RAFAEL MONTEIRO SEIFERT

SOFTWARE PARA A SIMULAÇÃO E TELE-OPERAÇÃO DE UMA MÁQUINA FERRAMENTA

Monografia apresentada a Escola Politécnica da Universidade de São Paulo para a Conclusão de Curso.

São Paulo
2010

RAFAEL MONTEIRO SEIFERT

SOFTWARE PARA A SIMULAÇÃO E TELE-OPERAÇÃO DE UMA MÁQUINA FERRAMENTA

Monografia apresentada a Escola Politécnica da Universidade de São Paulo para a Conclusão de Curso.

Curso de Graduação: Engenharia Mecatrônica

Orientador: Prof. Dr. Fabrício Junqueira

São Paulo

2010

Aos meus pais e meus amigos.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DA ENGENHARIA MECÂNICA/NAVAL DA ESCOLA POLITÉCNICA (EPMN) – USP.

Seifert, Rafael Monteiro
Software para a simulação e tele-operação de uma máquina ferramenta / R.M. Seifert. -- São Paulo, 2010.
63 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1. Maquinas-ferramenta 2. Softwares I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrô-nica e de Sistemas Mecânicos II. t.

Agradecimentos

Ao meu orientador Prof. Dr. Fabrício Junqueira pela sua constante orientação e pela ajuda nos momentos difíceis do trabalho.

Aos professores da Escola Politécnica da USP, por terem me passado seu conhecimento e me ensinado a enfrentar desafios.

A toda a minha família pelo carinho e apoio desde sempre.

Aos meus amigos e colegas de graduação pela companhia e amizade ao longo destes anos.

A todos aqueles que contribuíram de direta ou indiretamente na produção deste trabalho.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

No contexto do desenvolvimento da internet, o conceito de *e-manufacturing*, que possibilita o acesso remoto à máquina ferramenta por meio de uma rede de comunicação, se torna cada vez mais atraente. Isto permitirá integrar a tele-operação de uma máquina ferramenta com a possibilidade de emular a usinagem em um ambiente de manufatura virtual, tornando mais eficiente a análise e o controle dos fluxos e processos de produção. O objetivo deste trabalho é a implementação de um *software* que simule uma máquina CNC e que permita acesso remoto via internet. O *software* deve interpretar programas em código G e permitir sua validação e submissão por meio da rede, assim como fornecer ao usuário informações quanto às características físicas e a capacidade de usinagem da máquina. Ainda, ele deve possuir um ambiente gráfico que simule o processo de usinagem, permitindo o acompanhamento das operações. No futuro, este trabalho pode servir de base para o desenvolvimento de sistemas de seleção automática de equipamentos.

Palavras chave: Máquina ferramenta, *e-manufacturing*, manufatura virtual, CNC, *software*.

Abstract

In the context of the recent development of the internet, the concept of e-manufacturing, which gives the possibility of accessing machine tools remotely by using a network, becomes increasingly attractive. This will allow tele-operation to integrate with the emulation of a machining process through virtual manufacturing, making the analysis and control of production flow and processes more efficient. The goal of this project is to implement a computer program which simulates a computer numerically controlled (CNC) machine tool and enables internet based remote access. The software must compile, validate and submit a G-code program to the machine. It should also inform the client of machine capacity and its physical characteristics, as well as present a graphic interface capable of simulating the machining process, thus permitting the software to monitor the operations. In the future, this project may provide a base for the development of automatic machine selection systems.

Keywords: Machine tool, e-manufacturing, virtual manufacturing, CNC, software.

1. Introdução

A manufatura mecânica é a atividade de transformar a matéria prima em produto acabado. Este produto pode ser um bem final ou intermediário, ou seja, pode ser utilizado em outros setores da indústria. Esta segunda classe de produtos é de extrema relevância, dado que a maior parte das indústrias utiliza, em algum momento, componentes metálicos fabricados por meio de processos relacionados à manufatura mecânica como: torneamento, fresamento, forjamento, soldagem ou estampagem, entre outros (ALTINTAS, 2000).

Os processos de usinagem, como torneamento, fresamento e furação, diferenciam-se pelo fato de que removem material da peça que está sendo fabricada. O serviço oferecido por máquinas deste tipo apresenta, em geral, dimensões limitadas pois as máquinas precisam abrigar o material a ser usinado. Além disso, as operações são repetidas muitas vezes, pois a produção geralmente é em série e em escala. Estes processos possibilitam a obtenção de elevada precisão, e o grau de tecnologia empregada é alto. Partes usinadas alimentam todos os setores industriais, sob a forma de parafusos, pinos, eixos e processos de melhoria de acabamento, por exemplo (ALTINTAS, 2000) (STOETERAU, 2007).

A partir da década de 50, a qualidade e velocidade das operações de usinagem foram impulsionadas pelo surgimento e o desenvolvimento de uma tecnologia que recebeu o nome de NC (*Numeric Control*), e posteriormente CNC (*Computer Numeric Control*). Este foi um grande avanço no sentido de automatizar a produção e melhorar a qualidade do produto (ALTINTAS, 2000).

Na manufatura de usinagem, a computação se mostrou uma poderosa ferramenta em vários aspectos, como supervisão de sinais e sensores, comunicação entre pessoas e entre máquinas, armazenamento de dados, execução de cálculos, etc. Com a tecnologia CNC, pode-se delegar uma tarefa, em forma de instruções, a uma máquina ferramenta, tornando a operação livre da interferência humana.

A lista de instruções é escrita numa linguagem conhecida como código G, padronizada pela ISO (*International Organization for Standardization*) (MUNDO CNC, 2008). A sequência de comandos em G é interpretada pela máquina CNC, que a executa, comando a comando. Só com isso já é possível reduzir o tempo gasto com a usinagem de uma peça, pois os intervalos entre as etapas da operação são eliminados. Além disso, para produção em série, basta trocar a peça a ser usinada e repetir o código. Isto simplificou muito o trabalho dos operadores e elevou a qualidade do produto por causa da acurácia obtida.

O código G pode ser interpretado por um programa de computador que simula virtualmente as etapas da operação e, com isso, é possível identificar possíveis erros de codificação antes de colocar a peça à prova. Esta simulação pode, ainda, apresentar de forma gráfica e em tempo real todo o processo de usinagem de uma peça, para melhor visualização das instruções em linguagem G (MOLLICA, 2001).

1.1. Motivação

O mercado atualmente é caracterizado por acirrada competição internacional, por produtos cada vez mais complexos e por grande dinâmica inovadora. Juntamente com os ciclos de inovação cada vez mais reduzidos, os ciclos de vida dos produtos e o tempo para lançá-los no mercado são cada vez menores (PORTO e PALMA, 2000) (SOUZA, *et al.*, 2002).

Considerando esse contexto, juntamente com o grande avanço tecnológico, cada vez mais as empresas estão buscando novas formas de alcançar vantagem competitiva e introduzir novos produtos no mercado mais rapidamente e a um custo menor, como utilizando, por exemplo, o ambiente de Manufatura Virtual. A idéia é que esse novo ambiente aborde todo o processo de desenvolvimento, simulação e fabricação do produto, possibilitando a execução dessas atividades no computador, ou seja, virtualmente, antes de realizá-las no mundo real, independentemente do grau de complexidade da forma e da estrutura de um produto (SOUZA, *et al.*, 2002).

1.2. Objetivo

A proposta deste trabalho é a implementação um ambiente de manufatura virtual. Mais especificamente, um sistema computacional capaz de interpretar código G, verificar sua sintaxe, simular um processo de usinagem e exibir graficamente o percurso da ferramenta e o *status* da operação, com o intuito de simular a disponibilização de máquinas reais na web, como serviços. Entretanto, devido ao alto custo deste tipo de equipamento, torna-se mais viável o desenvolvimento de um *software* que simula seu comportamento, com base nas informações sobre seu funcionamento e possíveis sinais de sensores. Com este sistema, é possível a submissão remota de programas e comandos, bem como a visualização de operações e *status*.

A ferramenta utilizada para o desenvolvimento de todo o *software* é o Microsoft Visual Studio 2010. O programa que representa a máquina virtual foi codificado em uma estrutura que, posteriormente, poderá ser substituída pela máquina real com baixo impacto para o restante do código.

Este trabalho possibilitará, no futuro, integrar completamente os níveis de simulação e fabricação, de forma que, a partir das informações de cada máquina em uma rede de manufatura, o sistema consiga alocar automaticamente as tarefas para os equipamentos conectados, otimizando a utilização destes. Os critérios para isto seriam ocupação ou não do equipamento e os atributos específicos, como material usinado e velocidade máxima de avanço, por exemplo.

Com isto, espera-se alcançar uma redução no tempo de entrega, acelerando a produção. Isto seria obtido graças ao alto grau de automação do sistema, que poderia tomar decisões de seleção de equipamento. Além disso, não haveria a necessidade de acompanhamento físico da operação, já que esta tarefa poderia ser feita à distância, pela web.

1.3. Organização

Este relatório possui quatro seções principais. Na Revisão Bibliográfica consta toda a referência utilizada para a formulação do projeto, que inclui uma descrição de máquina CNC e código G, uma explicação dos conceitos de

Manufatura Virtual e *e-Manufaturing*, assim como as definições e as arquiteturas de *web services* e de UML. Nessa seção são especialmente importantes as descrições dos diagramas de caso de uso, classes e sequências da UML, pois eles formam a base da modelagem da arquitetura do *software*.

Na seção Detalhamento do projeto consta a modelagem do *software,* com suas funcionalidades e especificações.

Em seguida, na Descrição do programa, são explicadas detalhadamente as principais funções do programa, a sua interface gráfica e como é feito o acesso remoto.

A seção de Testes realizados apresenta os resultados de duas simulações, sendo uma de operação local e a outra via web.

Finalmente, na seção Conclusões, discutem-se os resultados atingidos, as limitações do projeto e os pontos de melhoria.

2. Revisão Bibliográfica

2.1. Máquinas CNC

Máquinas de comando numérico são equipamentos de usinagem que são controlados por dispositivo eletrônico capaz de receber informações por meio de entrada própria, compilar estas informações e transmiti-las em forma de comando à máquina operatriz, de modo que esta, sem a intervenção do operador, realize as operações na sequência programada. Estas máquinas se diferenciam das convencionais por não necessitarem de acessórios que proporcionem o controle dos movimentos da máquina, tais como gabaritos, cames, limites etc. e mesmo a interferência direta do operador. Estes movimentos são comandados por meio dos dados de entrada, garantindo uniformidade e qualidade de peça e lotes (MOLLICA, 2001).

Além de solucionar usinagem de peças de grande complexidade, as máquinas de controle numérico auxiliam na redução de tempos improdutivos, no posicionamento e retirada da ferramenta de corte, e ainda introduziram grande flexibilidade no processo de fabricação, pois para se alterar a operação basta que se modifique a sequência de instruções (MOLLICA, 2001).

Em 1947, John Parsons, da Parsons Corporation, iniciou experimentos com a idéia de utilizar dados de curvatura de três eixos para controlar movimentos de ferramentas em máquinas para a produção de componentes para a indústria de aviões. Em 1948, Parsons foi contratado pela Força Aérea dos Estados Unidos para construir o que viria a ser o primeiro comando numérico. Em 1951, o projeto foi assumido pelo MIT (*Massachusets Institute of Technology*). Em 1952, o comando numérico ficou pronto e demonstrou que movimentos simultâneos com três eixos eram possíveis, usando um controlador construído em laboratório e um eixo árvore (spindle) vertical. Em torno de 1955, depois de alguns refinamentos, o comando numérico tornou-se disponível para a indústria (BRUNE, 2008).

Nos anos 70, máquinas operatrizes de Controle Numérico Computadorizado (CNC) foram desenvolvidas com microcomputadores usados como unidades de controle. Com os avanços em eletrônica e computação,

sistemas modernos CNC empregam diversos microprocessadores de alto desempenho e CLPs (Controlador Lógico Programável) que trabalham coordenadamente, em paralelo. Os sistemas atuais de CNC permitem, simultaneamente, controle de posição dos servomotores e das velocidades em todos os eixos, monitoramento do desempenho do controlador e da máquina ferramenta, programação computacional com suporte gráfico, monitoramento do processo de corte e medição automática, possibilitando operações totalmente livres de interferência humana (ALTINTAS, 2000).

Um torno CNC pode ser visto na Figura 2.1.

Figura 2.1. Torno CNC[1].

Com o surgimento do controle numérico foi necessário desenvolver uma linguagem compreensível pelos controles das máquinas e esta deveria ser padronizada para que minimizasse o efeito "Torre de Babel"[2] tão comum em tecnologias emergentes. Deste modo a EIA Standards, (Associação das industrias elétricas dos EUA) e posteriormente e em nível mundial a ISO (*International Organization for Standardization*) adotaram algumas

[1] Imagem retirada da web - http://www.limabonfa.com.br/images/Torno_CNC.gif
[2] Em engenharia, diz-se da multiplicidade de linguagens incomunicáveis, dificultando a integração de componentes.

prerrogativas, dentre elas a distinção entre código G (*general* ou *preparatory*) e código M (*miscelaneous*) (MUNDO CNC, 2008).

A linguagem de programação de CNC é comumente conhecida como código G, mas engloba tanto as funções G quanto as funções M. Consiste em linhas de código que são interpretadas pela máquina e traduzidas em movimentos ou outras ações. A Tabela 2.1 mostra a lista de códigos G e a Tabela 2.2 apresenta os códigos M de acordo com a norma ISO 1056.

Tabela 2.1. Código G pelo Padrão ISO 1056.

Código G	Função
G00	Posicionamento rápido
G01	Interpolação linear
G02	Interpolação circular no sentido horario (CW)
G03	Interpolação circular no sentido anti-horario (CCW)
G04	Temporização (Dwell)
G05	Não registrado
G06	Interpolação parabólica
G07	Não registrado
G08	Aceleração
G09	Desaceleração
G10 a G16	Não registrado
G17	Seleção do plano XY
G18	Seleção do plano ZX
G19	Seleção do plano YZ
G20	Programação em sistema Inglês (Polegadas)
G21	Programação em sistema Internacional (Métrico)
G22 a G24	Não registrado
G25 a G27	Permanentemente não registrado
G28	Retorna a posição do Zero máquina
G29 a G32	Não registrados
G33	Corte em linha, com avanço constante
G34	Corte em linha, com avanço acelerando
G35	Corte em linha, com avanço desacelerando

Tabela 2.1. Código G pelo Padrão ISO 1056 (Continuação).

Código G	Função
G36 a G39	Permanentemente não registrado
G40	Cancelamento da compensação do diâmetro da ferramenta
G41	Compensação do diâmetro da ferramenta (Esquerda)
G42	Compensação do diâmetro da ferramenta (Direita)
G43	Compensação do comprimento da ferramenta (Positivo)
G44	Compensação do comprimento da ferramenta (Negativo)
G45 a G52	Compensações de comprimentos das ferramentas
G53	Cancelamento das configurações de posicionamento fora do
G54	Zeragem dos eixos fora do zero fixo (01)
G55	Zeragem dos eixos fora do zero fixo (02)
G56	Zeragem dos eixos fora do zero fixo (03)
G57	Zeragem dos eixos fora do zero fixo (04)
G58	Zeragem dos eixos fora do zero fixo (05)
G59	Zeragem dos eixos fora do zero fixo (06)
G60	Posicionamento exato (Fino)
G61	Posicionamento exato (Médio)
G62	Posicionamento (Groceiro)
G63	Habilitar óleo refrigerante por dentro da ferramenta
G64 a G67	Não registrados
G68	Compensação da ferramenta por dentro do raio de canto
G69	Compensação da ferramenta por fora do raio de canto
G70	Programa em Polegadas
G71	Programa em metros
G72 a G79	Não registrados
G80	Cancelamento dos ciclos fixos
G81 a G89	Ciclos fixos
G90	Posicionamento absoluto
G91	Posicionamento incremental
G92	Zeragem de eixos (mandatório sobre os G54...)
G93	Avanço dado em tempo inverso (Inverse Time)
G94	Avanço dado em minutos

Tabela 2.1. Código G pelo Padrão ISO 1056 (Continuação).

Código G	Função
G95	Avanço por revolução
G96	Avanço constante sobre superfícies
G97	Rotação do fuso dado em RPM
G98 e G99	Não registrados

Tabela 2.2. Código M pelo Padrão ISO 1056.

