o advento dos computadores e principalmente após o surgimento do microcomputador pessoal, lançado no início da década de oitenta, houve a implantação de sistemas computacionais nas mais variadas áreas de atuação humana. Estes computadores, seus programas e a comunicação entre eles fizeram surgir uma nova e diversificada tecnologia denominada de forma generalizada de tecnologia da informação.

A tecnologia de informação pode promover vários graus de mudança nas empresas, indo desde uma mudança marginal a mudanças de longo alcance. Laudon (2001), apresenta quatro tipos de mudança organizacional que foram possibilitadas pela tecnologia de informação: automação, racionalização, reengenharia e mudanças de paradigmas. Eles têm um grau crescente de recompensas e riscos. A automação consiste basicamente em se automatizar através de sistemas de informação, processos realizados manualmente ou de forma semimanual tal como automatizar uma operação de inspeção de qualidade. Os ganhos em produtividade estão assegurados, porém são modestos. Um passo adiante seria o da racionalização. Com a automação, freqüentemente surgem novos gargalos na produção e tornam os arranjos existentes em procedimentos e estruturas dolorosamente incômodos. A racionalização é a busca do ajuste dos procedimentos operacionais padrões, eliminando os gargalos óbvios, de forma que a automação possa tornar os procedimentos mais eficientes. Um tipo mais poderoso de mudança organizacional é a reengenharia empresarial, na qual os processos empresariais são analisados, simplificados e reprojetados. A reengenharia envolve a reconsideração radical do fluxo de trabalho e dos processos empresariais usados para produzir produtos e serviços, com a mente voltada para a redução radical de custos dos negócios. A mudança de paradigma leva, utilizando SI, à mudança da estrutura organizacional da empresa como um todo, transformando o modo como a organização conduz seus negócios ou até mesmo a natureza do negócio como um todo. Pode-se citar como exemplos de mudança de paradigmas, a vendas de computadores e automóveis pela internet.

Este novo modelo de negócios substitui o antigo modo de trabalhar com grandes lojas e grandes estoques para visitação por sistemas de informação. Este modo ágil e cômodo leva a um novo modo de se pensar a empresa e o negócio.

Redes de computadores e a Internet

Uma rede de computadores tipicamente contém componentes de Hardware e Software que necessitam trabalhar em conjunto para transportar informação. Os diferentes componentes em uma rede podem se comunicar aderindo a um conjunto comum de regras que os habilitam a conversar entre si. Este conjunto de regras e procedimentos que regem a transmissão entre dois pontos numa rede é denominado protocolo.

Numa rede corporativa, o hardware, o software, as telecomunicações e os recursos de dados da organização são organizados para dar maior poder de computação aos computadores e para criar uma rede ao redor da empresa ligando redes menores. Essas redes corporativas podem estar ligadas a redes de outras organizações e à internet. A rede corporativa pode aumentar a produtividade e ser uma vantagem competitiva se as comunicações de dados puderem ser feitas sem problemas pelo emaranhado de redes eletrônicas da organização, conectando tipos diferentes de máquinas, pessoas, sensores, bancos de dados, divisões funcionais departamentos e grupos de trabalhos. Essa capacidade de computadores e dispositivos baseados em computador se comunicarem entre si e compartilharem informação de modo significativo sem a intervenção humana é chamada de conectividade.

Existem diversos modelos para se alcançar conectividade nas redes de comunicação. O modelo “Transmition Control Protocol/ Internet Protocol”, TCP-IP por ser usado na Internet tem alcançado grande utilização. O Modelo TCP-IP foi desenvolvido pelo Departamento de Defesa dos Estados Unidos em 1972 e seu objetivo foi ajudar cientistas a unir diferentes computadores.

A internet é uma rede distribuída, os seus custos são baixos por não ter objetivos de lucros financeiros. Os fluxos de informações são encaminhados e direcionados pelas redes que a compõem de forma que o custo é sempre de chamada local. Ela esta baseada no protocolo TCP-IP e é baseada na arquitetura cliente/servidor. Os servidores dedicados à internet são o coração da informação na rede.

A conectividade global da internet e sua facilidade de uso podem fornecer às empresas acesso a negócios ou a pessoas que normalmente estariam fora do seu alcance. Ela está se tornando rapidamente a tecnologia de escolha para o comércio eletrônico porque ela oferece às

empresas um modo fácil de se ligar a outros negócios e indivíduos a um custo muito baixo. Os problemas de segurança têm se mostrado o grande desafio para a ampla utilização da internet como meio de comércio, segundo a Pesquisa Nacional de Segurança da Informação (2002), 66% dos usuários deixam de comprar pela internet por causa da sensação de falta de segurança, e entre as empresas que admitem ter sofrido problemas com segurança da informação 55% apontaram a internet como o principal ponto de invasão.

As organizações podem usar padrões internet de rede e de tecnologia Web para criar redes particulares chamadas intranets. As intranets podem criar aplicativos de rede que podem rodar sobre muitos diferentes tipos de computadores por toda a organização. Algumas empresas permitem que pessoas e organizações externas a ela tenham acesso limitado a suas intranet. As intranets particulares que são estendidas a visitantes de fora da empresa são chamadas de extranets. As extranets são especialmente úteis para ligar as organizações com clientes ou parceiros comerciais. Elas são usadas freqüentemente para fornecer a disponibilidade dos produtos, preços e dados de embarque e intercambio eletrônico de dados.

3.6.1 Organizando Dados e Informação

Sem os dados em mídia eletrônica e a capacidade de processá-los, uma organização não teria condições de completar com sucesso a maioria de suas atividades empresariais. Dados são registros, observações e anotações, sobre o estado do mundo. Os dados, armazenados em computadores, são geralmente organizados em uma hierarquia, que começa com o menor dado usado pelos computadores, o byte, e vai progredindo. Os bits, dígitos binários unidade básica de armazenamento de informação, podem ser organizados em unidades chamadas bytes, um byte corresponde a oito bits. Cada byte representa um caractere, que é o bloco básico da construção da informação. Um caractere pode consistir de letras, dígitos e símbolos especiais. Os caracteres formam os campos. Um campo é tipicamente um nome, um número ou uma combinação de caracteres que, de alguma forma, descreve um aspecto de um objeto empresarial. Uma coleção de campos relacionados forma um registro. Uma coleção de registros relacionados é um arquivo. No nível mais alto desta hierarquia esta o banco de dados, uma coleção de arquivos integrados e relacionados.

⇒ Tipos de Banco de Dados

Para que um banco de dados possa ser utilizado, os dados que o compõem devem ser estruturados segundo uma determinada ordem. Os banco de dados foram criados com sucesso com diferentes modelos tais como: hierárquicos, em redes, relacionais, orientados a objetos. Os primeiros bancos de dados basearam-se no modelo de redes ou no modelo hierárquico. O modelo hierárquico apresenta os dados numa estrutura em árvore. Um segmento superior está logicamente conectado a um segmento mais baixo, numa relação pai-filho. Um segmento pai pode ter mais de um filho, mas um filho somente um pai. Tanto no sistema gerenciador de banco de dados hierárquico como no em redes os dados são vinculados fisicamente entre si por uma série de ponteiros que formam cadeias de segmentos de dados relacionados. O modelo de dados em rede é uma variação do modelo hierárquico de dados. Enquanto as estruturas hierárquicas descrevem relacionamentos de um para muitos, as estruturas de rede descrevem os dados logicamente como relacionamentos de muitos para muitos. Os segmentos pais podem ter múltiplos filhos, e um filho pode ter mais de um pai. Os banco de dados hierárquicos e em rede são usados extensivamente em topologias centralizas baseadas em computadores de grande porte, os chamados “mainframes”. A principal vantagem dos modelos hierárquicos e de redes é a eficiência do processamento, porém eles não são facilmente modificados, apresentando pouca flexibilidade.

Um banco de dados relacional consiste em uma coleção de tabelas. É construído geralmente a partir de um modelo conceitual. Este modelo é uma representação em alto nível de um empreendimento. Nele são identificados os objetos importantes, as entidades, junto com seus atributos e as relações entre estas diversas entidades.Tanto as entidades como as relações são posteriormente representadas por meio de tabelas. Cada linha de uma tabela contém os valores dos n atributos de um elemento em particular. Uma linha individual é denominada Tupla. Uma Tupla numa tabela representa um relacionamento entre um conjunto de valores. Uma vez que uma tabela é uma coleção de relacionamentos, existe uma correspondência entre o conceito de tabela e o conceito matemático de relação. Por essa correspondência entre tabela e relação, originou-se o nome "modelo relacional" (Silberschatz, 1999).

Os Bancos de Dados Relacionais são os mais utilizados atualmente, e devido a sua importância são analisados no próximo tópico deste capítulo.

Recentemente a tecnologia de Banco de Dados tem sido utilizada em aplicações fora do reino convencional do processamento de dados e, em geral não mantém as características típicas dos bancos de dados relacionais. Podemos citar como exemplo destas aplicações o projeto auxiliado por computador, CAD, onde o sistema armazena dados pertencentes ao projeto de engenharia, incluindo os componentes do item que está sendo projetado, o inter-relacionamento de componentes e versões antigas de projeto. As estruturas complexas que compõe um projeto fogem das características comuns das estruturas convencionais onde os itens consistem de registros de comprimento fixo, normalmente pequeno com campos atômicos, ou seja, sem estrutura. Outros casos também falham na aplicabilidade de sistemas de banco de dados tradicionais tais como: aplicativos multimídia, sistemas de informação de escritórios, bancos de dados de hipertexto, inteligência artificial e engenharia de software auxiliada por computador. Nestes casos os bancos de dados orientados a objetos são indicados.

Os bancos de dados baseados em objetos surgiram da necessidade de suportar a programação baseada em objetos. Os programadores de Smalltalk e C++ precisavam de um depósito para o que eles chamavam de dados persistentes, ou seja, dados que permanecessem após a conclusão de um processo (Martin,1994). Neste modelo os objetos são armazenados diretamente no banco de dados sem a necessidade de conversão dos objetos em outro formato. Desta forma os objetos tornam-se persistentes e plenamente acessíveis pelas linguagens e seus programadores o que evita abordagens diferentes entre a programação e manipulação dos objetos para o armazenamento das informações a eles relacionados.

Um Sistema de Banco de Dados Orientado a Objetos, SBDOO, é basicamente um sistema em que a unidade de armazenamento é o objeto, com o mesmo conceito das linguagens de programação orientadas a objetos. A diferença fundamental é a persistência dos objetos, ou seja, os objetos continuam a existir após o encerramento do programa (Nassu,1997) .

Os SGBOO melhoram o entendimento das consultas e as tornam mais próximas da visão que o usuário tem do problema. Assim a representação do domínio do problema e a manipulação de sua representação informacional se tornam mais intuitiva e simples.

Ruschel (1996), analisa em seu trabalho uma lista de SBDOO, produtos desenvolvidos por diferentes empresas, e indica a inexistência de um modelo padrão de objetos entre os SBDOOs

citados o que representa uma grande desvantagem dos Sistemas de Banco de dados orientados a objetos. Tal falha prejudica a portabilidade e interoperabilidade dos SBDOOs, desmotivando a aceitação e adoção pelos seus potenciais usuários.

Surgiram três abordagens diferentes que associam o paradigma orientado a objetos a Banco de dados, ou à persistência de objetos. Uma abordagem partiu de grupos de linguagens de programação que procuraram incorporar a persistência aos objetos com que operavam. Outra abordagem surgiu de grupos de banco de dados, que procuraram acrescentar aos sistemas de Banco de dados a capacidade de modelagem de objetos complexos e uma linguagem orientada a objetos. Outra abordagem partiu de grupos que tentaram algo novo que oferecia de forma nativa, sem adaptações, as duas facilidades (Mendes,1996).

Os sistemas de banco de dados baseados no modelo relacional orientado a objetos, fornecem um caminho de migração conveniente para usuários de bancos de dados relacionais que desejam usar características orientadas a objetos. Estes sistemas de banco de dados permitem que objetos complexos sejam armazenados como tuplas de tabelas. Estes objetos podem ser constituídos por outros objetos, e a manipulação destes objetos e suas particularidades ficam bastante simplificadas porque eles não perdem a idéia de todo. O mesmo não ocorre em um modelo relacional, quando os atributos de uma mesma entidade se fragmentam quando somos obrigados a armazená-los em Tuplas diferentes.

Alguns sistemas modernos baseados em objetos tentam integrar uma linguagem baseada em objetos com um banco de dados em um único pacote inteiriço. As operações podem ser definidas para cada classe de objetos, mas o programador não precisa ler e gravar explicitamente as informações na memória permanente.

$\Rightarrow$ Gerenciamento de informações

Um dos meios mais básicos para gerenciamento de dados é por meio de arquivos independentes. Assim, os dados sobre cada problema particular são coletados e gerenciados através de uma aplicação específica. Para cada aplicação em particular, um ou mais arquivos de dados são criados. Esta abordagem tradicional faz com que os dados de áreas diferentes sejam tratados de forma independente. Isto favorece a duplicação de dados nas diferentes aplicações,

dados que podem com o tempo se tornar conflitantes, quebrando a confiança de que os dados usados em um aplicativo serão os mesmos nos demais. A operação eficiente de um negócio requer um alto grau de integridade de dados, o que faz com que esta abordagem se aplique apenas a sistemas de pequeno porte.

Os gerenciadores de banco de dados passaram a utilizar e compartilhar uma mesma coleção de arquivos inter-relacionados. Esta abordagem tem um maior controle da redundância de dados, utiliza mais eficientemente o espaço de armazenamento, aumenta a integridade dos dados, e dá uma maior flexibilidade no uso dos dados. Além disto oferece a capacidade de compartilhar as fontes dos dados e informações, um fator vital na coordenação de respostas globais das organizações. Este software gerenciador de dados é conhecido por Sistema Gerenciador de Banco de Dados, SGDB. O principal objetivo de um SGDB é proporcionar um ambiente tanto conveniente quanto eficiente para a recuperação e armazenamento das informações do banco de dados.

Um banco de dados deve ser projetado para armazenar todos os dados relevantes para empresa e fornecer acesso rápido e modificações fáceis. Stair (1998), cita que para se construir um Banco de Dados deve-se tratar as seguintes questões:

- Que dados devem ser coletados e a que custo?
- Que dados devem ser fornecidos a qual usuário?
- Como os dados devem ser arranjados de forma que façam sentido para um determinado usuário?
- Onde os dados devem estar fisicamente localizados?

3.6.1.1 Banco de dados relacionais

O modelo lógico relacional criado por Edgar F. Codd, nos anos 70, começou a ser realmente utilizado nas empresas a partir dos meados dos anos 80. A abordagem relacional está baseada no princípio de que as informações em uma base de dados podem ser consideradas como relações matemáticas e que estão representadas de maneira uniforme, através do uso de tabelas

bidimensionais. Este princípio coloca os dados dirigidos para estruturas simples de armazenar dados, as tabelas, e nas quais a visão do usuário é privilegiada.

Esta visão de dados organizados em tabelas oferece um conceito simples e familiar para a estruturação dos dados, sendo um dos motivos do sucesso de sistemas relacionais de dados. Certamente, outros motivos para esse sucesso incluem o forte embasamento matemático por trás dos conceitos utilizados em bancos de dados relacionais e a uniformização da linguagem de manipulação de sistemas de bancos de dados relacionais através da linguagem, "Structured Query Language", SQL. Em 1982, o "American National Standard Institute", ANSI, tornou a SQL a linguagem padrão para sistemas de gerenciamento de bancos de dados relacionais. A SQL é formada por um conjunto bem simples de sentenças em inglês, oferecendo um rápido e fácil entendimento. Além disto ela é oferecida em praticamente todos os SGBD. Com isto as aplicações podem ser realizadas em ambientes diferentes sem que seja necessária uma reciclagem profunda da equipe de desenvolvimento.

Atualmente os bancos de dados relacionais são predominantes nas empresas. Eles são indicados para situações nas quais as estruturas dos dados são relativamente estáveis, e volumes grandes de dados de pouca complexidade devem ser armazenados e processados. Uma das razões da popularidade dos sistemas relacionais é a sua facilidade de manipulação e entendimento (Machado, 1996).

O projeto dos bancos de dados é feito normalmente a partir de uma modelagem do empreendimento em questão. Um modelo conceitual é uma tentativa de se capturar as características importantes ou representar o empreendimento em questão (Gersting, 1998). A modelagem tem sido aplicada como meio para a obtenção das estruturas de dados que levem ao projeto de banco de dados, porém, esta modelagem também busca representar um ambiente observado, servir como instrumento para comunicação, favorecer o processo de verificação e validação do modelo, ajudar a melhor capturar os aspectos dos objetos observados e os relacionamentos entre estes objetos, servir como referencial para o projeto e a programação e para outros modelos, definir conceitos que devem permanecer consistentes.

A obtenção de um modelo lógico a partir de um modelo conceitual pode ser feita pela aplicação de um conjunto de regras bem definidas. Estas regras fazem a derivação do modelo

conceitual em um modelo relacional. No modelo relacional, os dados são representados na forma de tabelas (relações), ou seja, através de linhas (tuplas) e colunas (domínios). Não podem existir dois elementos iguais dentro de uma relação. Uma chave designa o conceito de item de busca, ou seja, um dado que será empregado nas consultas à base de dados. A chave primária é o atributo de uma tabela que identifica univocamente uma tupla. Uma chave estrangeira é uma chave primária de uma tabela que aparece repetida em outra, elas são os elos de ligação entre as tabelas. Resumidamente é um modo para o estabelecimento do relacionamento entre duas tabelas.

Cougo (1997), apresenta a obtenção de um modelo relacional por meio de três atividades distintas: normalização das estruturas de dados, derivações das agregações e das generalizações especializações, e derivação dos relacionamentos. A normalização tem um caráter organizativo e está baseada na aplicação sistemática das formas normais que são regras para verificação, validação e ajuste das estruturas de dados. A derivação das agregações, generalizações e especializações é a estratégia para reduzir estas estruturas consideradas especiais, para estruturas convencionais (entidades relacionamentos). A derivação dos relacionamentos é bastante simples e poderá ser feita automaticamente por ferramentas CASE. As derivações dos relacionamentos são feitas a partir da análise dos relacionamentos um para um, um para muitos, e muitos para muitos, com e sem atributos.

O modelo relacional deve proporcionar a integridade dos dados, ou seja, não podem ocorrer dois dados logicamente incompatíveis sobre um mesmo fato provocando inconsistências e causando a perda da confiabilidade das informações contidas no banco de dados. Para isto são impostas as seguintes regras de integridade:

- Integridade de entidade: uma chave primária não pode conter um valor nulo.Essa restrição simplesmente confirma que cada tupla tem que ter um valor em sua chave primária para distinguí-la das outras e que todos os atributos na chave primária são necessários para identificar, univocamente, uma tupla.

- Integridade dos dados: Os valores de um atributo devem, de fato, pertencer ao domínio do atributo.

- Integridade referencial: Se uma determinada tabela A possui uma chave estrangeira que é chave primária em uma outra tabela B, então ela deve ser igual a um valor da chave primária existente em B ou ser nula. Em outras palavras, não pode existir na chave estrangeira, um valor que não exista na tabela na qual ela é chave primária.

⇒ Bancos de Dados e os Processos de Fabricação

Devido à entrada de Máquinas Ferramentas CNC na produção os setores produtivos das empresas tiveram que trabalhar com uma quantidade de informações cada vez maior sobre materiais, ferramental e parâmetros dos processos de usinagem.

Boehs (2002), cita que a informatização pode ser um fator importante para a resolução de muitos problemas ligados à organização das ferramentas, pois esta permite que informações sejam registradas, atualizadas e acessadas mais fácil e rapidamente por todos os setores que delas necessitam fazer uso.

Segundo Mesquita et. Al., (2002), as indústrias que lidam com usinagem e fabricação geram uma grande quantidade de informações relativas aos processos de usinagem. Entretanto, não se tornou um hábito para as indústrias usar estas informações, que representariam uma vantagem estratégica, apesar de já existirem sistemas dedicados a armazenar estes dados. Os Bancos de Dados de usinagem são responsáveis por armazenar tais informações e têm papel central nos sistemas que auxiliam a fabricação, tanto para os sistemas dedicados ao gerenciamento das ferramentas quanto àqueles sistemas que elaboram os programas CNC.

Sistemas de banco de dados de usinagem são essenciais para a seleção de condições de corte otimizadas durante o planejamento do processo (Santos et. Al., 2002). Isto implica em menores custos, pois o uso correto das ferramentas nos processos de fabricação com seus parâmetros de corte otimizados, influenciam diretamente nos tempos de produção. Além disto, estes sistemas de bancos de dados viabilizam o fim do trabalho manual de consulta a catálogos de fabricantes e aumentam a eficiência dos processos.

É fato que os Bancos de Dados são eficazes para a otimização dos processos de usinagem, porém, para que isto possa ocorrer seus dados devem corresponder à realidade dos processos. É

necessário que os Bancos de Dados de Usinagem estejam atualizados, e seus dados correlacionados, de fato, aos processos que ocorrem no chão de fábrica. A exemplo; Coppini e Baptista (2002), apresentam um método para o aproveitamento das informações geradas no ambiente fabril. Este método obtém a partir dos processos executados no chão de fábrica os coeficientes X e K de Taylor essenciais para otimização das condições de usinagem.

3.6.1.2 Sistemas Cliente Servidor

A arquitetura cliente servidor pressupõe a existência da interligação de computadores através de uma rede. Nesta rede teremos dois grandes grupos de computadores: aqueles que fornecem serviços, "os servidores", e os que se utilizam destes serviços "os clientes". Há dois tipos de servidores, os de arquivos e de aplicativos. O servidor de arquivos distribui os programas e arquivos de dados, quando solicitados pelos clientes, porém o processamento é realizado no computador cliente. Um servidor de aplicações, tal como um banco de dados, mantém os programas e arquivos de dados de uma dada aplicação. Quando o cliente faz uma solicitação o servidor de aplicações fará o processamento e enviará de volta somente os resultados necessários, não o arquivo inteiro. O processamento é feito principalmente pelo servidor, embora o cliente também possa processar os resultados, uma vez recebidos do servidor.

As redes de computadores permitem que algumas tarefas sejam executadas no servidor do sistema e outras sejam executadas no cliente. Esta visão de trabalho tem levado ao desenvolvimento de sistemas de banco de dados Cliente-servidor. Neste modelo há o armazenamento de um grande volume de dados em um computador central, e minimiza-se a mudança dos dados para computadores cliente. Desta forma o computador central manipula os dados e envia de volta para o cliente apenas um volume limitado de informações. Isto leva a um maior desempenho, menor quantidade de dados em trânsito, um melhor gerenciamento do acesso concorrente aos dados, melhor proteção e segurança.

Assim, em bancos de dados que utilizam arquitetura cliente-servidor teremos superficialmente duas categorias de funcionalidades, o servidor atuando como "back end" que gerencia as estruturas de acesso, desenvolvimento e otimização de consultas, controle da concorrência e recuperação das informações, e o cliente que consiste de formulários, geradores de relatórios, interfaces gráficas que interagem com o usuário.

Os servidores de transações ou consultas proporcionam uma interface por meio do qual os clientes solicitam ações, eles executam e mandam de volta aos clientes os resultados. Os servidores de dados permitem que os servidores interajam com os clientes, de tal forma que podem criar, atualizar, ler e remover arquivos. Sistemas servidores de dados são usados em redes locais, nas quais há conexões de alta velocidade entre clientes e servidores, os equipamentos clientes são comparáveis em capacidade de processamento com os equipamentos servidores e as tarefas executadas são do tipo processamento intensivo.

O modelo cliente servidor pode, quando bem administrado, trazer economias de custo de longo prazo, por evitarem o uso de computadores de grande porte, além de possibilitarem maior controle sobre os trabalhos, e flexibilizam a capacidade de melhor resposta às mudanças dos negócios. Os custos iniciais dos sistemas cliente/servidor podem ser altos, e os sistemas são mais complexos do que um sistema de computador de grande porte centralizado.

3.7 Abordagens para o Desenvolvimento de sistemas

Desenvolvimento de sistemas é a atividade de criar novos sistemas empresariais, ou modificar os existentes. As etapas de desenvolvimento podem variar de um projeto para outro, porém dentro do enfoque tradicional de desenvolvimento de sistemas, há as seguintes fases técnicas: concepção e viabilidade; análise; projeto; implementação; manutenção e revisão. O sistema passa por este ciclo de vida. O ciclo de vida termina quando é iniciado um novo projeto para substituí-lo devido à obsolescência, mudanças tecnológicas que o inviabilizem, ou a necessidade de uma revisão tão profunda que o projeto de um novo sistema se torna mais recomendável.

Normalmente a maior parte dos esforços de trabalho de desenvolvimento de um sistema segue a seqüência de etapas apresentadas. Porém, na realidade, as atividades de uma fase posterior podem revelar necessidades de modificar decisões e resultados das etapas anteriores. Portanto, o ciclo de desenvolvimento de um sistema não é linear. Existe um inter relacionamento de uma fase com outra, assim pode-se passar de uma fase para a seguinte e depois voltar para fases anteriores para fazer correções e aprimoramentos.

A primeira etapa de concepção e viabilidade vem da identificação de possíveis problemas e oportunidades e seus escopos são considerados à luz das metas das empresas. A análise de sistemas busca um entendimento geral da solução, tentando responder a questão: O que o sistema de informação deve fazer para resolver o problema? Esta fase busca compreender os processos, levantar informações através de documentação e entrevistas com pessoas. O principal resultado da análise de sistemas é uma lista de requisitos e das prioridades do sistema.

O projeto do sistema busca responder a seguinte questão: Como o sistema de informação fará o que é necessário? A implementação de sistemas abrange criar ou adquirir programas e equipamentos, desenvolver sistemas através da sua codificação ou parametrização, contratação e treinamento de pessoal, preparação e especificação do local de trabalho, instalação dos sistemas, teste, partida e operação. A implementação resulta em um sistema de informações implantado e trabalhando de forma efetiva. O propósito da manutenção e revisão dos sistemas é fazer alterações no sistema de acordo com a dinâmica de trabalho da área onde ele esta inserida. Esta atividade é necessária para se manter o sistema operando de forma eficiente, eliminar erros, aumentar a sua eficiência.

O desenvolvimento de sistemas normalmente envolve o esforço de um conjunto de pessoas. Esta equipe é responsável pela identificação dos objetivos do sistema e de produzi-lo para atingir tais objetivos. Os usuários são aqueles que irão interagir regularmente com o sistema. Os especialistas de desenvolvimento de sistemas costumam ter formação de Sistemas de Informação e dependendo do tipo de projeto podem incluir analistas de sistemas e programadores de softwares. Um analista de sistemas é um profissional especializado em analisar e projetar sistemas comerciais. Em pequenas empresas, muitas vezes uma única pessoa desempenha diferentes papéis dentro da equipe de desenvolvimento.

A atividade de se desenvolver Sistemas de Informação não se restringe a automatizar processos ou sistemas existentes. É também necessário repensar o domínio do problema onde o sistema estará inserido. Assim todos os sistemas e processos empresariais com o qual o novo sistema trabalhará devem ser avaliados. A reestruturação correspondente é que possibilitará grandes benefícios para a empresa.

Qualquer que seja a motivação, dar partida em um projeto de desenvolvimento de sistemas significa mudar a maneira como as coisas são feitas no momento. A extensão e o grau do apoio gerencial em geral, principalmente o apoio gerencial superior, influenciam substancialmente o provável sucesso ou fracasso do esforço de desenvolvimento de um sistema.

Os fatores para se iniciar um novo sistema de informação são os seguintes:

- Problemas com o sistema existente.
- Aproveitar novas oportunidades.
- Aumento da concorrência.
- O desejo de tornar mais eficaz o uso das informações.
- Crescimento organizacional, fusão ou aquisição corporativa.
- Mudança no mercado ou ambiente externo.

Os sistemas de informação devem estar em sintonia com o planejamento estratégico da empresa e com o plano de sistemas de informação. O plano de sistemas de informação refere-se à atividade de traduzir as metas estratégicas e organizacionais da empresa em iniciativas da área de tecnologia de Informação.

Grandes sistemas técnicos como SPT e SIG normalmente tem requisitos bem definidos e necessitam de uma análise rigorosa e formal, especificações predefinidas e controles estritos sobre o processo de construção de sistemas, por isto utilizam o ciclo tradicional de desenvolvimento de sistemas. Porém esta abordagem é cara, demorada e pouco flexível.

Outras abordagens para a construção de sistemas podem ser utilizadas dependendo da situação, podemos citar como mais importantes: prototipagem, pacotes de softwares aplicativos, desenvolvimento pelo usuário final, e terceirização.

A prototipagem, ou abordagem evolucionista,é mais indicada quando existir alguma incerteza sobre exigências e, ou, soluções para o problema envolvido. Ela consiste na construção de um sistema experimental de forma rápida simplificada e barata para os usuários finais

avaliarem. Pela interação com o protótipo, os usuários podem ter uma melhor idéia dos requisitos necessários para satisfazer as suas necessidades em termos de sistemas de informação e qual seriam as melhores soluções para isto. O processo de desenvolvimento por prototipagem é essencialmente iterativo e tem como grande vantagem a possibilidade de promover ativamente mudanças no projeto do sistema. Este modelo de desenvolvimento pode também ser usado em partes dos sistemas que precisam evoluir por meio de ciclos de ajustes tal como a interface com o usuário final. A prototipagem estimula o intenso envolvimento do usuário final em todo o ciclo de desenvolvimento.

Outra estratégia é a compra de um pacote de software aplicativo. Os pacotes existem porque têm muitas aplicações que são comuns a várias organizações empresariais, tais como: folha de pagamento, contas a receber, controle de estoque. Um pacote de software tem a maior parte dos projetos e programas já escritos e testados, de modo que custos e prazos para o desenvolvimento de um novo sistema são reduzidos consideravelmente. Além disto os fornecedores do pacote podem também oferecer contratos de manutenção contínua e assistência técnica. Normalmente os pacotes necessitam de uma adaptação à conjuntura de cada empresa, pois as desvantagens dos pacotes podem ser grandes em um sistema complexo e específico. Isto ocorre em aplicações empresariais com características únicas, de tal forma que uma aplicação nova praticamente tem que ser desenvolvida dentro da integridade do pacote. Muitas vezes é necessário o uso de softwares especiais para integrar os dados de dois sistemas necessários, porém diferentes e independentes. Este trabalho de adaptação do pacote de software a realidade da empresa pode demandar grande esforço de programação e ficar tão caro e consumir tanto tempo que eliminam muitas das vantagens dos pacotes de softwares. Estes softwares devem ser bem avaliados na fase de análise, os critérios mais importantes devem ser: Funções oferecidas, flexibilidade, uso amigável, recursos necessários de software e hardware, exigências de Banco de dados, esforço necessário para instalação, contrato de manutenção e assistência técnica, documentação, custo e fornecedor. Quando uma solução de pacote de software é selecionada, a organização não possui mais controle total sobre o processo de projeto do sistema. Se o pacote de software não puder ser adaptado para a organização, a organização terá que se adaptar ao pacote e mudar os seus procedimentos.

Com linguagens de quarta geração, linguagens gráficas e ferramentas de computador os usuários finais podem acessar dados, criar relatórios e desenvolver sistemas de informações inteiros sozinhos. Este método é o desenvolvimento pelo usuário final. Muitos desses sistemas desenvolvidos pelos usuários finais podem ser desenvolvidos muito rapidamente. Com este método há uma melhoria na determinação das exigências, à medida que os usuários especificam suas próprias necessidades. Observa-se um aumento do envolvimento do usuário com os sistemas e uma maior satisfação e, como desenvolvem e controlam seus próprios sistemas os usuários usam mais os sistemas. Ao mesmo tempo, o desenvolvimento de sistemas pelos próprios usuários traz riscos organizacionais, pois ocorre fora dos mecanismos tradicionais de administração e controle de sistemas de informações. Quando os usuários criam suas próprias aplicações e arquivos, torna-se difícil determinar onde os dados estão localizados. Outra implicação é quanto à segurança uma vez que os usuários podem utilizar seu conhecimento sobre os sistemas para acessos indevidos e fraudes. De tal forma que esta abordagem deve ser restrita e controles tornam-se necessários. Para gerenciar os aplicativos desenvolvidos por usuários é necessária a criação de centros de informação. Estes centros assistirão os usuários em seus trabalhos, com a aprovação da alta administração. Eles podem ajudar os usuários a encontrar as ferramentas e aplicações que os tornarão mais produtivos, evitam a criação de aplicativos redundantes e promovem o compartilhamento de dados e apóiam os usuários na solução de problemas.

Uma outra abordagem é a terceirização, quando não se deseja utilizar recursos internos para construir ou operar sistemas de informações. Este método é interessante para situações em que a área de sistemas de informações teve seu custo aumentado muito rapidamente, tem necessidades flutuantes de sistemas, dificuldades em acompanhar o ritmo do desenvolvimento tecnológico, ou para liberar talento escasso e caro para atividades de retorno mais alto. A empresa terceirizada por sua vez pode oferecer melhores preços devido à economia de escala e por ter conhecimento, habilidades, experiências e capacidade que podem ser compartilhados entre vários clientes. Os sistemas mais interessantes para se utilizar terceirização são aqueles que a empresa sente pouca oportunidade para se distinguir competitivamente. Porém, existe uma tecnologia já consolidada. Quando os sistemas não são vitais para a empresa. Quando os recursos do sistema próprio da empresa são limitados, ineficazes e tecnologicamente inferiores. A terceirização deve ser bem avaliada, pois ela pode trazer problemas para a organização. Ela pode perder o controle de sua função de sistemas de informações, colocando-a numa posição desvantajosa, tendo que

aceitar quaisquer taxas e ações por parte de sua terceirizada. Segredos comerciais ou informação proprietária podem vazar para os concorrentes. As organizações necessitam gerenciar o terceirizado como gerenciam o seu próprio departamento de sistemas de informação, estabelecendo critérios de avaliação, e garantindo o pleno funcionamento dos sistemas ou serviços terceirizados conforme o planejado. Toda a prestação de serviço deve ser formalizada por contrato. Na definição do contrato deve atentar para os seguintes detalhes: o contrato deve apresentar todos os custos envolvidos, direitos de ambas as partes ao término do contrato, possíveis indenizações, no caso de perdas provocadas por uma das partes, direitos de propriedade sobre os dados, direitos de propriedade intelectual, repasse de informações técnicas e documentação, possibilidades de alterações, padrões de segurança, padrões de qualidade (Dias,2000). Enfim os contratos devem ser cuidadosamente elaborados de forma a garantir o desejado e abranger ajustes se a natureza dos negócios mudar. A terceirização deve estar alinhada com o objetivo estratégico para a terceirização.

