DISSERTAÇÃO DE MESTRADO

CONCEPÇÃO DE UM *FRAMEWORK* PARA MONITORAMENTO E TELEOPERAÇÃO DE MÁQUINAS-FERRAMENTA CNC VIA INTERNET ADERENTE À INDÚSTRIA 4.0

LUIZ EDUARDO SANTOS DE OLIVEIRA

Brasília, Maio de 2017

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

CONCEPÇÃO DE UM *FRAMEWORK* PARA MONITORAMENTO E TELEOPERAÇÃO DE MÁQUINAS-FERRAMENTA CNC VIA INTERNET ADERENTE À INDÚSTRIA 4.0

LUIZ EDUARDO SANTOS DE OLIVEIRA

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM SISTEMAS MECATRÔNICOS

APROVADA POR:

Prof. Dr. Alberto José Alvares, PPMEC/UnB
Orientador

Prof. Dr. Daniel M. M. Arboleda, PPMEC/UnB
Membro Interno

Prof. Dr. Renan Bonnard, MENESR/França
Membro Externo

BRASÍLIA/DF, 12 DE MAIO DE 2017

Santos de Oliveira, Luiz Eduardo
Concepção de um *Framework* para Monitoramento e Teleoperação de Máquinas-Ferramenta CNC via internet aderente à Industria 4.0 / LUIZ EDUARDO SANTOS DE OLIVEIRA. –Brasil, 2017.
205p.

Orientador: Alberto José Alvares

Dissertação (Mestrado) – Universidade de Brasília – UnB
Faculdade de Tecnologia – FT
Programa de Pós-Graduação em Sistemas Mecatrônicos – PPMEC, 2017.

1. Indústria 4.0. 2. MTConnect. 3. OPC. 4. CNC. 5. Teleoperação. I. Alberto José Alvares, orientador. II. Universidade de Brasília. III. Faculdade de Tecnologia.

Agradecimentos

Por esse feito e todo o aprendizado recebido ao longo desse curso gostaria de agradecer primeiramente ao arquiteto do universo, e verdadeira essência dos seres humanos, Deus.
Agradeço ao povo Brasileiro que com seus impostos ajuda a manter a educação pública superior no Brasil, e assim, me deu a oportunidade de fazer esse curso de mestrado.
Presto meus agradecimentos ao CNPQ e CAPES, pelo apoio em termos de infraestrutura de trabalho e financeiro.
Ao meu orientador, Prof. Alberto José Álvares, pelo apoio e conhecimentos transmitidos, e ao coordenador do programa de pós-graduação em sistemas mecatrônicos, Prof. Edson Paulo da Silva.
Dedico profunda gratidão aos meus pais, Luiz Alberto e Inez Júlia, pelo amor e incentivo transmitidos ao longo desses semestres. Também pelo incentivo e muito mais, agradeço aos meus irmãos: Ana Rita, André Luiz e Allan Alberto; aos meus cunhados: Flávia Roberta e Júlio Tavares, que ajudaram a trazer ao mundo três preciosidades, meus sobrinhos: Alice Júlia, Ana Júlia e Luiz Alberto Neto.
Gostaria também de prestar os meus agradecimentos aos amigos e colegas, como: Giselle Leite, Juan Toquica, Manuel Abadia, Ofelia Marin, Maurício Merino, Sergio Petruz, Efraim Rodriguez, Christian Riaño, Jairo Muñoz, Oscar Anacona, Willian Gabalan, Sergio Reyes, Joao Cuni, Luis Aramis, Herlys Torres, e a todos aqueles colegas que compartilhei momentos, me incentivou e de alguma forma colaborou para a concretização desse êxito.
Faço um agradecimento especial a Francisca Nobre de Lima, pelos momentos compartilhados, pela amizade e o incentivo concedido nos meses anteriores à conclusão desse trabalho.

LUIZ EDUARDO SANTOS DE OLIVEIRA

RESUMO

As empresas vêm passando por constantes mudanças organizacionais a fim de se manterem competitivas em um mundo em que as comunicações estão cada vez mais rápidas e em que o setor empresarial atua em um nível global. Observando isso, a indústria manufatureira investe fortemente em insumos como as tecnologias de informação e comunicação (TIC) a fim de obterem informações e conhecimento necessários sobre os seus processos para que melhores decisões possam ser tomadas e as empresas tornem-se mais eficientes. Essa globalização dos mercados lançou para as empresas de manufatura o desafio de criarem uma maior proximidade com seus consumidores e parceiros com a finalidade de fabricar produtos que atendam verdadeiramente as expectativas dos *stakeholders*. Nessa tarefa a Internet assume um papel importante como elemento de integração entre pessoas e empresas.

Uma prova da força da Internet é o expressivo tamanho que o comércio eletrônico adquiriu ao logo dos anos. Outras aplicações para Internet vêm sendo implementadas com êxito. No âmbito da manufatura, acompanhando as mudanças nas estratégias de produção focadas no uso intensivo das tecnologias da informação, desenvolveu-se o conceito de manufatura remota e distribuída através da internet, que implementou paradigmas como a *E-manufacturing* (manufatura eletrônica). Esta forma um ambiente de manufatura baseado em TICs, especialmente em tecnologias de rede, incluindo a internet, utilizando métodos de trabalho vinculados a *Collaborative e-Work* (Trabalho eletrônico colaborativo). Nessa perspectiva, é possível vincular os estudos relacionados a manufatura eletrônica aos primeiros movimentos no sentido de desenvolver as Fabricas Inteligentes (*Smart Factories*), diretriz principal da estratégia Indústria 4.0.

A era da Indústria 4.0, implica, entre outros fatores, na utilização intensiva de produtos mecatrônicos e sistemas de tecnologia da informação (TI). Nesse contexto estão os sistemas de Internet das Coisas Industrial (IIoT) para ampliar ainda mais a imersão nos ambientes de produção. Uma das principais expectativas com relação a esses sistemas é o aumento do nível de interoperabilidade entre os mais diversos dispositivos da manufatura e a capacidade de monitoramento e controle desses equipamentos de forma mais distribuída.

Um dos grandes desafios das empresas de manufatura é a aquisição de dados de forma integrada em uma planta industrial composta por máquinas CNC com diferentes tecnologias e fabricantes, que desenvolvem sistemas para comunicação com esses dispositivos utilizando protocolos proprietários. Analisando essa realidade, e aproveitando o fato de que padrões como MTConnect e OPC-UA estão criando o caminho para o desenvolvimento da nova era indústria, este trabalho apresenta a arquitetura de um *framework* implementado na forma de um sistema cliente-servidor baseado na internet para o monitoramento e a teleoperação máquinas-ferramenta CNC, que apresente atributos em conformidade com a estratégia "Indústria 4.0".

A concepção do *framework*, e a subsequente implementação, foi realizada através de uma abordagem metodológica que envolveu o uso de diagramas IDEF0 (*Integration Definition language for Function Modeling*), para a definição da modelagem funcional do *framework*, e a metodologia de projeto axiomático para o refinamento do modelo em uma especificação mais detalhada da arquitetura na forma de um sistema baseado na web para o monitoramento e a teleoperação de um centro de torneamento.

O trabalho de implementação computacional fundamentou-se no desenvolvimento, na herança de módulos de software previamente desenvolvidos e na integração desses módulos na forma de serviços através de um sistema na web. Foi desenvolvido um serviço de monitoramento baseado no protocolo MTConnect, para fornecer *streaming* de dados de fabricação do centro de torneamento. Para atividades de supervisão foi implementado um servidor OPC para Web (OPCWeb), em que buscou-se criar um alternativa ao servidor OPC-UA, que é baseado em padrões para internet. Como uma forma de manter a identidade do sistema dentro do contexto das aplicações de telemanufatura foi incorporado ao sistema serviços de teleoperação como comando remoto DNC (*Distributed Numerical Control*) através mecanismo CGI (*Common Gateway Interface*) e *streaming* de video do chão-de-fábrica com *Applets* Java. Todos esses servidores integrados por uma interface Web para formar o sistema CyberDNC.

A forma final do sistema foi avaliada através de casos de uso da aplicação focados na conectividade entre os servidores e os clientes web disponíveis, e adequado funcionamento da aplicação como um todo. Os testes e validação do sistema (*framework*) foram realizados para o centro de torneamento Romi Galaxy 15M com CNC Fanuc 18i-Ta.

ABSTRACT

The companies have going through constant organizational changes in order to remain competitive in a world where communications are increasingly fast and the business sector operates on a global level. Noting that manufacturing industry invests heavily in inputs such as information and communication technologies (ICT) in order to obtain information and knowledge about your processes so that better decisions can be taken and companies become more efficient. This globalization of markets has released to manufacturing companies the challenge of creating greater proximity with its customers and partners in order to manufacture products that truly meet the expectations of stakeholders. In this task the Internet plays an important role as a factor of integration between people and companies.

A proof of the strength of the Internet is the significant size of the e-commerce acquired over the years. Other Internet applications are being implemented successfully. In the context of the manufacture, tracking changes in the production strategies focused on intensive use of information technologies, has developed the concept of remote manufacturing and distributed via the internet, which implemented paradigms like E-manufacturing (electronic manufacturing). This way a manufacturing environment based on ICTs, especially in network technologies, including the internet, using working methods linked to Collaborative e-Work (Collaborative electronic Work). In this perspective, it is possible to link the electronic manufacturing-related studies to the first movements towards developing Smart Factories, main Industry 4.0 strategy guideline.

The industry 4.0 era, implies, among other factors, on intensive use of mechatronics products and systems of information technology. In this context are the systems of Industrial Internet of things (IIoT) to further expand the immersion in production environments. One of the main expectations with regard to these systems is the increased level of interoperability between different manufacture devices and monitoring and control capacity of the equipment in a more distributed way.

One of the greatest challenges of manufacturing companies is the acquisition of data seamlessly in an industrial plant composed of CNC machines with different technologies and manufacturers, to develop systems for communication with these devices using proprietary protocols. Analyzing this reality, and taking advantage of the fact that standards such as OPC-UA and MTConnect are creating the path for the development of the new era industry, this paper presents the architecture of a framework implemented as a client-server system based on internet for monitoring and teleoperation CNC machine tool, which shows attributes in accordance with the strategy "Industry 4.0".

The design of the framework, and the subsequent implementation, was performed through a methodological approach that involved the use of IDEF0 (Integration Definition for Function Modeling language)

for defining the functional modeling of the framework, and Axiomatic Design methodology for the refinement of the model in a more detailed specification of the architecture as a web-based system for monitoring and teleoperation of a turning center.

The task of computational implementation was based on the development, in the inheritance of previously developed software modules and integration of these modules in the form of services through a web system. It was developed a monitoring service based on MTConnect protocol, to provide manufacturing data streaming of turning center. For supervisory activities was implemented an OPC server for Web (OPCWeb), which sought to create an alternative to OPC-UA server, which is based on internet standards. As a way to keep the identity of the system within the context of telemanufacturing applications, was built into the system teleoperation services as DNC (Distributed Numerical Control) remote commands using CGI protocol (Common Gateway Interface) and video streaming of the shop-floor through Java Applets. All these servers integrated by a web interface to form the CyberDNC system.

The final form of the system was evaluated through use cases of the application focused on connectivity between the servers and the web clients available, and the suitable application's operation as a whole. The system (framework) testing and validation were carried out for the turning center Romi Galaxy 15M-Ta with CNC Fanuc 18i-Ta.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS E ABREVIATURAS

AD	Axiomatic Design
ADo-oSS	Axiomatic Design of Object-Oriented Software Systems
AJAX	Asynchronous Javascript And XML
API	Application Programming Interfaces
CAD	Computer-Aided Design
CAD/CAM	Integração entre CAD e CAM
CAE	Computer-Aided Engineering
CAM	Computer-Aided Manufacturing
CAPP	Computer Aided Process Planning
CGI	Common Gateway Interface
CLP	Controlador Lógico Programável
COM	Component Object Model
CNC	Comando Numérico Computadorizado
CPS	Cyber-physical Systems
DCOM	Distributed Component Object Model
DNC	Distributed Numeric Control
E-Mfg	Electronic Manufacturing
ERP	Enterprise Resource Planning
FMC	Flexible Manufacturing Cell
FOCAS1	Fanuc Open CNC API Specification 1
GUI	Graphical User Interface
GRACO	Grupo de Automação e Controle

HTML	Hiper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IDEF0	Integration DEFinition language for Function Modeling
IHM	Interface Homem-Máquina
IoT	Internet of Things
ISO	Internacional Organization for Standarization
MDI	Manual Data Input
MES	Manufacturing Execution Systems
NC	Numerical Command (Comando Numérico)
NIST	National Institute of Standards and Technology
OPC	Object Linking and Embedding for Process Control
OPC-DA	OPC Data Access
OPC-UA	Open Process Control United Archtecture
PA	Projeto Axiomático
PC	Personal Computer
PLC	Programmable logic controller
PMC	Programmable Machine Control
REST	Representational State Transfer
SOA	Service-oriented Architecture
SCADA	Supervisory Control and Data Acquisition
SHDR	Simple Hierarchical Data Representation
SMTP	Simple Mail Transfer Protocol
STEP	Standard for the Exchange of Product Model Data
TCP/IP	Transmission Control Protocol/Internet Protocol
TIC	Tecnologia da Informação e Comunicação
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	eXtensible Markup Language
WSDL	Web Services Description Language

Capítulo 1

Introdução

O mundo empresarial passa por constantes mudanças impulsionadas pela necessidade das organizações se manterem competitivas em um mercado global e responderem aos anseios de consumidores exigentes com demandas variadas e contínuas. Nesse contexto, as empresas estão vendo a necessidade de adequar os seus processos em estruturas que garantam uma produção mais eficiente, com mais qualidade, menores custos e tempos de ciclo reduzidos. No âmbito das empresas de manufatura os fatores de competitividade estão intimamente relacionados ao sistema de produção, envolvendo uma adequada gestão do chão-de-fábrica e uma clara visão sobre o seu estado operacional. Observando isso, a indústria manufatureira investe cada vez mais na integração de suas instalações físicas por meio das tecnologias de informação e comunicação (TIC) a fim de obter informações sobre o processo produtivo de forma detalhada, precisa e em tempo real, e com isso decisões possam ser tomadas, produtos sejam aprimorados e as expectativas individuais dos consumidores possam ser atingidas.

A globalização dos mercados lançou um desafio às empresas de manufatura de todo o mundo, uma vez que produtos e componentes variados e de boa qualidade podem ser produzidos em diferentes países e disponibilizados para compradores em todo o globo a um custo acessível. Esse ambiente gera a necessidade de as empresas criarem uma maior proximidade com seus consumidores e parceiros com a finalidade de fabricar produtos ainda mais condizentes com as expectativas dos *stakeholders*. Para isso, atribui-se a Internet o papel de elemento-chave que permitirá uma integração e comunicação rápida entre pessoas e empresas. Uma primeira prova dessa força da Internet está associada ao tamanho expressivo que o comércio eletrônico adquiriu nas economias ao redor do mundo nos últimos anos.

Em um passado recente a Internet tem sido empregada com sucesso em atividade diversas e importantes como a manutenção remota de equipamentos (ROCKWELL, 2002) e a tele-cirurgia conduzida através de um robô comandado remotamente por um cirurgião situado a quilômetros do paciente (SCIENTIFIC-AMERICAN, 2000). No âmbito da manufatura, acompanhando as mudanças nas estratégias de produção focadas no uso intensivo das tecnologias da informação, foram realizados uma série de estudos relacionados ao uso da Internet aplicado a manufatura remota e distribuída, implementando paradigmas como a *E-manufacturing* (manufatura eletrônica) (ALVARES et al., 2002; ALVARES; SILVA; FERREIRA, 2005; ALVARES; FERREIRA, 2008; ALVARES, 2005; BENAVENTE, 2011).

Os aspectos que caracterizam a manufatura eletrônica coadunam com a nova era da indústria manufa-

tureira, a quarta revolução industrial, também chamada de Indústria 4.0. Esta fundamenta-se na integração de todos os elementos que compõem o sistema de manufatura através das tecnologias da informação e comunicação para o desenvolvimento das *Smart Factories* (fábricas inteligentes). A *E-manufacturing* (*e-Mfg*) forma um ambiente de manufatura baseado em TICs, especialmente em tecnologias de rede, incluindo a Internet, utilizando métodos de trabalho vinculados a *Collaborative e-Work* (trabalho eletrônico colaborativo), empregado no ciclo de desenvolvimento de produtos que envolvem sistemas CAD/CAPP/-CAM integrados (LEE, 2003; NOF, 2004). Nessa perspectiva os sistemas para a Indústria 4.0 abrangem os sistemas de *e-Mfg*, mas expandem o conceito de integração de informações para o nível de sensores e atuadores ao ter como base os *Cyber-physical Systems* (Sistemas Ciber-físicos, CPS) e *Internet of Things and Services* (Internet das Coisas e Serviços).

Os CPSs e a introdução da Internet das Coisas e da Internet de Serviços (IoT e IoS) na indústria é o que caracteriza a era da industria 4.0 (KAGERMANN; WAHLSTER; HELBIG, 2013). Os sistemas Ciber-físicos representam a junção de sistemas embarcados, a Internet, dados e serviços disponíveis *on-line* (MACDOUGALL, 2014). Esses sistemas estão focados em uma rede integrada de processos e recursos físicos que permite a fusão de sistemas reais e virtuais (GORECKY et al., 2014). Segundo McDougall (MACDOUGALL, 2014), CPSs criam as bases para a criação de uma Internet das Coisas, que combinada com a Internet dos Serviços, formam o que é conhecido como Indústria 4.0.

A Internet das Coisas é uma rede inteligente que conecta todas as coisas a Internet, a exemplo de sensores e atuadores, com a finalidade de permitir a comunicação entre esses dispositivos através de protocolos conhecidos, permitindo a identificação, o rastreamento, a localização, o monitoramento e o gerenciamento inteligente dessas entidades (GUBBI et al., 2013). É a expansão das redes baseadas na Internet, a qual amplia a comunicação entre pessoas, para pessoas e coisas, ou coisas e coisas (CHEN et al., 2014).

Sistemas de software de apoio à manufatura desempenham um papel importante dentro organizações industriais há algumas décadas. No âmbito da Indústria 4.0, esses sistemas expandem o seu alcance, fazendo parte de uma rede em que tudo representa um objeto mapeável através de um endereço único. Alguns desses sistemas formam interfaces de usuário conectadas a sistemas embarcados baseados em softwares de alto desempenho. Essa interação, segundo McDougall (MACDOUGALL, 2014), cria um mundo completamente novo em termos de funcionalidades dos sistemas. Um exemplo desses sistemas são os atuais telemóveis que oferecem um pacote completo de aplicações e serviços que ultrapassam as funções de telefonia para as quais esses dispositivos foram originalmente projetados.

Os sistemas para a Indústria 4.0 estão fundamentados na integração de dados e serviços virtualmente localizados que estão vinculados aos recursos físicos da manufatura, através de uma rede inteligente (*Cloud Computing)*. Essa integração é garantida através do uso de padrões de referência. A adoção desses padrões são apontadas por Kagermann et al. (KAGERMANN; WAHLSTER; HELBIG, 2013) como um requisito para a adequada implementação das tecnologias que compõem a quarta revolução industrial. Nesse nível, destacam-se padrões de conectividade como o OPC-UA (*Open process control united architecture*) e o mais recentemente desenvolvido, MTConnect. Albert (ALBERT, 2015) cita esses como padrões-chave para a promoção da IoT.

O MTConnect é um padrão aberto (*royalty-free*) e foi desenvolvido com o propósito de se tornar um protocolo que promovesse a integração de máquinas CNC (Comando Numérico Computadorizado). Esse é

baseado nos principais padrões existentes na indústria da manufatura e de software, como HTTP (*Hypertext Tranfer Protocol*) e XML (*Extensible Markup Language*) (SOBEL, 2010). O OPC-UA foi desenvolvido pela *OPC Foundation* e representa uma evolução do OPC clássico, baseado em tecnologia COM/DCOM, para formar um padrão de interoperabilidade independente de plataforma para a troca de dados entre o chão-de-fábrica e os níveis de decisão estratégicos das organizações (MAHNKE; LEITNER; DAMM, 2009). Enquanto o MTConnect facilita a conexão entre máquinas-ferramenta CNC e outros equipamentos em rede para coleta de dados de fabricação, o OPC-UA promove a interoperabilidade necessária para a comunicação de dados em toda a planta (ALBERT, 2015).

A forte presença da tecnologia da informação na manufatura já era prevista em estratégias como *e-Work* (Trabalho Eletrônico), definida pelo PRISM Center (NOF, 2004) como qualquer atividade produtiva colaborativa, apoiada por computador e comunicação entre organizações altamente distribuídas de pessoas e/ou robôs, ou sistemas autônomos. O *e-Work* é formado por *e-activities* (Atividades eletrônicas), das quais a *e-Manufacturing* faz parte. Nessa perspectiva, o trabalho eletrônico está inserido no principio do que é conhecido como a Internet de Serviços (IoS). Atividades antes executadas presencialmente são mapeadas virtualmente e transformadas em serviços integrados disponibilizados remotamente através da Internet.

O *E-work* inclui aplicações de telerobótica, telemanufatura, teleoperação e serviços (ALVARES, 2005). Neste trabalho, a Teleoperação remota é um dos objetivos tratados e é incluída no contexto das Fabricas Inteligentes como um recurso para garantir onipresença e informação de realimentação para uma ação sobre o processo de fabricação, mais especificamente a manufatura por usinagem.

Com isso, o presente trabalho propõe o planejamento e a implementação de um *framework* para monitoramento e teleoperação de máquinas-ferramenta CNC, tendo como elemento de teste e validação um centro de torneamento CNC da marca Romi modelo Galaxy 15M, provido com controlador Fanuc 18i-Ta. A implementação tem como resultado um sistema com arquitetura cliente/servidor baseada na Web, combinado com elementos aderentes a Indústria 4.0, como um serviço MTConnect para acesso a dados de CNC, serviço OPC via Web para interação com a máquina associada as funções de PMC/PLC (*Programmable machine control*/*Programmable logic controller*). Além dessas funções, há os serviços de comando remoto via HTTP e programas CGI, e monitoramento por imagem através de um serviço de *streaming* de vídeo via Internet (WebCam). A arquitetura do sistema é modular, então os testes de validação foram conduzidos por módulo (servidor) e com todos os serviços funcionando de forma integrada por meio de uma interface gráfica de usuário disponibilizada para *browser*.

1.1 Contextualização do Problema

Na indústria manufatureira a obtenção de informações precisas e rapidamente sobre processos de fabricação com o propósito de apoiar tomadas de decisão efetivas e estratégicas, é uma necessidade de empresas modernas que querem se manter competitivas. Há a necessidade de se conhecer a real capacidade das instalações de produção, e assim torná-las mais eficientes. A solução para essa questão passa pela sistematização e automatização de atividades com a menor interferência humana possível, o que proporciona um aumento na confiabilidade das informações.

É essencial o acesso integrado a dados relacionados ao processo produtivo para apoiar as necessida-

des de informação para os vários níveis organizacionais da empresa, e instalações de produção flexíveis e inteligentes que possuam a capacidade responder prontamente as mudanças de cenário nas atividades rotineiras da produção.

No âmbito da quarta revolução industrial, o conceito de acesso à informação é ampliado. Os elementos que orbitam em torno da cadeia produtiva estão conectados e integrados por meio de uma rede inteligente, que é conhecida como Internet das Coisas. Esse novo contexto permite que a produção seja configurada não apenas de modo mais flexível, mas aproveita os benefícios oferecidos por processos de gerenciamento e controle diferenciados.

A próxima geração das Fábricas Inteligentes definem a plena promoção da integração de dados e a estreita aproximação entre produto, processo, *stakeholders* e usuários finais. Com isso, é essencial o desenvolvimento e o domínio de tecnologias para a construção de sistemas que permitam a integração inteligente entre coisas, pessoas e recursos. "Coisas"podem representar máquinas ou produtos. Permitir que pessoas e coisas interajam de forma mais natural possível é parte do que propõe a estratégia para a iniciativa Indústria 4.0.

A construção do caminho para uma manufatura mais integrada passa pela adoção de arquiteturas de referência e tecnologias de rede de comunicação que possibilitem uma conectividade com custos reduzidos, maior eficiência e maior disponibilidade. Diante disso, Kagermann et al.(KAGERMANN; WAHLSTER; HELBIG, 2013) expõem um conjunto de recomendações para a implementação da iniciativa Indústria 4.0, uma dessas recomendações são ações voltadas para a padronização e a adoção de padrões abertos para a construção de um modelo de referência que conjugue todas as perspectivas dos elementos que compõem um sistemas produtivo, tais como processos de manufatura, dispositivos, aplicações de software para o ambiente de manufatura e para os níveis de planejamento, e na perspectiva de engenharia em um sistema de manufatura. A carência de uma arquitetura geral para o desenvolvimento das Fábricas inteligentes conduz empresas e estudiosos da manufatura ao desenvolvimento de projetos e propostas de arquitetura de sistemas aderentes a Indústria 4.0, tendo como base, uma ou a fusão de mais de uma das perspectivas citadas por Kagermann *et al.* (KAGERMANN; WAHLSTER; HELBIG, 2013).

Após uma análise das perspectivas de “Aplicações de software” e dos “Dispositivos em rede” é possível agir no sentido de projetar e implementar arquiteturas que envolvam sistemas baseados em serviços Web, sejam integrados através da internet a coisas, pessoas e outros serviços; que gerem informações em tempo real, tenha elevada disponibilidade, sejam orientados a serviços (SOA) e empreguem padrões abertos. Nesse universo persistem atividades como monitoramento através da web via *browser* e dispositivos móveis, supervisão distribuída de máquinas, análises de tendências com bancos de dados situados na nuvem (*Cloud computing*), entre outros. No âmbito dos padrões abertos estão os protocolos aplicados à Internet (TCP/IP, HTTP, SMTP, etc) e protocolos para promover conectividade e interoperabilidade na manufatura como o MTConnect e o OPC-UA.

A Teleoperação promove a intervenção em processos ou em dispositivos localizados fisicamente distantes por um operador remoto. Telepresença se refere à intensa utilização de realimentação sensorial para a teleoperação, provendo ao operador realismo baseado na ideia de “presença à distância” (BENAVENTE, 2007). Segundo Kagermann et al.(KAGERMANN; WAHLSTER; HELBIG, 2013), o futuro da Telepresença no âmbito da Indústria 4.0 está em plataformas situadas na nuvem onde sistemas de fabricação se

conectarão de modo similar a elementos de uma rede social, e lá encontrarão os especialistas mais adequados para resolver questões específicas. Esses especialistas terão acesso a um conjunto de recursos, dentre os quais a teleoperação para diferentes finalidades, a exemplo de atuação remota sobre os processos de fabricação.

1.2 Definição do Problema

O problema de pesquisa deste trabalho de mestrado consiste na concepção de uma plataforma associada ao desenvolvimento de um sistema de supervisão, monitoramento e teleoperação remota através da Internet de um centro de torneamento CNC, e que possua características em conformidade com o paradigma da Indústria 4.0. Para isso, incluindo em sua arquitetura elementos recomendados para a implementação dessa nova iniciativa no âmbito da manufatura. Dentre esses elementos está o fato do sistema ser configurado como um serviço disponível na internet e empregar padrões classificados como candidatos para a construção da indústria do futuro, como os protocolos MTConnect e OPC para internet (associado a mecanismo de acesso a web). Estes protocolos associados promovem a ampla conectividade e a necessária interoperabilidade entre uma ampla gama de dispositivos na manufatura.

A iniciativa Indústria 4.0 foi proposta pelas organizações de fomento a inovação na manufatura da Alemanha na forma de um conjunto de recomendações de implementação. No entanto, as referências nesse tema são poucas e o desenvolvimento dessas propostas não possui uma arquitetura de referência que combine todas as perspectivas que envolvem a manufatura. A premissa básica desse novo paradigma é que todos os insumos que constituem o sistema de produção, sejam elementos do mundo físico e virtual, estejam integrados e sejam rastreáveis em uma rede inteligente baseada na Internet.

O projeto da arquitetura do sistema de monitoramento/supervisão e teleoperação via internet procura cumprir as três premissas que caracterizam os desenvolvimentos para a Indústria 4.0, apontados no relatório *Recommendations for implementing the strategic initiative Industrie 4.0* (KAGERMANN; WAHLSTER; HELBIG, 2013) feito pela Acatech, Academia de Ciência e Engenharia da Alemanha, os quais são:

- Integração horizontal através de redes de valor;
- Integração digital de ponta-a-ponta de engenharia por toda a cadeia de valor;
- Integração vertical e sistemas de manufatura em rede.

O sistema deve prever a possibilidade de integração com módulos CAPP e CAM como em arquiteturas integradas CAD/CAPP/CAM através da internet, conforme demonstram Álvares (ALVARES, 2005) e Benavente (BENAVENTE, 2007; BENAVENTE, 2011), promovendo potencialmente integração vertical e solução digital de ponta-a-ponta.

Dentro de organizações com os seus processos altamente integrados é essencial a presença de um formato dados universal para o intercâmbio de dados entre os diferentes níveis organizacionais. O formato XML é amplamente utilizado em aplicações de software distribuídas que transmitem dados através de simples requisições HTTP (GET). Qualquer aplicação de diferentes níveis organizacionais de que acessam

recursos e serviços através da Internet tem capacidade de realizar requisições HTTP. Esses atributos garantem que dados produzidos pelos servidores localizados nos controladores de máquinas CNC, por exemplo, após serem processados e analisados possam ser acessados pelos níveis gerenciais mais altos de uma organização, como os níveis de planejamento, se estiverem disponíveis no formato XML. Essa característica dá a arquitetura do sistema o suporte a integração vertical.

1.3 Objetivos da Dissertação

1.3.1 Objetivo Geral

Conceber a arquitetura de uma plataforma para o monitoramento e a teleoperação de máquinas-ferramenta CNC via Internet com tecnologias e mecanismos que sejam aderentes estratégia "Indústria 4.0", como os padrões MTConnect e OPC para Internet, de forma que a partir desse *framework* seja implementado um sistema e serviço Web que faça a comunicação e possibilite a operação onipresente de um centro de torneamento da marca Romi, modelo Galaxy 15M, provido com controlador Fanuc 18i.

1.3.2 Objetivos Específicos

Para atingir o objetivo geral no presente trabalho, foram estabelecidos os objetivos específicos a seguir:

- Adequar a metodologia de projeto axiomático de software a concepção de um sistema Cliente/Servidor para Monitoramento e Teleoperação do centro de torneamento CNC Romi Galaxy 15M.
- Incluir no projeto do *framework* serviços baseados em protocolos candidatos a padrão de conectividade para à Industria 4.0, como o MTConnect e OPC para internet.
- Incluir na arquitetura proposta elementos de Manufatura Eletrônica (*e-Mfg*), como a teleoperação através da captura de imagens da célula de manufatura e comando remoto da máquina CNC via internet.
- Desenvolver um adaptador MTConnect para o CNC Fanuc 18i-Ta através do encapsulamento dos métodos fornecidos pela API Focas 1 (*Fanuc Open CNC Application Programming Interface Specification version 1*) da Fanuc, para a comunicação com um Agente MTConnect.
- Implementar, associado a um servidor OPC-DA (*OPC Data Access*), uma API (*Application Programming Interface*) para OPC, com a função de *gateway*, para a captura e acionamento através da web de funções de CLP do centro de torneamento Romi-Galaxy 15M, de forma que represente uma alternativa a um servidor OPC-UA.
- Mapear as funções o protocolo Focas1/DNC1 da Fanuc para implementação de atividades de DNC (Comando Numérico Distribuído, traduzido do inglês) para o envio de comandos via Internet ao controlador da máquina Romi Galaxy 15M.
- Integrar os serviços de monitoramento, supervisão e teleoperação implementados através do desenvolvimento de uma interface de usuário (GUI) para Web.

- Validar a arquitetura do *framework* através da implementação e teste dos módulos associados a arquitetura, tendo como elemento de prova o centro de torneamento Romi Galaxy 15M com CNC Fanuc 18i-Ta.

1.4 Abordagem Metodológica

O projeto dessa dissertação utiliza uma abordagem metodológica mista usufruindo da finalidade e da versatilidade de três diferentes ferramentas metodológicas, são elas: projeto axiomático, IDEF0 (*Function Modeling*) e diagramas UML (*Unified Modeling Language*).

O projeto axiomático baseia-se nas necessidades dos usuários do sistema para construir uma relação entre o domínio funcional (requisitos funcionais, FR) e o domínio físico (parâmetros de projeto, DP) de projeto, caracterizado por sucessivas decomposições de parâmetros de projeto em requisitos funcionais e o simultâneo mapeamento em matrizes de projeto que permitem visualizar os módulos do sistema de dão origem a arquitetura de sistema e fornecem a sequencia de execução do projeto através de diagrama de fluxo, produto final da metodologia. No processo de projeto do PA, a cada nova decomposição, busca-se sempre atender ao axioma 1 da independência e o axioma 2, da informação.

O uso do projeto axiomático neste trabalho está vinculado a definição dos módulos que irão compor a arquitetura detalhada do sistema e a sua sequencia de implementação. O PA é a maneira estabelecida para complementar a metodologia definida pelos diagramas IDEF0, que fornecem o modelo funcional e dá um perspectiva mais genérica do *framework*, que é implementado com o auxílio do projeto axiomático.

A IDEF0 permite descrever, através de uma hierarquia de diagramas, o modelo funcional (portanto, fluxo de informações dos processos) do sistema que se pretende analisar ou implementar. Não foi uma metodologia concebida para modelar aspectos temporais (RABELO, 2016). Pode ser usada para modelar uma ampla variedade de sistemas automatizados e não automatizados (COLOQUHOUN; BAINES; CROSSLEY, 1993). Liu, Sun e Mahdavian (LIU; SUN; MAHDAVIAN, 2008) afirmam que a flexibilidade da IDEF0 reside na capacidade de permitir a análise de complexos sistemas, onde há a necessidade de estudar múltiplos níveis de detalhamento das atividades de tal forma que o analista possa entender o sistema. Essa técnica apresenta as seguintes características: generalidade (para sistemas variáveis e complexos); rigor e precisão (modelos corretos e aplicáveis); concisão (facilidade de compreensão e consenso); conceito (representa requisitos funcionais) e flexibilidade (para várias fases do ciclo de vida de um projeto). Para Choo e Lee (CHOO; LEE, 1999) a técnica de modelagem de funções IDEF0 é usada para garantir a completa captura das informações requeridas.

Neste trabalho, a IDEF0 possibilita a definição do modelo do *framework* e auxilia no levantamento das necessidades dos usuários para o projeto axiomático, atuando como uma entrada para esta outra metodologia.

A UML é uma linguagem-padrão para a elaboração da estrutura de projetos de software, podendo ser empregada para a visualização, a especificação, a construção e a documentação de arquiteturas que façam uso de sistemas complexos de software. Neste trabalho, a UML é utilizada principalmente para documentar detalhes dos módulos da arquitetura do *framework* que possuam elementos de software programados com

linguagem de programação orientada a objeto e na interface gráfica de usuário.

1.5 Contribuições do Trabalho

As contribuições deste trabalho estão relacionadas a proposta e o desenvolvimento de uma arquitetura de uma plataforma para telemanufatura com características aderentes ao paradigma da Indústria 4.0.

Apresenta-se também diagramas IDEF0 associados ao projeto axiomático como uma abordagem metodológica promissora para o desenvolvimento de sistemas para a manufatura da quarta revolução industrial. Dentro dessa perspectiva, o uso do projeto axiomático gerou um artigo que foi aceito, apresentado e publicada no *The 10th International Conference on Axiomatic Design*, realizado da China, no mês de setembro de 2016. O trabalho recebeu o título "Axiomatic design applied to the development of a system for monitoring and teleoperation of a cnc machine through the internet"e resultou em uma publicação com efeito duplo, sendo publicado tanto nos anais da conferência, quanto na revista científica *Procedia CIRP* (vol.53, páginas.198-205).

O projeto do trabalho, que é inédito, combina em uma mesma interface de usuário através da internet as características e vantagens dos padrões de comunicação industrial, MTConnect e OPC, a fim de promover monitoramento, supervisão e teleoperação em centros de processamento CNC.

1.6 Estrutura do Documento

O presente trabalho foi organizado em uma estrutura de oito capítulos juntamente com esta introdução. Os capítulos foram dispostos logicamente em um sequência que favorece a compreensão das etapas do projeto de forma simplificada.

O capítulo 2 deste trabalho refere-se da revisão bibliográfica sobre os conceitos e referências acerca da Indústria 4.0, como Internet das Coisas (IoT), Sistemas Ciber-físicos (CPS) e Fábricas Inteligentes, apontando também potenciais aplicações e o desafios no âmbito da quarta revolução industrial. A revisão bibliográfica tem uma continuação no capítulo 3, onde apresenta-se os padrões candidatos para o desenvolvimento da indústria do futuro, como o MTConnect e OPC-UA. São também analisados conceitos associados a manufatura eletrônica (*e-Mfg*), teleoperação, controle supervisório, entre outros.

O capítulo 4 mostra a pesquisa bibliográfica sobre a metodologia de projeto axiomático, apresentando toda a sequência de etapas do processo de projeto para a definição da arquitetura de um sistema dentro dessa abordagem.

No capítulo 5 apresentada a metodologia empregada no desenvolvimento do trabalho. Nesse capítulo expõe-se a abordagem mista, envolvendo o projeto axiomático para o planejamento da arquitetura do sistema com elementos utilizados e desenvolvidos no âmbito deste trabalho e UML para a documentação do sistema por meio de diagrama de pacotes, diagramas de classe, diagramas de casos de uso e diagramas de atividade, associados a modelagem funcional do *Framework* através de diagramas IDEF0.

O capítulo 6 trata da implementação da arquitetura descrita na metodologia. São expostos os deta-

lhes do trabalho de desenvolvimento e configuração dos servidores que integram a arquitetura do sistema. O servidor MTConnect descrito pela demonstração dos detalhes do desenvolvimento do Adaptador e da configuração do Agente MTConnect. São descritas as características do servidor OPC-DA configurado e a sua integração com um módulo *gateway* para permitir conectividade via internet, que também é detalhado. Apresenta-se também as características do interface Web de usuário desenvolvida, com a integração dos servidores anteriormente citados. Nesse capítulo também expõe-se a incorporação das atividades de comando remoto via internet utilizando *scripts* CGI (*Common Gateway Interface*) e do servidor de *streamming* de imagens (WebCam).

O capítulo 7 é dedicado a apresentação dos testes e validação da arquitetura. Aí são descritos os ensaios de operação do centro de torneamento Romi Galaxy 15M (CNC Fanuc 18i-Ta) com o servidores de monitoramento/supervisão e os serviços de teleoperação implementados. Demonstra-se a versatilidade do servidor MTConnect através do teste de conectividade do software Agente com diferentes clientes disponíveis. São feitos também testes sobre o funcionamento da interface Web cliente desenvolvida, integrando todos os serviços implementados.

Por fim, há o capítulo 8, em que expõe-se as conclusões acerca dos resultados obtidos no trabalho, e onde também são estabelecidas propostas para trabalhos futuros.

Capítulo 2

Revisão Bibliográfica: Indústria 4.0

Neste capítulo expõe-se os principais conceitos estudados e analisados acerca do paradigma da Indústria 4.0 para a realização da pesquisa necessária para o projeto e implementação da arquitetura do *framework*.

2.1 Conceito de Indústria 4.0

Indústria 4.0 é o nome dado para a chegada da quarta revolução industrial, a nova era da indústria que está centrada no uso intensivo de recursos avançados de Tecnologia da Informação e Comunicação (TIC) com a finalidade de garantir maior flexibilidade a sistemas e processos de produção, como uma consequência da crescente complexidade de produtos e cadeias de suprimento (FALLER; FELDMÜLLER, 2015). O termo surgiu para nomear a estratégia do governo da Alemanha para o estímulo ao desenvolvimento de uma manufatura de última geração dentro de um conjunto de ações que também vem sendo adotadas em outros países do mundo.

As revoluções industriais caracterizam-se por mudanças abruptas e radicais, motivadas pela incorporação de tecnologias, que tem reflexo no campo econômico, social e político. Até os dias atuais a o consenso da existência de três revoluções industriais. A primeira ocorreu entre os anos de 1760 e 1840, impulsionada por tecnologias como máquinas a vapor e hidráulicas, e tem como símbolo dessa fase o advento do primeiro tear mecanizado. A segunda revolução deu-se entre o final do século XIX e início do século XX, sendo representada pela introdução da eletricidade, da linha de montagem e do surgimento da produção em massa baseada na divisão do trabalho. A terceiro estágio da industrialização começou na década de 1960, e é marcado pelo desenvolvimento dos semicondutores, e tecnologias como Mainframes e computadores pessoais. Foi a era do surgimento e difusão da eletrônica e das tecnologias da informação e comunicação (TICs), que mais tarde, nos anos 1990, viu o aparecimento da Internet (FDC, 2016). A Figura 2.1 ilustra brevemente as sucessivas revoluções industriais e o que caracteriza cada uma.

Com a evolução de algumas das tecnologias da terceira revolução industrial, juntamente com o advento e a incorporação de outras tecnologias, estudiosos sugerem que a partir do começo do século XXI, teria-se iniciado uma quarta revolução industrial, que na Alemanha recebeu o nome de "Indústria 4.0".