Código M	Função
M00	Parada programa
M01	Parada opcional
M02	Fim de programa
M03	Liga o fuso no sentido horário (CW)
M04	Liga o fuso no sentido anti-horário (CCW)
M05	Desliga o fuso
M06	Mudança de ferramenta
M07	Liga sistema de refrigeração numero 2
M08	Liga sistema de refrigeração numero 1
M09	Desliga o refrigerante
M10	Atua travamento de eixo
M11	Desliga atuação do travamento de eixo
M12	Não registrado
M13	Liga o fuso no sentido horário e refrigerante
M14	Liga o fuso no sentido anti-horário e o refrigerante
M15	Movimentos positivos (aciona sistema de espelhamento)
M16	Movimentos negativos
M17 e M18	Não registrados
M19	Parada do fuso com orientação
M20 a M29	Permanentemente não registrado
M30	Fim de fita com rebobinamento
M31	Ligando o "Bypass"
M32 a M35	Não registrados.
M36	Acionamento da primeira gama de velocidade dos eixos

Tabela 2.2. Código M pelo Padrão ISO 1056 (Continuação).

Código M	Função
M37	Acionamento da segunda gama de velocidade dos eixos
M38	Acionamento da primeira gama de velocidade de rotação
M39	Acionamento da segunda gama de velocidade de rotação
M40 a M45	Mudanças de engrenagens se usada, caso não use, Não
M46 e M47	Não registrados.
M48	Cancelamento do G49
M49	Desligando o "Bypass"
M50	Liga sistema de refrigeração numero 3
M51	Liga sistema de refrigeração numero 4
M52 a M54	Não registrados.
M55	Reposicionamento linear da ferramenta 1
M56	Reposicionamento linear da ferramenta 2
M57 a M59	Não registrados
M60	Mudança de posição de trabalho
M61	Reposicionamento linear da peça 1
M62	Reposicionamento linear da peça 2
M63 a M70	Não registrados.
M71	Reposicionamento angular da peça 1
M72	Reposicionamento angular da peça 2
M73 a M89	Não registrados.
M90 a M99	Permanentemente não registrados

2.2. Manufatura Virtual e e-Manufacturing

2.2.1. Manufatura Virtual

Dado que a proposição é criar um *software* que seja um ambiente de simulação de máquinas ferramentas, é importante atentar ao conceito de Manufatura Virtual e aos paradigmas associados à sua definição pertinentes ao projeto em questão para determinar suas potenciais aplicações.

O termo Manufatura Virtual passou a ser utilizado em meados dos anos 90, em parte como resultado da iniciativa do Departamento de Defesa dos

EUA, na qual a evolução do ambiente de defesa e das estratégias de aquisição propiciou o desenvolvimento da capacidade de confirmar a viabilidade de novos sistemas de armas antes de comprometer recursos de produção. Até metade dessa década, os primeiros trabalhos neste campo foram realizados, na maior parte dos casos, por organizações como as indústrias aeroespacial e automobilística (BANERJEE e ZETU, 2001).

Os sistemas de Manufatura Virtual são modelos integrados baseados em computador que representam tanto o comportamento como o esquema físico e lógico dos sistemas de manufatura reais (ONOSATO e IWATA, 1993);

A Manufatura Virtual também pode ser definida como “um ambiente integrado e sintético da manufatura exercido para melhorar todos os níveis de controle e decisão” (LAWRENCE ASSOCIATES INC, 1994).

Um sistema de Manufatura Virtual contém um modelo do produto, do processo e dos recursos, além de contar com uma prototipagem virtual (KIMURA, 1993). Consequentemente, por meio de um sistema de Manufatura Virtual é possível examinar vários dos paradigmas da engenharia relacionados à manufatura, tais como o projeto do produto, o planejamento do processo, e o controle da operação no chão-de-fábrica (LEE e NOH, 1997).

Quando novos sistemas ou produtos são desenvolvidos, os custos do ciclo de vida são determinados por diversas decisões tomadas nas primeiras etapas do ciclo de desenvolvimento, na fase de definição do conceito. Corrigir erros encontrados no produto final ou em estágios avançados do processo de desenvolvimento causados por decisões tomadas nas etapas iniciais envolve mudanças no projeto que consomem tempo e recursos. O principal benefício esperado com a utilização de sistemas de Manufatura Virtual é a redução do tempo de ciclo e do custo de desenvolvimento de produto (LEE, CHEUNG e LI, 2001).

Desta forma, a Manufatura Virtual propõe prover uma estratégia para a integração de vários processos de manufatura associados à fabricação e à montagem de um produto, sendo que tais processos podem ser simulados em um computador por intermédio de um ambiente de modelagem e simulação. A capacidade de um sistema de Manufatura Virtual contempla todas as variáveis

do ambiente de produção, das informações de projeto aos processos de chão de fábrica e às transações empresariais. Ou seja, ela considera as interações dos processos de produção, planejamento do processo e da montagem, programação, logística das linhas da empresa e os processos associados, como contabilidade, compras e gerenciamento (SOUZA, *et al.*, 2002).

Como pode-se ver pela Figura 2.2, o escopo do sistema de Manufatura Virtual engloba os 3 diferentes paradigmas a seguir (SOUZA, *et al.*, 2002):

- Orientado para o projeto: fornece informações sobre a manufatura ao processo de desenvolvimento, permitindo a simulação de diversas alternativas de manufatura e a criação de protótipos no computador. Consiste no uso de simulações baseadas na manufatura para otimizar o projeto do produto em uma meta específica da manufatura, como por exemplo o DFA (*Design for Assembly*) ou a sua flexibilidade;
- Orientado para a produção: fornece capacidade de simulação aos modelos dos processos de manufatura com o propósito de permitir o planejamento de recursos necessários, a avaliação de diversas alternativas de processamento e a geração e validação de novos planos de processo de modo rápido e econômico, sem que para isso sejam executados processos reais;
- Orientado para o controle: oferece um ambiente para a simulação dos modelos de controle de chão de fábrica, assim como de uma máquina real, permitindo a otimização de seus parâmetros durante o ciclo de produção existente. Portanto, este paradigma oferece um ambiente para a avaliação de projetos novos ou revisados em relação às atividades atuais de chão de fábrica.

Do modo como os paradigmas relacionados à Manufatura Virtual foram apresentados, um *software* de simulação de máquinas ferramenta se aproxima mais dos paradigmas orientados para o projeto e de controle. As suas funcionalidades compreendem:

- Fornecer um ambiente para que os projetistas avaliem a manufaturabilidade e a viabilidade do projeto de um produto por meio da construção de um protótipo virtual;
- Simulação da máquina na qual será realizada a usinagem. Parâmetros como a sua dimensão e o comportamento dinâmico da máquina durante a usinagem constituem informações importantes para a realização do projeto;
- Teste e validação da precisão do projeto de um produto. Isto inclui atividades como a validação de programas de controle numérico, a verificação da trajetória da sua ferramenta e a checagem de possíveis colisões durante a usinagem.

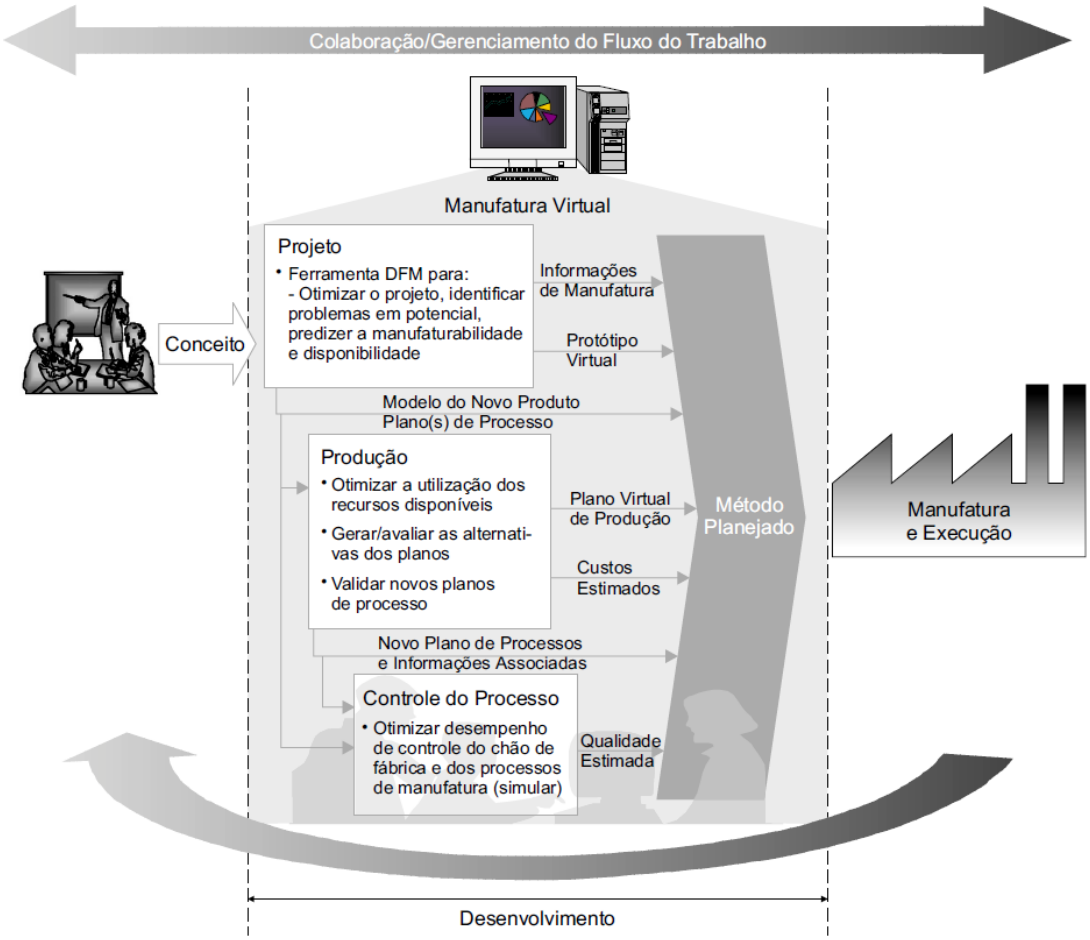

Figura 2.2. Ambiente da manufatura virtual (PORTO e PALMA, 2000).

2.2.2. *e-Manufacturing*

A *e-Manufacturing* se dá com a implementação de redes de comunicação em todos os níveis da manufatura. Nos níveis inferiores, redes de comunicação fornecem uma maior confiabilidade e visibilidade à infra-estrutura de uma planta de fábrica. Ainda, eles capacitam o controle distribuído da planta e a interoperabilidade entre máquinas.

Em níveis mais altos, redes de comunicação podem padronizar serviços de forma a permitir que a automação da programação, o controle e o diagnóstico dos processos de manufatura sejam feitos no escopo da fábrica e não mais apenas no escopo da máquina (MOYNE e TILBURY, 2007).

A rede a ser estabelecida para este projeto será baseada na estrutura de *web services.* Ela é baseada em uma arquitetura cliente/servidor, que trabalha de forma interativa e cooperativa, solicitando e provendo serviços.

2.3. Web Services

2.3.1. Definição

W3C Working Group Note - Web Service Architecture (WSA)

O relatório técnico WSA da *World Wide Web Consortium* (W3C), escrito no período de 2002 a 2004, fornece uma definição de um *web service* ao descrever as características comuns e necessárias à maioria dos *web services*.

Este relatório não visa especificar como um *web service* deve ser implementado, e sim guiar a comunidade a uma maior compreensão do conceito de *web service* e das relações entre os componentes de sua arquitetura.

Assim, a definição de *web service* aqui apresentada, assim como o esquema geral de sua arquitetura foram consultadas no relatório WSA.

Definição e propriedades

Web service é um sistema de *software* designado a servir de suporte para interações entre máquinas de forma interoperável por uma rede de

comunicação. Ele possui uma interface descrita em um formato processável por máquina. Outros sistemas interagem com o *web service* de acordo como o especificado pela sua descrição, utilizando-se de protocolos de comunicação baseados na sintaxe XML (*eXtensible Markup Language)* (W3C WORKING GROUP NOTE, 2004).

Como o XML é um padrão aberto de codificação de informações, os *web services* são independentes em relação a sistemas operacionais, linguagem de programação e *hardware*. Isso significa que aplicativos codificados em diferentes linguagens de programação podem facilmente trocar informações entre si através da Internet ou de uma rede local por meio de *web services,* como mostra a Figura 2.3. (ALTOVA, 2006).

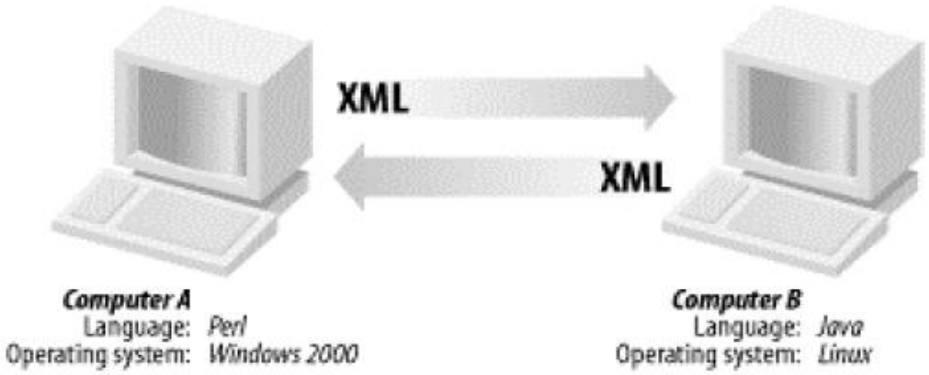

Figura 2.3. Esquema básico de *web service* (CERAMI, 2002).

Embora não seja estritamente necessário, é desejável que um *web service* tenha outras duas propriedades adicionais (CERAMI, 2002):

- *Web services* devem ser auto-explicativos: para facilitar a integração com outros sistemas, um *web service* deve incluir pelo menos uma documentação textual. É interessante que ele inclua uma interface pública, codificada em XML, que possa ser utilizada para identificar todos os métodos públicos, seus parâmetros e valores retornados;
- *Web services* devem ser localizáveis: ao criar um *web service*, é recomendável que exista uma forma de publicá-lo. Da mesma forma, é interessante que exista um mecanismo pelo qual interessados possam procurá-lo e assim encontrar sua interface pública.

O desenvolvimento baseado em *web services* está se tornando a estratégia dominante para a representação e distribuição de informações através de múltiplos sistemas, inclusive para aqueles que não foram inicialmente planejados ou desenvolvidos para interagirem entre si (BELL, DE CESARE, *et al.*, 2006).

2.3.2. Arquitetura

Arquitetura do ponto de vista dos agentes

Existem duas formas de visualizar a arquitetura de um *web service* (CERAMI, 2002). A primeira delas é examinar as funções de cada agente na sua arquitetura.

Um *web service* é uma abstração que deve ser implementada por um agente concreto. Portanto, um agente é um *software* que envia e recebe mensagens, enquanto que o serviço em si é o recurso caracterizado pelo conjunto abstrato de funcionalidades fornecidas (W3C WORKING GROUP NOTE, 2004).

São três os principais agentes presentes na arquitetura de um *software*:

- Provedor: o provedor implementa o *web service* e o torna disponível na rede;
- Solicitante: é qualquer cliente do *web service.* Ele pode utilizar um *web service* existente ao estabelecer uma conexão e enviar uma solicitação no formato XML;
- Registro: é um diretório centralizado de *web services*, permitindo aos desenvolvedores publicar novos serviços e encontrar os já existentes.

A Figura 2.4 mostra os principais agentes na arquitetura de uma *web service,* assim como as interações que ocorrem entre elas.

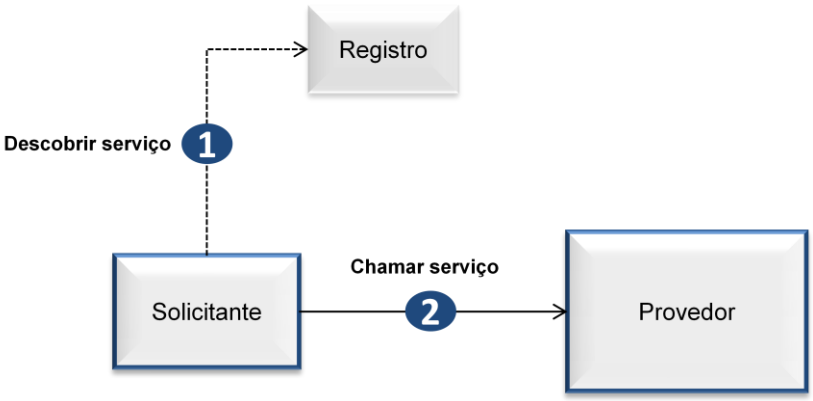

Figura 2.4. Principais agentes de um *web service* (CERAMI, 2002).