3.7.1 Engenharia de Software assistida por computador

Os sistemas de apoio ao desenvolvimento de software, ou também denominados de ferramentas CASE (Computer Aided Software Engineering), são ferramentas para o desenvolvimento de softwares e busca a automação de metodologias passo a passo para desenvolvimento de software e de sistemas, visando a reduzir a quantidade de trabalho repetitivo que o desenvolvedor precisa fazer, liberando-o para tarefas mais criativas. As ferramentas CASE automatizam muitas das tarefas requeridas no esforço de desenvolvimento de sistemas. Facilitam a documentação dos sistemas e a coordenação dos esforços de desenvolvimento de equipes. Fornecem instrumentos gráficos automáticos para a produção de gráficos e diagramas, geradores de telas e relatórios, dicionários de dados, ferramentas de análise e checagem, geradores de código e geradores de documentos. Em geral as ferramentas CASE aumentam a produtividade, elas utilizam metodologias conhecidas, trazem uma maior disciplina ao projeto. Além disto melhoram a comunicação entre usuários e desenvolvedores, organizam componentes, geram código, fazem engenharia reversa.

4 O Paradigma da Orientação a Objetos

Neste capítulo serão apresentados os principais conceitos do paradigma orientado a objetos que formam a base para o desenvolvimento do sistema aqui em questão. Serão abordadas a programação, projeto e análise, e metodologias orientadas a objetos. Dentre as metodologias orientadas a objetos será dada ênfase ao Processo de Desenvolvimento de Software Unificado, ou simplesmente Processo Unificado, proposto por Booch, Jacobson e Rumbaugh e o processo ICONIX voltado para desenvolvimento de sistemas de tamanho pequeno e médio. Também será apresentada de forma sucinta a Linguagem Unificada de Modelagem (UML).

A análise de negócio e o gerenciamento de pessoas, tempo e recursos e o controle de qualidade para a atividade de produção de sistemas orientados a objeto são áreas de grande importância e interesse dentro da engenharia de software, e, embora estejam relacionadas a este trabalho vão além de seu escopo.

4.1 Introdução

Graham (1994), enfatiza que o termo "Métodos Orientados a Objetos" refere-se a muitas coisas, em particular pode se referir à programação orientada a objetos, projeto orientado a objetos, análise orientada a objetos, banco de dados orientados a objetos, interface gráfica orientada a objetos. De fato, esta frase, se refere de forma geral a toda uma filosofia de desenvolvimento de sistemas.

Neste trabalho, método orientado a objetos é uma forma sistemática e particular de procedimento para criar ou abordar “software”, fundamentado no conceito de objeto. Uma metodologia orientada a objetos é um sistema de métodos orientados a objetos usados em engenharia de software.

Historicamente o interesse pelo desenvolvimento de softwares utilizando métodos orientados a objetos começou com a programação orientada a objetos e mais recentemente se voltou para as outras áreas de desenvolvimento de sistemas. A programação orientada a objetos surge dentro do contexto da “crise do software”, termo cunhado na Conferência de Engenharia de software da NATO em 1968 na Europa. Naquela época, concluiu-se que os métodos de produção de software não eram adequados para as necessidades crescentes de qualidade, menores custos, maior eficiência e reutilização, previsibilidade de prazos e custos (Buzato e Rubira 1998). A orientação a objetos, embora ofereça novas ferramentas e uma abordagem diferente para o desenvolvimento de software, é mais uma abordagem evolucionária do que revolucionária.

Através da técnica da orientação a objetos é possível que softwares sejam construídos a partir de entidades abstratas “o objeto” que têm características e comportamento específico. Estes objetos por sua vez, podem ser construídos a partir de outros objetos que por sua vez podem ser construídos a partir de outro conjunto de objetos e assim sucessivamente. Desta forma o uso do modelo de objetos permite que um software seja mais facilmente escrito a partir de componentes já existentes, e que novos componentes possam ser desenvolvidos a partir dos já existentes, sem afetar os componentes originais o que chamamos de reusabilidade e extensibilidade respectivamente.

Os benefícios advindos do modelo de objetos são vários, como por exemplo, abstração de dados/encapsulamento, modularidade, hierarquia, reutilização de código, facilidade de extensão e manutenção (Buzato, 1998). Graham (1994), cita outros benefícios tais como: O aumento da produtividade e qualidade do desenvolvimento de software pela reusabilidade de classes bem projetadas e testadas. Maior flexibilidade e menor custo de manutenção dos sistemas devido principalmente a extensibilidade e herança. Facilidade na descrição e construção de interfaces gráficas e sistemas distribuídos devido à comunicação entre objetos com o uso de mensagens. Maior segurança dos sistemas pelo ocultamento de dados.

Além das vantagens citadas acima, o paradigma da orientação a objetos permite aos desenvolvedores de software pensar e tratar os problemas relacionados aos sistemas de forma semelhante àquela com que raciocinamos o mundo a nossa volta usando como elemento a idéia de objeto. Isto facilita a atividade de desenvolver sistemas. Relacionado a este fato, porém mais voltado à análise, Rumbaugh, define orientação a objetos como uma nova maneira de pensar os problemas utilizando modelos organizados a partir de conceitos do mundo real (Furlan,1998).

Outra grande vantagem da abordagem orientada a objetos, é que ela permite que um conjunto de conceitos e notação seja usado através de todo o ciclo de vida de um software. Assim em todas as atividades: concepção, análise, projeto, implementação, gerenciamento, etc, utiliza-se os mesmos conceitos e nomenclatura, facilitando a comunicação.

Graham (1994), também cita potenciais problemas quanto à distribuição, disseminação, desenvolvimento e gerenciamento de bibliotecas de objetos e seus custos. Há também problemas relacionados ao aprendizado dos métodos orientados a objetos, resistência às mudanças na cultura do desenvolvimento de sistemas e custos de treinamento e reeducação. Buzato (1998), afirma que o fato das linguagens, metodologias e ferramentas orientadas a objeto ainda estarem em desenvolvimento é limitação ao modelo de objetos. Outra limitação da orientação a objetos é o trabalho em grupo pouco desenvolvido, e a necessidade de maneiras novas para gerência, teste e qualidade de software.

4.2 Conceitos Básicos

Com o paradigma da orientação a objetos é possível representar em software objetos que encontramos no mundo real ou em um mundo imaginário. Na abordagem orientada a objetos, através do conceito de objetos há a unificação de dois elementos que na programação tradicional eram tratados de forma separada: os dados e as operações, Figura 4-1.

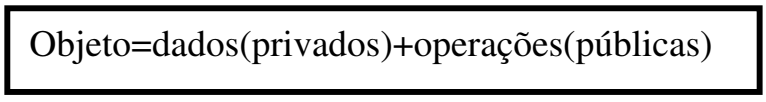

Figura 4-1 - Conceito de objeto segundo a programação orientada a objetos

A unificação destes dois elementos de programação resulta no que chamamos de encapsulamento. O encapsulamento permite o ocultamento de informações, os dados internos ao objeto só podem ser alterados por meio de operações apropriadas, isto permite que a alteração no objeto não cause alteração em outras partes do sistema. Portanto, o termo objeto significa a combinação de dados e programas unidos para representar uma entidade do mundo real ou de um mundo imaginário. Independentemente do tamanho e da complexidade dos objetos, para cada um deles são associados um estado e um comportamento. O estado é representado pelos valores dos atributos dos objetos. O comportamento é definido pelos programas que atuam sobre o estado do objeto. Estes programas são denominados de métodos. É possível que o estado de um objeto seja modificado por outros objetos do mesmo domínio através do envio de mensagens. Os objetos encapsulam os dados e métodos, ou seja, os usuários não podem ver e manipular internamente o objeto, mas podem utilizar o objeto chamando os métodos, através de mensagens, desenvolvidos previamente para este fim.

A partir da observação de um conjunto de objetos pode-se agrupá-los por suas características comuns, o que se denomina classificação. Por exemplo, duas pessoas João e Pedro podem ser classificadas como pessoas. Por sua vez João e Pedro podem ser compreendidos como instâncias da classe pessoa, a instanciação é o processo inverso à classificação. A partir da abstração classificação/instanciação surge o conceito de classe, uma classe é a descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e significado (Booch et. al. ,1999). O termo objeto, no discurso usual, algumas vezes se refere à classe e outras a instâncias de uma classe.

Pode-se, a partir de duas ou mais classes, abstrair uma mais geral que as outras. Denomina-se este processo de generalização. A generalização permite que todas as instâncias de uma categoria específica sejam consideradas instâncias de uma categoria mais abrangente, porém o inverso não é sempre verdadeiro.Por exemplo, um automóvel é um veículo, mas nem todo veículo é um automóvel. O processo oposto à generalização é a especialização. Em sistemas orientados a objetos é possível que classes herdem características de uma classe mais geral chamada de superclasse ou classe pai. Herança é o compartilhamento de atributos e operações entre classes em um relacionamento hierárquico. Através da herança as classes filhas podem utilizar o código já desenvolvido na classe pai. Se for conveniente, porém, as características

herdadas podem ser sobrecarregadas, ou seja, redefinidas, ou características específicas podem se acrescentadas para lidar com particularidades e exceções.

A agregação é uma abstração que permite a identificação das classes que compõem outras classes. É a idéia do todo em relação as suas partes constituintes. O seu processo inverso é a decomposição.

Quando um sistema tem muitos detalhes relevantes para serem representadas através apenas de uma abstração, esta pode ser decomposta em uma hierarquia de abstrações. Uma hierarquia permite que detalhes relevantes sejam introduzidos de forma controlada. Existem dois tipos de hierarquias de abstrações fundamentais: hierarquias de agregação e de generalizações. Nas hierarquias de generalização/especialização, a noção de herança esta implícita.

4.3 Fases de desenvolvimento de um software

Segundo Buzato e Rubira (1998), em geral, de forma independente de metodologia, o processo de desenvolvimento de software consiste das seguintes etapas:

1-especificação do problema

2-entendimento dos requisitos

3-planejamento da solução, e

4 implantação de um programa numa dada linguagem de programação.

A especificação do problema esta ligada à compreensão do contexto onde o software irá atuar. O software que será subsistema de um sistema maior. Este sistema maior, no qual o software vai funcionar, define o que se chama o domínio do problema. Este universo inclui todas as fontes de informação e todas as pessoas relacionadas. A falta de conhecimento do contexto pode ser considerada o mais significativo problema no desenvolvimento industrial de software.

A atividade de Análise de Sistemas, em seu significado mais amplo, consiste em compreender o domínio do problema (o ambiente, os usuários e suas características) e utilizar esta compreensão para a análise de requisitos e a concepção em alto nível, ou seja, para fazer

respectivamente a identificação dos requisitos e a especificação de um sistema que possa atendê-los (Nizoli, 2001).

Requisito é uma função ou característica de um sistema que é necessária ao usuário, do ponto de vista de quem desenvolve sistemas pode ser simplesmente algo que precisa ser modelado.O processo de extração de requisitos é impreciso e difícil, o que torna a sua total automatização, se não impossível, pouco provável (Carvalho e Chiossi, 2001). Qualquer que seja a metodologia adotada para o desenvolvimento, é a partir dos requisitos que se basearão todos os outros passos para se alcançar o objetivo final, o software.

A análise começa normalmente com uma solicitação e com uma série de discussões e entrevistas que envolvem as pessoas interessadas de forma a precisar e concordar com o conjunto de requisitos propostos através de um documento de requisitos. Um documento de requisitos tem duas finalidades: formalizar as necessidades do cliente, e estabelecer uma lista de prioridades, em busca de funcionalidade.

Uma vez que as especificações de requisitos tenham sido satisfeitas inicia-se o trabalho de modelagem, ou seja, a criação de um conjunto de modelos que representam diversas visões do sistema que esta sendo construído.

Depois de ter analisado o problema, precisa-se decidir como abordar o projeto. O projeto do sistema é a estratégia de alto nível para resolver o problema e elaborar a solução. O projeto do sistema inclui decisões sobre a organização do sistema em subsistemas, a alocação de subsistemas aos componentes de hardware e de software e importantes decisões conceituais e políticas que forma infraestrutura do projeto detalhado (Rumbaugh,1994).

A codificação é uma extensão do processo de projetar. Ela deve ser direta, quase mecânica, porque todas as decisões difíceis devem ter sido tomadas durante o projeto. O código deve ser uma simples tradução das decisões do projeto nas particularidades de uma linguagem específica.

4.4 Desenvolvimento de software Orientado a Objetos

Um dos mais urgentes interesses da indústria de computação atual é criar softwares e sistemas corporativos mais rápidos e completos a um custo mais baixo. Isto leva a softwares mais complexos, embora mais complexos, necessitam ser mais confiáveis.

Nas últimas décadas houve o aprimoramento constante das técnicas e metodologias para a engenharia de software visando acompanhar a evolução natural do conhecimento.

A necessidade do desenvolvimento de melhores softwares aplica-se tanto à indústria de software como dentro de empresas de todo o tipo que desenvolvem suas próprias aplicações. A criação e modificação destas aplicações devem ser rápidas para acompanhar as mudanças dos empreendimentos e do ambiente onde estão inseridas.

Métodos orientados a objetos procuraram tanto redefinir quanto estender os métodos existentes. Eles abandonam o tradicional modelo clássico em cascata, utilizado na análise estruturada, que define fases fixas como especificação, análise, projeto passando para o modelo em espiral o qual o desenvolvimento de software se move em uma espiral evolucionária, onde as fases de análise, projeto, evolução e modificação são executadas de forma iterativa e incremental, buscando englobar as melhores características do ciclo de vida clássico com o do paradigma evolutivo.

Nos métodos orientados a objetos o projetista não começa pela identificação das diferentes funcionalidades dos sistemas, mas pela identificação dos objetos que compõem o sistema. O modelo de objetos leva a uma melhor integração das diferentes fases porque todas as fases utilizam uma mesma ferramenta conceitual, os objetos.

A análise e o projeto orientados a objetos lidam tipicamente com o problema de desenvolvimento de sistema de software grandes e complexos, sujeitos a mudanças. Na construção deste tipo de sistemas, a programação dos componentes individuais freqüentemente não é o problema mais difícil. O que é realmente desafiador é a decomposição e integração de tais componentes durante as fases de análise e projeto, objetivando a obtenção de um sistema bem estruturado que seja de fácil manutenção, entendimento e modificação/adaptação (Buzato e Rubira, 1998).

Devido a isto, a tecnologia de desenvolvimento de software que utiliza os conceitos da Orientação a Objetos tem impulsionado muito o desenvolvimento de software e atualmente o paradigma da orientação a objetos é o mais disseminado. Martin e Odell (1992), sugerem que a Análise e o projeto orientados a objetos combinados a outras tecnologias de software tais como: ferramentas CASE, geradores de código, máquinas de inferência, inteligência artificial, Cliente servidor, programação paralela, Bancos de dados Relacionais, Bancos de dados Orientados a Objetos, Sistemas gerenciadores de Banco de Dados, integração à Internet, Bibliotecas de classes, trazem o grande avanço à indústria do software.

4.5 Metodologias Orientadas a Objetos

Uma metodologia de desenvolvimento de software consiste em uma linguagem de modelagem, com notação, principalmente gráfica, para especificação, documentação, comunicação e visualização de artefatos, e um processo que é um conjunto de diretrizes que organizam o desenvolvimento do software e um conjunto de instrumentos que permitem que artefatos sejam construídos (Mendes, 2003). Um artefato é um conjunto de informações utilizado ou produzido por um processo de desenvolvimento de software.

Os métodos orientados a objetos começaram a aparecer entre meados de 70 e início dos anos 80 do século XX, porém este número saltou para mais de 50 entre 1989 e 1994 (Furlan,1998). Dentre as metodologias pioneiras que surgiram neste período e que se pode destacar devido a suas contribuições às técnicas orientadas a objeto atuais são:

- Object-Oriented Design (OOD), de Booch;
- Object Modeling Techique (OMT), de Rumbaugh
- Object-Oriented Analysis (OOA) de Coad e Yourdon
- Objectory, de Jacobson

Estes quatro métodos citados são apresentados resumidamente a seguir:

A metodologia OMT (Object Modeling Technique), desenvolvida por Rumbaugh é amplamente considerada como uma das mais completas formas de análise de sistemas baseadas

em objetos (Graham,1994). Este método originou-se do trabalho de James Rumbaught e seus colegas na "General Eletric". A metodologia consiste na construção de um modelo de um domínio do problema e na posterior adição dos detalhes de implementação durante o projeto do sistema. A metodologia consiste nas etapas de Análise, Projeto do sistema, Projeto dos Objetos e implementação (Mastelari e Coppini, 2002). A técnica de modelagem de objetos faz uso de três modelos para descrever um sistema: o modelo de objetos, que descreve os objetos do sistema e seus relacionamentos; o modelo dinâmico que descreve as interações entre os objetos do sistema; e o modelo funcional que descreve as transformações dos dados do sistema. A descrição completa do sistema exige todos os três modelos. Os três modelos do sistema são desenvolvidos e posteriormente refinados durante as demais fases do método. A primeira etapa da análise dos requisitos é a construção de um modelo de objetos. A OMT apresenta heurísticas para a determinação das classes a partir da descrição do problema. Muito da notação e simbologia padronizada na UML provem da OMT.

Análise Orientada a Objetos – AOO, apresentado por Coad e Yourdon (1992) é apresentado e revisado em cinco níveis: nível de assunto, nível de classe & objeto, nível de estrutura, nível de atributo, nível de serviço. Estes cinco níveis se assemelham a transparências, que quando colocadas umas sobre as outras, mostram cada vez mais detalhes. Elas levam a criar uma estrutura estável para análise e especificação, onde os níveis superiores são mais estáveis com o tempo. Os assuntos, classes e objetos, variam pouco com o tempo, já os atributos e serviços são mais voláteis.

Na modelagem da AOO é feita a distinção entre objetos e classes. O número de classes em um modelo AOO depende da extensão e da profundidade pretendida do sistema no domínio do problema, e das responsabilidades do sistema neste domínio. Os objetos e classes em um modelo orientado a objetos se relacionam uns com os outros formando através de conectores e associações. Temos as estrutura generalização especialização, Gen_espec, a estrutura de agregação todo parte. Assim o método OOA consiste nos seguintes passos: Identificar classes e objetos, identificar estruturas, definir sujeitos, definir atributos, definir serviços. O diagrama principal é o de classes objetos e utiliza conectores e associações para representar os

relacionamentos de generalização/especialização, todo/parte, de objeto a objeto, ou uma conexão de mensagem.

Object-Oriented Design (OOD), de Booch, propôs um método que consistia no emprego de técnicas de projeto orientado a objeto depois estendido para contemplar também a análise orientada a objetos. O autor descreve um objeto como sendo um modelo do mundo real que consiste de dados e habilidades para o tratamento destes dados. Neste método uma vez tendo sido localizados os objetos, esses passam a servir de base para os módulos do sistema ou são considerados módulos relacionados. Seus conceitos também foram amplamente utilizados na elaboração da UML.

Objectory de Jacobson(OOSE): O método Objectory, proposto por Ivar Jacobson, foi concebido com o intuito de melhor representar a realidade do domínio do problema, levando em conta o uso futuro que o sistema terá. Em sua metodologia introduz os casos de uso ("use cases") foco de sua metodologia. Os casos de uso tornaram-se populares e tiveram sua utilização disseminada com o surgimento da UML. A análise no método de Jacobson é baseada em modelos de requerimentos e análise que consistem de um conjunto de casos de uso, de um modelo de domínio de problema e de uma descrição da interface do sistema.O método OOSE utiliza uma notação simplista, porém tem sido adaptado para engenharia de negócio, onde idéias são usadas para modelar e melhorar processos.

Nizoli (2001), apresenta que o maior problema tanto do método OMT como OOA reside no fato de que as técnicas para a identificação dos objetos, das classes, dos atributos e das operações são muito heurísticas. Existe uma grande dificuldade em definir exatamente quais são os objetos essenciais para o sistema particular sem saber como o sistema será utilizado. É possível observar que cada método sugere diferentes heurísticas para a identificação dos componentes da estrutura.

Entre as metodologias recentes podemos destacar o Processo Unificado proposto por Booch, Jacobson e Rumbaugh, Booch, devido ao fato de seus autores também terem criado a UML e ferramentas CASE para trabalhar com esta metodologia. A metodologia ICONIX proposta por Rosenberg e Scott, (2001) é baseada no Processo Unificado. O desenvolvimento de

software tem também forte influência dos métodos ágeis tal como a metodologia XP (Extreme Programming). A seguir é feita uma breve descrição destas metodologias.

4.5.1 Modelagem Ágil (Agile Modeling)

Esta é uma metodologia baseada na experiência prática, para a modelagem e documentação de sistemas de software (Ambler, 2003). A Modelagem Ágil (AM) é uma coleção dos valores, princípios, e práticas para modelar software que pode ser aplicado em um projeto do desenvolvimento do software de uma maneira eficaz. O AM não é um processo prescritivo, em outras palavras ele não define procedimentos detalhados para criar modelos, ao invés disto fornece um conjunto de valores, princípios e práticas com o objetivo de tornar este processo mais eficaz. Os seus principais valores são: comunicação entre todos que suportam o projeto, simplicidade para desenvolver a solução mais simples possível, realimentação através do retorno o mais freqüente e cedo possível a respeito dos esforços realizados, coragem para tomar decisões, humildade para admitir que se pode não saber tudo e que outros tem valor a acrescentar ao esforço do projeto.

4.5.2 Programação Extrema (Extreme Programming)

Esta metodologia aplica os conceitos da programação ágil. Ela é indicada para domínios de problemas cujos requisitos mudam rapidamente, em situações em que o cliente não tem uma noção clara do que o sistema deve fazer, e nos casos que envolvam altos riscos do não cumprimento do cronograma por exigência e rigor por parte dos clientes na data de entrega do produto, ou por ser um desafio novo e totalmente desconhecido.

A metodologia XP utiliza um pequeno conjunto de regras e práticas descritas por Wells (2003). A metodologia é resumida pela Figura 4-2. Ela pode ser resumida pela seguinte dinâmica: Os usuários descrevem de forma sucinta (os requisitos são obtidos de forma detalhada com o usuário quanto eles forem ser implementados) o que o sistema deve fazer para eles através das ‘estórias dos usuários’. A partir delas é feita uma estimativa pouco precisa do tempo necessário preparando-se para o “plano de versão”.

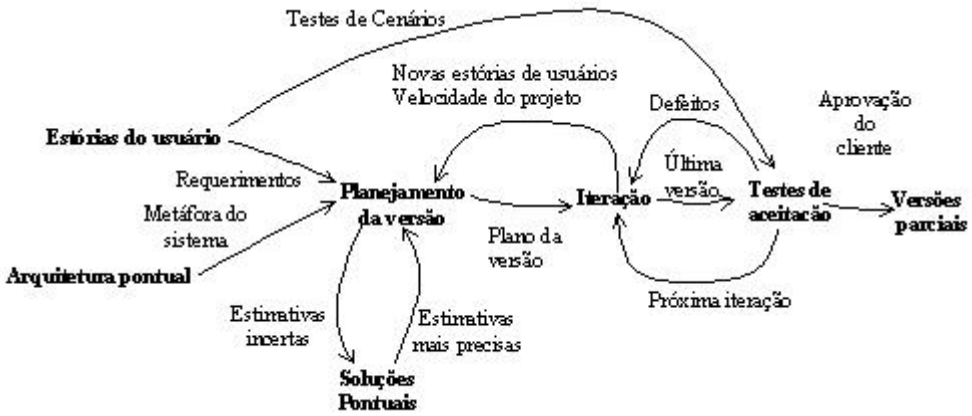

Figura 4-2 - Diagrama Esquemático da Programação Extrema – fonte Wells,2003

Para avaliar a dimensão de problemas técnicos ou do projeto que possam ser considerados críticos ou desafiadores são criadas soluções pontuais. Uma solução pontual é um programa muito simples para explorar soluções potenciais. O objetivo é reduzir os riscos que envolvem um problema técnico de forma a aumentar a confiabilidade da estimativa de uma história do usuário. A partir das estórias dos usuários, e soluções pontuais é feito o planejamento da versão. O planejamento de uma versão é realizado com todos clientes e desenvolvedores. Nesta atividade determinam-se prazos a partir de estimativas de tempo e prioridades entre as histórias dos usuários para a realização de uma nova iteração. O número de iterações em um desenvolvimento de software é bastante grande e elas têm duração constante e breve, na ordem de uma a três semanas. Para o início de cada iteração é feita uma nova reunião de planejamento e assim se faz o acompanhamento do desenvolvimento como um todo.Os prazos para realizar as atividades dentro de uma iteração devem ser cumpridos caso isto não seja possível deve-se realizar uma nova reunião de planejamento.Durante a reunião de planejamento as histórias do usuário selecionadas são traduzidas em testes de aceitação. O cliente especifica cenários para testar quando uma história do usuário foi desenvolvida corretamente.Os clientes são responsáveis em verificar a corretude destes testes, uma história do usuário não é considerada completa até que tenha passado no teste de aceitação. Os testes de aceitação devem ser automatizados e rodados novamente a cada iteração futura. O grupo de desenvolvimento necessita liberar periodicamente, a cada iteração, versões parciais funcionais aos clientes. Isto é crítico para obter valioso retorno dos clientes em tempo para que possa fazer alterações necessárias e importantes no desenvolvimento

do sistema, deste modo problemas podem ser detectados e facilita a comunicação entre desenvolvedores e clientes.

Na metodologia XP há uma grande necessidade de interação e envolvimento dos usuários entre as etapas. Isto a princípio não é ruim, porém, muitas vezes não é possível.

Em XP a codificação é a principal tarefa, não é dada grande ênfase à documentação do projeto, pois se pressupõem que o sistema está em continua mudança. O código passa a ser esta documentação. Esta abordagem dificulta principalmente a comunicação e a visualização do sistema como um todo.

4.5.3 O Processo Unificado

O Processo Unificado é descrito em seus conceitos básicos no trabalho de Booch et al. (1999). O processo unificado faz uso da UML para preparar a planta-baixa do software.

Esta metodologia é baseada em casos de uso, centrada em arquitetura, iterativa e incremental. Os casos de uso do sistema formam um modelo de casos de uso que descreve a funcionalidade completa do sistema segundo a visão dos clientes ou usuários do sistema. O papel da arquitetura de software é similar ao exercido, em sua natureza, pela arquitetura em uma construção. Ambas arquiteturas permitem que se visualize uma imagem completa do que será construído. De forma resumida, a arquitetura de um sistema pode ser descrita como diferentes visões do sistema que será construído. O desenvolvimento do sistema é dividido em partes menores ou mini-projetos, cada um destes mini-projetos é uma iteração que resulta em um incremento.

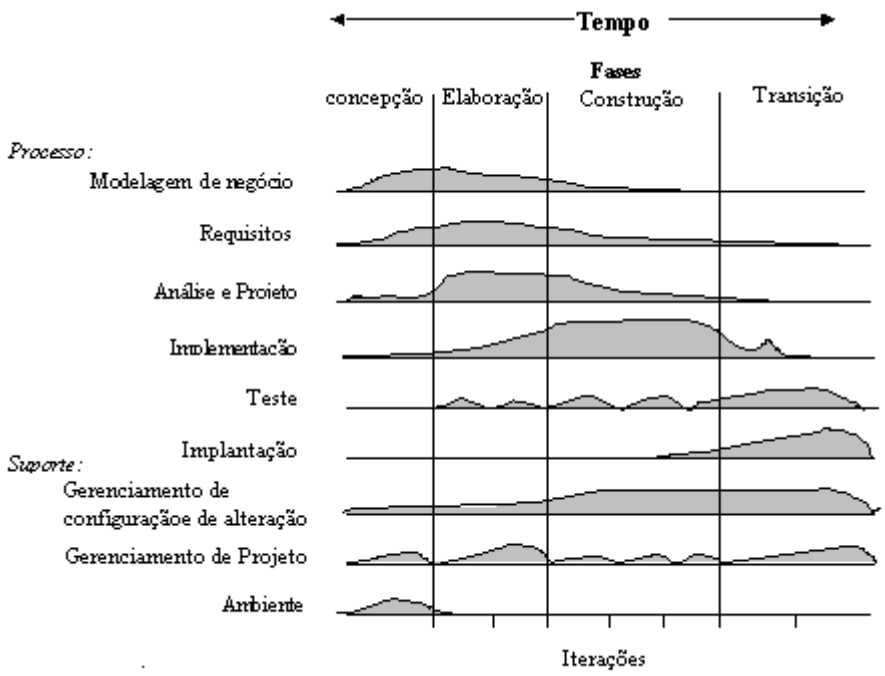

Figura 4-3 - O processo Unificado, fonte Booch, 1999.

A Figura 4-3 representa o método que é descrito em duas dimensões: A primeira dimensão é o tempo, e mostra como o desenvolvimento é ordenado e expresso em termos de ciclos, fases, iterações e datas importantes.A segunda dimensão indica as principais atividades a serem trabalhadas. Elas são desenhadas ortogonalmente ao tempo.

A vida de um sistema é constituída por uma série de ciclos, cada ciclo é concluído com uma versão do produto para os clientes. Cada ciclo tem as seguintes fases: concepção, elaboração, construção e transição.

Na fase de concepção é estabelecido o caso de negócio para o sistema e delimitado o escopo do projeto. Para tal, identifica-se os atores que estarão interagindo com o sistema definindo a natureza dessa interação em uma perspectiva de alto nível, ou seja, descrevem-se os casos de uso com caráter mais significativo. O caso de negócio inclui critérios de sucesso, taxa de risco, estimativa de recursos necessários e um plano mostrando as datas principais de entrega de resultados. Neste estágio a arquitetura é temporária, sendo apenas um esboço contendo os subsistemas mais cruciais, a fase seguinte de elaboração deve ser planejada em detalhes. Ao final

desta fase, são examinados os objetivos do ciclo de vida do projeto para se decidir sobre a continuidade ou não de seu desenvolvimento.

Na fase de elaboração faz-se uma análise mais refinada do sistema juntamente com um plano detalhado do trabalho a ser feito. Pesquisa-se o campo ou atividade do qual o sistema fará parte, ao que denominamos domínio do problema. Deve-se estabelecer uma arquitetura com fundação sólida, ou seja, as diferentes visões do sistema devem ser construídas mantendo coerência entre si. As decisões da arquitetura devem ser feitas com a compreensão do sistema como um todo. Isto implica na descrição da maior parte dos requisitos do sistema. Para verificar a arquitetura implementa-se um sistema para demonstrar as escolhas da arquitetura e executar os casos de uso mais significativos. Ao término desta fase, examinam-se os objetivos do sistema, o seu escopo, a escolha da arquitetura, a resolução dos principais riscos e toma-se a decisão de prosseguir ou não com a construção.

Durante a fase de construção, um produto completo é desenvolvido de maneira iterativa e incremental para que esteja pronto para a fase de transição. Nesta fase, a arquitetura torna-se um sistema maduro sendo totalmente estável. Implica na descrição e formatação dos casos de uso remanescente. Entretanto, os desenvolvedores podem descobrir melhores maneiras para estruturar o sistema, possuindo então a liberdade de sugerir pequenas alterações na arquitetura. São feitos testes de software. Ao final desta fase decide-se se o software, local e usuários estão todos prontos para entrar em operação.

Na fase de transição, o software é disponibilizado à comunidade usuária.Esta fase tipicamente começa com o lançamento de uma versão do sistema para teste de validação e aceitação e envolve a correção de defeitos e deficiências. Outras atividades incluídas na fase de transição são treinamento, suporte ao usuário e manutenção do sistema.

As fases podem ser subdivididas em iterações. Cada iteração resulta em um incremento do sistema podendo-se, assim, visualizar a sua evolução e os seus resultados são vistos logo. Cada iteração inclui as diversas atividades, porém com diferentes ênfases dependendo da fase. Durante a concepção o foco é a captura dos requisitos. Na elaboração o foco recai sobre a análise e projeto, na construção a implementação é a atividade central e na transição a atividade central é a distribuição do software. Caso a iteração atinja seus objetivos, o desenvolvimento passa a

próxima iteração caso contrário, os desenvolvedores retomam as decisões prévias e tentam uma nova abordagem.

O processo unificado consiste de nove atividades principais: a modelagem de negócio, levantamento das necessidades, análise e projeto, implementação, teste, distribuição, gerenciamento da configuração, gerenciamento do projeto, ambiente.

A modelagem de negócio descreve a estrutura e dinâmica da organização.

O levantamento de necessidades descreve o método baseado em use cases para elucidar os requerimentos. Deve levar desenvolvedores e usuários a concordarem sobre uma descrição comum. O diagrama de caso de uso é a principal técnica utilizada na fase de levantamento de necessidades, muito embora um diagrama de classe de alto nível possa ser especificado (Furlan, 1998).

A análise e o projeto descrevem as múltiplas visões da arquitetura. A análise e o projeto mostram como o sistema será materializado na fase de implementação. Busca-se construir um sistema estruturado e robusto que execute, em um ambiente de implementação específico, as funções e tarefas descritas nos casos de uso. Para representar este modelo temos o diagrama de objetos, que são divididos em diagramas de classes e diagramas de instâncias. O diagrama de instâncias é utilizado para particularizar situações formando cenários e exemplos para discussão. Um dado diagrama de classes corresponde a um conjunto infinito de diagramas de instâncias. Os diagramas UML mais empregados durante a fase de análise são: caso de uso, classe, seqüência e estado. Por outro lado, o modelo do projeto serve como abstração do código fonte, atuando como esboço de sua estrutura. Os diagramas da UML mais empregados durante a fase de projeto são: Classe, colaboração, atividade, componente e implantação.

A implementação leva em conta o desenvolvimento o teste das unidades e a integração. A fase de implementação ou construção ocorre no instante em que as classes são programadas tomando-se por base o modelo de projeto. O sistema é percebido através da implementação, produzindo-se os arquivos de código fonte que resultarão em um sistema executável.

No Processo unificado as atividades de análise (levantamento dos requisitos, análise e projeto) são mais elaboradas que na XP. Porém, quanto menos estáveis sejam os requisitos,

menos eficaz se torna o trabalho de análise. Nestes casos os métodos ágeis se mostram mais indicados.

4.5.4 A metodologia ICONIX

A metodologia ICONIX é baseada no Processo Unificado, e como este é iterativo e incremental. Ela é orientada por casos de uso e busca a modelagem por um conjunto mínimo de diagramas UML, porém deixa em aberto a possibilidade de selecionar outros aspectos da UML para suplementar este material básico. É um processo enxuto e robusto voltado para trabalho em grupos com poucos integrantes e desenvolvimentos de tamanho pequeno e médio. Ele esta representado pela Figura 4-4.

Dentro do processo ICONIX um dos primeiros passos é a criação de um modelo de casos de uso. Os casos de uso são usados para capturar os requisitos de alto nível do sistema segundo a visão do usuário. Eles também são utilizados como unidade de trabalho, entrega e estimativa da evolução do desenvolvimento do sistema. O levantamento dos casos de uso é uma técnica onde o usuário descreve suas expectativas de forma geral e normalmente pouco precisa, é, portanto segundo este aspecto um processo informal de modelagem. Este modelo substitui a tradicional especificação funcional, (Jacobson,1992), e responde a questão “O que o sistema deve fazer, para qual usuário?” A captura dos requisitos dos usuários é feita pelo detalhamento dos cursos típicos de eventos em cenários de interação entre os usuários e o sistema.