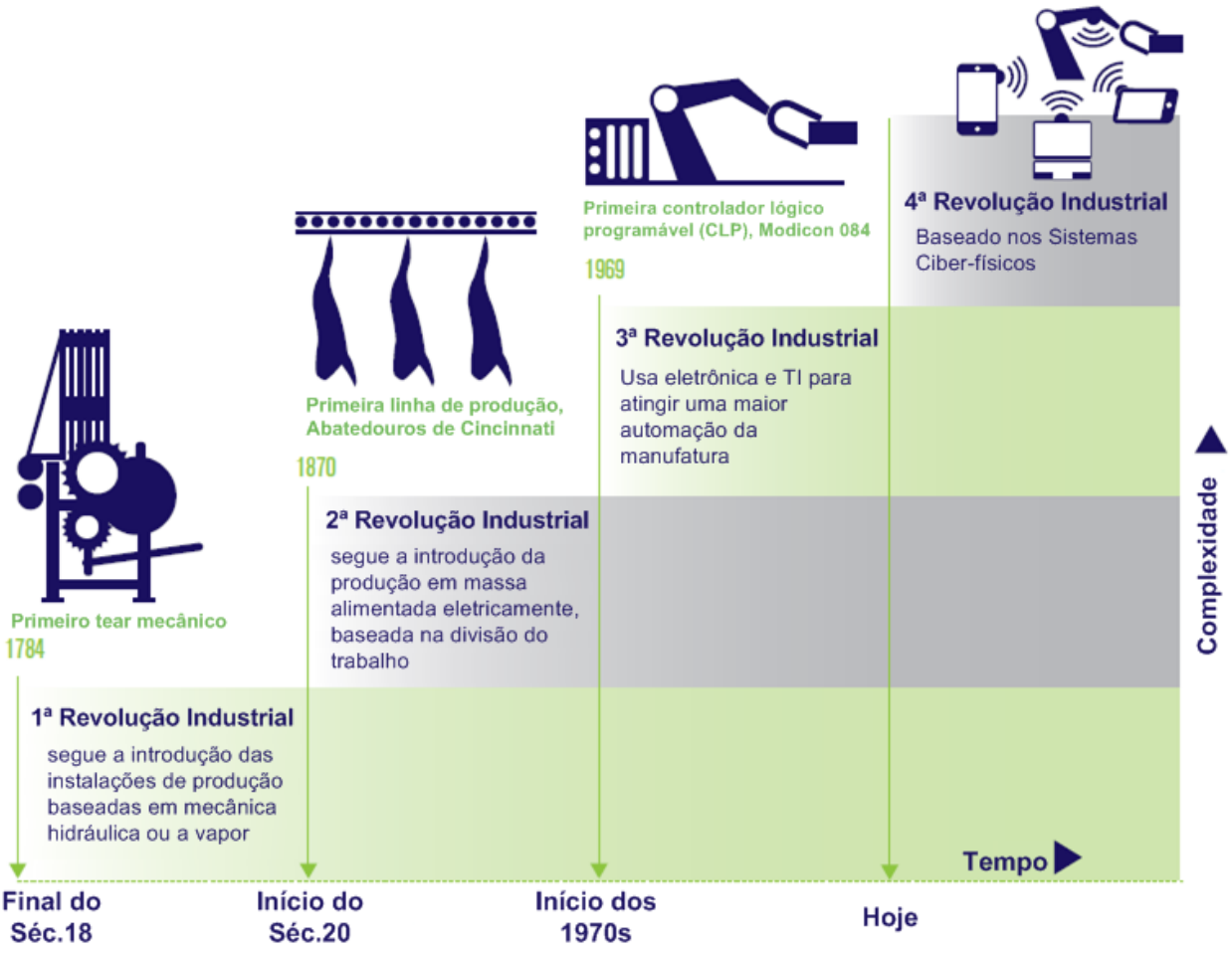

Figura 2.1: As quatro fases da revolução industrial, adaptado de (KAGERMANN; WAHLSTER; HELBIG, 2013)

Esse novo paradigma da indústria tem como principal propósito a consolidação das fábricas inteligentes onde de destacam a disseminação dos Sistemas Ciber-físicos (CPSs), evolução dos sistemas embarcados, permitindo a implementação da Internet das Coisas (IoT) e de Serviços(IoS), fazendo de cada insumo, recurso ou produto um objeto conectado e mapeável em rede.

Os Sistemas Ciber-físicos correspondem à evolução dos sistemas embarcados na forma de dispositivos distribuídos e conectados a redes sem fio com capacidade de comunicação entre si e com a Internet. Isso resulta na convergência entre o mundo físico e virtual (ciberespaço). Acompanhando esse movimento está o aprimoramento dos protocolos de rede como o IPv6, que provê endereços de rede suficientes para permitir conectividade com objetos inteligentes via Internet. Esse fato cria as condições para a criação da Internet das coisas e Serviços, em que recursos, informações, objetos e pessoas estarão conectados para propósitos específicos. Essa evolução tecnológica afeta a indústria, e no âmbito da manufatura é descrita como o quarto estágio da industrialização, ou Indústria 4.0 (KAGERMANN; WAHLSTER; HELBIG, 2013).

A Figura 2.2 fornece uma visão geral da Indústria 4.0, em que todos os elementos que compõem a cadeia de valor são mapeados virtualmente de modo que todos estejam acessíveis em qualquer lugar e possam se comunicar através de uma rede colaborativa com o propósito de suprir necessidades comuns.

Todos os elementos desse cenário estão na forma de serviços disponíveis através da Internet (KAGERMANN; WAHLSTER; HELBIG, 2013). Nesse sentido, a integração das tecnologias de IoT e Computação na nuvem é promissora por permitir a criação de ambientes inteligentes que combinam serviços oferecidos pelos múltiplos elementos dessa rede, e escalabilidade para suportar o amplo número de usuários de modo confiável e descentralizado (GUBBI et al., 2013).

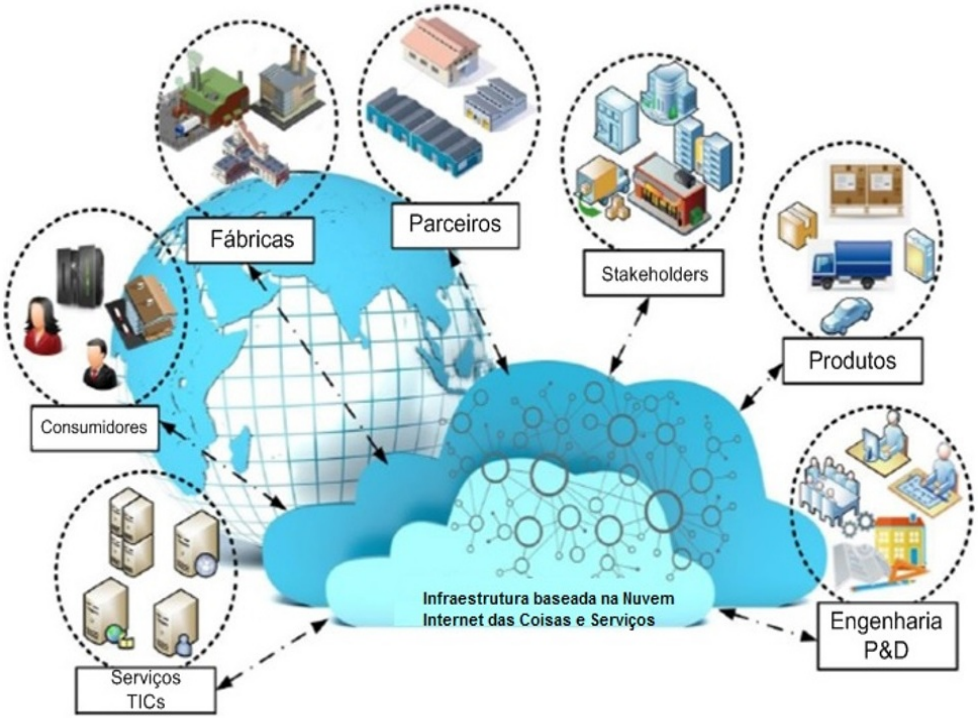

Figura 2.2: Visão geral da Indústria 4.0, adaptado de (PISCHING et al., 2015)

O principal propósito da Indústria 4.0 é a constituição das fabricas inteligentes. Estas são capazes de gerenciar a complexidade dos processos, operam com menos interrupções e de modo mais eficiente. Numa fábrica inteligente a interação entre seres humanos, máquinas e recursos é tão natural que, segundo Kagermann *et al.* (KAGERMANN; WAHLSTER; HELBIG, 2013), assemelha-se a uma rede social. Nessas instalações, sistemas ciber-físicos monitoram processos físicos da fábrica e tomam decisões descentralizadas. Os sistemas físicos são convertidos em objetos da Internet das coisas, comunicado-se e colaborando uns com os outros e com seres humanos em tempo real através de uma rede sem fio conectada à Web (MARR, 2016).

Segundo Marr (MARR, 2016) para uma fábrica ou sistema ser considerado parte da Indústria 4.0, deve possuir as características de:

a) interoperabilidade entre dispositivos, máquinas, sensores e pessoas;

b) transparência na informação, em que os sistemas contextualizam a informação criando uma copia do mundo físico no mundo virtual através de sensores de dados;

c) auxilio técnico, pois os sistemas podem auxiliar humanos na tomada decisões e na resolução de problemas, bem como auxiliar quando a tarefa é complexa e insegura para humanos.

Junto a isso, a possibilidade de realizar tomadas de decisões de modo descentralizado, destacando-se o papel dos sistemas ciber-físicos.

2.2 Sistemas Ciber-físicos (CPS)

Uma conceituação geral coloca os CPSs como sistemas formados por entidades virtuais colaborativas intensamente conectadas ao mundo físico circundante e aos seus processos em andamento, consumindo e fornecendo acesso a dados e serviços de processamento de dados disponíveis na Internet (ACATECH, 2011; NIST, 2013) . Conforme o *Report of the Steering Committee for Foundations and Innovation for Cyber-Physical* (NIST, 2013), os CPSs tem uma grande potencial para modificar vários aspectos da vida, permitindo o desenvolvimento de conceitos como carros autônomos, cirurgia robótica, edifícios inteligentes, rede elétrica inteligente, manufatura inteligente e dispositivos médicos implantados.

Especificamente no âmbito da manufatura, *Cyber-physical Systems* (CPS) correspondem à fusão dos sistemas físicos e virtual em uma rede integrada que reúne recursos físicos e processos (Gorecky et al., 2014; MacDougall, 2014). Nessas redes circulam múltiplas informações de diferentes fontes que são monitoradas e sincronizadas entre o chão-de-fábrica e o ciberespaço (Lee et al., 2015). Kagermann *et al.* (KAGERMANN; WAHLSTER; HELBIG, 2013) classificam como CPSs, máquina inteligentes e instalações de produção capazes de trocar informação de modo autônomo, realizar acionamentos e controlar uns aos outros de forma independente.

Segundo MacDougall (MACDOUGALL, 2014), os Sistemas Ciber-físicos criam as condições para a Internet das Coisas (Internet of Things , IoT) que combinada com a Internet de Serviços (IoS) viabilizam a Indústria 4.0. A interação entre sistemas embarcados baseados em software de alto desempenho e interfaces de usuário dedicadas integrados por redes digitais criam um novo universo de funcionalidades para sistemas. Com isso, os CPSs representam uma quebra de paradigma diante dos modelos de negócios existente ao permitir o desenvolvimento de novas aplicações, serviços e cadeia de valor. A evolução dos sistemas embarcados em Sistemas Ciber-físicos com Internet das Coisas, dados e serviços é ilustrada na Figura 2.3.

Na Figura 2.3 sistemas embarcados fechados (ex. aribag) representam o ponto de partida. Foram realizados desenvolvimentos no campo dos sistemas embarcados conectados a redes locais. Mais recentemente, estratégias foram elaboradas para expandir essas redes globalmente (e.g. junção de rodovias inteligente em redes que utilizam informações sobre o tráfego). Os CPSs representam a próximo estágio para a criação das cidades inteligentes através da criação de uma Internet das Coisas, Dados e Serviços.

Uma especialização do conceito de Sistemas Ciber-físicos são os Sistemas de Produção Ciber-físicos (CPPS, sigla em inglês de Cyber-physical production system). Estes compreendem máquinas, sistemas de armazenagem, logística e instalações de produção inteligentes que foram mapeados digitalmente e permitem a integração com base nas TICs de ponta-a-ponta, desde a logística de entrada até a logística de saída e serviços, passando pela produção e pelo marketing (KAGERMANN; WAHLSTER; HELBIG, 2013; MACDOUGALL, 2014; MONOSTORI, 2015). Também tem a função de possibilitar a comunicação entre máquinas, humanos e produtos. Os elementos de um CPPS serão capazes de adquirir e processar dados, autocontrolar certas tarefas e interagir com humanos através interfaces (MONOSTORI, 2015), conforme

Figura 2.3: A evolução dos sistemas embarcados até a Internet das Coisas, dados e Serviços, adaptado de (MACDOUGALL, 2014)

ilustra a Figura 2.4.

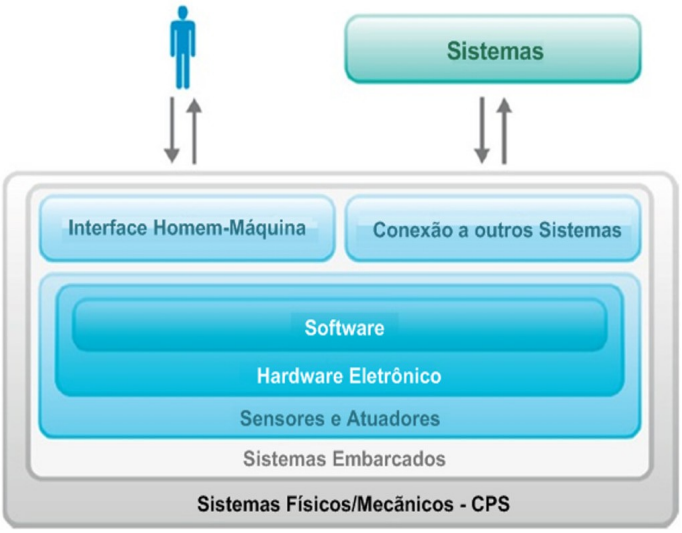

Figura 2.4: Interação humanos-máquinas em Sistemas de Produção Ciber-físicos, adaptado de (BRETTEL et al., 2014)

Monostori (MONOSTORI, 2015) lista algumas expectativas em relação aos CPSs e CPPSs:

- Robustez em todos os níveis;
- Auto-organização, auto-manutenção,
- auto-reparo;

- Segurança;
- Diagnostico remoto;
- Controle em tempo real
- Navegação autônoma;
- Transparência;
- Previsibilidade;
- Eficiência;
- Modelo correcional, etc.

O mesmo autor, em extensa revisão bibliográfica, chama de "raízes"dos CPPSs na manufatura trabalhos pioneiros realizados em área associadas ao tema, e faz considerações acerca dos desafios em termos de Pesquisa e Desenvolvimento no âmbito dos CPPSs. Dentre os estudos pré-revolução industrial 4.0, destacam-se:

- Sistemas de manufatura inteligente (IMS, sigla em inglês) em que através de informações incompletas ou imprecisas problemas imprevistos possam ser solucionados. Nesse campo também foram abordados temas como métodos de inteligência artificial e aprendizagem de máquina.
- Sistemas de Manufatura Biológicos (BMS, sigla em inglês) os quais trabalham com conceitos biologicamente inspirados como auto-crescimento, auto-organização, adaptação e evolução.
- Sistemas de Manufatura Reconfiguráveis (RMS, sigla em inglês), em que máquinas e controladores são reconfiguráveis.
- Fábricas Digitais (*Digital Factories*), em que destaca-se o mapeamento da maioria dos processos técnicos e de negócios no mundo digital.
- Sistemas de Manufatura Holônicos (HMS, sigla em inglês), em que as entidades apresentam como principais características a autonomia e a cooperação.
- Metodologias de síntese emergente.
- Estruturas de produção modificáveis.
- Empresas de manufatura responsivas e cooperativas.
- Coevolução de produtos, processos e sistemas de produção.
- Tratamento de complexidade em manufatura e engenharia.

Há um grande número de importantes desenvolvimentos no campo dos CPSs e CPPSs como em controle cooperativo, sistemas multi-agentes, sistemas de complexos adaptativos, sistemas emergentes, redes de sensores, *data mining*, etc; no entanto, muito pode ser realizado, Mosnostori (MONOSTORI, 2015) relaciona algumas desafios em termos de P&D:

- Sistemas autônomos (ao menos parcialmente) e adaptativos baseados em contexto.
- Sistemas de produção cooperativos.
- Sistemas dinâmicos de previsão e identificação.
- Programação robusta.
- Fusão de sistemas reais e virtuais.
- Simbiose homem-máquina (incluindo robô-humano).

2.3 Internet das Coisas (IoT)

2.3.1 Definição

A Internet das coisas (IoT), depois da própria internet, é considerada como uma onda tecnológica e econômica na indústria global da informação. Segundo Chen *et al*. (CHEN et al., 2014), a IoT é uma rede inteligente que conecta todas as coisas à Internet com a finalidade de trocar informações e localizar dispositivos através protocolos pré-estabelecidos. Para a IoT atinge o objetivo de promover uma localização inteligente, identificação, rastreamento, monitoramento e gestão de coisas. É a extensão e expansão do conceito de redes baseadas na Internet, a qual expande o paradigma da comunicação de pessoas para pessoas, para pessoas e coisas ou entre coisas e coisas.

No paradigma da Internet das Coisas (IoT), muitos dos objetos que são parte dos ambientes da vida moderna são conectados em rede de alguma forma. Tecnologias de IDentificação por Radio Frequência (RFID) e sensores em rede se proliferarão a fim viabilizar esse novo paradigma, em que Sistemas de Informação e Comunicação estarão imperceptivelmente incorporados ao ambiente ao redor. Isso resulta na geração de enormes quantidades de dados que precisam ser armazenados, processados e apresentados de forma continua, eficiente e facilmente interpretável (GUBBI et al., 2013).

Não há uma definição final e absoluta para IoT, uma vez que os especialistas na área possuem diferentes perspectivas sobre o assunto. Entretanto, seguindo a evolução tecnológica, é possível identificar algumas características que a IoT deve ter (CHEN et al., 2014):

1. Percepção Abrangente - usando recursos com RFID, sensores e códigos de barra para obter informações em qualquer lugar e a qualquer momento. Nessa perspectiva, os sistemas de comunicação e a informação serão embarcados de forma imperceptível e naturalmente no ambiente. As redes de sensores permitirá que as pessoas interajam remotamente com o mundo real. As tecnologias de identificação citadas identificarão tanto objetos quanto lugares.
2. Transmissão Confiável - Um conjunto de tecnologias de redes de rádio, redes de telecomunicação e a Internet permitirão a transmissão e a disponibilidade de informações entre objetos de rede a qualquer momento. IoT cria a interação entre o mundo físico, o mundo virtual, o mundo digital e a sociedade.

3. Processamento Inteligente - darão suporte às aplicações para IoT através da coleta de dados para bancos de dados, de várias tecnologias de computação inteligente, incluindo a computação na nuvem. Provedores de serviços de rede podem processar um grande volume de mensagens instantaneamente através da computação na nuvem. Assim, essa tecnologia será a promotora da IoT.

A Figura 2.5 contextualiza a IoT no âmbito da Industria 4.0. A associação entre Sistemas Ciber-físicos (CPS) e IoT propõe a virtualização e o monitoramento de recursos físicos e serviços, tornando-os objetos virtuais o quanto for possível, a fim de possibilitar a integração do mundo virtual com o real (KAGERMANN; WAHLSTER; HELBIG, 2013). Produtos, pessoas, objetos e serviços tornam-se parte de um grande grupo de serviços virtuais, em um ambiente onde as atividades são mapeadas na forma de serviços (everything-as-services) (LI; WEI, 2012). Com isso, no setor industrial, processos de manufatura também podem ser convertidos para serviços virtuais (Manufacturing Processes as Services). Todos esses serviços convergem para uma Internet de Serviços (IoS), integrados sob tecnologias baseadas na nuvem, permitindo que esses recursos sejam compartilhados localmente ou entre parceiros distribuídos (PISCHING et al., 2015).

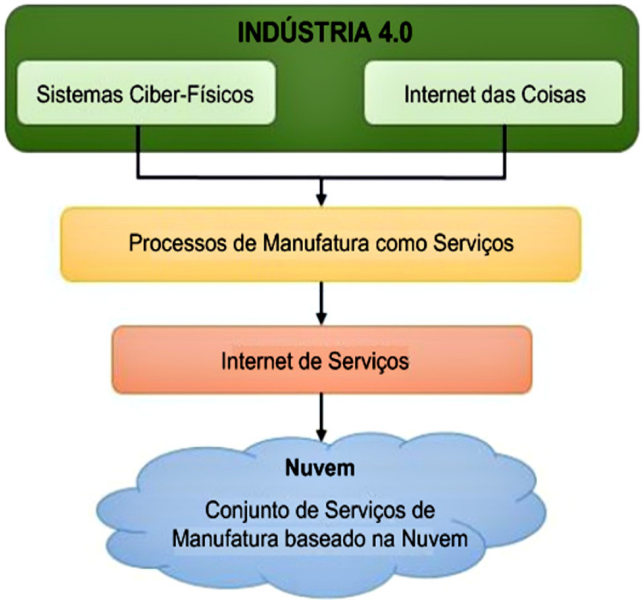

Figura 2.5: IoT no contexto da Industria 4.0, adaptado de (PISCHING et al., 2015)

Uma arquitetura ou um modelo de referencia para o desenvolvimento da IoT consiste de varias camadas, que compreendem a camada de aquisição de dados de campo, mais próxima do equipamentos, até a camada de aplicação de mais alta nível em temos de interface de dados. Segundo Bandyopadhyay e Sen (BANDYOPADHYAY; SEN, 2011), uma arquitetura em camadas tem que ser projetada de forma que possa atender as necessidades de varias tipos de organização, instituições e sociedades. Com esse propósito Atzori e Morabito (ATZORI L; MORABITO, 2010) apresentaram uma arquitetura genérica para IoT, ilustrada pela Figura 2.6.

A arquitetura em camadas é dividida em duas, um bloco de camadas inferior, representado por aquelas com a função de capturar dados, e um bloco superior responsável pela utilização dos dados em aplicações.

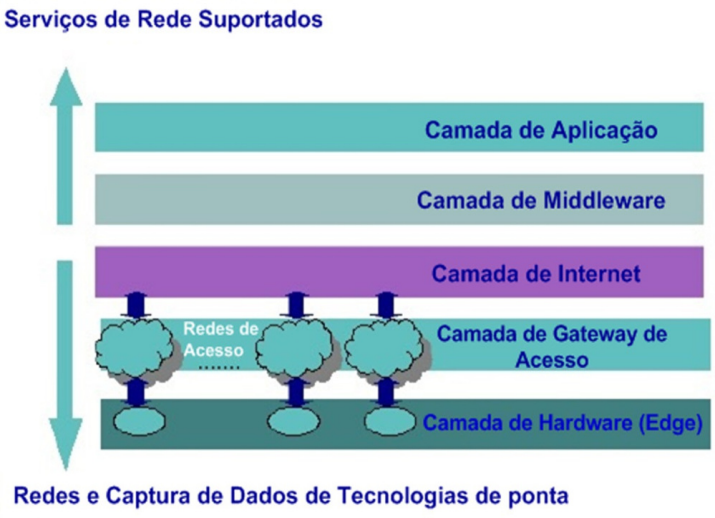

Figura 2.6: Arquitetura em camadas para IoT, adaptado de (ATZORI L; MORABITO, 2010)

Cada camada é brevemente descrita abaixo:

Camada de Borda (*Edge Layer*): é a camada de hardware que consiste em sensores de rede, sistemas embarcados, *Tags* de RFID (*Radio-Frequency Identification*) e leitores ou sensores de precisão de diferentes tipos. Muitos desses elementos de hardware fornecem identificação e armazenamento de informação (e.g. Tags RFID), coleta de informação (e.g. redes de sensores), processamento de informação (e.g. processadores de borda embarcados), comunicação, controle e atuação.

Camada de *Gateway* de Acesso (*Access gateway layer*): corresponde ao primeiro estágio da manipulação de dados. Essa camada trata do roteamento de mensagens, publicação e subscrição, e também realiza a comunicação entre plataformas, se necessário.

Camada *Middleware* (*Middleware layer*): Essa é umas das camadas mais criticas e opera de modo bidirecional. Atua como uma interface entre a camada hardware e a camada de aplicação. É responsável por funções criticas como gerenciamento de dispositivos e informação e também trata de questões como filtragem de dados, agregação de dados, análise semântica, controle de acesso, descoberta de informações como o serviço de informação EPC (*Eletroninc Product Code*) e ONS (*Object Naming Service*).

Camada de Aplicação (*Application Layer*) : Esta camada corresponde ao topo da pilha e tem a função de fornecimento de várias aplicações para diferentes usuários em IoT. Essas aplicações podem prover de diferentes industrias verticais como: manufatura, logística, saúde, segurança pública, alimentação, etc.

2.3.2 Elementos e Tecnologias-chave da IoT

Gubbi *et al.* (GUBBI et al., 2013) classificam as tecnologias que permitirão o desenvolvimento da IoT em três categorias: (a) Hardware - sensores, atuadores e hardware de comunicação embarcada; (b) *Middleware* - ferramentas de computação e armazenamento sob demanda para análise de dados e (c) Apresentação - novas ferramentas de visualização e interpretação fáceis de entender que podem ser amplamente acessadas em diferentes plataformas e que podem ser projetadas para diferentes aplicações. Algumas tec-

nologias que enquadram-se nessas categorias são:

Identificação por Radio Frequência (RFID) : o principio dessa tecnologia são microchips projetados para permitir a comunicação de dados sem fio. Esses componentes ajudam na identificação de tudo a que elas estejam fixadas, atuando como um código de barras eletrônico (WELBOURNE et al., 2009; JUELS, 2006). Esse recurso já resultou em muitas aplicações, em particular no setor de varejo e na gestão da cadeia de suprimentos. Outras aplicações podem ser encontradas em área de transporte (substituição de bilhetes, adesivos de registro) e aplicações de controle de acesso. Etiquetas passivas estão sendo usadas também em muitos cartões bancários e em pedágios rodoviários, que estão entre as primeiras implementações globais. Os leitores de RFID ativos têm seu próprio suprimento de bateria e podem instanciar a comunicação. Das várias aplicações, a principal aplicação de etiquetas RFID ativas está em contêiner de porto (JUELS, 2006) para monitoramento de carga.

Redes de sensores sem fio (WSN) : é centrada na implementação de dispositivos miniatura de baixa potência e de baixo custo para utilização em aplicações de detecção remota. Avanços tecnológicos no campo dos circuitos integrados de baixa potência e da comunicação sem fio tem permitido a viabilidade na utilização de uma rede de sensores constituída por um grande número de sensores inteligentes, permitindo a coleta, o processamento, análise e a distribuição de informações valiosas, capturadas em vários ambientes (AKYILDIZ et al., 2002). Os dados dos sensores são compartilhados entre nós de sensores e enviados para um sistema distribuído ou centralizado para análise.

Esquemas de Endereçamento - A capacidade de identificar exclusivamente "coisas"é fundamental para o sucesso do IoT. Isso não só nos permitirá identificar de forma exclusiva bilhões de dispositivos, mas também para controlar dispositivos remotos através da Internet. Para tratar disso, o sistema URN (*Uniform Resource Name*) é considerado fundamental para o desenvolvimento de IoT. URN cria réplicas dos recursos que podem ser acessados através do URL. Com grande quantidade de dados espaciais sendo coletados, muitas vezes é muito importante aproveitar os benefícios dos metadados para transferir as informações de um banco de dados para o usuário através da Internet (HONLE et al., 2005). O IPv6 também dá uma ótima opção para acessar recursos de forma exclusiva e remota.

Análise e Armazenamento de dados - uma das mais importantes funções desse campo emergente é tratar da grande quantidade de dados criada. O armazenamento, a propriedade e a expiração dos dados tornam-se questões críticas. Os dados têm de ser armazenados e utilizados de forma inteligente para um monitoramento e uma atuação inteligentes. É importante desenvolver algoritmos de inteligência artificial que poderiam ser centralizados ou distribuídos com base na necessidade. Esses sistemas apresentam características como a interoperabilidade, integração e comunicação adaptativas. Eles também têm uma arquitetura modular, tanto em termos de design do sistema de hardware, bem como pelo desenvolvimento de software e geralmente são bem adaptáveis a aplicações de IoT. Mais importante ainda, uma infraestrutura centralizada para suportar o armazenamento e análise é necessária (GUBBI et al., 2013).

Tecnologias de Visualização - a visualização é crítica para uma aplicação IoT, pois isso permite a interação do usuário com o ambiente. Com os recentes avanços nas tecnologias de tela sensível ao toque, o uso de *tablets* e *smartphones* tornou-se muito intuitivo. Para que uma pessoa leiga se beneficie plenamente da IoT, é preciso criar uma visualização atraente e fácil de entender. A evolução tecnológica permite que mais informações sejam fornecidas de forma significativa para os consumidores finais. A conversão

de dados em conhecimento é crítico para as tomadas de decisões rápidas. No entanto, a extração de informações significativas de dados brutos não é trivial. Isto engloba tanto a detecção de eventos como a visualização dos dados brutos e modelados associados, com a informação representada de acordo com as necessidades dos usuários finais (GUBBI et al., 2013).

2.3.3 Aplicações da IoT

Existem vários domínios de aplicação que serão impactados pela emergente Internet das coisas. Na literatura as aplicações atualmente em curso e as potenciais implementações no campo da IoT são categotizadas de diferentes formas. Chen *el al.* (CHEN et al., 2014), em relação a Gubbi *et al.* (GUBBI et al., 2013), e Bandyopadhyay e Sen (BANDYOPADHYAY; SEN, 2011), apresentam a classificação mais objetiva, porém mais abrangente das capacidades de aplicação da IoT. Dessa forma, o potencial de aplicação da IoT pode ser resumido como segue:

- *Detecção de localização e compartilhamento de informações de localização*: O sistema de IoT pode coletar informações de localização de terminais e nós de IoT, e então, fornecer serviços com base nas informações de localização coletadas. As informações de localização incluem informações de posição geográfica de GPS, Cell-ID, RFID, etc. e informações de posições relativas ou absolutas entre as coisas.

- *Sensoriamento do ambiente (Environment Sensing)*: o sistema de IoT pode coletar e processar todos os tipos de parâmetros ambientais físicos ou químicos através dos terminais localmente ou amplamente implantados. Informações ambientais típicas incluem temperatura, umidade, ruído, visibilidade, intensidade de luz, espectro, radiação, poluição (CO, CO2, etc.), imagens e indicadores de corpo.

- *Controle Remoto*: Sistemas IoT podem controlar terminais IoT e executar funções com base nos comandos da aplicação combinados com informações coletadas de requisitos de serviço e coisas.

- *Redes Ad-Hoc*: sistema IoT terá capacidade de rede de auto-organizada rapidamente e pode interoperar com a camada de rede/serviço para prover serviços relacionados (HUANG et al., 2014). E uma rede do veículo, a fim de transferir os dados, a rede entre os veículos e/ou infra-estruturas rodoviárias podem ser rapidamente auto-organizadas.

- *Comunicação segura*: sistema IoT pode ainda estabelecer um canal de transmissão de dados seguro entre a aplicação ou plataforma de serviço e terminais IoT com base nos requisitos do serviço.

Na prática, uma aplicação IoT consiste em diferentes tipos de recursos, e podem ser baseadas nas exigências da área de aplicação. O quadro da Figura 2.7 mostra exemplos de diferentes aplicações da IoT por área.

		Detecção e Compatilhamento de Localização	Sensoriamento de Ambiental	Controle Remoto	Rede *Ad hoc*	Comunicação Segura
E-Saúde	Monitoramento	•	•		•	•
	Home care	•	•			•
Sistema de Transporte Inteligente	Frota Inteligente	•	•			•
	Automotivo	•	•	•	•	•
Cidade Inteligente	Monitoramento Ambiental	•	•			•
	Segurança	•	•			•
	Rastreamento de Alimento	•				•
	Agricultura Inteligente	•	•	•		•
Indústria	Monitoramento de Processo		•	•		•
	Gestão Logística	•				•

Figura 2.7: Áreas tecnológicas-chave para o desenvolvimento da IoT, adaptado de (CHEN et al., 2014)

2.4 Fábricas Inteligentes - *Smart Factories*

O principal objetivo da Industria 4.0 é consolidar as "Fábricas Inteligentes"para permitir a criação de produtos e serviços personalizados sob-demanda que atendam às necessidades de cada consumidor (PISCHING et al., 2015).

A fusão dos mundos físicos e virtual através dos sistemas ciber-físicos e a fusão resultante de processos de negócios e processos técnicos estão abrindo caminho para uma nova era industrial sintetizada pelo conceito de "Fábrica Inteligente"(*Smart Factory*) no âmbito da estratégia Industria 4.0.

A implementação de sistemas de Ciber-físicos em sistemas de produção faz emergir as "fábricas inteligentes". Processos, recursos e produtos de fábricas inteligentes são caracterizados por sistemas de ciber-físicos, fornecendo maior eficiência e vantagens em termos de custos, em comparação com sistemas de produção clássicos. As Fábricas de inteligentes são projetadas conforme práticas de negócios sustentáveis e orientadas a serviço. Essas fábricas primam por adaptabilidade, flexibilidade, auto-adaptabilidade e características de aprendizagem, tolerância a falhas e gestão de riscos (MACDOUGALL, 2014).

Altos níveis de automação é inerente às fábricas inteligentes. Isso é possível graças a uma rede flexível de sistemas de produção baseados em sistemas ciber-físicos que, em grande medida, supervisionam automaticamente os processos de produção. Sistemas de produção flexíveis, que são capazes de responder em quase tempo real, permitem que os processos de produção internos sejam radicalmente otimizados. As vantagens de produção não estão limitadas exclusivamente às condições de produção pontuais, mas também podem ser otimizadas de acordo com uma rede global de unidades de produção adaptáveis e auto-organizáveis pertencentes a mais de um operador (Figura 2.8) (MACDOUGALL, 2014).

Isso representa uma revolução na produção em termos de inovação, custo, economia, tempo e a criação de um modelo *bottom-up* (de baixo para cima) de geração de valor na produção, cuja capacidade de rede

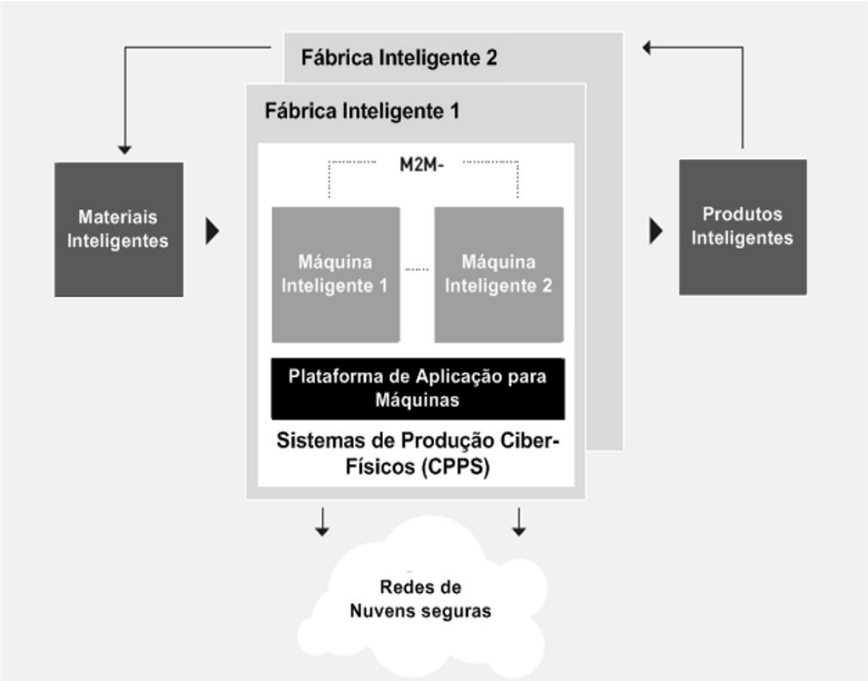

Figura 2.8: Esquema de Fábricas Inteligentes na Industria 4.0, adaptado de (MACDOUGALL, 2014)

cria novas e mais oportunidades de mercado. A produção na fábrica inteligente traz consigo inúmeras vantagens sobre a produção e manufatura convencionais (MACDOUGALL, 2014). Essas incluem:

- Processos de produção CPS-otimizado: "unidades"de fábrica inteligente são capazes de determinar e identificar seu(s) campo(s) de atividade, as opções de configuração e as condições de produção, bem como se comunicar de forma independente e sem fio com outras unidades;
- Manufatura otimizada de produtos customizados para o cliente através da compilação inteligente para o sistema de produção ideal que considera fatores do produto, custos, logística, segurança, confiabilidade, tempo e considerações de sustentabilidade;
- Produção eficiente de recurso;
- Ajustes à força de trabalho humana para que a máquina se adapte ao ciclo de trabalho humano.

2.5 Potenciais Aplicações na Indústria 4.0

O conjunto de exemplos de implementações do paradigma Indústria 4.0 mostrado neste tópico foi extraído do Relatório Final do Grupo de trabalho para a Indústria 4.0 (KAGERMANN; WAHLSTER; HELBIG, 2013), que trata das recomendações para implementações da iniciativa Industria 4.0. Com isso, os exemplos mencionados aqui terão mais um efeito orientativo, do que expor desenvolvimentos práticos da indústria ou meio acadêmico em curso, uma vez que os casos nesse âmbito ainda são escassos e estão em um estágio experimental.

Os exemplos de aplicações citados tratam especificamente de direções que podem ser tomadas para as seguintes situações: eficiência energética em linhas de produção; infraestrutura de sistemas de engenharia de ponta-a-ponta na cadeia de valor; manufatura customizada; telepresença e; substituição de fornecedores durante crises no controle da produção.

O relatório menciona a aplicação de ações para a redução do consumo de energia em linha de montagem de veículos enquanto a mesma não está em uso. Segundo o documento (KAGERMANN; WAHLSTER; HELBIG, 2013), atualmente muitas linhas de produção ou partes delas continuam trabalhando e consumindo energia durante intervalos, finais de semana e turnos onde não há produção. Por exemplo, 12% do consumo total de energia de uma linha de montagem do corpo do veículo que usa a tecnologia de soldagem a *laser* ocorre durante as pausas na produção. A linha opera cinco dias por semana em um padrão de três turnos. O equipamento permanece ligado nos finais de semana para que no começo da semana possa ser ligado imediatamente. Os 90% do consumo de energia durante as pausas na produção é representado por: robôs (20% a 30%), extratores (35% a 100%) e fontes de *laser* e seus sistemas de resfriamento (0% a 50%).

Medidas para alavancar o potencial de eficiência energética incluem os casos de robôs que serão desligados durante o processo, mesmo durante curtas pausas na produção. Durante pausas mais longas na produção, eles entrarão em um tipo de modo de espera, conhecido como modo *Wake-On-LAN*. Os extratores usarão motores de velocidade controlada que podem ser ajustados para atender a restrições. No caso das fontes de *laser*, sistemas completamente novos são a única maneira de fornecer melhorias. Em conjunto, estas medidas permitem uma redução de 12% do consumo total de energia (de 45.000kWh/w para cerca de 40.000kWh/w), juntamente com um corte de 90% no consumo de energia durante as pausas de produção (KAGERMANN; WAHLSTER; HELBIG, 2013).

No âmbito da integração da cadeia de valor, os sistemas de suporte de TI trocam informações através de uma variedade de interfaces, mas só podem usar essas informações quando tratar de casos individuais específicos. Não há uma visão global da perspectiva do produto que está sendo fabricado. Como resultado, os clientes não podem selecionar livremente todas as funções e recursos de seus produtos, mesmo que tecnicamente seja possível permitir que eles façam isso.

O desenvolvimento baseado em modelos através da CPS permite a implantação de uma metodologia de ponta-a-ponta, modelada e digital que cobre todos os aspectos, desde as necessidades dos clientes até a arquitetura e a fabricação do produto final. Isso permite que todas as interdependências sejam identificadas e representadas em uma cadeia de ferramentas de engenharia de ponta-a-ponta. O sistema de produção é desenvolvido em paralelo com base nos mesmos paradigmas, o que significa que ele sempre acompanha o ritmo do desenvolvimento do produto. Como resultado, torna-se viável fabricar produtos individuais (KAGERMANN; WAHLSTER; HELBIG, 2013).

Outro exemplo analisado trata da customização da manufatura. Nas cadeias de valor dinâmicas na Indústria 4.0 é possível a coordenação do projeto, configuração, pedidos, planejamento, produção e logística de produtos específicos para clientes específicos. Isso também fornece a oportunidade de incorporar pedidos de última hora para que as alterações imediatamente antes ou mesmo durante a produção. A indústria automotiva atual, por exemplo, é formada por linhas de produção estáticas de difícil reconfiguração para a produção de diferentes variedades de um produto. Software MES (*Manufacturing Execution Systems*) tem sua funcionalidade baseada nos equipamentos da linha de produção e, portanto, caracterizado-se como es-

tático também. O trabalho dos colaboradores é determinados pelo ritmo da linha de montagem e, portanto, normalmente é monótono. Como consequência disso, não é possível incorporar solicitações individuais de cada cliente para incluir elementos de outro grupo de produtos feito pela mesma empresa (KAGERMANN; WAHLSTER; HELBIG, 2013). Na era da Indústria 4.0 aparecem as linhas de produção dinâmicas. A reconfiguração dinâmica de linhas de produção possibilita a mistura e a combinação dos equipamentos que compõem um carro, por exemplo. Além disso, variações individuais (por exemplo, montagem de uma cadeira de outra série de veículo) podem ser implementadas em qualquer momento em resposta a questões logísticas (tais como gargalos) sem serem limitados por intervalos de tempo definidos centralmente. As soluções de TI para sistemas MES agora constitui um componente central do início ao fim - desde o projeto até a montagem e a operação.

No que se refere aos sistemas de Telepresença (serviços remotos) a promessa da Indústria 4.0 é que os sistemas de manufatura funcionem como "máquinas sociais- em redes similares as redes sociais - e se conectarão automaticamente a plataformas de telepresença baseadas na nuvem, a fim de localizar os especialistas adequados para lidar com um problemas específico (KAGERMANN; WAHLSTER; HELBIG, 2013). Por exemplo, especialistas poderão então utilizar plataformas de conhecimento integradas, ferramentas de videoconferência e métodos de engenharia aperfeiçoados para executar serviços de manutenção remotos tradicionais de forma mais eficiente através de dispositivos móveis. Além disso, máquinas poderão aumentar e expandir continuamente as suas próprias capacidades conforme a situação exigir, atualizando ou carregando automaticamente as funções e dados relevantes através de canais de comunicação padrão e seguros com as plataformas de telepresença.