Em um cenário hipotético, supondo que a empresa A faça um pedido de compra de peças para a empresa B e que a empresa A deseja monitorar o *status* do seu pedido por meio de um aplicativo que controla seu estoque, pode-se estabelecer os seguintes procedimentos:

- O aplicativo que controla o estoque da empresa A se conecta ao registro de serviços, e inquire se há um serviço de monitoração do *status* de pedidos providenciado pela empresa B;
- O aplicativo então se conecta ao servidor da empresa B e recupera a descrição do serviço desejado, que contém detalhes completos de como o aplicativo deve se conectar ao serviço;
- De posse dessas informações, o aplicativo pode chamar o serviço e retornar o *status* do pedido.

Neste cenário, a Figura 2.4 teria como provedor do *web service* o servidor da empresa B e o solicitante seria o aplicativo de estoque da empresa A. As mudanças, assim como as respectivas transações entre cada um dos agentes estão representadas na Figura 2.5.

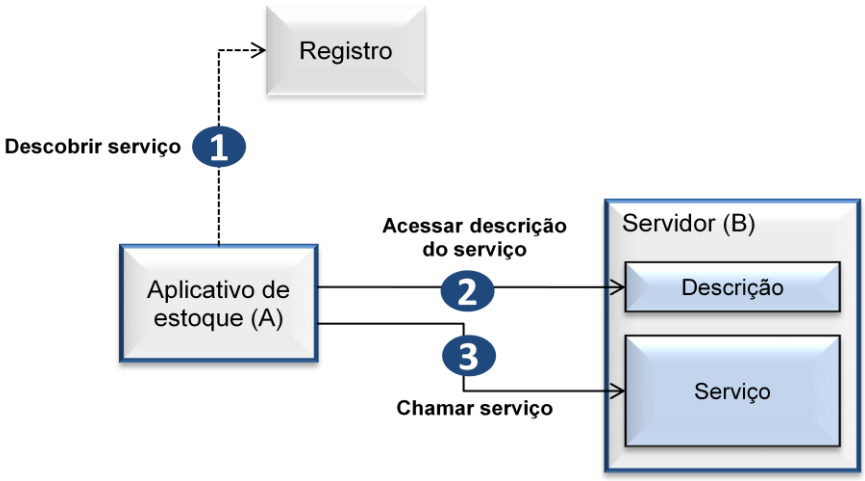

Figura 2.5. Descrição do sistema de monitoração do *status* de um pedido (CERAMI, 2002).

Arquitetura do ponto de vista da pilha de protocolos

A segunda forma de visualizar sua arquitetura é justamente examinar as múltiplas camadas que compõem a sua pilha de protocolos (CERAMI, 2002). Como mostra a Figura 2.6, existem quatro camadas:

- Localização do serviço: responsável por centralizar *web services* em um registro comum, provendo a funcionalidade de publicação e busca. Tais procedimentos são feitos via UDDI (*Universal Description, Discovery and Integration*);
- Descrição do serviço: responsável por descrever a interface pública de um *web service* específico. Atualmente, tal descrição é realizada por meio da linguagem WSDL (*Web Service Description Language*);
- Protocolo de mensagens: responsável por codificar mensagens em um formato XML padrão para que ambos os lados da transação consigam entendê-la. O protocolo SOAP (*Simple Object Access Protocol*) é utilizado nessa camada;
- Comunicação: responsável pelo transporte de mensagens entre as aplicações. São utilizados protocolos padrões de rede como o HTTP (*HyperText Transfer Protocol*), SMTP (*Simple Mail Transfer Protocol*) e FTP (*File Transfer Protocol*).

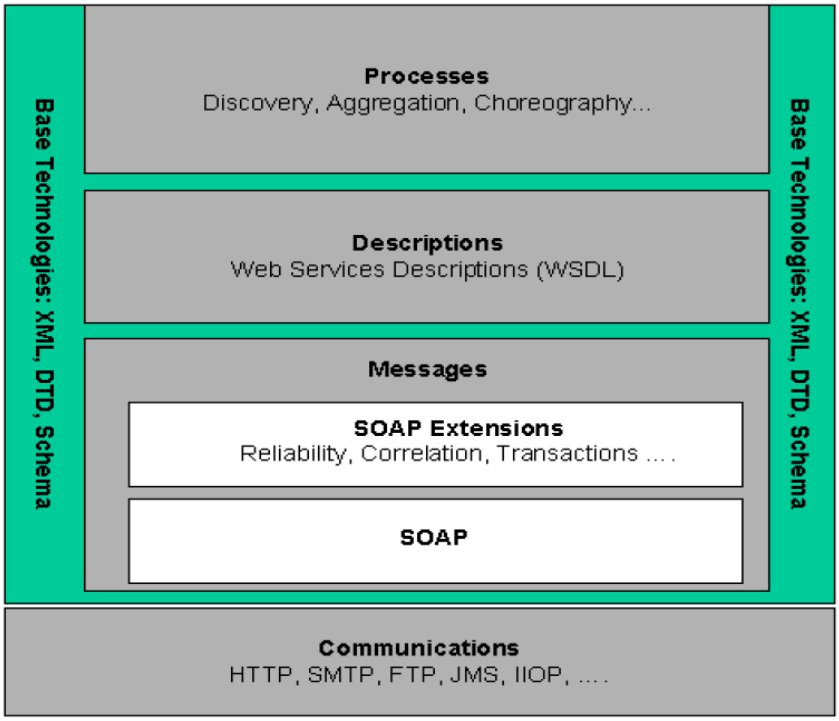

Figura 2.6. Pilha de protocolos de uma *web service* (W3C WORKING GROUP NOTE, 2004).

2.3.3. Tecnologias associadas

Web services são tipicamente construídas sobre as duas principais tecnologias apresentadas na seção anterior: WSDL e SOAP. Ambos utilizam um formato baseado em XML e assim como a sintaxe em que se baseiam, eles são especificados pela *World Wide Web Consortium* por meio de documentos de recomendação.

Antes de implementar o *web service,* seus desenvolvedores criam uma definição no formato de um documento WSDL, que descreve a sua funcionalidade e a localidade na rede. Esta informação pode então ser inserida em um registro UDDI, permitindo que clientes possam procurar na rede e encontrar os serviços de que necessitam, muito embora um *web service* não precise ser localizável, como discutido na seção 2.3.1. Com base na informação no registro UDDI, o cliente usa as instruções no documento WSDL para construir mensagens SOAP e trocar dados com o *web service* por meio de algum protocolo padrão como o HTTP (ALTOVA, 2006). As tecnologias WSDL e SOAP serão apresentadas em maior detalhe a seguir.

Web Service Description Language (WSDL)

A WSDL fornece um modelo em um formato XML especificamente para descrever *web services*. Essencialmente, a WSDL descreve quatro informações importantes sobre o *web service* (CERAMI, 2002):

- Tipos de dados utilizados por todas as mensagens de entrada e saída;
- Interface pública e todos os seus métodos disponíveis;
- Protocolo de transporte a ser utilizado e suas características;
- Endereço de rede pelo qual o *web service* pode ser acessado.

A sua especificação define uma linguagem em que é possível descrever em blocos separados tanto a funcionalidade abstrata oferecida por um serviço quanto os seus detalhes concretos, como a sua localidade e de que forma ela é oferecida. Dentro de cada bloco, a descrição utiliza alguns elementos para dividir perspectivas de projeto independentes (W3C RECOMMENDATION, 2007).

Como pode se ver na Figura 2.7, ***description*** deve ser o elemento base de qualquer documento WSDL. Ele define o nome da *web service,* declara mútiplos espaços de nomes usados ao longo do documento e contém todos os outros elementos descritos a seguir.

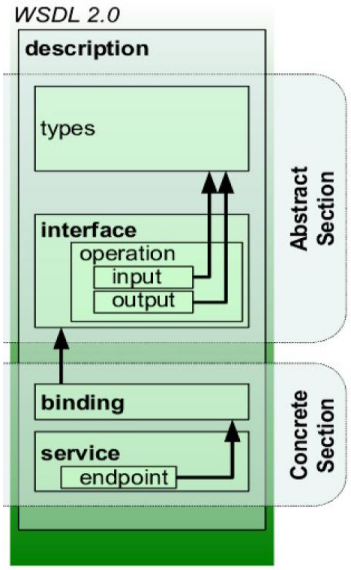

Figura 2.7. Especificação de um documento WSDL.

No nível abstrato, a WSDL descreve um *web service* em termos das mensagens que ele envia e recebe. Mensagens são descritas independentemente de um formato de transmissão específico, através do XML. Neste nível, o WSDL possui os seguintes elementos:

- ***types***: descreve todos os tipos de dados utilizados entre o cliente e o servidor;
- ***operation***: pode ser comparado à uma chamada de função em uma linguagem de programação tradicional. Está definida neste elemento a forma de codificação da mensagem associada à operação. As mensagens contém zero ou mais elementos ***types***, referenciando parâmetros ou valores de retorno;
- ***interface***: agrupa as operações que podem ser realizadas pelo *web service*.

No nível concreto da especificação, existem os seguintes elementos:

- ***binding***: especifica os detalhes quanto ao formato de transmissão via rede de uma ou mais ***interfaces***. Informações relevantes ao SOAD são definidas aqui.
- ***endpoint***: associa um endereço de rede a uma ***binding***. Ou seja, define o endereço a ser utilizado para invocar um serviço.
- ***service***: conjunto de ***endpoints***. Portanto, pode ser considerado como sendo os endereços referentes à todas as operações executáveis pelo *web service.*

Simple Object Access Protocol (SOAP)

O SOAP é um protocolo para troca de informações estruturadas em plataformas descentralizadas e distribuídas. Ela se baseia na sintaxe da XML para fundamentar a construção de mensagens que podem trafegar através de diversos protocolos de transporte. Usualmente, o SOAP negocia RPCs (*remote procedure calls*) utilizando o HTTP. O SOAP é especificado de tal forma que ele seja independente de qualquer linguagem de programação e também de quaisquer semânticas específicas à implementação (W3C RECOMMENDATION, 2007).

A especificação do SOAP define três principais partes (CERAMI, 2002):

- Envelope da mensagem: contém os dados a serem transferidos entre o cliente e o servidor. Isto inclui desde os nomes, parâmetros e valores de retorno dos métodos a serem invocados, até informações sobre quem deve processar o conteúdo do envelope e, em caso de erros, como codificar mensagens de erro;
- Regras de codificação: a transferência de dados exige que ambas as máquinas estabeleçam um padrão de regras para codificar tipos de dados definidos pelas aplicações. Portanto, O SOAP inclui suas próprias convenções para a codificação;
- Convenções RPC: para trocas de mensagens em mão dupla, o SOAP define uma convenção para representar chamadas de procedimento remoto e suas respostas. Isso permite que um cliente especifique o nome de um método remoto, inclua o número de parâmetros e receba uma resposta do servidor.

2.4. UML

A UML (*Unified Modeling Language*) é uma linguagem padrão para a elaboração da estrutura de projetos de *software*. Ela pode ser empregada para a visualização, especificação, construção e a documentação de artefatos que façam uso de sistemas complexos de *software* (BOOCH, RUMBAUGH e JACOBSON, 2005).

Os esforços para a criação de uma linguagem unificada de modelagem de sistemas de *software* começaram oficialmente em 1994, com a participação de diversas empresas parceiras e dos autores dos principais métodos de modelagem utilizados na época. Tal colaboração resultou na UML 1.0, oferecida para padronização ao OMG (*Object Management Group*), em 1997.

A OMG é a responsável pela manutenção e atualização, tendo lançado diversas revisões da UML. Em 2005, foi lançado a UML 2.0, uma importante revisão incluindo muitos recursos adicionais. Os documentos de especificação da UML 2.0 se encontram no site da OMG, em http://www.omg.org/spec/UML/2.0/.

2.4.1. Arquitetura

Visualizar, especificar, construir e documentar sistemas complexos de *software* são tarefas que requerem a visualização desses sistemas sob várias perspectivas. Diferentes participantes, entre eles os usuários finais, desenvolvedores e gerentes, trazem contribuições próprias ao projeto e observam o sistema de maneira distinta em momentos diferentes ao longo do desenvolvimento.

Portanto, a arquitetura de um sistema complexo de *software* pode ser modelada mais adequadamente por cinco visões interligadas, em que cada uma constitui uma projeção de sua organização e estrutura (BOOCH, RUMBAUGH e JACOBSON, 2005). A Figura 2.8 mostra um esquema representando as diversas perspectivas de modelagem.

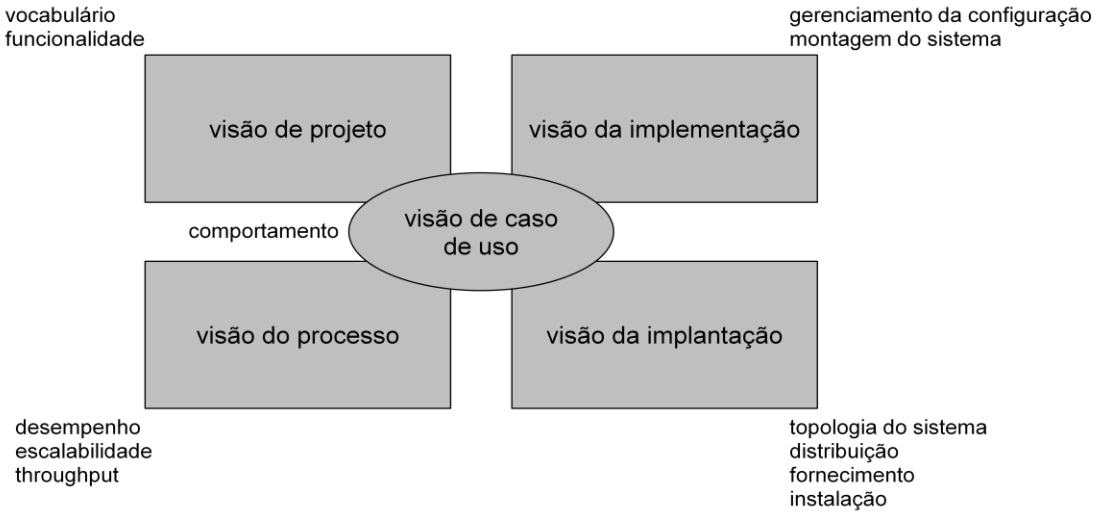

Figura 2.8. A modelagem da arquitetura de um sistema de *software* (BOOCH, RUMBAUGH e JACOBSON, 2005).

- Visão de caso de uso: abrange os casos de uso que descrevem a funcionalidade do sistema conforme é visto pelos seus clientes ou usuários finais. Essa visão especifica os requisitos a que o sistema deve atender, e que determinam a forma da arquitetura do sistema;
- Visão de projeto abrange as classes lógicas e interfaces que formam o vocabulário do sistema, assim como os

relacionamentos existentes entre elas que se traduzem no comportamento do sistema como um todo;

- Visão de processo: mostra o fluxo de controle entre as várias partes, incluindo mecanismos de concorrência e de sincronização. Essa visão cuida principalmente das questões referentes ao desempenho, à escalabilidade e ao *throughput* (taxa de transferência de dados) do sistema.
- Visão de implementação: abrange os componentes e os artefatos utilizados para a montagem e fornecimento do sistema físico. Essa visão envolve o gerenciamento das versões do sistema e diz respeito ao mapeamento de classes lógicas para arquivos e demais artefatos físicos;
- Visão de implantação: abrange os nós que formam a topologia de *hardware* em que o sistema é executado. Essa visão direciona a distribuição, o fornecimento e a instalação das partes que constituem o sistema físico.

Cada uma dessas visões envolve uma modelagem estrutural do sistema, em que são focados seus itens estáticos, assim como uma modelagem comportamental de sua dinâmica. Em conjunto, essas visões captam as decisões mais importantes sobre o sistema. Individualmente, cada uma delas permite voltar sua atenção para uma perspectiva do sistema e analisar suas decisões com clareza.