Como é uma metodologia orientada por casos de uso significa que se deve escrever o manual do usuário primeiro, e só então escrever o código. Esta prática reforça a noção fundamental de que um sistema deve se conformar às necessidades dos usuários, em vez de seus usuários terem que se conformar ao sistema (Rosenberg e Scott, 2001). É importante que o usuário participe ativamente da determinação dos casos de uso e que faça ao final uma revisão e validação destes casos. Erros nesta etapa podem se mostrar críticos e terem um custo muito alto se forem detectados tardiamente. A participação dos usuários traz segurança aos desenvolvedores uma vez que parte da responsabilidade sobre o sucesso ou fracasso depende deste trabalho conjunto. Tendo em mente que está lidando com informações validadas pelo usuário, parte das preocupações dos desenvolvedores pode ser transferida para questões que realmente necessitem de maior esforço.

A modelagem do domínio do problema é a tarefa de descobrir classes que representem as principais coisas e conceitos do sistema e seus inter relacionamentos. O termo domínio do problema refere-se à área que abrange as coisas do mundo real e os conceitos relacionados ao problema para o qual o sistema esta sendo projetado para resolver. Nesta tarefa, para a obtenção das classes e seus relacionamentos, faz-se o uso da análise gramatical da declaração em alto nível do domínio do problema, requerimentos de baixo nível e do conhecimento especialista do problema. A partir desta atividade há a proposição do modelo do domínio que é uma representação inicial do modelo de objetos do sistema. Este modelo vai sendo refinado conforme o projeto evolui, e novos objetos vão sendo incluídos até constituir o modelo de objetos final representado pelo diagrama de objetos. A modelagem do domínio revive a análise gramatical também utilizada por Rumbaugh (1994) em seu método OMT ("Object Modeling Technique").

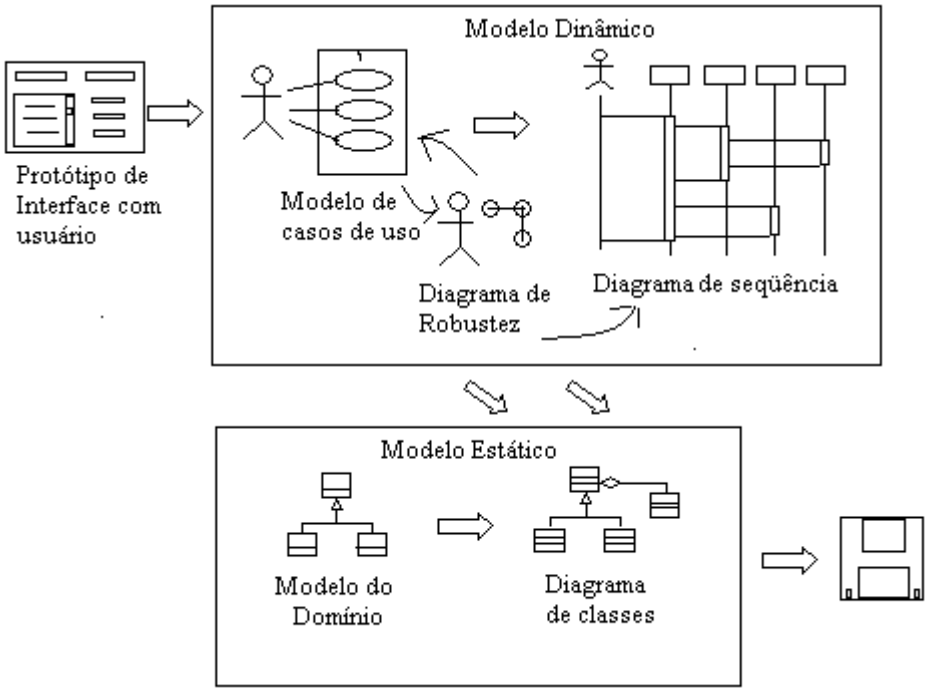

Figura 4-4 – Diagrama esquemático do processo ICONIX,

Fonte Rosenberg e Scott, 2001

Esta metodologia utiliza técnica para, a partir dos casos de usos descobrir os objetos que compõem o sistema classificando-os em objetos limites, entidades, e controladores esta técnica é denominada modelagem de robustez (Scott, 2001). Os objetos limites são aqueles com os quais

os atores interagem, por exemplo, as telas. As entidades são as classes que irão armazenar as informações do sistema e formam a estrutura do sistema. Os controladores fazem a conexão entre os demais e entre si, eles usualmente servem como apresentação de funcionalidades e comportamento do sistema requerido pelos casos de uso. As regras para a implementação deste modelo são: Os atores do sistema somente podem se comunicar com os objetos limites, os objetos limites só podem se comunicar com os controladores e atores, as entidades só podem se comunicar com os controladores, e os controladores podem se comunicar com outros controladores, entidades e objetos limites, mas não com os atores. Esta técnica faz uma ligação entre a análise, "o que", e o projeto "como", refinando os casos de uso e o modelo. Com a análise de robustez busca-se o refinamento e padronização dos casos de uso e do modelo do domínio, e como é um esboço do modelo dinâmico do sistema diminui o tempo da construção dos diagramas de seqüência. Esta técnica se torna de mais valiosa quando os casos de uso são grandes.

Na fase de projeto, onde se determina "como" realmente o software irá atuar faz-se a modelagem através de diagramas de seqüência. Um diagrama de seqüência mostra uma interação, isto é, uma seqüência de mensagens trocadas entre vários objetos num determinado contexto. Este diagrama enfatiza a comunicação e a passagem de controle entre os objetos ao longo do tempo. A partir dos casos de uso busca-se determinar a interação entre os objetos em uma linha do tempo definindo quais objetos são responsáveis por quais comportamentos. Desta forma é possível distribuir as operações entre as classes. Para cada unidade do comportamento dentro do caso de uso, devem-se identificar as mensagens e os métodos necessários. Uma mensagem é uma comunicação entre objetos (emissor e receptor) que veicula informação na expectativa de provocar uma resposta, é representada por uma seta horizontal, do emissor para o receptor, com nome e possíveis argumentos. Os tipos de mensagens são síncronos e assíncronos. As mensagens síncronas são àquelas em que o emissor fica parado esperando a resposta do receptor, corresponde tipicamente à chamada de operação ou procedimento no receptor. Nas mensagens assíncronas o emissor não fica parado a espera de uma resposta, corresponde tipicamente a envio de sinal entre objetos concorrentes.

Esta metodologia utiliza intensivamente os diagramas de seqüência para guiar a implementação. Com estes diagramas determinam-se claramente os objetos a comunicação entre

eles, as principais seqüências de ação, e as principais operações a serem implementadas nas classes.

Neste processo há uma intensa interação com o modelo de objetos, de forma que ele vai sendo atualizado conforme as operações vão sendo alocadas às classes. Ao final é criado o diagrama de classes.

4.6 A Linguagem de Modelagem Unificada - UML

Aqui serão descritos de forma sucinta os principais conceitos da UML, com a apresentação de definições, simbologia, diagramas e notação padronizada através da UML. As definições e vocabulário foram obtidos dos textos: Booch et. Al. (1999), Furlan (1999) e Popkin (1998).

4.6.1 Introdução

A UML, Unified Modeling Language, é uma linguagem de modelagem que prescreve um conjunto padrão de diagramas e notação para a modelagem de sistemas segundo o paradigma da Orientação a objetos, e descreve a semântica subjacente do que estes diagramas e símbolos significam.

A UML é uma linguagem de modelagem. Uma linguagem proporciona um vocabulário e as regras para a combinação das palavras deste vocabulário com o propósito de comunicação. Uma linguagem de modelagem é uma linguagem cujo vocabulário e regras focam a representação conceitual e física de um sistema.

A UML surgiu da união de forças entre Grady Booch e James Rumbaugh através da Rational Corporation para forjar uma unificação completa de seus trabalhos, os métodos de Booch e OMT que estavam crescendo e que eram reconhecidos mundialmente. Em outubro de 1995 lançaram um rascunho do Método Unificado, sendo este o primeiro resultado concreto de seus esforços. A este grupo veio se juntar, Ivar Jacobson fundindo o seu método OOSE gerando a UML. Visto que havia muitas notações e métodos usados para o projeto orientado a objetos, através da UML eles propunham uma notação única para os profissionais de sistemas utilizarem. A idéia central da UML é, a utilização de uma só linguagem para o desenvolvimento de sistemas,

de tal forma que analistas desenvolvendo diferentes sistemas poderiam compreender rapidamente um o projeto dos outros.

A UML, não é uma metodologia. Muitas metodologias consistem, pelo menos em princípio, de uma linguagem de modelagem e um procedimento de uso dessa linguagem, a UML não prescreve explicitamente esse procedimento de utilização. A UML apresenta como criar e interpretar modelos, porém ela não apresenta quais e quando os modelos devem ser criados.

A UML, também não é uma linguagem de programação visual, mas seus modelos podem ser diretamente conectados a uma variedade de linguagens de programação. Coisas que são mais facilmente compreendidas graficamente são feitas usando os diagramas UML, coisas que são mais bem apresentadas textualmente são feitas utilizando uma linguagem de programação associada.

Este mapeamento dos modelos UML para uma linguagem de programação associada, por exemplo, C++, Java, Object Pascal, permite a geração do código a partir dos modelos formando uma estrutura, "um esqueleto", a base para o início organizado da programação. O reverso também é possível. Podem-se reconstruir modelos UML a partir de um sistema implementado. Esta engenharia reversa requer um suporte de ferramentas para realizar tal tarefa e necessita também intervenção humana, para situações onde haja perda de informações.

É consenso o fato de que a UML não é somente uma simples padronização em busca de uma notação unificada, já que contém conceitos novos, não encontrados em outros métodos orientados a objeto. De fato, os autores da linguagem não inventaram a maioria das idéias: seu papel foi identificar, selecionar e integrar as melhores práticas existentes (Nizoli, 2001).

O tempo de desenvolvimento de software pode ser significantemente reduzido com a utilização de modelos orientados a objetos. Feng, 2003, apresenta um modelo de informações para processos de manufatura voltados para a integração do projeto com o planejamento do processo utilizando UML.

4.6.2 Uma visão geral da UML

Para se compreender a UML, é necessário conhecer os seus três principais elementos: Os seus blocos de modelagem básicos, as regras que regem como estes blocos interagem e os mecanismos comuns a toda linguagem.

O vocabulário da UML abrange três tipos de blocos de modelagem: coisas, relacionamentos e diagramas.

Coisas são abstrações elementares do modelo tais como classes, casos de uso, interações. Na UML há quatro tipos de coisas: coisas estruturais, coisas comportamentais, coisas de agrupamento e coisas para anotações. As coisas estruturais são os substantivos do modelo, representando elementos físicos ou conceituais. As coisas comportamentais são a parte dinâmica dos modelos da UML, são os verbos dos modelos, representando o comportamento no tempo ou no espaço. As coisas de agrupamento formam a parte organizacional dos modelos UML. Elas são caixas nas quais os modelos podem ser decompostos. As coisas para anotações formam a parte explicativa dos modelos UML.

Ao se modelar um sistema é necessário além de se identificar as coisas que formam o seu vocabulário também representar como estas coisas se relacionam uma com as outras. Na UML há quatro tipos de relacionamentos: dependências, associações, generalizações e realizações.

$\Rightarrow$ Diagramas

Um diagrama é uma representação gráfica de um conjunto de elementos, freqüentemente traduzidos para grafos direcionados onde os vértices são coisas e os arcos são relacionamentos. Os diagramas são utilizados para se visualizar o sistema de diferentes perspectivas tais como a sua perspectiva estática, dinâmica ou funcional. A UML oferece nove diagramas para a modelagem de sistemas, a saber:

Diagrama dos casos dos usuários ou diagramas de "use cases"

Modelagem de casos de uso, em inglês "use cases", é utilizada como técnica de modelagem dos requerimentos de um sistema sob a perspectiva do usuário. Os casos de uso são

próprios para modelar como um sistema ou negócio trabalha, ou como os usuários desejam que ele trabalhe sem ter que especificar como ele será implementado. O objetivo final de um projeto de software é satisfazer os requerimentos dos usuários de um sistema. Um caso de uso representa um requerimento funcional do sistema como um todo. Os caso de uso são geralmente o ponto de partida da análise orientada a objetos com UML. Um caso de uso descreve um conjunto de seqüências, onde cada seqüência representa a interação de coisas externas ao sistema, os atores, com o sistema. Um ator é um agente que interage com o sistema, um tipo de usuário ou categoria com papel definido, podendo incluir: Seres humanos, máquinas, dispositivos ou outros sistemas. Um diagrama de caso de uso é um diagrama que mostra um conjunto de casos de uso atores e seus relacionamentos. Este diagrama tipicamente é aplicado para modelar o contexto do sistema e para modelar os requerimentos do sistema. A Figura 4-5 apresenta o um diagrama de caso de uso elementar com um só caso de uso.

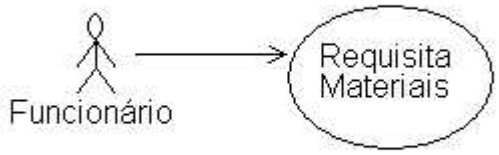

Figura 4-5 Diagrama de caso de uso elementar

Diagrama de classes e diagramas de Objetos

O modelo de objetos mostra a estrutura estática de dados do mundo real e a organiza em partes manipuláveis. Este modelo costuma ter melhor definição, menor dependência de detalhes, maior estabilidade e ser mais facilmente compreendido. Para representar este modelo temos os diagramas de classes e os diagramas de objetos. O diagrama de classes mostra um conjunto de classes e seus relacionamentos. O diagrama de objetos é utilizado para particularizar situações formando cenários e exemplos para discussão. Um dado diagrama de classes corresponde a um conjunto infinito de diagramas de objetos.

Uma classe é a descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica. Classes individuais são representadas em UML como um retângulo sólido com um, dois ou três compartimentos. O primeiro é para o nome da classe o segundo e para os atributos da classe e o terceiro para as suas operações. O primeiro é obrigatório e os outros opcionais. Os atributos e as operações de uma classe podem ter visibilidade pública, protegida, privada, representada nos diagramas respectivamente pelos símbolos: + , # , - . A Figura 4-6 representa as classes que irão formar o simulador do sistema.

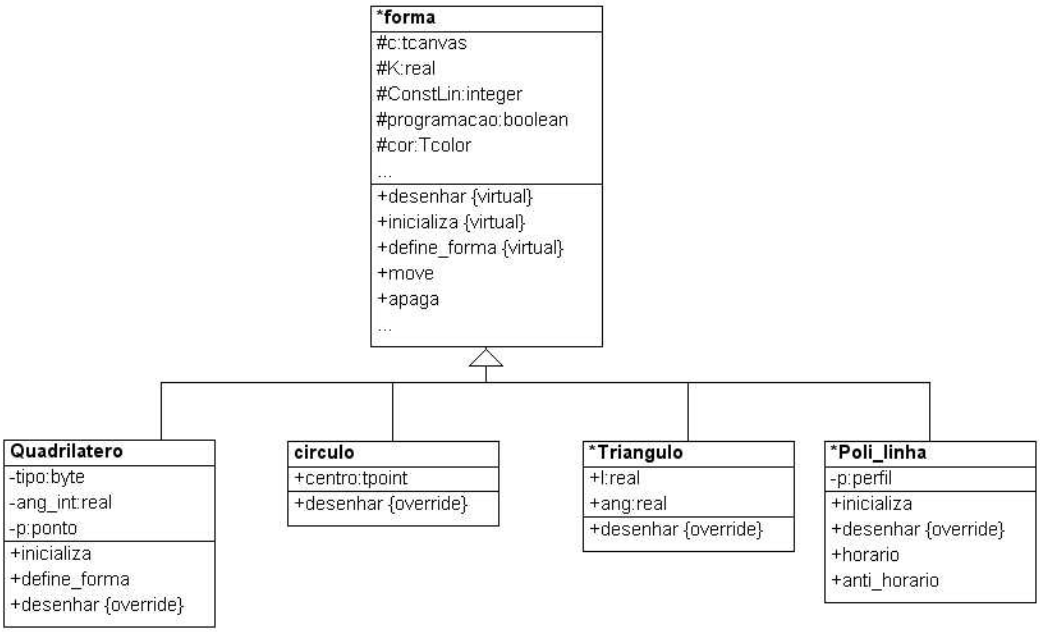

Figura 4-6 Diagrama de classes

Quando se cria um modelo identificamos não somente as coisas que compõem este modelo, mas também modelamos como estas coisas se relacionam umas com as outras. Na modelagem orientada a objetos são os relacionamentos que conectam os diversos objetos em um modelo de objetos. Os três tipos especialmente importantes de relacionamentos são: dependências, generalizações e associações.

A dependência é uma relação semântica entre duas coisas no qual a alteração em uma das coisas, a independente, pode afetar a semântica da outra coisa, a dependente. É indicada graficamente por uma linha tracejada, possivelmente direcionada para aquela coisa da qual se

depende e ocasionalmente rotulada. Usa-se dependência quando se quer mostrar que uma coisa utiliza a outra.

A generalização é uma relação entre uma coisa geral, a superclasse, e especializações desta superclasse. Esta relação é chamada “um-tipo-de” onde uma coisa é uma especialização de uma outra coisa mais geral. Uma classe filha, especialização, herda os atributos e operações da classe pai, a superclasse. Desta maneira os objetos filhos compartilham a estrutura e o comportamento dos pais. Generalização significa que um objeto filho pode ser usado onde a classe pai, aparece, mas não o contrário. Em outras palavras, generalização significa que a classe filha pode ser representada pela classe pai. Uma operação da classe filha que tem a mesma declaração que a da classe pai sobrecarrega esta operação, ao que se denomina polimorfismo. Graficamente a generalização é representada por uma linha sólida com uma grande seta triangular apontada para a classe pai.

Associação é uma relação que descreve um conjunto de vínculos entre elementos de modelo. Quando duas classes (ou mesmo uma classe consigo própria) apresentam interdependência onde determinada instância de uma delas origina ou se associa a uma ou mais instâncias da outra, dizemos que ela apresenta uma associação. Graficamente uma associação é representada por uma linha sólida que conecta classes. Cada associação é acompanhada por um nome, o papel de cada lado da relação, a sua multiplicidade e se for o caso a indicação de agregação, ou seja, um símbolo em forma de diamante. A agregação é um tipo especial de associação, representando uma relação estrutural entre um todo e suas partes. A multiplicidade de uma associação especifica quantas instâncias de uma classe relacionam-se a uma única instância de uma classe associada. Ela restringe a quantidade de objetos relacionados.Uma linha sem símbolos de multiplicidade indica uma associação um para um. Existem terminadores especiais de linha para indicar certos valores comuns de multiplicidade. Um asterisco é o símbolo para muitos, significando zero ou mais. Uma bola vazia indica opcional, significando zero ou um. No caso geral, a multiplicidade é escrita junto à extremidade de linha. Devemos, na análise, primeiramente determinar os objetos, as classes, as associações e depois decidir sobre suas multiplicidades.

Diagramas de interação: diagrama de seqüência e de colaboração

Para se modelar os aspectos dinâmicos de um sistema constrói-se cenários que envolvem a interação entre certos objetos de interesse e as mensagens que podem ser disparadas entre eles. Modelam-se estes cenários por meio de diagramas de interação, dois dos cinco modelos previstos na UML para a modelagem dos aspectos dinâmicos dos sistemas. Um diagrama de interação apresenta um conjunto de objetos e seus relacionamentos incluindo mensagens que são enviadas entre eles. Estes diagramas podem ser construídos de duas maneiras; enfatizando-se a ordem de mensagens entre os objetos, representados pelo diagrama de seqüência, ou enfatizando-se a organização dos objetos que enviam e recebem mensagens, formando o diagrama de colaboração. De qualquer forma os diagramas são semanticamente equivalentes, pode-se converter de um para o outro sem perda de informação.

Graficamente um diagrama de seqüência apresenta objetos arranjados no eixo X ordenados no tempo ao longo do eixo Y. Cada objeto tem sua linha de vida representada por uma linha tracejada vertical e representa a existência de um objeto por um período de tempo. A maioria dos objetos tem sua existência por toda a interação, porém objetos podem ser criados de forma que sua linha da vida começa com a mensagem “create”.Da mesma forma objetos podem ser destruídos com a mensagem “destroy”, finalizando a sua linha da vida. Há no diagrama também a representação do foco de controle, feito através de um retângulo estreito sobre a linha da vida, e mostra o período de tempo durante o qual o objeto esta realizando uma ação. A parte superior do retângulo está alinhada com o início da ação e sua parte inferior com o seu término. A Figura 4-7 apresenta um diagrama de caso de uso onde o administrador do sistema cadastra um usuário para que este possa ter acesso aos seus serviços.

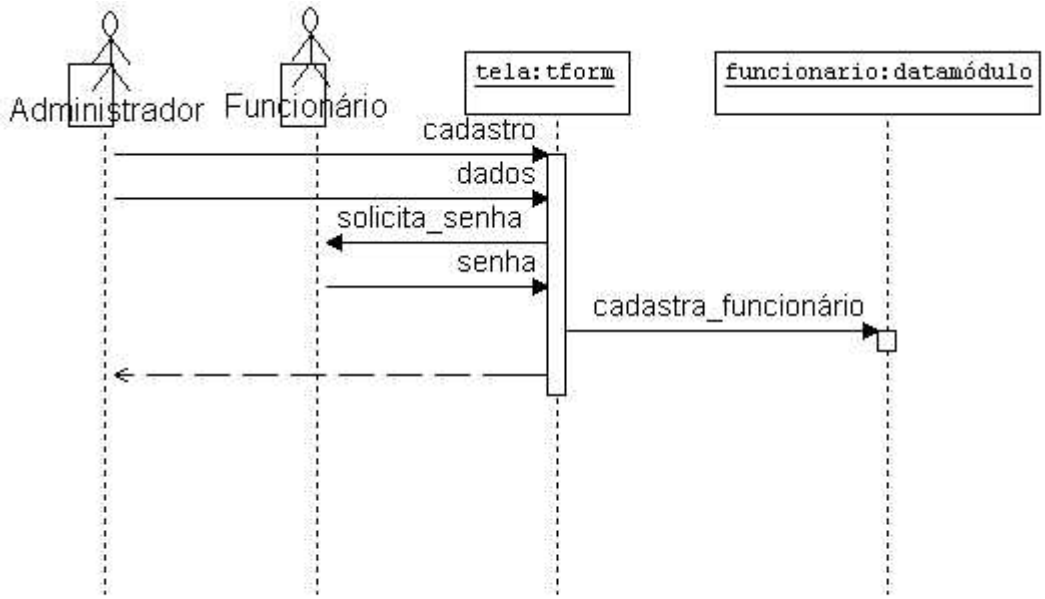

Figura 4-7 Diagrama de seqüência

Graficamente o diagrama de colaboração é um grafo onde os vértices são objetos e os arcos são mensagens ou ligações. Um diagrama de colaboração é formado inicialmente colocando-se os objetos que formam a interação. Posteriormente são ligados, e estas ligações são adornadas com os nomes das mensagens que estes objetos recebem e enviam. A numeração das mensagens indica a ordem delas no tempo.

Diagramas de atividades

O diagrama de atividades é uma variante do diagrama de estado, e também, é um fluxograma interfuncional com a possibilidade do uso de barras de sincronismo para representar concorrência entre as atividades. Ele mostra o fluxo de controle de atividade para atividade podendo ser utilizado para descrever os processos de negócios do domínio do problema.

Uma atividade conceitualmente é alguma tarefa que precisa ser feita, independentemente se for por um computador ou uma pessoa.

Quando uma ação é finalizada o fluxo de controle passa imediatamente para a próxima ação. Este fluxo é especificado através de uma transição, que é representada graficamente como

uma linha direcionada. Podem ocorrer ramificações no fluxo de controle devido às decisões que devem ser tomadas. As decisões são representadas graficamente por losangos, e são definidas por uma expressão lógica que estará expressa no diagrama. Podem ocorrer também fluxos que são concorrentes, de forma que um simples fluxo se bifurca em dois ou mais fluxos paralelos concorrentes de controle. De forma análoga, dois ou mais fluxos de controle diferentes, podem ser sincronizados e convergir para um só fluxo de controle de saída. A sincronização destes fluxos é representada através de uma barra de sincronização representada por uma linha em negrito. A Figura 4-8 apresenta um diagrama de atividades que mostra uma solicitação de materiais pela produção ao estoque.

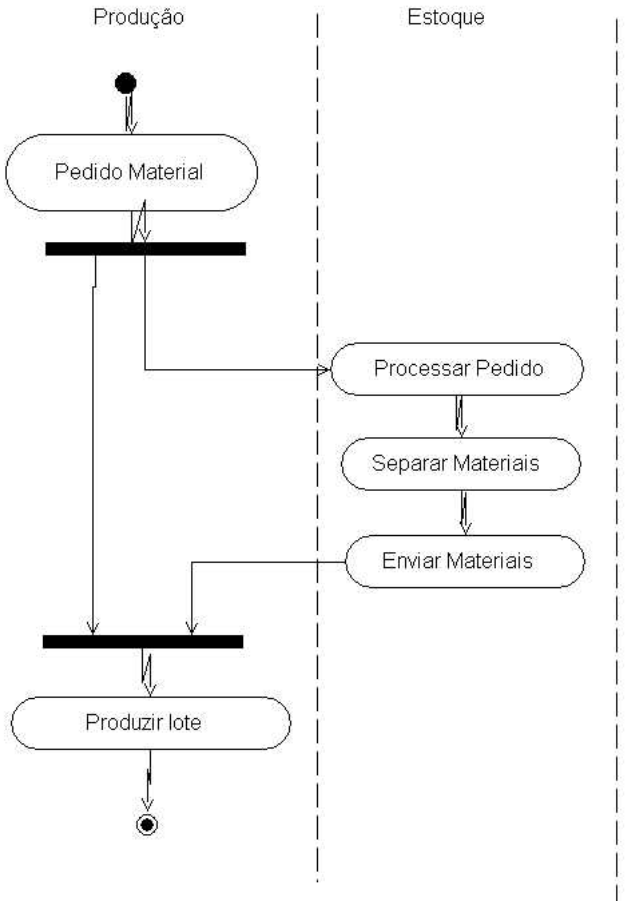

Figura 4-8 Diagrama de atividades

Quando use cases interagem entre si, os diagramas de atividades são uma técnica interessante para representar e facilitar a compreensão deste comportamento.

Diagrama de estados

Um diagrama de estados, é um grafo direcionado cujos vértices são os estados e os arcos transações rotuladas com os nomes dos eventos. Eles são utilizados para modelar aspectos dinâmicos de um sistema. Podemos utilizar diagramas de estados para especificar a vida de instâncias de classes, um caso de uso ou de um sistema. Os diagramas de estado são constituídos pelos estados e pelas transições.

Uma transição é uma relação entre dois estados indicando que um objeto passará para um novo estado quando um evento específico ocorrer e as condições necessárias forem satisfeitos. Uma transição é indicada por seta e é acompanhada pelo evento que determina esta transição e também pela condição a ser satisfeita entre colchetes, caso seja necessário.

Um estado é uma condição ou situação na vida de um objeto durante o qual satisfaz algumas condições, realiza algumas atividades ou espera por eventos. Um estado estável representa uma condição na qual um objeto pode existir por um período identificável de tempo. Um estado é representado por uma figura arredondada contendo um nome opcional.

Chamamos de eventos os estímulos externos que um sistema recebe e em resposta a eles altera seu estado, é a especificação de uma ocorrência que tem localização no tempo e no espaço. Um estímulo de um objeto para outro é também um evento. Um evento idealmente não tem duração.

Os diagramas de estados podem representar ciclos de vida, com um estado inicial indicado por um círculo cheio; estados intermediários, e um estado final representado por um círculo cheio dentro de um outro. Alguns diagramas representam ciclos contínuos onde não há interesse pelo seu início e finais, mas apenas pelo seu contínuo funcionamento.

Diagramas de Componentes

Um diagrama de componentes mostra um conjunto de componentes e seus relacionamentos. Eles são usados para ilustrar a implementação de um sistema. Os componentes normalmente estão relacionados a uma ou mais classes, interfaces e colaborações. Usam-se componentes para modelar coisas físicas que residem em um nó tal como executáveis,

bibliotecas, tabelas, arquivos e documentos. Um componente tipicamente representa um pacote físico de outros elementos lógicos tal como classes, interfaces e colaborações.

Diagramas de Distribuição

Este diagrama mostra um conjunto de nós e seus relacionamentos. São usados para ilustrar a distribuição estática dos sistemas e o hardware. Estes diagramas estão relacionados com os diagramas de componentes, pois um nó tipicamente reúne um ou mais componentes.

Um nó é um elemento físico que existe e representa um recurso computacional, geralmente tendo memória e freqüentemente capacidade de processamento. Representam o hardware no qual componentes são distribuídos e executados.

$\Rightarrow$ Regras

Como qualquer linguagem a UML tem um conjunto de regras que especificam como criar modelos bem formados, ou seja, que sejam autoconsistentes e que tenham harmonia com todos os seus modelos relacionados. A UML tem regras para nomes, escopo, visibilidade, integridade e execução.

A UML tem os seguintes mecanismos comuns que são aplicados consistentemente através da linguagem: Especificações, adornamentos, divisões comuns e mecanismos de extensibilidade. Os mecanismos comuns à linguagem buscam simplificar e harmonizar a modelagem.

4.7 A modelagem Orientada a Objetos e Bancos de Dados Relacionais

O paradigma baseado em objetos é versátil. Ele não só oferece uma sólida base para o projeto de sistemas e para a programação como também pode ser usado para projetar bancos de dados (Rumbaugh,1994).

O paradigma orientado a objetos é baseado na construção de aplicações a partir de objetos que tem ambos dados (atributos) e comportamento, enquanto que o paradigma relacional é baseado no armazenamento de dados. No paradigma orientado a objetos os relacionamentos são estabelecidos de forma distinta do modelo relacional. No modelo relacional os relacionamentos

são implementados basicamente pela duplicação de dados entre as tabelas. Esta diferença fundamental resulta em uma combinação não ideal de dois paradigmas (Ambler,2003).

Porém, muitos sistemas orientados a objetos fazem uso de dados armazenados em bancos de dados. O desenvolvimento destes sistemas pode ser feito utilizando a análise orientada a objetos sendo necessário, ao se projetar o banco de dados, o mapeamento das estruturas do paradigma orientado a objetos para o paradigma relacional. O mapeamento é uma atividade de projeto que pode ser feita de forma sistemática seguindo-se algumas diretrizes. No caso de mapeamento de classes para tabelas temos as seguintes diretrizes principais: Os atributos de uma classe são mapeados como colunas de tabelas. Classes são mapeadas como tabelas, normalmente não em uma relação um para um. Há três modos fundamentais de se mapear herança em um banco de dados relacional. Usando uma só tabela para estabelecer toda uma hierarquia de classes. Usar uma tabela para cada uma das classes concretas. Usar uma tabela para cada classe da hierarquia. Os relacionamentos em um banco de dados relacional são estabelecidos através de chaves estrangeiras. Através delas estabelece-se uma relação entre um registro em uma tabela com um registro em outra tabela. A implementação de um relacionamento um para um ou um para vários é feita normalmente pela inclusão da chave de uma das tabelas na outra. Para relacionamentos vários para vários são necessárias tabelas associativas. Elas estabelecem a relação entre duas ou mais tabelas em um banco de dados relacional.

4.8 A programação orientada a objetos e o sistema Delphi

O sistema Delphi, (Cantù, 2000) integra em um só ambiente a linguagem Object Pascal, orientada a objeto, com recursos de acesso, gerenciamento e criação de banco de dados relacionais, editor de interfaces gráficas, hierarquia de classes,programação cliente servidor, serviços distribuídos e serviços para internet. Este sistema é a ferramenta principal que será usada neste trabalho no desenvolvimento do protótipo do sistema de informações para usinagem .

O sistema Delphi utiliza unidades, “units”, que são bibliotecas que reúnem os elementos fundamentais de um programa Pascal tais como, referências a outras bibliotecas, constantes, tipos de dados, variáveis, procedimentos e as suas implementações. As unidades são as verdadeiras estruturas de reutilização de código. No sistema Delphi as classes são desenvolvidas nas unidades.

No Object Pascal as variáveis de um tipo de classe fazem apenas referência às instâncias da classe, ou seja, são ponteiros para as instâncias. Portanto, quando se declara uma variável de um determinado tipo de classe o objeto não é criado na memória. O objeto deve der criado explicitamente através do método 'create'. Exceções a este modo de operar são os formulários e seus componentes que são construídos automaticamente pelo Delphi.

Na definição das classes o Object Pascal utiliza três especificadores de acesso: 'private', quando os atributos e métodos não são acessíveis fora da definição da classe; 'public' quando os atributos e métodos são acessíveis livremente de qualquer outra parte do programa e 'protected' quando apenas a classe atual e suas subclasses podem acessar os campos e métodos herdados.

Através destes especificadores de acesso as classes garantem sua integridade e controlam o acesso a seus dados, distinguindo os acessos que operam na sua estrutura interna daqueles que são convenientemente deixados visíveis. Os seus dados internos são encapsulados em sua estrutura e são somente manipulados através de métodos projetados para manipulá-los especificamente.

O Object Pascal permite a herança simples. Com a herança uma nova classe, a filha, herda todos os métodos e atributos da classe pai. Uma propriedade interessante, devido à herança, é que podemos atribuir uma instância da classe filha a uma variável da classe pai. Porém o inverso não é verdadeiro. Isto é possível porque a variável filha contém todas os componentes da variável pai. Em Delphi podemos implementar a propriedade do polimorfismo através de funções virtuais e sobrecarga destas funções.

O sistema Delphi tem uma hierarquia de classes que facilita a programação. Muito do trabalho de desenvolvimento passa a ser a de compreender como estes recursos funcionam, como integrá-los, e criar novas classes derivadas daquelas já definidas na hierarquia. A estrutura de biblioteca de classes do Delphi chamada de 'Biblioteca de Componentes Visuais' é formada a partir da classe 'Tobject'. Todas as demais classes são subclasses desta classe originária.

O desenvolvimento de interfaces gráficas com o usuário no sistema Delphi, devido às facilidades disponíveis, é uma atividade quase lúdica, um ponto forte deste sistema. A estrutura principal deste tipo de interface é o formulário, um objeto padrão que fornece uma área de edição

gráfica onde se constrói, usando os elementos visuais disponíveis. O Delphi enquadra controles predefinidos do sistema Microsoft Windows, tais como: botões, caixas de seleção, rótulos estáticos, campos de edição, caixas de listagem, caixas de combinação e barras de rolagem, em alguns de seus componentes básicos. Assim, ele mantém uma conexão íntima com o sistema Operacional. Desta forma usando controles baseados nos recursos do sistema operacional os programas podem ser migrados facilmente para versões do Sistema operacional e manter todos os recursos fornecidos pela versão específica. O uso dos controles do Windows é uma maneira eficiente de reutilizar código e também ajuda a reduzir o tamanho do código compilado.

A linguagem Object Pascal do sistema Delphi não pode ser considerada uma linguagem de programação persistente, mas uma linguagem com recursos de manipulação de dados entre os quais a SQL. Isto exige que a análise não seja totalmente orientada a objetos, mas híbrida, exigindo as devidas considerações em relação ao modelo relacional de tratamento de dados. Além disto o programador deve despender substancial quantidade de código para carregar e descarregar os dados do banco de dados adaptando-os aos objetos em questão (Lischner,2000).