Nos casos em que o problema a ser tratado são mudanças emergências de fornecedor devido falhas no controle da produção, há estratégias pensadas para solução de tal problema no escopo das soluções das Fábricas do Futuro. Atualmente, falhas súbitas dos fornecedores resultam em custos adicionais significativos e atrasos na produção e, portanto, envolvem grandes riscos para as empresas. Eles precisam tomar decisões rápidas sobre qual fornecedor alternativo usar como cobertura.

Na Indústria 4.0 será possível simular todas as etapas do processo de fabricação e demonstrar a influência de cada uma na produção. Isso inclui a simulação nos níveis de estoque, transporte e logística; a capacidade de rastrear o histórico de uso de componentes e a aquisição de informações relativas ao tempo que os componentes podem ser mantidos até expirarem. Isto permitirá calcular os custos de montagem específicos do produto e reduzir ao mínimo a reconfiguração dos recursos de produção. Também será possível avaliar os riscos relevantes, em que serão simulados os diferentes custos e margens de fornecedores alternativos, incluindo a simulação do impacto ambiental associado ao uso de um fornecedor sobre outro. Todas essas operações baseadas na "Nuvem"de fornecedores (KAGERMANN; WAHLSTER; HELBIG, 2013).

2.6 Desafios para a Indústria 4.0

A implementação da visão da Indústria 4.0 envolverá um processo que progredirá a taxas diferentes em empresas e setores individuais. Tendo como referencia indústria alemã, uma das mais preparadas para a efetiva implantação das fábricas inteligentes, realizou-se uma pesquisa no ano de 2013 pelas associa-

ções BITKOM, VDMA e ZVEI, e verificou-se que 47% das empresas pesquisadas disseram que estavam ativamente envolvidas na Indústria 4.0, 18% dessas empresas declararam que estavam envolvidas na investigação sobre o tema e 12% afirmaram que estão colocando o paradigma em prática (KAGERMANN; WAHLSTER; HELBIG, 2013). Com base nessa mesma pesquisa, foram identificados os três principais desafios relacionados à implementação da visão da Indústria 4.0, que envolve: padronização, organização do trabalho e disponibilidade de produto.

O principal grupo de trabalho envolvido no desenvolvimento da estratégia "Indústria 4.0"(KAGERMANN; WAHLSTER; HELBIG, 2013) também considera que as seguintes medidas são fundamentais para permitir uma transição suave para a Indústria 4.0 por parte das empresas:

- A implementação de soluções CPS habilitadas para tempo real gerará grandes demandas pela disponibilidade de serviços e infraestrutura de rede em termos de espaço, qualidade técnica e confiabilidade.
- Os processos de negócios na manufatura ainda são estáticos e são implementados por meio de sistemas de software extremamente inflexíveis. No entanto, eles não podem simplesmente ser substituídos subitamente por sistemas orientados a serviços. Será essencial integrar as novas tecnologias nas mais antigas (ou vice-versa) - os sistemas antigos precisarão ser atualizados com sistemas com suporte a tempo real. Desafio que possui uma relação com o problema deste trabalho, que propõe a incorporação de um módulo de monitoramento *on-line* junto a um sistema de teleoperação com recursos previamente implementados.
- A taxa de desenvolvimento de novos modelos de negócios para a fabricação na Internet das Coisas e Serviços se aproximará da taxa de desenvolvimento e dinamismo da própria Internet.
- Os funcionários estarão envolvidos desde os primeiros estágios do projeto sócio-técnico inovador da organização do trabalho, CPD e do desenvolvimento tecnológico.
- Para conseguir a transição para a Indústria 4.0, será necessário que a indústria das TICs (que está acostumada a ciclos de inovação curtos) trabalhe em estreita colaboração com os fabricantes de máquinas e fábricas e fornecedores de sistemas mecatrônicos (que tendem a pensar em termos de ciclos de inovação mais longos), a fim de desenvolver modelos de negócios aceitáveis para todos os parceiros.

Capítulo 3

Revisão Bibliográfica: Conceitos e Tecnologias Fundamentais

A emergente IoT industrial requer a adoção de um conjunto de novos padrões para que modelos de referência sejam desenvolvidos e, consequentemente, a Industria 4.0 ser consolidada mais rapidamente. Albert (ALBERT, 2015) aponta como padrões-chave para a construção desse caminho o MTConnect, matido pelo *MTConnect Institute*, e o OPC-UA, evolução do OPC clássico, desenvolvido pela *OPC Foundation*.

3.1 MTConnect

Na conceituação de Willian Sobel (SOBEL, 2010), colaborador do MTConnect Institute, o MTConnect é um padrão baseado em um protocolo aberto para a integração de dados. "MTConnect não pretende substituir as funcionalidades do produtos existentes, mas atua para aprimorar as capacidades de aquisição de dados de dispositivos e aplicações, e move-se no sentido de criar um ambiente *plug-and-play* que represente redução de custo de integração".

Esse protocolo foi desenvolvido com base nos mais dominantes padrões da manufatura e da indústria de software, o que maximiza o número de ferramentas disponíveis para sua implementação e fornece um maior nível de interoperabilidade com outros padrões e ferramentas nessas indústrias (SOBEL, 2010).

O MTConnect veio responder a um dilema industrial. Uma instalação de manufatura típica tem centenas de máquinas e sistemas operacionais independentes associados para assegurar um produto fabricado no tempo, com a qualidade e o custo efetivos. Essas máquinas acumulam informações sobre sua operação e, frequentemente, estão impossibilitadas de se comunicarem com outros dispositivos. Mesmo havendo exceções, esse é o caso preponderante, é difícil comunicar informações processar dados entre essas máquinas e sistemas. O resultado disso é uma maior dificuldade na otimização, coordenação e rastreamento de dados para assegurar que máquinas, fábricas e sistemas operem em um nível aceitável (RANGARAJAN; DORNFELD; WRIGHT, 2003).

No entendimento de Silva et al. (SILVA et al., 2010), há muitos dados disponíveis no chão-de-fábrica, porém, o fato de as plantas de fabricação utilizarem máquinas de diferentes fabricantes implica no uso de

diferentes padrões de entrada e saída de dados, o que torna trabalhosa a tarefa de projetar e configurar um sistema que possa se comunicar e coletar dados sobre o funcionamento de todas as máquinas ao mesmo tempo. Isso faz com que sistemas de monitoramento tenham que ser projetados especialmente para cada planta, representando maiores custos e menor eficiência.

Há a necessidade de uma interface padronizada para máquinas-ferramenta e outros equipamentos que garantam estreita integração e interoperabilidade. O MTConnect é um padrão que surgiu para tentar responder a essa necessidade ao reduzir a questão das "ilhas"de tecnologia (Fig.3.1) e criar um canal de comunicação entre/para máquinas-ferramenta que utilizem uma linguagem comum, conforme ilustra a Fig.3.2.

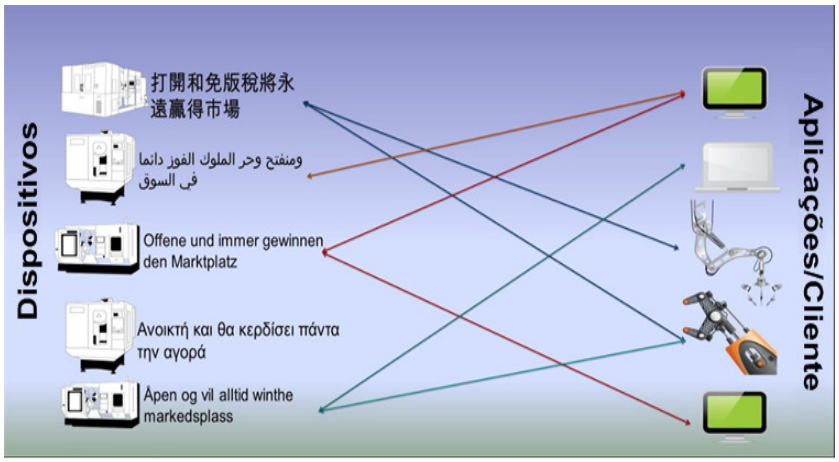

Figura 3.1: Máquinas de diferentes fabricantes baseadas em padrões proprietários, adaptado de (EDSTROM, 2013)

Figura 3.2: MTConnect como um padrão universal para conectar equipamentos de manufatura com aplicações, adaptado de (EDSTROM, 2013)

O MTConnect foi criado com o propósito de se tornar um padrão entre os fabricantes de máquinas CNC, como são o Bluetooth e o USB em suas respectivas áreas (WARNDORF; FOX; SOBEL, 2007). A

adoção de um formato único para distribuição de dados facilita grandemente a implementação de sistemas de monitoramento e análise dados. O fato de o protocolo ser baseado em HTTP permite o monitoramento remoto e através da Internet de vários dispositivos de linhas de produção simultaneamente.

O padrão MTConnect é baseado em XML, que é um padrão amplamente utilizado e oferece uma representação flexível para troca de dados semi-estruturados. É aberto e livre de royalties, para garantir que ele seja utilizado em seu máximo potencial. A interoperabilidade permitida pelo protocolo estimula o desenvolvimento de hardware e software por terceiros, tornando processo de manufatura mais produtivo. Essa abordagem permite a conectividade do nível mais básico do processo produtivo, próximas às peças e máquinas no chão-de-fábrica, até as ferramentas no nível de projeto e planejamento de processo (VIJAYARAGHAVAN et al., 2008). A Figura 3.3 apresenta uma visão resumida da utilização do MTConnect na integração de um sistemas de manufatura para a interoperabilidade inter e intra sistemas.

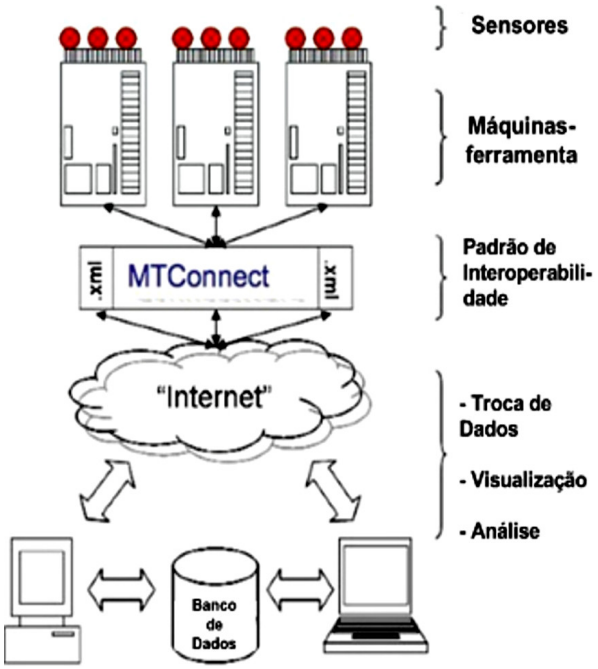

Figura 3.3: Integração de Sistema de Manufatura usando MTConnect, adaptado de (VIJAYARAGHAVAN et al., 2008)

Tipicamente as arquiteturas de sistemas que empregam MTConnect são formadas por alguns componentes que caracterizam o padrão. Eles são:

Adaptador

Como o MTConnect é uma padrão em desenvolvimento, a grande maioria dos CNCs são incompatíveis com ele. Para implementar o protocolo com essas máquinas é necessário o uso de um elemento adaptador. Em geral, o adaptador é um software que tem a função de realizar a "tradução"do formato proprietário utilizado pelo CNC para o padrão MTConnect.

Agente

Este tem a função de receber, armazenar e dar uma destinação, mediante requisição HTTP, de da-

dos fornecidos pelo adaptador. O *agente* é o elemento central do MTConnect. Os dados coletados por ele são arranjados no formato XML e repassados para aplicações cliente, que os utilizam para diferentes finalidades.

Dispositivo

São equipamentos ou grupo de componentes capazes de fornecer dados para a aplicação. Um dispositivo é uma entidade que possui pelo menos um controlador conduzindo sua operação.

Cliente

O Cliente representa todas as aplicações (*mobile App*, aplicações Web para *browser*,etc) com capacidade de conectar o Agente através de requisições HTTP a fim de obter dados para diferentes finalidades. De acordo com Silva *et al.* (SILVA et al., 2010), o padrão MTConnect foi desenvolvido com uma estrutura para que clientes possam ser desenvolvidos sem a preocupação com o formato em que os dados foram coletados.

O MTConnect é um padrão desenvolvido com base em uma arquitetura modular, isso permite o uso de diferentes ferramentas e métodos para conectar os dispositivos.

De acordo com Rondon (RONDON, 2010), é possível identificar até quatro diferentes arquiteturas para implementar o padrão (Fig. 3.4). Considerando a primeira opção de arquitetura (a) apresentada na Figura 3.4, observa-se o caso ideal de implementação da arquitetura, em que o dispositivo é complemente compatível com o MTConnect, tendo o Agente como elemento nativo do controlador. Na arquitetura (B) o Adaptador é incorporado ao dispositivo, mas o Agente é instalado como um servidor em um hardware separado. As arquiteturas (C) e (D) ilustram os casos em que os dispositivos não são compatíveis com o MTConnect, ou são dispositivos legados. Nesses casos é necessário programar o Agente e o Adaptador. Sendo que deve ser desenvolvido um Adaptador para cada modelo de CNC, por possuírem protocolos de comunicação proprietário.

A estrutura de dados que o Agente transmite para uma aplicação cliente é descrito em um documento XML de nome Devices.xml (Fig.3.5). Este arquivo é estruturado de forma que caracterize um controlador (*Device*), com seus componentes (Components), e dados (DataItems) gerados por eles. O arquivo mapeia a estrutura do dispositivo real. Para que um dado possa ser enviado para o Agente, ele precisa estar descrito no documento, caso contrário resulta em um erro de requisição HTTP. Sobel (SOBEL, 2010) lista e define os principais elementos que compõe o arquivo *Devices.xml* e componentes do padrão MTConnect, alguns dos quais são descritos abaixo:

Alarm - indica que um evento quer requer atenção e indica um desvio da operação normal.

Application - um processo ou um conjunto de processos que acessa o Agente MTConnect para realizar alguma tarefa.

Component - Parte de um dispositivo que contem subcomponentes ou itens de dados. Os componentes são blocos que compõe os dispositivos.

Probe - Uma requisição que resulta da descrição da configuração e da composição do dispositivo.

Stream - Coleção de *Events*, *Condition* e *Samples* organizados por Dispositivos (*Devices*) e Compo-

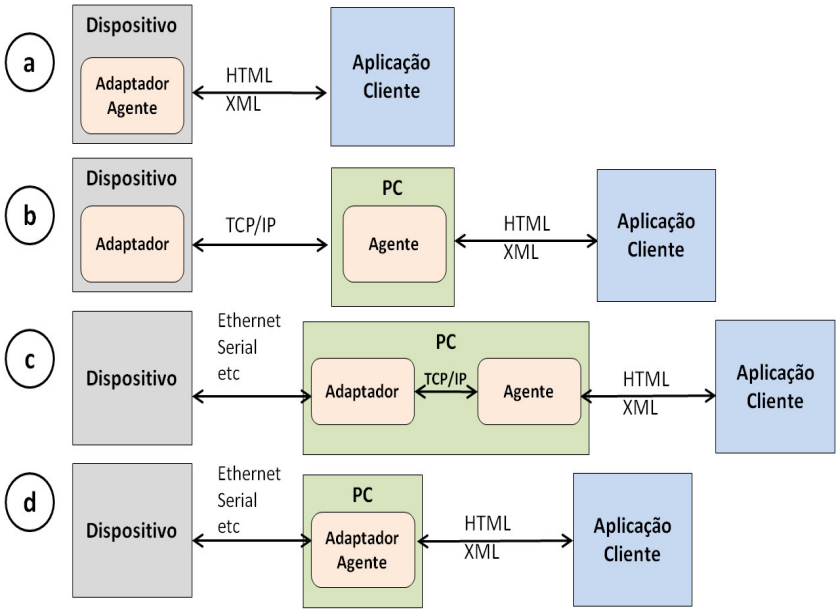

Figura 3.4: Possíveis arquiteturas para implementação do padrão MTConnect, adaptado de (RONDON, 2010)

nentes (*Components*).

Current - Requisição feita pelo Agente para valores atuais de todos os itens de dados especificados. Se nenhum item é especificado o Agente retorna os valores de todos os componentes disponíveis.

Sample - Um dado exato de uma serie de dados contínuos. Exemplo: a posição de um eixo em um dado momento.

Event - Representa quando ocorre uma alteração de estado em um determinado momento. É importante comentar que um evento não ocorre sob frequência pré-definida.

DataItem - Representa uma descrição ou valor de qualquer informação que possa ser coletada do Agente.

Condition - Uma informação que o dispositivo fornece como um indicador da sua saúde e capacidade de funcionar. Uma condição pode ser : *Normal*, *Warning*, *Fault* ou *Unavailable*. Um único tipo de condição pode ter várias *Faults* ou *Warnings* a qualquer momento. Esse comportamento é diferente de *Events* e *Samples* onde um *DataItem* deve ter apenas um único valor em um dado momento.

As Figuras 3.6 e 3.7 representam os fluxos entre as quatro entidades que compõe a arquitetura MTConnect: O *Name Service* (um servidor LDAP que converte nomes de dispositivos para a URI do Agente), *Application* (uma aplicação de usuário que faz uso especial dos dados do dispositivo), *Agent* (o processo de coleta de dados do dispositivo e entregá-lo aos aplicativos), e *Device* (um equipamento).

O diagrama da Figura 3.6 ilustra a inicialização de um Agente (*Agent*) e a comunicação com o dispositivo (*Device*). Inicialmente o Agente conecta-se e autentica-se com o *Name Service* (*LDAP Server*). Após isso, o Agente registra sua URI com o *Name Service* para que ele possa ser localizado. Em seguida, conecta-se ao dispositivo (*Device*) usando a API do dispositivo ou outro processo especializado. O

```
- <MTConnectDevices xsi:schemaLocation="urn:mtconnect.org:MTConnectDevices:1.3 /schemas/MTConnectDevices_1.3.xsd">
    <Header creationTime="2017-05-22T14:50:59Z" sender="mtcagent" instanceId="1490681234" version="1.3.0.9" assetBufferSize="1024" assetCount="0" bufferSize="131072"/>
  - <Devices>
    - <Device id="dev" iso841Class="6" name="VMC-3Axis" sampleInterval="10" uuid="000">
        <Description manufacturer="SystemInsights"/>
      - <DataItems>
          <DataItem category="EVENT" id="avail" type="AVAILABILITY"/>
          <DataItem category="EVENT" id="dev_asset_chg" type="ASSET_CHANGED"/>
          <DataItem category="EVENT" id="dev_asset_rem" type="ASSET_REMOVED"/>
        </DataItems>
      - <Components>
        - <Axes id="ax" name="Axes">
          - <Components>
            - <Rotary id="c1" name="C">
              - <DataItems>
                - <DataItem category="SAMPLE" id="c2" name="Sspeed" nativeUnits="REVOLUTION/MINUTE" subType="ACTUAL" type="SPINDLE_SPEED" units="REVOLUTION/MINUTE">
                    <Source>spindle_speed</Source>
                  </DataItem>
                - <DataItem category="SAMPLE" id="c3" name="Sovr" nativeUnits="PERCENT" subType="OVERRIDE" type="SPINDLE_SPEED" units="PERCENT">
                    <Source>SspeedOvr</Source>
                  </DataItem>
                - <DataItem category="EVENT" id="cm" name="Cmode" type="ROTARY_MODE">
                  - <Constraints>
                      <Value>SPINDLE</Value>
                    </Constraints>
                  </DataItem>
                  <DataItem category="CONDITION" id="Cloadc" type="LOAD"/>
                  <DataItem category="CONDITION" id="Csystem" type="SYSTEM"/>
                  <DataItem category="SAMPLE" id="c13" name="Cload" nativeUnits="PERCENT" type="LOAD" units="PERCENT"/>
                </DataItems>
              </Rotary>
            - <Linear id="x1" name="X">
              - <DataItems>
                  <DataItem category="SAMPLE" id="x2" name="Xact" nativeUnits="MILLIMETER" subType="ACTUAL" type="POSITION" units="MILLIMETER"/>
                  <DataItem category="SAMPLE" id="x3" name="Xcom" nativeUnits="MILLIMETER" subType="COMMANDED" type="POSITION" units="MILLIMETER"/>
                  <DataItem category="SAMPLE" id="n3" name="Xload" nativeUnits="PERCENT" type="LOAD" units="PERCENT"/>
                  <DataItem category="CONDITION" id="Xloadc" type="LOAD"/>
                  <DataItem category="CONDITION" id="Xsystem" type="SYSTEM"/>
                </DataItems>
              </Linear>
            - <Linear id="y1" name="Y">
              - <DataItems>
                  <DataItem category="SAMPLE" id="y2" name="Yact" nativeUnits="MILLIMETER" subType="ACTUAL" type="POSITION" units="MILLIMETER"/>
                  <DataItem category="SAMPLE" id="y3" name="Ycom" nativeUnits="MILLIMETER" subType="COMMANDED" type="POSITION" units="MILLIMETER"/>
                  <DataItem category="SAMPLE" id="y4" name="Yload" nativeUnits="PERCENT" type="LOAD" units="PERCENT"/>
                  <DataItem category="CONDITION" id="Yloadc" type="LOAD"/>
                  <DataItem category="CONDITION" id="Ysystem" type="SYSTEM"/>
                </DataItems>
              </Linear>
            - <Linear id="z1" name="Z">
              - <DataItems>
                  <DataItem category="SAMPLE" id="z2" name="Zact" nativeUnits="MILLIMETER" subType="ACTUAL" type="POSITION" units="MILLIMETER"/>
                  <DataItem category="SAMPLE" id="z3" name="Zcom" nativeUnits="MILLIMETER" subType="COMMANDED" type="POSITION" units="MILLIMETER"/>
                  <DataItem category="SAMPLE" id="z4" name="Zload" nativeUnits="PERCENT" type="LOAD" units="PERCENT"/>
                  <DataItem category="CONDITION" id="Zloadc" type="LOAD"/>
                  <DataItem category="CONDITION" id="Zsystem" type="SYSTEM"/>
                </DataItems>
```

Figura 3.5: Arquivo *Devices.xml*

dispositivo envia dados para o Agente ou o Agente sonda o dispositivo por dados.

Na Figura 3.7 mostra como todos os principais componentes do MTConnect se interrelacionam e como as quatro operações básicas são usadas para localizar e comunicar-se com o Agente em relação ao dispositivo. O processo de comunicação entre os elementos da arquitetura é descrito nas etapas abaixo (SOBEL, 2010):

1. O dispositivo envia continuamente informações ao Agente. O Agente coleta e salva as informações com base em sua capacidade de armazenar informações. O fluxo de dados do dispositivo para o Agente depende da implementação. O fluxo de dados pode inciar uma vez que um pedido tenha sido emitido a partir de uma aplicação cliente, a critério do Agente.

2. O aplicação localiza o dispositivo usando o *Name Service* com a sintaxe LDAP padrão que é interpretada da seguinte maneira: a fresa(*mill*) está na unidade organizacional Equip que está no domínio *example.com*. O registro LDAP para este dispositivo conterá uma URI que o aplicação pode usar para contatar o Agente.

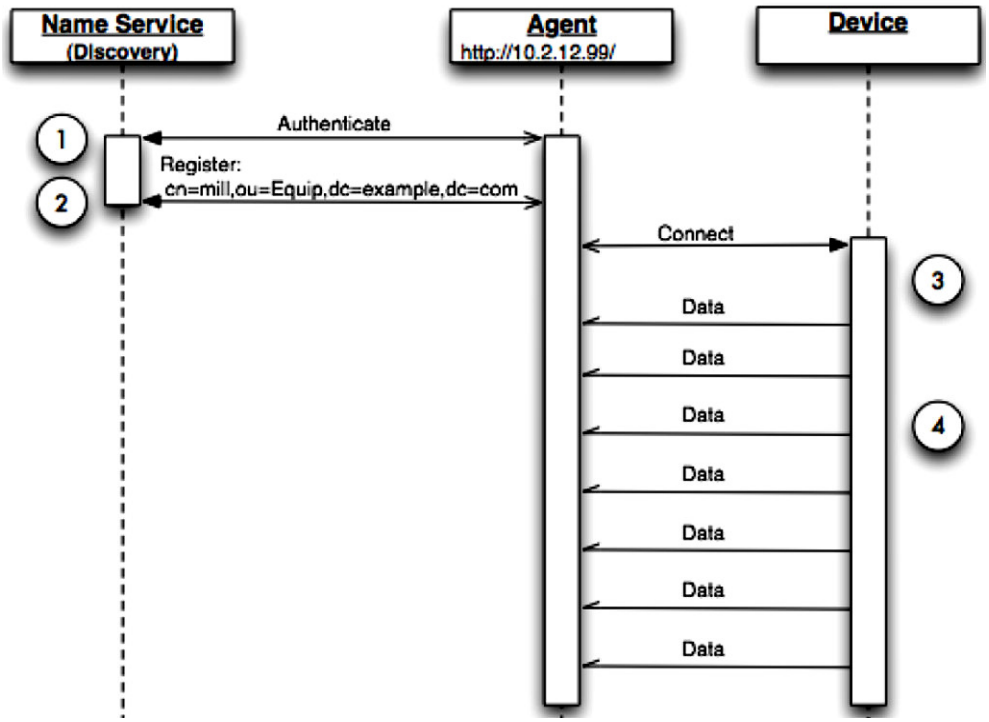

Figura 3.6: Inicialização do Agente (SOBEL, 2010)

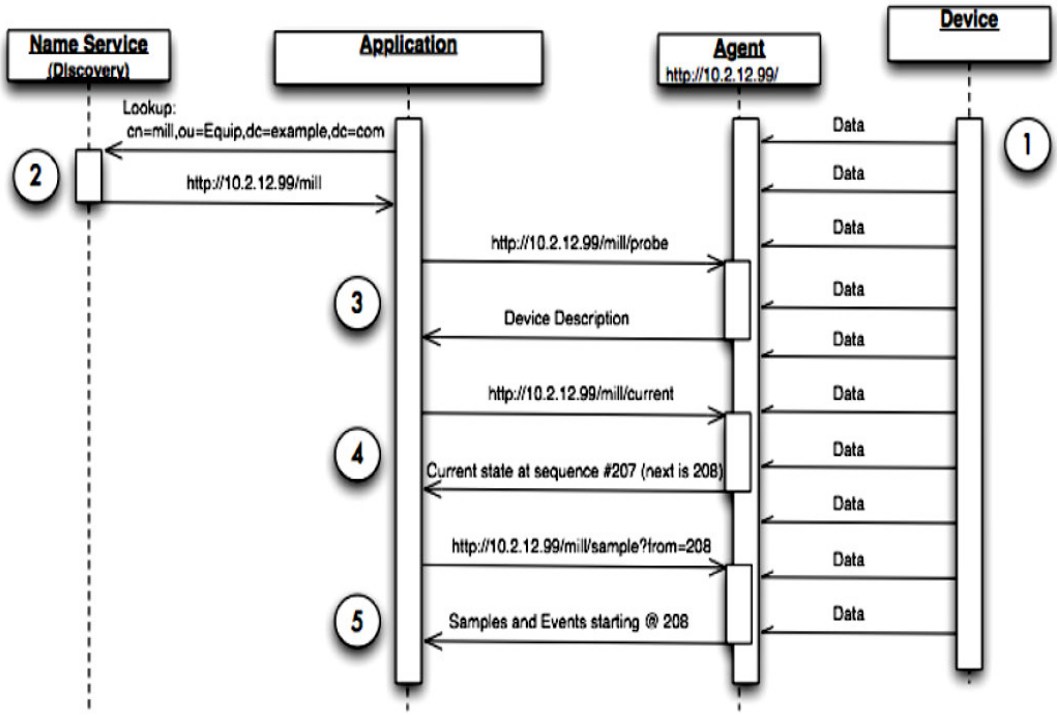

Figura 3.7: Comunicação com a Aplicação (SOBEL, 2010)

3. A Aplicação tem a URI para entrar em contato com o Agente, e assim o dispositivo *Mill*. O primeiro passo é um pedido de informação descritiva do dispositivo utilizando o pedido *Probe*. O *Probe* retornará a composição dos componentes do dispositivo, bem como todos os *DataItems* disponíveis.

4. O aplicativo solicita o estado atual (*current*) do dispositivo. Os resultados conterão o fluxo (*stream*) do dispositivo e todos os fluxos (streams) de componentes para esse dispositivo. Cada um dos *DataItems* reportará seus valores como *Samples*, *Events* ou *Condition*. O aplicativo receberá o Next-Serquence do Agente para usar no pedido de amostra (*Sample*) subseqüente.

5. A Aplicação usa o número *nextSequence* para amostrar(*sample*) os dados do Agente a partir do número de seqüência. Os resultados serão *Events*, *Condition* e *Sample*; e a contagem (*count*) não é especificada, portanto, o padrão é 100.

3.2 OPC

O uso de recursos de Tecnologia da Informação aplicada à automação industrial cresceu a partir dos anos 90. Um dos esforços mais relevantes nas ultimas duas décadas no âmbito de software para automação, foi à padronização do acesso aos dados a equipamentos onde diversos sistemas, protocolos e interfaces eram usados (GONÇALVES, 2012).

Anteriormente os produtores de equipamentos de automação tinham que desenvolver os seus próprios *drivers* para que as aplicações de supervisão e controle pudessem se comunicar com as máquinas (Figura 3.8), o que era um problema para os vendedores de software de IHMs (Interface Homem-Máqquina) e SCADA (Controle Supervisório e Aquisição de Dados, traduzido do inglês). Com isso, uma força tarefa foi iniciada com o objetivo de definir um padrão de *drivers* para o *Plug-and-Play* de equipamentos, permitindo um acesso padronizado aos dados de automação sobre plataforma Windows. O resultado foi a criação da especificação OPC *Data Access*, no ano de 1996. Neste mesmo período foi criada a *OPC Foundation*, uma organização sem fins lucrativos que tem como objetivo manter e desenvolver esse padrão (MAHNKE; LEITNER; DAMM, 2009).

3.2.1 OPC Clássico

Após a criação da OPC Foundation a organização trabalhou no sentido de definir interfaces de software para padronizar o fluxo de informações no nível de campo até o nível de gerenciamento. Com isso, três importantes especificações OPC foram inicialmente desenvolvidas, baseado nas diferentes necessidades dentro das aplicações industrais: *Data Access* (DA), *Alarm & Events (AE)* e *Historical Data Access* (HDA).

O OPC usa uma arquitetura cliente-servidor para a troca de informações. Um Servidor OPC encapsula as informações de processo e disponibiliza na sua interface. Um Cliente OPC conecta-se ao Servidor OPC e pode acessar e consumir os dados oferecidos. As aplicações consomem e fornecem dados que podem ser tanto do cliente quanto do servidor (GONÇALVES, 2012) (Figura 3.9).

O protocolo OPC clássico é baseado na tecnologia COM (*Component Object Model*) e DCOM (*Distributed Component Object Model*) da Microsoft. Esses modelos fornecem um mecanismo transparente para

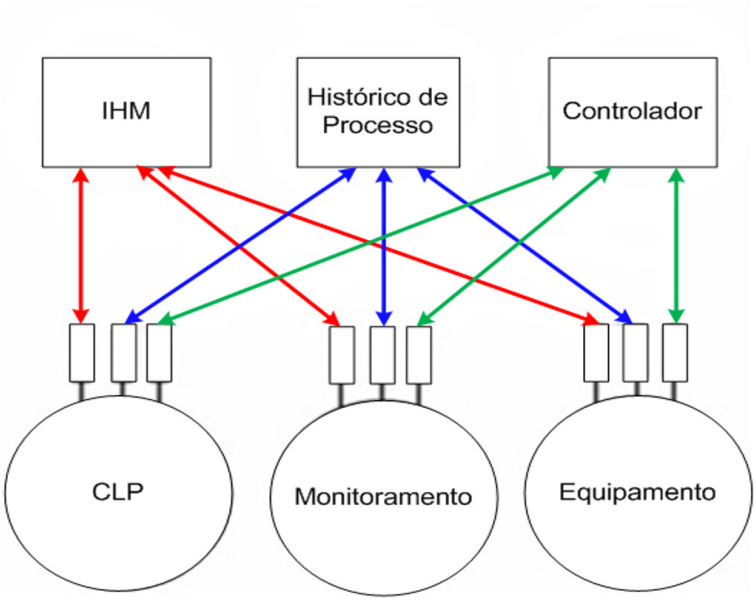

Figura 3.8: Soluções proprietárias, adaptado de (KONDOR, 2008)

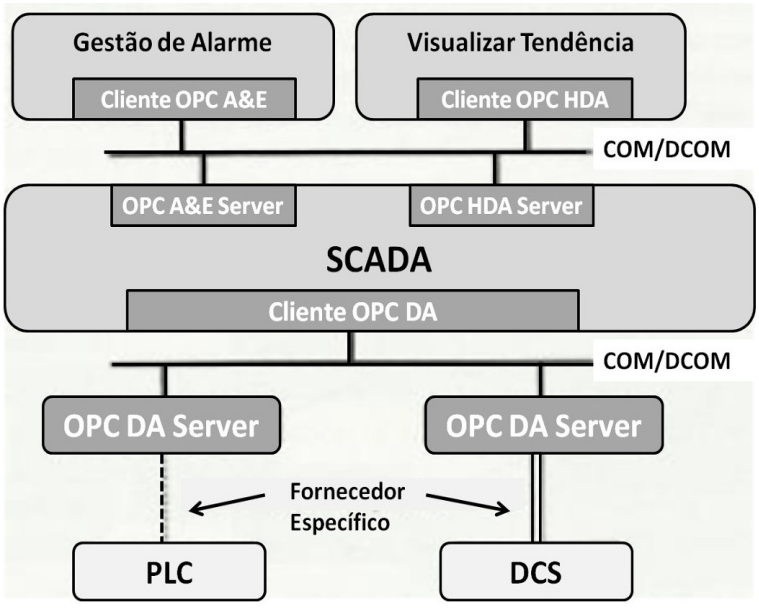

Figura 3.9: Caso típico de comunicação Clinte/Servidor OPC, adaptado de (GONÇALVES, 2012)

um cliente chamar métodos sobre um objeto COM, em um servidor rodando em um mesmo processo, em outro processo, ou em outro nó na rede (MAHNKE; LEITNER; DAMM, 2009). Essa tecnologia está disponível nos computadores executando sistema operacional Windows, fato que fez reduzir o esforço inicial da especificação do padrão, do tempo de desenvolvimento das especificações e produtos, além de reduzir o tempo de chegada do OPC ao mercado. Essa característica inicial foi fundamental para o sucesso do OPC.

Segundo Burke *et al.* (BURKE; IWANITZ; LANGE, 2010), as duas principais desvantagens do OPC clássico são a dependência da plataforma Windows e os problemas da tecnologia DCOM na utilização de comunicação remota com o OPC. A tecnologia DCOM é de difícil configuração, tem períodos longos e não configuráveis de *timeout*, e não podem ser usados para comunicação através da Internet.

Algumas das últimas especificações OPC clássico publicadas são apresentadas na lista abaixo:

- *OPC Overview* (v1.00.1.00) - Descrição geral das finalidades de cada especificação OPC;
- *OPC Common Definitions and Interfaces* (v1.10) - Definição das funcionalidades básicas para as demais especificações.
- *OPC Data Access Specification* (v3.00.1.00) - Definição da interface para leitura e escrita de dados em tempo real.
- *OPC Historical Data Access Specification* (v1.20.1.00) - Definição da interface para acesso a dados históricos.
- *OPC Alarms and Events Specification* (v1.10.1.00) - Descreve a interface para o monitorar áreas e notificar os clientes sobre as condições de alarme.
- *OPC XML-DA Specification* (v1.01.1.00) - Integração entre OPC e XML para aplicações via Internet (Web).
- *OPC Data eXchange Specification* (v1.00.1.00) - Descreve o comportamento do das comunicações Servidor/Servidor ou as comunicações Dispositivo/Dispositivo.
- *OPC Complex Data Specification* (v1.00.1.00) - É uma extensão das especificações existentes do OPC DA, adicionando uma camada de dados complexos/estruturados, como XML, etc.
- *OPC Security Specification* (v1.00.1.00) - É uma extensão das especificações OPC existentes adicionando uma camada pra controle de acesso seguro a aplicações e dados.
- *OPC Batch Specification* (v2.00.1.00) - Esta especificação é uma extensão da OPC Data Access Specification, e define a interface de acesso aos dados de processos por batelada (batch).
- *OPC Commands* (v1.00.0.29) - Define os meios para descobrir, entender e controlar os comandos (métodos) que um servidor fornece.

A Figura 3.10 mostra o conjunto da família de especificações OPC e o relacionamento entre elas. São interfaces que atendem a diferentes finalidades nas aplicações de campo. Elas podem ser utilizadas de forma independente uma das outras ou combinadas.

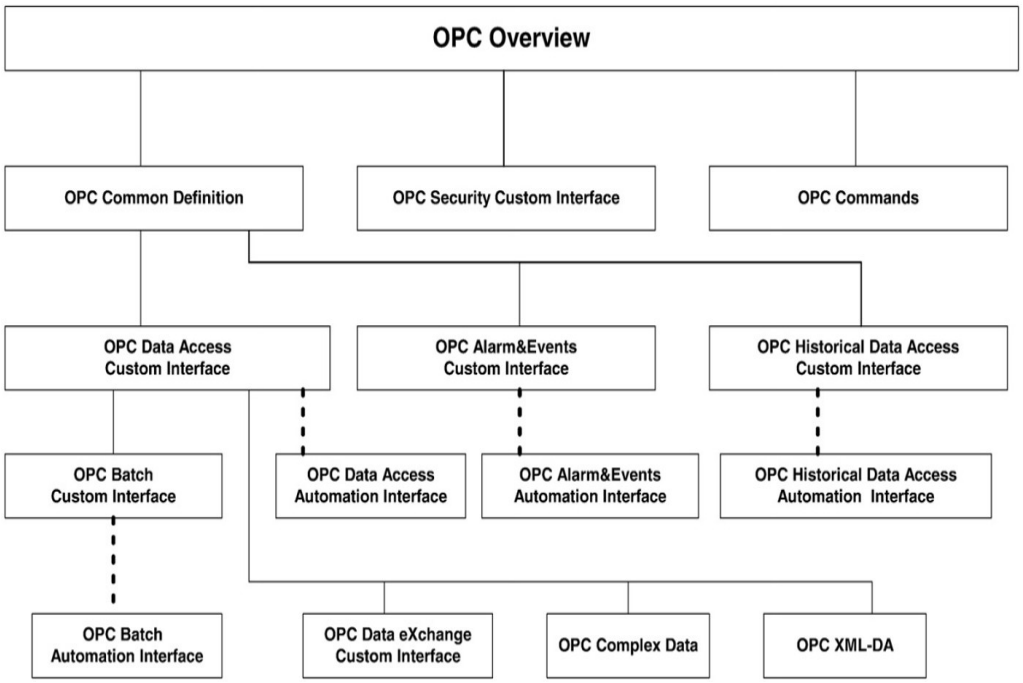

Figura 3.10: Visão geral das especificações OPC e seus relacionamentos (HONG; JIANHUA, 2006)

As aplicações que utilizam o protocolo OPC são classificadas de acordo com sua funcionalidade: Cliente OPC ou Servidor OPC. O servidor OPC Data Access compreende objetos e interfaces de servidor, uma área de armazenamento de dados, uma interface gráfica de usuário (GUI) e um driver de hardware (LING; CHEN; YU, 2004) (Fig. 3.11).

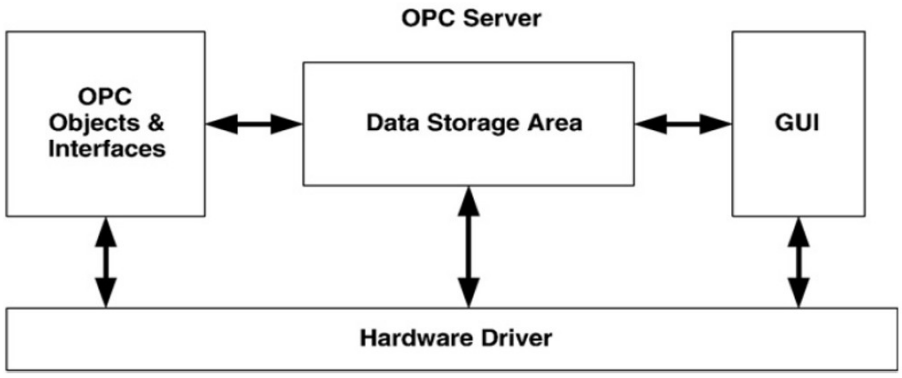

Figura 3.11: Arquitetura geral de um servidor OPC (LING; CHEN; YU, 2004)

Os dados extraídos do dispositivo são usados por aplicações de software especiais (por exemplo, SCADA, DCS e ERP) e software de terceiros (por exemplo, Delphi e Visual Basic) com os servidores OPC. Objetos OLE ou componentes OPC tem que usar funções OPC padrão ao implementar procedimentos de comunicação. Essas funções OPC diferem para cada dispositivo, mas eles empregam os mesmos padrões. Independentemente do servidor OPC em que essas funções são implementadas e os dados são solicitados, a resposta deve ser a mesma.

Os clientes OPC normalmente são produtos para controle e monitoramento de dados. Aplicações cliente OPC podem ser desenvolvidas em diversas linguagens de programação, como Java, C++, .NET *framework*, linguagem *script*, etc. A aplicação cliente pode acessar o servidor OPC através de duas interfaces: *OPC Custom Interface* e *OPC Automation Interface*. Aplicações desenvolvidas em linguagens de programação que suportam chamadas de funções por ponteiro (C,C++, Delphi, etc.) podem acessar o servidor através de interface Custom. Os casos em que as aplicações são programadas em linguagens que não suportam ponteiros (Visual Basic, .Net, etc.), o acesso ao servidor é feito por interface OPC Automation, que atuam como um proxy entre as aplicações clientes e a interface Custom (Figura 3.12) (IWANITS; LANGE, 2002).