A modelagem de uma aplicação monolítica, executada em um único equipamento como é o caso deste projeto, permite focar nas visões de casos de uso, projeto e processo, em detrimento principalmente das perspectivas de implementação e implantação.

2.4.2. Diagramas

Um diagrama é uma apresentação gráfica de um conjunto de elementos, geralmente representados como um gráfico conectado de vértices (itens) e arcos (relacionamentos) (BOOCH, RUMBAUGH e JACOBSON, 2005).

Para a modelagem do sistema de *software* referente à este projeto, são utilizados essencialmente os diagramas de classes, de casos de uso e de sequência.

Antes de detalhar cada diagrama, é importante explicar os tipos de relacionamentos que existem e como eles são representados graficamente, pois eles são comuns aos diagramas da UML.

Relacionamentos

Um relacionamento é uma conexão entre itens, e é representado graficamente com tipos diferentes de linhas para diferenciar os tipos de relacionamentos. Na modelagem orientada a objetos, existem três tipos de relacionamentos especialmente importantes (BOOCH, RUMBAUGH e JACOBSON, 2005).

Uma dependência[3] é um relacionamento de utilização, determinando que um item usa as informações e serviços de outro. Ele é representado graficamente com linha tracejadas apontando o item do qual o outro depende, como é mostrado na

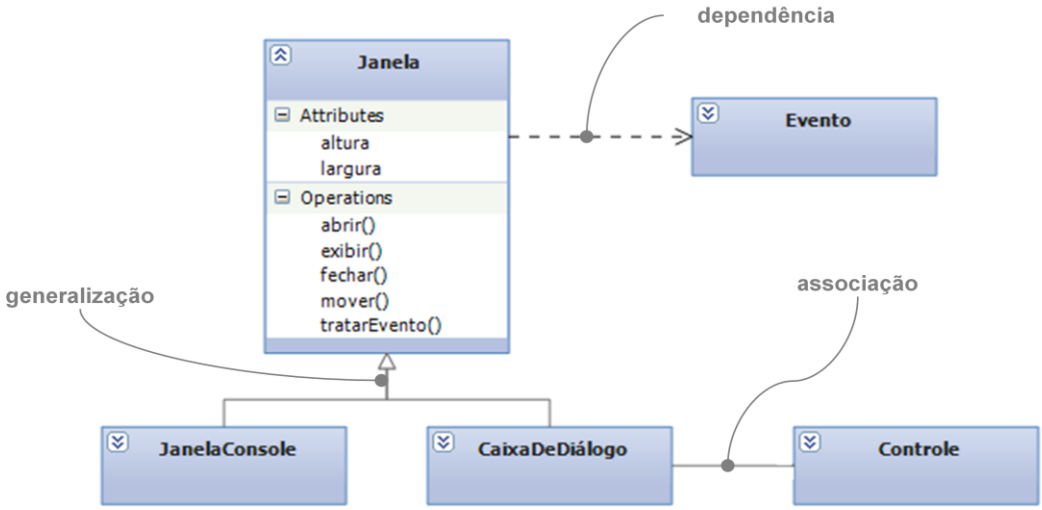

Figura 2.9 entre os itens **Janela** e **Evento**.

[3] Sublinhado: Conceitos-chave da UML e de programação orientada a objetos.

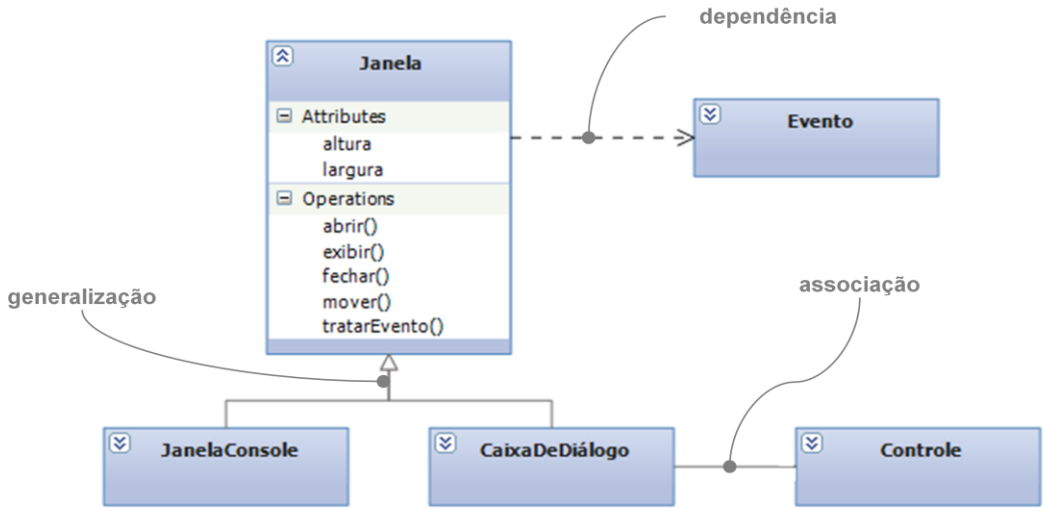

Figura 2.9. Relacionamentos.

Uma generalização é um relacionamento entre itens gerais (pai) e tipos mais específicos destes (filhos), como ocorre entre **Janela** e **CaixaDeDiálogo** na

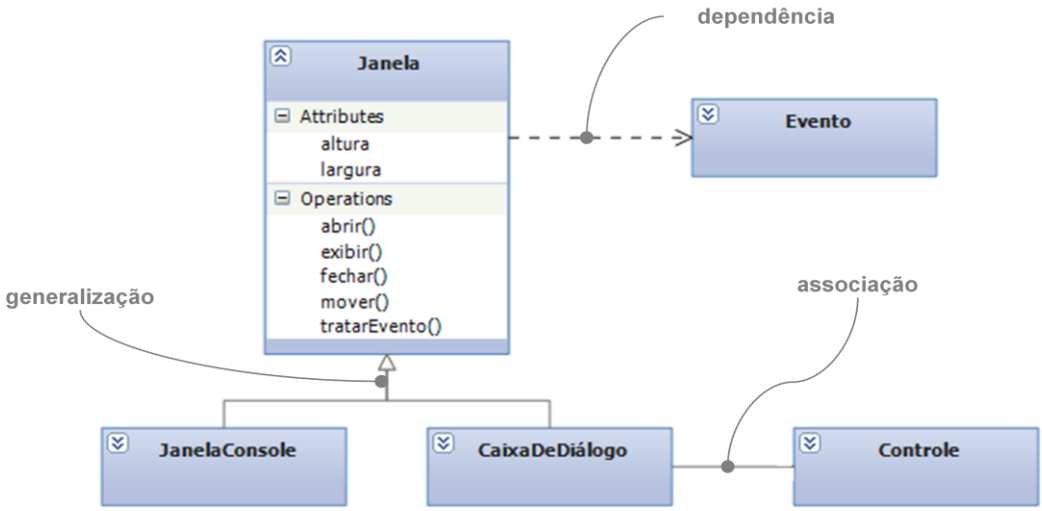

Figura 2.9. Isto significa que **CaixaDeDiálogo** herda todas as propriedades de **Janela** além de possivelmente possuir atributos e operações próprias. Este relacionamento é representado por uma linha sólida com uma grande seta triangular apontando o pai.

Finalmente, uma associação é um relacionamento estrutural que especifica objetos de um item conectados a objetos de outro item. Uma associação é representada graficamente como uma linha sólida.

Diagrama de classes

Classes são os blocos de construção mais importantes de qualquer sistema orientado a objetos. Ela é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica (BOOCH, RUMBAUGH e JACOBSON, 2005). Uma classe é representada graficamente como um retângulo, geralmente dividido em três partes que listam seu nome, seus atributos e operações (métodos), como mostra a Figura 2.10.

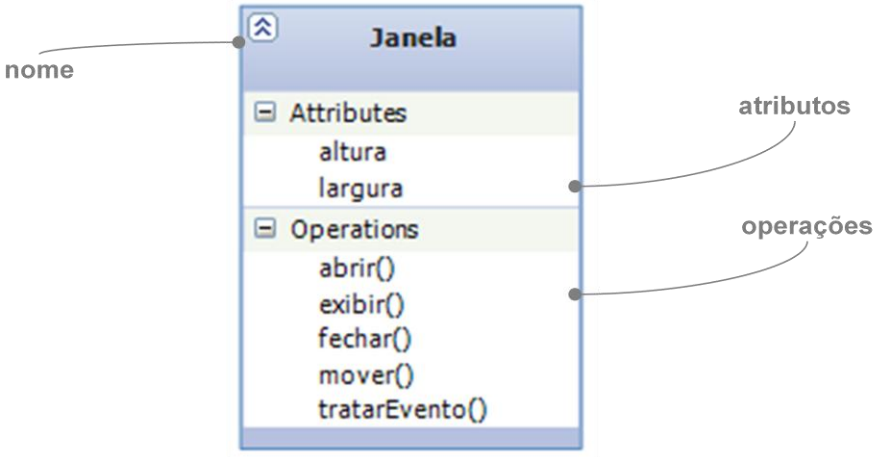

Figura 2.10. Classe: nome, atributos e operações.

Ainda, como pode ser visto na Figura 2.11, a UML fornece a possibilidade de modelagem de várias propriedades avançadas das classes, de seus atributos e de suas operações:

- tipo de dado e assinatura;
- visibilidade: elementos públicos, protegidos e privados;
- escopos de instância e estática;
- elementos abstratos e polimórficos;
- multiplicidade.

Encontrados com grande frequência na modelagem de sistemas orientados a objetos, o diagrama de classes fornece a visão estática do projeto de um sistema. O diagrama de classes contém os seguintes elementos:

- Classes;

- Relacionamentos de dependência, generalização e associação;

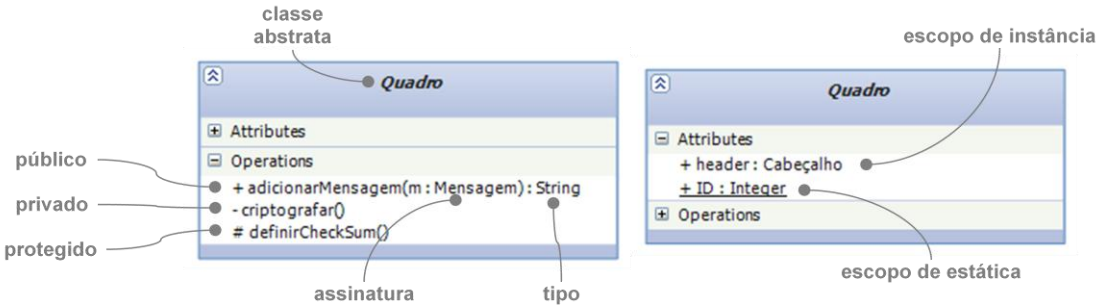

Figura 2.11. Classes avançadas.

Pode-se reconhecer esses elementos na Figura 2.12, que representa a organização de uma empresa por meio de um diagrama de classes.

Diagrama de casos de uso

Um caso de uso é uma descrição de sequências de ações que um sistema executa para produzir um resultado de valor observável por um ator (BOOCH, RUMBAUGH e JACOBSON, 2005).

Os diagramas de casos de uso têm um papel central para a modelagem do comportamento de um sistema, ou mesmo de uma classe. Eles fazem com que o sistema representado se torne acessível e compreensível, ao exibir graficamente os seus requisitos do ponto de vista de um ator externo e permitir a visualização da fronteira do sistema em termos dos comportamentos que fazem parte dele. Os diagramas de casos de uso contêm os seguintes elementos:

- Assunto;
- Casos de uso;
- Atores;
- Relacionamentos de dependência, generalização e associação;

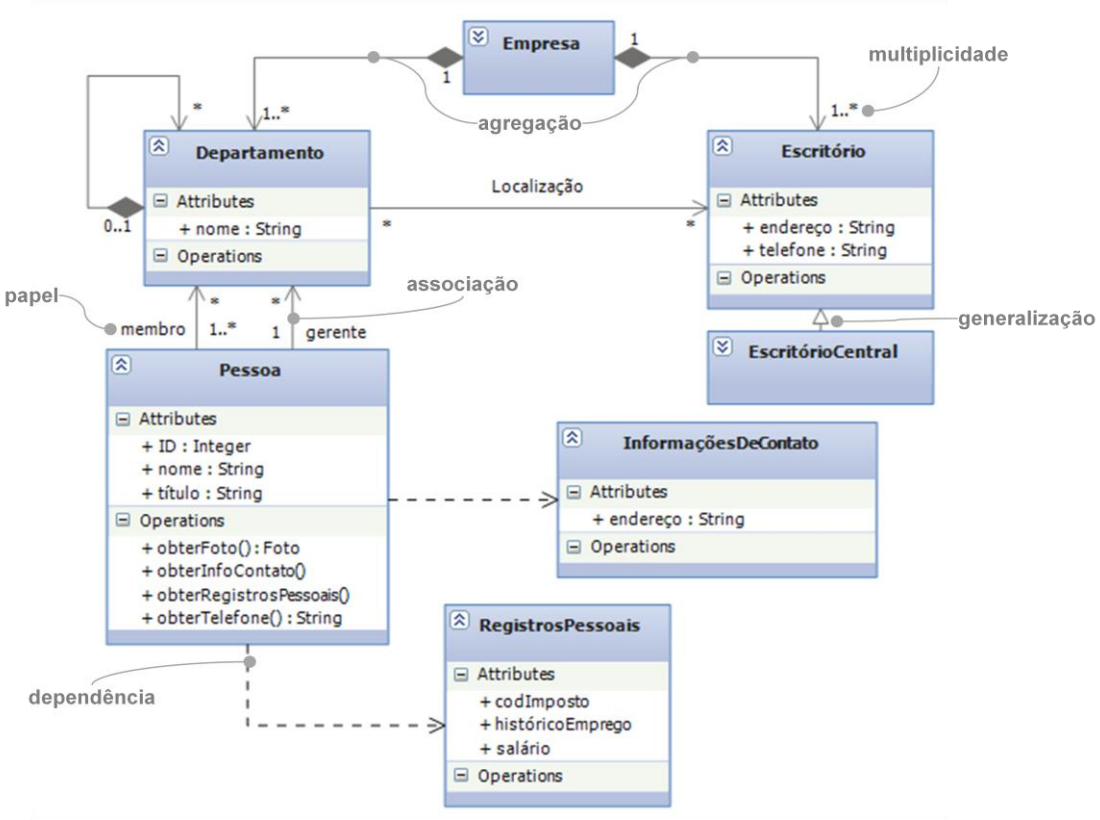

Figura 2.12. Um diagrama de classes.

Um caso de uso é representado graficamente por meio de uma elipse, como mostra a Figura 2.13. Os atores representam o papel que um humano, um dispositivo de *hardware* ou até mesmo outro *software* desempenham com o sistema em questão e são esquematizados por figuras de palito.

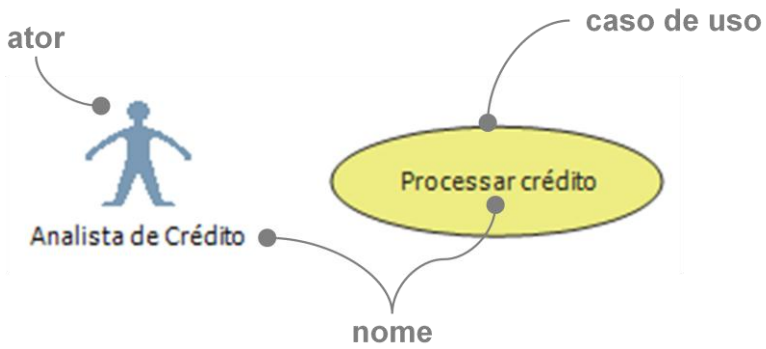

Figura 2.13. Atores e casos de uso.

Os atores poderão estar conectados aos casos de uso somente pela associação. Isto indica que o ator e o caso de uso se comunicam entre si, com a possibilidade de enviar e receber mensagens.

Dois casos de uso também podem se relacionar por uma especificação de generalização, inclusão ou extensão existente entre eles. Esses relacionamentos são mostrados na Figura 2.14.