4.9 Objetos e a integração dos sistemas

Ferman (1997), apresenta as quatro técnicas de integração de sistemas mais utilizadas, cada uma representando um diferente nível de complexidade e funcionalidade, Figura 4-9.

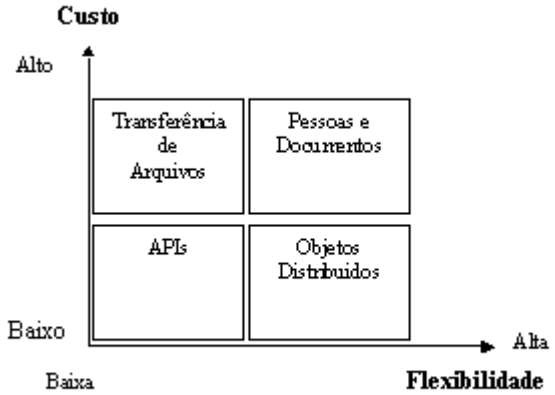

Figura 4-9- Tecnicas de Integração, fonte: Ferman,1997

⇒ Pessoas e documentos: A integração da maioria dos sistemas na maior parte das empresas de manufatura é realizada por pessoas utilizando papéis processados manualmente ou de forma semi-automática. Este método tem um custo alto, ocorrem muitos erros e atrasos, porém é um modo flexível e dinâmico.

⇒ Transferência de arquivos: A transferência de arquivos contendo dados de um sistema para outro é uma abordagem que pode ser facilmente implantada. A extração dos dados de um sistema, o armazenamento e o carregamento dos dados no sistema destino, através da utilização de arquivos, reduz a possibilidade de erros. O processo de transferência de arquivos pode ser iniciado manualmente ou ocorrer em períodos de tempo pré-determinados. Como o processo não ocorre em tempo real, ambos os sistemas mantém cópias dos dados. Esta abordagem reduz desperdícios, porém é menos flexível que a integração por pessoas e documentos.

⇒ API (Application Programming Interface): São programas aplicativos com interface apropriada para situações específicas que são projetados para suprir a necessidade de dados entre os sistemas. Eles freqüentemente removem a necessidade de dados redundantes em ambos os sistemas permitindo acesso em tempo real ao outro sistema. As particularidades das APIs de cada fornecedor resultam na necessidade de mão de obra com habilidades extremamente específicas para o desenvolvimento dos programas de integração. Problemas de segurança podem surgir com esta abordagem. Deve-se destacar, também, que se um dos sistemas for substituído o programa de integração precisa ser refeito (Zancul et. Al. 1999).

⇒ Interfaces baseadas em Objetos Distribuídos: O uso de objetos distribuídos possibilita a integração, por meio de interfaces padronizadas, de sistemas heterogêneos, ou seja, sistemas diferentes baseados em plataformas diferentes localizados em hardware remoto ou não em uma dada rede de computadores.

4.9.1 Arquitetura para acesso a dados

Na integração entre sistemas de natureza distinta através de API's pode-se fazer o acesso direto, por meio dos serviços de rede, a uma base de dados do outro sistema utilizando, por

exemplo, funções SQL. Isto resulta em uma integração muito flexível, porém, existe o risco de se quebrar a lógica do sistema devido à falta de entendimento da base de dados.

Um modo mais seguro é realizado por meio de programas específicos de integração utilizando, ou não, objetos distribuídos. Estas interfaces disponibilizam os serviços que fazem o acesso à base de dados. Estes serviços devem ser robustos de tal forma a determinar claramente o que pode ser feito. Tudo o que não for expressamente permitido é proibido. Isto protege e isola as tabelas de acessos indevidos e minimizam a possibilidade de corrupção dos dados.

Para ambientes que necessitem de acesso a dados de sistemas gerenciadores de bancos de dados da Microsoft estas interfaces podem ser implementadas a partir de objetos ADO ("Activex Data Objects"). Estes objetos estão inseridos dentro da estratégia "Microsoft -Universal Data Access" conhecida pela sigla MDAC ("Microsoft Data Access Components"). Esta estratégia foi desenvolvida para permitir aos utilizadores o acesso às diversas fontes de dados, tais como servidores de bancos de dados e servidores de informação baseados em WEB. Esta estratégia disponibiliza uma interface padrão, onde o acesso aos dados, consultas e ou alterações são feitos de forma transparente. As especificações são independentes de como é feito o armazenamento dos dados, das ferramentas de busca e das linguagens utilizadas. Nesta estratégia está inserida a arquitetura de objetos ADO que são de mais alto nível e por isto são mais fáceis de utilizar. O modelo ADO define uma coleção de objetos que podem ser utilizados em qualquer linguagem de programação em plataformas que suportem COM.

O COM ("Component Object Model) é uma tecnologia desenvolvida pela Microsoft que define um modo padronizado para um módulo cliente e um módulo servidor se comunicarem através de uma interface específica". Um módulo pode ser compreendido como um aplicativo ou uma biblioteca. Os dois módulos podem ser executados no mesmo computador ou conectados através de rede.

4.9.2 Objetos Distribuídos

Um sistema distribuído compreende uma coleção de computadores (nós) autônomos, conectados através de rede física e lógica, equipados com software projetado para produzir uma facilidade computacional integrada. Tais sistemas são construídos sobre plataformas de

“Hardware” geralmente variantes no número e, muitas vezes heterogêneas. O conceito de objetos distribuídos, por sua vez, segue os princípios do modelo cliente-servidor e do paradigma orientado a objetos, trazendo o melhor dos dois mundos: a abstração, que leva à redução da complexidade no desenvolvimento das aplicações, e à concentração de dados comuns (compartilhados por diversas aplicações) num único lugar, o que evita replicações desnecessárias e leva à especialização de serviços (Coelho 1998).

Estes sistemas devem atuar com diferentes equipamentos, sistemas operacionais e possibilitar o uso de serviços desenvolvidos em diferentes linguagens, ou seja, tem a característica de serem compostos de elementos heterogêneos. A característica de heterogeneidade impõe a necessidade de especificações abertas, com interfaces padronizadas e públicas, levando ao desenvolvimento de “middlewares” abertos. Um “middleware” é uma camada de software, residente acima do sistema operacional e do substrato de comunicação, que oferece abstrações de alto nível, com objetivo de facilitar a programação distribuída. As abstrações oferecidas fornecem uma visão uniforme na utilização de recursos heterogêneos existentes nas camadas de sistema operacional e redes (Montez, 1997).

As arquiteturas DCOM(“Distributed Component Object Model”) e o CORBA(“Common Object Request Broker Architecture”) especificam ‘middlewares’. O DCOM foi desenvolvido pela Microsoft e só possui suporte em ambientes “Windows”, e é uma extensão de rede da tecnologia de modelo de objeto componente, COM. Através da tecnologia DCOM, objetos podem fazer chamadas a métodos de outros objetos remotos de forma transparente.

O padrão CORBA é determinado pelo OMG (Object Management Group, 2003), que é um consórcio internacional formado por empresas e instituições de pesquisa para definição de padrões na área de objetos distribuídos. Alguns dos padrões definidos pela OMG incluem CORBA, XML, UML e IIOP. CORBA define um tipo de "barramento de *software*" que permite que componentes de “*software”* sejam conectados juntos formando um sistema coeso.

A arquitetura CORBA permite às aplicações fazerem solicitações a objetos, de uma forma transparente e independente, indiferente à linguagem, sistema operacional, hardware, ou considerações de localização.

Nesta arquitetura é feita a especificação da funcionalidade de um condutor lógico de mensagens, conhecido como ORB ("Object Request Broker"). Ele é responsável por todos os mecanismos requeridos para os objetos enviarem e receberem requisições e, da mesma maneira, receberem respostas as suas requisições, Figura 4-10.

A CORBA utiliza a IDL ("Interface Definition Language"), que é uma linguagem puramente declarativa baseada em C++. São descritos em IDL apenas os tipos, as constantes e as operações necessárias para especificar uma interface de objeto.

Este "barramento de software" possibilita aos usuários escolherem entre diversas possibilidades aquelas que se adequam às suas necessidades. Além disto, possibilitam a intensa troca de serviços e informações (sem redundância) através de interfaces, sem a necessidade do conhecimento de como estes serviços foram implementados ou como serão executados, ou seja, propiciam um alto grau de integração. Como exemplo, pode-se citar o trabalho de Wang (2002) que apresenta o modo de produção baseada em uma rede CAD/CAM dispersa geograficamente para a produção de motocicletas. Através de uma rede dispersa de projeto e manufatura, e das tecnologias Web, CORBA, e STEP, pequenas e médias empresas independentes produtoras de motocicletas, sem considerar o tamanho da empresa, a localização geográfica, ambientes computacionais, e das tecnologias e processos disponíveis, podem se unir em uma empresa virtual e assim rapidamente desenvolver produtos para um mercado que sofre rápidas mudanças.

Baseando a integração de dois sistemas em definições padronizados, podem-se isolar ambos os sistemas proprietários atrás de interfaces padrões. Isto permite que especialistas de cada sistema trabalhem independentemente e que seus trabalhos sejam reutilizados para serem integrados aos outros sistemas. Com o tempo cada companhia irá querer integrar seus sistemas aos outros (Ferman,1997).

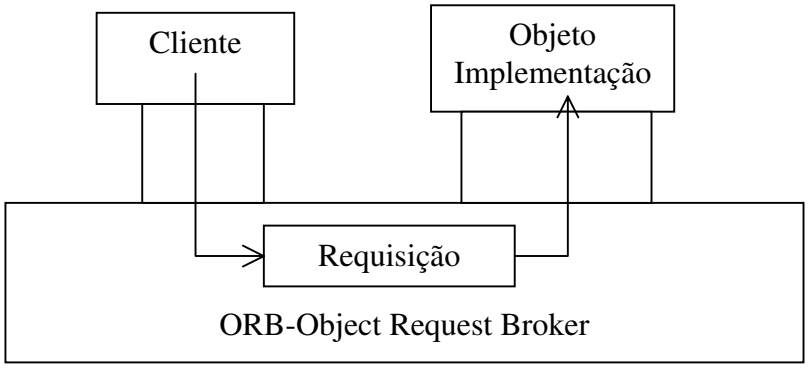

Figura 4-10 Uma requisição passando do cliente para a implementação no objeto. Fonte: Adaptado de OMG

5 Metodologia de desenvolvimento

Neste capítulo é apresentada uma metodologia para o desenvolvimento de software. Ela é projetada para o trabalho em pequenos grupos de desenvolvimento e é indicada para projetos de tamanho pequeno e médio.

As atividades de gerenciamento foram muito simplificadas. Elas não receberam ênfase pelo fato do desenvolvimento não ter sido realizado através de uma equipe típica de desenvolvimento.

5.1 Considerações preliminares

O termo teoria de sistemas foi introduzido pelo biólogo Ludwig von Bertalanffy, o qual iniciou um movimento para uma ciência unificada . A teoria de sistemas procura a formulação de um esquema teórico e sistemático, que permita a descrição de todas as relações que se apresentam no mundo real. Com o advento da Teoria de Sistemas tornou-se evidente e indisfarçável a natureza sistêmica das organizações em geral e das empresas em particular (Chiavenato, 1995).

O processo de fabricação dentro de uma empresa de manufatura pode ser compreendido, segundo a teoria de sistemas, como um subsistema que recebe insumos de entrada (“inputs”), agrega-lhes valor e gera produtos de saída (“outputs”) para um cliente interno ou externo. No caso dos processos de fabricação os insumos e os produtos são tangíveis.

Dentro do conceito da teoria de sistemas, somente o processo apresenta uma saída como produto final. As demais saídas são em forma de decisões que vão conduzir a atuação do processo. Dessa forma, apesar de o processo não ser um elemento de decisão, é uma parte muito importante do sistema, pois é ele que irá dar condições de avaliar decisões tomadas nas camadas superiores (Kwasnicka, 1995).

Automatizar um processo não é sinônimo de melhorar um processo, pois ao automatizar um mau processo, tem-se um mau processo executado com rapidez e com menos esforço. Assim, deve-se ter como objetivo, um método de desenvolvimento de sistema de informação que reorganiza os processos e automatiza os processos certos (Kintschner, 2003).

Problemas organizacionais trazem impactos sobre o desenvolvimento dos sistemas, portanto, é fundamental que os especialistas envolvidos tenham uma visão integrada dos problemas organizacionais e de desenvolvimento de sistemas (Bio,1991).

Desta forma, ao se desenvolver um sistema de informação para a área de fabricação é um momento oportuno para buscar a melhoria dos seus processos através da eliminação de atividades desnecessárias, ineficientes ou que tenham se tornado obsoletas.

Outros fatores de sucesso de desenvolvimento e ou implantação de software são (Correa,1998), (Hull, 2001):

- Estar em sintonia com os objetivos estratégicos da empresa
- Ser gerenciado por pessoas que entendem de mudança organizacional e de negócio.
- Deve ser realizado em equipe com os usuários e pessoal interno da própria empresa e que compreendam os processos envolvidos.
- Comprometimento da alta direção.
- Ter uma boa e robusta metodologia de desenvolvimento com uma abordagem iterativa e incremental e baseada em componentes.

- Gerenciamento dos requisitos para determinar, organizar e documentar as funcionalidades do sistema e suas restrições. Avaliar as alterações dos requisitos.

- Visualização gráfica dos modelos.

- Verificação da qualidade do software através de testes ao final das iterações que permitam que inconsistências nos requerimentos, projetos e implementações sejam descobertos logo de início quando os custos são relativamente baixos.

- Controle de mudanças.

A metodologia descrita a seguir tem como objetivo o desenvolvimento de sistemas utilizando um conjunto mínimo de ferramentas e atividades usando modelagem orientada a objetos.

Ao se aplicar esta metodologia devem ser levados em consideração os fatores de sucesso apresentados anteriormente.

Esta metodologia presume a utilização de uma ferramenta CASE que suporte a geração dos diagramas UML, geração de automática de código e engenharia reversa para obter diagramas a partir do codigo desenvolvido.

Esta metodologia tem duas fases principais: a análise e especificação dos requisitos, e a construção do sistema. Pode haver realimentação da fase de construção para especificação porém isto é indicativo de que as especificações iniciais foram imprecisas ou da necessidade de se avaliar um novo ciclo de vida para o sistema.

5.2 Fase 1: Análise e especificação dos requisitos

Esta etapa inicial do desenvolvimento tem como objetivo a determinação dos objetivos, alternativas e restrições. Determina-se o que deverá ser desenvolvido, e para quem. O resultado final é a definição dos requisitos que o produto de software deverá possuir.

A análise busca o estudo e a compreensão do ambiente onde o sistema será inserido, o qual denomina-se domínio do problema.O domínio do problema é um campo de atividade sob

estudo ou consideração. Um dos maiores problemas encontrados pelos analistas de sistemas é o estudo do domínio do problema e a identificação de suas características (Coad e Yourdon,1992).

Erros nesta etapa inicial podem se mostrar críticos e terem um custo muito alto se forem detectados tardiamente no processo de desenvolvimento.

5.2.1 Levantamento dos Requisitos

⇒ Atividade: Levantamento de informações do Domínio do problema

O levantamento de informações do domínio do problema tem como resultado a descrição do problema, uma síntese após a compreensão em alto nível do domínio do problema. A descrição do problema é feita de forma narrativa e apresenta os razões e objetivos do projeto.

⇒ Atividade: Representar e analisar os processos relacionados

Os processos relacionados ao sistema que será desenvolvido devem ser representados e analisados. Nesta etapa os processos que não agregam valor ou muito simples e ou com baixo nível de automação devem ser analisados com cuidado pois podem ser aglutinados ou eliminados. Processos realizados em duplicidade devem ser eliminados.

⇒ Atividade: Levantamento dos requisitos de alto nível

Faz-se inicialmente o levantamento dos requisitos de forma narrativa produzindo o Documento de Requisitos, que serve como material de discussão e estudo. Neste documento faz-se a identificação das funcionalidades do sistema.

Nesta atividade o problema pode ser subdividido em módulos. Um módulo é uma funcionalidade bem definida do sistema maior, e que apresenta uma interface bem definida e normalmente pequena com os outros módulos. Este módulo irá se constituir de um pacote de classes associações, operações, eventos e restrições. As iterações em cada uma das fases são determinadas pela conclusão das atividades associadas a estes módulos. Uma iteração é um conjunto de atividades, com um plano básico e critérios de avaliação que resultam em uma versão da documentação ou do software.

O resultado deste trabalho é o documento de requisitos do sistema.

O texto deve ser revisado por todos os envolvidos no projeto e pelos potenciais usuários do sistema.

⇒ Atividade: Determinação dos casos de uso

Identifica-se inicialmente a partir do documento de requisitos os principais casos de uso e os atores do sistema. O resultado desta atividade é uma lista com os nomes de cada caso de uso suas descrições e respectivos atores associados.

Estes casos de uso representam os requisitos do sistema em alto nível e servem com unidade de trabalho e medida de sua evolução e entrega ao usuário.

⇒ Atividade: Primeira validação pelos usuários

A lista de casos de uso é apresentada aos usuários para validação, o usuário aprova os casos de uso ou reprova apresentando nestes casos as razões para tal.

⇒ Atividade: Correção dos casos de usos não aprovados

É feita uma análise e os erros são corrigidos ou as razões para a não correção são apresentadas. Este trabalho é apresentado ao usuário voltando a um novo ciclo de validação. Este ciclo iterativo irá até que os problemas sejam resolvidos. Ao final desta etapa é gerada a lista de casos de uso revisada. Os diagramas de casos de uso são desenvolvidos.

⇒ Atividade: Modelagem do domínio

A modelagem do Domínio é a tarefa de descobrir os objetos que representam as coisas e os conceitos que envolvem o domínio do problema do sistema em questão

O modelo do domínio é feito concomitantemente ao levantamento de requisitos.

Nesta modelagem se busca descobrir quais são as principais classes de objetos que formarão a estrutura do sistema. É essencialmente um trabalho de abstração do analista para descobrir as classes que representam as principais entidades e conceitos do sistema e como se

relacionam. O modelo do domínio serve como uma síntese da idéia do sistema e glossário para o levantamento de requisitos. O modelo do domínio é representado pelo Diagrama de Classes do sistema.

⇒ Atividade: Determinação do curso típico de eventos de cada caso de uso

A partir de cada caso de uso da lista são elaborados os cursos típicos de eventos que são compostos por interações entre o ator e o sistema. Um curso típico de ação sempre inicia com uma ação do ator e prossegue com a interação entre o ator e respectivas respostas do sistema. Podem surgir caminhos alternativos e exceções que devem ser considerados. Protótipos das interfaces com os usuários podem ser utilizados para facilitar a determinação da interação sistema usuário. Nesta etapa é interessante o uso de diagramas de seqüência.

⇒ Atividade: Segunda validação dos usuários

Todo o material elaborado é apresentado aos usuários para uma segunda validação. Assim o trabalho de análise e toda a modelagem poderão ser avaliados pelo usuário. Além das listas de casos de uso com suas descrições, interfaces, diagramas, são apresentados os respectivos cursos típicos de eventos e seus caminhos alternativos. É através dos cursos típicos de eventos que o usuário tomará conhecimento sobre o que é feito para se cumprir determinada funcionalidade do sistema, ou seja, quais passos devem ser seguidos para chegar-se a um dado objetivo.O usuário deverá aprovar os casos de usos ou reprová-los, neste caso apresentando as razões da reprovação.

⇒ Atividade: Correção dos casos de usos não aprovados

Após a correção dos erros, os casos de uso vão sendo reapresentados para novas tentativas de validação. Este ciclo irá até a completa compreensão e aprovação dos casos de uso.

5.3 Fase 2: Construção do sistema

5.3.1 projeto

Uma vez determinado “O que deve ser feito”, prossegue-se com o projeto onde a questão principal é “como” o sistema será implementado. No projeto, decide-se como o problema será

resolvido, primeiro em alto nível e depois em níveis cada vez mais detalhados. Nesta etapa os principais recursos da ferramenta CASE a serem utilizados são o editor de diagramas de seqüência e os geradores de código.

⇒ Atividade: Determinação das principais funções

A partir do curso típico de eventos de cada caso de uso cria-se a seqüência de troca de mensagens entre os objetos do respectivo caso de uso. A troca de mensagens se dá entre atores, objetos limites e entidades. Os atores são elementos externos ao sistema que o estimulam. Um ator representa uma função que um usuário pode assumir em relação a um sistema. Também pode ser uma entidade tal como outro sistema ou um banco de dados externo ao sistema que esta sendo modelado. Os objetos limite são as interfaces gráficas com o usuário tais como telas e formulários, e as entidades são os objetos do sistema onde os dados normalmente ficam armazenados. As regras são simples: os atores só podem se comunicar com objetos limites. As entidades não podem se comunicar diretamente com os atores.

Nesta atividade, ao se representar os cursos típicos e alternativos de eventos usando os objetos do sistema em diagramas de seqüência, determinam-se as principais funções que compõem cada caso de uso e as interações entre as principais classes descobertas no modelo do domínio. Desta forma detalha-se o comportamento dos objetos e descobre-se a localização apropriada para atributos e operações.

A partir dos diagramas de seqüência o gerador de código da ferramenta CASE cria a estrutura do sistema em código numa determinada linguagem, por exemplo, Object Pascal. Esta estrutura será utilizada na programação das classes e funções.

⇒ Atividade: Definição das ferramentas de programação, algoritmos e estruturas de dados.

Nesta etapa devem ser definidas as ferramentas para: codificação do sistema, modelagem do sistema, desenvolvimento das tabelas, aplicativos e Gerenciamento do Banco de Dados.

Devem ser escritos os principais algoritmos a serem utilizados no sistema.

Devem ser definidas as interfaces gráficas e as estruturas de dados que serão utilizadas.

5.3.2 Implementação e teste

⇒ Atividade: Codificação e testes

Uma vez gerado o código base do sistema com a estrutura das classes e suas principais operações, elas devem ser codificadas em uma linguagem de programação. Como as classes e operações estão claramente planejadas e definidas a tarefa de programação pode ser facilmente dividida, caso se esteja trabalhando em equipe. Na atividade de codificação, certamente outras classes e operações acessórias devem ser criadas, porém elas deverão ser consistentes com as que foram planejadas. O projeto de classes e operações facilita a organização e dificulta a criação de estruturas conflitantes que não foram planejadas. Por outro lado impossibilita que estruturas essenciais que foram planejadas sejam omitidas.

Testes intensivos são conduzidos para certificar-se de que o sistema produza os resultados desejados. Teste é um conjunto de atividades que pode ser planejado antecipadamente e realizado sistematicamente (Pressman,1995). Os testes de unidade consistem em testar cada unidade de programação separadamente. Eles são escritos simultaneamente à codificação das funcionalidades de tal forma que estas são desenvolvidas e prontamente testadas. Os testes de integração visam verificar o sistema funcionando como um todo.

O teste de aceitação fornece a certificação final de que o sistema está pronto para ser usado num ambiente de produção.

⇒ Atividade: Modelagem de Classes

Conforme o sistema vai sendo programado, classes acessórias e novas operações vão sendo incluídas, com isto é necessária a atualização do modelo de classes. Isto pode ser feito por meio de engenharia reversa. A ferramenta Case interpreta o código e atualiza o modelo de classes criando o diagrama de classes atualizado. Este diagrama é importante na manutenção do sistema.

6 Desenvolvimento do Sistema

Neste capítulo é apresentado o desenvolvimento do sistema objeto deste trabalho utilizando-se a metodologia apresentada. Toda a notação e diagramas utilizados seguem o padrão UML. O protótipo, resultado do desenvolvimento, apresenta as principais funcionalidades previstas, e poderá ser utilizado como a base do desenvolvimento de uma versão completa.

O sistema tem como função principal a programação de centros de torneamento em pequenas e médias empresas do setor metal mecânico que trabalham na produção de pequenos lotes de peças utilizando máquinas CNC, podendo ser utilizado para fins de ensino.

6.1 Fase1: Análise e especificação dos requisitos

6.1.1 Levantamento dos requisitos

⇒ Documento dos requisitos

Visão Geral do Sistema:

O sistema é constituído de cinco módulos principais: O editor de programas CNC para torneamento, o simulador do processo de torneamento, o sistema de gerenciamento do estoque, o sistema de gerenciamento de compras, e a base de dados de materiais e ferramentas, Figura 6-1.

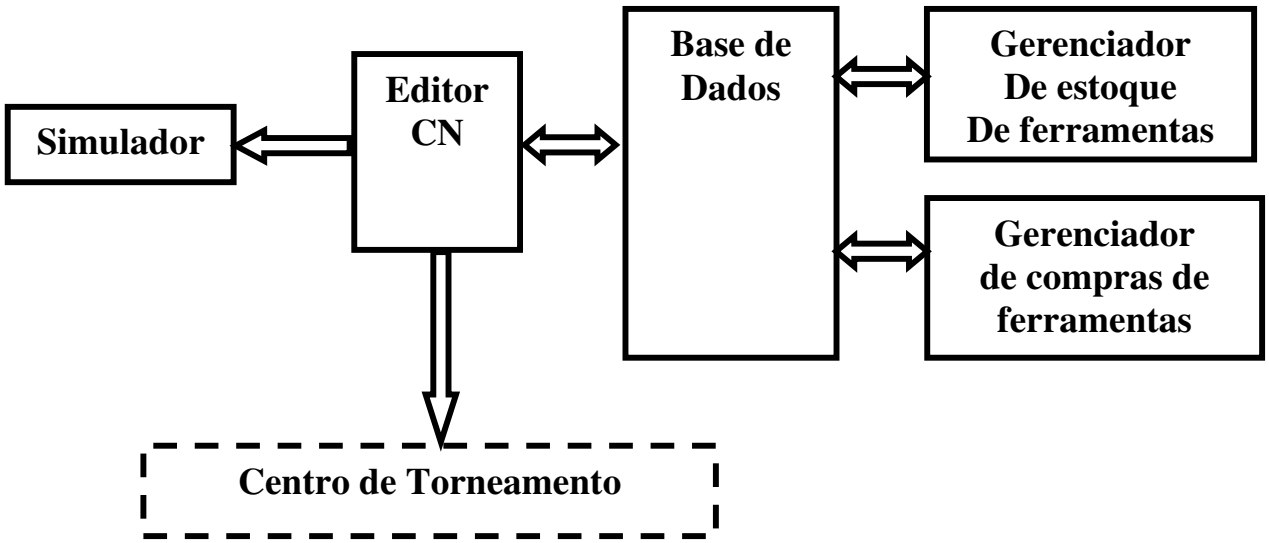

Figura 6-1 - O editor de programas CN

O Editor

O editor tem uma interface que atua principalmente sobre um objeto peça. O objeto peça armazenará todos os dados, geométricos e tecnológicos de uma peça rotacional. Através do editor é possível ao operador coletar, apresentar e alterar os dados de uma peça, enviar para a máquina programas CN, receber das máquinas programas CN, processar os dados fornecidos pelo programador gerando automaticamente o programa CN, salvar as descrições das peças na forma de arquivos, emitir orçamentos e estimativas de tempo.

Deverá ser desenvolvida uma função para o cálculo da quantidade de ferramentas para o lote de ferramentas desejado. Isto será feito para cada operação em específico, ou seja, em cada operação, será feito o cálculo de tempo efetivo de corte para cada aresta de corte. Assim, com a informação do tamanho do lote podemos calcular quantas arestas serão necessárias e, portanto requisitar os insertos de forma mais criteriosa e precisa, requisitando somente o número necessário de ferramentas ao estoque.

Para se descrever geometricamente uma peça utiliza-se perfis. Como no processo de torneamento existe a simetria de revolução, dois perfis, um representando a peça antes do processo e a outra com o perfil final, descrevem o processo quanto ao seu aspecto geométrico. O perfil final pode, dependendo da peça, ser desdobrado em dois: o perfil externo e o perfil interno. O perfil interno é aquele que descreve a peça em sua parte mais próxima ao eixo de revolução, e geralmente se realiza a partir de uma operação de furação. Na confecção de uma peça pode ser utilizado apenas um destes perfis ou os dois. Um perfil é descrito através das coordenadas cartesianas de pontos e curvas que ligam estes pontos. Estas curvas pertencem a apenas dois grupos: lineares (retas) e circulares. Operações que geram um perfil mais complexo ou detalhes, como sangramento e rosqueamento são tratados e acrescentados posteriormente à definição dos perfis.

A descrição geral de uma peça é armazenada em uma forma textual seguindo uma sintaxe apropriada gerando o programa neutro. Este programa necessita ser adaptado às características de cada centro de torneamento, portanto, é necessária a conversão do programa neutro para a linguagem apropriada de cada CNC. Isto é feito por um programa ao qual denomina-se pos-processador. É necessário um pós-processador específico para cada tipo de CNC.

Na programação o usuário pode partir de um arquivo de peça já existente ou começar o trabalho a partir de um arquivo novo que representa uma nova peça.

Caso o usuário inicie com um arquivo novo, o usuário deve iniciar o processo pela opção torneamento. Caso escolha esta opção, somente estará ativa a opção roteiro para programação que o guiará. Este roteiro inicia com uma tela para definição de dados da máquina e da peça. Os dados da maquina são: O comando do centro de torneamento, a unidade a ser utilizada (milímetros ou polegadas), e a referência da programação (raio ou diâmetro). Por definição a máquina tem torre traseira e não será considerado o contraponto. Os dados da peça são: O tipo da peça bruta (cilíndrica ou forjada), operações que serão utilizadas (externas, internas, externas e internas), o algoritmo para modo de produção com cálculo da velocidade de corte (máxima produção, mínimo custo, definida pelo usuário), o tamanho do lote de peças que serão produzidas, e material da peça que o usuário selecionará de um dos materiais disponíveis no banco de dados de materiais.

O sistema editor deve ter a disposição informações sobre o estoque de matéria prima e ferramentas.

Caso o programador selecione os modos de produção: máxima produção ou mínimo custo o sistema fará o cálculo dos parâmetros tecnológicos usando a fórmula de Taylor para cada operação a partir dos dados armazenados no banco de dados das ferramentas selecionadas (X e K de Taylor).

Caso o usuário tenha determinado que a peça bruta é um cilindro (e não um forjado) e necessita de operações externas e internas e que as velocidades de corte das operações são definidas pelo usuário. O sistema apresenta uma tela para obtenção do comprimento e largura da peça bruta e solicita a descrição dos perfis externo e interno. A descrição do perfil é feita em uma tela onde os segmentos são definidos pelo ponto inicial, ponto final e o tipo de interpolação (linear, circular horária, circular anti-horária). Caso o tipo de interpolação seja circular será solicitado também o raio. O usuário poderá acompanhar a descrição do perfil graficamente por meio de uma simulação no canto inferior esquerdo. O usuário pode percorrer a descrição do perfil através de comandos para avançar e recuar, corrigindo os dados se necessário. Ele define o final do perfil através do botão finaliza. Caso deseje finalizar no meio de um perfil já definido os demais segmentos serão eliminados, nestes casos onde serão eliminadas definições sempre deverá ser apresentada uma mensagem de advertência e confirmação. Por definição o ponto de início e final do perfil deve fazer intersecção com o perfil bruto criando uma região fechada no plano. O valor do sobremetal não será representado graficamente, seu valor será acrescido ao perfil acabado pelo algoritmo que faz o cálculo dos percursos das ferramentas. Este processo será feito uma vez para o perfil externo e outra para o interno. Feito isto o usuário definirá as operações que serão realizadas, através de uma tela para definição das operações. Esta tela obtém as seguintes informações: Tipo da operação(Desbaste externo, desbaste interno, acabamento externo, acabamento interno, furação, roscamento, sangramento), posição no magazine onde está a ferramenta referente à operação, o ponto de troca desta ferramenta, o sentido de rotação do eixo-árvore(horário, anti-horário), fluido de corte (sim, não), profundidade de corte, avanço, velocidade de corte, e a ferramenta e inserto serão selecionados pelo usuário entre os disponíveis no banco de dados de ferramentas. Dependendo do tipo de operação, outros dados serão solicitados. No caso da operação de desbaste será pedido o sobremetal em x e z. De forma

semelhante à definição dos perfis, o usuário pode percorrer a descrição das operações através de comandos para avançar e recuar, corrigindo os dados se necessário. Para determinar o final das operações o usuário deve acionar o botão finaliza. Caso deseje finalizar no meio de uma lista de operações já definidas as demais operações serão eliminadas.

Caso o usuário tenha determinado que a peça bruta é um forjado e necessita de operações externas e internas e o algoritmo para o modo produção é definido pelo usuário. Serão solicitados os sobre-metais que o forjado oferece em relação aos perfis, os demais passos são iguais ao caso da peça bruta cilíndrica.

Caso o usuário restrinja as operações a externas ou internas, serão solicitados apenas dados de perfil e operações relativas a estas operações.

Caso o usuário tenha determinado que a peça bruta é cilíndrica e necessita de operações externas e internas e o algoritmo para o modo produção é de máxima produção ou mínimo custo o processo se difere do descrito no parágrafo anterior pela tela de operações, onde não serão solicitados os dados tecnológicos da ferramenta (profundidade de corte, avanço, velocidade de corte), estes dados serão obtidos do banco de dados ao se fazer à seleção das ferramentas.

Uma vez definidos os perfis e as operações o sistema gera o programa CN referente à descrição apresentada e apresenta uma simulação na tela. Feito isto solicita um nome para salvar o programa, que será salvo em um arquivo com extensão CNC e o programa neutro com extensão NTR. Caso o usuário não deseje salvar o trabalho ele escolhe o botão cancelar.

Feito o programa caso o usuário selecionar o comando orçamento é apresentado um orçamento com os custos por peça e para o lote total que será fabricado. Os tempos necessários para cada operação serão calculados pela Equação 10 localizada no Anexo II - Cálculo dos tempos efetivos de corte. A partir dos dados: tamanho do lote, das constantes X e K de Taylor, e do tempo calculado o sistema calcula o número de insertos que irá necessitar para cada operação.

Feito o programa caso o usuário selecione o comando requisição é feita a solicitação ao estoque as peças e materiais para a produção das peças, na quantidade especificada no orçamento. O usuário deverá informar seus dados e senha para que a transação seja concluída. Se os

materiais estiverem com estoque abaixo do estoque de segurança deve ser feita a solicitação automática de compras destes materiais.

Feito o programa, caso o usuário desejar pode apresentar no ambiente de edição o programa neutro e gerar automaticamente o programa CN selecionando as opções programa_neutro e programa CN incluídas na opção torneamento.

Feito o programa caso o usuário deseje imprimir o texto que esta no ambiente de edição deve selecionar impressão na opção arquivo.

Caso o usuário decida utilizar um arquivo de peça já existente ele pode alterar esta peça selecionando a opção editar. Desta opção ele pode editar os dados de configuração da peça, os dados dos perfis e os dados das operações.

O Simulador

O simulador deve interpretar o programa CN e apresentar o processo de torneamento graficamente. Este subsistema apresenta a peça bruta, os perfis, uma ferramenta por vez e os movimentos envolvidos. A peça sempre será representada no primeiro quadrante do espaço cartesiano com referência na origem das coordenadas. Com o uso do simulador, o programador pode visualizar o processo que está sendo desenvolvido e assim detectar problemas tais como dados de entrada errados que possam causar colisões. Desta forma o simulador é um recurso a mais que garante a qualidade do programa e a segurança do processo.