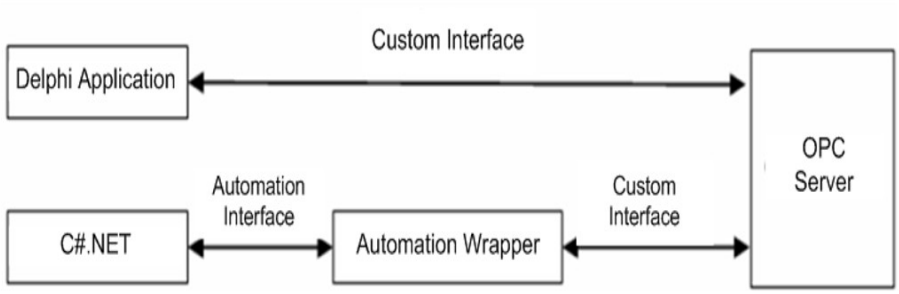

Figura 3.12: Acesso ao servidor OPC através de interfaces *Custom* e *Automation*, adaptado de (IWANITS; LANGE, 2002)

Um Cliente OPC pode se conectar a Servidores OPC produzidos por um ou mais fornecedores. O código fornecido pelo fornecedor determina os dispositivos e dados aos quais cada servidor tem acesso, os nomes dos dados e os detalhes sobre como o servidor acessa fisicamente esses dados (OPC-FOUNDATION, 2003).

Em alto nível de software, um servidor OPC é composto de vários objetos: servidor, grupo e os itens. O objeto Servidor OPC mantém informações sobre o servidor e serve como um recipiente para objetos de Grupo OPC (*OPC Group*). O objeto de *OPC Group* mantém informações sobre si próprio e fornece o mecanismo para conter e organizar logicamente Itens OPC (*OPC Item*).

Os Grupos OPC proporcionam aos clientes uma maneira de organizar os dados. Por exemplo, o grupo pode representar itens em uma amostra ou relatório de um operador específico. Os dados podem ser lidos e escritos. Conexões com exceção também podem ser criadas entre o cliente e os itens do grupo e podem ser habilitadas e desabilitadas quando necessário. Um cliente OPC pode configurar a frequência que um servidor OPC deve fornecer as atualizações de dados para o cliente OPC (OPC-FOUNDATION, 2003).

Dentro de cada Grupo o cliente pode definir um ou mais itens OPC (Figura 3.13).

Os itens OPC representam conexões com fontes de dados dentro do servidor. Um item OPC, a partir da perspectiva da interface personalizada, não está acessível como um objecto por um cliente OPC. Portanto, não há nenhuma interface externa definida para um item OPC. Todo acesso a itens OPC é feito através de um objeto de grupo OPC que "contém"o item OPC ou simplesmente onde o item OPC é definido. Associado a cada item há um Valor (*Value*), Qualidade do sinal (*Quality*) e Tempo de captura do dado (*Time Stamp*) (OPC-FOUNDATION, 2003).

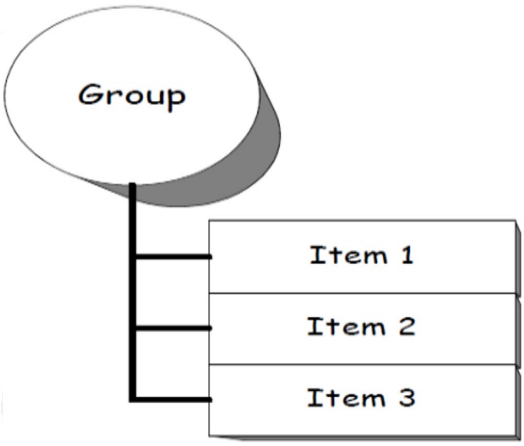

Figura 3.13: Relacionamento Grupo/Item (OPC-FOUNDATION, 2003)

Os itens não são as fontes de dados - eles são apenas conexões com eles. Por exemplo, as tags em um sistema DCS existem, independentemente de um cliente OPC estar acessando elas ou não. O Item OPC representa o endereço dos dados, e não a fonte física real dos dados que o endereço faz referência.

3.2.2 OPC-UA

OPC *Unified Architecture* (OPC UA) é a atual tecnologia da OPC Foundation para o transporte segura, confiável e interoperável de dados brutos e informações pré-processadas do chão-de-fábrica em sistemas de planejamento da produção (OPC-FOUNDATION, 2009). Foi uma especificação projetada para ser independente de plataforma, dimensionável, segura e fornecer alto desempenho (ELSHAFEI, 2014). Não por acaso, tem como tecnologias centrais para seus desenvolvimentos XML, SOAP e *Web Services*.

Segundo Mahnke *et al.* (MAHNKE; LEITNER; DAMM, 2009), o OPC *Unified Architecture* nasceu da vontade de criar um verdadeiro substituto para tudo que era baseado nas especificações COM, sem perder algumas características e desempenho. Para Burke *et al.* (BURKE; IWANITZ; LANGE, 2010), algumas razões motivaram o desenvolvimento dessa nova especificação, como:

- Iminente descontinuação do COM/DCOM.
- Limitações do modelo DCOM.
- Ter o OPC independente de plataforma, não apenas Windows.
- A necessidade de alto desempenho na comunicação através de Web Services.
- A integração de todas as ferramentas em um modelo unificado.
- Suporte a estruturas complexas de dados
- Comunicação sem perdas de dados de processo
- Aumento da segurança contra acessos não autorizados.

Foi uma especificação projetada especificamente para ambientes industriais. Possui varias soluções de comunicação disponíveis e muitas vantagens, a exemplo de (MTCONNECT-INSTITUTE-OPC-FOUNDATION, 2013):

- Modelo de segurança avançado;
- Protocolo de comunicação com tolerâncias a falhas;
- e uma estrutura de modelagem da informação que permite aos desenvolvedores de aplicações representarem dados de forma que faça sentido para cada um deles.

Segundo Gonçalves (GONÇALVES, 2012), os alicerces do OPC UA são o mecanismo de transporte e modelagem de dados.

O transporte é formando por diferentes mecanismos otimizados para diferentes necessidades. A primeira versão do OPC-UA possui por definição a utilização do protocolo TCP para alto desempenho em comunicação Intranet tão bem como um mapeamento para aceitar padrões da Internet, como *Web Services*, XML e HTTP (OPC-FOUNDATION, 2009).

A modelagem de dados é estabelecida por um modelo de informação para Servidores OPC-UA que permite aos usuários organizar os dados e a sua semântica de um modo estruturado. O modelo de informação constitui o espaço de endereço (*Address Space*) dos Servidores OPC-UA. É uma rede de nós com suas propriedades e relacionamentos. Um espaço de endereço do servidor consiste dos seguintes tipos de elementos (OPC-FOUNDATION, 2009):

- *Object*: Um nó que representa um elemento físico ou abstrato de um sistema. Os objetos são modelados usando o *OPC-UA Object Model*. Sistemas, subsistemas e dispositivos são exemplos de Objetos. Um Object pode ser definido como uma instância de um *ObjectType*.
- *ObjectType*: Um nó que representa a definição de tipo para um *Object*.
- *Variable*: Uma *Variable* é um nó que contém um valor.
- *VariableType*: Nó que representa a definição de tipo para uma *Variable*.
- *DataType*: Uma instância de um nó *DataType* que é usado em conjunto com o atributo *ValueRank* para definir o tipo de dados de uma variável.
- *ReferenceType*: Um nó que representa a definição de tipo de uma referência. O *ReferenceType* especifica a semântica de uma referência. O nome de um *ReferenceType* identifica como os nós de origem estão relacionados aos nós de destino e geralmente reflete uma operação entre os dois, como "A contem B".
- *Method*: Uma função de *software* chamada que é um componente de um objeto.
- *View*: Um subconjunto específico do *Address Space* que é de interesse para o cliente.

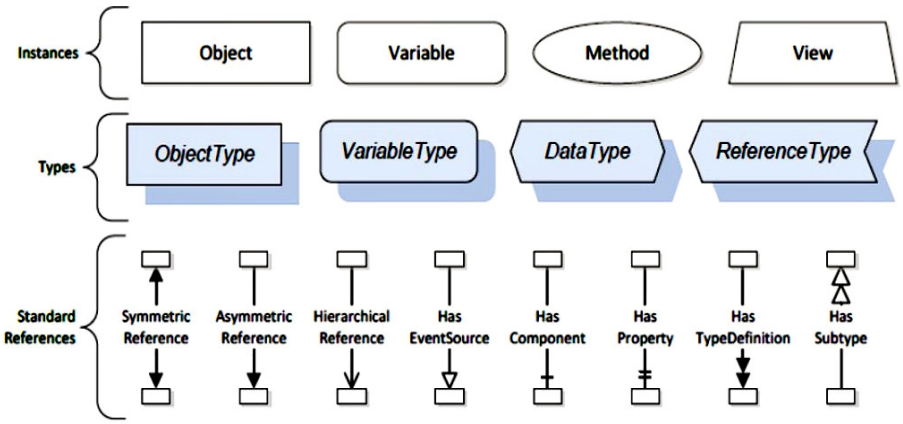

Figura 3.14: Notação do modelo de informação OPC-UA (MTCONNECT-INSTITUTE-OPC-FOUNDATION, 2013)

A notação dos modelos de informação OPC-UA é ilustrada na Figura 3.14.

Os aplicações Cliente/Servidor usam APIs (Application Programming Interface) OPC-UA Cliente e Servidor para troca de dados. APIs OPC-UA Cliente/Servidor é interface interna que isola o código da aplicação cliente/servidor de uma pilha de comunicação OPC-UA. A pilha OPC-UA converte chamadas da API Cliente/Servidor OPC-UA em mensagens e as envia através da entidade de comunicação relacionada; por outro lado, cada mensagem recebida da entidade de comunicação subjacente é entregue à aplicação Cliente/Servidor pela pilha de comunicação OPC-UA (CAVALIERI; CHIACCHIO, 2013).

A Figura 3.15 mostra a arquitetura de um cliente OPC-UA; uma aplicação cliente usa a API do cliente OPC-UA para enviar serviços OPC-UA e publicar solicitações ao servidor OPC-UA, e receber respostas de serviço e notificações do servidor OPC-UA. O OPC-UA. A Pilha de Comunicação converte chamadas da API cliente em mensagens e as envia através da entidade de comunicação realcionada ao servidor; a pilha também recebe as Mensagens de Resposta e Notificação da entidade de comunicação relacionada e entrega-as para a aplicação Cliente através da API do Cliente OPC-UA (CAVALIERI; CHIACCHIO, 2013).

A Figura 3.16 mostra a arquitetura do servidor OPC-UA. A Aplicação do Servidor é o código que implementa a função do Servidor. Objetos reais são objetos físicos ou de software que são acessíveis através do servidor OPC-UA ou que ele mantém internamente; os exemplos incluem dispositivos físicos e contadores de diagnóstico. Objetos particulares, chamados *nós* (*Nodes*), são usados pelo servidor OPC-UA para representar objetos reais, suas definições e referências; o conjunto de nós é chamado *Address Space* (Espaço de Endereço). Os nós são acessíveis a clientes usando serviços OPC-UA (interfaces e métodos). A Figura 3.16 também mostra os nós no *Address Space*, as referências entre eles (desenhadas por arcos conectando os nós) e suas relações com os objetos reais (CAVALIERI; CHIACCHIO, 2013).

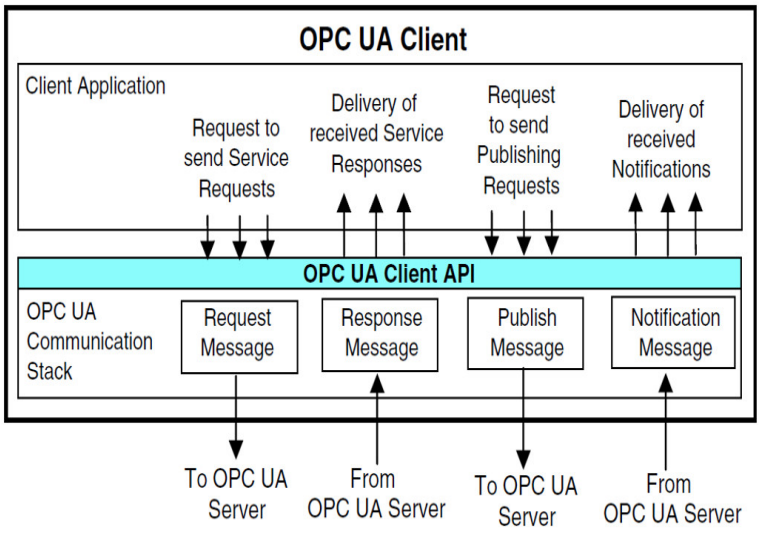

Figura 3.15: Cliente OPC-UA (CAVALIERI; CHIACCHIO, 2013)

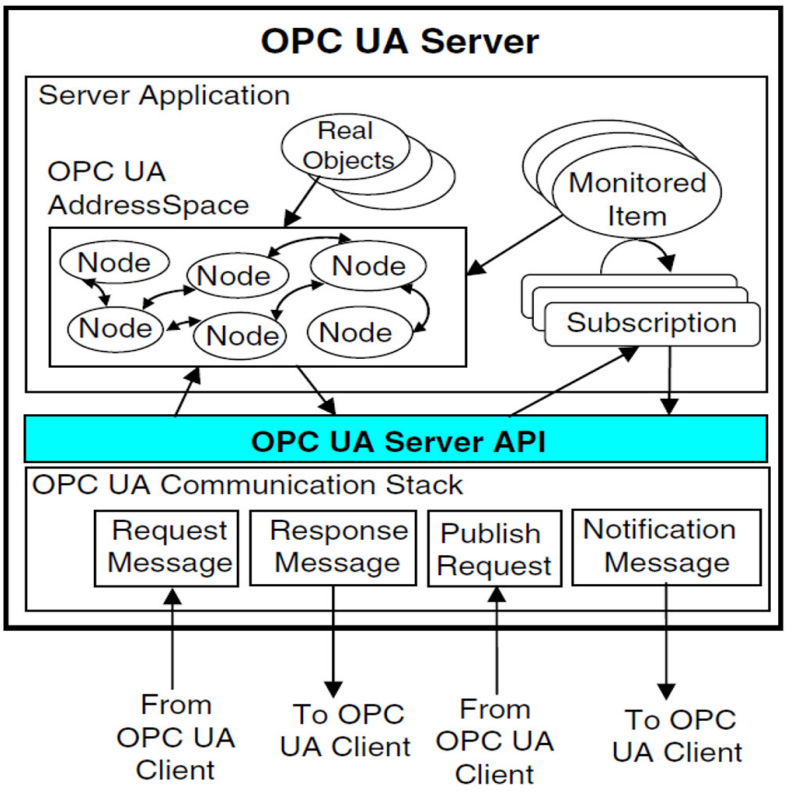

Figura 3.16: Servidor OPC-UA (CAVALIERI; CHIACCHIO, 2013)

3.3 Trabalhos anteriores relacionados: MTConnect e OPC

Há uma significativa produção bibliográfica relacionada a concepção de aplicações relacionadas a conectividade de dispositivos industriais envolvendo a especificação OPC clássica. Dentro desse universo estão os trabalhos voltados para a proposição de arquiteturas de monitoramento e controle distribuído

do chão-de-fábrica baseado no acesso OPC através da internet. Com base nesse paradigma, trabalhos vêm sendo desenvolvidos explorando as características da emergente especificação da OPC *Foundation*, o OPC-UA.

O MTConnect é um padrão recente, mas a sua versatilidade e os benefícios potenciais gerados a partir de seu uso, torna essa especificação objeto de intensa investigação dentro do meio acadêmico e industrial.

Pensando em fornecer um visão sobre o que foi desenvolvido no âmbito da automação para acesso a dados da manufatura através da internet associada aos protocolos MTConnect e OPC clássico, tecnologias centrais para o desenvolvimento deste trabalho, o quadro da Tabela 3.1 apresenta uma breve descrição e análise de trabalhos anteriores desenvolvidos sob essa perspectiva.

Com base em uma análise da bibliografia selecionada, observa-se que a maioria dos trabalhos desenvolvidos utilizam exclusivamente uma das especificações, ou OPC ou MTConnect. Não são explorados nos trabalhos as vantagens e atributos dos dois padrões associados em uma mesma arquitetura. O MTConnect é um padrão focado no monitoramento e interoperabilidade de dispositivos CNC, enquanto que o OPC tem a capacidade de fornecer controle supervisório para praticamente toda uma planta industrial.

Uma característica dos trabalhos envolvendo tanto OPC quanto MTConnect é que o monitoramento dos dispositivos é tratado como um recurso de apoio ao controle e/ou ao tratamento mais elaborado dos dados de campo, seja para a análises desses dados com a finalidade de suportar tomadas de decisões, seja para realizar medições, garantia de qualidade ou a otimização de processos.

3.4 *Cloud Computing*

Com o rápido desenvolvimento das tecnologias da informação e de rede, como a Internet, a virtualização e a computação em rede, a *Cloud Computing* (Computação na Nuvem) (DIKAIAKOS et al., 2009; SCHONWALDER; FOUQUET M. RODOSEK; HOCHSTATTER, 2009; MISRA; MONDAL, 2011; MELL; GRANCE, 2009) tornou-se uma nova tendência para as aplicações de Internet. É definida pelo *National institute of Standards and Technology* (NIST) da seguinte forma:

"*Cloud Computing* é um modelo que permite acesso onipresente, conveniente e sob demanda a rede para compartilhar um conjunto de recursos computacionais configuráveis (e.g., redes, servidores, armazenamento, aplicações e serviços) que possam ser rapidamente providos e lançados com mínimo esforço gerencial e interação com o provedor do serviço".

Segundo Bughin *et al.* (BUGHIN; CHUI; MANYIKA, 2010), colaboração, IoT e a nuvem são identificadas como tendências tecnológicas de negócios-chave que irão remodelar as empresas mundialmente. O principal tarefa da computação na nuvem é a prestação de serviços de computação sob demanda com alta confiabilidade, escalabilidade e disponibilidade em um ambiente distribuído.

A computação na nuvem está surgindo como um dos principais facilitadores na indústria de manufatura, podendo transformar modelos de negócio tradicionais na manufatura, promovendo o alinhamento da inovação de produto com a estratégia de negócio, e a criação de redes de fábricas inteligentes focadas em colaboração efetiva. Há duas formas sugeridas para adoção da computação na nuvem, a versão da manufatura com adoção direta das tecnologias da computação na nuvem, e a Manufatura na nuvem (*Cloud*

Tabela 3.1: Referências bibliográficas relacionadas ao projeto de sistemas envolvendo MTConnect e/ou OPC

Autor(es)	**Trabalho**
(SAHIN; BOLAT, 2009)	Utiliza a arquitetura distribuída do OPC (DOPC) para realizar o monitoramento e controle remoto de diferentes dispositivos baseado na web. Arquiteturas de controle local desenvolvidas em N pontos podem se comunicar uns com os outros e um ponto de controle remoto em uma página web dinâmica construída usando *Active Server Pages* (ASP).
(LEE; HU, 2008)	Usam a comunicação DCOM entre um cliente OPC DA e o servidor OPC DA, os dados tranferidos para o cliente OPC DA são convertidos em formato XML para que os dados possam ser acessados através da Internet por um cliente XML-DA que se comunica com um servidor XML (servidor web) para obter esses dados. Devido às possíveis limitações de desempenho, é improvável que o OPC XML-DA integre aplicações em tempo real, embora seja comumente usado como uma ponte entre a empresa e a rede de controle.
(FERNANDES; TORRISI; BRANDÃO, 2009)	Utiliza uma tecnologia de transporte com a função de gateway (OPC Server para HTTPs) chamada CyberOPC, para propor uma arquitetura para executar o ajuste remoto de sistemas de controle industrial usando a Internet, cumprindo requisitos de segurança e desempenho aceitáveis.
(EDRINGTON et al., 2014)	Este trabalho apresenta um sistema Web para monitoramento de uma máquina-ferramenta compatível com MTConnect, que também realiza análises e notificação de eventos da máquina para máquinas compatíveis com MTConnect. A aplicação fonece a gerente de produção informações necessárias para para melhorar a eficiência do processo no chão de fábrica e aumentar a eficácia geral do equipamento (OEE).
(FRANÇA; TORRISI; BOTTENE, 2013)	O artigo descreve a implementação de uma arquitetura universal para monitorar controladores de máquina CNC empregando o protocolo de interface Mtconnect, chamada OPF Monitoring Framework. Um estudo de caso a apresentado voltado para a captura de dados de uma máquina de inspeção CNC com operação desenvolvida em ESPRIT CAM.
(SILVA et al., 2011)	O objetivo deste artigo é o desenvolvimento de um sistema de supervisão baseado na web para operação de esmerilhamento. O padrão MTConnect foi usado para adquirir dados do CNC. As combinações particulares dos parâmetros mais adequados do CNC são determinadas para definir o status atual da máquina.
(MICHALOSKI et al., 2013)	Este trabalho propõe um a arquitetura de um sistema de manufatura para prover um através da Web, dados em tempo real e estatísticas para a garantia da qualidade. Tudo baseado na integração de duas especificação abertas: MTConnect e QMResults (Quality Measurement Results).
(VIJAYARAGHAVAN; DORNFELD, 2010)	Apresenta-se um sistema para o monitoramento do consumo de energia desenvolvido usando o MTConnect e um sistema de motor de regras e processamento de evento complexo (CEP), para suportar o racionamento de dados e processamento de informações. Esses dados e informações são armazenados na nuvem e vão alimentar análises de alta frequência de dados em series temporais.

Manufacturing). A computação em nuvem atua em algumas das principais áreas de fabricação, tais como TI, modelos de negócios *pay-as-you-go* (pague pelo que usa), aumento de produção sob demanda e flexibi-

lidade na implantação e personalização de soluções. Na manufatura na nuvem os recursos distribuídos são encapsulados em serviços na nuvem e gerenciados de forma centralizada. Os clientes podem usar serviços na nuvem de acordo com suas necessidades. Os usuários da nuvem podem solicitar serviços que vão desde o projeto de produto, fabricação, testes, até a distribuição e operações de logística reversa, envolvendo todas as etapas do ciclo de vida de um produto (XU, 2012).

Um grande princípio da computação na nuvem é tudo ser tratado como um serviço (*Everything as a service* - XaaS), a exemplo de SaaS (*Software as a Service* - Software como serviço), PaaS (*Platfom as a Service* - plataforma como serviço) e IaaS (*Infrastructure as a Service* - infraestrutura como serviço). Esses serviços definem uma estrutura de sistema em camadas para Computação na Nuvem (Fig. 3.17). Na camada de infraestrutura, processamento, armazenamento, redes e outros recursos fundamentais de computação são definidos como serviços padronizados na rede. Os clientes dos provedores de nuvem podem implantar e executar sistemas operacionais e software para as infraestruturas relacionadas. A camada média, ou seja, PaaS fornece abstrações e serviços para desenvolvimento, teste, implementação, hospedagem e manutenção de aplicações em ambiente de desenvolvimento integrado. A camada de aplicação fornece um conjunto completo de aplicações de SaaS. A camada de interface de usuário na parte superior permite a interação perfeita com todas as camadas subjacentes de XaaS (PALLIS, 2010).

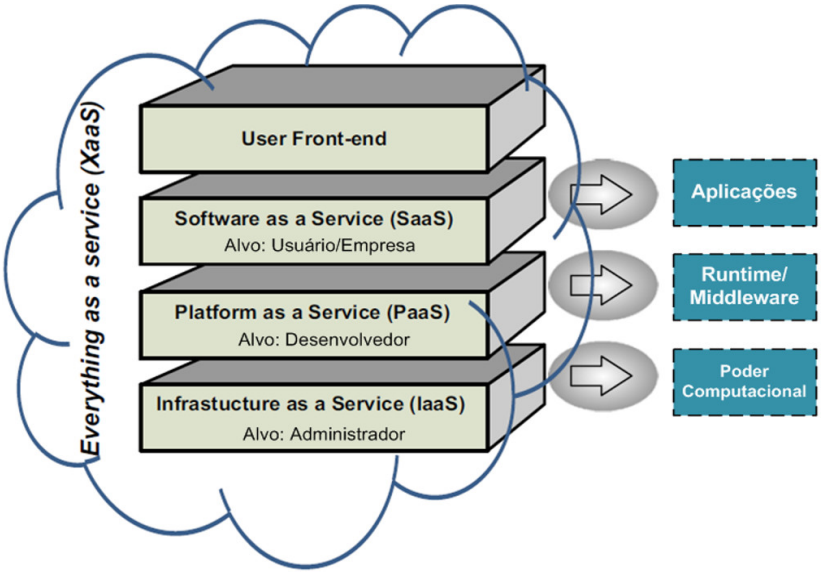

Figura 3.17: Computação na nuvem: Tudo como um serviço, adaptado de (XU, 2012)

A computação na nuvem é considerada como um campo de pesquisa multidisciplinar como resultado da evolução e convergência de várias tendências de computação. Mais precisamente a evolução orientada a negócio da computação em rede (FOSTER et al., 2008). Implementar a computação em nuvem significa uma mudança de paradigma de negócios e infra-estrutura de TI, onde o poder de computação, armazenamento de dados e serviços são terceirizados para terceiros e disponibilizados como *commodities* para empresas e clientes (XU, 2012).

Alguns requisitos arquiteturais gerais para o desenvolvimento da Computação da Nuvem são apresentados por Rimal *et al.* (RIMAL et al., 2011). Os mesmos autores classificaram esses requisitos para fornecedores, empresas e usuários.

3.4.1 Requisitos dos Fornecedores

No ponto de vista dos fornecedores, é necessária uma arquitetura de serviço altamente eficiente para apoiar a infraestrutura e os serviços, a fim de fornecer serviços virtualizados e dinâmicos. Software como serviço (SaaS), Plataforma como serviço (PaaS) e Infraestrutura como serviço (IaaS) são três tipos comuns de modelos de prestação de serviços. Esses serviços geralmente são fornecidos por meio de interfaces padrão da indústria, como *Web Services*, arquitetura orientada a serviços (SOA) (OPEN-GROUP, 2010) ou serviços REST (*REpresentational State Transfer*) (FIELDING, 2000):

1. Software como Serviço (SaaS), algumas vezes denominado Aplicação com serviço Serviço (AaaS), oferece uma plataforma multi-inquilino, em que recursos comuns e uma única instância do código-objeto de uma aplicação e o banco de dados subjacente são usados para suportar vários clientes simultaneamente. Para este fim, o SaaS também é referido como o modelo de Fornecedor de Serviço de Aplicação (Application Service Provider-ASP). Exemplos de provedores chave são o sistema de Gerenciamento de Relacionamento com o Cliente (CRM) da força de vendas, o NetSuite e o aplicação de produtividade do *Google Office*. Uma consideração importante em SaaS é a integração efetiva com outras aplicações (XU, 2012).

2. Plataforma como Seviço (PaaS) fornece aos desenvolvedores uma plataforma que inclui todos os sistemas e ambientes que compõem o ciclo de desenvolvimento, teste, implementação e hospedagem de sofisticadas aplicações web como um serviço fornecido por uma plataforma baseada na nuvem. PaaS comumente encontrados inclui Facebook F8, Salesforge App Exchange, Google App Engine, Bunzee conectar e Amazon EC2. PaaS pode oferecer uma série de serviços prontamente disponíveis, o que significa que PaaS pode suportar várias aplicações na mesma plataforma (XU, 2012).

3. Infraestrutura como um Serviço (IaaS) é às vezes chamada de Hardware como um Serviço (HaaS). O IaaS promove um esquema de pagamento baseado em uso, o que significa que os clientes pagam conforme usam. Este serviço é extremamente útil para usuários corporativos, pois elimina a necessidade de investir na construção e no gerenciamento de seus próprios sistemas de TI. Outra vantagem importante é a habilidade de ter acesso ou usar a tecnologia mais recente à medida que ela surge. Sob demanda, auto-sustentável ou auto-ajuste, multi-inquilino, categorização de clientes são os principais requisitos de IaaS (RIMAL et al., 2011). GoGrid, Mosso/ Rackspace, MSP On-Demand e masterIT são alguns dos provedores pioneiros IaaS.

Outros requisitos essenciais são arquitetura ser centrada em serviço, ter qualidade de serviço (QoS) entre fornecedor e usuário final, interoperabilidade associada a um formato padrão de dados e integração entre aplicações, tolerância a falhas, balanceamento de carga de trabalho entre as entidades da nuvem (e.g. servidores, dispositivos de armazenamento, redes e recursos de TI) e gerenciamento da virtualização de recursos.

3.4.2 Requisitos das Empresas

Segundo Xu (XU, 2012), há quatro modelos de implementação da nuvem: publico, privado, comunitário e hibrido. Diferentes tipos de modelos de implementação adequam-se a diferentes situações. Nuvem

publica provê o conceito-chave de compartilhar os serviços e a infraestrutura fornecida por um fornecedor a parte em um ambiente multi-inquilino (WU; YANG, 2010). Nuvem privada envolve o compartilhamento de serviços e infraestrutura fornecida por uma organização ou seu fornecedor de serviço especificado em uma ambiente de inquilino único (*single-tenant*). Aplicações de atividades centrais e de missão critica das empresas é frequentemente mantida em uma nuvem privada. Nuvem comunitária é compartilhada por várias organizações e é apoiada por uma comunidade que compartilha interesses e preocupações (MELL; GRANCE, 2009). Nuvem hibrida consiste de múltiplas nuvens internas (privadas) e externas (públicas). A complexidade acrescida de determinar como distribuir aplicações entre nuvens publicas e privadas poder ser um desafio. Para Xu (XU, 2012), as empresas precisam alavancar estrategicamente os quatro modelos de implementação na nuvem.

Os diferentes modelos de serviços discutidos anteriormente (SaaS, PaaS e IaaS) ocupam diferentes níveis de requisitos de segurança no ambiente da nuvem. A IaaS é a fundação de todos os serviços na nuvem, com a PaaS construída sobre ela e a SaaS, por sua vez, construída sobre a PaaS. Assim como as capacidades são herdadas, assim são as questões de segurança da informação e os riscos.

3.4.3 Requisitos dos Usuários

Os requisitos dos usuários são o terceiro fator chave para uma adoção voluntária e bem-sucedida de qualquer sistema em nuvem em uma empresa. Para os usuários, a confiança é muitas vezes uma grande preocupação. Nuvem baseada em confiança é, portanto, um recurso essencial e mandatório (RYAN, 2011).

Quando se fala em usuários finais individuais e faturamento e medição baseadas em consumo em um sistema na nuvem, uma analogia pode ser feita com a medição de consumo e alocação de água, gás ou eletricidade em uma base de unidades de consumo. O gerenciamento de custos é importante para tomar decisões de planejamento e controle. Análise da repartição de custos, rastreio de atividade utilizada, gestão de custos adaptáveis, transparência do consumo e faturamento são também considerações importantes.

No que se refere à privacidade na computação na nuvem, alguns dados de usuários (considerado como sua propriedade intelectual) são armazenados em centrais de megadados localizados no ciberespaço. Em casa ambiente a privacidade torna-se a principal questão (RYAN, 2011; CAVOUKIAN, 2008). Há uma grande resistência e relutância de uma empresa armazenar qualquer dado vital na nuvem. Mas, há várias tecnologias que podem melhorar a integridade, a confidencialidade e a segurança de dados nas nuvens, por exemplo, LANs virtuais, caixas intermediarias de rede (*Firewalls* e filtros de pacotes) (XU, 2012).

3.4.4 Computação na Nuvem no contexto da Manufatura

Acredita-se que a Computação em Nuvem pode desempenhar um papel crítico na realização da filosofia DAMA (*Design Anywhere, Manufacture Anywhere*). A abordagem DAMA demanda a capacidade de trocar dados de projeto e fabricação em várias plataformas. A DAMA também ajuda a estabelecer vínculos entre planejamento de recursos de manufatura, planejamento de recursos empresariais, planejamento de recursos de engenharia e gerenciamento de relacionamento com clientes (XU, 2012).

A computação na nuvem está avançando rumo à integração das estruturas de organizações. A indús-

tria de transformação já começa a colher os benefícios da adoção da Nuvem, em um movimento para a adaptação para uma fabricação mais "inteligentes". O custo benefício de adoção de nuvens em uma empresa de manufatura típica pode ser múltiplo. Para Shalini (SHALINI, 2010), as economias obtidas com a eliminação de funções que foram essenciais na manufatura tradicional podem ser significativas. Com soluções baseadas na Nuvem, algumas customizações de aplicações e ajustes conforme as necessidades das empresas no nível de processo podem ser tratadas pelo setor de TI da empresa, juntamente com algumas das tecnologias inteligentes de computação na nuvem. Quando uma forma diferente da execução de um processo é iniciada, a equipe de TI pode fazer a mudança acontecer perfeitamente e em menos tempo.

Computação na nuvem também pode ser usada para melhorar muitos outros aspectos de empresas de fabricação, movendo processos tradicionais para a nuvem a fim de melhorar a eficiência operacional. De acordo com Bughin *et al.* (BUGHIN; CHUI; MANYIKA, 2010), colaboração em escala usando a tecnologia de Nuvem é uma tendência emergente de negócios. Com a adoção de tecnologias na Nuvem, a colaboração empresarial pode acontecer em uma escala muito mais ampla. Dentro da organização, planejamento da demanda e organização da cadeia de suprimentos podem ser integrados a um sistema baseado na Nuvem, permitindo que diferentes partes da organização sondar as oportunidades nas quais suas equipes de vendas estão trabalhando.

O movimento no sentido de expandir a utilização da Computação na Nuvem dentro as empresas de manufatura leva o nome de Manufatura na Nuvem (*Cloud Manufacturing*). Esse paradigma é parte do esforço de transição para fabricação orientada a produção para a fabricação orientada a serviços. Como a Computação na Nuvem, a Fabricação na Nuvem é considerada como um novo domínio multidisciplinar que engloba tecnologias como Fabricação em rede, Rede de Fabricação (MGrid), Fabricação Virtual, Manufatura ágil, Internet das Coisas e, claro, Computação na Nuvem. A Manufatura na Nuvem reflete tanto o conceito de "integração de recursos distribuídos"quanto o conceito de "distribuição de recursos integrados"(XU, 2012).

Na Manufatura na Nuvem, os recursos distribuídos são encapsulados em Serviços na Nuvem e gerenciados de forma centralizada. Os clientes podem usar os Serviços na Nuvem, de acordo com suas necessidades. Usuários da Nuvem podem solicitar serviços que vão desde o projeto de produto, fabricação, testes, gestão e todas as outras fases de um ciclo de vida do produto. Uma plataforma de serviços na Nuvem de fabricação realiza pesquisas, mapeamento inteligente, recomendações e execução a de um serviço (XU, 2012). A Figura 3.18 ilustra a estrutura de um sistema de fabricação na Nuvem que consiste em quatro camadas: camada recurso de fabricação, camada de serviço virtual, camada de serviço global e camada de aplicação.

3.5 Service-Oriented Architecture (SOA)

Alguns padrões, como MTConnect, STEP (*STandard for the Exchange of Product model data*) e outros, têm sido úteis para permitir integração e intercâmbio de dados entre diferentes sistemas de software de engenharia. Nesse sentido, a indústria de manufatura está em um esforço para converter as funcionalidades fornecidas por esses sistemas de software em serviços. Essa tendência ganhando força com a chegada da Computação em Nuvem, que permite a virtualização de recursos de computação e comunicação.

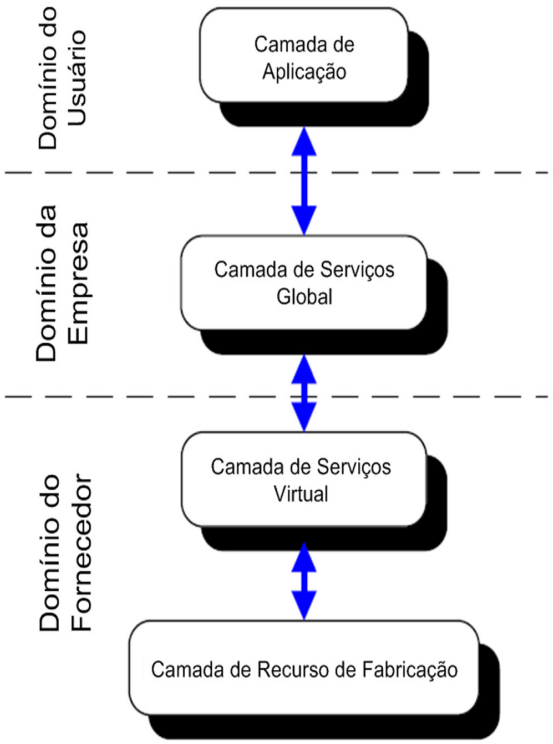

Figura 3.18: Arquitetura em camada de um sistema de Manufatura na Nuvem, adaptado de (XU, 2012)

A Arquitetura Orientada a Serviços (SOA) tem como objetivo alcançar um ambiente distribuído e com baixo acoplamento, na forma uma loja de componentes de software baseada em Nuvem. No paradigma orientado a serviços, os componentes de software são vistos como fornecedores de funcionalidades por meio de serviços independentes. Os serviços são virtualizações de componentes de software. Ou seja, os consumidores de serviços não precisam saber como os provedores de serviços oferecem seus serviços - de onde, por qual ou por quantos componentes de software (IVEZIC; KULVATUNYOU; SRINIVASAN, 2014).

O paradigma orientado a serviços enfatiza a visibilidade e a semântica que possibilitam a correspondência entre necessidades e capacidades, e a composição de capacidades para atender a essas necessidades. A visibilidade e a semântica são possibilitadas por descrições e acordos de serviço que capturam informações essenciais dos consumidores, e que os provedores de serviços precisam estar cientes e concordar (IVEZIC; KULVATUNYOU; SRINIVASAN, 2014).

O SOA é comumente implementado através *Web Services*, que se referem a um conjunto de padrões de várias organizações de desenvolvimento padrão; Esses padrões incluem *Web Service Description Language* (WSDL) e *Business Process Execution Language* (BPEL). No entanto, tais serviços também podem ser implementados usando outras estratégias. Recentemente, a implementação de SOA usando *Representational State Transfer* (REST), também conhecido como RESTful Web Services, ganhou aceitação generalizada . A implementação RESTful é considerada mais simples e mais fácil de usar do que a implementação baseada em WSDL.

Apesar do SOA fornecer o paradigma e tecnologia para permitir a composição dinâmica de serviços

de informações de engenharia, não é em si uma solução para problemas de domínio específico, vários outros serviços de informação de engenharia podem ser compostos usando SOA para formar soluções para domínios específicos (IVEZIC; KULVATUNYOU; SRINIVASAN, 2014).

3.6 Web Services

Como indica o próprio nome, um serviço web é um tipo de aplicação web, ou seja, uma aplicação normalmente executada sobre o protocolo HTTP (*Hyper Text Transport Protocol*). Um serviço web é, portanto, uma aplicação distribuída cujos componentes podem ser implementados e executados em dispositivos distintos como, por exemplo, PCs e dispositivos móveis (MACHADO-JUNIOR, 2014).

Os serviços web podem ser divididos basicamente em dois grupos: os serviços baseados em SOAP (*Simple Object Access Protocol*) e os REST (*Representational State Transfer*). A distinção entre ambos não muito evidente, mas o que se sabe é um serviço baseado em SOAP entregue sobre HTTP é um caso especial dos serviços REST.

3.6.1 REST - *Representational State Transfer*

O REST por si só não representa uma arquitetura, mas sim um conjunto de restrições que, quando aplicadas na concepção de um sistema, cria um estilo de arquitetura de software (MACHADO-JUNIOR, 2014). Um sistema escrito no estilo REST é denominado RESTful, carregando as seguintes características:

- Ser um sistema cliente-servidor.
- Ser independente de estado, ou seja, cada requisição deverá ser independente das outras.
- Tem que suportar um sistema de cache, a infraestrutura de rede deve suportar caches em diferentes níveis.
- Ser acessível de maneira uniforme, cada recurso deve ter um endereço exclusivo e um ponto de acesso válido.
- Tem que ser em camadas e deve suportar escalabilidade.

Um sistema no estilo RESTful pode ser implementado em qualquer arquitetura de rede disponível, de forma que não foi necessário a criação novas tecnologias ou protocolos de rede.

Segundo Machado Junior (MACHADO-JUNIOR, 2014), enquanto o SOAP é um protocolo de mensagens, o REST é um estilo de arquitetura de software para sistemas hipermídia distribuídos denominados recursos. Recurso RESTful é tudo que possa ser endereçável pela web. Isso significa que sistemas em que textos, gráficos, áudio e outras mídias são armazenadas em uma rede e interconectadas através de *hiperlinks*, podendo ser acessados e transferidos entre clientes e servidores.

O REST foi desenvolvido juntamente com o protocolo HTTP 1.1 e, diferente do SOAP que estabelece um protocolo para comunicação de objetos e serviços, utiliza corretamente os verbos HTTP (GET, POST,

PUT e DELETE) para criar serviços que poderiam ser acessados por qualquer tipo de sistema (FIDEL, 2015). Esses métodos HTTP ganham correspondencia com operações CRUD (*Create*, *Read*, *Update*, *Delete*). Cada requisição HTTP inclui um dos métodos apresentados na Tabela 3.2 para indicar qual a operação CRUD que deve ser realizada sobre o recurso.

Tabela 3.2: Métodos HTTP e correspondentes operações CRUD

Métodos HTTP	**Operação**
GET	Lê um recurso
POST	Cria um novo recurso a partir dos dados requisitados
PUT	Atualiza um recurso a partir dos dados requisitados
DELETE	Remove um recurso

As características da arquitetura RESTful foram utilizadas como uma extensão para o OPC-UA (GRÜNER; PFROMMER; PALM, 2015; GRÜNER; PFROMMER; PALM, 2016). Neste trabalho, a arquitetura RESTful foi implementada e avaliada na forma de um *Web Service* associado a um servidor OPC-DA clássico (tecnologia COM/DCOM) com função de *gateway*, para a transmissão de informações de status e atuação em um centro de torneamento CNC, representando uma alternativa ao uso de servidor OPC-UA. Essa opção de implementação foi considerada por ser uma estilo de arquitetura orientada a recurso, em que interfaces simples e uniformes são utilizadas para o intercâmbio de representações desses recursos, a exemplo um procedimento remoto que acessa um servidor de dados e permite a comunicação com um cliente HTTP.