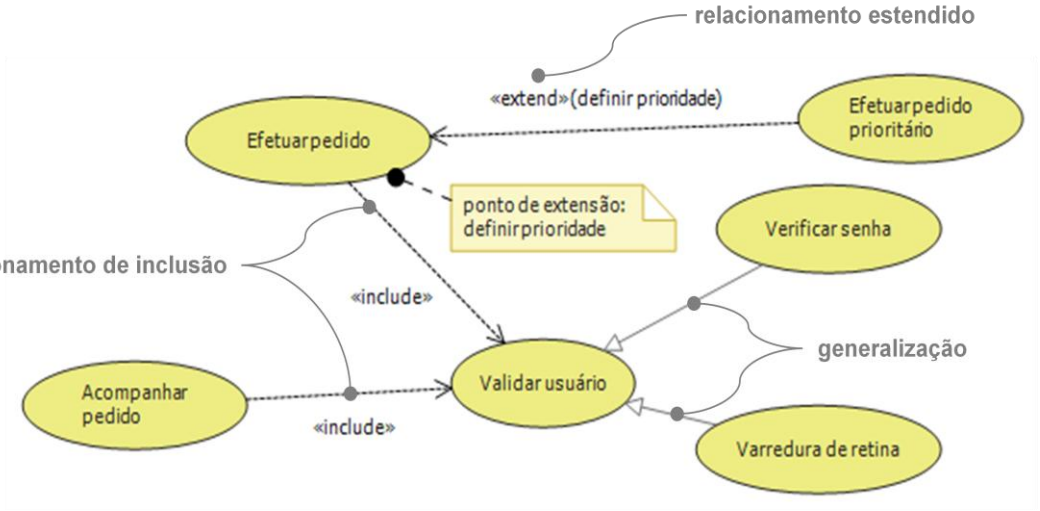

Figura 2.14. Generalização, inclusão e extensão.

Um relacionamento de inclusão significa que o caso de uso base agrega o comportamento do caso de uso incluído. O caso de uso incluído nunca permanece isolado, sendo apenas instanciado como parte de alguma base maior. Assim, é possível evitar descrever o mesmo fluxo de eventos várias vezes ao incluir o comportamento comum entre vários casos de uso em um caso de uso próprio (o caso de uso base).

Por sua vez, um relacionamento de extensão significa que o caso de uso base agrega o comportamento de outro caso de uso em somente em um local especificado indiretamente pelo caso de uso estendido, denominado ponto de extensão. Este relacionamento pode ser empregado para modelar comportamentos opcionais do sistema ou subfluxos executados somente sob determinadas condições.

A inclusão e a extensão podem ser representadas como um relacionamento de dependência, estereotipados como <<*include*>> e <<*extend*>>.

Finalmente, o assunto representa a fronteira do sistema a ser descrito pelo diagrama e é exibido como um retângulo, contendo o conjunto de elipses de casos de uso e seus relacionamentos. Na Figura 2.15, é mostrado um

exemplo de diagrama de caso de uso descrevendo o comportamento de um telefone celular.

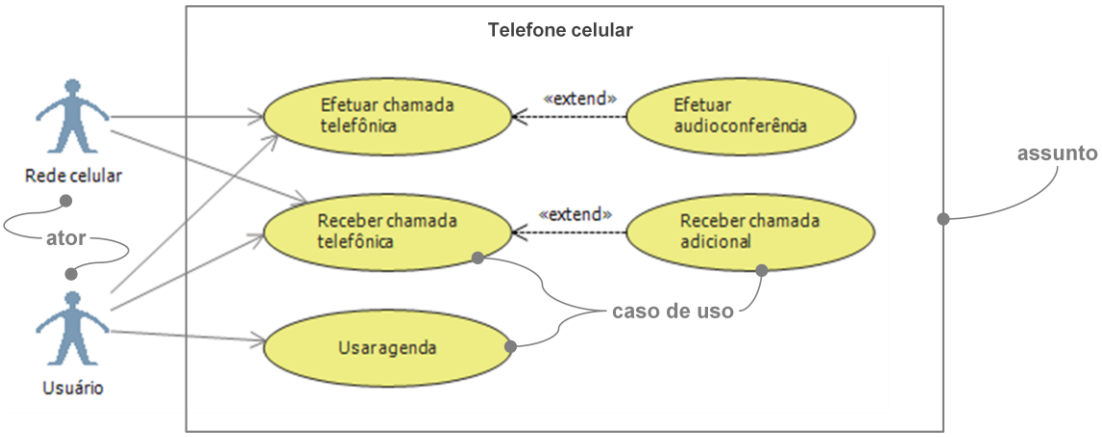

Figura 2.15. Um diagrama de caso de uso.

Diagrama de sequência

Um diagrama de sequência mostra uma interação, formada por um conjunto de objetos e seus relacionamentos, dando ênfase à ordenação temporal das mensagens trocadas entre tais objetos (BOOCH, RUMBAUGH e JACOBSON, 2005). Diagramas de sequência podem ser utilizados para fazer a modelagem de um determinado fluxo de controle de um caso de uso. O diagrama de sequência contém os seguintes elementos:

- Papéis ou objetos;
- Mensagens;

Conforme mostra a Figura 2.16, um diagrama de sequência é formado colocando-se primeiro todos os objetos que participam da interação no nível superior do diagrama, ao longo do eixo X. Tipicamente, o objeto que inicia a interação é colocado mais à esquerda e objetos mais subordinados vão sendo colocados à sua direita. A seguir, as mensagens que esses objetos enviam e recebem são colocadas ao longo do eixo Y, em ordem crescente de tempo, de cima para baixo. Isso proporciona ao leitor uma clara indicação visual do fluxo de controle ao longo do tempo.

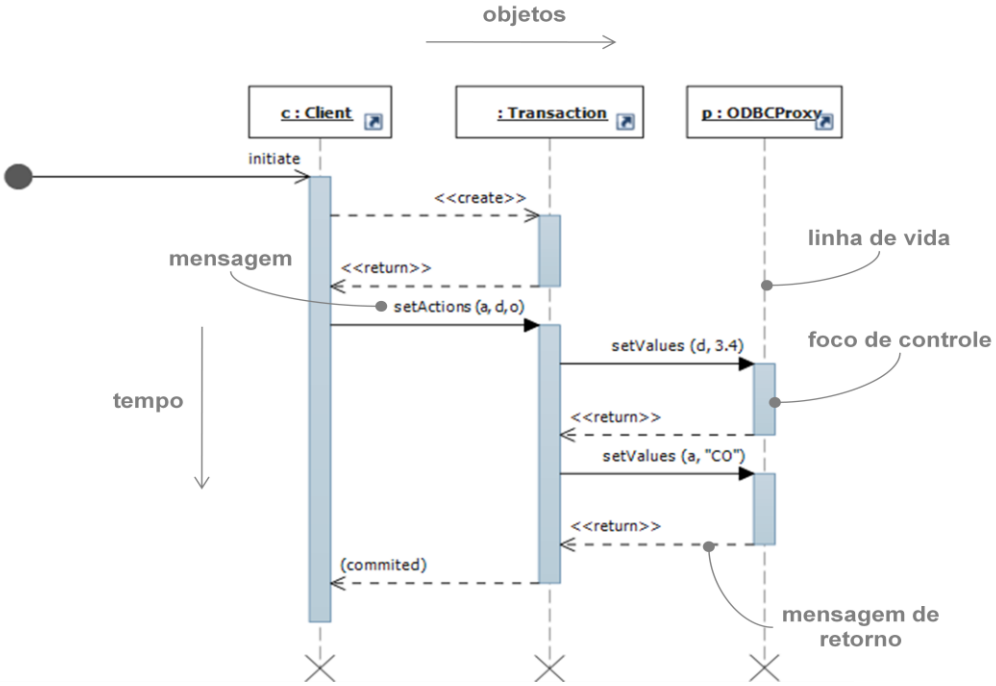

Figura 2.16. Um diagrama de sequência.

A linha tracejada vertical, chamada linha de vida, representa a existência do objeto durante um período de tempo. A existência de alguns objetos, como *c:Client* e *p:ODBCProxy* no exemplo, será igual à duração da interação. Outros, como o *:Transaction*, poderão ser criados e destruídos no decorrer da interação. Nesses casos, suas linhas de vida se iniciam com a mensagem estereotipada como <<*create*>> e são terminadas com a mensagem estereotipada como <<*destroy*>>.

O principal conteúdo em um diagrama de sequência é o conjunto de mensagens. Uma mensagem é representada graficamente por uma seta que vai de uma linha de vida para outra, sempre apontando para o destinatário. O retângulo alto e estreito que aparece sobre a linha de vida, geralmente iniciado com uma mensagem é o foco de controle. Ele mostra o período durante o qual um objeto está desempenhando uma ação.

Um diagrama de sequência pode mostrar fluxos de controle condicionais e iterativos (*loops*) por meio de operadores de controle. Um operador de controle é apresentado como uma região retangular no diagrama de sequência. Ela possui um rótulo de texto no canto superior esquerdo para informar o tipo do operador. Para operadores condicionais e iterativos, os rótulos são **Se/Senão** e **Laço**, respectivamente.

3. Detalhamento do projeto

O *software* em questão possui uma interface gráfica para acompanhamento da operação de usinagem, bem como uma área de edição de texto, para alterar e criar programas G, com um verificador de sintaxe. A simulação é acompanhada por meio de uma área de visualização, que mostra o percurso da ferramenta durante a usinagem.

O *web service* permite acesso remoto às funcionalidades do programa, como verificação de sintaxe, acompanhamento da operação de usinagem e envio de comandos à máquina.

O usuário pode interagir com o sistema através do *web service* ou por meio da própria interface gráfica do programa.

A máquina virtual simula o funcionamento de uma máquina CNC. Este comportamento se traduz em alguns parâmetros específicos do modo de operar do equipamento, como por exemplo o tempo gasto para o referenciamento dos eixos de coordenadas, o tempo para a retirada da peça fabricada, a velocidade máxima dos motores e a resolução do sistema de movimentação.

A Figura 3.1 mostra como se relacionam os componentes do sistema.

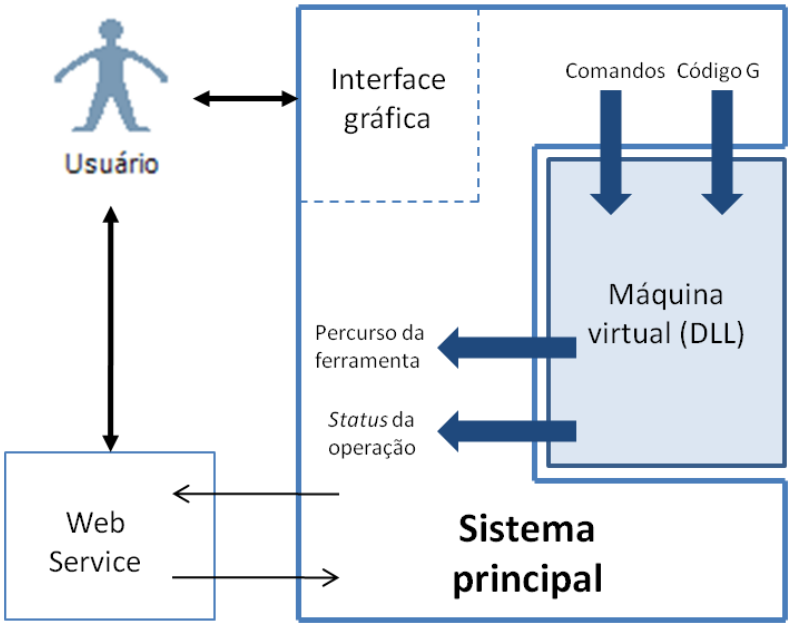

Figura 3.1. Esquema do projeto.

A utilização de uma máquina virtual no sistema se deve ao fato da máquina CNC real ser muito cara. Contudo, é possível a adaptação do *software* para a comunicação com um equipamento real, já que a máquina virtual procura simular exatamente seu comportamento, inclusive em relação à troca de informações. Esta adaptação é ilustrada na Figura 3.2.

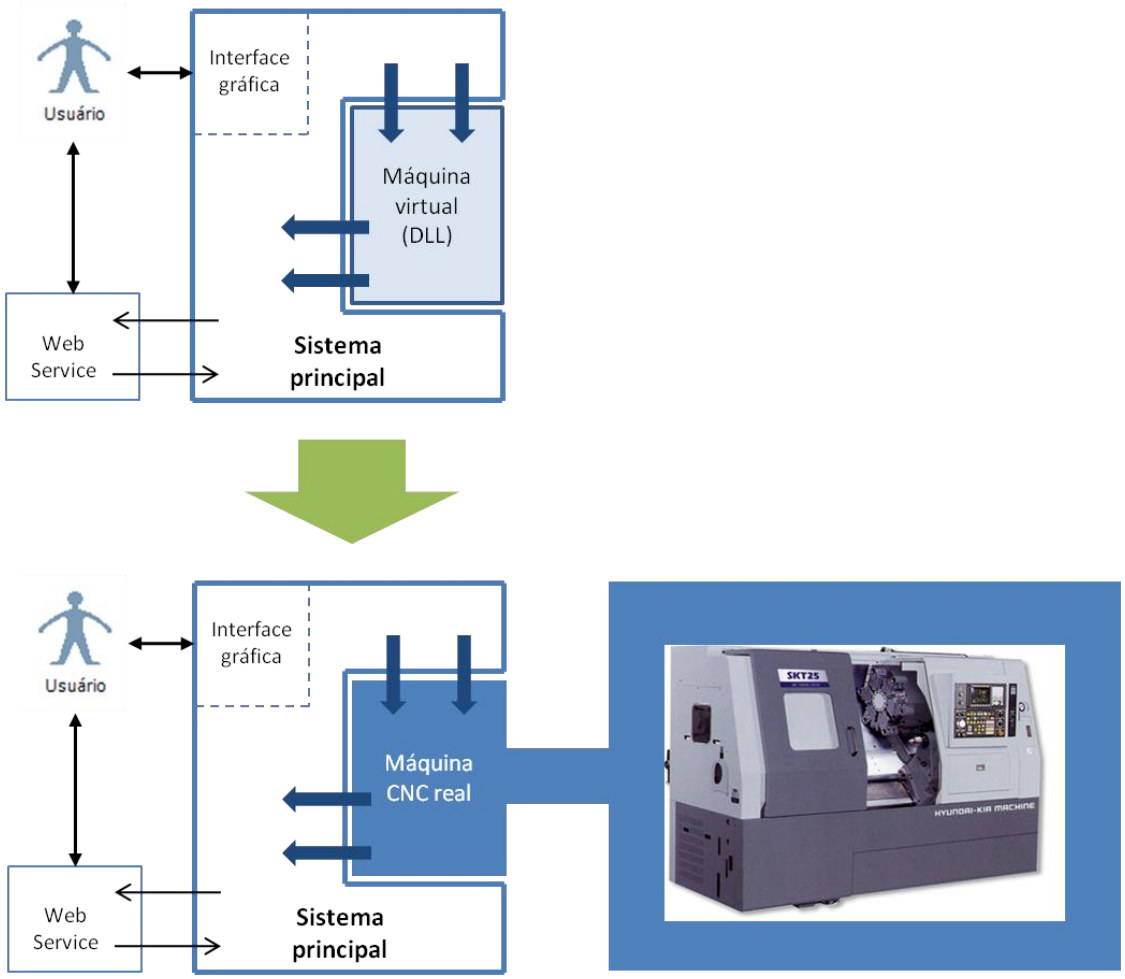

Figura 3.2. Substituição da máquina virtual pela real.

3.1. Arquitetura do *software*

Para a descrição detalhada da estrutura do projeto, será utilizada a linguagem mais aceita e empregada atualmente para este fim, a UML (*Unified modeling language*), por meio do Microsoft Visual Studio 2010. Em primeiro lugar, são ilustradas quais as possíveis ações do usuário do sistema (diagrama de casos de uso). Em seguida, como as entidades se relacionam e quais são

seus membros (diagrama de classes), e por fim é apresentado o processo detalhado de quatro atividades (diagramas de sequência).

3.1.1. Diagrama de casos de uso

Na Figura 3.3 estão representadas as principais ações que o usuário pode realizar.

Figura 3.3. Diagrama de Casos de Uso.