Este simulador trabalhará com dois tamanhos de apresentação. O primeiro com tela pequena, utilizada principalmente na descrição de perfis. O segundo com tela grande será usada na simulação das operações que constituem a peça.

As funcionalidades do simulador serão: gerar formas (perfis, peça bruta e ferramenta), mover formas linearmente, mover formas em movimento circular horário e movimento circular anti-horário.

O objeto peça controla o simulador, especificando através de mensagens ao objeto ferramenta as ações que o simulador realizará.

Banco de dados

- O Sistema de estoque

O sistema de estoque apóia principalmente o estoquista no seu trabalho de gerenciamento do almoxarifado. O almoxarifado visa a fiel guarda dos materiais confiados pela empresa, objetivando sua preservação e integridade até o consumo final. Segundo Viana (2000), atualmente pode-se definir almoxarifado como o local destinado à fiel guarda e conservação de materiais, em recinto coberto ou não, adequado a sua natureza, tendo a função de destinar espaços onde permanecerá cada item aguardando a necessidade do seu uso, ficando sua localização, equipamentos e disposição interna condicionados à política geral de estoques da empresa.

Pressupõe-se que estoque tem características de consumo regular. O estoquista pode, se concluir que um ou mais itens serão utilizados com regularidade, incluir novos itens no estoque.

O estoquista é apoiado pelo SI nas seguintes atividades: cadastrar itens de estoque, receber materiais, processar pedido, solicitar compra, armazenar, separar e enviar materiais. Na atividade de receber materiais o sistema apóia o estoquista ao apresentar os locais adequados de armazenamento. Na atividade de processar pedido o sistema apóia o estoquista ao apresentar os locais dos itens solicitados e a quais locais enviar os materiais.

O cadastro de itens de estoque inclui um novo item de estoque no banco de dados.

- O sistema de Compras

O sistema de compra apóia o trabalho de gerenciamento dos pedidos de compras. Este sistema apresenta as funções: cadastrar fornecedores novos, processar pedidos de compra de materiais, autorizar o fornecedor a realizar o fornecimento dos pedidos de compra e controlar o encerramento dos processos. Cada uma destas operações deverá necessariamente ser feita por um funcionário com os atributos de comprador com o fornecimento de sua identificação e senha. As solicitações de compra devem passar por análise e serem autorizadas.

- A administração do sistema

O sistema deverá ter um responsável pelo sistema e pela sua integridade, o administrador. A sua função operacional básica é o cadastro dos funcionários e de seus dados e senhas.

⇒ Atividade: Representar e analisar os processos relacionados

A Figura 6-2 presenta através de um diagrama de atividades os principais processos relacionados com o Sistema de Informação. A descrição destes processos é feita a seguir.

Produção:

A função produção na organização representa a reunião de recursos destinados à produção de seus bens e serviços (Slack,1997).

Programar Peça: O programador apoiado pelo sistema de edição de programas CN, objeto deste trabalho, faz a sua descrição geométrica e das operações necessárias utilizando os formulários apresentados pelo editor. Ao final o sistema gera o programa CN da peça.

Requisitar Material: O programador, uma vez gerado e testado o programa CN e aprovado o orçamento, faz a requisição dos materiais.

Produzir lote: A partir do programa CN é produzido o lote de peças.

Devolver Material: O operador faz a devolução das ferramentas que não foram utilizadas e também aquelas que podem ser reutilizadas.

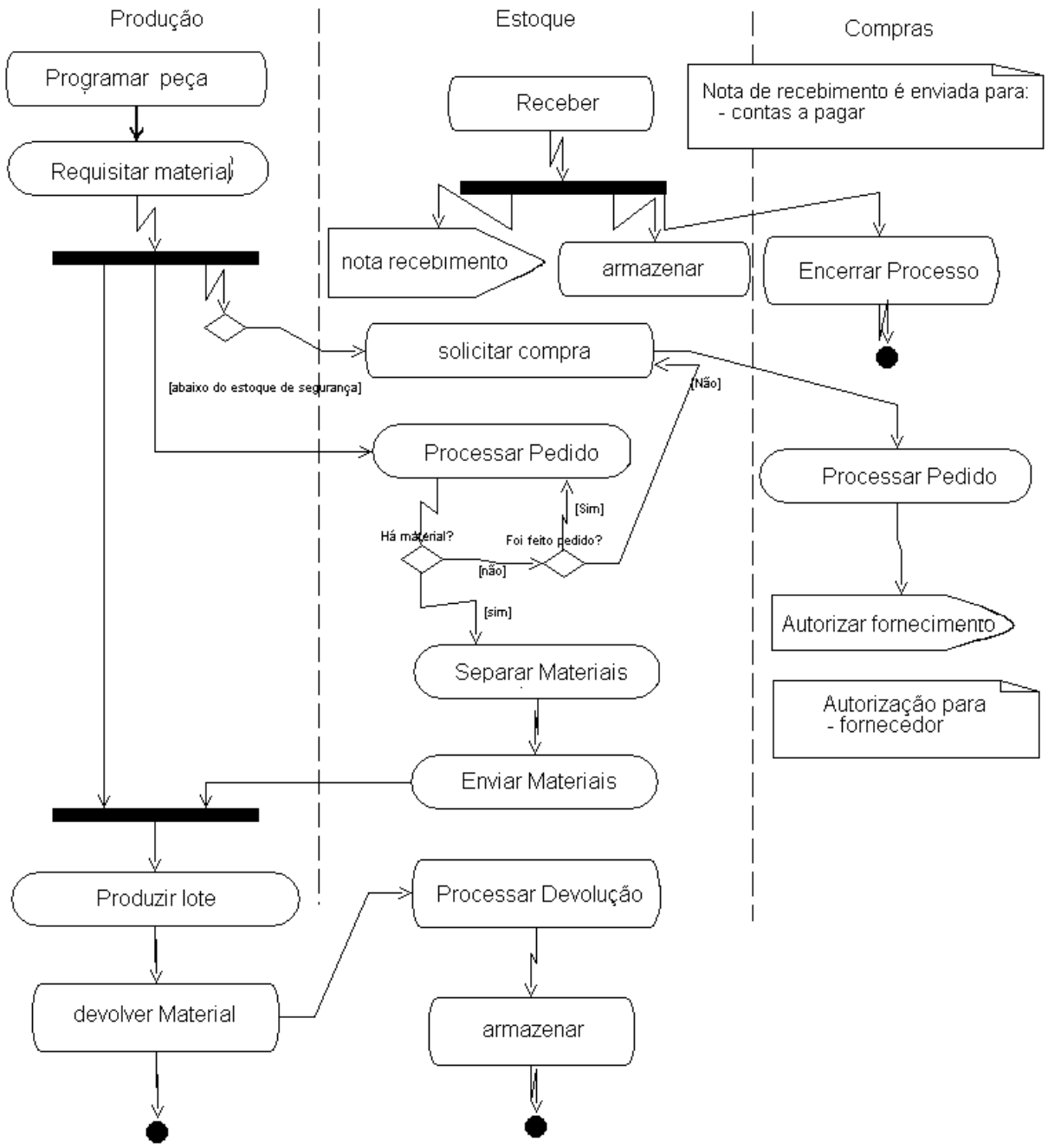

Figura 6-2 Diagrama de atividade representando os processos relacionados

Estoque:

Receber: É o processo de recebimento do material, o estoquista confere o material com a nota fiscal e com os respectivos registros e controles de compra. Desta confrontação o material pode ou não estar de acordo. Caso não esteja de acordo é executado o procedimento de entrada com divergências. Caso esteja de acordo é feita a nota de recebimento de materiais que propicia o encerramento do processo em compras e o pagamento através de contas a pagar. O processo de entrada com divergências não será tratado pelo SI.

Armazenar: É o processo de alocar os materiais em seus devidos lugares. Cada item deve ter seu local de armazenamento definido no cadastro do item.

Solicitar Compra: A área de estoque pode solicitar a compra de itens que estão em falta ou que estão abaixo do estoque de segurança. Esta solicitação pode ser ativada pela produção caso detecte que o item em questão esteja com o seu estoque abaixo do estoque de segurança.

Processar Pedido: O estoquista verifica a requisição dos materiais e com a devida autorização atende as solicitações, faz a lista dos itens a serem separados sua localização e quantidade. Caso algum item esteja em falta o estoquista verifica se já foi solicitada a compra, caso não tenha sido o estoquista faz a solicitação. Após o envio de cada item ao solicitante é dado como atendido e concluído.

Separar Materiais: Com a lista de itens que contém a descrição dos materiais, localização, quantidade total e quantidade requisitada, é feita a separação dos materiais.

Enviar Materiais: Os itens solicitados são acondicionados em meio de transporte adequado e entregues aos solicitantes.

Processar Devolução: Caracteriza-se como entrada de material no estoque, estando na prática, sujeita aos mesmos crivos de recebimento, sob o enfoque do SI é um modo diferenciado da operação receber.

Compras:

O setor de compras tem por finalidade suprir as necessidades da empresa mediante a aquisição de materiais e/ou serviços, emanadas por solicitações dos usuários.

Processar pedido: A partir das solicitações de compra é feita a montagem do processo de compra, faz-se a seleção dos fornecedores e a contratação do fornecimento.

Autorizar Fornecimento: enviado o pedido ao fornecedor com as especificações necessárias, quantidades e prazo de entrega acordado.

Encerrar Processo: Recebido e inspecionado o material o processo de compra é encerrado.

⇒ Determinação dos casos de usos

A Figura 6-3 e a Figura 6-4 apresentam os Diagramas de casos de uso de alto nível do sistema. O texto que descreve estes casos de uso está incluído no Anexo III - Casos de uso.

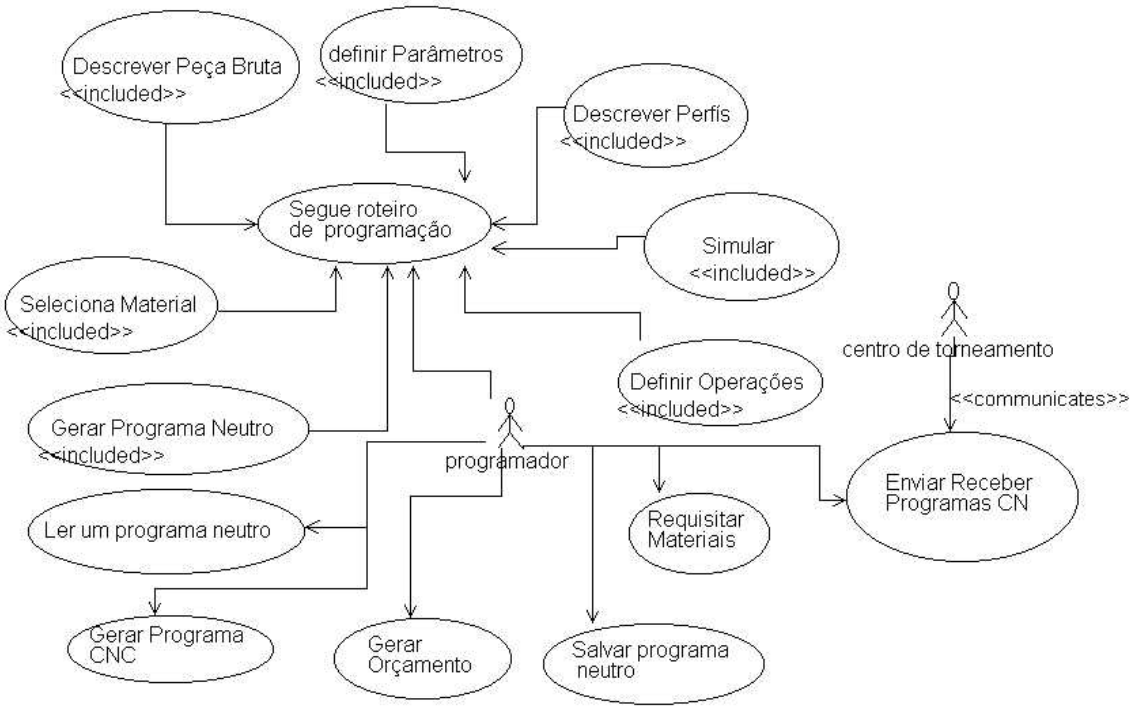

Figura 6-3 -Diagrama de casos de uso do Sistema Editor CNC

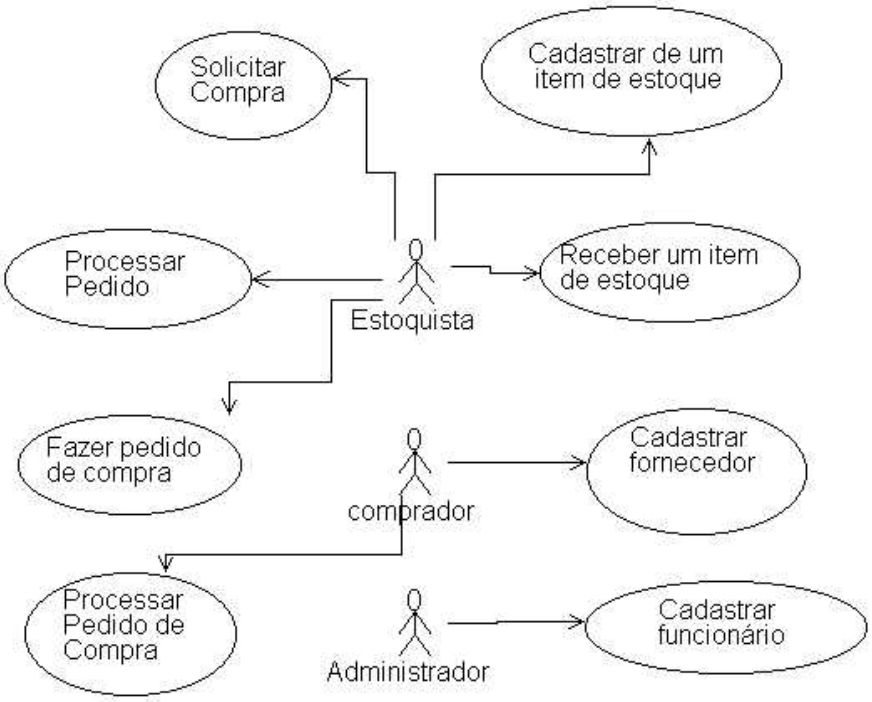

Figura 6-4 Caso de Uso do Banco de Dados

⇒ Modelo do Domínio

Foi proposto o modelo do domínio representado pela Figura 6-5, que serviu como planta baixa do sistema e um glossário para o levantamento dos requisitos. Neste diagrama de classes estão representadas apenas as operações que foram implementadas, no sistema completo as operações de torneamento interno e as demais seriam representadas.

As classes que compõem este modelo são:

Peca – É a classe principal do sistema. Armazena todos os dados geométricos e tecnológicos de uma peça sob processo de torneamento.

Perfil – É uma lista de segmentos. Os perfis armazenam os dados da descrição geométrica da peça.

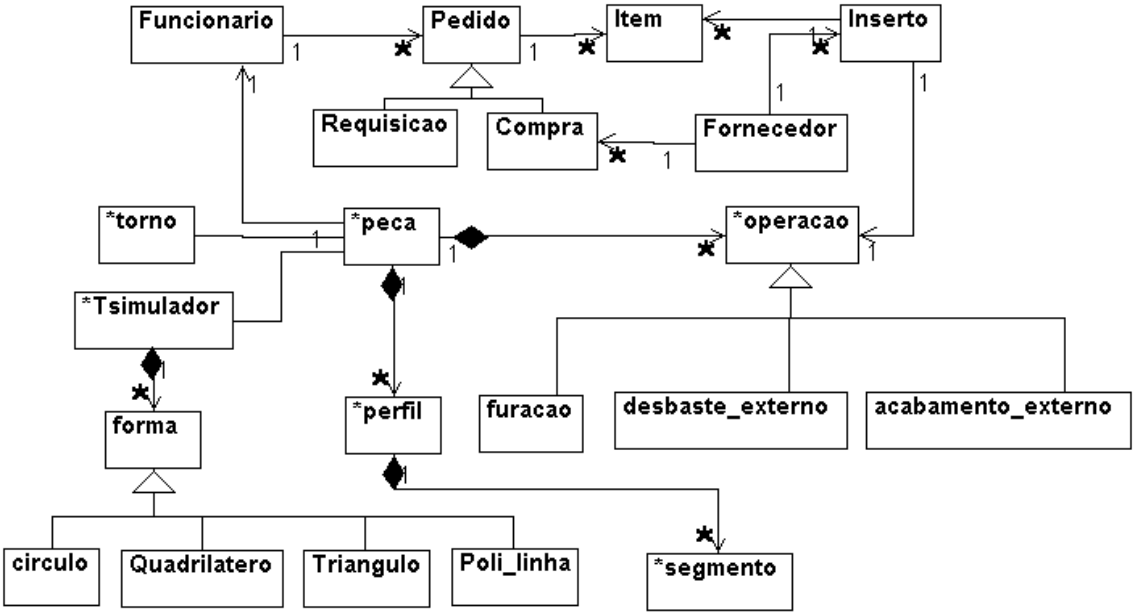

Figura 6-5 - Diagrama de classes representando o domínio do sistema

Segmento – É uma classe que armazena e processa as informações de um ponto meta no plano e uma curva que liga um ponto origem a este ponto. Esta curva pode ser linear ou circular.

Simulador – É a classe responsável pela simulação gráfica do processo a partir do programa CN gerado pela classe peça.

Forma - Classe genérica que dá origem às figuras geométricas que constituem o simulador.

Quadrilatero: Forma geométrica utilizada no simulador para representar peça ou/e ferramenta com forma retangular, quadrada ou de losango.

Triangulo: Forma geométrica utilizada no simulador para representar ferramenta com forma triangular.

Circulo: Forma geométrica utilizada no simulador para representar ferramenta com forma circular.

Poli_linha: Forma geométrica composta por curvas lineares e circulares, é utilizada para representar um perfil.

Torno – Classe que armazena os dados associados à máquina ferramenta, ao comando numérico e, também, aos tempos e custos associados à máquina que realiza o processo de torneamento.

Operacao – Classe genérica que armazena as informações sobre as operações mais comuns no torneamento.

Desbaste_externo – Operação que armazena as informações e representa o comportamento de um processo de desbaste externo de uma peça.

Acabamento_externo – Operação que armazena as informações e representa o comportamento de um processo de acabamento externo de uma peça.

Furacao - Operação que armazena as informações e representa o comportamento de um processo de furação de uma peça em um processo de torneamento.

Funcionarios – Classe que armazena os dados dos funcionários que compõem a empresa onde o sistema atuará.

Materiais – Classe genérica que armazena os dados sobre os materiais utilizados na empresa.

Materia_prima – Classe derivada de materiais que armazena as informações sobre as matérias primas que serão utilizadas no processo de torneamento.

Ferramenta – Classe genérica derivada da classe materiais que armazena os dados sobre as ferramentas do processo de torneamento. São insertos e porta ferramentas.

Insertos – Ferramenta que armazena os dados sobre insertos do processo de usinagem. Os insertos tem uso intensivo nos processos de usinagem.

Porta-Ferramenta – Ferramenta que armazena os dados sobre porta-ferramentas do processo de usinagem realizado com máquinas CNC.

Pedido – Classe que armazena os dados genéricos das transações feitas pelos funcionários solicitando materiais e ferramentas que possibilitem a execução dos processos.

Requisicao – Classe derivada de pedido que armazena as informações sobre os pedidos de materiais do programador ao estoque.

Compra - – Classe derivada de pedido que armazena as informações sobre os pedidos de materiais do estoquista a compras.

Item – Classe que armazena as informações discriminadas dos pedidos.

6.2 Fase2: Construção do Sistema

6.2.1 Projeto

⇒ Elaboração dos Diagramas de seqüência

Os diagramas de seqüência desenvolvidos estão incluídos no Anexo IV - Diagramas de seqüência. Os diagramas de seqüência apresentam os cursos típicos de eventos na interação entre os usuários e os sistemas. São montados a partir dos casos de usos apresentados na fase de levantamento de requisitos. São fundamentais para determinar as principais funções do sistema e a distribuição dos métodos entre as classes.

⇒ Algoritmos:

Os algoritmos apresentam a base lógica das principais operações realizadas pelo sistema. Por serem sucintos são utilizados como ferramenta preliminar à programação. Os principais algoritmos utilizados estão no Anexo V - Algoritmos.

⇒ Estruturas de Dados

Na definição de uma peça utilizam-se perfis com um número inicialmente desconhecido de segmentos. O número e tipo das operações para a fabricação da peça podem variar durante a sua programação. Cada peça necessita de um conjunto diferente de ferramentas. Além disto, durante a programação da peça é interessante que o programador possa navegar pelos dados de perfis e operações para fazer correções, eventualmente incluindo ou eliminando novos segmentos e operações.

Para que seja possível a alocação dinâmica e para se obter uma maior facilidade de acesso aos dados dos perfis e operações estas classes serão implementadas como listas duplamente encadeadas. Estas estruturas permitem uma melhor racionalização do uso da memória dinâmica, e oferece uma grande flexibilidade na alteração, inserção e eliminação de dados.

⇒ Ferramentas para o desenvolvimento

As ferramentas utilizadas neste trabalho para a modelagem e implementação do sistema são:

-**Sistema Delphi da Borland versão 5.0.** Sistema de programação, descrito na seção 4.8, usado para o desenvolvimento das interfaces do editor, codificação do editor e do simulador em Object Pascal, desenvolvimento das interfaces de acesso ao banco de dados de materiais, e geração do programa executável.

-**Sistema de modelagem WithClass 2000-** Ferramenta CASE para a modelagem em alto nível do sistema, edição e geração gráfica dos diagramas, documentação, geração das unidades de codificação para o sistema Delphi, e também através de engenharia reversa, obtenção das classes já desenvolvidas para uso na modelagem. Este sistema utiliza o padrão UML para a confecção dos modelos.

-Sistema Access 2000 do pacote Microsoft Office 2000- para geração de tabelas e desenvolvimento do aplicativo de banco de dados de materiais e respectivas visões do administrador, estoquista e do comprador.

6.2.2 Implementação e teste

⇒ Diagrama lógico do Banco de Dados

O banco de dados foi desenvolvido a partir da modelo do domínio seguindo as regras de mapeamento descritas na seção 4.7. As principais tabelas e seus relacionamento são apresentados pela Figura 6-6. O banco de dados de ferramenta utilizou informações do catálogo da Sandvik (1998).

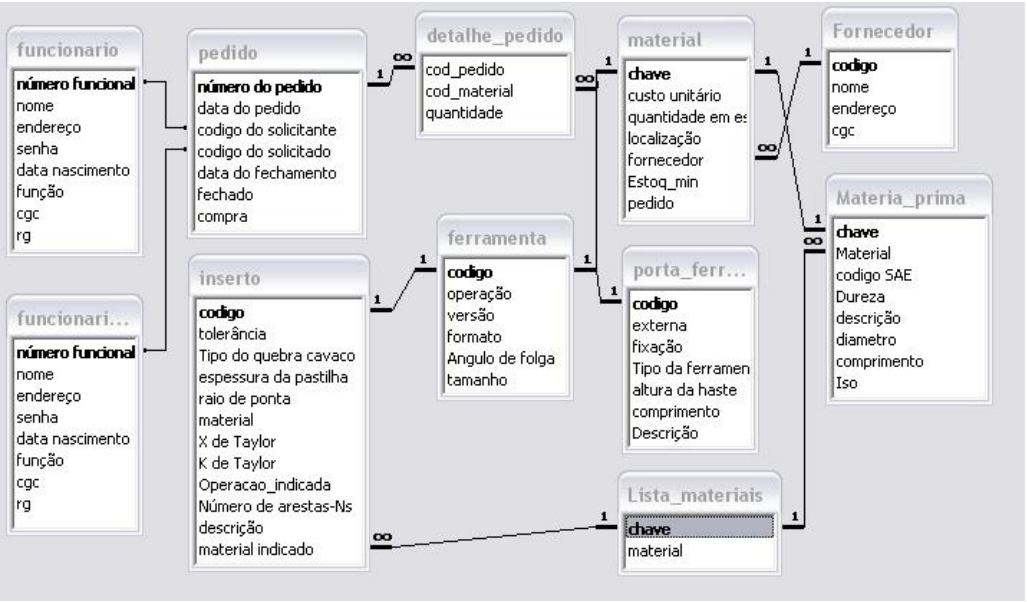

Figura 6-6 -Diagrama com as principais tabelas e seus relacionamentos

⇒ Programação do Pós Processador

Neste trabalho foi desenvolvido um pós-processador específico para o comando numérico "Mach-10". A partir deste, caso outros pós processadores sejam necessários podem ser desenvolvidos.

A principal função do pós-processador é converter a descrição da peça na forma de um programa neutro para o conjunto de comandos necessários à máquina onde a peça será produzida. O programa neutro contém as informações sobre as características geométricas e tecnológicas para a produção da peça e também características do ferramental.

O pós processador, a partir da informações contidas no programa neutro, gera automáticamente os percursos necessários para a execução da peça. O processo de torneamento pode ser representado no plano, o que facilita esta atividade. O eixo Z sempre refere-se ao eixo árvore, o outro eixo é o X, formando assim um sistema de coordenadas cartesianas. Todo o movimento da ponta da ferramenta é descrito neste plano XZ em relação a uma origem pré-estabelecida.

Para que a programação CN possa ser gerada automaticamente é necessária uma base de conhecimento que reproduz o modo de um programador atuar. Ela é também responsável em avaliar a viabilidade da realização da peça rotacional sem a intervenção humana. A base de conhecimento é um conjunto de regras de produção do tipo "se- então- senão ". Esta base, pode ser facilmente incrementada com novas regras para tratar de situações não previstas inicialmente. Através destas regras pode-se avaliar a viabilidade da execução da peça, se o ferramental é apropriado, assim como determinar a necessidade de operações de faceamento e desbastes preparatórios, inversão do posicionamento da peça e decidir pelos melhores ciclos oferecidos pelas máquinas. A partir da descrição geométrica da peça, o pós-processador, de forma analítica percorre o perfil e assim detecta as reentrâncias da peça e seus respectivos ângulos máximos e mínimos. Com estes dados determina a estratégia necessária para o torneamento e as restrições de forma dos insertos, ou mesmo pode determinar a inviabilidade da realização da peça com o ferramental disponível. Este modo de programação utiliza conceitos de sistemas especialistas.

⇒ Diagrama de classes do Sistema Editor CNC

O diagrama de classes apresenta as principais classes implementadas na construção do sistema. Este diagrama é gerado por engenharia reversa, ou seja, ele é gerado pela ferramenta CASE a partir do código do sistema. Por representar o sistema ao final da sua programação, é mais rico em detalhes.

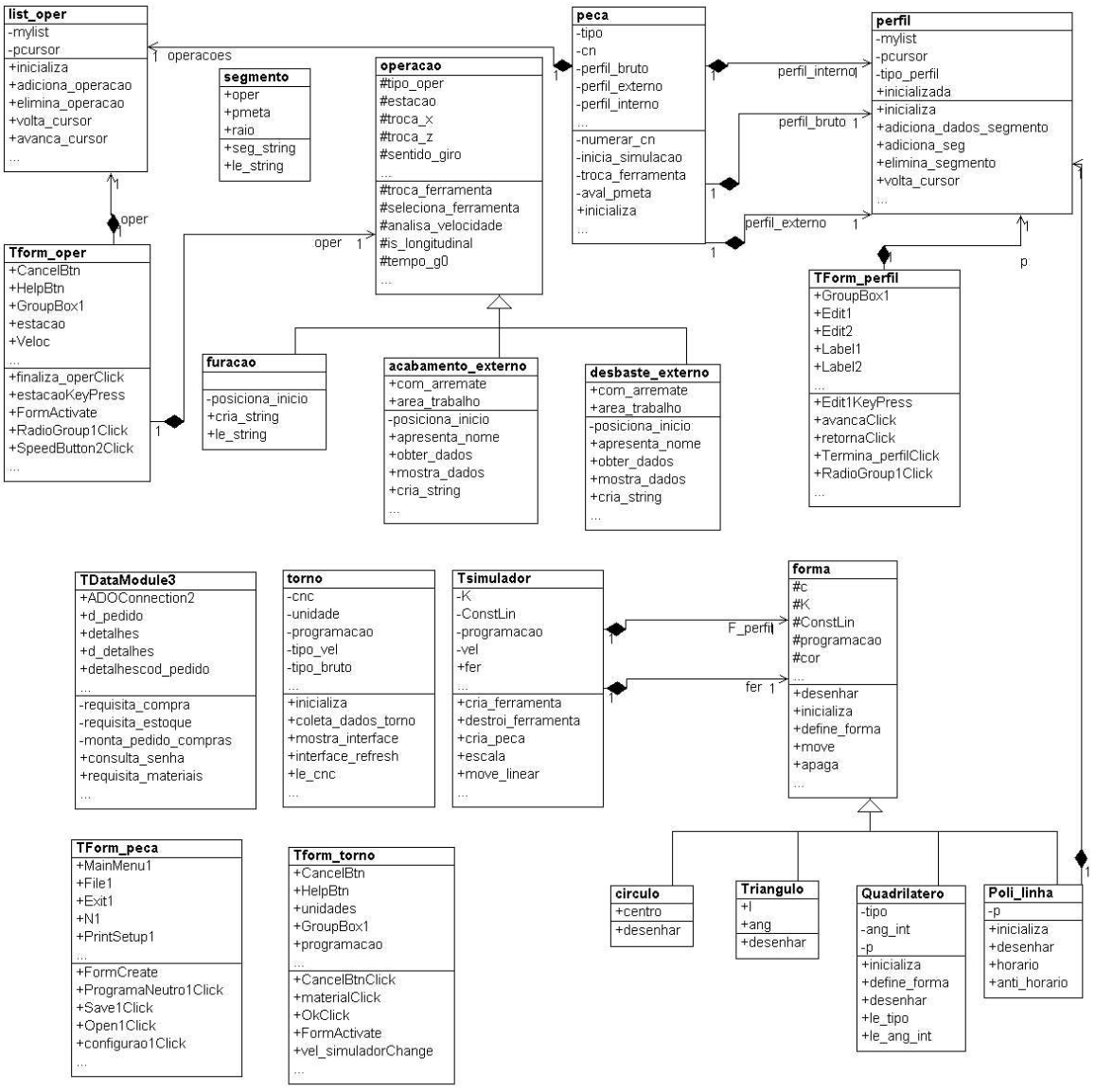

Figura 6-7 - Diagrama de classes com a descrição parcial das classes implementadas

Apresentação do curso típico de enventos na programação de uma Peça

A interação entre o programador e o sistema se inicia pelo formulário principal representado pela Figura 6-8.

.

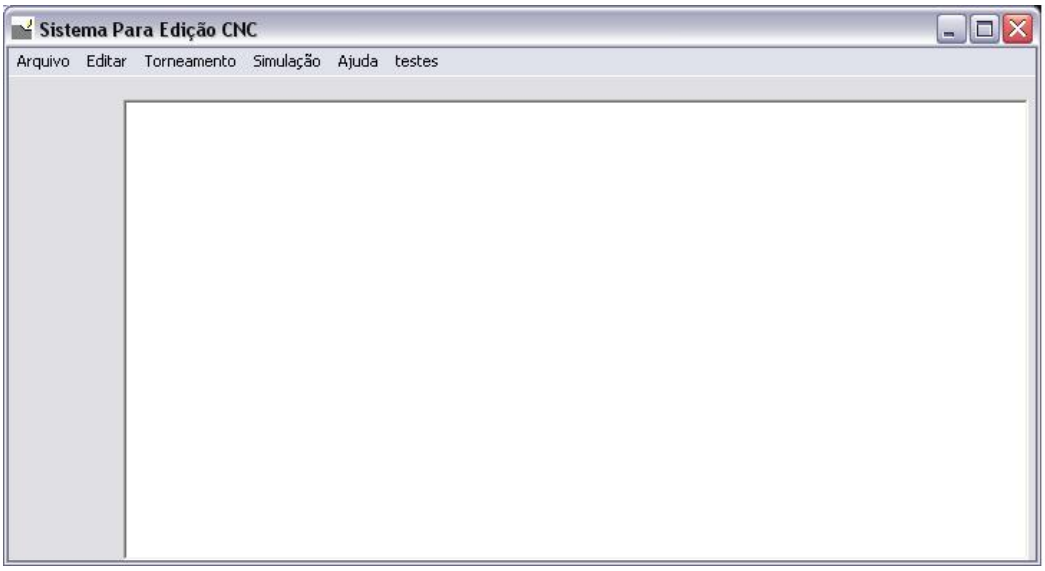

Figura 6-8 Tela principal do sistema para programação de uma peça

Na tela principal, seguindo o curso típico de programação, é feita a seleção da opção: "roteiro de programação" do menu torneamento. A partir deste evento é gerada a janela de diálogo representada pela Figura 6-9. Interagindo com esta tela o programador seleciona qual comando númerico esta trabalhando e determina os parâmetros básicos de programação.

A partir do botão Material da Peça da tela de definição de parâmetros, Figura 6-9, é gerada a janela de diálogo para a seleção da matéria prima da peça, Figura 6-10. Nesta janela o programador seleciona inicialmente o tipo do material e o sistema apresenta todos os materiais daquele tipo. O programador seleciona então entre os materiais disponíveis àquele projetado para a peça, e finaliza com um "ok".

Figura 6-9 Definição dos parâmetros da programação

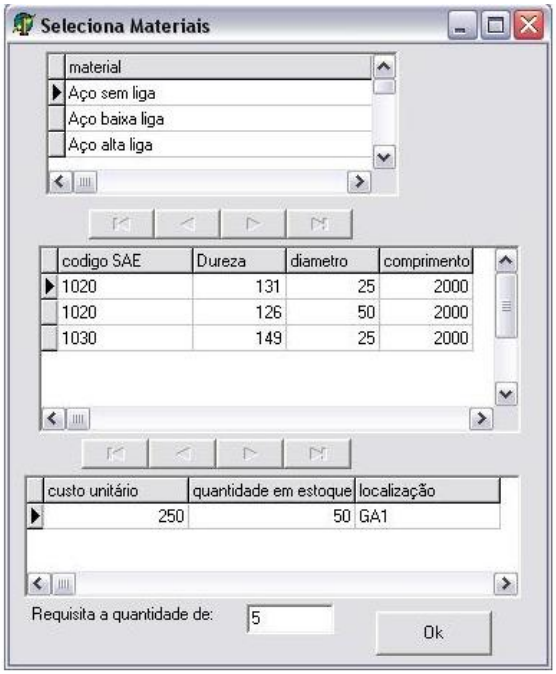

Figura 6-10 Tela para seleção do material da peça

Figura 6-11 Entrada dos dados dos segmentos que compõe o perfil da peça

Continuando, o sistema solicita as dimensões da peça bruta. O programador apresenta as dimensões da peça bruta e confirma. Feito isto é apresentada uma seqüência de telas para que o usuário defina a lista de segmentos que compõem o perfil todo Figura 6-11.