3.6.2 SOAP - *Simple Object Access Protocol*

SOAP é um protocolo leve para a troca de informações em um ambiente descentralizado e distribuído. É um protocolo baseado em XML que consiste em três partes: um envoltório que define uma estrutura para descrever o que está em uma mensagem e como processá-la, um conjunto de regras de codificação para expressar instâncias de tipos de dados definidos pelo aplicativo e uma convenção para representar chamadas de um procedimento remoto e respostas (W3C, 2000).

O SOAP tem o potencial de ser usado em combinação com uma variedade de outros protocolos, a principal é a combinação do SOAP com o HTTP No entanto, as únicas ligações definidas neste documento descrevem como utilizar o SOAP em combinação com o HTTP e suas extensões.

A Figura 3.19 apresenta um típico procedimento de operação do protocolo SOAP. Nesse exemplo, um cliente SOAP através de sua biblioteca solicita um serviço enviando uma mensagem SOAP ao servidor que por sua vez envia outra mensagem SOAP com a resposta do serviço correspondente.

Em um serviço web baseado em SOAP, um cliente geralmente faz uma chamada de procedimento remoto ao servidor solicitando uma operação do serviço web. Essas operações requisição/resposta sobre mensagens SOAP padronizadas permitem que o cliente e o servidor sejam escritos em diferentes linguagens de programação.

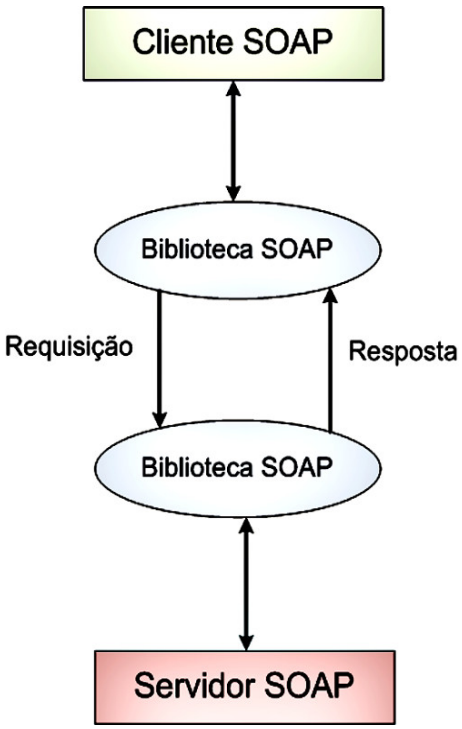

Figura 3.19: Operação de uma Web Service baseado em SOAP, adaptado de (MACHADO-JUNIOR, 2014)

Dentro do conjunto de especificações do padrão OPC, o OPC XML-DA foi a primeira especificação *Web Service* do OPC. Ela utiliza o protocolo SOAP como base para a estruturação das mensagens que são entregues.

3.7 Manufatura Eletrônica

A manufatura eletrônica (E-manufacturing) pode possuir diferentes significados a depender do foco de sua aplicação, o que levou a vários pesquisadores a estabelecer diferentes definições para essa metodologia. Ela trata da utilização da Internet e das tecnologias do comércio eletrônico nas indústrias transformadoras para o compartilhamento de informações em tempo real por diferentes níveis de uma empresa. As definições mais populares de *E-manufacturing* (*E-Mfg*) são dados por AMR (AMR, 2000) e Koc *et al.* (KOC et al., 2004).

Koc *et al.* (KOC et al., 2004) define a manufatura eletrônica como um sistema de transformação que permite que as operações de fabricação alcancem um desempenho preditivo de inatividade quase zero e tenha a capacidade de sincronizar os sistemas de negócios através do uso de tecnologias para Web e que independam de plataforma. A AMR dá uma definição mais detalhada para *E-mfg*:

"O centro de uma estratégia de manufatura eletrônica é um roteiro tecnológico para a transparência da informação entre o cliente, as operações de fabricação e os fornecedores. Uma estratégia de manufatura eletrônica leva em conta os processos de *e-business*, como a fabricação sob demanda (*build-to-order*) ou manutenção centrada em confiabilidade, e gera diretrizes para a implantação de sistemas de produção. A

estratégia de *e-manufacturing* abrange as estratégias de e-business e fabricação e cria um roteiro para o desenvolvimento e a implementação das mesmas na planta."

Outra definição é que E-manufacturing é uma como uma metodologia que possibilita a integração das operações de manufatura com os objetivos funcionais da empresa através do uso da internet (KOC; LEE, 2002). Para Molina e Santaella (MOLINA; SANTAELLA, 2006) o uso da *e-mfg* possibilita que as empresas obtenham conhecimento com base em criação de valor e fabricação sob demanda, permitindo que as empresas integrem os dados, informações e os conhecimentos necessários para clientes, fornecedores e a empresa.

A manufatura eletrônica inclui a capacidade de monitorar os ativos do chão-de-fábrica e prever a variação e a perda de desempenho para que dinamicamente sejam reprogramadas as operações de produção e manutenção, e sincronizar ações para conseguir a completa integração entre sistemas de manufatura e aplicações de níveis superiores na organização, conforme ilustra a Fig. 3.20. Também fornece trocas eficientes de informações configuráveis entre unidades de fabricação, sistemas CRM (*Customer Relationship Management*) e sistemas SCM (*Supply Chain Management*).

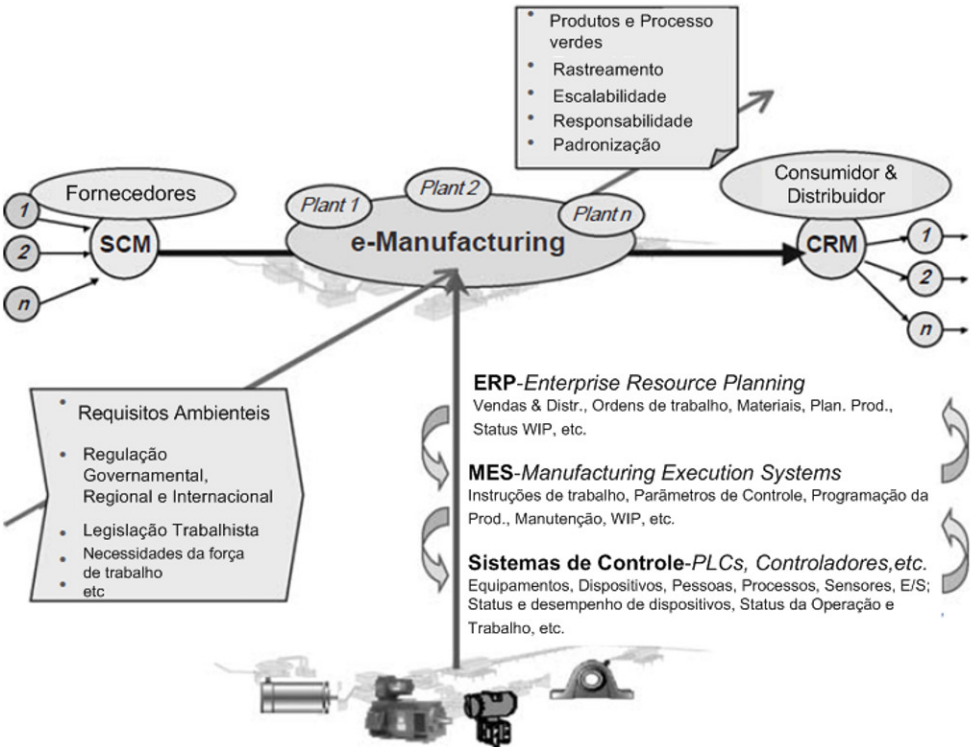

Figura 3.20: Integração da manufatura eletrônica e sistemas de *e-business*, adaptado de (LEE, 2003)

A *E-manufacturing* baseada na Internet abrange um conjunto de atividades de fabricação *on-line* envolvendo produtos e serviços, que inclui projeto de produtos, controle de produção e monitoramento de condições, gerenciamento de cadeia de suprimentos, serviços de manutenção e venda pela Internet. O principal elo entre a *e-manufacturing* e o desenvolvimento das fabricas inteligentes está na fabricação integrada com a tecnologia de Internet e nas potenciais mudanças na relação das pessoas com o trabalho. Essa relação fica evidente em algumas das características da manufatura eletrônica selecionadas por (CHENG;

BATEMAN, 2008):

Digitalização - Qualquer informação relacionada com a fabricação que possam ser digitalizadas também podem ser armazenadas e acessadas através da Internet dentro ou fora de uma empresa de fabricação.

Globalização - A natureza global da Internet fornece às empresas de manufatura a infraestrutura para apoiar seus engenheiros, parceiros e clientes com acesso à informação, independentemente de onde eles estão fisicamente.

Mobilidade - A Internet permite acessar informações de qualquer lugar e a qualquer hora, o que pode melhorar a agilidade e a capacidade de resposta de uma empresa às necessidades dos clientes.

Trabalho colaborativo - a tecnologia da Internet suporta dados compartilhamento e colaboração de trabalho. As empresas podem criar equipes de desenvolvimento, onde os membros da equipe podem residir em diferentes áreas geográficas. As informações do projeto e a comunicação interativa podem ser baseadas na Internet. Uma variedade de ferramentas de colaboração, como grupos de notícias, grupos de bate-papo, painéis de boletim, ferramentas de projeto e fabricação integrados, podem ser usadas para que os membros da equipe possam comunicar e compartilhar ideias, informações e dados com eficiência e eficácia.

Imediatismo - ao acessar um site, a extranet de um fornecedor ou a intranet da empresa, as informações mais recentes sobre a fabricação podem ser acessadas instantaneamente. Os engenheiros podem ter acesso em tempo real à informação sempre que necessário.

A *E-manufacturing* está modificando as características da operações de fabricação, que variam conforme o setor, a Figura 3.21 ilustra esse movimento.

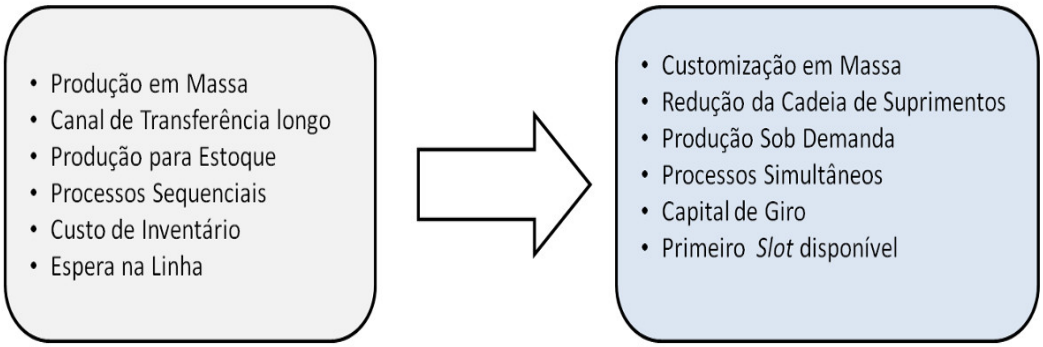

Figura 3.21: Transformação gerada pela *e-manufacturing*

3.8 Manufatura Remota

A manufatura remota, ou telemanufatura, é definida por Malek *et al.* (MALEK; WOLF; GUYOT, 2008) como uma atividade em que uma empresa cliente utiliza serviços oferecidos por uma outra empresa especializada (servidores), disponibilizados através das supervias da informação, como a Internet, a fim de executar, em tempo real, operações e processos necessários para o projeto e a produção de bens. Com isso, a telemanufatura participa de todo o ciclo de desenvolvimento do produto, começando na concepção,

e atravessando as etapas da fabricação, chegando até a distribuição do produto.

Um dos primeiro projetos que trouxe a luz o paradigma de manufatura remota via Internet foi desenvolvido por Bailey (BAILEY, 1995) com o conceito de *Telemanufacturing Facility* (TMF) para alavancar a capacidade de prototipagem rápida através da Internet. Ao longo história da telemanufatura surgiram empreendimento com alto nível de especialização com o projeto Cybercut (http://cybercut.berkeley.edu), um sistema desenvolvido na Universidade de Berkeley, projetado para modelar e usinar peças prismáticas (BROWN; WRIGHT, 1998). Outro exemplo foi desenvolvido por ÁLVARES *et al.* (ALVARES et al., 2002) onde o sistema permite a teleoperação de uma máquina de oxi-corte CNC, bem como a geração do programa NC da peça a ser produzida utilizando um ambiente de modelagem CAD/CAM (http://weboxicorte.graco.unb.br), denominado de *WebOxiCorte*. Os princípios de Telemanufatura e da Teleoperação também foram desenvolvidos na arquitetura da metodologia de integração CAD/CAPP/CAM dentro da manufatura remota, o sistema *WebMachining* (http://webmachining.graco.unb.br) (ALVARES, 2005).

Sistemas de teleoperação de equipamentos industriais enquadram-se no contexto de telemanufatura (MALEK; WOLF; GUYOT, 2008) no que se refere ao controle da manufatura em operações de chão-de-fábrica, bem como nos ambientes computacionais integrados de CAD/CAPP/CAM para desenvolvimento de produto. Também realizaram-se trabalhos no âmbito da teleoperação aplicada ao controle de robôs industriais e robôs moveis, descritos em Álvares e Tourino (ALVARES; TOURINO, 2000). Esses sistemas são atualmente disponibilizados através da Internet/Intranet, utilizando recursos da Web.

O ambiente de Telemanufatura normalmente baseia-se em uma arquitetura cliente/servidor voltada para o trabalho cooperativo, com interatividade e o compartilhamento de recursos de forma distribuída (PRADHAN; HUANG, 1998; ADAMCZYK; KOCIOLEK, 2001).

3.9 Controle Supervisório

No sentido estrito, controle supervisório significa que um ou mais operadores humanos estão intermitentemente programando e continuamente recebendo informações de um computador que faz um ciclo de controle autônomo através de efetuadores e sensores artificiais para o processo controlado ou ambiente de tarefa (SHERIDAN, 1992).

Em um sentido menos estrito, controle supervisório significa que um ou mais operadores humanos estão intermitentemente programando e continuamente recebendo informações de um computador que interliga através de efeitos e sensores artificiais para o processo controlado ou ambiente de tarefa (SHERIDAN, 1992).

Em ambas as definições o computador transforma as informações do humano para o processo controlado e do processo controlado para o humano, mas apenas na definição estrita o computador fecha um controle circular que exclui o elemento humano, e assim fazendo do computador um controlador autônomo de algumas variáveis.

A Figura 3.22 mostra cinco diagramas de sistemas homem-máquina que caracterizam o controle supervisório em relação aos extremos: Controle Manual e Controle Completamente Automático. Os elementos

comuns aos cinco diagramas são os *displays* e controles interligados com o operador humano, e sensores e atuadores interagindo com o processo controlado e tarefa de ambiente. O sistema 1 representa o controle manual convencional (não auxiliado por computador). No sistema 2 há um auxilio computacional, seja em um ou em ambos, ciclo de atuação e sensoriamento. Este corresponde a definição menos restrita de controle supervisório. Em ambos os casos percebe-se que todas as decisões de controle dependem de um operador humano. Se o operador para, o controle para.

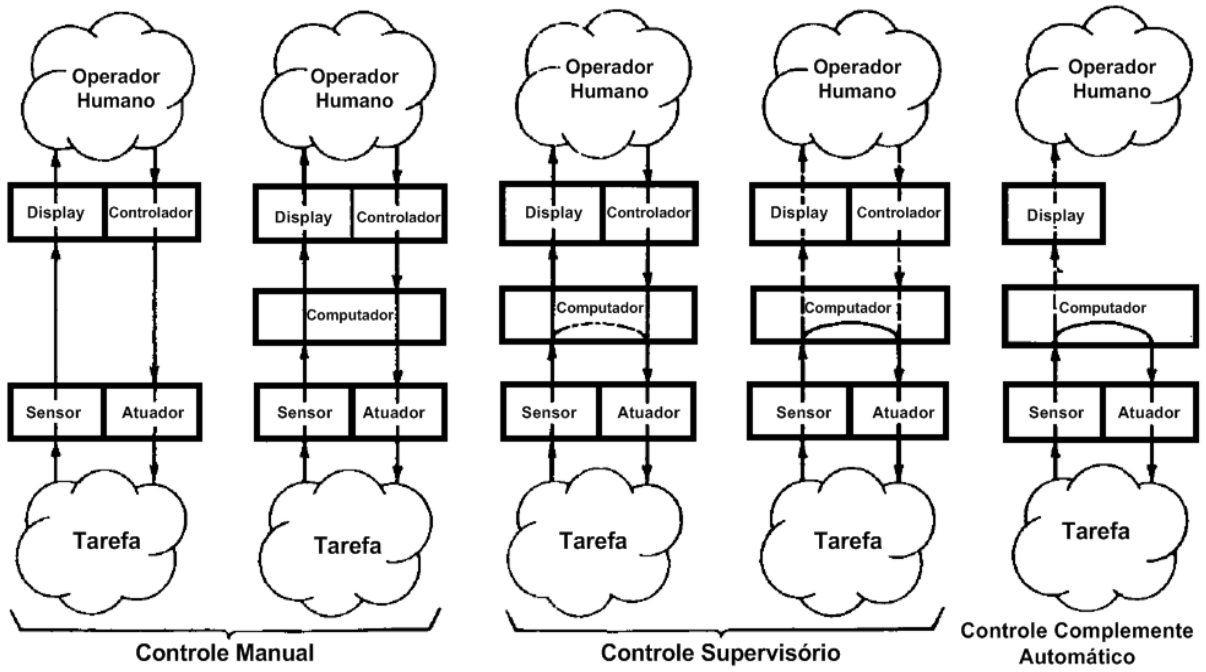

Figura 3.22: Espectro de modos de controle, adaptado de (SHERIDAN, 1992)

Quando uma menor (sistema 3) ou uma grande (sistema 4) fração de controle é realizada por malhas de controle fechadas diretamente através do computador e excluindo o ser humano, este é o controle de supervisão no sentido estrito. Se o sistema de controle é configurado essencialmente como controle automático (sistema 5) - ou seja, se o operador humano pode-se observar, mas não pode influenciar o processo - não é mais o controle de supervisão (SHERIDAN, 1992).

3.10 Teleoperação

Há situações em que a operação de determinados dispositivos precisa ser realizada a distância através de um operador remoto. Esses dispositivos remotamente operados realizam tarefas especificas e são aplicadas a diferentes finalidades como a operação remota de robôs, cirurgias médicas à distância, exploração submarina, operação de máquinas, entre outros. Segundo Álvares (ALVARES, 2005), a execução dessas atividades de modo ideal requer um sistema que tenha a mesma capacidade de um operador humano para reconhecer qual tarefa tem que ser realizada e possuir flexibilidade para realizar manipulações físicas, de maneira análoga à operação de um humano.

No campo da operação remota existe a Teleoperação que, segundo Nof (NOF, 1999), representa o

controle contínuo e direto de um teleoperador, que corresponde a sistema manipulador que requer comandos e ou supervisão de um ser humano remotamente. Por extensão, a telepresença refere-se à utilização de intensa realimentação sensorial para a teleoperação (BENAVENTE, 2007), fornecendo realismo para o operador em uma abordagem que garanta um maior nível de "proximidade"ao objeto manipulado, mesmo estando a grandes distâncias deste objeto. Nof (NOF, 1999) afirma que um sistema dito Teleoperador consiste de uma unidade remota (manipulador), uma unidade de comando para entrada dos comandos do operador (interface homem/máquina) e um canal de comunicação representando a ligação da unidade de comando com a unidade remota.

Inicialmente a Teleoperação foi desenvolvida com finalidade de manipulação de materiais radioativos. A teleoperação permite que um operador exerça força e realize movimentos em uma máquina remota, e ainda receba realimentação sensorial, seja através de dados visuais, sonoros ou táteis, ou o conjunto desses. Após a introdução da teleoperação, foi possível o desenvolvimento de interfaces capazes de prover uma interação satisfatória entre homem e máquina, permitindo que serviços de grande destreza fossem realizados (ALVARES, 2005).

Algumas classificações foram propostas para descrever a teleoperação. Um dessas categoriza os sistemas de teleoperação com base no grau de automação do sistema, em um espectro que vai da mínima até a máxima autonomia (SHERIDAN, 1984; ZHAI; MILGRAM, 1991). Essa classificação inclui:

- controle manual sem auxílio computacional;
- controle manual com significativo auxílio ou transformação computacional;
- controle supervisório com predomínio do controle realizado pelo operador humano;
- controle supervisório com predomínio do controle realizado pelo computador;
- controle completamente automático, onde os operadores humanos observam o processo sem intervenções.

A relação interface homem-máquina promovida pela teleoperação possui os seguintes modelos (ZHAI; MILGRAM, 1992):

a) Modelo Mestre-Escravo;

b) Modelo de Telepresença;

c) Modelo Professor-Aluno e;

d) Modelo Supervisor-Companheiro.

Quando efetivada a comunicação entre homem e máquina deve-se observar a relação entre máquina remota e seu ambiente. Com isso, são propostos esquemas de classificação de modelos de ambientes:

a) Ambiente Remoto Totalmente Modelado;

b) Ambiente Remoto Parcialmente Modelado e;

c) Ambiente Remoto Desconhecido.

O nível de autonomia é uma questão bastante relevante, pois define o quanto o dispositivo teleoperado pode funcionar independentemente do operador humano. Isso depende a estratégia de controle implementada em um computador localizado no site remoto. O operador pode estabelecer altos níveis de controle ao computador remoto, o qual pode executar os níveis mais baixos de controle (ALVARES, 2005). Adicionalmente, para o controle efetivo de dispositivos remotos alguns aspectos devem ser considerados:

- o atraso entre uma ação de controle do operador e sua correspondente visualização, como informação de realimentação para a sua ação, mostrada no seu *display*;
- largura de banda do sistema de comunicação, que determinam uma parcela importante do atraso relacionado à ação de controle;
- experiência na tarefa de controle contínuo, manual, do dispositivo controlado, questões de segurança e erros de posicionamento e autonomia (SHERIDAN, 1992).

3.10.1 Sistemas de Teleoperação e de Manufatura Remota via Internet

Os sistemas de teleoperação remota via Internet desenvolvidos até hoje estão em grande parte associados a atividades de telemanufatura e telerrobótica, e nesses sistemas o monitoramento é parte do controle do dispositivo na forma de realimentação do resultado de comandos efetuados remotamente.

Há um grupo de projetos que foram desenvolvido no âmbito da Telemanufatura que envolve todo o ciclo de desenvolvimento de produto, indo da concepção até a fabricação do produto final. Projetos como esses são parte de empresas altamente especializadas que oferecem a seus clientes serviços em ambientes voltados para telemanufatura. Essas empresas utilizam de forma ampla a integração entre sistemas de projeto, planejamento de processo, fabricação e gestão (sistemas CAE/CAD/CAPP/CAM/ERP, entre outros), que fornecem informações específicas de cada sistema a fim de auxiliar os clientes no desenvolvimento de novos processos e produtos (AHN; C.; P., 1999; MALEK; WOLF; GUYOT, 2008). Dentre os projetos desenvolvidos no campo da telemanufatura é possível destacar os seguintes:

Cybercut (http://cybercut.berkeley.edu)

Foi desenvolvido na Universidade de Berkeley, é um sistema CAD/CAPP/CAM baseado na Internet que possibilita a modelagem de peças prismáticas para usinagem (BROWN; WRIGHT, 1998). É constituído pelos seguintes componentes:

a) Sistema CAD escrito em linguagem de programação Java utilizando applets via Web. Este módulo do sistema baseia-se no conceito de *Destructive Solid Geometry* (DSG), em que são removidos de uma peça em formato de paralelepípedo entidades geométricos (*features*) através do fresamento da peça bruta. Baseado nesse conceito, o processo de manufatura é formado a medida de o processo de projeto avança.

b) Um sistema de Planejamento de Processo Auxiliado por Computador (CAPP), que tem acesso a uma base de conhecimento com informações sobre ferramentas e dispositivos de fixação.

c) Fresadora CNC de arquitetura aberta (modelo HAAS/VF-1) que recebe informações dos sistemas de projeto e planejamento em linguagem de alto nível e executa a usinagem.

Dessa forma, um projetista com um navegador Web com acesso a interface CAD do sistema a partir da Internet pode utilizar essa ferramenta para prototipagem rápida online. O projeto desenvolvido pode ser carregado em um formato específico no servidor *Cybercut*, o qual produzirá o planejamento de processo e a geração do código G adequado para a fresadora.

No *Cybercut* a modelagem por *features* de usinagem possibilita uma rica interoperabilidade entre o projeto e manufatura, o que é adequado para sistemas de projeto e manufatura de peças prismáticas. Durante a criação das *features*, um verificador de regras de manufatura garante a manufaturabilidade do projeto (BENAVENTE, 2007).

Sistema *WebOxiCorte*

Este sistema foi desenvolvido por Álvares et al. (ALVARES et al., 2002) e permite a teleoperação de uma máquina de oxi-corte CNC através da Internet, além de gerar o programa NC da peça a ser produzida através de um ambiente de modelagem CAD/CAM desenvolvido em linguagem Java.

O sistema é formado por uma interface gráfica GUI (*Graphical User Interface*) (Figura 3.23) para teleoperação de uma máquina de oxi-corte da White Martins (AutoCut 2.5L) com CNC da MCS Engenharia (MCS-520). A GUI disponibiliza ao usuário remoto funções para o controle da máquina, bem como de um ambiente CAD/CAM para geração do código G baseado no padrão RS-274. É baseado na arquitetura cliente/servidor, em que é um navegador de Internet, e o servidor um computador pessoal com sistema operacional Linux. A interface de usuário de foi desenvolvida em HTML e Java, e o servidor é composto por um conjunto de programas escritos em linguagem C para controle da máquina (ALVARES et al., 2002).

Nesse sistema a comunicação entre servidor e máquina é realizada por meio do protocolo DNC do CNC. A GUI de teleoperação é provida com *applets* Java com funções de controle da máquina, e imagens capturadas por Webcam. Com esses recursos o usuário é capaz de enviar e executar programas, bem como operações básicas na máquina.

No ambiente CAD/CAM o projeto é baseado em *features*, que nesse caso consiste basicamente em linhas e arcos (peças em 2D). O *applet* dessa aplicação permite a modelagem peça usando coordenadas cartesianas, além de efetuar a geração do código G. Na mesma interface é possível inserir dados referentes ao planejamento do processo. Com isso, é possível ter simultaneamente as perspectivas de projeto e da manufatura durante a fase de desenvolvimento do produto.

A Figura 3.24 apresenta a arquitetura detalhada do sistema de teleoperação *WebOxiCorte*. O servidor *WebOxiCorte* é constituído de servidor de vídeo e servidores de teleoperação da máquina. Este disponibilizam serviços de comando, *download*, *upload* e execução de programas, tratamento de erros e outras funções associadas ao protocolo DNC do controlador da máquina.

O servidor de vídeo captura as imagens da máquina e as distribui através do protocolo TCP/IP. Os servidores de controle da máquina trabalham de modo bidirecional, recebendo comandos e enviando infor-

Figura 3.23: Interface gráfica com o usuário (ALVARES et al., 2002)

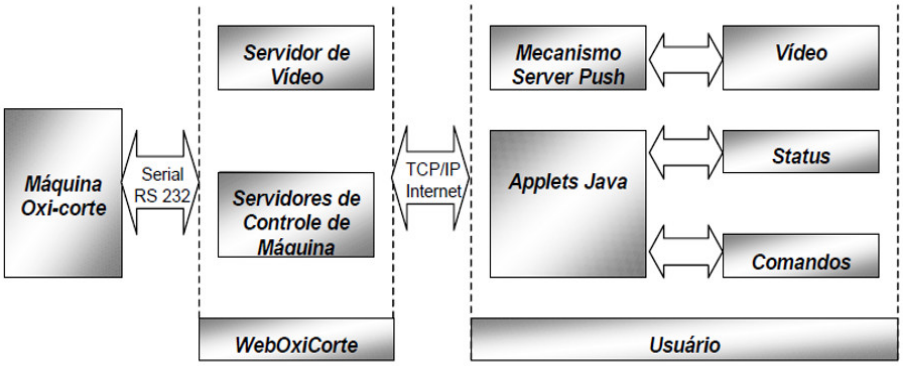

Figura 3.24: Arquitetura detalhada do sistema WebOxiCorte (ALVARES et al., 2002)

mações de status da máquina através da Internet.

Sistema *Webmachining*

O *Webmachining* (ALVARES, 2005) é um sistema parte de uma proposta metodológica voltada para a integração CAD/CAPP/CAM para a manufatura de peças rotacionais utilizando a Internet. Essa metodologia também emprega a modelagem baseada na síntese da peça por *features* de projeto (para operações de furação e subtração para operações de fresamento e furação), e através disso é possível atividades projeto (CAD), planejamento de processo (CAPP) e fabricação(CAM).

O sistema *Webmachining* foi concebido em um ambiente distribuído de agentes de software interoperáveis formando uma "Comunidade de Agentes de Manufatura". Segundo Wooldridge (WOOLDRIDGE, 2007), um Agente é uma entidade de computador que é capaz de realizar ações autônomas em nome de seu

proprietário. Isto é conseguido através da incorporação de comportamentos, crenças, desejos e intenções que irão guiar o agente para tomar decisões.

A arquitetura do sistema é dividida em três camadas: Projeto, Planejamento de Processo e Fabricação. Na camada de fabricação reside o agente de fabricação, chamado *Webturning*. Este permite teleoperação de um centro de torneamento CNC via Internet e a monitoração do processo de torneamento.

O *Webturning* é baseado em uma arquitetura cliente-servidor utilizando a metodologia descrita em Alvares e Romariz (ALVARES; ROMARIZ, 2002). É constituído de quatro módulos básicos, sendo três servidores e um cliente. Entre os módulos servidores estão:

- Servidores de *streaming* de vídeo e áudio (WebCam);
- Servidor Focas 1 (*Fanuc Open CNC API Specification*), sendo representado pelo CNC Fanuc 18i-ta do centro de torneamento;
- Servidor de teleoperação WebDNC que atua entre o CNC e o cliente usando mecanismos acesso através da Web, como CGI e *inetd*.

O cliente é a uma interface gráfica desenvolvido em HTML e Java que é acessível através de um navegador de Internet.

Por meio da WebDNC, servidor e também interface gráfica de usuário do *Webturning* (Figura 3.25), é possível a execução de comandos no CNC do centro de torneamento Galaxy 15M, da marca Romi. Esses comandos são possíveis graças a API e *driver* FOCAS 1/Ethernet que permite a comunicação com controlador através funções de DNC (Comando Numérico Distribuído). Essas funções incluem execução, *download* e *upload* de programas, tratamento de erros, entre outras funções disponibilizadas pelo fabricante na especificação da API.

O servidor WebDNC trabalha de modo bidirecional, recebendo comandos através da Internet e enviando dados sobre a operação e funcionamento do centro de torneamento via protocolo FOCAS1/DNC1. Esse servidor atua entre os clientes (Web) e servidor FOCAS1, localizado no centro de torneamento. Nessa arquitetura o servidor de vídeo e áudio WebCam é responsável pela captura de imagens e som da célula de manufatura onde está a máquina-ferramenta, e a realização da sua distribuição através do protocolo TCP/IP.

O *Webturning* foi uma das principais referências no desenvolvimento da arquitetura do sistema implementado neste trabalho. O modelo de operação cliente/servidor, bem como os serviços de teleoperação (WebCNC) e de streaming de vídeo/áudio (WebCam) foram incorporados ao *framework* implementado.

A teleoperação voltada para controle de sistemas robóticos, ou Telerrobótica, tendo a Internet como canal de comunicação, é objeto de trabalho de vários grupos de pesquisa em todo o mundo. Esses grupos produziram um significativo número de aplicações relevantes, dentre as quais, algumas foram relacionadas.

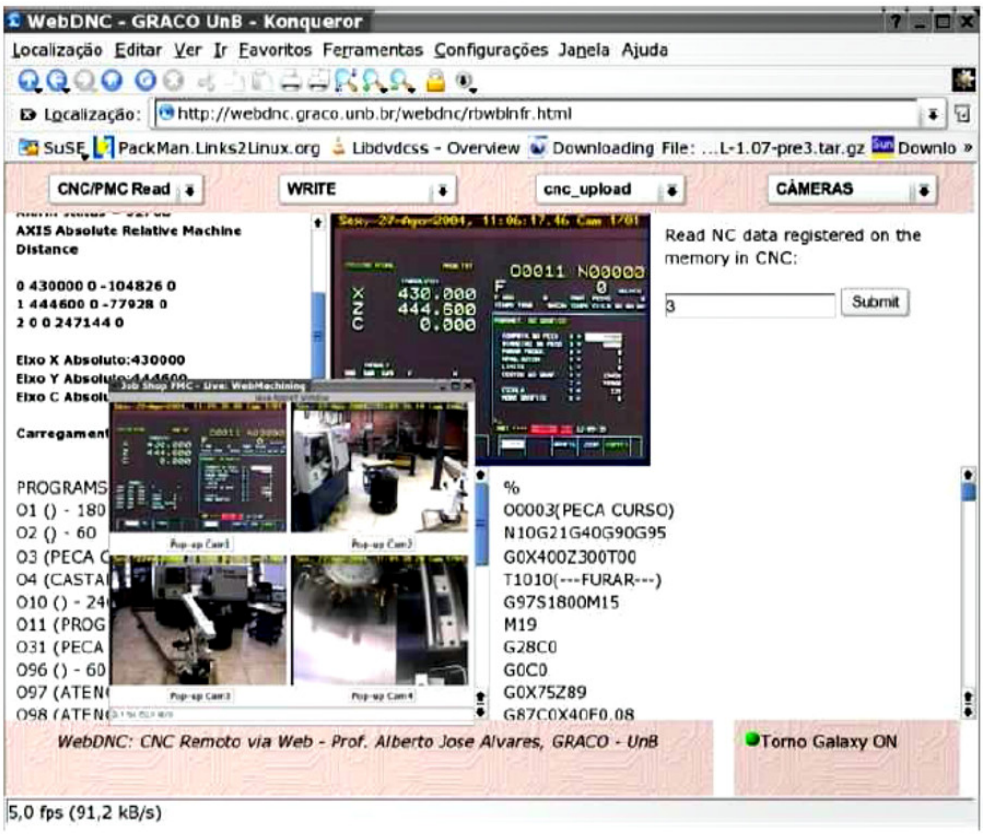

Figura 3.25: WebDNC: Interface gráfica de teleoperação e monitoramento remoto do sistema *Webturning* (ALVARES, 2005)

3.10.2 Restrições e Desafios da Teleoperação

A largura de banda e os atrasos inerentes ao protocolo TCP/IP impõem restrições técnicas aos sistemas teleoperados via Internet. Segundo Álvares et al. (ALVARES et al., 2002), para resolver esse problema é necessário dotar a aplicação, no lado servidor, de mecanismos que possibilitem a tomada de decisão em situações críticas, sem depender do lado do cliente, no caso do usuário/operador.

Como a Internet apresenta larguras de banda heterogênea, aplicações em tempo real para captura de vídeo apresentam sérias restrições. Para vencer essas restrições é necessário utilizar compressão de dados e conexão de Internet em alta velocidade (ALVARES et al., 2002).

Devido ao atraso inerente ao TCP/IP, deve-se ter cuidado nas ações de comando executadas remotamente. Para exemplificar, o menor tempo de ciclo de varredura de funções de CLP no centro de torneamento modelo Galaxy 15M da Romi é de 13 ms (ALVARES, 2005). O TCP/IP não permite uma aplicação em tempo real para tempos de ciclo dessa ordem de grandeza, sendo necessário tomar medidas de segurança quando da implementação de um sistema de teleoperação via TCP/IP (Web).

No caso da manufatura, uma alternativa para tornar os sistemas de teleoperação mais seguros é provê-los com serviços de *streaming* de dados associados aos atuais padrões conectividade MTConnect, para monitoramento em "tempo real"ou "quase tempo real", e OPC-UA para supervisão. Esses tecnologias permitiriam um nível significativo de confiança na atividade de teleoperação.

Capítulo 4

Revisão Bibliográfica: Projeto Axiomático (*Axiomatic Design*)

Este capítulo apresenta as principais etapas e os aspectos que caracterizam a metodologia de projeto axiomático e a sua extensão para o projeto de sistemas de software orientados objeto (ADo-oSS, sigla do inglês). Uma técnica conhecido por auxiliar no processo criativo durante as etapas iniciais do planejamento de projetos complexos.

4.1 Definição

O Professor Nam P. Suh, idealizador da metodologia de Projeto Axiomático (*Axiomatic Design* - AD), pondera em seu trabalho (SUH, 1990), que na perspectiva do AD o projeto representa a criação de soluções sintetizadas na forma de produtos, processos ou sistemas que satisfazem as necessidades percebidas do cliente através do mapeamento entre Requisitos funcionais (*Functional Requements*, FR) e Parâmetros de projeto (*Design Parameters*, DP). Segundo Gonçalves-Coelho (GONÇALVES-COELHO, 2009), é uma metodologia de projeto que usa métodos de matrizes a fim de desdobrar as necessidades dos clientes do projeto em requisitos funcionais (FR), parâmetros de projeto (DP) e variáveis de processo (PV).

Através da abordagem de AD o projeto inicia com o estabelecimento das necessidades dos clientes (*Customer Needs*, CN), e através dessas, abstrair os requisitos funcionais (FRs) do sistema projetado. Os FRs representam as metas do projeto ou o que (*What*) precisa ser realizado, e correspondem ao domínio funcional do projeto. Os Parâmetros de Projeto (DPs) representam o como (*How*) os requisitos funcionais são satisfeitos e formam o domínio físico. A Figura 4.1 mostra os domínios de projeto.

As necessidades dos clientes integram o domínio do consumidor. Essas necessidades devem ser mapeadas para o domínio funcional onde são traduzidas em um conjunto de requisitos funcionais (FRs), os quais, por definição, devem serem independentes. Também são definidas restrições (*Constraits*, Cs) para o novo projeto, na forma de premissas fundamentais do mesmo. As restrições devem ser obedecidas durante todo o processo de projeto e estão relacionadas aos FRs, DPs e variáveis de processo (*Processes Variables*, PVs), conforme ilustra a Figura 4.1. Os FRs são mapeados para o domínio físico e DPs são mapeados para

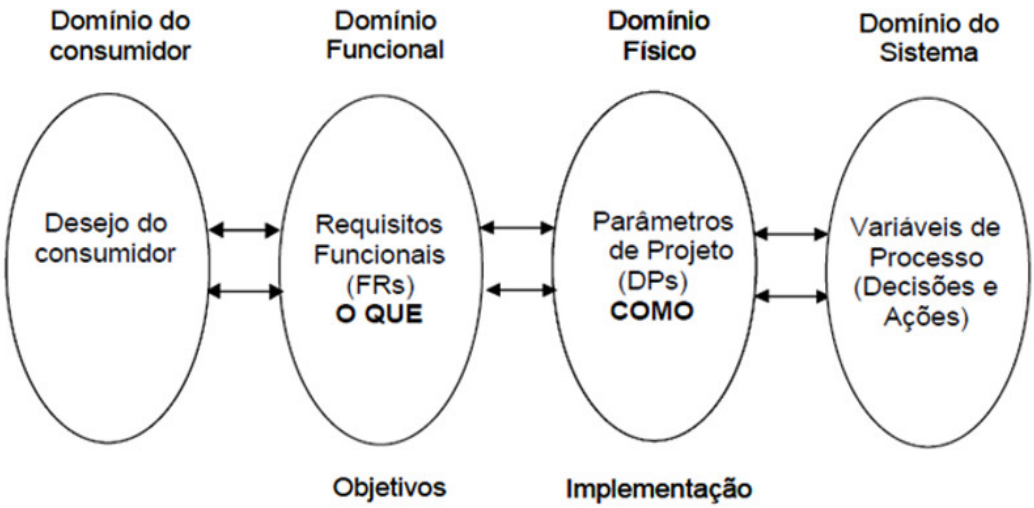

Figura 4.1: Domínios de projeto (SUH, 1990)

o domínio de processo na forma de variáveis de processo.

4.2 Decomposição de um Projeto

O mapeamento entre domínios significa o detalhamento do projeto de um produto ou sistema em um procedimento de decomposição por "zigue-zague"entre domínios (funcional e físico), formando uma estrutura hierárquica de projeto. Após a definição de um requisito funcional de mais alto nível, o parâmetro de projeto correspondente deve ser selecionado. Uma vez que um requisito funcional foi atendido por um correspondente parâmetro de projeto, esse FR pode ser decomposto em sub-requisitos, em um processo de repetição. Esse procedimento de decomposição em "zigue-zagues"é ilustrado na Figura 4.2.

Para que o mapeamento entre domínios possa ser satisfatório é necessária à obediência ao Axioma 1 - da independência, e o Axioma 2 - para a minimização do conteúdo da informação de projeto (SUH, 1990).