- **Verificar sintaxe:** O usuário pode verificar um código G para encontrar erros e então tentar corrigi-los. Estes erros podem ser códigos G inválidos, parâmetros insuficientes ou inconsistências no programa em geral;

- **Carregar código G:** Para submeter o programa, o usuário carrega o código G na memória da máquina;
- **Executar programa:** O usuário pode ordenar o início da operação de usinagem apenas se já tiver carregado o código na máquina. Neste momento, o sistema faz uma última verificação da sintaxe, para garantir que a operação não apresentará erros de programação;
- **Parar execução:** O usuário ordena a interrupção imediata da usinagem. Esta parada pode ser definitiva ou apenas uma pausa;
- **Acompanhar operação:** Enquanto a máquina CNC está trabalhando, a tela do programa exibe o percurso da ferramenta, possibilitando um monitoramento do processo;
- **Checar *status* da máquina:** Em qualquer momento, é possível checar se a máquina está usinando, livre, em pausa, retirando a peça ou se preparando para fabricar uma nova peça;
- **Abrir arquivo[4]:** Um arquivo de texto com um código G pode ser aberto na tela, editado e posteriormente submetido à máquina para ser executado;
- **Salvar arquivo[4]:** O código G pode ser editado pelo usuário e depois gravado no computador como um arquivo de texto.

3.1.2. Diagrama de classes

A seguir, está apresentado o diagrama de classes do *software* desenvolvido, onde é possível identificar que tipos de ação cada classe pode realizar e que tipo de relação há entre as entidades do sistema (Figura 3.4).

[4] Operação apenas local, não envolvendo a máquina virtual nem o *web service*.

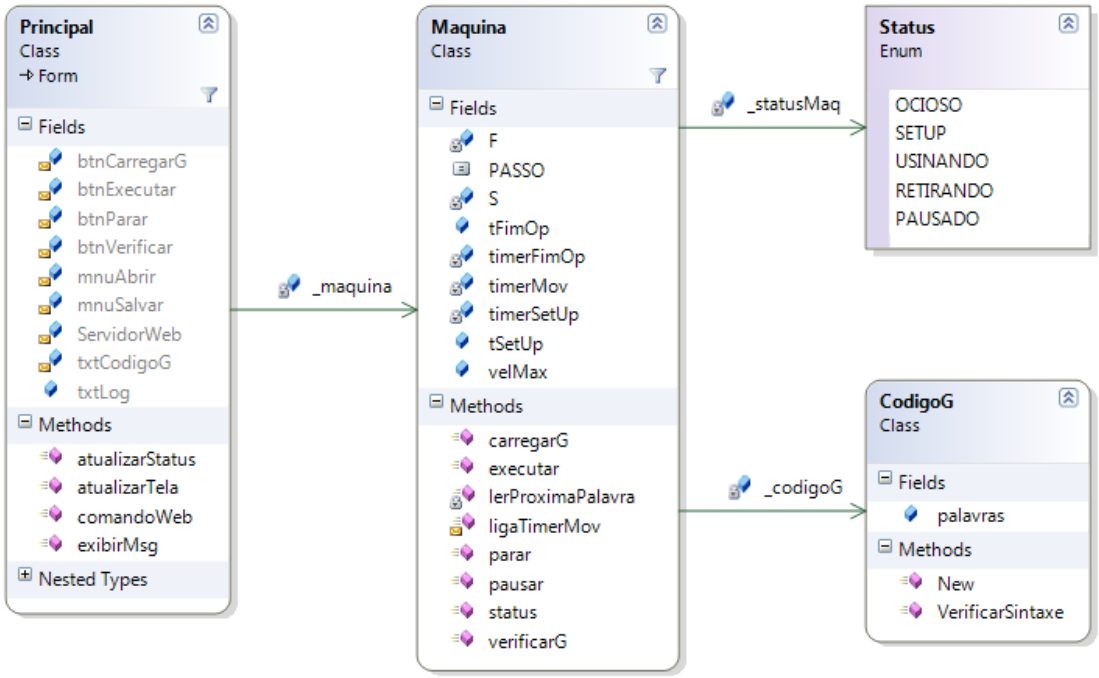

Figura 3.4. Diagrama de Classes.

Principal

Esta classe é a interface com o usuário, que contém os botões, caixas de texto e outros componentes. Seus principais membros são:

- maquina[5]: objeto da classe Maquina, que corresponde à máquina virtual que está sendo monitorada. Esta instância é criada na inicialização do sistema;
- txtCodigoG[6]: caixa de texto presente na tela do *software*, na qual pode ser escrito um programa G;
- txtLog: área de exibição das mensagens do sistema ao usuário;
- ServidorWeb: objeto responsável pelo recebimento dos comandos via web, a partir da definição de um endereço IP e de uma porta;
- btnCarregarG: botão utilizado para submeter o código à máquina, para ser executado;
- btnExecutar, btnParar: comandos para o controle da operação da máquina CNC;

[5] Fonte diferenciada: componentes, operações e nomes de classes.
[6] O prefixo txt nos componentes indica que se trata de uma caixa de texto, enquanto btn indica que é um botão e mnu é usado para itens da barra de menu.

- btnVerificar: verificação do código contido no txtCodigoG;
- mnuAbrir, mnuSalvar: operações comuns de arquivos.

As principais operações que compõem esta classe são:

- atualizarStatus: atualiza o indicador de *status* da máquina na tela do programa;
- atualizarTela: função para exibição do percurso e das coordenadas da ferramenta, em milímetros;
- comandoWeb: interpreta e executa o comando recebido pelo ServidorWeb e, em seguida, envia uma resposta ao usuário remoto;
- exibirMsg: exibe uma mensagem no txtLog. Utilizada pela máquina para comunicar ao usuário qualquer informação relevante.

Maquina

Esta classe representa a máquina virtual e possui os parâmetros necessários para a simulação. Seus principais atributos são:

- codigoG: objeto da classe CodigoG que é utilizado na operação;
- F, S: armazenam a velocidade de avanço e rotação da ferramenta, respectivamente;
- PASSO: constante que armazena a resolução do equipamento, isto é, a menor variação de posição possível;
- statusMaq: informação sobre o *status* da máquina; pode adotar qualquer valor da enumeração Status;
- tFimOp: tempo de retirada da peça da máquina. Inclui o tempo de parada completa do equipamento e de abertura da comporta;
- tSetUp: tempo necessário para a colocação da peça e o referenciamento do sistema de coordenadas da máquina;
- timerFimOp, timerSetUp: contadores de tempo responsáveis por simular a retirada da peça e o *set up* da máquina, respectivamente;

- timerMov: responsável pelo controle da movimentação da ferramenta, parametrizado pela velocidade de avanço F. Além disso, chama a função atualizarTela constantemente, para exibir o trajeto;
- velMax: velocidade máxima dos atuadores do equipamento. Um comando de avanço rápido do código G resulta em um movimento com esta velocidade.

As operações desta classe são:

- executar, pausar, parar: executam os comandos da classe Principal, de início, pausa e fim de operação;
- carregarG: cria uma instância para o campo codigoG a partir de um texto;
- verificarG: responsável por chamar a função de verificação de sintaxe quando a operação de mesmo nome na classe Principal é acionada;
- lerProximaPalavra: responsável pela interpretação do codigoG, percorrendo os comandos e executando as tarefas apropriadas. Caso tenha lido um comando de movimento, aciona a função ligaTimerMov;
- ligaTimerMov: calcula a direção e a velocidade do movimento da ferramenta, e em seguida ativa o timerMov;
- status: utilizada pelas outras classes para a checagem do *status* da máquina.

CodigoG

Esta classe representa o programa G em si, criado a partir de um texto. Possui o campo palavras, que armazena todos os comandos do código, e as funções New, que cria uma instância da classe CodigoG, e VerificarSintaxe, que localiza erros de programação no código G e é utilizado pela função verificarG da máquina.

Status

Representa uma enumeração dos possíveis *status* da máquina, que são:

- OCIOSO: significa que a máquina está ociosa, livre para realizar alguma tarefa;
- PAUSADO: a máquina está pausada no meio de uma operação de usinagem;
- RETIRANDO: situação de fim de operação, na qual a peça fabricada é retirada da máquina;
- SETUP: preparação para início de operação, que envolve a colocação precisa da peça na máquina e o referenciamento dos eixos;
- USINANDO: a máquina está realizando um processo de usinagem, ou seja, está ocupada.

3.1.3. Diagramas de sequência

Os casos de uso podem ter nomes ou descrições muito genéricas. Para isso, empregam-se os diagramas de sequência, que explicam de forma mais detalhada o processo do caso de uso.

Os processos que envolvem diretamente a máquina e o programa principal estão detalhados nas Figura 3.5, Figura 3.6, Figura 3.7, e Figura 3.8. Eles mostram a verificação, o carregamento, a execução do código G e a parada da operação, respectivamente.

O caso de uso “Acompanhar operação” está atrelado à execução do programa (Figura 3.7).

Verificar código G (Figura 3.5)

Para a verificação do código G, o usuário deve fazer uso da função verificarG, que é acionada por um botão na tela do programa. Esta função chama outra de mesmo nome, pertencente ao objeto maquina, que instancia um objeto CodigoG a partir do texto recebido e verifica sua sintaxe com a função VerificarSintaxe.

A lista de erros retorna à máquina e em seguida à classe Principal. Os erros são informados na tela.

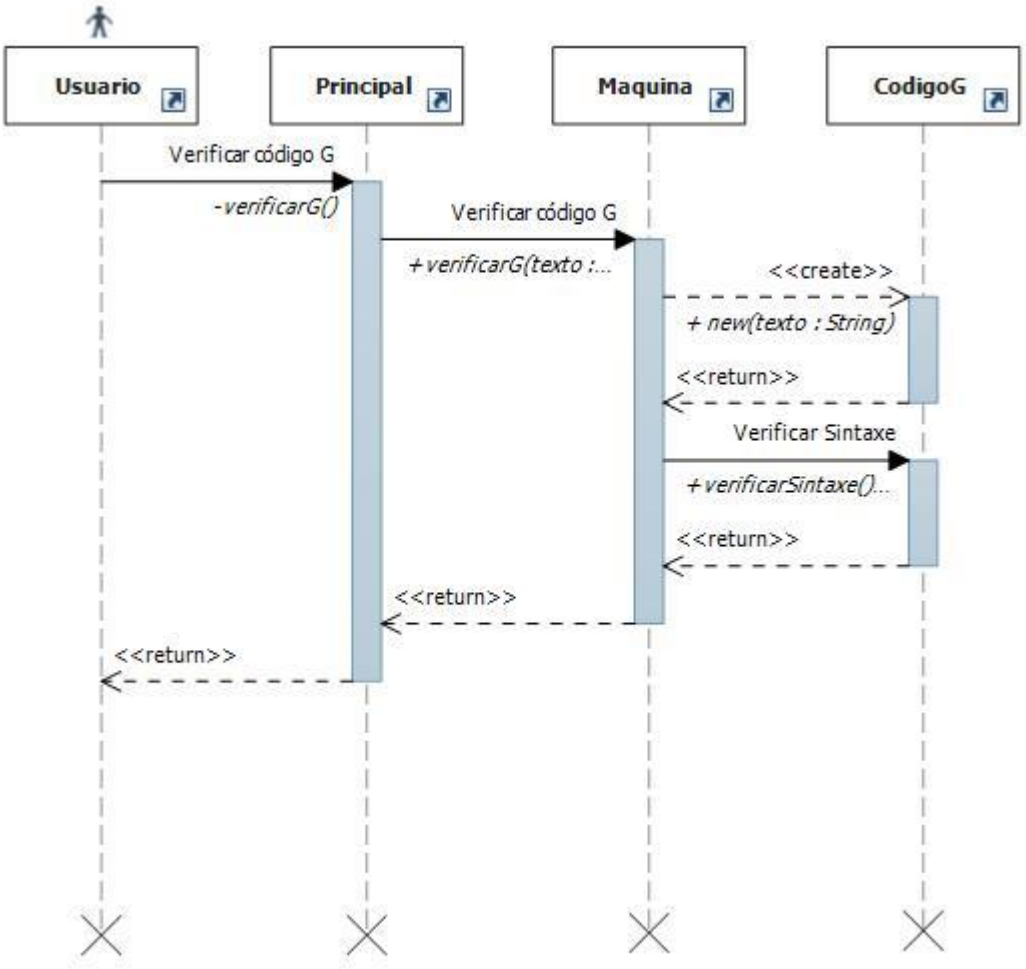

Figura 3.5. Diagrama de sequência - Verificar código G.

Carregar programa G (Figura 3.6)

Quando o usuário clica no botão btnCarregarG, o conteúdo do txtCodigoG é enviado para a máquina, que cria uma instância da classe CodigoG e a armazena no campo correspondente na máquina.

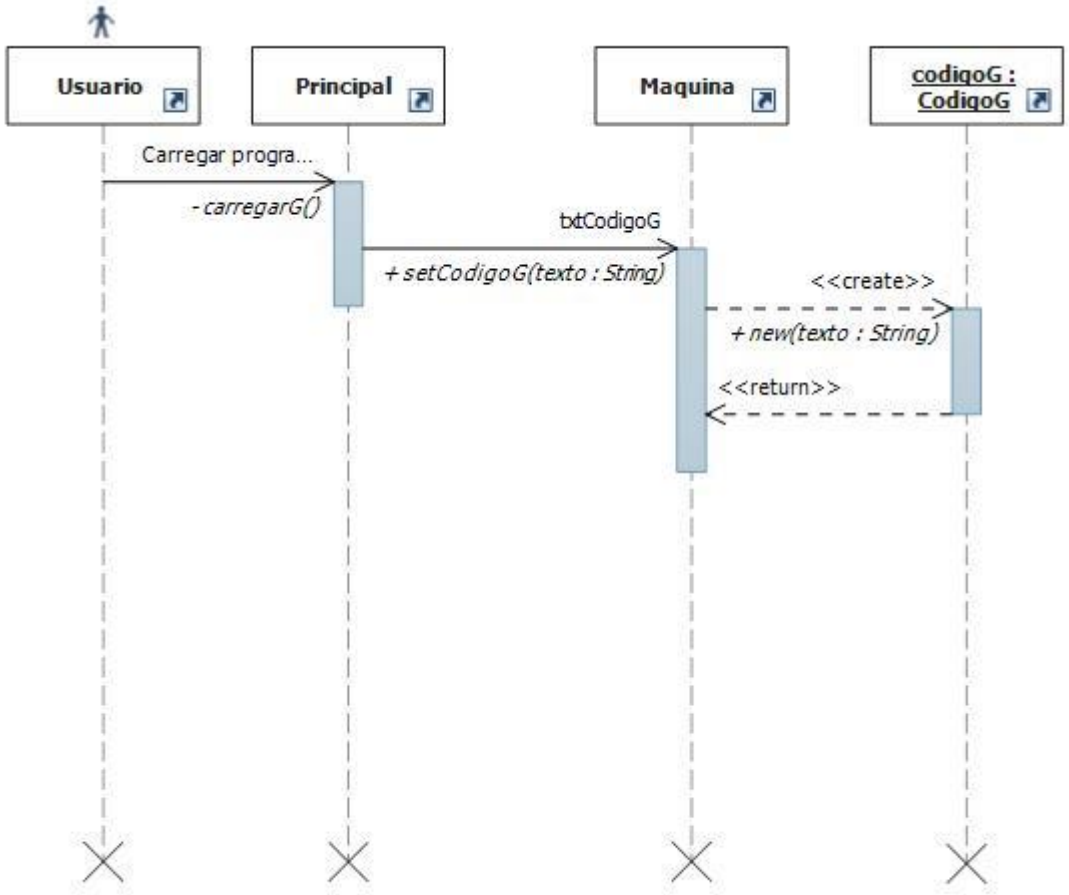

Figura 3.6. Diagrama de sequência - Carregar programa G.

Executar código G (Figura 3.7)

O processo de execução do código G começa com uma verificação da sintaxe, para garantir que não haverá erros de programação. Caso não sejam encontrados erros, o controle de operação começa a atuar, e o usuário é informado do início da operação. Caso contrário, uma mensagem comunicando a existência de erros é exibida ao usuário.

Enquanto a máquina está usinando a peça, a tela do programa principal é constantemente atualizada. Ao final da operação, o programa exibe uma mensagem indicando o fim da execução do programa G.