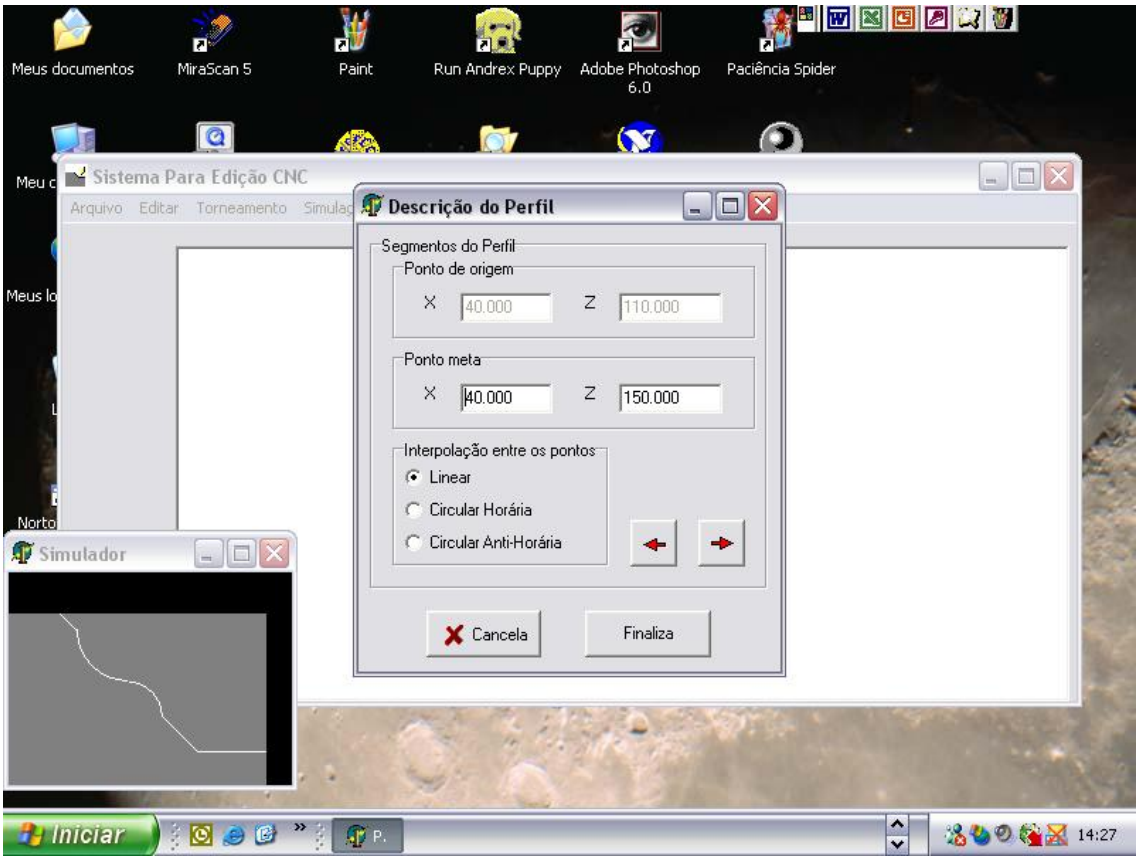

Figura 6-12 Visão geral da definição do perfil e o simulador

A Figura 6-12 apresenta o procedimento de descrição de um perfil em seu final. Tem –se assim uma visão geral do sistema da janela para descrição dos segmentos do perfil e a sua representação através do simulador.

Uma vez determinado as dimensões da peça bruta e os perfis da peça acabada é necessário definir as operações que serão realizadas, isto é feito pela tela de operações, Figura 6-13.

Figura 6-13 Tela para a definição das operações com as ferramentas já selecionadas e parâmetros determinados

Para cada operação é necessária a seleção das ferramentas. Para isto, o programador tem que acionar o botão "seleciona ferramenta" apresentando a janela de diálogo para a seleção de ferramentas, Figura 6-14. O Programador escolhe inicialmente a forma do inserto e o seu tamanho e o sistema faz uma busca no banco de dados e apresenta todos os insertos e porta-ferramentas que se têm as características desejadas. O programador seleciona as ferramentas que melhor se adequam à operação e aciona o botão "seleciona".

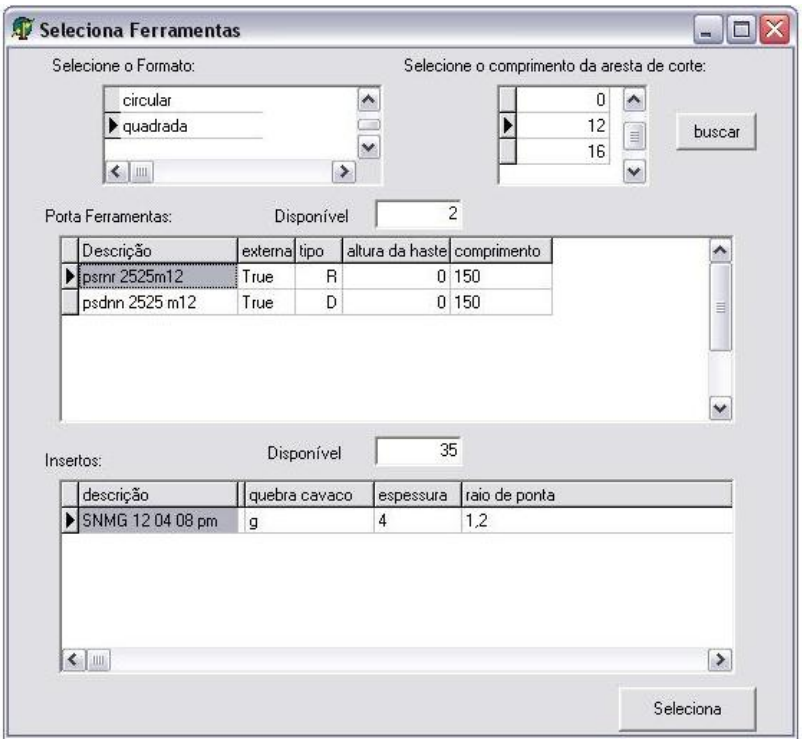

Figura 6-14 Tela para a seleção das ferramentas da operação

O sistema de posse das informações geométricas e tecnológicas da peça e dos dados das ferramentas das operações gera o programa neutro da peça, Figura 6-15.

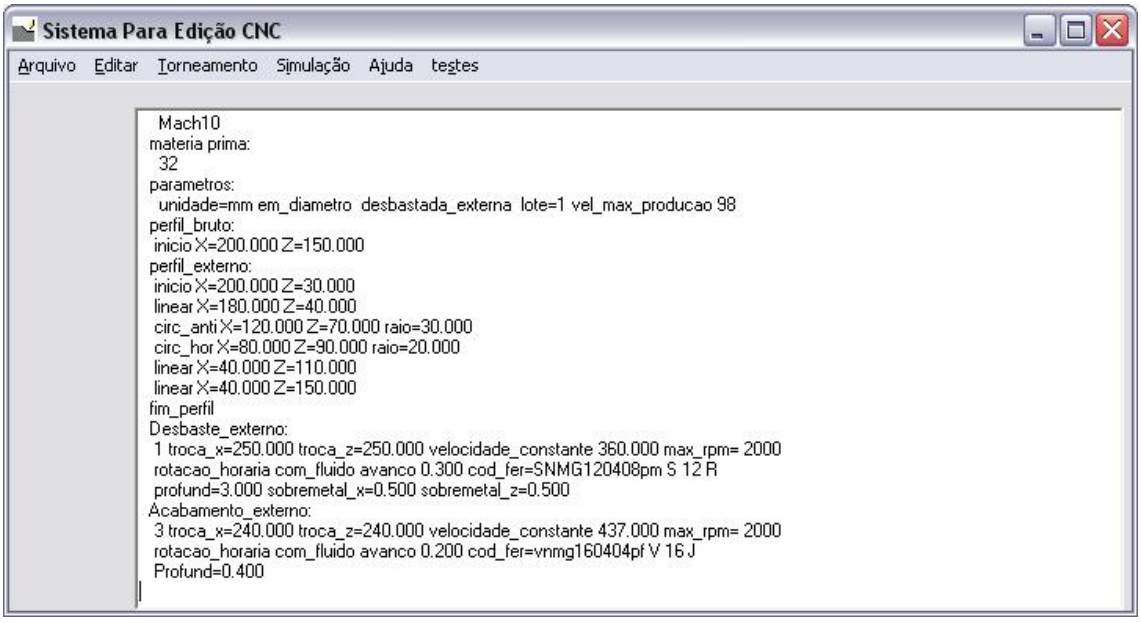

Figura 6-15 Apresentação do programa neutro referente a peça

A partir do programa neutro o sistema pode gerar o programa CN da peça específico para o comando númerico selecionado, Figura 6-16. Uma vez gerado o programa CN pode-se simular o processo de fabricação da peça, Figura 6-17.

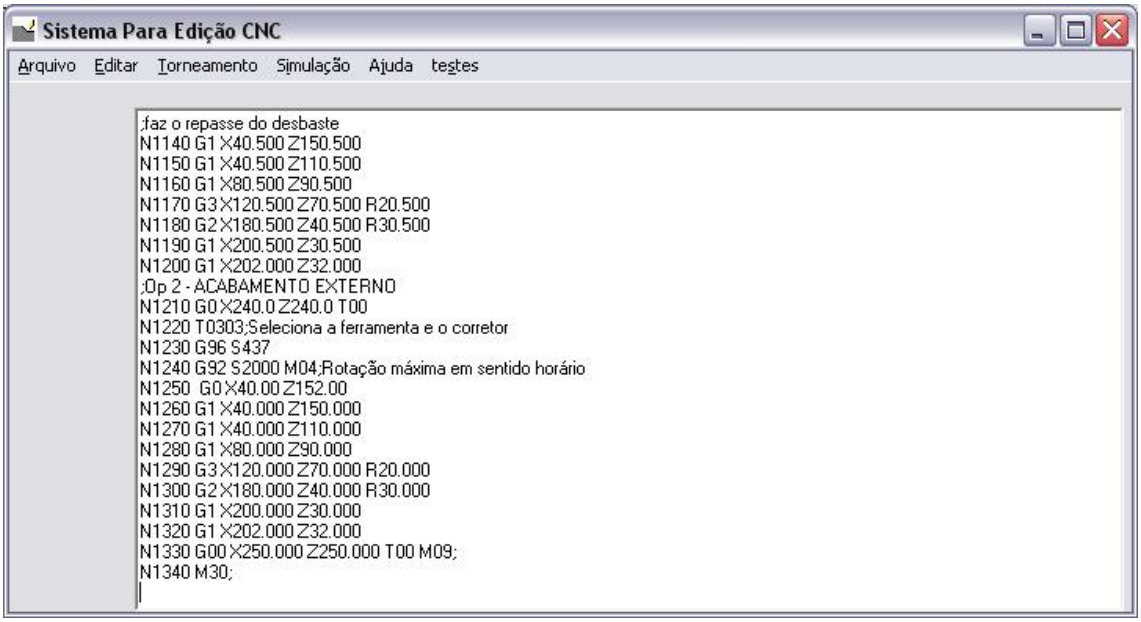

Figura 6-16 Progama CNC da peça

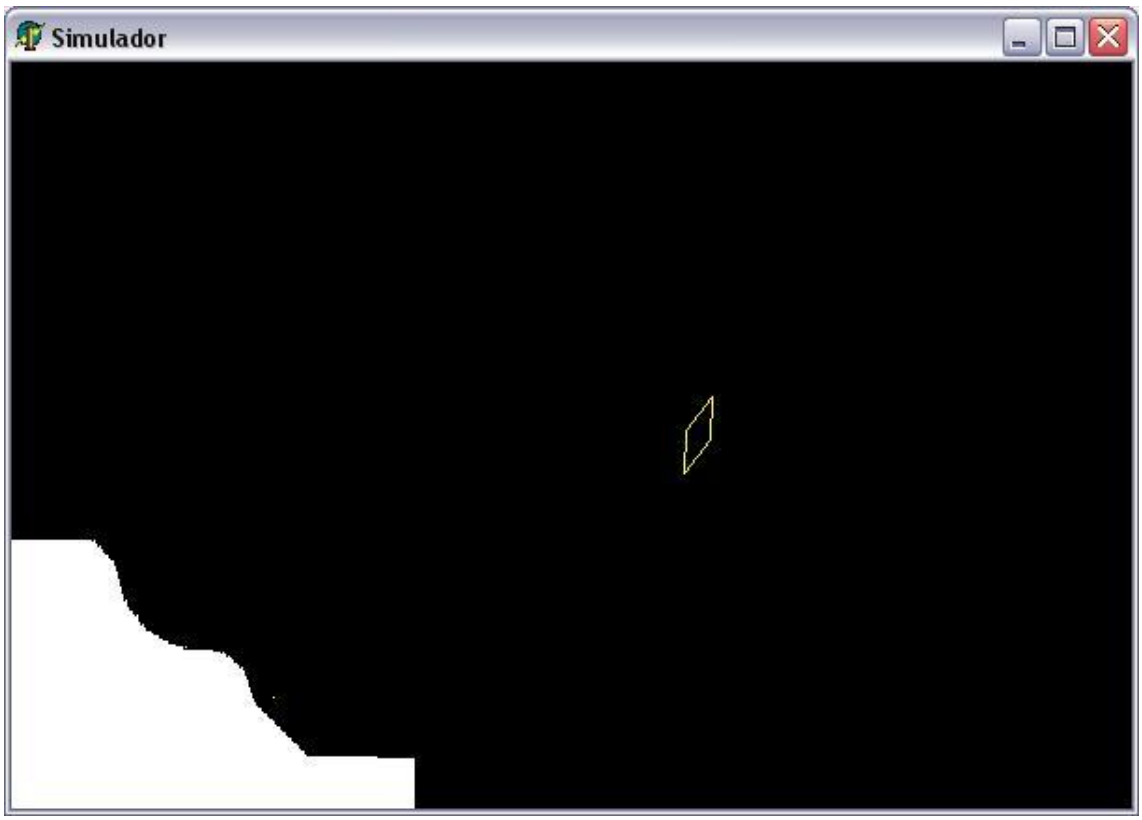

Figura 6-17 Tela apresentando o final da simulação da usinagem

7 Conclusões

Neste capítulo são apresentadas as conclusões finais sobre trabalho desenvolvido.

Este trabalho teve como objetivo estudar os ambientes de manufatura de pequenas e médias empresas que utilizam tecnologia CNC e os sistemas de informação associados a eles. A partir da compreensão deste ambiente, dos sistemas de informação e dos recursos da tecnologia da informação, apresenta uma contribuição ao processo de integração de informações da manufatura através de um sistema de informação que auxilia a produção de peças rotacionais, porém integrado à área de materiais. Desta forma o trabalho desdobra-se em duas tarefas práticas: Propor uma metodologia de desenvolvimento de software orientada a objetos e usando esta metodologia desenvolver um sistema de informação de usinagem integrado com o estoque, e compras apropriado para pequenas e médias empresas.

A metodologia foi projetada buscando atender ambientes de desenvolvimento para pequenas e médias empresas. Portanto, sofre as seguintes restrições: indicada para projetos de pequeno e médio porte, requisitos razoavelmente bem determinados, trabalho em equipes pequenas que podem ser externas ou estarem ligadas as áreas de engenharia ou sistema de informação, porém, que tenham uma interação muito grande com o domínio do problema.

Apoiado na aplicação da metodologia desenvolvida é possível obter as seguintes conclusões:

1) Delimitada pelas restrições apresentadas, a metodologia se mostra robusta por utilizar um conjunto mínimo, porém suficiente de modelos, ferramentas e atividades, implicando que, para se utilizar esta metodologia de desenvolvimento de software não são necessárias ferramentas CASE caras e complexas, o tempo de aprendizado do pessoal de desenvolvimento é minimizado, e a comunicação com o usuário é facilitada.

2) A metodologia é orientada a casos de usos que são validados pelo usuário. Isto resulta em requisitos de sistemas mais precisos e leva a crer que eles têm uma probabilidade maior de estarem corretos, ou seja, que reflitam as reais necessidades dos usuários, de tal forma que os requisitos e seus derivados não terão que ser retrabalhados. A participação ativa do usuário traz uma maior responsabilidade por parte deste no sucesso do projeto. Os casos de uso podem ser utilizados como medida da evolução do desenvolvimento do sistema e atribuição de tarefas.

3) As etapas de Análise e Projeto da metodologia têm poucas atividades de fácil aprendizado e realização, porém, levam a um efetivo planejamento do trabalho de desenvolvimento do sistema. Por meio do gerador de código os cursos típicos de ação dos casos de uso são transformados na estrutura do programa a ser desenvolvido. Isto facilita o projeto do sistema e torna a codificação disciplinada, evitando mudanças do sistema na sua implementação.

Do desenvolvimento do protótipo projetado conclui-se:

1) A utilização de interface baseada em objetos, com serviços bem estabelecidos, para a integração entre o sistema (que faz a programação CN) e o banco de dados de materiais, preservam a integridade dos dados que estão sendo utilizados e propiciam um controle sobre estes acessos.

2) A integração entre o sistema que faz a programação das peças com o estoque e compras torna o processo da produção das peças mais breve porque as solicitações de materiais passam a ser uma rotina integrante da programação e não mais uma “outra” atividade a ser realizada. Os tempos de preenchimento de formulários e transporte de papéis são eliminados. As solicitações de materiais são feitas de acordo com o contexto do programa

CN tornando o processo como um todo mais simples. A integração da produção com o estoque e compras faz com que ao término da programação da peça a solicitações de materiais (ao estoque e compras) estejam prontas, tornando o processo mais completo.

3) O trabalho de programação CN utilizando interfaces gráficas com linguagem técnica-mecânica torna esta atividade mais intuitiva e próxima do ambiente de chão de fábrica.

4) As consultas ao banco de dados de materiais restritas às condições do contexto da programação facilitam muito o trabalho de seleção das ferramentas por parte do programador.

5) Os serviços de geração automática do programa CN, cálculo de condições otimizadas de usinagem, cálculo dos custos e emissão automática do orçamento torna o processo mais rápido, com menor custo, e podem propiciar um melhor atendimento ao cliente.

6) Com o cálculo criterioso das necessidades de material ao se programar a peça, principalmente do número de arestas de corte para um dado lote, evitam-se desperdícios e minimiza-se o estoque necessário destas ferramentas.

7) A integração do sistema de programação de peças com compras torna o processo de suprimento de ferramentas e materiais mais rápido e preciso

8) O sistema desenvolvido pode ser utilizado para fins de ensino.

Bibliografia

Agostinho, O. L. *Integração Estrutural dos Sistemas de Manufatura como pré-requisito de Competitividade.* Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 1995. 146p. Tese (Livre Docência).

Ambler S. W., *The Official Agile Modeling(AM) Site,* Disponível em :< http://www.agilemodeling.com/>, acesso em:20/mai/2003.

Ambler S. W., *Mapping Objects to Relational Databases,* Disponível em :< http://AmbySoft.com/mappingObjects.pdf>, acesso em:20/mai/2003.

Bio S. R., *Sistemas de Informação – Um enfoque empresarial*, São Paulo: Editora Athas, 1991, 183p.

Black, J. T., *O Projeto da Fábrica com Futuro*, Porto Alegre: Editora Bookman, 1991,288p.

Boehs, L., Xavier A. F., De Bortolo M. A., Gonçalves M. A., Uma Abordagem Teórica e Prática sobre Gerenciamento de Ferramentas de Usinagem, II Congresso Nacional de Engenharia Mecânica, 12 a 16 agosto de 2002, João Pessoa, PB. *Anais do II CONEM,* Associação Brasileira de Engenharia Mecânica. 2002.

Booch, G., Rumbaugh J., Jacobson I., *The Unified Modeling Language User Guide,* Reading Massachusetts: Addison-Wesley, 1999,482p.

Buzato, L. E., Rubira C. M. F., *Construção de Sistemas Orientados a Objetos Confiáveis,* Rio de Janeiro: DCC/IM 11 [a] Escola de Computação, 1998, 160p.

Cantù M., *Delphi5 a Bíblia*, São Paulo: Makron Books, 2000, 860p.

Carvalho A. M.R. B., Chiossi T. C. S., *Engenharia de Software*, Campinas: Editora da UNICAMP, 2001, 148p.

Chiavenato I., *Administração de Empresas Uma Abordagem Contingencial*, São Paulo: McGraw-Hill, 1995, capítulo2, pg 41a104.

Coad P., Yourdon E., *Análise Baseada em Objetos*, São Paulo: Editora Campos, 1992, 225p.

Coelho, André Luís Vasconcelos, *Caracterização de um serviço de gerencia distribuído para objetos multimídia persistentes* Campinas: Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas, São Paulo, 1998, capítulo2, pg 11 a 32. Dissertação de mestrado.

Coppini, Nivaldo Lemos; Baptista, Elesandro Antônio, *Machining Process Improvement by Practical Tests in Shop Floor,* International Conference on Advanced Manufacturing Systems and Technologu, Udine. Sixth International Conference on Advanced Manufacturing and technology, University of Udine, 2002. v. 1, p. 121-128.

Cougo P., *Modelagem Conceitual*, Rio de Janeiro: Editora Campus, 1997,284p.

Corrêa, H. L., *ERPs: Por que as implantações são tão caras e raramente dão certo?,* Anais do SIMPOI- FGV, 1998.

Davenport , T. H., *Ecologia da Informação* ,São Paulo: Editora Futura, 1998, 316p.

Degarmo E. P.,Black J.T.,Kosher R.A., *Materials and Processes in Manufacturing*, Nova York: John Wiley & Sons, 1999, 1257p.

Dias C., *Segurança e Auditoria da Tecnologia da Informação*, Rio de Janeiro: Axcel Books do Brasil editora, 2000, 218p.

Diniz, A. E., Marcondes F. C., Coppini N. L., *Tecnologia da Usinagem dos Materiais*, São Paulo: MM Editora, 1999, 242p.

Feng S. C., Song E.Y., A Manufacturing Process Information Model for Design and Process Planning Integration, *Journal of Manufacturing Systems,* v.22:1, p 1-15, 2003.

Ferneda, Amauri Bravo. *Integração CAD e CAM: uma contribuição ao estudo de Engenharia Reversa.* São Carlos: Escola de Engenharia de São Carlos, Universidade de São Paulo, 1999. 104p. Tese (Mestrado).

Ferman, J. H., *ERP to PDM Integration* , Disponível em :< http://www.pdmic/IPDMUG/wp_erp_pdm.html>, acesso em:09/set/2003.

Furlan J. D., *Modelagem de Objetos, através da UML*, São Paulo: Makron Books, 1998, 329p.

Gersting J. L., *Fundamentos Matemáticos para a Ciência da Computação*, Rio de Janeiro: Editora LTC, 1998, 518p.

Graham I. , *Object Oriented Methods*, London:Addison-Wsley Publishing Company, 1994,471p.

Granville W. A., Elementos de Cálculo Diferencial e Integral, Rio de Janeiro: Editora Científica 1996.

Groover M. P., *Fundamentals of Modern Manufacturing – Materials, processes and systems*, New Jersey: Prentice - Hall, 1996, 1061p.

Hull, M., Taylor P., Hanna J., Miller R., Software Development – an assessment, *Information and Software Technology,* v.44, p.1-12, 2002.

Jacobson, I. et. al., *Object Oriented Software Engineering*, New York: ACM Press, Addison-Wesley,1992.

Kang, M., Han J., Moon J. G., An Approach for Interlinking Design and Process Planning, *Journal of Materials Processing Technology,* v.139, p.589-595, 2003.

Kalpakjian S., Schmid S. R. *Manufacturing Engineering and Technology,* Prentice Hall, 2001. Cap 13, p. 255-280.

Kintschner, Fernando Ernesto, *Método de Modelagem de Processos para Apoio ao Desenvolvimento de Software,* Campinas: Faculdade de Engenharia Mecânica , Universidade Estadual de Campinas, 2003. 161p. Tese (Doutorado).

Kobryn, C., UML 2001: A Standardization Odyssey*, Comunication of the ACM*, outubro 1999, vol 42,n. 10, p. 29-37.

Kwasnicka E. L., *Introdução à Administração*,São Paulo: Editora Athas, 1995, 271p.

Laudon K. C., Laudon J. P., *Gerenciamento de Sistemas de Informação*, Rio de Janeiro: Editora LTC, 2001 , 433p.

Lischner R., *Delphi o guia essencial*, Rio de Janeiro: Editora Campus, 2000, 605p.

Machado F.N.R., Abreu M., *Projeto de Banco de Dados*, São Paulo: Editora Érica, 1996, 298p.

Martin J., *Princípios de Análise e Projeto Baseados em Objetos*, São Paulo: Editora Campos, 1994, 486p.

Martin J., Odell J.J., *Object Oriented Analysis and Design,* New Jersey: Prentice –Hall, 1992, 513p.

Mastelari, Niederauer, *Desenvolvimento de um editor/simulador para centros de torneamento CN*. Campinas: Faculdade de Engenharia Mecânica, Universidade de Campinas, 1996. Dissertação (Mestrado).

Mastelari Niederauer; Coppini Nivaldo L., *Developing an integrated Manufacturing System using Object Analisys,*12th International Flexible Automation and Intelligent Manufacturing (FAIM), july 15th –17th, 2002, Dresden University of Technology , Dresden, Alemanha, pg. 1340-1349.

Meireles, Gustavo Suriani de Campos. *Desenvolvimento de sistema de aquisição de dados em operações de usinagem visando o monitoramento de linhas ou células de produção*. São Carlos: Universidade de São Paulo, Escola de Engenharia de São Carlos, 2000. 97p. Tese (Mestrado).

Mendes, M. A. S, *Metodologias Orientadas a Objetos,* disponível em : <www.assespro-mg.org.br/docs/3ocafe.pdf> . Acesso em 14/04/2003.

Mendes, R. M. *Estudo sobre a Passagem de Especificações Orientadas a Objeto para Implementação no SGBDOO O2*. Porto Alegre: Instituto de Informática, Universidade Federal do Rio Grande do Sul, 1996. 116p. Tese (Mestrado).

Mesquita, N. G. M. et Alli. Banco de dados de ferramentas de corte para um sistema CAD/CAPP/CAM. *Máquinas e Metais,* n.442, p.128-139, novembro 2002.

Montez C., *Um Modelo de Programação para Aplicações de Tempo Real em Sistemas Abertos*, Monografia do Exame de Qualificação de Doutorado, Departamento de Automação e Sistemas, Universidade Federal de Santa Catarina, Julho de 1997.

Nassu, E. A. *Banco de Dados Orientados a Objetos e uma Proposta para um Modelo Conceitual*. São Paulo: Instituto de Matemática e Estatística, 1997. 129p. Tese (Mestrado).

Nizoli, Ricardo Miguens *Validação de Requisitos de Sistemas para Geração de use cases.* Porto Alegre: Instituto de Informática, Universidade Federal do Rio Grande do Sul, 2001. 108p. Tese (Mestrado).

Object Management Group, *CORBA FAQ & Resource,*Site Disponível em :< http://www.omg.org/>, acesso em:20/mai/2003.

Pesquisa Nacional de Segurança da Informação, Rio de Janeiro, Modulo Security, 8ª pesquisa anual, setembro de 2002, disponível em: www.modulo.com.br, acesso em 01/setembro/2003.

Popkin Software and Systems *Modeling Systems With UML,* Disponível em :< http://www.popkin.com/customers/customer_service_center/downloads/downloads.htm>, acesso em:20/mai/2003.

Pressman, R. S., Engenharia de Software, São Paulo: Makron Books, 1995, 1056 p.

Rosenberg D.,Scott K., *Applying Use Case Driven Object Modeling with UML: An Annotated e-Commerce Example,* Addison Wesley 1st, 2001, 176p.

Rumbaugh J., Blaha M., Premerlani W., Eddy F. Lorensen W., *Modelagem e Projetos Baseados em Objetos*, Rio de Janeiro: Editora Campus, 1994, 652p.

Ruschel Regina Coeli. *Um modelo de Objetos para Sistemas de Banco de Dados.* Campinas: Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, 1996. 158p. Tese (Doutorado).

Sandvik, *Ferramentas Para torneamento*, Catálogo do fabricante, 1998.

Santos, R. L. A., Raymundo E. A., Ribeiro M. V., Banco de Dados em Usinagem:uma Proposta, II Congresso Nacional de Engenharia Mecânica,12 a 16 agosto de 2002, João Pessoa,PB. *Anais do II CONEM,* Associação Brasileira de Engenharia Mecânica. 2002, p.

K., Rosenberg D., Successful Robustness Analysis . *Software Development,* v. 39, n. 17, p. 3853-3861, março de 2001.

Shaw, Michael J., Informaton-Based Manufacturing with the Web, *The International Journal of Flexible Manufacturing Systems,,* v.12, p.115-129, 2000.

Silberschatz A., Korth H. F., Sudarshan S., *Sistema de Banco de Dados,* São Paulo: Makron Books, 1999, 778 p.

Simon, Alexandre Tadeu. *Condições de Utilização da Tecnologia CNC: Um Estudo para Máquinas-Ferrramenta de Usinagem na Indústria Brasileira*. Campinas: Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2001. 136p. Tese (Mestrado).

Slack, N., Chambers S., Harland C., Harrison A., Johnston R., *Administração da Produção*, São Paulo: Editora Atlas, 726p,1997.

Stair, R.M., *Princípios de Sistemas de Informação*, Rio de Janeiro: Editora LTC, 1998, 451p.

Vassel Clive, Computer Integrated Manufacturing, and Small and Medium Enterprises, *Computers & Industrial Engineering,* v.37, p.425, 428, 1999.

Viana, J. J. *Administração de Materiais,* São Paulo: Editora Atlas, 2000, 448p.

Wells,D., *Extreme Programming: A gentle Introduction,* Disponível em:< http://www.extremeprogramming.org/>, acesso em 10/mai/2003.

Wang, X. K., Research on a disperded networked CAD/CAM system of motorcycle, *Journal of Materials Processing Technology,* v. 129, p. 658-662, 2002.

Zancul, E. S., Guerrero, V, Rosenfeld, H, C.B.M., *Análise Das Abordagens De Integração Entre Sistemas PDM e ERP*, VIII Congresso e Exposição Internacionais da Tecnologia da Mobilidade – SAE BRASIL 99, São Paulo,1999.

Zhang, X.M., Liu F., Lei Q.,Dan B., Integrated DNC: a case study. *International Journal of Production Research,* v. 39, n. 17, p. 3853-3861, maio de 2001.

Anexos

As equações do subtítulo 10.1 foram obtidas a partir dos livros de Diniz, (1999) e Granville (1996), e juntamente com as equações de cálculos de tempo formam a base para os algoritmos que serão utilizados para cálculos de custos e velocidades do sistema.

Anexo I- A equação de Taylor, custos de produção e as velocidades de máxima produção e de mínimo custo:

A equação de Taylor, Equação 1, é obtida experimentalmente a partir do levantamento das curvas de desgaste da ferramenta para diferentes velocidades de corte. Desta forma se obtém uma relação entre o tempo de vida da ferramenta e a velocidade de corte utilizada.

$$T = K.v_c^{-x}$$

Equação 1

Dado que:

T : Tempo de vida da ferramenta em minutos

v_c: velocidade de corte em metros por minuto

Para a otimização dos parâmetros de usinagem utiliza-se o intervalo de máxima eficiência determinado pelas velocidades de mínimo custo (V_{co}) e de máxima produção (V_{cmxp}). A Equação 2 representa a velocidade de máxima produção.

$$V_{cmxp} = \sqrt[x]{\frac{K}{(x-1).t_{ft}}}$$

Equação 2

Dado que:

t_{ft}:Tempo de troca da ferramenta em minutos

Os custos diretamente envolvidos com a produção de uma peça são:

$$K_p = (\frac{t_1}{60} - \frac{1}{Z}).(Sh + Sm) + \frac{t_c}{60}.(Sh + Sm) + \frac{t_c}{T}.(K_{ft} + \frac{t_{ft}}{60}(Sh + Sm))$$

Equação 3

$$t_1 = t_s + t_a + \frac{t_p}{Z} - \frac{t_{ft}}{Z}$$

Equação 4

Dado que:

Kp: custo de produção por peça em reais

Sh: custo operador em reais/hora

Sm: custo da máquina reais/hora

t_c = tempo de corte em minutos.

$t_{1:}$ é o tempo improdutivo, o tempo improdutivo independe da velocidade de corte e sua unidade é o minuto.

t_s : tempo secundário, colocação e fixação da peça, inspeção e retirada da peça em minutos.

t_a : tempo de aproximação e afastamento em minutos.

t_p : tempo de preparo da máquina em minutos.

t_{ft} : tempo de troca da ferramenta em minutos.

Z: lote de peças

Além destas definições temos:

O custo da ferramenta por vida é dado por $K_{ft,}$, Equação 5, pois utilizam-se pastilhas intercambiáveis.

$$K_{ft} = \frac{Vsi}{N_{fp}} + \frac{Kpi}{Ns}$$

Equação 5

Dado que:

Vsi :custo de aquisição do porta ferramenta em R$.

N_{fp} = vida média da porta ferramentas, em quantidade de arestas de corte, até sua possível inutilização.

Kpi = custo de aquisição da pastilha intercambiável.

Ns = número de arestas de corte da pastilha intercambiável.

A Equação 3 pode ser resumida pela Equação 6.

$$K_p = C_1 + \frac{t_c}{60}.C_2 + \frac{t_c}{T}.C_3$$

Equação 6

Dado que:

C_1 = constante independente da velocidade de corte em reais/hora.

C_2 = soma das despesas com mão de obra e com máquina em reais/hora

C_3 = constante de custo relativo à ferramenta.

A velocidade de mínimo custo é dada por:

$$V_{co} = \sqrt[x]{\frac{C_2.K}{60.(x-1).C_3}}$$

Equação 7

Anexo II - Cálculo dos tempos efetivos de corte

A Figura 0-1 representa o processo onde uma ferramenta hipotética faz um trajeto do ponto P até o ponto Q em um movimento de hélice que sai do plano x-z. No plano x-z a ferramenta percorre a curva f(z). O avanço f e a velocidade de corte V_c desta ferramenta são constantes.