A relação de mapeamento de FRs em DPs é representada por um vetor. A matriz de projeto (DM - *Design Matrix*) corresponde ao relacionamento entre FRs e DPs.

$$\{FRs\} = \{DM\}\{DPs\} \tag{4.1}$$

Um elemento qualquer da matriz de projeto é dado por:

$$DM_{ij} = \frac{\partial FR_i}{\partial DP_j} \tag{4.2}$$

que forma uma constante em projetos lineares. Para satisfazer o Axioma da independência a matriz de projeto resultante deve corresponder a uma matriz diagonal ou triangular. No primeiro caso a matriz é chamada de projeto desacoplado, e no segundo caso, quando o resultado do mapeamento é uma matriz

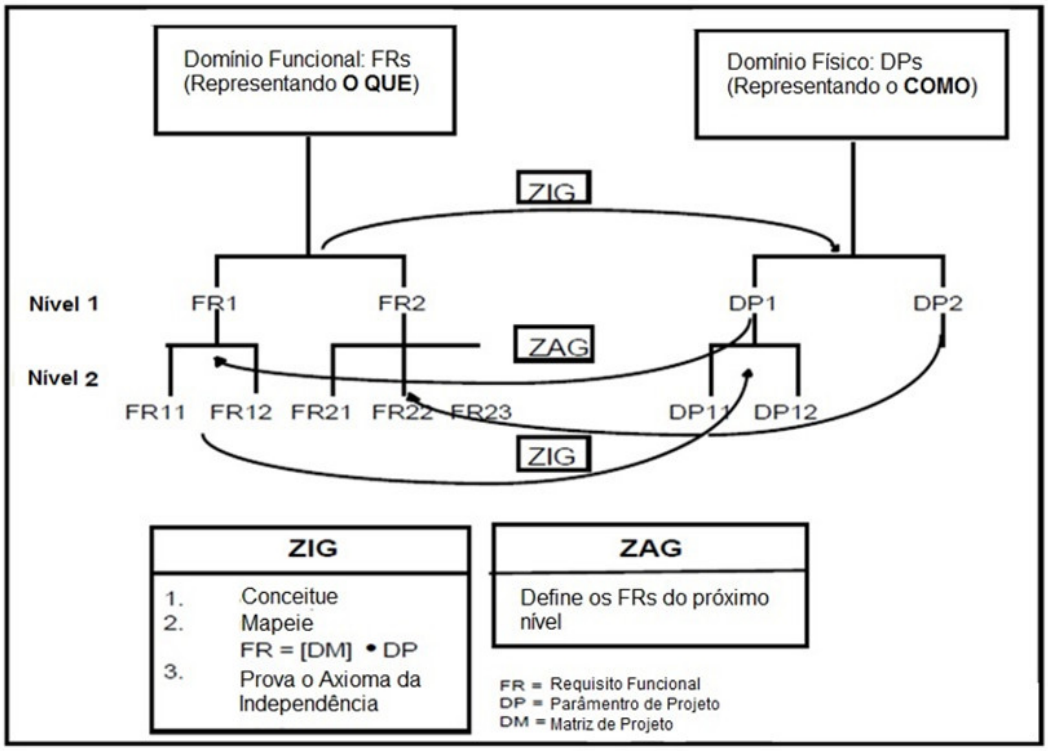

Figura 4.2: Zigue-Zague entre domínios para definir a hierarquia de projeto (SUH, 1990)

triangular, corresponde ao caso de um projeto semi-acoplado. Um terceiro tipo é o projeto acoplado, que não corresponde aos outros dois casos. Graficamente, os três casos podem ser representados conforme mostra a Figura 4.3.

O Axioma da informação, referente ao conteúdo da informação, é definido como a probabilidade logarítmica de um dado parâmetro de projeto (DP) satisfazer um determinado requisito funcional (SUH, 1990). Para Cochran e Reynal (COCHRAN D.S.AND REYNAL, 1996), esse axioma representa a probabilidade de obter com sucesso FRs e DPs. Dessa forma, o conteúdo da informação pode ser calculado como:

$$I = \sum_{i=1}^{n} [\log_2 \frac{1}{p_i}] \tag{4.3}$$

onde pi é a probabilidade de DPj satisfazer FRi, e n o numero total de FRs.

4.3 Projeto Axiomático de Software

Inicialmente o projeto axiomático representou um poderoso conjunto de procedimentos para auxiliar no desenvolvimento de projetos otimizados de sistemas de hardware. No entanto, percebeu-se o potencial do projeto axiomático em ser aplicado a outras disciplinas como sistemas de manufatura e engenharia (CLAPIS P.J.AND HINTERSTEINER, 2000). Esse fato fez com que houvesse o interesse em estender o escopo de aplicação do Projeto Axiomático a projetos menos tradicionais, como o desenvolvimento de

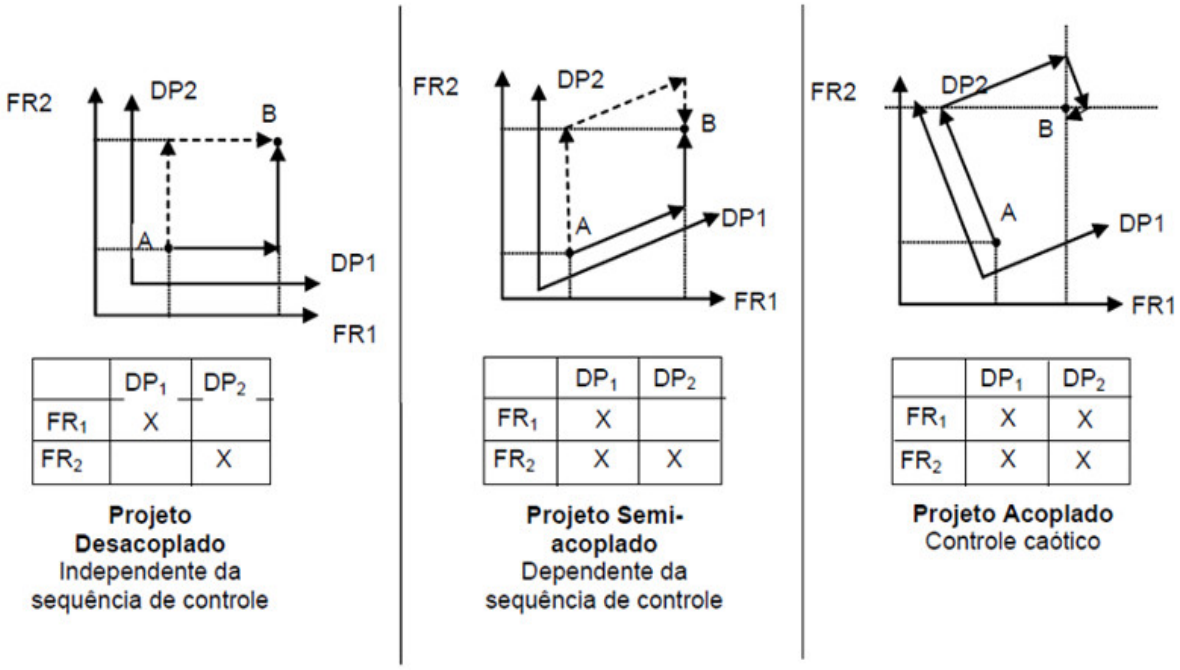

Figura 4.3: Tipos de projeto (COCHRAN D.S.AND EVERSHEIM; SESTERHENN, 2000)

software a partir dos anos de 1990.

Um trabalho pioneiro sobre o projeto de software baseado em Projeto axiomático foi realizado por Kim, Suh e Kim (KIM S.-J.AND SUH; S.-G., 1991). Nesse trabalho os autores fornecem as bases científicas para o projeto de software, como uma forma de superar a carência de uma metodologia que possuísse um conjunto de princípios fundamentais e fornecesse critérios para tomadas de decisão.

O projeto de sistemas de software baseados na Teria de Projeto Axiomático (AD) independe de linguagem de programação ou algoritmo computacional. A metodologia fornece uma plataforma para o projeto de software de todos os sistemas. Isso inclui tanto aqueles sistemas que envolvem apenas software, quanto àqueles sistemas formados por hardware e software. De acordo com Suh (SUH, 2001), "o projeto de software baseado em projeto axiomático é auto-consistente, provê inter-relacionamentos e combinações entre módulos, são fáceis de trocar, modificar e estender. Sendo esse o resultado de tomadas de decisões corretas em cada estágio de mapeamento e decomposição". De acordo com mesmo autor, a importância que o software adquiriu em várias áreas exigiu um projeto de software baseado menos em metodologias *ad hoc*, e mais em abordagens sistemáticas como o projeto axiomático.

A abordagem axiomática para o projeto de software foi planejada para superar algumas das deficiências das práticas existentes e criar uma plataforma cientifica para a construção de software. Conforme Suh (SUH, 2001), o AD difere significativamente várias metodologias utilizadas na engenharia de software:

- É baseada na ideia de que o software deve ser projetado primeiro para atender um conjunto de requisitos funcionais e restrições antes de iniciar o processo de codificação. Também prove um projeto baseado em atividades como mapeamento, decomposição, e a criação de uma hierarquia de requisitos.

- Fornece um mecanismo rigoroso de decomposição e critérios de aceitação para o projeto do software. Tem como uma das principais saídas da metodologia o diagrama de fluxo, que descreve o exato relacionamento entre todos os módulos do sistema, e a visão da arquitetura do software e sua sequencia computacional.

- O projeto axiomático é aplicável a todos os projetos de software, seja ele formado apenas por módulos de software, seja ele parte de uma arquitetura que inclui sistemas de hardware.

- A estrutura do projeto axiomático não é especifica para um campo de atividade, como controle de maquinas, montagem, telecomunicações, etc. Os elementos adicionais introduzidos para o projeto de sistemas é igualmente válido para o projeto de software, como: diagramas de fluxo para o projeto de software, o conceito de "nível folha", e as características específicas de cada domínio.

- O diagrama de fluxo para a arquitetura do sistema é gerado como resultado da implementação do projeto axiomático. Diferentemente das abordagens heurísticas que dependem apenas da intuição e habilidade do programador. O uso do fluxograma organiza o trabalho colaborativo, permitindo a pronta identificação do papel de cada programador na equipe de projeto.

- Projeto axiomático cria software para controle em tempo real de sistemas de uma forma explicita. Dessa forma, o software pode ser eficiente, minimizando o conteúdo da informação do sistema.

- É uma boa ferramenta para a modelagem de sistemas que consistem de hardware e software porque a matriz de projeto gerada estabelece as relações entre FR/DP a serem modeladas.

- A metodologia de projeto axiomático permite reusabilidade e extensionalidade de módulos de software de forma explicita.

No contexto do projeto axiomático de software, a metodologia de AD também foi associada às técnicas de orientação a objeto. O uso exclusivo dos recursos de orientação a objeto não garantem confiabilidade ao software. A TOO não possui uma base fundamental para o projeto. Mesmo com a orientação a objeto, há muitos problemas que os programadores de software enfrentam durante o desenvolvimento e a manutenção do software no seu ciclo de vida. Segundo Suh (SUH, 2001), há várias razões para essas dificuldades, a principal é a falta critério para a classificação de um projeto como bom.

4.4 Teoria de Projeto axiomático para o Projeto de Software

O projeto de sistemas software não é muito diferente do projeto de outros tipos de sistemas. Na perspectiva do projeto axiomático, o software é um subconjunto de um projeto de sistema, e nesse sentido, está sujeito aos mesmos princípios e procedimentos da abordagem axiomática original, como aos axiomas da independência e da informação, a possuir domínios mapeáveis entre si, decomposição através de "zigue-zague", matriz de projeto, e a criação de hierarquia. Além disso, o conceito de requisito e parâmetro de "nível folha", e recursos como diagrama de estrutura de junção de módulo e a criação e o uso de fluxograma para definir a sequencia de execução do sistema, são igualmente validos para o projeto de software (KIM S.-J.AND SUH; S.-G., 1991; DO, 1999).

Similarmente a outros projetos de produto ou processo, no projeto de software também há uma interação entre os domínios de projeto que corresponde na forma de "o que"se quer fazer e o "como"alcançar o que se deseja. Isso é traduzido através da definição dos vetores correspondentes aos domínios e a posterior construção da arvore de projeto pelo processo de mapeamento e decomposição entre domínios.

No projeto axiomático de um sistema de software as características específicas das necessidades dos clientes (CN - *Costumer needs*), dos requisitos funcionais (FR - *Functional Requirement*), parâmetros de projeto (DP - *Design parameter*) e das variáveis de processo (PV - *Process variable*) dependem da natureza e dos objetivos do software (SUH, 2001). No contexto do desenvolvimento de software, as necessidades dos clientes (CN) residem no domínio dos usuários e são atributos que o software deve satisfazer. Os requisitos funcionais (FR), no domínio funcional, são saídas, especificações ou requisitos do sistema de software. Os DPs, na interpretação de Suh (SUH, 2001) podem ter os seguintes significados:

1. Entradas dos módulos no caso de algoritmos. Na programação orientada a objeto podem ser tratados como dados.

2. Sinal dos sensores de hardware e circuitos integrados de aplicação especificas (ASICs) para sistemas que envolver software e hardware.

3. Código do programa que gera entrada para os módulos.

As variáveis de processo (PVs) podem representar sub-rotinas, códigos de maquina ou compiladores. Os módulos correspondem a uma linha da matriz de projeto (DM - Design Matrix), mais precisamente o relacionamento formado quando um parâmetro de projeto é fornecido.

As etapas de um projeto axiomático de software são idênticas àquelas usadas no projeto de outros sistemas, mas requer uma base de conhecimento diferente.

Etapa 1 - Definição dos requisitos funcionais (FRs)

Inicialmente é necessária a determinação das necessidades dos usuários (CNs) as quais o sistema de software deve satisfazer. Em seguido, essas necessidades devem se traduzidas no domínio funcional na forma de requisitos funcionais (FRs), nesse momento também determinam-se as restrições de projeto (Cs). Os requisitos funcionais devem ser determinados se pensar-se nos DPs. Os FRs são definidos como o mínimo conjunto de requisitos independentes que o projeto deve satisfazer (SUH, 2001).

Etapa 2 - Mapeamento entre domínios

Próximo passo consiste em mapear FRs em DPs , do domínio funcional para o domínio físico, respectivamente. Verifica-se a aceitabilidade do conjunto de DPs através da obediência ao axioma da independência. A escolha dos DPs também deve considerar as restrições de projeto (Cs). Os DPs devem ajudar a mapear as variáveis de processo (PVs) no domínio de processos. As PVs podem representar sub-rotinas, códigos de máquina ou compiladores. No projeto de um sistema software os FRs representam as saídas de um software e os DPs entradas as quais caracterizam/controlam os FRs. O código de software corresponde a um conjunto de matrizes de projeto que tranformam DPs em FRs, em cada nível da hierarquia. A árvore hierárquica gerada representa o processo de projeto (SUH, 2001).

No processo de mapeamento, o projeto deve satisfazer o axioma da independência. Pelo axioma da independência, a independência funcional deve ser satisfeita através do desenvolvimento de projetos desa-

coplados ou semi-acoplados, teoricamente representados por uma equação da seguinte forma:

$$\begin{Bmatrix} FR_1 \\ FR_2 \end{Bmatrix} = \begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \end{Bmatrix}$$

Num projeto considerado ideal, o número de FRs é igual ao numero DPs. Isso é uma consequência do axioma da independência e do axioma da informação. Na escolha dos DPs, é preciso estar atento às restrições (Cs) e ao conteúdo da informação.

Os FRs, DPs e PVs devem ser decompostos até o "nível folha"na hierarquia de projeto. Nessa etapa são formados os módulos, que corresponde a linhas da matriz de projeto.

Etapa 3 - Matriz de Projeto e Diagrama de junção de módulos

No "nível folha"qualquer linguagem de programação de software, como C ++ ou Java, pode ser utilizada para escrever os módulos, desde que sejam consistentes internamente. Java requer o uso de técnicas de orientação a objetos.

Um meio para representar a relação entre módulos é o "diagrama de estrutura de junção de módulos". A junção de soma, S, é usada para representar um design desacoplado; A junção de controlo, C, é utilizada para a junção semi-acoplada; E a junção F é usada para um projeto acoplado. O mesmo sistema de notação será usado para o projeto de software.

Etapa 4 - Representação da Arquitetura do Sistema de Software

O fluxograma é utilizado como meio para representar a arquitetura do sistema, corresponde aos módulos de folhas. É considerado o meio mais conveniente para representar sistema de software. Com base no fluxograma, módulos individuas podem ser desenvolvidos e relacionados para criar um sistema de software para produzir FRs de mais alto nível.

O fluxograma também fornece o esquema de implementação de software, que é simples e gráfico. O fluxograma mostra a sequencia de computação /operação do software para atingir os FRs de mais alto nível. Para operar o software, a operação deve sempre começar a partir dos módulos mais internos que representam os níveis folha mais baixos em cada ramo principal. Em seguida, passa-se para os módulos de nível superior seguinte, acompanhando a sequencia indicada pela arquitetura do sistema.

A execução da arquitetura do sistema pode ser feita usando o comando de controle do sistema (SCC), que às vezes é chamado d comando do sistema e módulo de controle.

4.4.1 Projeto Axiomático de Sistemas de Software Orientados a Objeto

A conjunção entre os métodos Orientados a Objeto e o Projeto Axiomático permitiu a geração de uma abordagem genérica para o projeto de software chamada ADo-oSS (*Axiomatic Design of Object-Oriented Software Systems*). Esta metodologia está estruturada sob o modelo em V (Fig. 4.4) e combina o poder do Projeto Axiomático com as populares técnicas de programação orientada a objeto, e surgiu como uma maneira de superar as deficiências das técnicas existentes para suporte ao projeto de software, como extensos ciclos de depuração e testes. Essa metodologia tem como resultado final a arquitetura do sistema,

representada pelo fluxograma.

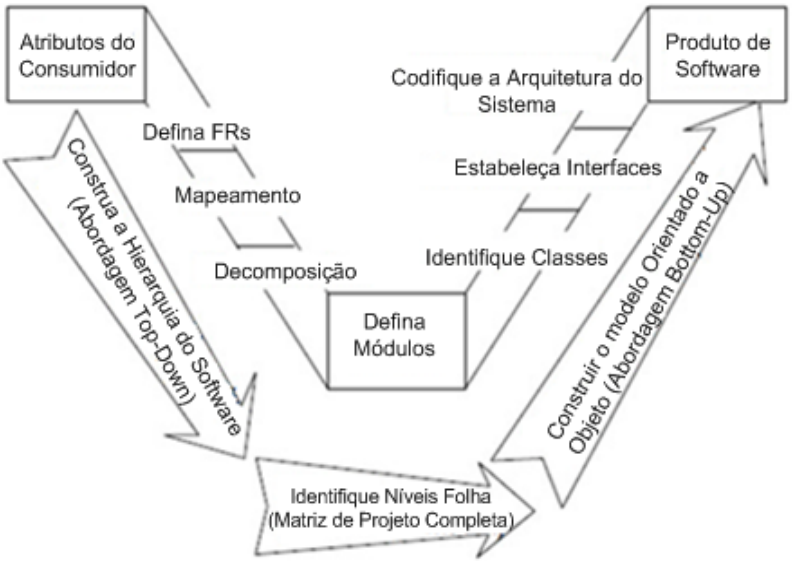

Figura 4.4: Processo de Projeto Axiomático para Sistema de Software Orientado a Objeto (Modelo V)

O fluxograma gerado pode ser aplicado a diferentes fins, por exemplo (SUH; DO, 2000a):

a) Melhoria do projeto proposto através da identificação de projetos acoplados.

b) Diagnóstico de falha iminente de um sistema complexo.

c) Redução do custo do serviço de manutenção de máquinas e sistemas.

d) Ordens de mudança de engenharia.

e) Atribuição de tarefas e gerenciamento de tarefas de projeto.

f) Gestão de tarefas de projeto distribuídas e colaborativas.

g) Reutilização e extensionalidade do software.

Utilizando o conceito de ADo-oSS, primeiro passo é projetar o software seguindo a abordagem de cima para baixo (top-down) do projeto axiomático, construir a hierarquia do software e, em seguida, gerar a matriz de projeto completa (isto é, a matriz de projeto que mostra toda a hierarquia do projeto) para definir os módulos. O passo final é construir o modelo orientado a objetos com uma abordagem de baixo para cima, seguindo o diagrama de fluxo do AD para o sistema projetado.

Para compreender ADo-oSS, é necessário relembrar as definições dos termos usados na técnica de Orientação a Objeto, e conhecer suas palavras equivalentes em projeto axiomático. O elemento fundamental na orientação a objeto é o Objeto (*Object*), que equivale a FRs. O projeto orientado a objetos decompõe um sistema em objetos. Os objetos "encapsulam"tanto os dados (equivalente a DPs) como o métodos (equivalente à relação entre FRi e DPi, ou seja, módulo) em uma única entidade. O objeto retém determinadas

informações sobre como realizar certas operações, usando a entrada fornecida pelos dados e métodos incorporados no Objeto. Em projeto axiomático um Objeto pode ser representado pela formula: FRi = Aij DPj (SUH; DO, 2000a).

A Orientação a objetos possui quatro definições para descrever suas operações: identidade, classificação, polimorfismo e relacionamento. Identidade significa que dados - equivalentes a DPs - são incorporados em objetos específicos. Os objetos são equivalentes a um FR - com uma relação especificada [FRi = Aij DPj] - de projeto axiomático, onde DPs são dados ou entradas, e Aij é um método ou um relacionamento. Classificação significa que os objetos com a mesma estrutura de dados (atributos) e comportamento (operações ou métodos) são agrupados em uma classe. O objeto é representado como uma instância de classe específica em linguagens de programação.

Às vezes um Objeto exibe um comportamento, que é um caso especial de FR. A relação entre Objetos e Comportamentos é comparada à decomposição de FRs na hierarquia de FR do projeto axiomático. Objeto é o "FR pai"em relação ao Comportamento o qual é o "FR filho"(SUH; DO, 2000a).

Segundo Suh (SUH, 2001), a distinção entre Superclasse, Classe, Objeto e Comportamento é necessária na Orientação a Objeto para tratar de FRs em camadas sucessivas de um projeto de sistema. Uma classe, como uma abstração de objetos, está no mesmo nível que um Objeto na hierarquia de FRs. No entanto, o Objeto é um nível superior ao Comportamento na hierarquia de FRs. O uso dessas palavras-chave, embora necessário em TOO, acrescenta complexidade desnecessária quando os resultados do projeto axiomático devem ser combinados com TOO. Portanto, é modificado o uso dessas palavras-chave em TOO.

Dessa forma, os Objetos são associados a índices para representar todos os níveis de FRs, ou seja, Classe, Objeto e Comportamento. Classe ou Objeto pode ser chamado Objeto i, que é equivalente a FRi. O Comportamento será denotado como Objeto ij que corresponde a FRij, um FR de segundo nível. Por outro lado, um FRijk de terceiro nível será denotado como Objeto ijk. Logo, Objetos i, ij e ijk são equivalentes a FRi, FRij e FRijk, respectivamente, que são FRs de três níveis sucessivos da hierarquia de FR.

Estabelecida a equivalência entre a terminologia de Projeto Axiomático com a Metodologia de Orientação a Objeto, as etapas do Projeto Axiomático de Sistema de Software Orientado a Objeto (ADo-oSS) podem ser divididas em:

a) Definição dos FRs do sistema de software - nessa primeira etapa na concepção de um sistema de software são determinados os atributos do cliente, no domínio do cliente, que o sistema de software deve satisfazer. Então, os requisitos funcionais (FRs) do software no domínio funcional e restrições (Cs) são estabelecidos para satisfazer as necessidades do cliente.

b) Mapeamento entre Domínios e a Independência funções de software - Neste passo mapeiam-se os FRs do domínio funcional no domínio físico, identificando os parâmetros de projeto (DPs). Os DPs são o "como"do projeto que satisfazem FRs específicos. Os DPs devem ser escolhidas de modo a serem consistentes com as restrições.

c) Decomposição de FRs, DPs, and PVs - Essa decomposição ocorre entre domínios por um método de "zigue-zague"e deve ocorrer até que o projeto possa ser implementado sem mais decomposição. As hierarquias de FRs, DPs, PVs e as matrizes correspondentes representam a arquitetura do sistema.

d) Definição de Módulos - Matriz de Projeto Completa - No caso do software, a matriz de projeto fornece duas bases importantes na criação de software. Uma base é que cada elemento na matriz de projeto pode ser um método (ou operação) em termos do método orientado a objetos. A outra base é que cada linha na matriz de projeto representa um módulo para satisfazer um FR específico quando um dado DP é fornecido. Os termos fora da diagonal na matriz de projeto são importantes, uma vez que as fontes de acoplamento são esses termos fora da diagonal. É importante construir a matriz de projeto completa com base no nível de folha FR-DP-Aij para verificar a consistência das decisões tomadas durante a decomposição.

e) Identificação de Objetos, Atributos e Operações - A folha é o objeto de nível mais baixo em um dado ramo de decomposição, mas todos os objetos de nível de folha podem não estar no mesmo nível se pertencerem a ramos de decomposição diferentes. Uma vez que os Objetos são definidos, os Atributos (ou dados) - DPs - e Operações (ou métodos) - produtos do módulo vezes DPs - para o Objeto devem ser definidos para construir o modelo de objeto. Essa atividade deve usar a tabela da matriz de projeto completa.

A tradução da matriz de projeto, com FRs e DPs, em uma estrutura Orientada a Objeto é ilustrada pela Figura 4.5.

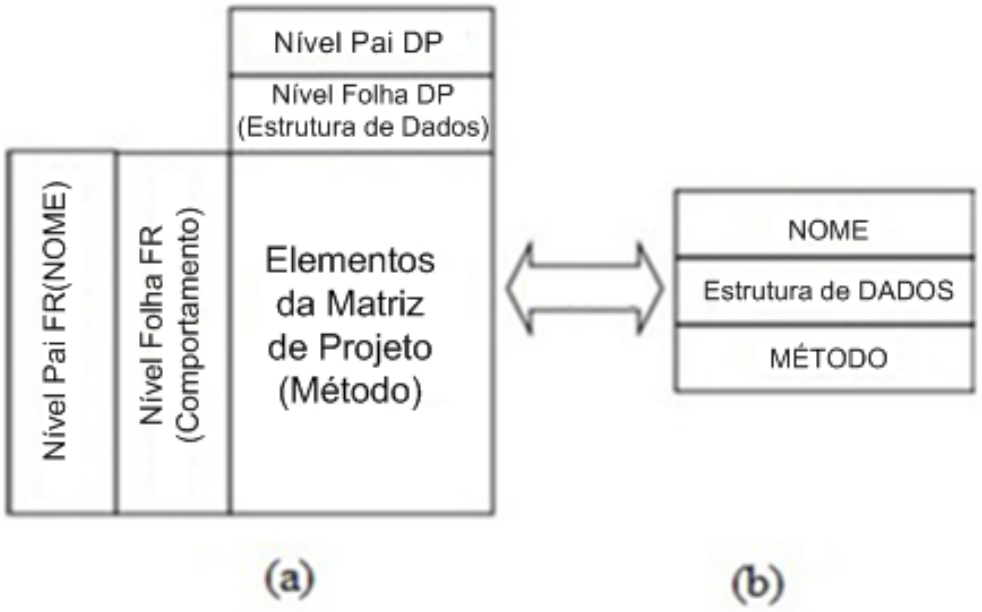

Figura 4.5: A correspondência entre a matriz de projeto completa e o diagrama de Classes (a) Matriz de projeto completa, (b) Diagrama de Classe, adaptado de (SUH; DO, 2000b)

f) A maioria dos esforços está focada nesta etapa do método orientado a objeto, uma vez que a relação é a principal característica. A metodologia de projeto axiomática apresentada utiliza os elementos fora da diagonal na matriz de projeto, bem como os elementos da diagonal em todos os níveis. Um elemento de matriz de projeto representa um relacionamento de ligação ou associação entre diferentes ramos de FR que têm um comportamento totalmente diferente.

Como foi citado anteriormente, todo processo de projeto axiomática inicia após o conhecimento das necessidades dos clientes do projeto e a para isso o uso de ferramentas e metodologias complementares para auxiliar na apreensão dessas informações de entrada se faz necessário. Nesse sentido, o projeto axiomático foi aplicado com o suporte da linguagem IDEF0 e da pesquisa bibliográfica específica para definição da necessidades dos potenciais usuários do *framework* proposto, que serão detalhadas no próximo capítulo.

Capítulo 5

Metodologia para a Concepção do *Framework*

5.1 Introdução

Neste capítulo apresenta-se a abordagem metodológica adotada para o projeto do *framework*. A proposta do trabalho trata de um sistema cliente-servidor que envolve diferentes módulos conectáveis através da internet, e assim, um sistema considerado complexo. Inicialmente, antes da aplicação da metodologia de Projeto Axiomático (do inglês, *Axiomatic Design* - AD), optou-se pelo implementação da modelagem IDEF0 (*Integration Definition language for function modeling*), que fornece uma visão geral dos múltiplos níveis de detalhamento das atividades que compõem o *framework*, enfatizando as principais entradas, saídas, os recursos utilizados (mecanismos) e regras (controles) do *framework*. Esses diagramas auxiliam no levantamento das necessidades dos usuários, que são importantes para o projeto axiomático, atuando como uma entrada para esta outra metodologia.

A técnica de Projeto Axiomático ajuda a modelar os detalhes do projeto pela especificação de parâmetros que compõem os seus módulos até a geração da arquitetura detalhada do sistema, juntamente com a sua sequência de implementação expressa no diagrama de fluxo gerado como resultado.

Por fim, utilizou-se a modelagem UML (*Unified Modeling Language*) para documentar detalhes dos módulos da arquitetura do sistema projetado, que possuem elementos de software programados com linguagem de programação orientada a objeto, e para interface gráfica de usuário desenvolvida. Diagramas de Classe UML são produzidos como um dos resultados da aplicação do "Projeto Axiomático para Sistemas do Software Orientados da Objeto"(do inglês, *Axiomatic Design of Object-Oriented Software Systems* - ADo-oSS), processo utilizado no contexto da aplicação da metodologia de projeto axiomático para os módulos do sistema com arquitetura orientada a objeto.

5.2 Modelagem IDEF0 do *Framework*

O diagrama IDEF0 fornece uma visão do fluxo de informações no *framework* e um gradual detalhamento das funções ou atividades (representado pela componentes cliente e servidor) que o compõem. As informações que percorrem as atividades e funções são classificadas em entrada (programa NC, por exemplo), saídas (monitoramento da máquina CNC e do processo, por exemplo), mecanismos, que representam recursos usados pelas atividades (máquina-ferramenta CNC, internet, banco de dados, etc), e controles, que correspondem regras que regulam e direcionam as atividades (especificações, protocolos, padrões, fluxo de realimentação, etc).

A utilização do IDEF0 dá a arquitetura do *framework* um aspecto mais genérico com a finalidade de permitir que a mesma atue como uma referência geral para futuros projetos. Nesse sentido, o nível principal da modelagem (A0), representado na Figura 5.1, leva o nome de *framework*.

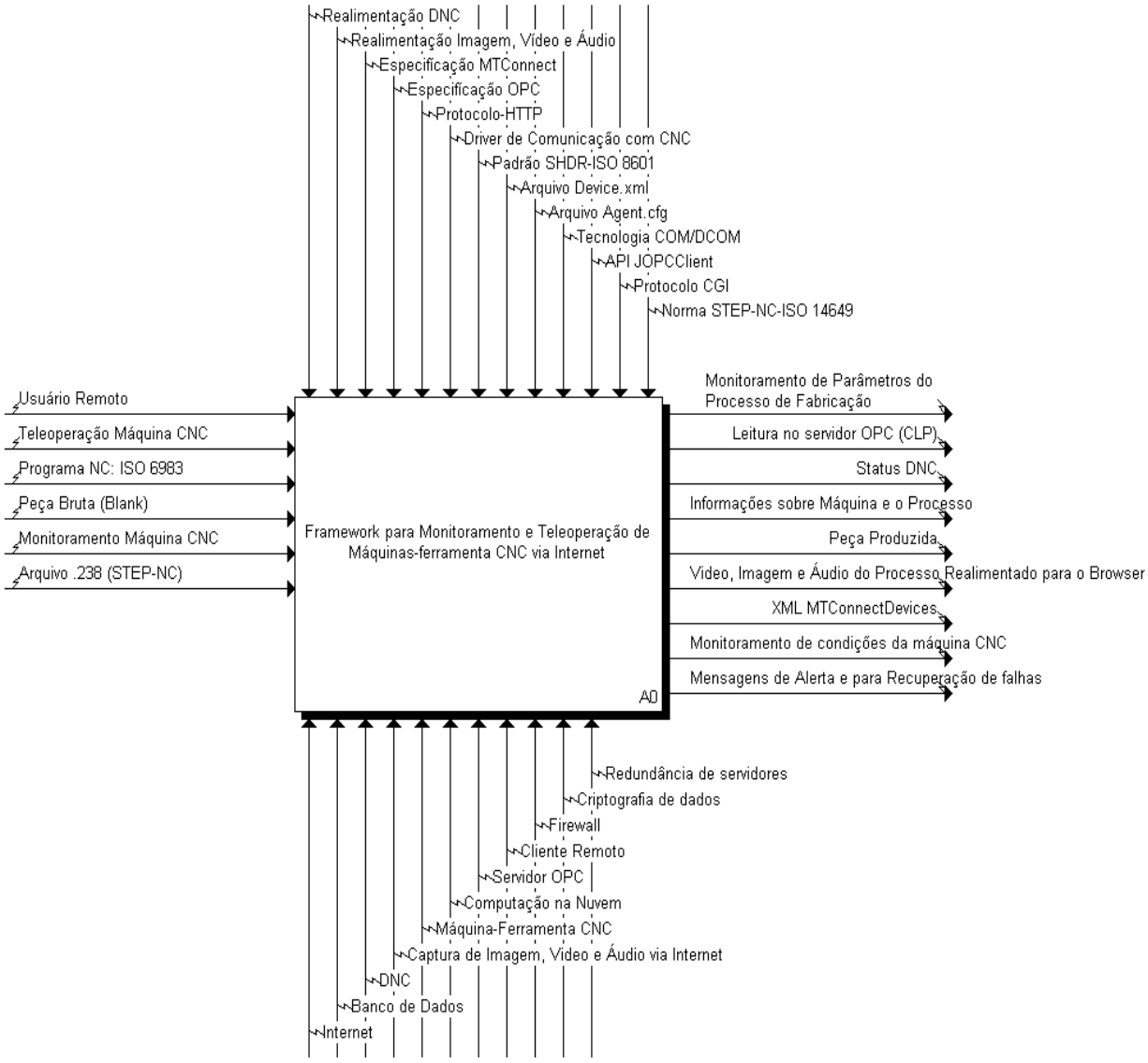

Figura 5.1: Modelagem IDEF0: *Framework* nível A0

Essencialmente, as entradas de um sistema projetado a partir dessa arquitetura contam com a atividade de monitoramento, a teleoperação, usuários situados remotamente, a peça bruta (*blanck*), que é uma entrada para a máquina-ferramenta onde está situado o CNC, e o programa NC, que contem o código de fabricação da peça.

As saídas do *framework* envolvem o produto dos serviços de monitoramento da máquina e do processo de fabricação, dos serviços de teleoperação, supervisão e monitoração visual.

Os mecanismos da arquitetura, representados pelas entradas da parte inferior da atividade, envolvem os aspectos essenciais que compõem a plataforma de monitoramento e teleoperação, que nessa modelagem possuem como uma de suas funções, posicionar o projeto no contexto dos sistemas para a Indústria 4.0, através da inclusão de recursos como: internet, computação na nuvem, bancos de dados (na nuvem), imagens de vídeo do chão-de-fábrica em tempo real, entre outros que estão representados na Figura 5.1.

Os controles representam basicamente as especificações candidatas a padrões para a Indústria 4.0, como a OPC (OPC-UA) e o MTConnect, protocolos para Web como o HTTP, e APIs (*Application Protocol Interfaces*) sugeridas como as principais para possibilitar a comunicação com os módulos servidores.

A decomposição do nível principal do modelo do *Framework* (Figura 5.2) resulta na estrutura básica do sistema, que é do tipo cliente-servidor. Na Figura 5.2, a atividade à direita corresponde aos servidores (A2), onde estão concentrados os serviços de teleoperação, monitoramento e supervisão. Na atividade cliente (A1) estão reunidos os clientes Web desses serviços, estejam eles utilizando um *browser* em um PC ou através de uma aplicação para dispositivos móveis.

Como na nova era da indústria os sistemas estão sendo pensados para serem independentes de plataforma pela disponibilização de dados e informações em formatos universais, os clientes dessa plataforma são formados por clientes com acesso a internet por meio de um software navegador, para atividades de teleoperação, monitoramento e supervisão, e clientes em aplicações para dispositivos móveis tipo *smartphone* e *tablet*, com funções de monitoramento/supervisão. A Figura 5.3 apresenta os clientes possíveis (A11, A12 e A13) com suas entradas, saídas, controles e mecanismos associados.

A Figura 5.4 mostra a estrutura fundamental e as comunicações dos servidores que compõem o *Framework*. Na arquitetura proposta, o monitoramento é realizado pelo serviço designado como Monitoramento/Supervisão (A21) que tem como insumos principais dados e comandos que formam basicamente requisições web. As saídas dessa atividade envolvem dados de supervisão e informações relacionadas ao status da máquina e do processo.

A atividade A22, servidor WebCam, que provê serviços *streaming* de vídeo e áudio, é parte dos recursos de teleoperação, mas devido ao pressuposto lógico de que é projetado com uma tecnologia diferente daquela empregada no servidor WebCNC, foi descrito como uma atividade separada.

O servidor WebCNC (A23), por sua vez, está diretamente vinculado ao CNC da máquina-ferramenta. Esse serviço concentra os algoritmos que fornecem os comandos associados a funções de DNC (Comando Numérico Distribuído, traduzido do inglês) que são executadas no controlador da máquina.

O servidor de Monitoramento/Supervisão ainda agrega o servidor de monitoramento MTConnect (A211) e o servidor OPC de supervisão via Internet, OPCWeb (A212), conforme representa a Figura 5.5. O servidor MTConnect produz principalmente o monitoramento através de *streaming* de dados de fabricação em

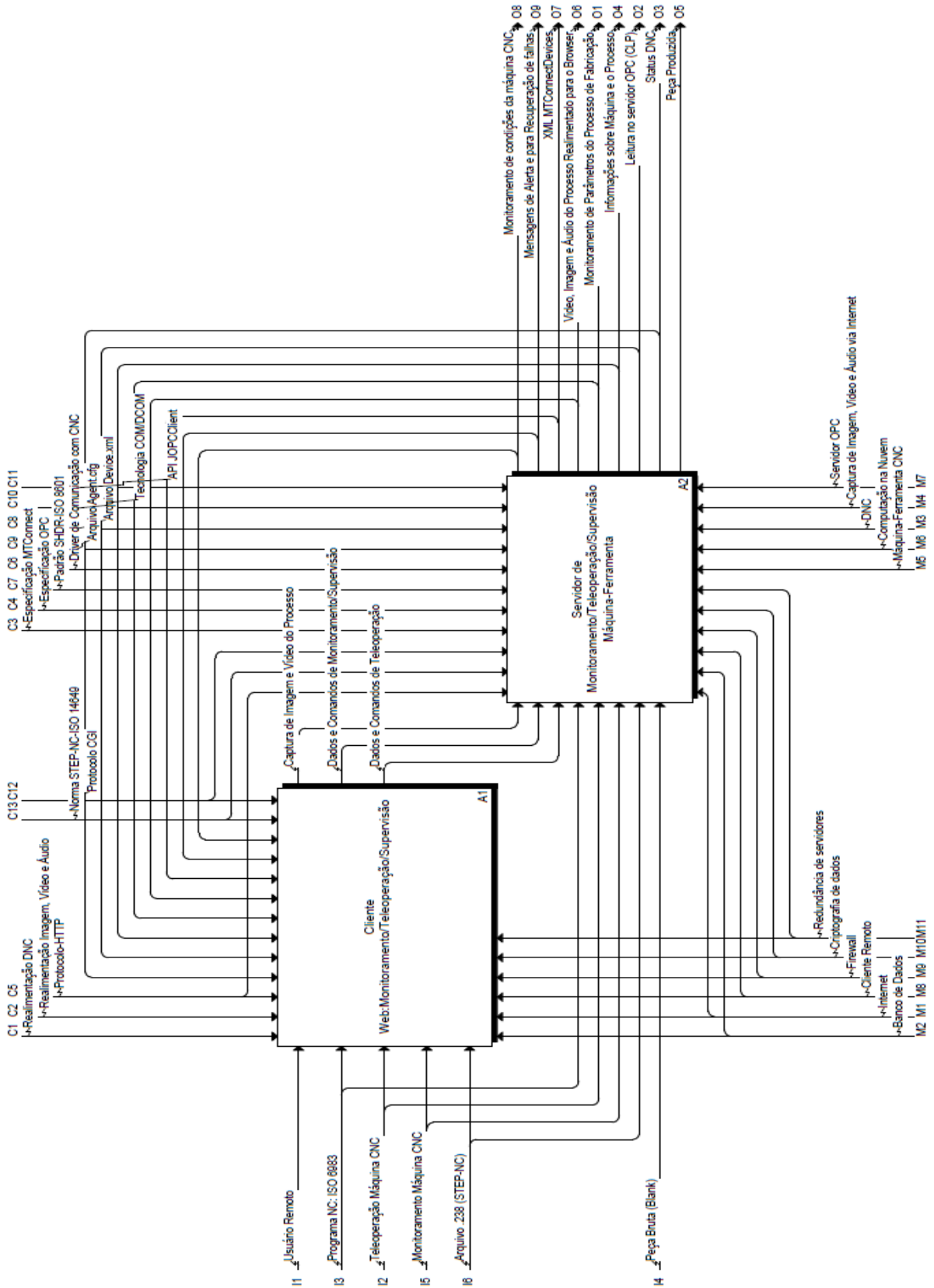

Figura 5.2: Atividades Cliente e Servidor da arquitetura proposta

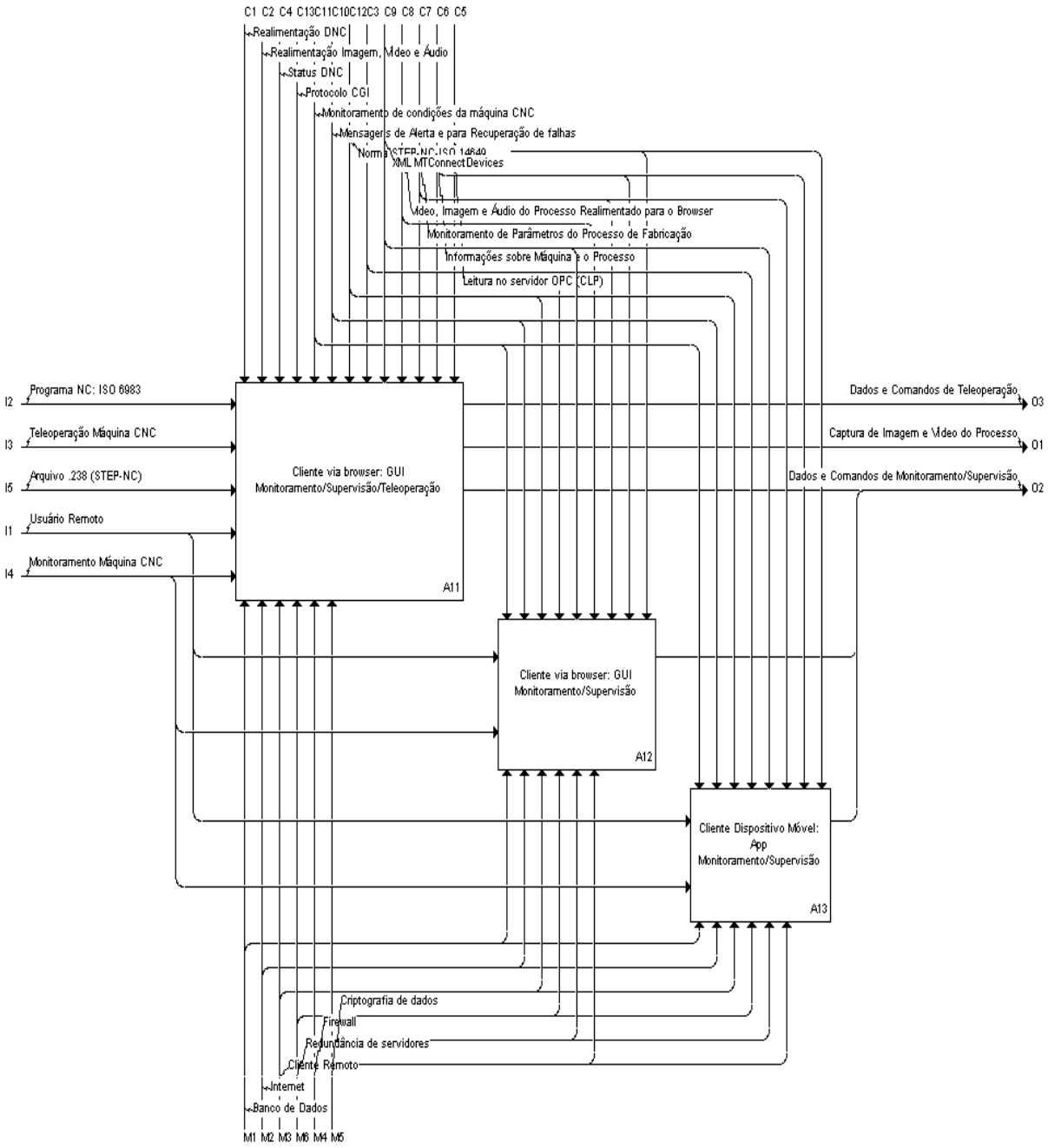

Figura 5.3: Atividades A11, A12 e A13: clientes da arquitetura

formato XML, com a estrutura de dados definida pelo esquema (*schema*) *MTConnectDevices*. O servidor OPCWeb recebe entradas na forma de dados e comandos para requisição de dados de saída com o *status* recente de parâmetros de CLP (Controlador Lógico Programável) ou para alterar esses parâmetros e fazer o acionamento de controles da máquina-ferramenta CNC.