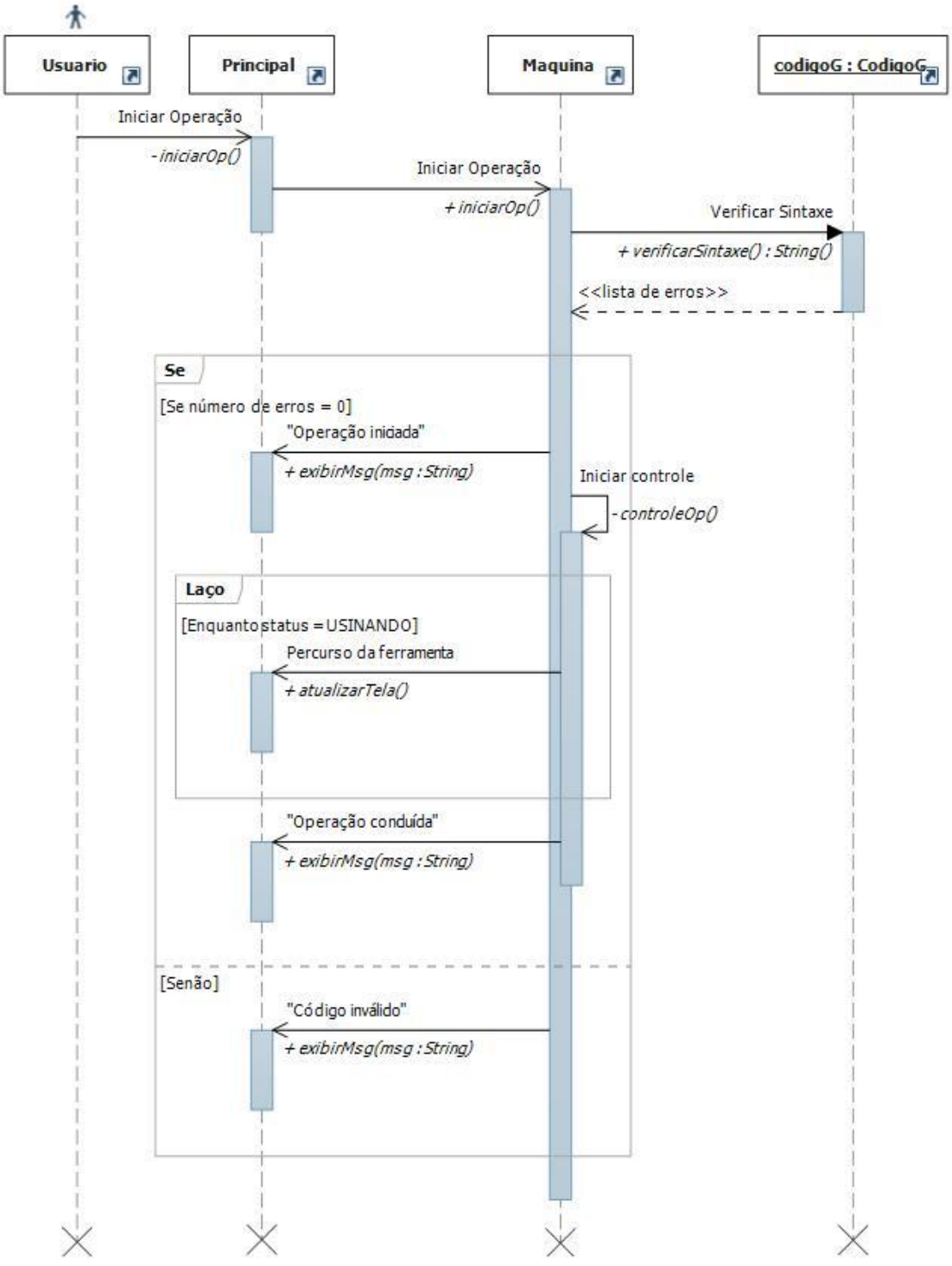

Figura 3.7. Diagrama de sequência – Executar código G.

Parar operação (Figura 3.8)

Para interromper a operação, o usuário aciona a função pararOp, que chama a função de mesmo nome na classe Maquina. Esta função altera o

status da máquina para OCIOSO, fazendo com que o controle da operação pare de trabalhar.

Figura 3.8. Diagrama de sequência – Parar operação.

4. Descrição do programa

4.1. Interface

O programa desenvolvido possui uma interface gráfica com botões, caixas de texto e uma ampla área de visualização, que permite a operação do *software* e a leitura de informações. A aparência da tela é mostrada na Figura 4.1.

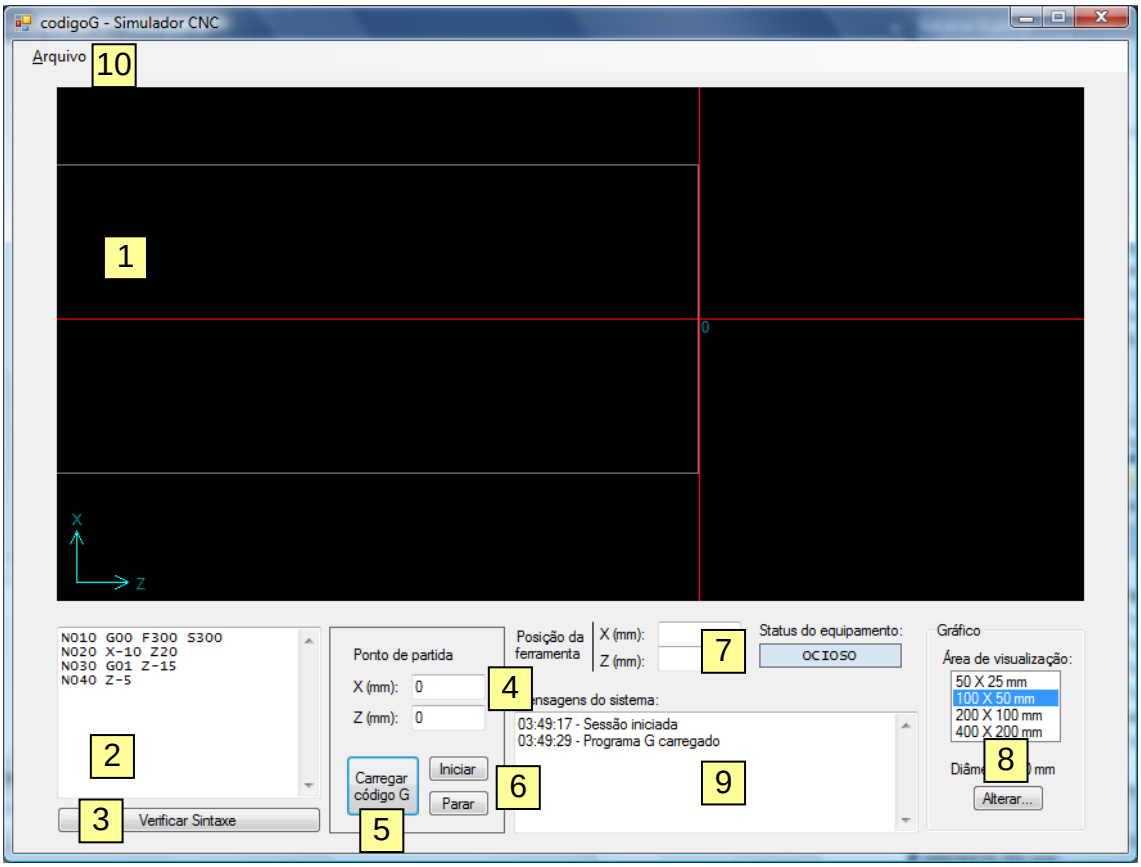

Figura 4.1. Interface gráfica do simulador.

1. **Área de visualização:** Esta é a região onde será descrita a trajetória da ferramenta. Podem-se ver os eixos X e Z (vermelho) e o contorno da peça a ser usinada (cinza).
2. **Área de edição do código G:** Nesta caixa de texto é possível escrever ou editar um programa G. Quando o usuário abre um arquivo de texto com um programa, ele é carregado neste campo.

3. **Botão Verificar Sintaxe:** Este botão chama a função verificarG. O código escrito na área de edição é submetido a uma verificação, a fim de identificar possíveis comandos inválidos ou informações faltantes.
4. **Ponto de partida:** Neste campo deve ser informada a posição de partida da ferramenta. A execução do código G entenderá que a ferramenta se encontrava inicialmente nesta posição.
5. **Botão Carregar código G:** Este botão é utilizado para submeter o código G à máquina CNC virtual, para que ele seja executado.
6. **Botões Iniciar / Parar:** Estes botões controlam a operação da máquina, ordenando o início da execução, a pausa, a retomada e o fim da operação.
7. **Posição da ferramenta:** Permite acompanhar em tempo real a posição X e Z da ferramenta, em milímetros, ao longo da usinagem. Esta posição é descrita em relação à origem dos eixos, localizada na extremidade da peça a ser usinada.
8. **Painel de ajuste da área de visualização:** Este painel apresenta quatro opções de tamanho da área de visualização. Tamanhos menores significam maior aproximação, logo possibilitam uma maior resolução e precisão do movimento da máquina. Entretanto, se a operação tiver um percurso muito longo, devem-se usar áreas maiores. Além disto, neste painel é possível alterar o diâmetro do cilindro representado na tela.
9. **Área de mensagens do sistema:** Exibe as mensagens que o programa comunica ao usuário, os erros do código G e a hora, minuto e segundo da ocorrência da mensagem.
10. **Menu Arquivo:** Neste menu constam as operações típicas de arquivo, ou seja, abrir e salvar. Um programa G que for aberto será mostrado na área de edição, para eventuais alterações e para ser carregado na máquina.

4.2. Principais operações

4.2.1. Verificação de sintaxe

A verificação do código G começa com o usuário clicando no botão Verificar Sintaxe. Com isso, o código escrito na área de edição é separado em linhas, e um novo objeto da classe CodigoG é criado a partir das linhas de código G da área de edição.

Em seguida, chama-se a função verificarSintaxe deste objeto, que percorre o código, de palavra[7] em palavra, verificando se há algum comando não reconhecido. Esta função retorna uma lista de erros, sendo que cada item da lista indica a linha e o tipo do erro encontrado.

Os comandos que o *software* em questão foi projetado para interpretar são:

- N###: Apenas numeração da linha do programa;
- S###: Determinação da rotação do eixo árvore;
- F###: Determinação da velocidade de avanço da ferramenta;
- G00: Posicionamento rápido (velocidade máxima do equipamento);
- G01: Interpolação linear (velocidade estabelecida pelo comando F);
- G90: Modo absoluto de interpretação das coordenadas;
- G91: Modo incremental de interpretação das coordenadas;
- X###: Coordenada no eixo X;
- Z###: Coordenada no eixo Z;
- M02: Fim do programa;
- M03: Liga eixo árvore da máquina no sentido horário.

Qualquer comando diferente destes é interpretado como inválido.

4.2.2. Execução do programa

Primeiramente carrega-se o código G na máquina, por meio do botão “Carregar código G”. Desta forma, ao clicar em “Iniciar”, a máquina faz uma

[7] Palavra, no contexto do código G, se refere a cada conjunto de uma letra e um número, ou seja, um comando. Exemplos: “F650”, “Z40.5”

verificação no código, e se não houver erros o *status* é atualizado para USINANDO, e a sub-rotina lerProximaPalavra é chamada.

Este método percorre o código, palavra por palavra, armazena as informações, e quando chega ao final da linha, ordena a movimentação da máquina de acordo com os comandos lidos, chamando o método ligaTimerMov. Este método implementa o algoritmo de Bresenham para a interpolação linear entre o ponto inicial e o final do movimento, como explicado por Flanagan (1996).

A movimentação é simulada por meio de um *timer* que dispara um evento repetidamente a cada intervalo de tempo. Este intervalo é uma propriedade do *timer* e é medido em milissegundos. O controle de velocidade da ferramenta é feito alterando-se o valor desta propriedade, ou seja, para velocidades maiores, intervalos menores, e vice-versa.

A cada intervalo de tempo, a máquina simula um pequeno avanço em direção ao ponto de destino, informado pelo código G, e atualiza o desenho do percurso da ferramenta na interface gráfica. Ao término da interpolação linear, a ferramenta terá percorrido a distância entre o ponto inicial e final a uma velocidade média F, dada em milímetros por segundo, que foi armazenada na máquina a partir da informação contida no código G.

4.3. Procedimento para operação local

Na situação em que o usuário do sistema opera diretamente o programa principal, isto é, localmente, será apresentada a ele a interface gráfica da Figura 4.1 quando o *software* for aberto.

Para uma operação típica de usinagem, o primeiro passo é a inserção de um código em linguagem G. O código pode ser digitado diretamente na área de edição ou então aberto de um arquivo de texto previamente escrito, por meio do menu Abrir. O código então deve ser submetido a uma verificação de sintaxe, com um clique no botão Verificar Sintaxe. Se houver erros, deve-se corrigi-los na área de edição, e em seguida fazer uma nova verificação. Caso contrário, o programa está pronto para ser carregado.

O próximo passo é carregar o programa na máquina, através de um clique no btnCarregarG. Em seguida, é preciso indicar a posição inicial, da

ferramenta nos eixos X (vertical) e Z (horizontal), isto é, em que ponto ela se encontra no começo da execução do código G. Esta localização é o ponto de partida do percurso da ferramenta.

Segue-se então uma série de ajustes na área de visualização. O usuário pode reposicionar o sistema de coordenadas na tela, alterar o diâmetro da peça e alterar a escala do desenho.

Neste ponto clica-se no botão Iniciar, para dar início à execução. Porém, se o código G contiver erros, o programa não será executado e uma mensagem será exibida, informando que o código é inválido. Caso o código seja válido, o indicador de *status* irá mostrar que a máquina está fazendo o *set up* para a operação, e dentro de alguns instantes o *status* muda para USINANDO e é possível ver na tela o percurso da ferramenta, bem como a localização da mesma em relação ao sistema de coordenadas, em milímetros.

A qualquer momento, a operação de usinagem pode ser pausada, com um clique no botão Pausar, e retomada, com um clique em Iniciar. Pode-se também interromper a execução, clicando em Parar.

Ao término da execução do programa G, o *status* da máquina mudará para RETIRANDO, e depois de alguns segundos mudará novamente para OCIOSO. Pode-se então executar uma nova operação com o mesmo código, por meio do botão Iniciar, ou então carregar um novo código G.

4.4. Procedimento para operação remota

Para a operação via web, o usuário conta com uma página web que contém campos para inserção da posição inicial da ferramenta, área de edição de código G, área de visualização da usinagem e os botões para carregar o código G, verificar a sintaxe, iniciar/pausar a operação e parar. Esta interface de acesso remoto pode ser visto na Figura 4.2.

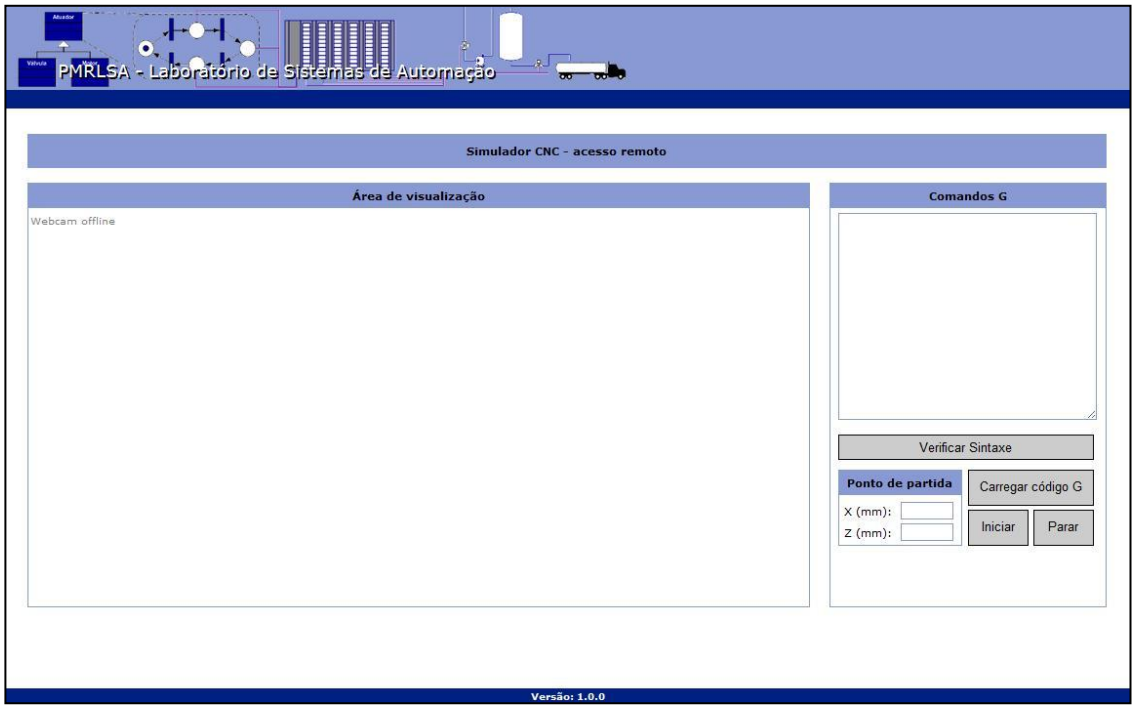

Figura 4.2. Interface web.