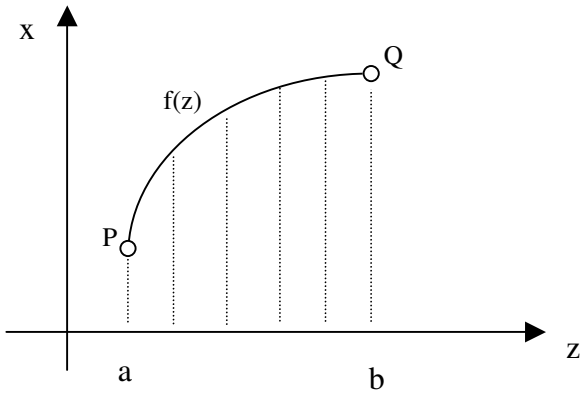

Figura 0-1 Percurso de uma ferramenta

A ferramenta no eixo x-z tem velocidade de avanço $V_{f.}$ dada pela Equação 8.

$$v_f = f.n = \frac{1000.v_c}{\pi.d}.f$$

Equação 8

O tempo efetivo de corte é $t_c=l_f/v_f$ e lf pode ser calculado pela Equação 9.

$$l_f = \int \sqrt{1+f'(z)^2}\,dz$$

Equação 9

Como d é 2*f(z) temos que o tempo t_c pode ser calculado pela Equação 10

$$t_c = K_0 * \int f(z) * \sqrt{(1 + f'(z)^2)} dz \ \textbf{com} \ Ko = \frac{\pi}{v_c . f .500}$$

Equação 10

Dado que:

Vc é a velocidade de corte em m/min

f é o avanço em mm/volta

t_c é o tempo de corte em minutos

⇒ Casos especiais de aplicação da Equação 10:

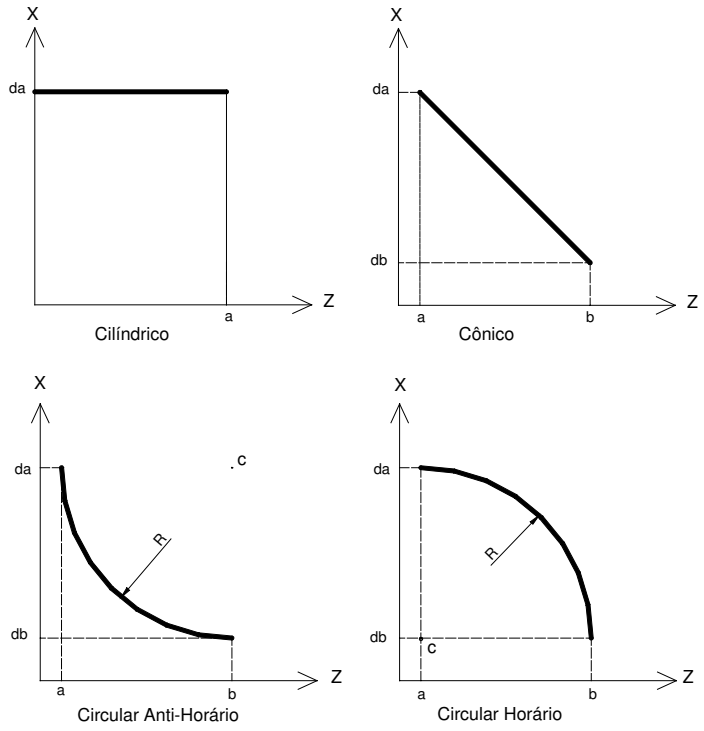

Figura 0-2 Casos especiais de trajetórias de torneamento

Torneamento cilíndrico, de acordo com a Figura 0-2 :

f(z)=da=d/2, onde d é o diâmetro do cilindro.

f'(z)=0

$$t_c = k_0 \int_0^a d_a .dz = \frac{\pi .d.a}{1000.f.V_c}$$

Equação 11

Torneamento cônico, de acordo com a Figura 0-2:

α e β são respectivamente o coeficiente angular e o linear da reta que representa o perfil cônico.

$$f(z) = \alpha * z + \beta \quad \text{onde:} \quad \alpha = \frac{da - db}{a - b} \text{ e } \beta = f(a) - \alpha * a$$

$$f'(z) = \alpha$$

$$C = \sqrt{(1 + \alpha^2)}$$

$$t_c = k_0 \int_a^b (\alpha z + \beta).C.dz = \frac{\pi}{v_c.f.500} . \left[\frac{\alpha C}{2} (b^2 - a^2) + \beta C (b - a) \right]$$

Equação 12

Dados o raio "r" e o ponto centro da circunferência "c" e suas coordenadas "c_z" e "c_x" temos:

Torneamento Circular anti-Horário, de acordo com a Figura 0-2:

$$f(z) = C_x - \sqrt{r^2 - (z - C_z)^2}$$

$$f'(z) = \frac{z - c_z}{\sqrt{r^2 - (z - c_z)^2}}$$

$$t_c = k_0.[\int_a^b c_x.r.\frac{1}{\sqrt{r^2 - (z - c_z)^2}}.dz - \int_a^b r.dz$$

Equação 13

$$t_c = \frac{\pi}{V_c.f.500}\{c_x r[\text{arcsen}(\frac{b - c_z}{r}) - \text{arcsen}(\frac{a - c_z}{r})] - r(b - a)\}$$

Equação 14

Torneamento Circular Horário, de acordo com a Figura 0-2

$$f(z) = C_x + \sqrt{r^2 - (z - C_z)^2}$$

$$f'(z) = \frac{C_z - z}{\sqrt{r^2 - (z - c_z)^2}}$$

$$t_c = \int_a^b c_{z.}.r.\frac{1}{\sqrt{r^2 - (z - c_z)^2}}.dz + \int_a^b r.dz$$

$$t_c = \frac{\pi}{V_c.f.500}\{c_x r[\text{arcsen}(\frac{b - c_z}{r}) - \text{arcsen}(\frac{a - c_z}{r})] + r(b - a)\}$$

Equação 15

Anexo III - Casos de uso

Nos casos de uso a seguir encontramos os seguintes atores:

Ator: O Programador, pertencente à classe Funcionário, é responsável pela programação das peças, é quem faz a descrição da peça, faz pedidos de materiais e ferramentas, pede orçamentos, transfere o programa para a máquina CN, edita os programas.

Ator: O Estoquista, pertencente à classe Funcionário, é o responsável pelo estoque, é quem faz o atendimento dos pedidos de materiais, seleciona e encaminha as peças e materiais para a produção, recebe os materiais, cadastra novos produtos e solicita compras.

Ator: O Comprador, pertencente à classe Funcionário, é o responsável pelos pedidos de compras, processa os pedidos de compras, cadastra novos fornecedores, acompanha e encerra os processos de compra, autoriza o fornecimento de materiais e ferramentas.

Ator: O Administrador, pertencente à classe Funcionário, é o responsável pelo sistema, ele é quem cadastra os usuários do sistema e permite que ele seja utilizado.

Ator: Comando Numérico: é o equipamento que recebe os programas CNC gerados pelo sistema, ou envia programas armazenados em sua memória.

1. Caso de uso: Fazer roteiro de programação

   Descrição:

   1.1. O programador solicita um roteiro de programação.

   1.2. O sistema apresenta um roteiro e através deste roteiro o programador é guiado durante a programação.

   1.3. O programador apresenta os dados necessários para a programação.

   1.4. O sistema apresenta ao final o programa neutro da peça.

2. Caso de uso: Seleção do material da peça

   Descrição:

   2.1. O programador solicita a tela para seleção do material da peça.

   2.2. O sistema apresenta os tipos de materiais disponíveis.

   2.3. O programador seleciona o tipo que deseja.

   2.4. O sistema acessa o banco de dados de materiais e apresenta todos os materiais daquele tipo suas características e quantidades.

   2.5. O programador seleciona um destes materiais para a fabricação da peça e confirma.

   2.6. O sistema fecha a tela.

   Caminho alternativo:

   - caso o programador cancele a operação o sistema retorna ao início.

3. Caso de uso: Definir parâmetros do centro de usinagem, parâmetros da programação da peça e seleciona material da peça.

   Descrição:

   3.1. O programador solicita tela para a configuração dos dados do centro de usinagem.

   3.2. O sistema apresenta uma tela para o programador definir os dados do centro de torneamento (dados gerais de programação e seleção do material da peça).

   3.3. O programador faz a entrada dos dados e aciona a seleção do material da peça.

   3.4. O sistema apresenta uma tela com os materiais disponíveis e suas quantidades.

   3.5. O programador seleciona o material.

   3.6. O sistema retorna à tela de configuração.

   3.7. O programador confirma os dados com um OK.

   3.8. O sistema finaliza a tela.

   Caminho alternativo:

- caso o programador cancele a operação o sistema retorna ao início.

4. Caso de uso: Descrever peça bruta

Descrição:

4.1. O programador solicita tela para descrever a peça bruta.

4.2. O sistema apresenta tela para determinar o tamanho da peça (comprimento e largura).

4.3. O programador faz a entrada dos dados e confirma.

4.4. O sistema finaliza a tela.

Caminho alternativo:

- caso o programador cancele a operação o sistema retorna ao início.

5. Caso de uso: Descrever perfis

Descrição:

5.1. O programador solicita tela para definir os perfis.

5.2. O sistema apresenta uma tela para definir a lista com todos os segmentos que descrevem um perfil da peça.

5.3. O programador faz a entrada de cada segmento e segue para o próximo até finalizar.

5.4. O sistema finaliza a tela.

Caminhos alternativos:

- Caso o programador cancele a operação todas as seleções serão desfeitas e retorna ao início. Esta opção deverá ser confirmada.

- Caso o programador não digite um valor válido uma mensagem é apresentada.

- Caso o programador selecione a opção voltar, são apresentados os dados do segmento anterior da lista e assim poderão ser editados.

- Caso o programador selecione a opção avançar, são apresentados os dados do segmento a frente, caso não exista ele será criado.

6. Caso de uso: Simular graficamente o perfil

Descrição:

6.1. O objeto peça solicita a descrição do perfil ao simulador.

6.2. O simulador interpreta os comandos

7. Caso de uso: Definir operações

Descrição:

7.1. O programador solicita tela para a definição das operações.

7.2. O sistema apresenta a tela para definição operações.

7.3. O programador seleciona a operação.

7.4. O sistema prepara a tela para a respectiva operação.

7.5. O programador aciona a seleção das ferramentas.

7.6. O sistema apresenta a tela para seleção de ferramentas e solicita dados para restringir busca no banco de dados de ferramenta.

7.7. O programador faz as entradas de dados referente às características das ferramentas que deseja.

7.8. O sistema apresenta as ferramentas disponíveis (detalhamento da ferramenta e sua quantidade).

7.9. O programador seleciona as ferramentas que deseja.

7.10. O programador faz a entrada dos dados e confirma.

7.11. O sistema armazena os dados em uma operação e insere a operação em uma lista de operações.

7.12. O programador deve selecionar finalizar para terminar a lista.

7.13. O sistema insere a ultima operação e fecha a tela.

Caminhos alternativos:

- Caso o programador cancele a operação todas as seleções serão desfeitas e retorna ao formulário principal. Esta opção deverá ser confirmada.

- Caso o programador não digite um valor válido uma mensagem é apresentada.

- Caso o programador selecione a opção voltar, são apresentados os dados da operação anterior da lista e assim poderão ser editados.

- Caso o programador selecione a opção avançar, são apresentados os dados da operação à frente, caso não ela não exista será criada.

8. Caso de uso: Gerar programa CN

Descrição:

8.1. O programador determina a partir dos dados apresentados a geração do programa CN.

8.2. O sistema gera o programa CN e o apresenta em um editor de texto.

Caminho alternativo:

- Caso os dados não sejam suficientes para a geração do programa CN é apresentada uma mensagem e retorna a tela principal.

9. Caso de uso: Gerar programa Neutro

Descrição:

9.1. O programador determina a partir dos dados apresentados a geração do programa Neutro.

9.2. O sistema gera o programa Neutro e o apresenta em um editor de texto

Caminho alternativo:

- Caso os dados não sejam suficientes para a geração do programa Neutro é apresentada uma mensagem e retorna à tela principal.

10. Caso de uso: Gerar Orçamento

Descrição:

10.1. O programador determina a partir dos dados apresentados a geração do orçamento de produção.

10.2. O sistema gera o orçamento e o apresenta em um editor de texto

Caminho alternativo:

- Caso os dados não sejam suficientes para a geração do orçamento é apresentada uma mensagem e retorna à tela principal.

11. Caso de uso: Salvar em arquivo

Descrição:

11.1. O programador solicita a gravação dos programas e dados referentes à peça em arquivos no disco rígido.

11.2. O sistema solicita onde deve salvar os arquivos e dados.

11.3. O programador informa o local onde os dados devem ser salvos e confirma.

11.4. O sistema salva arquivos e dados em disco.

Caminho alternativo:

- Caso os dados não sejam suficientes para salvar os programas é apresentada uma mensagem e retorna a tela principal.

12. Caso de uso: Requisição de materiais.

Descrição:

12.1. O programador selecione a opção requisições.

12.2. O sistema apresenta todos os itens requisitados e solicita confirmação.

12.3. O programador confirma

12.4. O sistema solicita o login e senha do programador.

12.5. O programador apresenta os dados.

12.6. O sistema faz a solicitação ao estoque das ferramentas e materiais necessários para a produção e deixa estes itens indisponíveis no banco de dados e fecha a tela de requisições.

Caminho alternativo:

- Caso haja algum problema na transação é apresentada uma mensagem informando as razões do problema e retorna ao inicio.

13. Caso de uso: Ler um programa Neutro.

Descrição:

13.1. O programador solicita a tela para a abertura de um arquivo que descreve uma peça já desenvolvida.

13.2. O sistema apresenta tela para o programador informar onde este arquivo se encontra.

13.3. O programador informa o local e confirma.

13.4. O sistema faz a leitura e atualiza todos os dados referentes à peça, e fecha a tela.

Caminho alternativo: Caso o arquivo não exista uma mensagem é apresentada.

14. Caso de uso: Simular processo de produção da peça.

Descrição:

14.1. O programador aciona a simulação da peça.

14.2. O sistema aciona o simulador para interpretar o programa CN e fazer sua simulação.

Caminho alternativo:

- Caso os dados não sejam suficientes para a simulação ou se algum problema ocorrer uma mensagem é apresentada e retorna a tela principal.

15. Caso de uso: Comunicação com o CNC

Descrição:

15.1. O programador solicita tela para comunicação com o comando numérico.

15.2. O sistema apresenta a tela ao programador solicitando operação.

15.3. O programador apresenta os dados e o local onde os dados estão para serem enviados ou onde serão recebidos e confirma.

15.4. O sistema aciona o CNC para estabelecer a comunicação e conclui a operação informando se a comunicação foi bem sucedida ou não.

Casos de uso do sistema de banco de dados:

16. Caso de uso: Cadastrar um item de estoque.

Descrição:

16.1. O estoquista solicita tela para cadastro de um novo item de estoque.

16.2. O sistema apresenta a tela de cadastro.

16.3. O estoquista faz a entrada dos dados referentes àquele item e confirma.

16.4. O sistema do banco de dados faz o cadastro nas tabelas referentes aos materiais.

Caminho alternativo:

- Caso os dados não sejam suficientes ou ocorra algum problema o sistema apresenta uma mensagem informando.

17. Caso de uso:recebe um item de estoque

Descrição:

17.1. O Estoquista recebe um item e solicita tela para atualizar os dados referentes a este material

17.2. O sistema do banco de dados apresenta a tela de alteração e solicita os dados.

17.3. O estoquista faz a alteração da quantidade de um novo item e confirma

17.4. O sistema do banco de dados faz a alteração e apresenta o local para o armazenamento.

Caminho alternativo:

- Caso os dados não sejam suficientes ou ocorra algum problema o sistema do banco de dados apresenta uma mensagem informando.

18. Caso de uso: Processar Pedido.

Descrição:

18.1. O Estoquista solicita todos os pedidos pendentes.

18.2. O sistema apresenta todos os pedidos pendentes.

18.3. O estoquista seleciona o pedido que irá processar.

18.4. O sistema apresenta os detalhes do pedido.

18.5. O estoquista avalia os dados do pedido e caso seja autorizado inicia seu processamento. Para cada item avalia: a quantidade disponível, a quantidade solicitada, sua localização. Faz a separação dos materiais e seu envio. Solicita tela para baixa do material.

18.6. O sistema apresenta tela para a retirada do material e solicita os dados, o estoquista apresenta os dados e confirma.

18.7. O sistema do banco de dados faz a alteração.

Caminho alternativo:

- Caso os dados não sejam suficientes ou ocorra algum problema o sistema do banco de dados apresenta uma mensagem informando.

19. Caso de uso: Cadastrar um novo fornecedor.

Descrição:

19.1. O Comprador a partir solicita a tela de fornecedores para fazer a inclusão de um novo fornecedor.

19.2. O sistema do banco de dados apresenta a tela e solicita os dados.

19.3. O comprador faz a entrada dos dados e confirma.

19.4. O sistema do banco de dados faz o cadastro.

Caminho alternativo:

- Caso os dados não sejam suficientes ou ocorra algum problema o sistema do banco de dados apresenta uma mensagem informando.

20. Caso de uso: Solicitar compra.

Descrição:

20.1. O estoquista ou o sistema editor verifica se um item de estoque esta abaixo do nível de segurança.

20.2. O sistema retorna informação.

20.3. O estoquista ou o sistema editor solicita a abertura de um pedido de compra. O sistema monta um pedido de compra e solicita os dados para o pedido.

20.4. O estoquista ou o sistema editor insere os dados referentes ao item que deve ser adquirido e confirma.

20.5. O sistema do banco de dados registra o pedido de compra.

Caminho alternativo:

- Caso os dados não sejam suficientes ou ocorra algum problema o sistema apresenta uma mensagem informando.

21. Caso de uso: Fazer cadastro de funcionários.

21.1. O administrador solicita tela para o cadastro dos funcionários.

21.2. O sistema apresenta tela para o cadastro de funcionários.

21.3. O administrador faz a entrada dos dados e solicita ao funcionário que digite o nome de acesso e a senha pelos quais será identificado.

21.4. O sistema do banco de dados faz o cadastro do funcionário.

Caminho alternativo:

- Caso os dados não sejam suficientes ou ocorra algum problema o sistema do banco de dados apresenta uma mensagem informando

22. Caso de uso: Processar pedido de compra

22.1. O comprador solicita os pedidos de compra pendentes.

22.2. O sistema apresenta os pedidos pendentes.

22.3. O comprador seleciona o pedido de compra que irá processar.

22.4. O sistema apresenta os detalhes do pedido.

22.5. O comprador a partir dos dados do pedido, e com a devida autorização, monta a autorização do fornecimento. Insere os dados referentes aos itens que devem ser adquiridos e confirma.

22.6. O sistema do banco gera a solicitação de fornecimento.

Caminho alternativo:

- Caso os dados não sejam suficientes ou ocorra algum problema o sistema apresenta uma mensagem informando.

Anexo IV - Diagramas de seqüência

Os diagramas a seguir representam, de forma detalhada, os principais cursos de eventos na interação entre usuários e o sistema. Eles foram montados a partir dos casos de usos apresentados na fase de levantamento de requisitos. Estes diagramas mostram a distribuição das operações entre as classes.

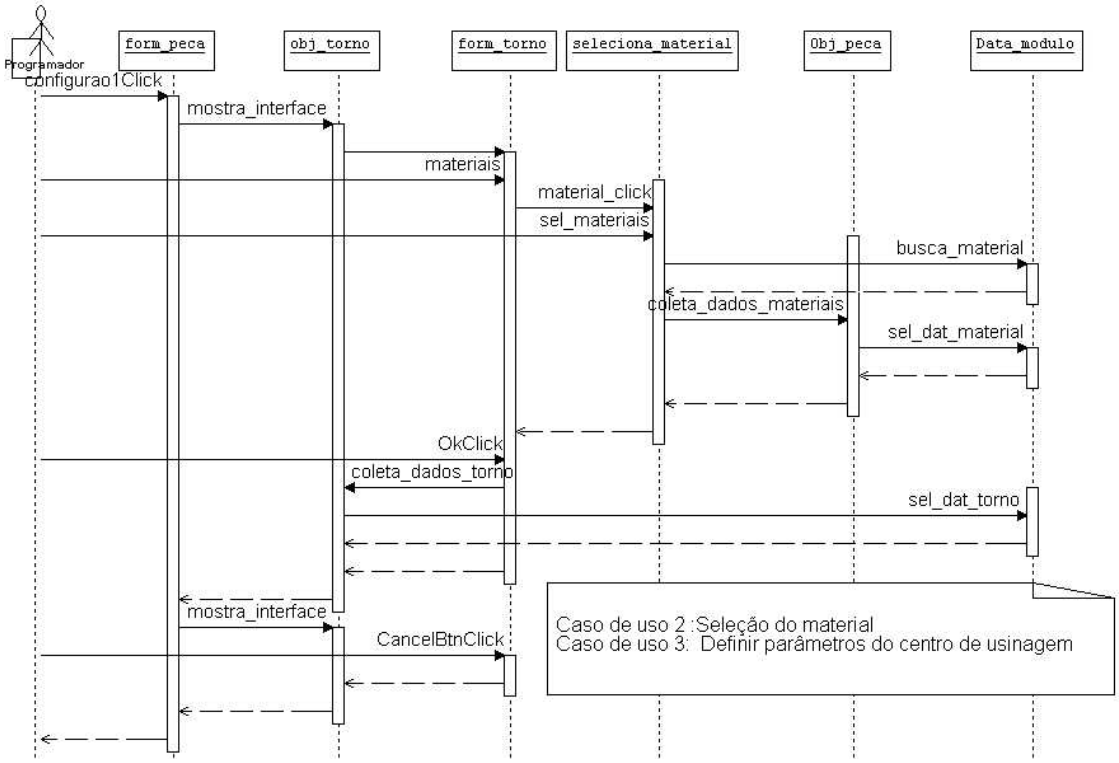

Figura 0-3 Diagrama de Seqüência referente aos casos de uso: Seleção do material e Definir Parâmetros do Centro de Usinagem

Figura 0-3 apresenta a interação entre o programador e as classes do sistema referentes à seleção do material e os parâmetros da programação e do centro de usinagem

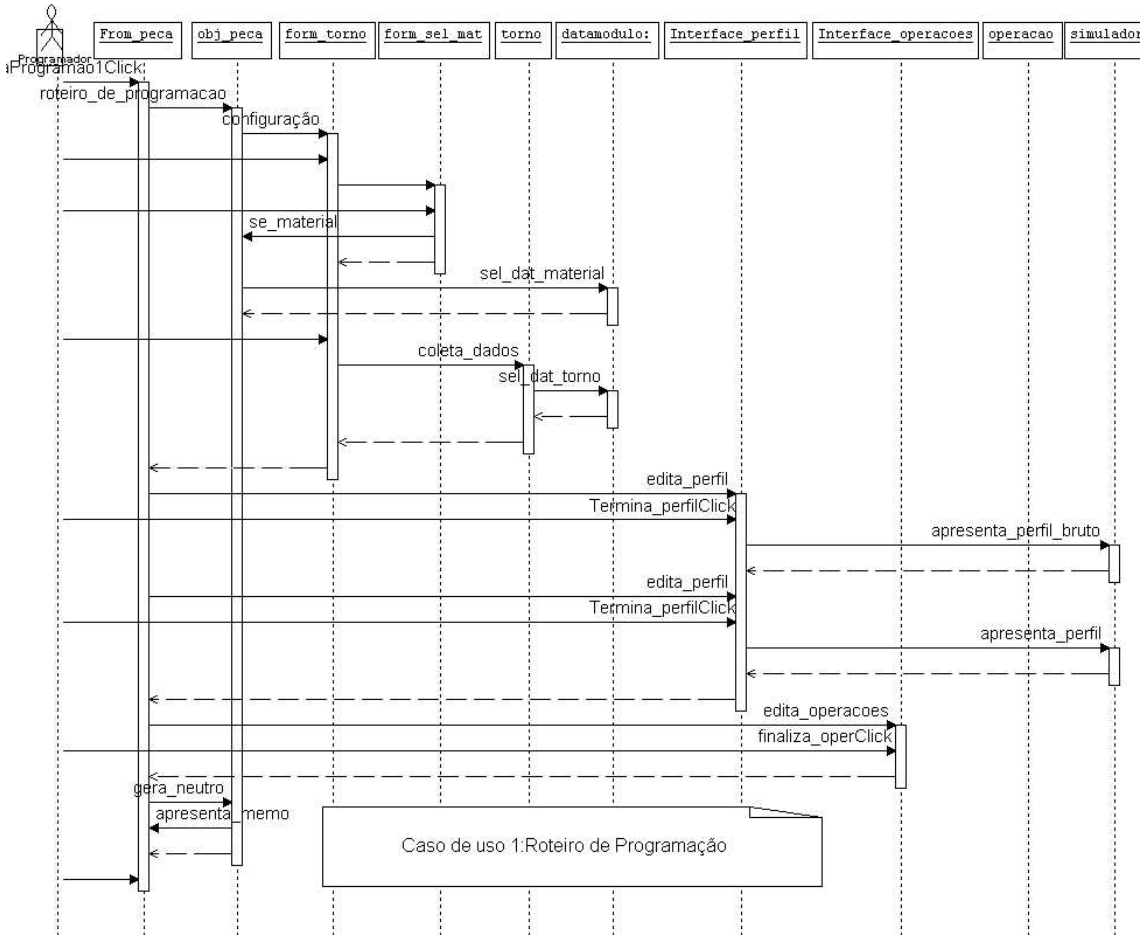

Figura 0-4 Diagrama de sequência referente ao caso de uso: Roteiro de Programação

A Figura 0-4 apresenta a interação entre o programador e as classes do sistema que conduzem o programador à produção de uma peça.

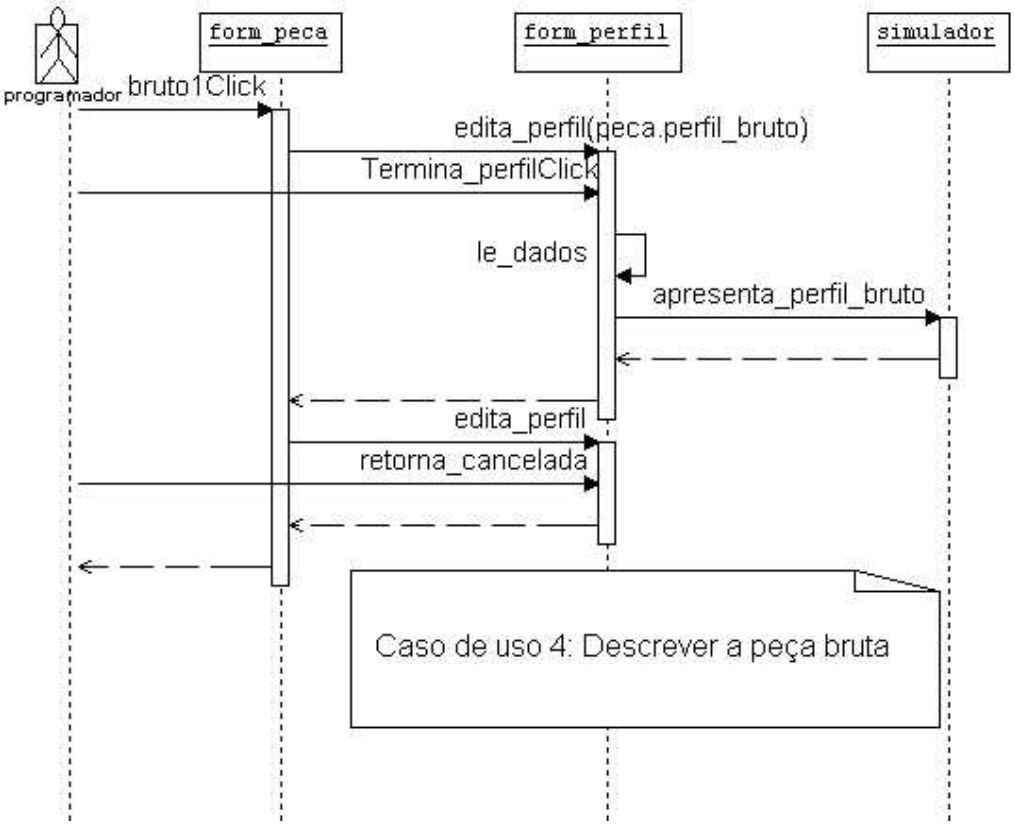

Figura 0-5 Diagrama de sequencia referente ao caso de uso: Descrever a Peça Bruta

O diagrama da Figura 0-5 apresenta a seqüência de interações para a descrição da peça bruta.

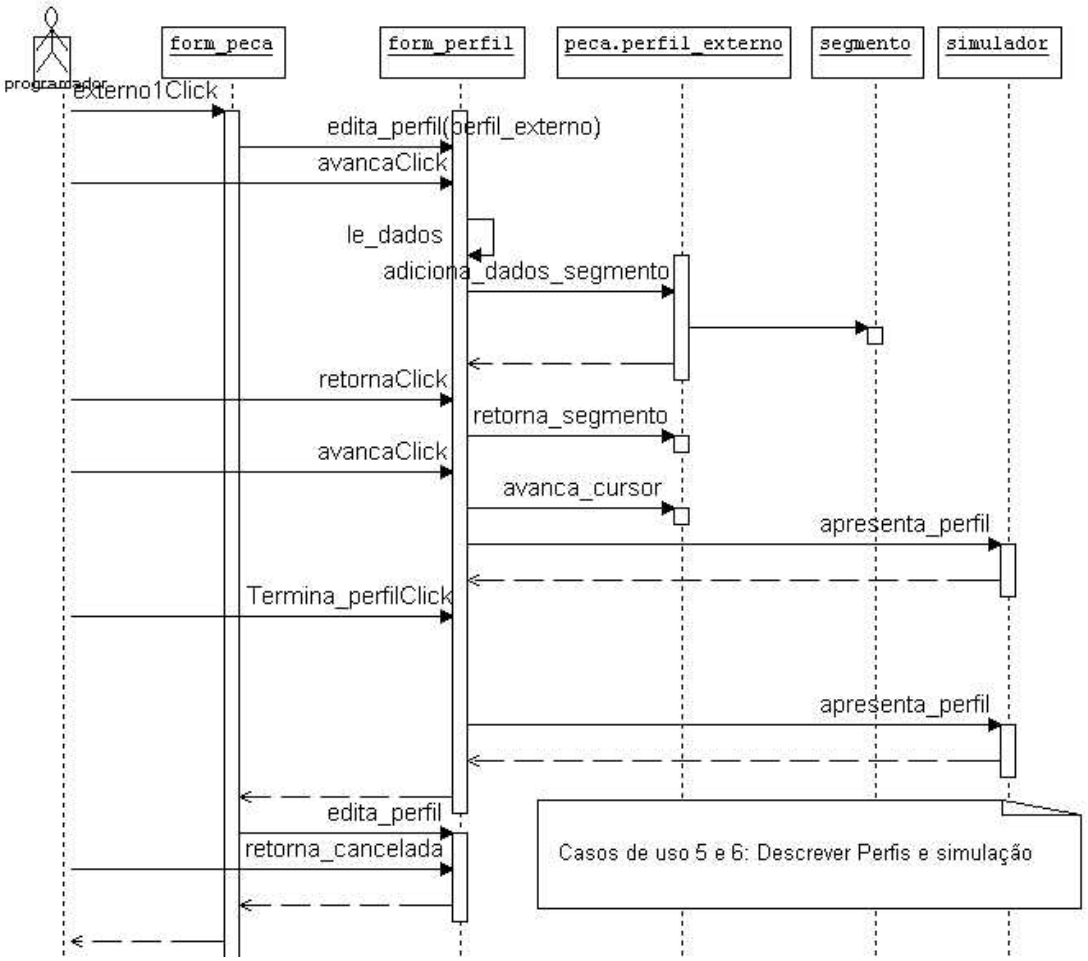

Figura 0-6 Diagrama de seqüência dos casos: Descrever Perfis e Simulação

O diagrama de Figura 0-6 apresenta as interações para a descrição de um perfil e a apresentação gráfica deste perfil pelo simulador.

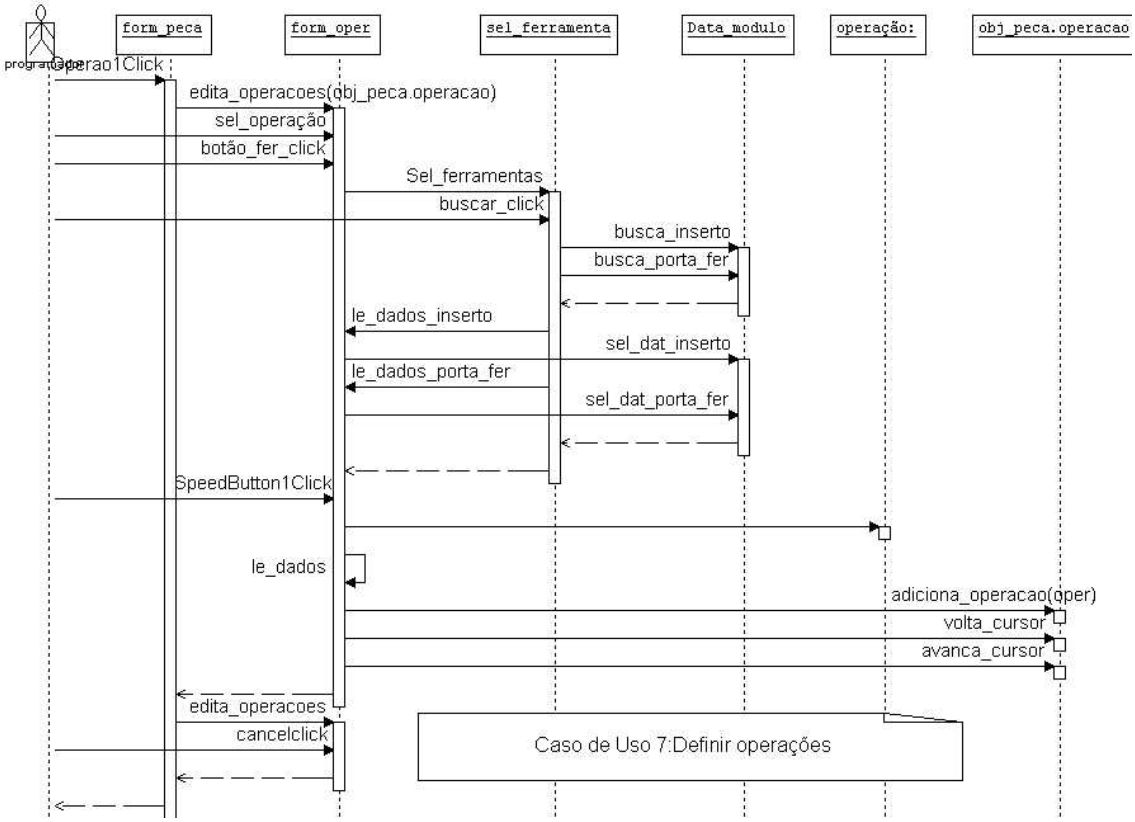

Figura 0-7 Diagrama de seqüência do caso de uso definir operações

O diagrama de seqüência representado pela Figura 0-7 mostra a definição de uma operação.