Essa é uma arquitetura geral de referência, logo, os mecanismos "banco de dados"e "computação na nuvem", para o servidor de monitoramento e supervisão, representam apenas propostas que não fazem parte dos elementos implementados. A visão de incluir esses recursos está relacionada ao fato de os projetos de plataformas dentro do paradigma da Indústria 4.0 terem como estratégico uma infraestrutura baseada na

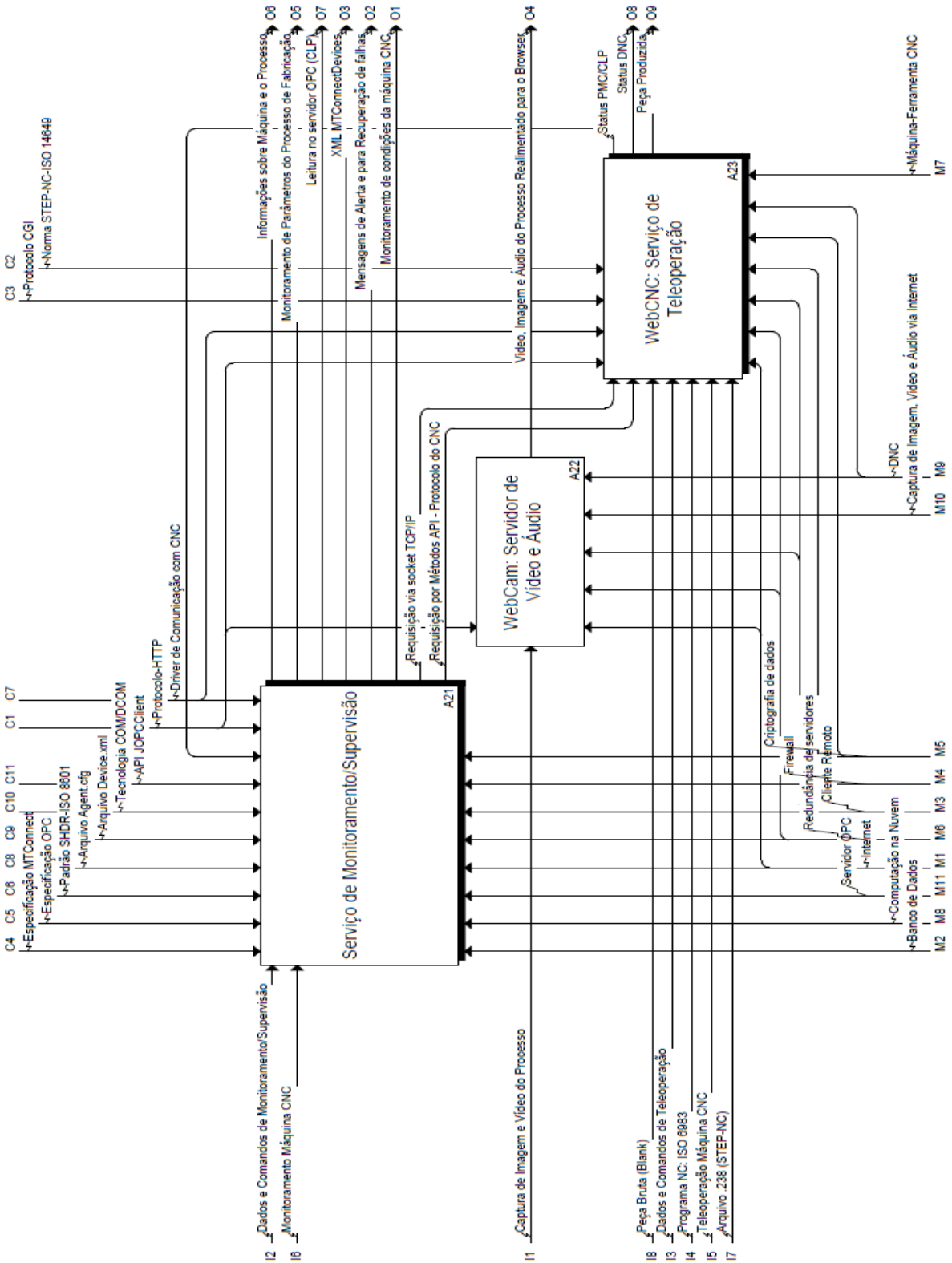

Figura 5.4: Atividades A21, A22 e A23: servidores da arquitetura

nuvem e em bases de dados robustas que auxiliam na produção dinâmica de conhecimento sobre o ambiente de manufatura.

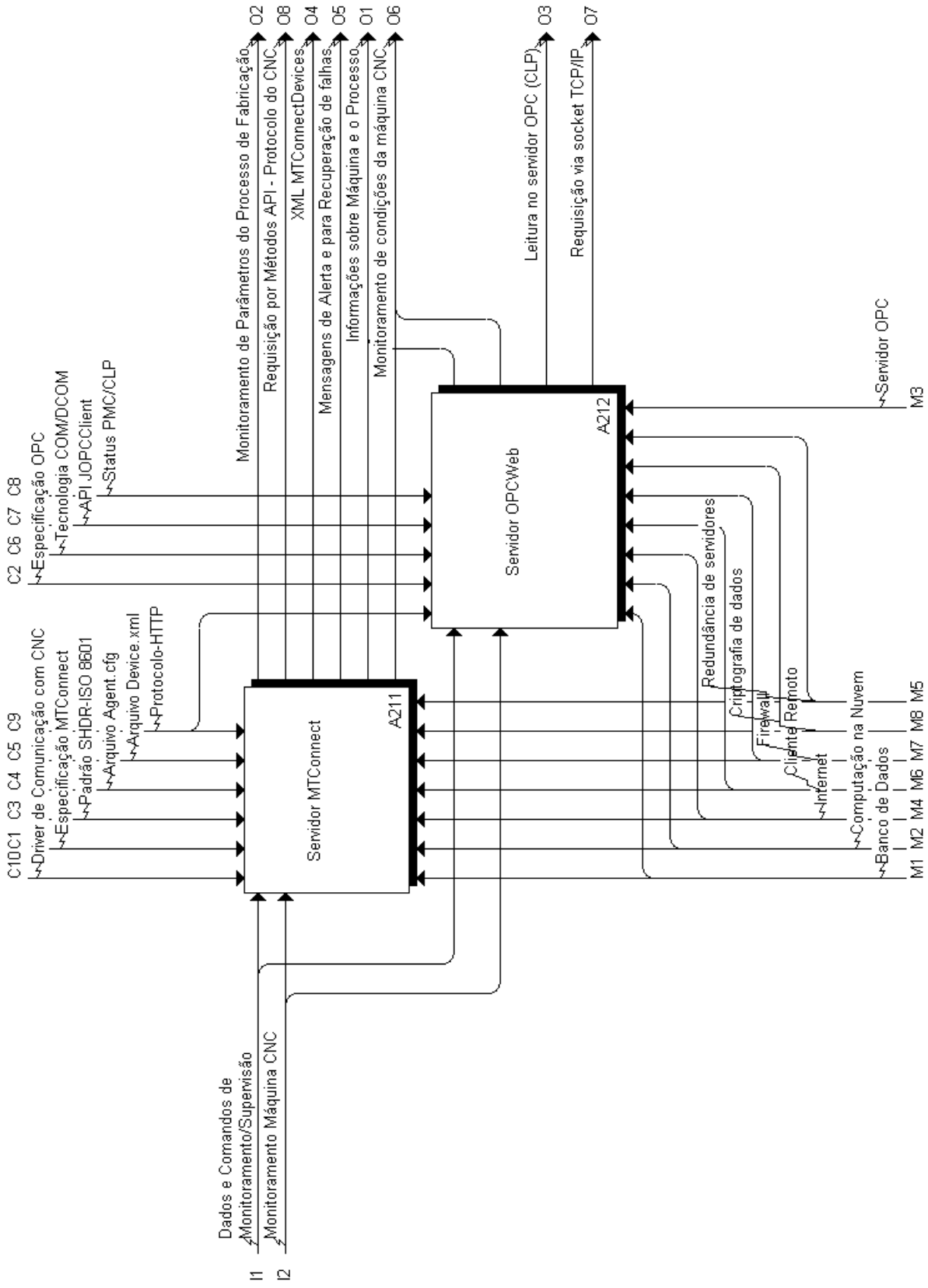

Figura 5.5: Atividades A211 e 212: Servidor de Monitoramento/Supervisão

Atualmente, na arquitetura de um serviço MTConnect é esperável a presença de um software adaptador do CNC da máquina-ferramenta, pois a maioria do CNCs disponíveis comercialmente ainda não possuem suporte nativo para MTConnect, e de um software *agente*. Dessa forma, o servidor MTConnect proposto aqui é dividido em *adaptador*, atividade A2111, e Agente, atividade A2112. O Adaptador comunica-se com o controlador através da API baseada no protocolo proprietário do fabricante do CNC. A resposta à requisição do *adaptador* traz dados que são convertidos em parâmetros e valores em formato de barras (protocolo SHDR- *Simple Hierarchical Data Representation*) (EDSTROM, 2013). Os dados representados nesse formato são compreendidos pelo *agente*, que os estruturam na representação XML dentro de um esquema (*schema*) definido pela especificação MTConnect. O Agente então disponibiliza os dados para requisição da aplicação cliente. A Figura 5.6 apresenta as atividades do servidor MTConnect.

A especificação OPC-UA (OPC *Unified Archtecture*) também é apontada como um dos protocolos que irão ajudar a construir o caminho para o pleno desenvolvimento da Indústria 4.0 (ALBERT, 2015). No entanto, ainda é uma especificação em desenvolvimento, com um número de referências reduzido em termos de projetos acadêmicos e com soluções comerciais ainda nas primeiras versões e dependente de licenciamento para uso, como: MatrikonOPC UA Suite (https://www.matrikonopc.com/opc-ua/index.aspx), Prosys OPC-UA (*Modbus Server*, *Historian* e *Gateway*) (https://www.prosysopc.com/products/), United Automation (*UaExpert* e *UaGateway*) (https://www.unified-automation.com/products.html), Kepware OPC UA Client driver (https://www.kepware.com/en-us/products/kepserverex/drivers/opc-ua-client/), etc. Com isso, em um esforço para superar essas dificuldades iniciais inerentes a implementação do padrão OPC-UA, nessa arquitetura optou-se pelo uso do conceito do OPC-UA, que é baseado em Arquitetura Orientada a Serviço (SOA - *Service-Oriented Archtecture*), para adaptar um software servidor OPC-DA (OPC *Data Access*) clássico para ser manipulado via internet por meio de um software *middleware* com a função de *gateway* entre o servidor OPC e o cliente final localizado na Web. Para a implementação desse *middleware* são avaliadas duas possibilidades, um *Web Service* do tipo RESTful escrito em Java e programas escritos em linguagem Python para interação com a máquina-ferramenta via internet utilizando mecanismo CGI (*Common Gateway Interface*).

Nessa proposta de arquitetura o serviço de supervisão OPC recebeu o nome de OPCWeb , e suas atividades estão representadas na Figura 5.7.

Por fim, a Figura 5.8 apresenta as atividades que correspondem ao servidor WebCNC (A23), que compõe parte dos serviços de teleoperação. Como já mencionado, o WebCNC está diretamente relacionado ao CNC da máquina-ferramenta. Suas atividades envolvem o próprio controlador da máquina como um servidor (A232) e o servidor WebDNC que concentra programas para execução de comandos DNC. Esses servidores têm como principais produtos as respostas das funções DNC executadas no CNC.

5.3 Projeto Axiomático na Concepção de um Sistema para Monitoramento e Teleoperação de Máquinas-ferramenta CNC através da Internet

A definição do modelo funcional do *framework* permitiu identificar os fluxos de informações entre as funções e atividades do framework, pela identificação dos parâmetros de entrada, das saídas, dos mecanismos (recursos) e dos controles (regras). Essas informações, juntamente com levantamento bibliográfico

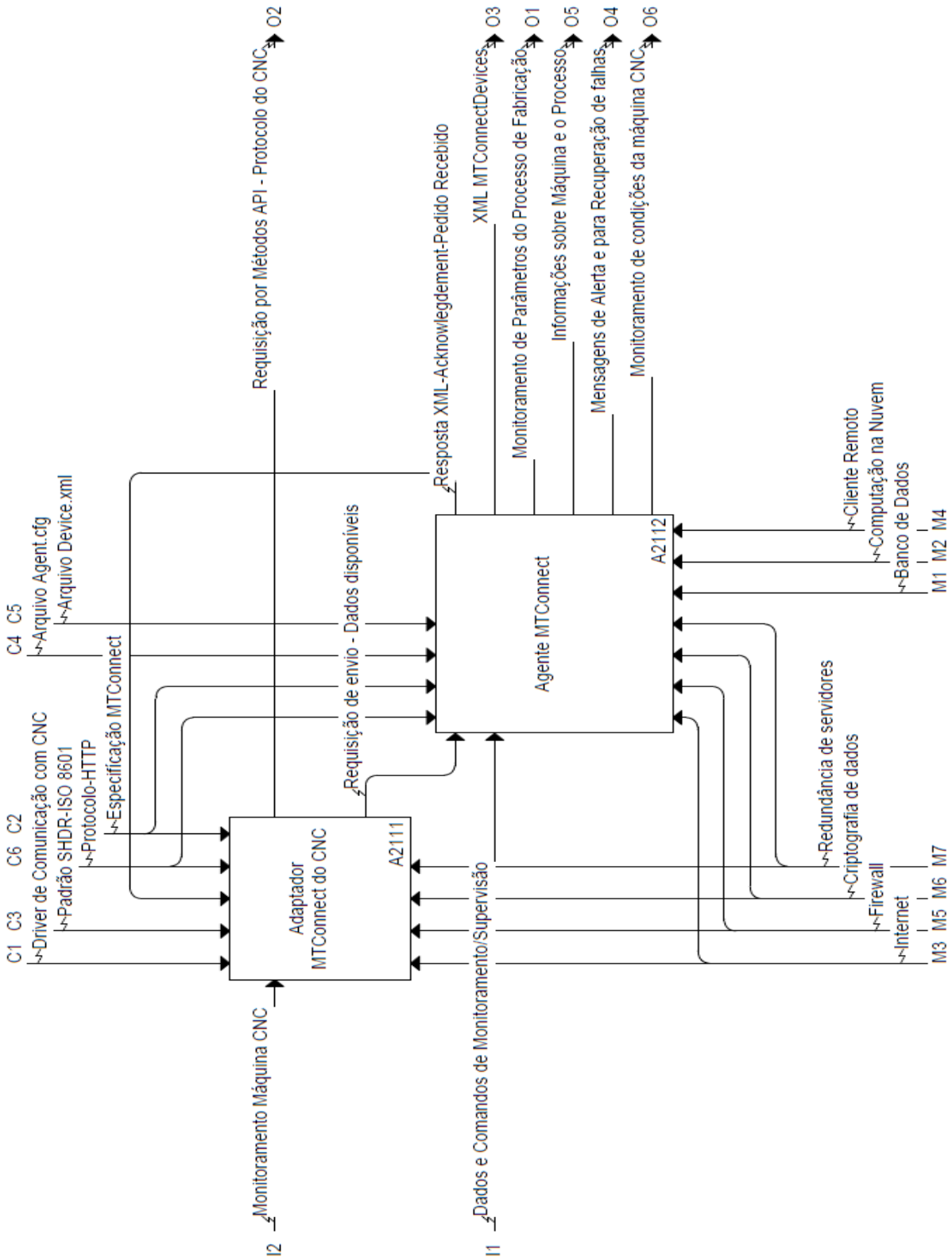

Figura 5.6: Atividades A2111 e A2112: Servidor MTConnect

acerca das tendências para a manufatura dos próximos anos, ajudam a produzir as principais necessidades para os potenciais usuários e as restrições dos sistemas projetados a partir dessa arquitetura. As necessida-

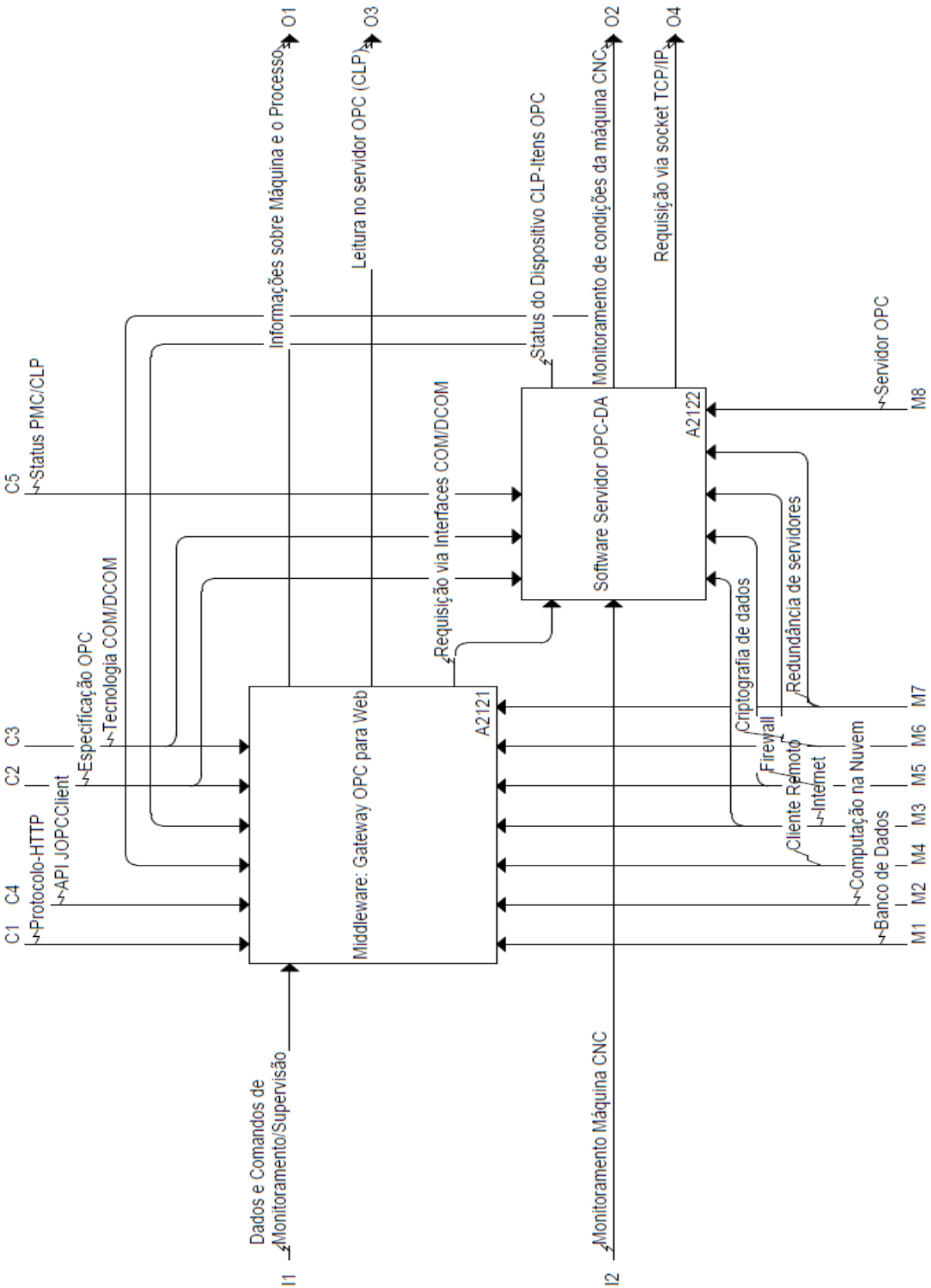

Figura 5.7: Atividades A2121 e A2122: Servidor OPCWeb

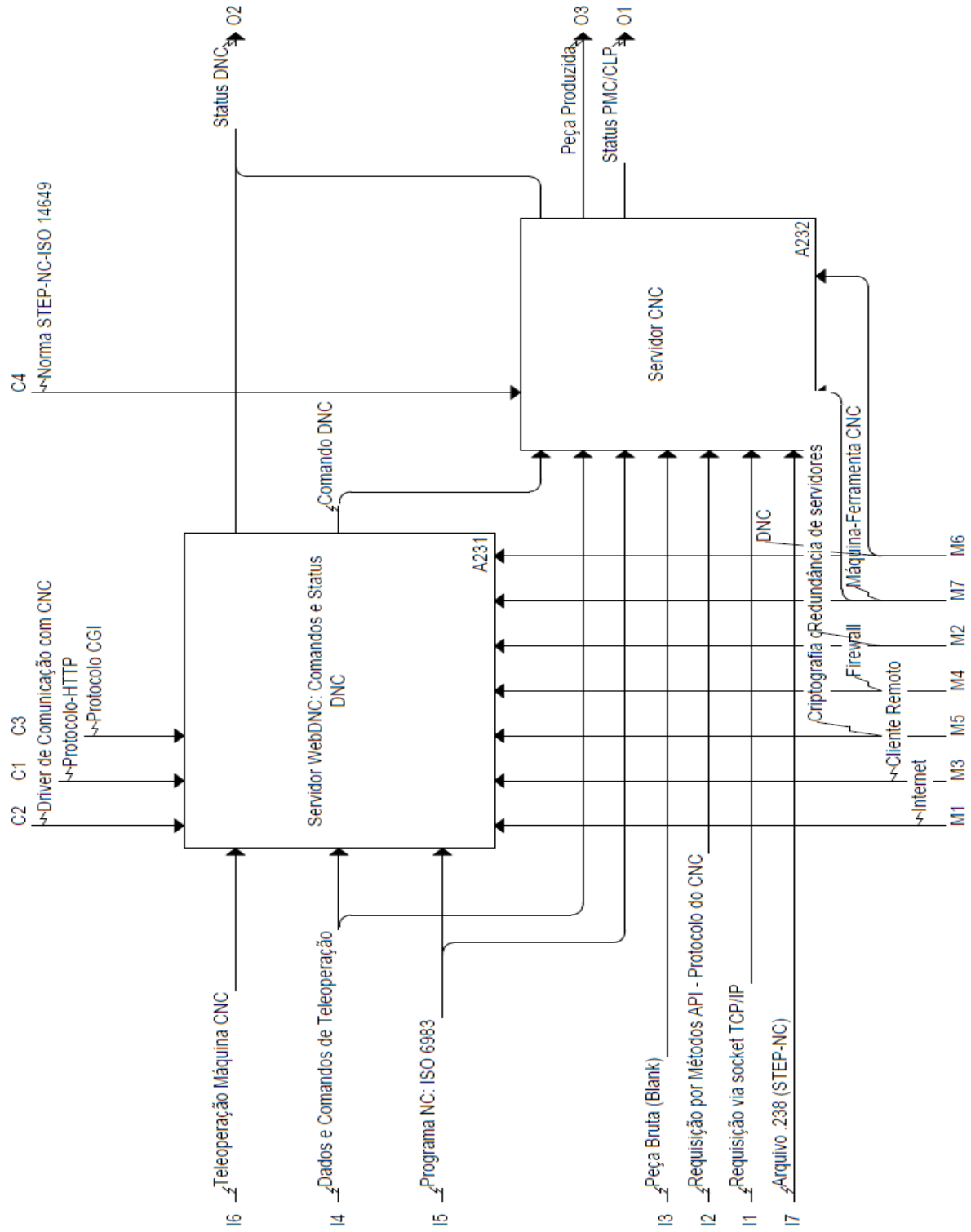

Figura 5.8: Atividades A231 e A232: Servidor WebCNC

des dos usuários do sistema são os insumos básicos para implementação do projeto axiomático.

O projeto axiomático auxilia na definição dos domínios do projeto para um maior detalhamento do mesmo, em um esforço que visa à determinação das especificações do sistema para a sua implementação.

Essa metodologia também fornece por meio de um diagrama de fluxo resultante a sequência de execução do projeto. Durante o processo de projeto são formados módulos, e grupos desses módulos correspondem aos componentes da arquitetura de detalhada, que neste trabalho representam os diferentes servidores e o cliente do sistema.

Os módulos de software que não são apenas herdados e/ou incorporados à arquitetura sistema, mas que se constatou a necessidade de serem programados em linguagens orientadas a objeto, são tratados paralelamente através do modelo V e abordagem ADo-oSS (*Axiomatic Design of Object-Oriented Software Systems*), que gera como resultado a estrutura de classes do software, formada pelo conjunto de módulos. Aqueles mapeamentos relacionados a módulos de software que não foram desenvolvidos, mas incorporados à arquitetura do sistema são tratados de forma semelhante a componentes de um sistema de hardware que são desenvolvidos com base em um projeto axiomático tradicional.

5.3.1 Definição de FRs e Mapeamento entre domínios

A primeira etapa da implementação da metodologia envolve a definição das necessidades dos usuários do produto do projeto no domínio do cliente. Este domínio é formado principalmente por empresas de manufatura e empresas fornecedoras de máquinas-ferramenta CNC envolvidas no desenvolvimento das fábricas inteligentes, e o meio acadêmico especializado.

As necessidades dos usuários foram mapeadas considerando duas fontes. A primeira está relacionada ao fluxo de informações definido na construção do diagrama IDEF0 da seção anterior, que dá uma visão geral de quais são as insumos (*inputs*) que o sistema requer para sua operação, como: usuário remoto, teleoperação da máquina CNC e monitoramento da máquina CNC; as saídas esperadas (*outputs*): monitoramento da máquina e do processo, leitura dos parâmetros de CLP (PMC), informações sobre a máquina e o processo, *streaming* do esquema *XML MTConnectDevices*, etc; principais recursos (mecanismos): internet, cliente remoto, servidor OPC, máquina-ferramenta CNC, etc, e; controles (regras): realimentação imagem/vídeo/áudio, *driver* de comunicação com CNC, protocolo HTTP, protocolo CGI, etc.

As informações provenientes do IDEF0 já fornecem subsídios para pular a fase de levantamento das necessidades dos usuários diretamente para a fase de definição dos requisitos funcionais (FRs) de projeto. No entanto, a fim de aprimorar o processo de projeto axiomático optou-se por realizar o levantamento das necessidades dos usuários considerando uma segunda fonte de dados.

A segunda fonte de informações para o mapeamento das necessidades de usuários para o projeto axiomático envolveu a análise da bibliografia relacionada a projeto axiomático de software (KIM S.-J.AND SUH; S.-G., 1991; SUH; DO, 2000a; SUH; DO, 2000b; SUH, 2001; HELANDER; JIAO, 2002; DO, 2004; JAMSHIDNEZHAD, 2004; PIMENTEL; STADZISZ, 2006; CLAPIS P.J.AND HINTERSTEINER, 2000; LI; TATE, 2011; TOGAY; CANIAZ; DOGRU, 2012), manufatura eletrônica (ALVARES et al., 2002; LEE, 2003; ALVARES, 2005; BENAVENTE, 2007; ALVARES; FERREIRA, 2008; BENAVENTE, 2011) e as aplicações para a manufatura do futuro e fábricas inteligentes (SHI et al., 2011; KAGERMANN; WAHLSTER; HELBIG, 2013; LEE et al., 2013; MACDOUGALL, 2014; LASI et al., 2014; FALLER; FELDMÜLLER, 2015; PISCHING et al., 2015; MONOSTORI, 2015; LEE; BAGHERI; KAO, 2015; FDC, 2016). Essa primeira etapa resultou na Tabela 5.1.

A primeira etapa da implementação da metodologia envolve a definição de requisitos de potenciais clientes (necessidades de clientes, CNs) no domínio do cliente. Este domínio é formado principalmente por empresas de manufatura e empresas fornecedoras de máquinas-ferramenta CNC envolvidas no desenvolvimento das Fábricas Inteligentes, e o meio acadêmico especializado. Essas necessidades foram mapeadas considerando os desafios tecnológicos dessas organizações para o desenvolvimento de produtos no âmbito da Indústria 4.0. Além disso, também foram utilizados como critérios de seleção das principais necessidades dos clientes, alguns dos conceitos associados a um Sistema Ciber-físico com Internet das Coisas, tais como: adoção da normalização e de arquiteturas de referência; Integração vertical e sistemas de manufatura em rede e; transparência de ponta-a-ponta.

Tabela 5.1: Necessidades dos Clientes

Necessidades dos Clientes do Projeto	Descrição
CN1	Visualizar o estado atual do processo
CN2	Visualizar o estado atual da máquina
CN3	Alertar quando da existência de problema no processo
CN4	Manipular dados do processo em um formato leve e universal
CN5	Possuir mecanismos segurança digital e recuperação de falhas
CN6	Dar uma visão do chão-de-fábrica em tempo real
CN7	Intervir no processo de manufatura remotamente
CN8	Interface amigável com o operador remoto
CN9	Receber informações sobre a ferramenta
CN10	Fornecer relatórios sobre o processo

O variado número de atividades e fluxo de informações representados nos diagramas IDEF0 refletem a complexidade do tipo de projeto em análise, uma vez que trata-se da arquitetura de um *framework* na forma de um sistema Cliente-Servidor que integra monitoramento e teleoperação via Internet, ou seja, envolve mais de uma camada de software. Destacando que o projeto proposto é um sistema de software e que deve ter elementos em conformidade com a Indústria 4.0.

Juntamente com o estabelecimento das necessidades dos usuários do sistema (clientes), são definidas as restrições de projeto (*Constraints*, *Cs*) que representam atributos que o projeto essencialmente deve ter, antes mesmo dos requisitos funcionais de primeiro nível do projeto serem abstraídos. As restrições identificadas nessa fase preliminar do projeto estão organizadas na Tabela 5.2.

A restrição **Cs1** refere-se ao requisito básico de um sistema para a Indústria 4.0, que é estar disponível e acessível através da internet, seguindo o paradigma de arquitetura orientada a serviço (SOA). Sistemas baseados nuvem serão de fundamental importância na quarta revolução industrial, mas requerem uma infraestrutura de internet robusta, com uma velocidade de troca de pacotes sempre crescente, que permita o desenvolvimento de aplicações que manipulem um grande volume de dados.

Tabela 5.2: Restrições do Projeto do *Framework*

Restrição	Descrição
Cs1	O *framework* é baseado na Internet, como um serviço
Cs2	Adotar padrões-chave para a Quarta Revolução Industrial
Cs3	Monitoramento e comando virtual
Cs4	Sistema Independente de plataforma

A adoção de padrões-chave **Cs2** é um dos principais fatores para caracterizar uma aplicação no âmbito da Indústria 4.0. Muitos desses padrões são vinculados a Tecnologia da Informação, com destaque para protocolos de troca de dados. Pela restrição **Cs3** deve ficar claro que um sistema para monitoramento e teleoperação através da Internet de uma célula de manufatura deve ter suas funções virtualmente acionadas por um usuário remoto.

O sistema ser independente de plataforma **Cs4** pode ser considerado como uma consequência da restrição **Cs1**, pois um sistema Web, por exemplo, é acessível por qualquer sistema operacional que possua um software navegador instalado. Outro significado da restrição **Cs4** é o fato desses sistemas poderem ser desenvolvidos e executados independente de plataforma, como é o caso dos sistemas para Web.

Estabelecidas as Restrições do Projeto, iniciou-se a definição dos Requisitos Funcionais do projeto (FR) e o mapeamento entre domínios. As Necessidades dos Clientes (CN) tem influência direta na geração dos FRs dos primeiros níveis da hierarquia de projeto.

O projeto tem como requisito funcional fundamental (FR0) a proposta de uma ferramenta para monitoramento e operação remota de máquinas-ferramenta CNC, com elementos técnicos e conceituais aderentes a Indústria 4.0. O Parâmetro de Projeto (DP0) correspondente, considerando o requisito **FR0** e as necessidades dos clientes (CNs), é "um *framework* baseado na Web para monitoramento e teleoperação de máquinas-ferramenta CNC". Movendo-se para o próximo nível na hierarquia de projeto, o parâmetro de projeto principal (DP0) é refinado nos requisitos funcionais definidos na Tabela 5.3, que também apresenta os mapeamentos desses requisitos em Parâmetros de Projeto de nível DPx. Esse mapeamento entre os domínios funcional e físico de primeiro nível resultou na matriz da Figura 5.9.

FR \ DP	DP1	DP2	DP3	DP4
FR1	X			
FR2		X	X	
FR3			X	
FR4	X			X

Figura 5.9: Decomposição para o primeiro nível

Tabela 5.3: Mapeamento de primeiro nível, FRx em DPx

x	FRx	DPx
1	Monitorar o processo de fabricação remotamente	Funções para monitorar o processo através da Web
2	Interatividade com o controlador da máquina	Funções para configuração e comando da máquina via Internet
3	Interface intuitiva com o operador remoto	Interface gráfica estruturada
4	Manter dados sobre o processo de fabricação em banco de dados	Módulo de software vinculado a BD para armazenar dados do processo

Nesse mapeamento de primeiro nível (FRx/DPx) verificou-se um significativo grau de desacoplamento, ou seja, há uma maior independência dos requisitos funcionais (Axioma 1) em relação aos parâmetros de projeto que o realizam. Mas percebe-se que nem todos os **FRs** da matriz estão associados a um único **DP** correspondente, **FR2** possui uma relação com **DP3**, e **FR4** com **DP1**. Durante o processo de abstração para a definição dos parâmetros de projeto concluiu-se haver uma relação entre a interatividade com controlador da máquina (FR2) com a construção de uma interface gráfica estruturada (DP3). Observou-se uma correlação entre manter dados sobre o processo em banco de dados (FR4) e a função de monitorar o processo via Internet (DP1). No entanto, com base em Suh (SUH, 2001), um projeto aceitável, os parâmetros de projeto (DPs) e os requisitos funcionais (FRs) estão relacionados de tal forma que um **DP** específico pode ser ajustado para satisfazer um **FR** sem afetar outros. Com isso, os mapeamentos **FR** x **DP** fora da diagonal poderiam ser suprimidos sem afetar significativamente os mapeamentos posteriores.

O próximo passo do processo de projeto foi a decomposição dos parâmetros de projeto **DPx** em requisitos funcionais de segundo nível (FR1.x, FR2.x, FR3.x e FR4.x). As Tabelas 5.4 a 5.7 apresentam o resultado dessa decomposição e o mapeamento nos parâmetros de projeto de segundo nível (DP1.x, DP2.x, DP3.x e DP4.x). Nas Figuras 5.10 a 5.13 ilustra-se o resultado gráfico desse mapeamento, representado pelas respectivas matrizes de projeto.

O mapeamento do requisito funcional **FR1** representado na Tabela 5.4 representou a decomposição em módulos do que representa os serviços de monitoramento da arquitetura do sistema. Isso inclui os objetos OPC (FR1.1 - FR1.6), dedicados à captura dos sinais dos LEDs dos botões do painel de operações do dispositivo, controlados por CLP. Também estão inclusos nesse mapeamento o servidor para *streaming* de vídeo (WebCam), representado pelo módulo "FR1.25 x DP1.25", e pelos módulos (DM1.7-DM1.24) que formam os métodos da classe que faz a interface com o CNC da máquina-ferramenta por meio da API e *driver* de comunicação com o controlador (Focas1/Ethernet). Esses módulos integrarão a principal classe do software Adaptador do servidor MTConnect, que é um software orientado a objeto. A matriz da Figura 5.10 apresentou um conjunto totalmente desacoplado, revelando um sistema com módulos independentes e, até a presente etapa, sendo considerado um bom projeto.

A Tabela 5.5 apresenta o mapeamento de segundo nível do requisito funcional **FR2**. Este requisito de-

Tabela 5.4: Mapeamento de segundo nível FR1.x

x	FR1.x	DP1.x
1	Adquirir Status da máquina (Power On/Off)	Objeto OPC para fornecer o status de acionamento dos motores da máquina
2	Adquirir Status da Porta da Máquina (Open/Closed)	Objeto OPC para captura do status da porta da máquina (*Closed-Locked*)
3	Adquirir Status da Refrigeração (On/Off/Auto)	Objetos OPC para captura do status do sistema de refrigeração
4	Adquirir status atual dos modos de operação do controlador (AUTO / EDIT / MDI /JOG)	Objetos OPC para captura do status atual dos Modos de Operaçao dos Controlador (On/Off)
5	Adquirir status das opções de execução do programa NC (SINGLE BLOCK / BLOCK DEL/ PROG TEST/ DRY RUN)	Objetos OPC para captura das opções de execução do programa NC (On/Off)
6	Monitorar Alarmes ativos (MC/NC Alarm)	Objeto OPC para monitorar a condição dos Alarmes de máquina e programa NC
7	Capturar o Modo de Controle do CNC	Método getControllerMode
8	Capturar status de disponiblidade do CNC para comunicação	*Available* para uma conexão válida com o CNC de máquina
9	Capturar o status de execução do controlador	Método getExecStatus
10	Capturar o estado atual do atuador de parada de emergência (Emergency Stop)	Método getEmgStop
11	Capturar informações sobre a ferramenta	Método getToolID
12	Capturar número de peças produzidas	Método getCounts
13	Capturar Informações sobre o programa NC em execução	Métodos para capturar Nome do Programa, bloco e linha em execução
14	Capturar mensagens do CNC	Método getMessages
15	Capturar Infomações sobre condições da máquina e do processo	Metodo getCondition
16	Capturar o Avanço atual da ferramenta	Método getActualFeedRate
17	Capturar o Avanço programado da ferramenta	Método getCommandedFeedRate
18	Capturar a Posição atual dos eixos	Método getActualAxesPos [Axes X/Z/C]
19	Capturar a Posição comandada dos eixos	Método getCommandAxesPos [Axes X/Z/C]
20	Capturar a Carga dos Eixos	Método getAxesLoad [Axes X/Z/C]
21	Capturar a Posição da Ferramenta	Posições atuais concatenadas [Xact/ Zact/Cact]
22	Capturar o modo do eixo rotacional	Método getRotaryMode
23	Capturar a Velocidade do eixo rotacional	Método getSpindleSpeed
24	Capturar a Carga dos Eixos Rotacionais	Método getSpindleLoad
25	Capturar os eixos contralados pelo controlador	Comando para concatenar eixos ativos em uma sequência de caracteres
26	Capturar imagens e aúdio do chão-de-fábrica	Servidor WebCAM (Graco/UnB)

clara a necessidade de interatividade com o controlador da máquina, e tem como resposta a possibilidade do desenvolvimento funções que permitam a configuração e o comando da máquina através de um sistema

DP / FR	DP1.1	DP1.2	DP1.3	DP1.4	DP1.5	DP1.6	DP1.7	DP1.8	DP1.9	DP1.10	DP1.11	DP1.12	DP1.13	DP1.14	DP1.15	DP1.16	DP1.17	DP1.18	DP1.19	DP1.20	DP1.21	DP1.22	DP1.23	DP1.24	DP1.25	DP1.26
FR1.1	X																									
FR1.2		X																								
FR1.3			X																							
FR1.4				X																						
FR1.5					X																					
FR1.6						X																				
FR1.7							X																			
FR1.8								X																		
FR1.9									X																	
FR1.10										X																
FR1.11											X															
FR1.12												X														
FR1.13													X													
FR1.14														X												
FR1.15															X											
FR1.16																X										
FR1.17																	X									
FR1.18																		X								
FR1.19																			X							
FR1.20																				X						
FR1.21																					X					
FR1.22																						X				
FR1.23																							X			
FR1.24																								X		
FR1.25																									X	
FR1.26																										X

Figura 5.10: Matriz de decomposição para DP1

para Internet (Web). Isso resultou em um mapeamento para um conjunto de requisitos funcionais relacionados a implementação de funções de comando e interação com máquina-ferramenta CNC. Com isso, foram estabelecidos requisitos a serem atendidos através da comunicação com o CLP da máquina, tornando adequada a implementação de recursos de escrita para um servidor OPC manipulável via Internet, representado pelos Parâmetros de Projeto do intervalo [DP2.1 -DP2.6].