Neste caso, o usuário deve inserir o código G na área de edição, e assim como no procedimento local, verificá-lo e carregá-lo. Deve-se então indicar o ponto inicial da ferramenta e clicar em Iniciar.

Da mesma forma que na operação local, o usuário pode então pausar ou parar a usinagem. A área de visualização é constantemente atualizada, para o acompanhamento do percurso da ferramenta.

5. Testes realizados

Para a visualização do funcionamento do sistema foram elaborados dois códigos G diferentes, para serem executados pelo *software* nos dois modos de operação: local e remoto.

5.1. Teste 1 – operação local

Para este teste, foi elaborado um programa G que diminui em alguns milímetros o raio da extremidade do cilindro. Usou-se um cilindro de diâmetro 40 mm, a área de visualização de 200 x 100 mm e ponto de partida (0, 0).

`N010 G00 F200 S300 M03`	Seleção de velocidades de avanço e rotação
`N020 X-25 Z10`	Posicionamento rápido em (10, -25)
`N030 G91 Z-10 X4`	Modo incremental, posicionamento rápido em (0, -16)
`N040 G01 X2`	Movimentação a 200 mm/min para (0, -14)
`N050 F100 Z-15`	Movimentação a 100 mm/min para (-15, -14)
`N060 X-2`	Movimentação a 100 mm/min para (-15, -16)
`N070 G00 Z15`	Posicionamento rápido em (0, -16)
`N080 G01 F200 X3`	Movimentação a 200 mm/min para (0, -13)
`N090 F100 Z-15`	Movimentação a 100 mm/min para (-15, -13)
`N100 X-3`	Movimentação a 100 mm/min para (-15, -16)
`N110 G00 Z15`	Posicionamento rápido em (0, -16)
`N120 G01 F200 X4`	Movimentação a 200 mm/min para (0, -12)
`N130 F100 Z-15`	Movimentação a 100 mm/min para (-15, -12)
`N140 X-4`	Movimentação a 100 mm/min para (-15, -16)
`N150 G00 G90 X-9 Z25`	Modo absoluto, posicionamento rápido em (10, -25)
`N160 M02`	Fim de programa

A tela do programa após a operação pode ser vista na Figura 5.1.

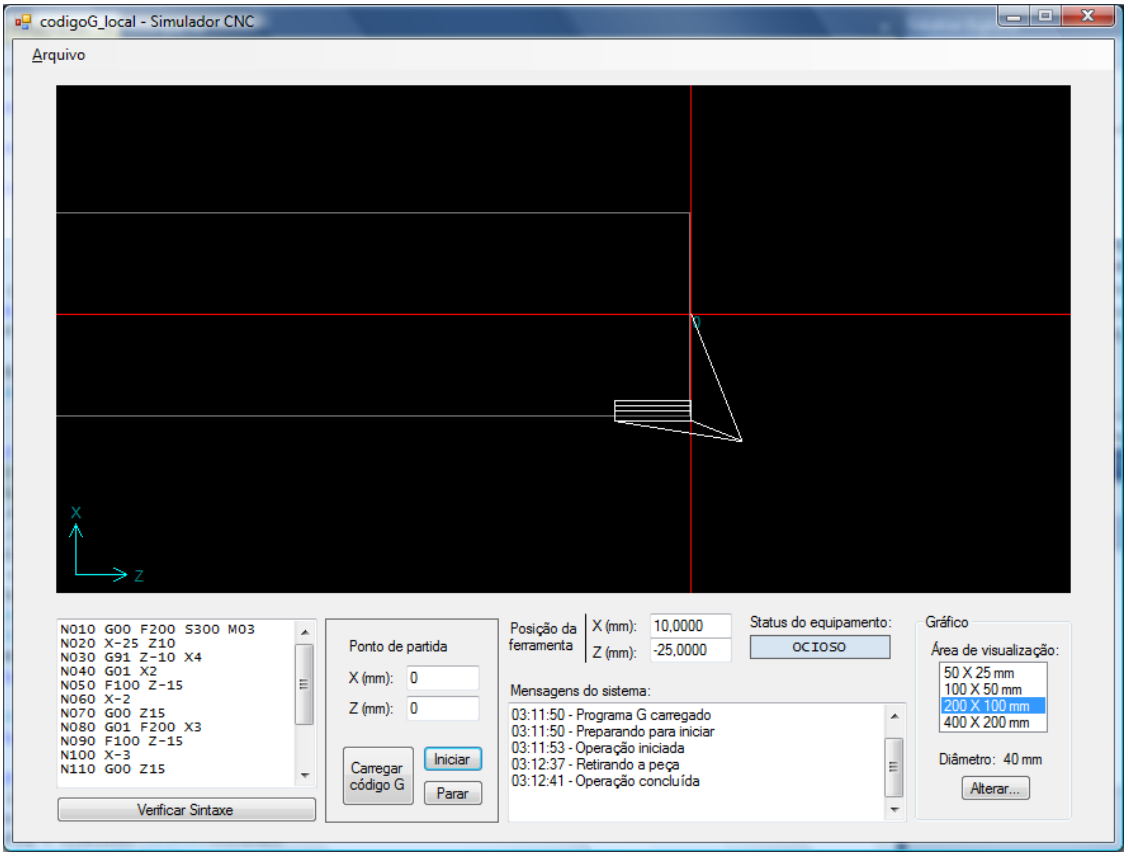

Figura 5.1. Tela do programa após o teste 1.

5.2. Teste 2 – operação remota

O código G utilizado neste teste é um programa que usina um chanfro de 45° em um cilindro e em seguida afasta a ferramenta da peça. O ponto inicial, desta vez, foi a posição (10, -10).

`N010 G00 F100 S300 M03`	Seleção de velocidades de avanço e rotação
`N020 X-25 Z10`	Posicionamento rápido em (10, -25)
`N030 G91 Z-9 X11`	Modo incremental, posicionamento rápido em (1, -14)
`N040 G01 Z-1`	Movimentação a 100 mm/min para (0, -14)
`N050 Z-1 X-1`	Movimentação a 100 mm/min para (-1, -15)
`N060 G00 Z2`	Posicionamento rápido em (1, -15)
`N070 X2`	Posicionamento rápido em (1, -13)
`N080 G01 Z-1`	Movimentação a 100 mm/min para (0, -13)
`N090 Z-2 X-2`	Movimentação a 100 mm/min para (-2, -15)
`N100 G00 Z3`	Posicionamento rápido em (1, -15)
`N110 X3`	Posicionamento rápido em (1, -12)
`N120 G01 Z-1`	Movimentação a 100 mm/min para (0, -12)

`N130 Z-3 X-3`	Movimentação a 100 mm/min para (-3, -15)
`N140 G00 Z4`	Posicionamento rápido em (1, -15)
`N150 X4`	Posicionamento rápido em (1, -11)
`N160 G01 Z-1`	Movimentação a 100 mm/min para (0, -11)
`N170 Z-4 X-4`	Movimentação a 100 mm/min para (-4, -15)
`N180 G00 G90 X-25 Z10`	Modo absoluto, posicionamento rápido em (10, -25)
`N190 M02`	Fim de programa

Após a operação, obteve-se o desenho do trajeto da ferramenta que está mostrado na Figura 5.2.

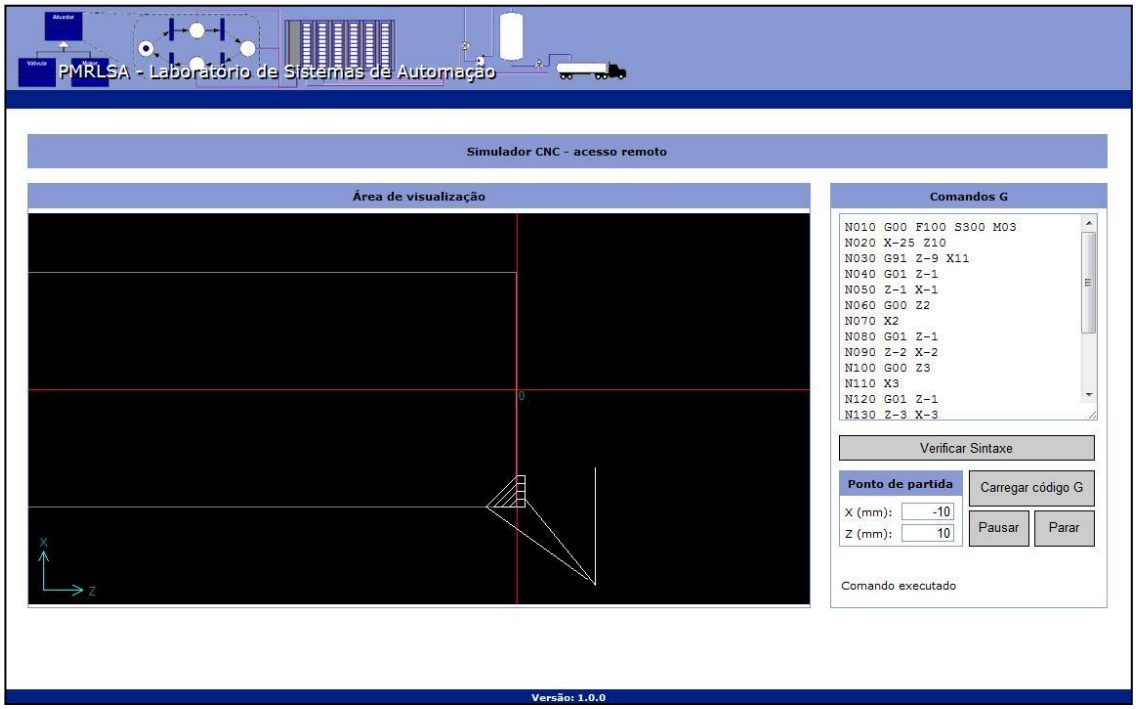

Figura 5.2. Tela da página web após o teste 2.

6. Conclusões

A ideia de um sistema que permita a simulação de um processo de usinagem, possibilitando redução de custos e tempo gasto para testes, e a tele-operação de um equipamento CNC, considerando o contexto de redução dos ciclos de inovação e acirrada competição internacional, foi o que motivou este trabalho.

Tendo isso em mente, foi proposto um sistema computacional capaz de interpretar código G, verificar a validade dos comandos, simular o processo de usinagem e exibir o trajeto percorrido pela ferramenta durante a operação.

Para atingir este objetivo, foi necessário o levantamento de informações relevantes ao desenvolvimento do projeto. Com base na teoria de Manufatura Virtual, Web Services e modelagem de sistemas em UML, foi possível criar um *software* cujo comportamento procura se aproximar do observado em máquinas CNC reais, inclusive no que diz respeito às limitações do equipamento. A principal ferramenta utilizada foi o Microsoft Visual Studio 2010, que possibilitou o desenvolvimento completo da solução, inclusive o bloco responsável pelo acesso via web.

Os testes realizados provam que o objetivo foi alcançado, pois tanto na operação local quanto na remota foi observado que o código G foi corretamente interpretado e o sistema exibiu graficamente o percurso do equipamento. Além disso, nenhuma funcionalidade apresentou problema.

Foi identificado que há uma limitação da velocidade de avanço da ferramenta, atrelada à velocidade de processamento do computador em que o programa está sendo executado, já que há muitos cálculos envolvidos na interpolação linear e no desenho da trajetória.

Os principais pontos de melhoria são:

- Aumentar o número de comandos G reconhecidos pelo programa;
- Melhorar a qualidade da exibição da imagem na operação remota;

- Aprimorar a visualização da usinagem, possibilitando que o usuário veja a forma da peça sendo alterada;
- Fazer melhorias gerais na aparência da interface gráfica;

Referências Bibliográficas

ALTINTAS, Y. **Manufacturing Automation**. [S.l.]: Cambridge University Press, 2000.

ALTOVA. **Web services: Benefits, challenges and a unique, visual development solution**. [S.l.], p. 21. 2006.

BANERJEE, P.; ZETU, D. **Virtual Manufacturing**. New York: John Wiley & Sons, 2001.

BELL, D. et al. A Framework for Deriving Semantic Web Services. **Springer Science and Business Media online**, 2006.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **The Unified Modeling Language User Guide**. 2. ed. [S.l.]: Addison-Weasley, 2005.

BRUNE, O. Comando numérico computadorizado. **Mecatrônica Atual**, 2008. Disponivel em: <http://www.mecatronicaatual.com.br/secoes/leitura/291>. Acesso em: 29 jun. 2010.

CERAMI, E. **Web Services Essentials - Distributed Applications with XML-RPC, SOAP, UDDI & WSDL**. 1ª Edição. ed. [S.l.]: O'Reilly, 2002.

FLANAGAN, C. The Bresenham Line-Drawing Algorithm. **Department of Computer Science - University of Helsinki**, 1996. Disponivel em: <http://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html>. Acesso em: 21 nov. 2010.

KIMURA, F. Product and Process Modeling as aKernel for Virtual Manufacturing Environment. **Annals of the CIRP**, v. 4, p. 147-150, fev. 1993.

LAWRENCE ASSOCIATES INC. Virtual manufacturing user workshop. **Technical Report**, 1994. Disponivel em: <http://www.isr.umd.edu/Labs/CIM/vm/lai1/final6.html>. Acesso em: 14 abr. 2010.

LEE, K. I.; NOH, S. D. Virtual Manufacturing System - A Test-Bed Of Engineering Activities. **CIRP Annals - Manufacturing Technology**, v. 46, p. 347-350, 1997.

LEE, W. B.; CHEUNG, C. F.; LI, J. G. Applications of virtual manufacturing in materials processing. **Journal of Materials Processing Technology**, v. 113, p. 416-423, 2001.

MOLLICA, M. F. **Simulador de Torno CNC**. Trabalho de Conclusão de Curso. Escola Politécnica da Universidade de São Paulo. São Paulo, p. 50. 2001.

MOYNE, J. R.; TILBURY, D. M. The Emergence of Industrial Control Networks for Manufacturing Control, Diagnostics, and Safety Data. **Proceedings of the IEEE** , v. 95, p. 29 - 47 , jan. 2007.

MUNDO CNC. Conceitos Básicos. **Mundo CNC**, 2008. Disponivel em: <http://www.mundocnc.com.br/basic4.php>. Acesso em: 29 julho 2010.

ONOSATO, M.; IWATA, K. Development of a Virtual Manufacturing System by Integrating Product Models and Factory Models. **Annals of the CIRP**, v. 4, fev. 1993.

PORTO, A. J.; PALMA, J. G. Fábrica do futuro: entenda hoje como sua indústria vai ser amanhã. In: _____ **Manufatura Virtual**. [S.l.]: Banas, 2000. Cap. 10, p. 89-97.

SOUZA, M. C. F. et al. Manufatura Virtual: Conceituação e Desafios. **Gestão & Produção**, v. 9, p. 297-312, dez. 2002.

STOETERAU, R. L. Introdução aos processos de usinagem. **Laboratório de Mecânica de Precisão**, 2007. Disponivel em: <http://www.lmp.ufsc.br/disciplinas/emc5240/Aula-01-U-2007-1-Introducao.pdf>. Acesso em: 21 Julho 2010.

STOETERAU, R. L. Introdução aos processos de usinagem. **Laboratório de Mecânica de Precisão**. Disponivel em: <http://www.lmp.ufsc.br/>. Acesso em: 21 Julho 2010.

W3C RECOMMENDATION. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). **World Wide Web Consortium**, 2007. Disponivel em: <http://www.w3.org/TR/soap12-part1/>. Acesso em: 15 jun. 2010.

W3C RECOMMENDATION. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. **World Wide Web Consortium**, 2007. Disponivel em: <http://www.w3.org/TR/wsdl20/>. Acesso em: 15 jun. 2010.

W3C WORKING GROUP NOTE. Web Services Architecture. **World Wide Web Consortium**, 2004. Disponivel em: <http://www.w3.org/TR/ws-arch/>. Acesso em: 15 jun. 2010.