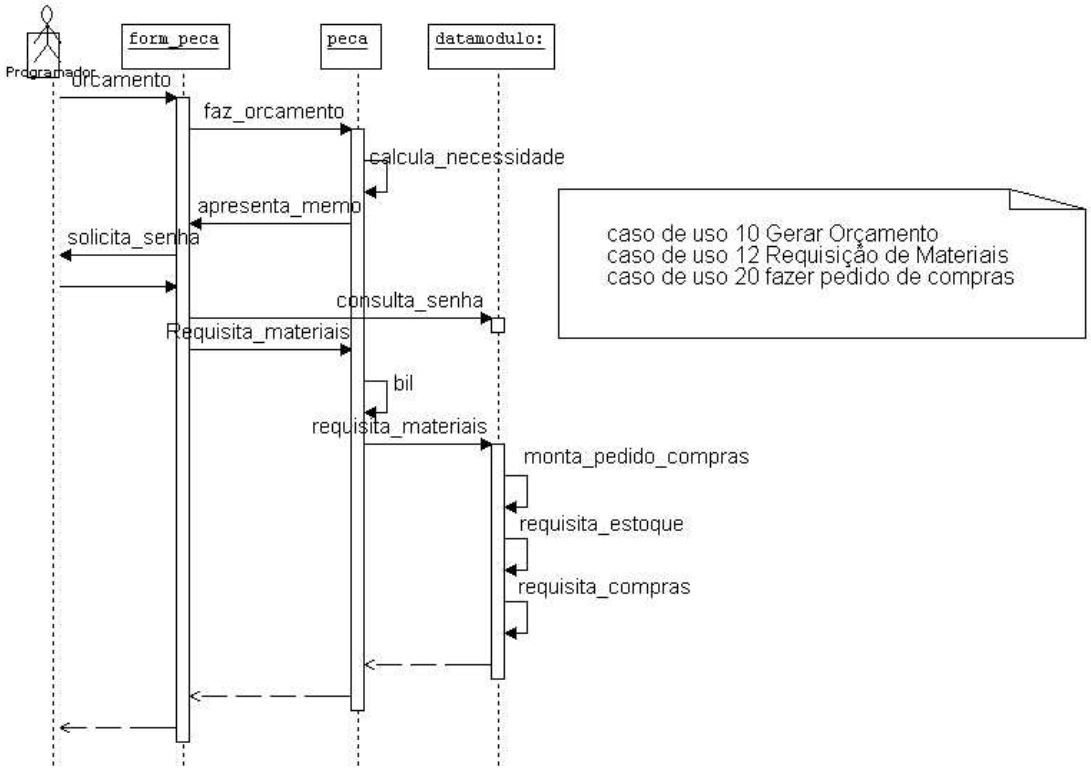

Figura 0-8 Diagrama de seqüência dos casos de uso: Gerar Orçamento e Pedidos

O diagrama apresentado na Figura 0-8 mostra as interações para a geração do orçamento e a solicitação dos pedidos ao estoque e, se o item estiver abaixo do nível de segurança, solicita também a compras.

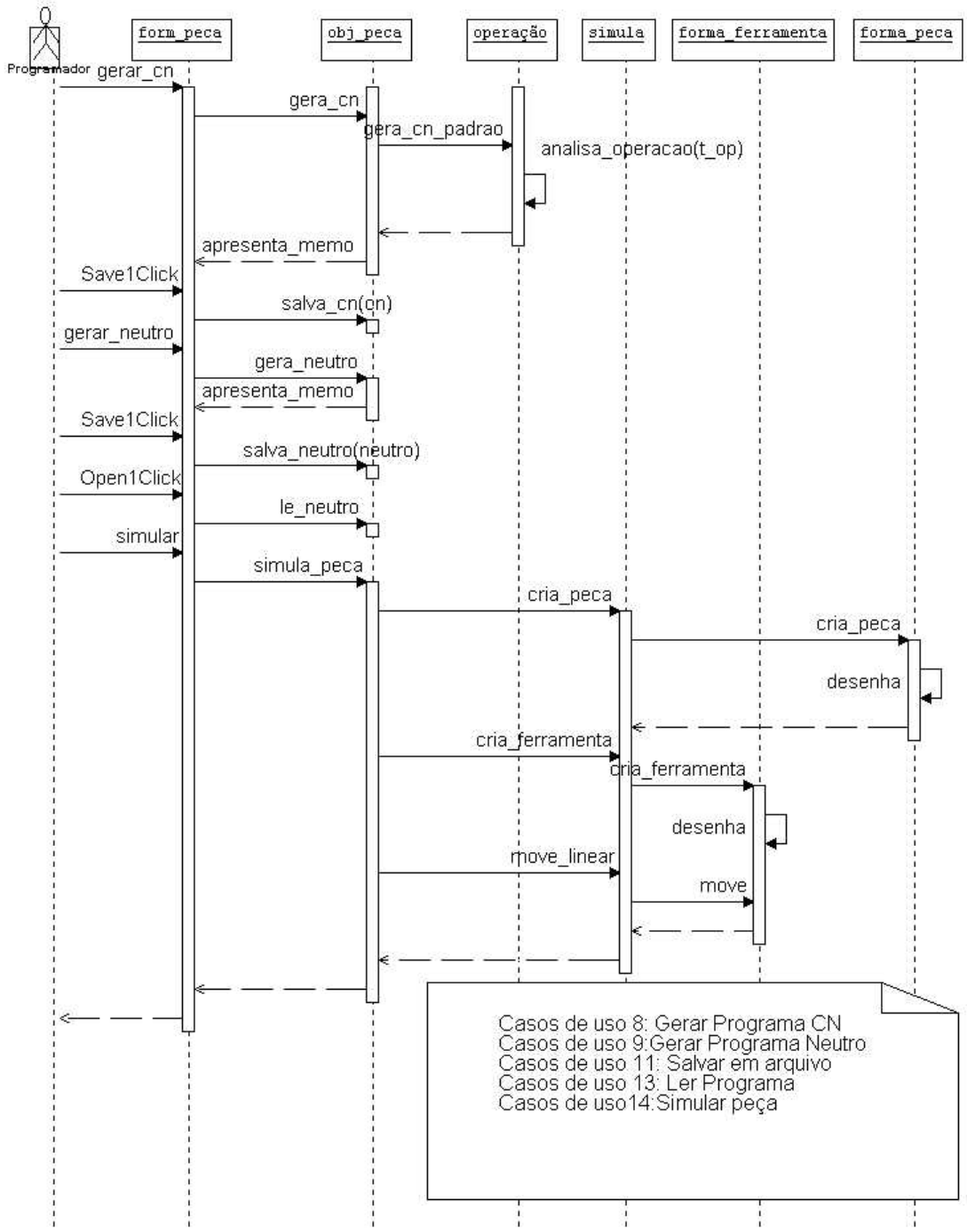

Figura 0-9 Diagrama de seqüência dos casos de uso das operações utilitárias e simulação

O digrama de seqüência da Figura 0-9 apresenta as operações utilitárias do sistema.

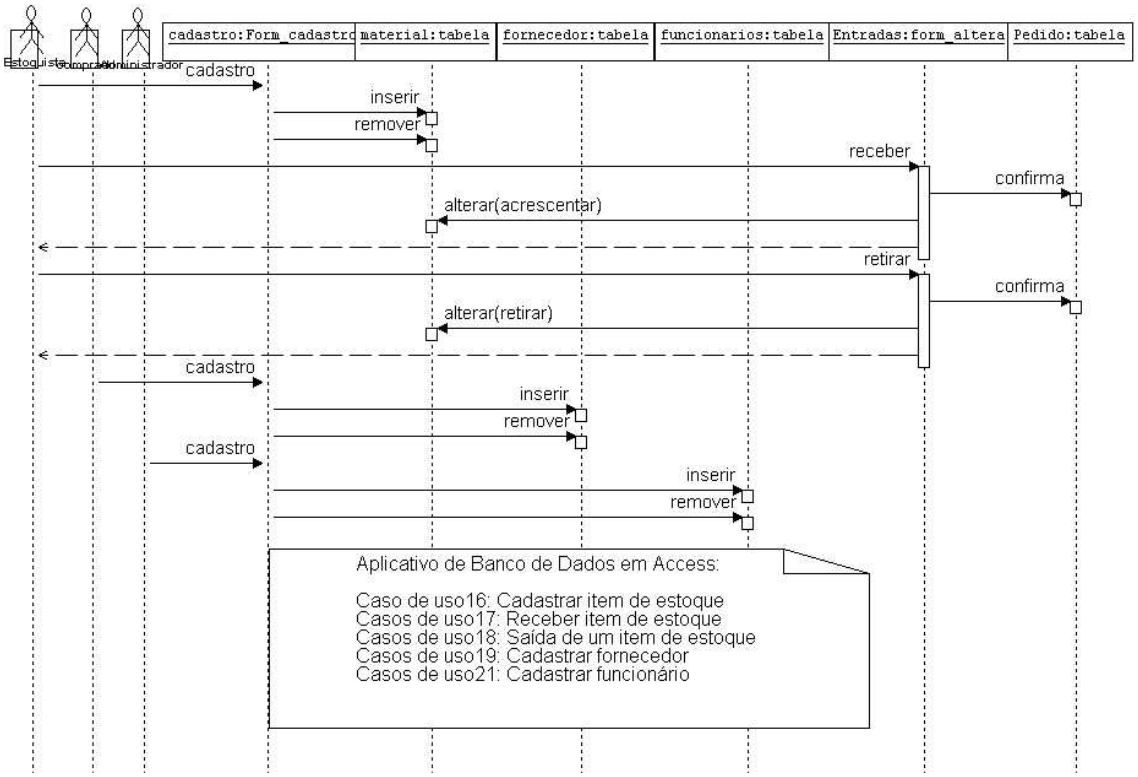

Figura 0-10 Diagramas de seqüência dos casos de uso do Aplicativo de Banco de Dados

O diagrama representado pela Figura 0-10 apresenta as interações referentes aos servidos providos pelo aplicativo de bando de dados.

Anexo V - Algoritmos

O Algoritmo requisita apresenta o processo para solicitação de informações e cálculos para produção de uma peça.

```
Algoritmo programa peça
        Ler dados do torno
 Apresentar matéria_prima disponível
  Ler dados matéria_prima
  Ler dados perfil
  Enquanto necessitar operação
   Ler dados operação
                Apresentar porta-ferramentas disponíveis da operação
   Ler dados porta-ferramentas
   Apresentar insertos disponíveis da operação
   Ler dados inserto
   Se vel=máxima produção
    Então Vc←vel_max_produção
    Senão se vel=mínimo custo
      Então Vc← vel_mínimo_custo
      Senão ler Vc
  Inserir operação na lista de operações da peca
  Gerar programa neutro da peça
  Apresenta programa neutro da peça
  para cada operação na lista de operações
       tc←0
       para cada percurso da operação
    tc←tc+ t_percurso(percurso)
  T←Tempo_de_vida(Vc)
  Calcular_número_de_arestas(tc,T)
  Insere custo_operação(tc,T) na lista de custos da peca
  Gerar programa CNC da peça
      Apresentar programa CNC da peça
  Apresentar orçamento da peça
  Ler confirmação
  Se confirmação = OK
   Então
    Apresentar materiais solicitados
    Concluir transação da requisição
Fim_algoritmo.
```

A função Vel_max_produção calcula a velocidade de máxima produção Vcmxp

```
função Vel_max_produção
   Leia K,x,Tft
   Vel_max_produção ← exp(1/x*ln(k/((x-1)*Tft)))
Fim_função
```

A função Vel_mínimo_custo calcula a velocidade de mínimo custo Vco

```
função Vel_mínimo_custo
   Leia K,x,Tft,Sh,Sm,Vsi,Nfp,Ns,Kpi
   Kft ← Vsi/Nfp + Kpi/Ns
   C2 ← Sh+Sm
   C3 ← Kft+Tft/60*C2
   Vel_min_custo ← exp(1/x*ln((C2*k)/(60*(x-1)*C3)))
Fim_função
```

A Função T_percurso calcula o tempo efetivo de corte da operação para cada um de seus percursos. Um percurso é um movimento da ferramenta entre dois pontos seguindo uma trajetória linear ou circular.

```
Função T_percurso(percurso)
   Leia f,Vc
   Se percurso= cilíndrico
     Então
       Leia da,a
      tpercurso←0.12*pi*da*a/(f*Vc)
     Senão
      se percurso= cônico
          então
     leia da,db,a,b
     alfa←(da-db)/(a-b)
     beta←da-alfa*a
     C←sqrt(1+sqr(alfa))
            tpercurso←(0.12*pi)/(Vc*f)*(alfa*C/2*(sqr(b)-sqr(a)+beta*C*(b-a))     senão
     leia a,b,r,Cx,Cz
           aux← Cx*r(arcsin((b-Cz)/r)-arcsin((a-Cz)/r)) – r(b-a)
     se percurso circular anti-horario
       então
    tpercurso←0.12*pi/(Vc*f)*(aux-r(b-a))
      senão
    tpercurso←0.12*pi/(Vc*f)*(aux+r(b-a))
fim_função
```

A função Tempo_de_vida calcula o tempo de vida da ferramenta dada velocidade de corte Vc

```
Função Tempo_de_vida(Vc)
   Leia x,K
   T←K/exp(x*ln(Vc))
Fim_função
```

A função calcula_número_de_arestas calcula o número necessário de arestas para executar a respectiva operação dados o tempo efetivo de corte e o tempo de vida da ferramenta.

```
função calcula_número_de_arestas(tc,T)
   leia Z
   calcula_numero_de_arestas←Z*tc/T
fim_função
```

A função calcula_custo_da_operação calcula o custo de uma operação passados por parâmetro os valores do tempo de corte, tc, e o tempo de vida da ferramenta, T.

```
Função custo_da_operação(tc,T)
   Leia Tft,Sh,Sm,Vsi,Nfp,Ns,Kpi,Z,ts,ta,tp
   Kft ← Vsi/Nfp + Kpi/Ns
   t1← ts+ta+tp/Z-tft/Z
   aux←sh+sm
   Custo_da_operação←(t1/60-1/Z)*aux+tc/60*aux+tc/T*(Kft+tft*aux)
Fim_função
```

A função compra calcula a quantidade de um certo item que deve ser adquirida do fornecedor, passado por parâmetro o valor que esta sendo requisitado do estoque. Esta função apresenta a lógica básica da solicitação automática de compras. O valor resultante da execução da função é a quantidade do item que deverá ser comprada do fornecedor.

As variáveis são:

- requisição: é a quantidade pedida pela produção ao estoque de um dado item.

- estoque: é a quantidade do item em estoque no momento da solicitação.

- estoque_segurança: é o valor de referência que desencadeia providências para ativação de encomendas.

- Pedidos_programados: Pedidos de compra enviados ao fornecedor e cujos itens estão sendo aguardados. O pedido, ao ser recebido, faz com que a respectiva quantidade do item seja transferida de pedidos_programados para o estoque.

```
Função compra(requisição)
        Ler estoque,estoque_segurança, pedidos_programados
        aux←estoque+ pedidos_programados-requisição
        se aux<estoque_segurança
         então    compra← estoque_segurança-aux
                  senão   compra←0
fim_função
```

Esta sub-rotina apresenta o orçamento com o custo unitário de uma peça e o custo total do lote de peças.

```
Subrotina  apresenta_orçamento
   Leia custo_matéria_prima, Z
        Custo_peça←custo_matéria_prima
   Para cada operação
    Custo_peça←custo_peça+custo_da_operação(tc,T)
   Custo_lote←Z*custo_peça
   Escreva custo_peça,custo_total
Fim_subrotina
```

Anexo VI - Classes Utilizadas

Aqui são apresentadas as interfaces das principais classes do sistema. O relacionamento entre estas classes esta representado no diagrama do domínio do problema, Figura 6-5 .As classes que representam o código das interfaces gráficas não são apresentadas.

Classe Peça:

```
unit Unit_peca;

interface
uses classes,dialogs,sysutils,
Controls,unit_torno,unit_perfil,interface_perfil,
     unit_segmento,interface_operacao,unit_list_oper,unit_operacao
     ,unit_desbaste_externo,unit_acabamento_externo,unit_calculos,
     unit_furacao,pedido,lista_materiais,data_modulo2;
type
 peca=class (tobject)
 private
  tipo:byte;{0-desbaste_externo,1-desbaste_interno,2-desbaste_interno_externo,
             4-forjado_externo,5-forjado_interno,6-forjado_externo_interno}
  cn:tstringlist;
  perfil_bruto:perfil;   {peca bruta}
  perfil_externo:perfil; {perfil externo}
  perfil_interno:perfil; {perfil interno}
  operacoes:list_oper;
  lote:longint; {NÚMERO PEÇAS QUE SERÃO PRODUZIDAS}
  n:word;{numero do bloco do programa CN}
  neutro:tstringlist;
  {dados referentes ao material}
  codigo_prima:longint; {codigo da materia prima}
  dureza:integer;{dureza da peça}
  quant_requisita:integer;
  custo_unitario:real; //da materia prima
  quant_estoque:integer; //da materia prima
  procedure numerar_cn;
  procedure inicia_simulacao(op:operacao;x,y:real);
  procedure troca_ferramenta(op:operacao;x,y:real);
  procedure aval_pmeta(s1,s2:string;var pmeta:ponto);
 public
  procedure inicializa;
  procedure reinicializa;
  procedure roteiro_de_programacao;
  procedure coleta_dados_materiais;
  procedure gera_neutro;
  procedure neutro_prg;
  procedure gera_cn;
  procedure simula_peca;
  procedure faz_orcamento(list_mat:list_material);
  procedure le_tipo_oper(t:byte);
  procedure le_lote(l:longint);
```

```
  procedure le_materia_prima(s:string);
  procedure le_codigo_prima(n:longint);
  procedure le_dureza(n:integer);
  procedure le_quant_requisita(n:integer);
  procedure le_quant_estoque(n:integer);
  procedure le_custo_unitario(x:real);
  procedure le_operacoes;
  function vel_max_prod(K,X,Tft:real):integer;
  function vel_min_custo(K,X,Tft,Sh,Sm,Vsi,Nfp,Ns,Kpi:real):integer;
  function retorna_tipo_oper:byte;
  function retorna_lote:longint;
  function retorna_existe_bruto:boolean;
  function retorna_diametro:real;
  function retorna_comprimento:real;
  function retorna_perfil_bruto:perfil;
  function retorna_perfil_externo:perfil;
  function retorna_perfil_interno:perfil;
  function retorna_operacoes:list_oper;
  function retorna_neutro:tstringlist;
  function retorna_cn:tstringlist;
  function retorna_codigo_prima:longint;
  function retorna_dureza:integer;
  function retorna_numero_op:integer;
  function retorna_existe_perfil_ext:boolean;
  function retorna_existe_perfil_int:boolean;
  function retorna_existem_operacoes:boolean;
  function retorna_custo_material:real;
  function retorna_quant_estoque:longint;
 end;
 var
  obj_peca:peca;
```

Classe Operação:

```
unit Unit_operacao;

interface
uses windows,classes,sysutils,unit_perfil,dialogs,unit_calculos,pedido,
     data_modulo,math;
type
 poperacao=^operacao;
 operacao=class(tobject)
  protected
   tipo_oper:shortint;  {0-DESBASTE externo   1-desbaste interno   }
                        {2-ACABAMENTO externo 3-acabamento interno }
                        {4-sangra ext.        5-sangra_interno     }
                        {6-rosc ext           7-rosca interna      }
                        {8-furação                                 }
   estacao:integer;
   troca_x:real;
   troca_z:real;
   sentido_giro:boolean;
   fluido:boolean;
   tipo_vel:boolean;
   vel:real;
```

```
   max_rpm:real; {maximo rpm permitido para a operação}
   tipo_avanco:boolean;
   avanco:real;
   sobre_x:real;
   sobre_z:real;
   profund:real;
   {para a simulação}
   forma:string;
   tamanho:integer;
   tipo_porta_fer:string; //daqui retira-se o angulo de posição
   {para o calculo do custo e das velocidades da operação}
   codigo_porta_fer:longint; {dados das ferramentas para posterior busca}
   codigo_inserto:longint;
   t_op:real;           //tempo de corte da operação em minutos
   ferramenta:string;
   custo_unit:real;     //custo de cada inserto
   Ns:integer;          //número de arestas por inserto
   quant:integer;       //número de insertos necessário para o lote
   quant_estoq:integer; //quantidade disponivel em estoque
   X_taylor:real;
   K_taylor:real;
   function troca_ferramenta:string;
   function seleciona_ferramenta:string;
   procedure analisa_velocidade(var l:tstringlist);
   function is_longitudinal:boolean;           //verifica se a peça é
longelínea
   function tempo_g0(p1,p2:ponto):real;
   function tempo_g1(p1,p2:ponto):real;
   function tempo_g2(p1,p2,pc:ponto;r:real):real;
   function tempo_g3(p1,p2,pc:ponto;r:real):real;
   procedure repassa_perfil(desb:real;pfim:ponto;p:perfil;l:tstringlist);
   procedure posiciona_inicio(var l:tstringlist;var x,z:real);virtual;
   procedure analisa_operacao(var l:tstringlist);virtual;
  public
   //calcula a vida da ferramenta a partir de x e k de taylor para velocidade
escolhida
   function T:real;
   //calcula o número de insertos necessarios
   function n_insertos:integer;
   function custo:real;
   procedure coleta_dat_inserto;
   procedure coleta_dat_porta_fer;
   procedure le_forma(s:string);
   procedure le_tamanho(n:integer);
   procedure le_tipo_porta_fer(s:string);
   procedure le_tipo(t:shortint);
   procedure le_estacao(e:integer);
   procedure le_troca_x(t:real);
   procedure le_troca_z(t:real);
   procedure le_sobre_x(t:real);
   procedure le_sobre_z(t:real);
   procedure le_sentido_giro(s:boolean);
   procedure le_fluido(f:boolean);
   procedure le_tipo_vel(t:boolean);
   procedure le_tipo_avanco(t:boolean);
   procedure le_vel(v:real);
   procedure le_avanco(a:real);
```

```
   procedure le_profund(p:real);
   procedure le_max_rpm(m:real);
   procedure le_t_op(t:real);
   procedure le_ferramenta(s:string);
   procedure le_custo_unit(c:real);
   procedure le_ns(n:integer);
   procedure le_quant(n:integer);
   procedure le_quant_estoq(n:integer);
   procedure le_codigo_porta_fer(cod:longint);
   procedure le_codigo_inserto(cod:longint);
   procedure le_x_taylor(x:real);
   procedure le_k_taylor(k:real);
   function retorna_tipo_oper:string;
   function apresenta_tipo:shortint;
   function apresenta_estacao:integer;
   function apresenta_troca_x:real;
   function apresenta_troca_z:real;
   function apresenta_sentido_giro:boolean;
   function apresenta_fluido:boolean;
   function apresenta_tipo_vel:boolean;
   function apresenta_tipo_avanco:boolean;
   function apresenta_vel:real;
   function apresenta_avanco:real;
   function apresenta_profund:real;
   function apresenta_max_rpm:real;
   function apresenta_forma:char;
   function apresenta_tipo_porta_fer:char;
   function apresenta_l:integer;
   function apresenta_ang_pos:real;
   function apresenta_sobre_x:real;
   function apresenta_sobre_z:real;
   function apresenta_cod_inserto:longint;
   function apresenta_cod_porta_fer:longint;
   function apresenta_tempo_efetivo_de_corte:real;
   function apresenta_nome:string;virtual;
   procedure obter_dados;virtual;
   {procedimemto que apresenta os dados de cada operação}
   procedure mostra_dados;virtual;
   {apresenta o neutro de cada operação}
   procedure cria_string_geral(s:tstringlist);
   {le as strings de operações comuns a todas as operações}
   procedure le_string_geral(dado,dado1,dado2:string);
   procedure cria_string(s:tstringlist);virtual;
   procedure le_string(dado,dado1,dado2:string);virtual;
   procedure gera_cn_padrao(l:tstringlist);
   procedure area_de_trabalho;virtual;
   procedure calcula_velocidade;virtual;
  end;
```

Classes derivadas da classe Operação:

```
unit Unit_desbaste_externo;

interface
uses classes, dialogs,forms,Windows, Messages, SysUtils,  Graphics, Controls,
  StdCtrls, Menus,unit_calculos, unit_operacao,unit_perfil;
```

```
type
desbaste_externo=class(operacao)
  private
   procedure posiciona_inicio(var l:tstringlist;var x,z:real);override;
  public

   com_arremate:boolean;{determina se faz facemanento ou ciclo de torneamento}
   function apresenta_nome:string;override;
   procedure obter_dados;override;
   procedure mostra_dados;override;
   procedure cria_string(s:tstringlist);override;
   {analisa a operação em específico}
   procedure analisa_operacao(var l:tstringlist);override;
   procedure le_string(dado,dado1,dado2:string);override;
   procedure area_de_trabalho;override;
   procedure calcula_velocidade;override;
  end;
=====================================================================
unit Unit_acabamento_externo;

interface
uses classes, dialogs,forms,Windows, Messages, SysUtils,  Graphics, Controls,
  StdCtrls, Menus,unit_calculos, unit_operacao;
type
acabamento_externo=class(operacao)
  private
   procedure posiciona_inicio(var l:tstringlist;var x,z:real);override;
  public
   area_trabalho:string; {define qual o tipo de operação esta-se trabalhando}
   function apresenta_nome:string;override;
   procedure obter_dados;override;
   procedure mostra_dados;virtual;
   procedure cria_string(s:tstringlist);override;
   procedure ciclo_torneamento(var l:tstringlist);
   {analisa a operação em específico}
   procedure analisa_operacao(var l:tstringlist);override;
   procedure le_string(dado,dado1,dado2:string);override;
   procedure calcula_velocidade;override;
  end;
=====================================================================

unit Unit_furacao;

interface
uses classes, dialogs,forms,Windows, Messages, SysUtils,  Graphics, Controls,
  StdCtrls, Menus,unit_calculos, unit_operacao;
 type
  furacao=class(operacao)
  private
   procedure posiciona_inicio(var l:tstringlist;var x,z:real);override;
  public
   procedure cria_string(s:tstringlist);override;
   procedure le_string(dado,dado1,dado2:string);override;
  end;
```

Classe Torno:

```
unit Unit_torno;

interface
 uses unit_calculos,data_modulo2,dialogs;
 type
  torno=class(Tobject)
  private
   cnc:string;
   unidade:boolean;{true:milimetros;false:polegadas}
   programacao:boolean;{true:diâmetro,false:raio}
   tipo_vel:byte;{define o tipo de velocidade de corte:0-maxima produçao 1-min
custo 2-definido pelo usuario}
   tipo_bruto:byte;{define tipo da peça 0-bruta 1-forjado}
   Sh:real;{Custo do operador por hora}
   Sm:real;{custo da maquina por hora}
   ta:real;{tempo de aproximação}
   ts:real;{tempo secundário}
   tp:real;{tempo de preparação}
   tft:real;{tempo de troca da ferramenta em minutos}
   cancela:boolean; {indica que a tela foi cancelada}
   vtrack:byte;//velocidade do simulador
  public
   procedure inicializa;
   procedure coleta_dados_torno;
   procedure mostra_interface;
   procedure interface_refresh;
   //----- Para leitura de dados----------------------
   procedure le_cnc(s:string);
   procedure le_unidade(b:boolean);
   procedure le_programacao(b:boolean);
   procedure le_tipo_vel(bt:byte);
   procedure le_sh(r:real);
   procedure le_sm(r:real);
   procedure le_tft(r:real);
   procedure le_ta(r:real);
   procedure le_tp(r:real);
   procedure le_ts(r:real);
   procedure tela_cancelada(b:boolean);
   procedure le_string(dado:string);
   procedure le_tipo_bruto(b:byte);
   procedure le_vtrack(v:byte);
   procedure vel_track(vel:longint);
   //----- Para apresentação de dados-----------------
   function retorna_cnc:string;
   function retorna_unidade:boolean;
   function retorna_programacao:boolean;
   function retorna_tipo_vel:byte;
   function retorna_tipo_bruto:byte;
   function retorna_sh:real;
   function retorna_sm:real;
   function retorna_tft:real;

   function retorna_ta:real;
   function retorna_tp:real;
   function retorna_ts:real;
   function retorna_cancel:boolean;
   function retorna_vtrack:byte;
```

```
   function cria_string:string;
  end;

 var obj_torno:torno;
```

Classe Perfil:

```
unit Unit_perfil;
interface
uses classes, dialogs,forms,Windows, Messages, SysUtils,  Graphics, Controls,
  StdCtrls, Menus,unit_calculos, unit_segmento;
 type
  {------Criação de um perfil: uma lista de segmentos---------------}
  perfil=class(tobject)
   private
    mylist:tlist;
    pcursor:integer;{indica o número do segmento}
    tipo_perfil:byte;{1-perfil_bruto, 2-outros}
   public
    inicializada:boolean;
    procedure inicializa(tipo:byte);
    procedure adiciona_dados_segmento(oper:byte;pmeta:ponto;raio:real);
    procedure adiciona_seg(seg:segmento);
    procedure elimina_segmento(n:integer);
    procedure volta_cursor;
    procedure avanca_cursor;
    procedure limpa_perfil;
    procedure vai_inicio;
    function retorna_tipo_perfil:byte;
    function retorna_string_segmento(n:integer):string;
    function retorna_segmento(n:integer):segmento;
    {retorna a posição onde esta o cursor}
    function posicao_cursor:integer;
    function tamanho_lista:integer;
    function is_longitudinal:boolean;
    function retorna_y_min:real;
    function retorna_y_max:real;
    function retorna_x_min:real;
    function retorna_x_max:real;
    function retorna_x(y:real):real;
    function retorna_y(y:real):real;
    {acha o segmento a partir de um valor de y. False se não achar}
    function acha_seg_x(y:real;var i:integer):boolean;
    function is_esquerda_direita:boolean;
    function faz_faceamento(var zi,zf:real):boolean;
   end;
```

Classe Segmento:

```
unit unit_segmento;

interface
 uses unit_calculos;
```

```
 type
{-----------Criação de uma classe de objetos -----------------------}
  segmento=class(tobject)
   private
   public
    oper:byte; {0:linear;1:circ_hor;2:circ_anti;4-inicio;5:peça_bruta}
    pmeta:ponto;
    raio:real;
    {Converte os dados de um segmento em uma string}
    function seg_string:string;
    {extrai os dados para o segmento a partir de uma string}
    procedure le_string(dado:string);
  end;
```

Classe Simulador:

```
unit simula;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,

unit_formas,unit_quadrilateros,unit_triangulos,unit_poli_linhas,unit_circulos,
  unit_calculos,unit_perfil,unit_torno,unit_peca,unit_segmento;

type
  Tsimulador = class(TForm)
  private
    { Private declarations }
    K:real; {constante de proporcionalidade}
    ConstLin:integer;{constante linear}
    programacao:boolean; {true:diametro;false:raio}
    vel:longint;
  public
  { Public declarations }
    fer:forma;
    peca:forma;
    F_perfil:forma;
    procedure cria_ferramenta(tipo:char;l,ref_x,ref_y,ang_pos:real);
    procedure destroi_ferramenta;
    procedure cria_peca(comp,diam:real);
    procedure escala(ptx:real);
    procedure move_linear(pmeta:ponto);
    procedure move_circ_hor(pmeta:ponto;r:real);
    procedure move_circ_anti(pmeta:ponto;r:real);
    procedure inicializa(tipo:byte;maior_dimensao:integer;prog:boolean); {1-
tela pequena , 2-tela grande}
    procedure desenha_peca_bruta(diam,comp:real);
    procedure apresenta_perfil(p:perfil);
    procedure simula_perfis;
    {ajusta velocidade de simulação}
    procedure track_vel(track:integer);
    procedure muda_vel(v:longint);
    function retorna_k:real;
    function retorna_vel:longint;
  end;
```

Classe Forma:

```
unit unit_formas;

interface
 uses graphics,windows,classes,unit_calculos,dialogs,math;
type

  forma=class
  protected
   c:tcanvas;
   K:real; {constante de proporcionalidade}
   ConstLin:integer;{constante linear}
   programacao:boolean;{se é em diametro ou raio}
   cor:Tcolor; {cor do objeto}
   fundo:Tcolor;{cor do fundo}
   referencia:ponto; {ponto de referencia do objeto no simulador}
   preenche:boolean; {o objeto sera preenchido? true=sim, para objetos nao
retangulares}
   ang:real;{ang_posição da ferramenta}
   l:real;{comprimento do lado}
  public
   procedure desenhar(c:tcanvas);virtual;
   procedure
inicializa(canvas:tcanvas;tipo_quadri:byte;k_pro:real;constlinear:integer;prog
:boolean);virtual;
   procedure define_forma(ponto_inicial:ponto;ang_posicao,comp:real);virtual;
   {funcaõ que move o objeto lineamente para do ponto atual para um ponto
meta}
   procedure move_linear(c:tcanvas;pmeta:ponto);
   procedure apaga(c:tcanvas);
   procedure desenha(c:tcanvas);
   procedure finaliza(c:tcanvas;p:ponto);
   procedure mov_circ_hor(c:tcanvas;pf:ponto;r:real);
   procedure mov_circ_anti(c:tcanvas;pf:ponto;r:real);
   {faz a transformação linear de um ponto p para pf }
   procedure converte(p:ponto;var pf:ponto);
   function converte_valor(valor:real):real;
   procedure define_cor(clinha,cfundo:tcolor);
   procedure preencher;
   procedure le_lado(lado:real);
   procedure le_ang(a:real);
   procedure le_cor(c:tcolor);
   procedure le_fundo(c:tcolor);
   procedure le_preenche(b:boolean);
   procedure le_ref(x,y:real);
  end;
```

Classes derivadas da Classe Forma:

```
unit Unit_quadrilateros;

interface
 uses unit_formas,unit_calculos,windows,graphics;
```

```
 type
 Quadrilatero=class(forma)
  private
   tipo:byte;{0: S-quadrada;
             {1: C-retangular com 80 graus ou }
             {  D-retangular com 55 graus ou V-retangular com 35 graus}
             {2: K-trapezoidal com 55 graus}
             {3: retangular para formar a peca comp=l diam=ang}
   ang_int:real;{angulo interno da ferramenta}
   p:ponto;
  public
   procedure
inicializa(canvas:tcanvas;tipo_quadri:byte;k_pro:real;constlinear:integer;prog
:boolean);override;
   procedure define_forma(ponto_inicial:ponto;ang_posicao,comp:real);override;
   procedure desenhar(c:tcanvas);override;
   procedure le_tipo(t:byte);
   procedure le_ang_int(a:real);
  end;
=========================================================================

unit Unit_triangulos;

interface
 uses unit_formas,unit_calculos,windows,graphics;
 type
   Triangulo=class(forma)
  public
   l:real;{comprimento do lado}
   ang:real;{ang_posição da ferramenta}
   procedure desenhar(c:tcanvas);override;
  end;
=========================================================================

unit unit_poli_linhas;

interface
uses unit_formas,unit_calculos,windows,graphics,unit_perfil,unit_segmento;
type
 Poli_linha=class(forma)
  private
   p:perfil;
  public
   //perfil:perfil;
   procedure
inicializa(canvas:tcanvas;escala:real;cl:integer;prog:boolean;per:perfil);
   procedure desenhar(c:tcanvas);override;   {procedure
desenhar_poli(c:tcanvas;q:array of tpoint); }
   procedure horario(raio:real;pant,pmeta:ponto;var
canto1,canto2,inicio,fim:ponto);
   procedure anti_horario(raio:real;pant,pmeta:ponto;var
canto1,canto2,inicio,fim:ponto);
  end;
=========================================================================

unit unit_circulos;
```

```
interface
uses unit_formas,unit_calculos,windows,graphics;
 type
 circulo=class(forma)
  private
    centro:tpoint;
  public
   procedure desenhar(c:tcanvas);override;
  end;
```