Os requisitos funcionais [FR2.7-FR2.11] da Tabela 5.5 correspondem as funções de interação com o controlador que são realizadas por meio do uso do *driver* e protocolo do fabricante do CNC (Focas1/DNC1), combinado com um mecanismo de acesso via Web, como CGI (*Common Gateway Interface*). Esses programas executam funções de DNC (*Direct Numerical Command*) para o envio de dados e recebimento de resposta do CNC da máquina.

A matriz gerada após a decomposição de **DP2** (Fig.5.11) mostra uma significativa independência entre módulos nessa parte do sistema, apresentando um bom grau de desacoplamento. A possível sequencia de execução obtida com a definição do parâmetro de projeto **DP2.4** pode ser suprimida sem influência significativa no desacoplamento dessa parte da arquitetura do sistema.

O requisito funcional **FR3** trata de uma interface gráfica de usuário (GUI) com o sistema. Os requisitos funcionais derivados da decomposição do parâmetro de projeto **DP3**: "Interface gráfica estruturada"estão

Tabela 5.5: Mapeamento de Segundo Nível FR2.x

x	FR2.x	DP2.x
1	Iniciar a execução do programa (Cycle Start)	Objeto OPC para iniciar a execução de um programa NC
2	Parar execução do programa (Cycle Stop/RESET)	Objeto OPC para interromper a execução de um programa NC
3	Ligar/Desligar Refrigeração (Coolant ON/OFF)	Objeto OPC para ligar/desligar refrigeração da máquina
4	Definir o modo de operação do controlador	Objetos OPC para Ativar/Desativar os modos de operação do controlador (AUTO / EDIT / MDI /JOG)
5	Ativar/Desativar opções de execução dos programas	Objeto OPC para Ativar/Desativar os opções execução dos programas (SINGLE BLOCK / BLOCK DEL/ PROG TEST/ DRY RUN)
6	Movimentação de eixos manualmente	Objeto OPC para manipular as funções das teclas direcionais dos eixos (+X/-X /+Z /-Z)
7	Buscar um programa especificado no CNC	Função da API do CNC para *Upload* de programa NC (receber)
8	Deletar um programa especificado	Função da API do CNC para excluir programa NC da memória
9	Listar programas NC do CNC	Função da API do CNC para transmitir lista de Programas NC gravados
10	Transmitir linhas de comando para o CNC	Função da API do CNC para executar linha de programa manualmente
11	Transmitir um programa NC para o controlador da máquina	Função da API do CNC para Download de programa NC (enviar)

em grande parte relacionados a usabilidade do sistema, conforme demonstra a Tabela 5.6. Essa relação entre os requisitos **FR3.x** reflete no resultado gráfico da matriz de projeto dessa decomposição. Através da Figura 5.12 identifica-se módulos altamente acoplado com muitos relacionamentos fora da diagonal principal. Esse nível de acoplamento revela também a significativa complexidade inerente à elaboração da interface com usuário na tarefa de integrar um conjunto de serviços para troca de dados em um mesmo ecrã, como parte de uma arquitetura cliente/servidor, e com funções correlacionadas.

O parâmetro de projeto **DP4** corresponde à entrada do módulo para um sistema de registro dos dados de operação da máquina e do processo de fabricação em um sistema de banco de dados para potenciais consultas e análises do processo a fim de favorecer possíveis melhorias e tomadas de decisão. Sua decomposição resultou nos requisitos funcionais **FR4.x**, exibidos na Tabela 5.7. É possível dizer que todo o mapeamento dos requisitos funcionais desse nível corresponde a funções de persistência de parâmetros do CNC em banco de dados, como: Carga dos eixos, Status das condições do processo, mudanças de status de operação da máquina, número de peças produzidas e mensagens de alerta do CNC.

A matriz resultante dessa decomposição (Fig.5.13) corresponde a um serviço com um projeto totalmente desacoplado. Essa informação garante que consultas de registros e análises de dados poderão ser executadas por categorias bem definidas.

FR \ DP	DP2.1	DP2.2	DP2.3	DP2.4	DP2.5	DP2.6	DP2.7	DP2.8	DP2.9	DP2.10	DP2.11
FR2.1	X										
FR2.2		X									
FR2.3			X								
FR2.4				X							
FR2.5					X						
FR2.6				X		X					
FR2.7							X				
FR2.8								X			
FR2.9									X		
FR2.10				X						X	
FR2.11											X

Figura 5.11: Matriz de decomposição para DP2

Tabela 5.6: Mapeamento de Segundo Nível FR3.x

x	FR3.x	DP3.x
1	Visualização do status da máquina	*Feedback* e Informações sobre a máquina em tempo real
2	Clareza de comunicação seguindo as convenções do mundo real	Informação através de palavras e frases objetivas e compreensíveis pelo usuário
3	Liberdade e controle do usuário	Interação e método para livrar-se de erros e problemas com o sistema
4	Uso intuitivo	interface estruturada por categoria de funções (Leitura/escrita)
5	Projeto minimalista	Mostrar apenas informações relevantes
6	Ajudar usuários a reconhecer, diagnosticar e recuperar-se de erros	Mensagens de erros claras que identifiquem o erro precisamente
7	Apresentação apropriada	Exibir somente informações necessárias no momento e lugar certo com o formato adequado
8	Capacidade de aprendizado	Um projeto que auxilia na aprendizagem através de exploração
9	Prevenção de erros	Projeto cuidadoso

Os módulos para um sistema de persistência de dados é definido na arquitetura, porém não entrou na implementação final, ficando apenas como sugestão para inclusão em trabalhos futuros.

DP / FR	DP3.1	DP3.2	DP3.3	DP3.4	DP3.5	DP3.6	DP3.7	DP3.8	DP3.9
FR3.1	X								
FR3.2	X	X		X	X	X		X	X
FR3.3	X		X			X		X	X
FR3.4				X		X	X	X	X
FR3.5		X		X	X	X	X	X	
FR3.6	X	X	X	X	X	X	X	X	X
FR3.7	X	X	X	X	X	X	X	X	X
FR3.8							X	X	X
FR3.9		X	X	X	X	X	X	X	X

Figura 5.12: Matriz de decomposição para DP3

Tabela 5.7: Mapeamento de Segundo Nível FR4.x

x	FR4.x	DP4.x
1	Manter dados sobre Carga dos Eixos (*load*)	Função para salvar a carga atual dos eixos (X/Z/C) da máquina
2	Manter dados sobre condições de operação do CNC (*condition*)	Função para salvar o status da condição de operação do CNC
3	Manter registros sobre Inicio e Parada da máquina	Função para registrar as mudanças de status de operação da máquina
4	Manter registros sobre número de peças produzidas	Função para registrar número de produzido de peças/dia
5	Manter registros de mensagens do CNC	Função para salvar mensagens de alerta enviadas pelo CNC

5.3.2 Definição da Matriz de Projeto Completa

Após o fim do processo de decomposição, inconsistências devem ser conferidas para confirmar a decomposição. Com isso, optou-se pela eliminação dos elementos a direita da diagonal principal na matriz de decomposição de **DP3** (Figura 5.12), dos módulos da interface gráfica de usuário (GUI), por constatar posteriormente a grande semelhança entre alguns de seus requisitos funcionais. Feito isso, as matrizes parciais no segundo nível de decomposição foram reunidas na matriz de projeto completa da Figura 5.14.

Essa Matriz de Projeto foi rearranjada a fim de aproximar módulos com mais similaridade em termos de tecnologia de implementação e para obter uma sequência de execução mais lógica. O resultado dessa ação é a Matriz de Projeto Completa da Figura 5.15. Após a transposição dos módulos da matriz é possível

DP / FR	DP4.1	DP4.2	DP4.3	DP4.4	DP4.5
FR4.1	X				
FR4.2		X			
FR4.3			X		
FR4.4				X	
FR4.5					X

Figura 5.13: Matriz de decomposição para DP4

agrupar e identificar nesses módulos os serviços que compõem a arquitetura do sistema.

Dessa forma, da Figura 5.15 foram extraídos os seguintes grupos de módulos:

(A) Módulo do servidor de *streaming* de vídeo (WebCam - Graco/UnB), que não foi programado no âmbito desse trabalho, mas foi selecionado para implementação.

(B) : Parte do Servidor de Monitoramento, que emprega o padrão MTConnect. Na matriz é representado pelo grupo de módulos (DM1.7-DM1.25) que formam a principal classe do software Adaptador (*FocasGateway*), que tem a função de fazer a comunicação do Adaptador com o CNC da máquina através da API desenvolvida pelo fabricante CNC. O Adaptador é desenvolvido e implementado nesse trabalho.

(C) : Compõe servidor de OPCWeb e é constituído por módulos de leitura (C1) e escrita (C2) em parâmetros de CLP da máquina-ferramenta.

(C) : Parte do serviço de comando remoto que utiliza protocolo DNC do controlador da máquina para o envio de comandos remotamente ao CNC para a execução de funções específicas na máquina.

(E) : Módulos que caracterizam o Cliente Web ideal.

(F) : Classe com função de *middleware* para o persistência em banco de dados de informações capturadas durante o monitoramento/supervisão do processo.

(G) : Funções para vincular o servidor de *streaming* de vídeo/áudio à interface Web cliente.

(H) : Módulos para a comunicação entre Adaptador do CNC e o Cliente MTConnect (função do Agente MTConnect).

(I) : Módulos que vinculam os serviços de *streaming* de dados MTConnect aos métodos para salvar dados de parâmetros de status da máquina em banco de dados.

(J) Módulos que fazem parte do servidor OPCWeb com função de *middleware* para associar funções de leitura no servidor (CLP da máquina) ao cliente Web.

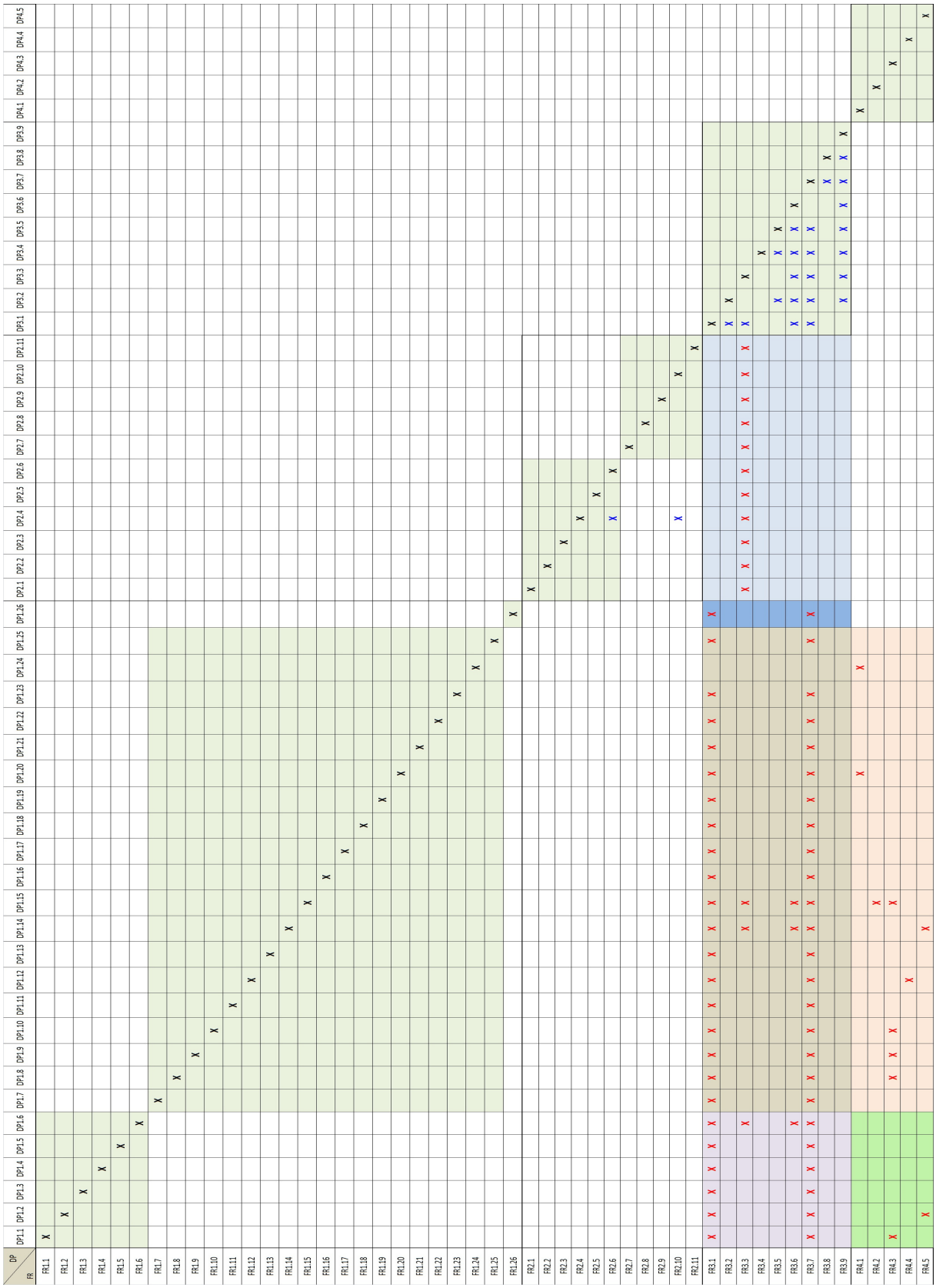

Figura 5.14: Matriz de Projeto Completa

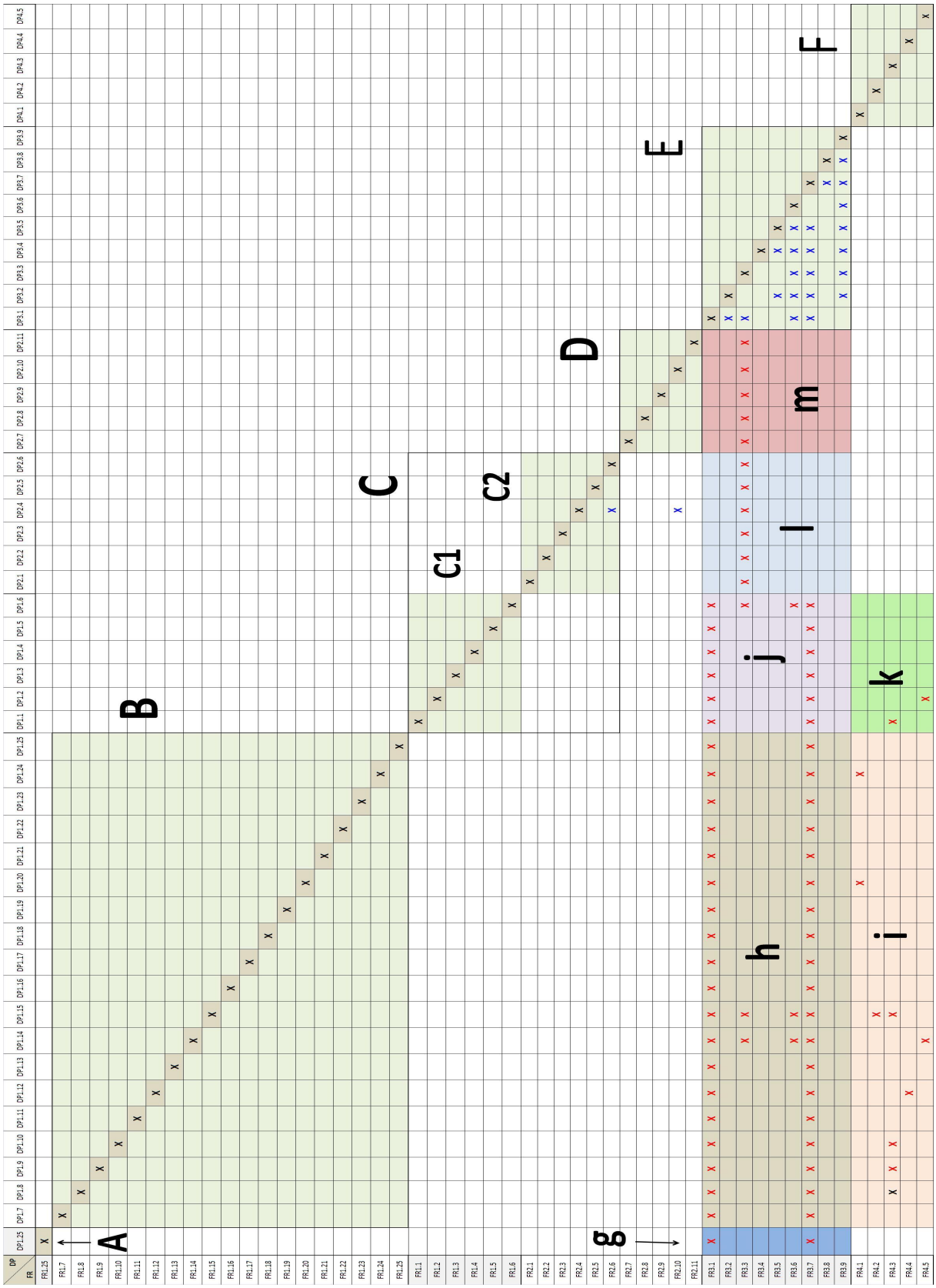

Figura 5.15: Matriz de Projeto Completa rearranjada

(K) Módulos que vinculam os serviços de supervisão OPC aos métodos para salvar o status de parâmetros de CLP da máquina em banco de dados.

(L) Módulos que fazem parte do servidor OPCWeb com função de *middleware* para associar funções de escrita no servidor (CLP da máquina) ao cliente Web.

(M) Módulos representando os programas para o envio de comandos DNC para o controlador da máquina através de mecanismos de acesso da Internet (CGI).

O grupo de módulos na matriz da Figura 5.15 que foram identificados com letras minúsculas porque representam módulos de ligação dos módulos dos servidores com os módulos da aplicação cliente ou aqueles módulos de entidades vinculadas a banco de dados. No que refere a este último caso, classes com função de manipulação e gravação de informações de fabricação em banco de dados não entram na implementação deste projeto, permanecem apenas como proposta na arquitetura do sistema para trabalhos futuros.

Os caracteres alfabéticos (A, B, C,...) utilizados para identificar os conjuntos de módulos na matriz de projeto completa (Figura 5.15) também permitem representar a sequência de execução do projeto (começando em "A"indo até "M") e auxiliam na construção de um dos principais resultados do projeto axiomático que é o diagrama de fluxo (Figura 5.16). Neste diagrama os módulos derivados da decomposição do parâmetro de projeto DP1 são agrupados no bloco **M1** (servidor de monitoramento), aqueles gerados a partir do elemento DP2 são reunidos no bloco **M2** (servidor de teleoperação), esses blocos são implementados independentemente a implementação de módulos precedentes e representam grupos de módulos desacoplados. Por isso, no diagrama da Figura 5.16, são associados por meio de elemento soma (**S**) para projetos desacoplados. Os módulos **M1** e **M2** vão se associar aos blocos semi-acoplados, **M3** (interface web cliente) e **M4** (aplicação para banco de dados de fabricação), através da junção de controle (**C**). O trabalho de desenvolvimento do projeto é considerado finalizado quando de todas as atividades dos blocos **M3** (M3.1-M3.9) e **M4** (M4.1-M4.5) forem implementados, para o caso de todos módulos de **M3** e **M4** entrarem para a arquitetura final de implementação.

Na Figura 5.15 o **C1** representa os objetos para aquisição de dados de parâmetros de CLP da máquina CNC através das *tags* de leitura do servidor OPC-DA (OPC *Data Access*). Os módulos de **C2** são implementados na forma de objetos para a escrita e alteração do *status* nos parâmetros de CLP da máquina através de *tags* de leitura/escrita do servidor OPC-DA. Na sequência, ainda pela implementação do servidor OPCWeb é também desenvolvido o software *middleware* com função de *gateway* ("i"e "l") entre o servidor OPC instalado no mesmo PC e a Internet, para a transmissão de dados e o recebimento de requisições Web. Na implementação desse *gateway* experimenta-se duas abordagens tecnológicas para avaliação, a primeira envolve o desenvolvimento de um *Web Service* RESTful que recebe requisições HTTP e produz respostas com dados provenientes do servidor OPC em formato XML. A segunda abordagem usa *scripts* escritos em linguagem Python associados à API *OpenOPC* que vem acompanhada de um serviço *gateway* que possui a capacidade de acessar servidores OPC de diferentes fornecedores. Os *scritps* em python são executados em um servidor web Apache utilizando mecanismo CGI (*common gateway interface*) em requisições HTTP.

Nesse diagrama de fluxo os elementos que foram mapeados para serem desenvolvidos em linguagens de programação orientadas a objeto (adaptador MTConnect e *Web Service* RESTful para o OPCWeb) são

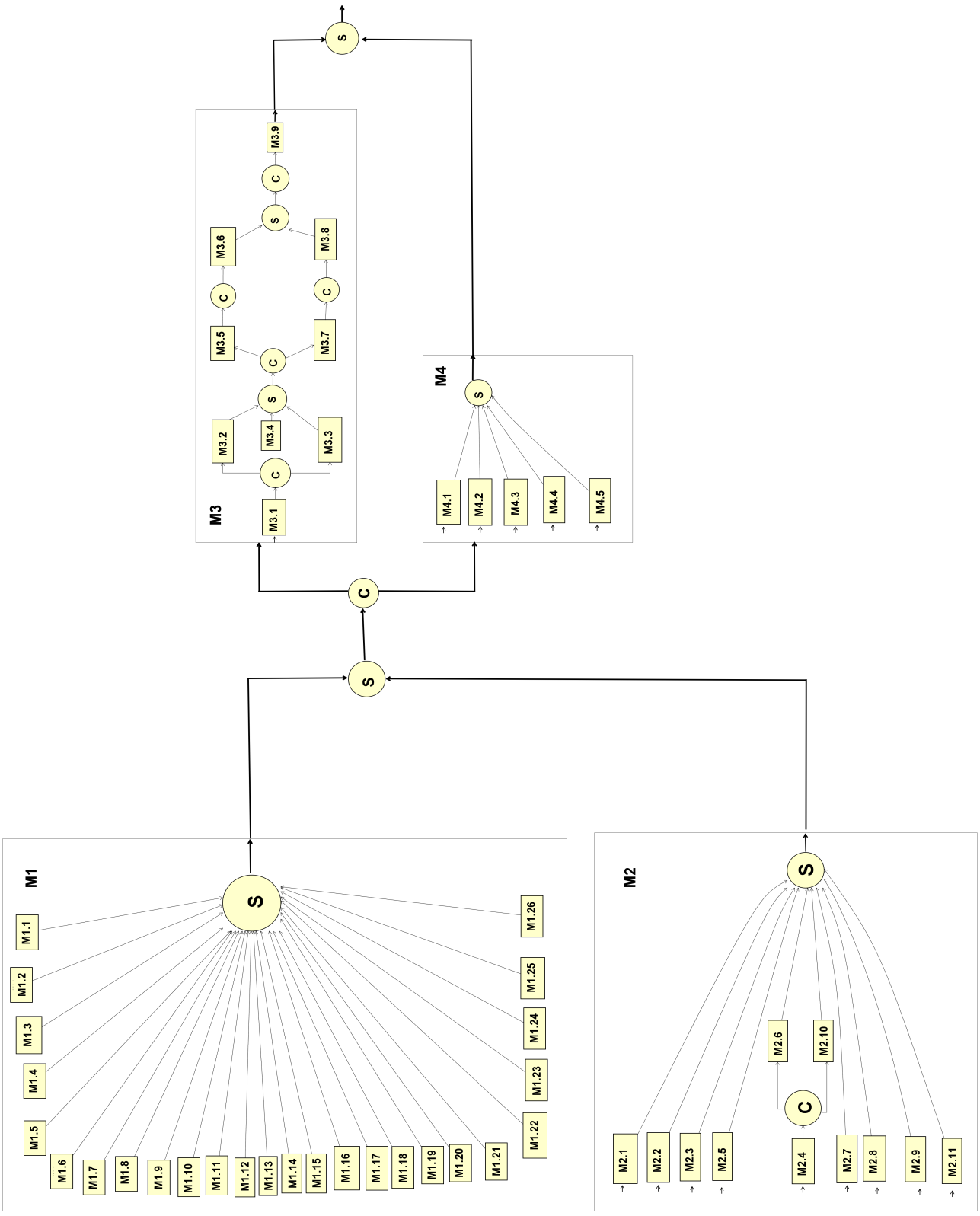

Figura 5.16: Diagrama de fluxo do sistema

tratados através de projeto axiomático para sistemas de software orientados a objeto, que possui como um de seus resultados os diagramas de classe que são ilustrados e descritos na seção 5.4.2.

Com base na matriz de projeto completa e no diagrama de relacionamento, os conjuntos de módulos identificados durante o projeto podem ser representados graficamente conforme a arquitetura detalhada da Figura 5.17, que fornece uma visão mais clara dos elementos da arquitetura proposta.

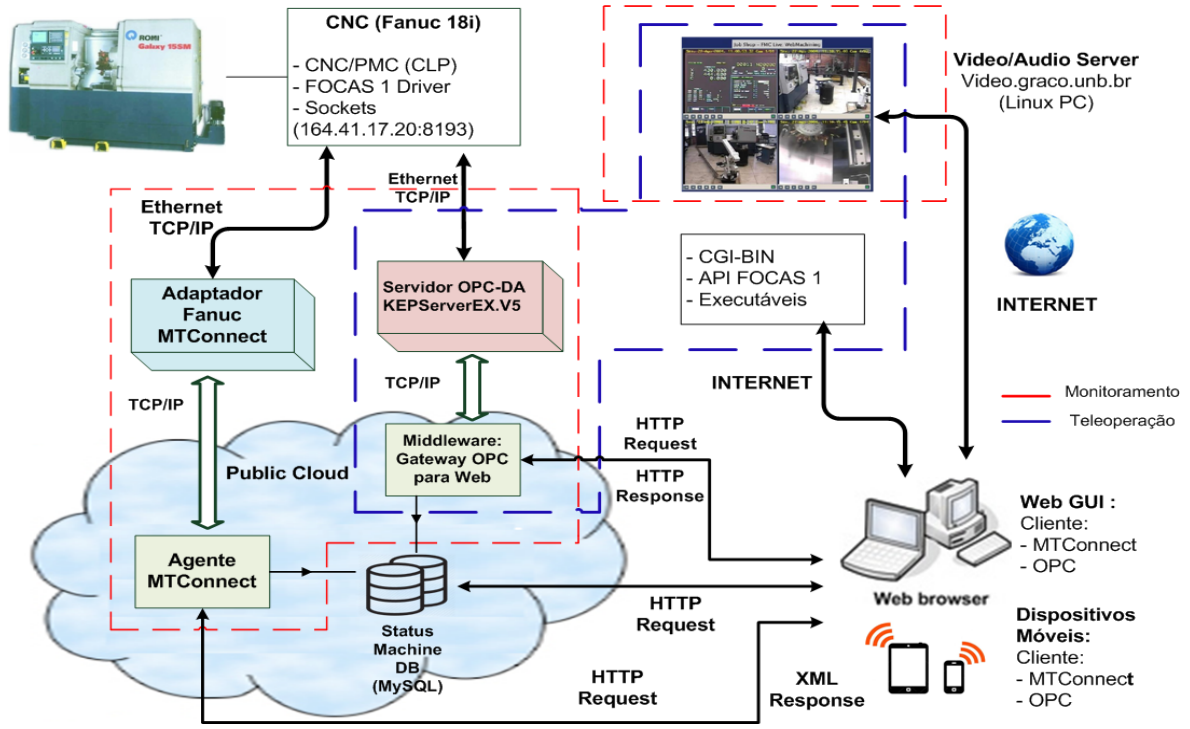

Figura 5.17: Arquitetura proposta para o sistema

Nesta proposta estão presentes os elementos definidos pelos módulos da matriz de projeto completa e as suas comunicações, acrescidos dos mecanismos de computação na nuvem com banco de dados para atividades de supervisão, atributos que associam a proposta da arquitetura ao paradigma de IoT e, consequentemente, a estratégia da Indústria 4.0.

5.4 Modelagem UML do Sistema

Está seção apresenta a modelagem UML do sistema e de seus elementos. Isso inclui a documentação do sistema sob a perspectiva do contexto (diagramas de casos de uso), da análise das atividades (diagrama de atividade) e do desenvolvimento (diagrama de pacotes). Esta última inclui os diagramas de classe dos elementos de software desenvolvidos.

É documentado o resultado da metodologia de projeto axiomático de software representado pelas classes obtidas a partir dos módulos da matriz de projeto completa, que correspondem a aplicações orientadas a objeto.

5.4.1 Diagrama de Pacotes

A modelagem UML permite a representação da arquitetura de um sistema do nível mais geral até o nível mais específica o detalhado. Com isso, aproveitou-se a capacidade desse padrão para representar a estrutura do sistema proposto em um diagrama de pacotes (Figura 5.18) em que os pacotes correspondem aos módulos do sistema. O diagrama de pacotes dá uma visão das dependências entre as partes de compõem o sistema.

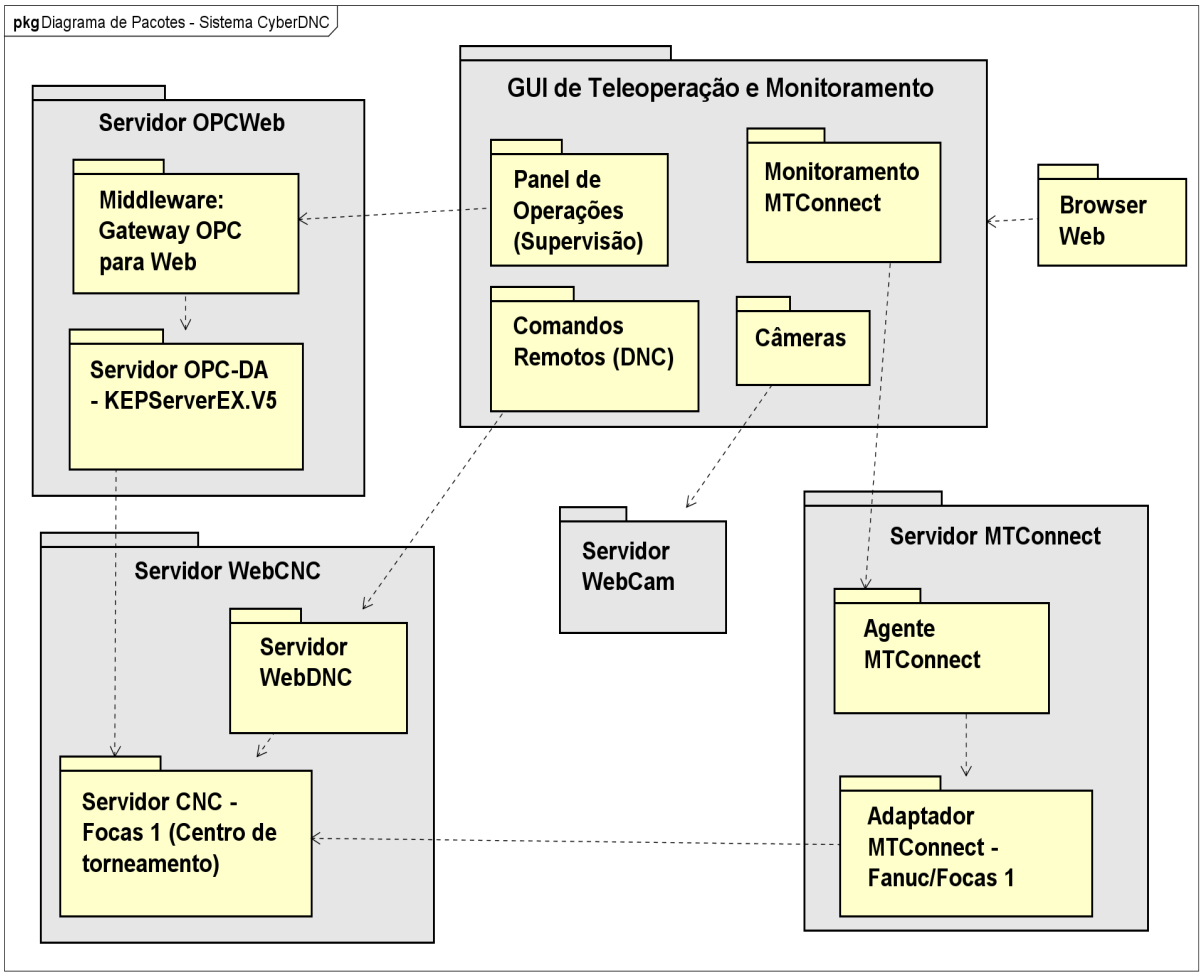

Figura 5.18: Diagrama de pacotes com as dependências entre os módulos do sistema

O diagrama de pacotes não tem como propósito fornecer detalhes da implementação, mas dá uma visão da arquitetura de como e por quais clientes um serviço é consumido, sendo que um serviço também pode ser um cliente ou servidor de outro serviço.

O pacote da GUI de monitoramento e teleoperação é o que possui o maior número de subpacotes: Painel de Operação (Supervisão), Monitoramento MTConnect, Comandos Remotos (DNC) e Câmeras. Esses correspondem aos elementos cliente que consomem dados e recursos dos respectivos servidores: Servidor OPCWeb, Servidor MTConnect, Servidor WebCNC e Servidor de vídeo (Câmeras).

O servidor OPCWeb, que é formado por um software *gateway* e pelo servidor OPC-DA, é um cliente

do servidor CNC-Focas1 que pertence ao pacote WebCNC, pois requer o acesso a dados na forma de sinais do CLP do máquina-ferramenta. O servidor MTConnect também é um cliente do servidor CNC-Focas 1 que fornece dados de parâmetros do CNC através do Adaptador provido com API Focas 1 (*Fwlib32.dll*). O Agente MTConnect usa o Adaptador para receber dados do CNC e é o servidor de dados em formato XML do módulo de monitoramento do cliente Web.

A interface de usuário do sistema é totalmente Web necessitando de um de software *browser* para acesso remoto à aplicação. Nesse mesmo cliente Web são disponibilizadas imagens da FMC (*Flexible Manufacturing Cell*) através do servidor de vídeo, WebCam.

5.4.2 Diagramas de Classe

Essa seção tem a função de fornecer o resultado complementar da aplicação da metodologia de Projeto Axiomática para os módulos de software orientados a objeto, descrita na seção 3.2. Esses módulos são tratados através do modelo V (vê) com a metodologia de projeto axiomático para sistemas software orientados a objeto (ADo-oSS) o Adaptador MTConnect desenvolvido em linguagem C# e o *Web Service* RESTful desenvolvido em Java, que é uma das opções de implementação para o *gateway* do servidor OPCWeb. As linguagens de programação utilizadas no desenvolvimento de cada aplicação foram escolhidas com base em critérios como: eficácia na programação e robustez da linguagem para o tipo especifico de aplicação (*Windows forms* e *Web Service RESTful*) e, a possibilidade de herança de códigos fonte de aplicações similares previamente desenvolvidas.

A Figura 5.19 apresenta o diagrama de classe do Adaptador com pacotes que representam os *namespaces* aos quais pertence cada classe.

O *namespace* "AdapterLab"é o principal da aplicação, porque foi criado junto com a criação do projeto da aplicação na IDE de desenvolvimento Visual Studio 2010. Nesse pacote está a classe padrão associadas inicialização da aplicação (*Program*) e a classe do formulário gerado quando da criação de um projeto do tipo *Windows Forms*, que recebe o nome de *MachineTool*, nome adequado diante do objetivo que se quer alcançar com o desenvolvimento da aplicação. Na classe *MachineTool* estão as referências ao parâmetros da máquina com as variáveis de atributos cujos tipos de dados representam dados de Eventos (*Event*), Amostras de dados (*Sample*) e Mensagens (*Message*), baseado na nomenclatura da especificação MTConnect. Nessa mesma classe foram criados os métodos associados a eventos, como os de iniciar (*start_Click(...)*) e parar (*stop_Click(...)*) a atividade do Adaptador, e um método temporizador (*gather_tick(...)*), que atualiza o valores dos atributos em intervalo pré-estabelecido.

O pacote *Fanuc* desenvolvido no contexto da implementação computacional do sistema contem a classe *FocasGateway* que recebeu esse nome por encapsular os métodos da API Fanuc-Focas 1 em uma representação mais simples, de mais fácil compreensão, em linguagem C# (C-Sharp). Esses tem por função a conexão e desconexão com o CNC do centro de torneamento, e a requisição de informações diversas como o status de execução, nome do programa em execução, ferramenta utilizada, posição atual dos eixos, entre outros. No mesmo *namespace*, a classe *FanucPath* tem apenas a função de fornecer informações relativas as condições de operação da máquina que tecnicamente estão vinculadas aos tipos de alarmes do CNC. Na especificação MTConnect os itens de dados (*DataItem*) da categoria *Condition* estão relacionados aos

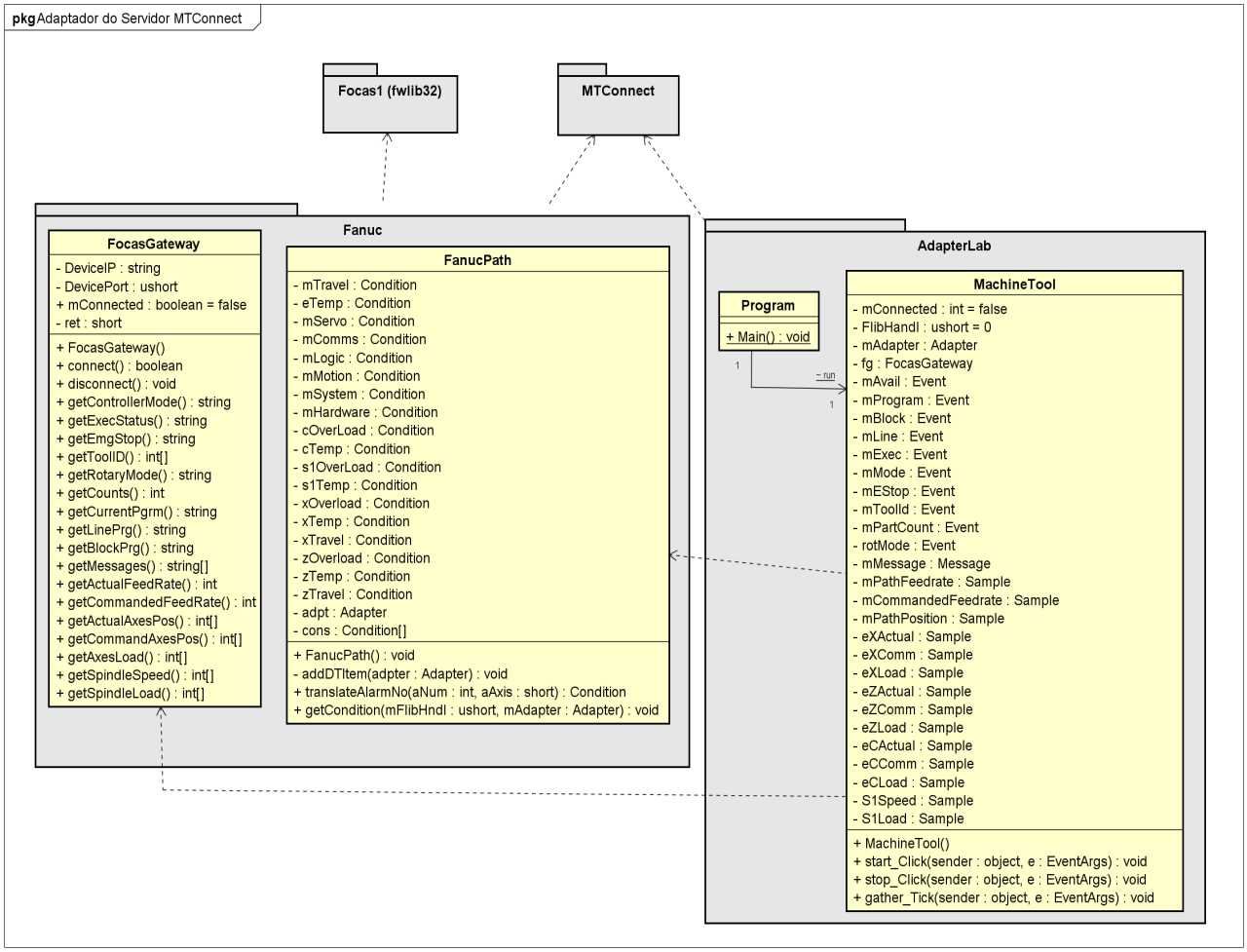

Figura 5.19: Estrutura de Classes do Adaptador Fanuc-Focas1 do servidor MTConnect

alarmes do CNC, diferentemente das primeiras versões da especificação onde havia um tipo especial de item de dados chamado *Alarm*, associado a categoria de item de dados *Event*.

O diagrama de classe da Figura 5.19 mostra ainda que tanto *namespace* AdapterLab, quanto o Fanuc tem uma relação de dependência com o pacote MTConnect (*https://github.com/mtconnect/dot_net_sdk*), um *toolkit* desenvolvido sob licença livre por William Sobel durante *MC2 Conference 2013*, na sessão *MC2 Adapter Lab Project*. William Sobel é um dos pioneiros do padrão e que ajudou a desenvolver toda a especificação MTConnect.

A Figura 5.20 detalha as classes do *namespace* MTConnect que foram empregadas na implementação do Adaptador desenvolvido neste trabalho.

A estrutura e nomenclatura das classes desse pacote são baseadas completamente na especificação MTConnect. A classe *Adapter* funciona como a entidade que manipula a estrutura de dados proveniente do dispositivo CNC, que inclui *DataItem* (Item de Dados) e *Asset* (Ativo). Este trabalho tratará especificamente dos *DataItems*, por representarem as principais informações que um controlador de uma máquina-ferramenta fornece.

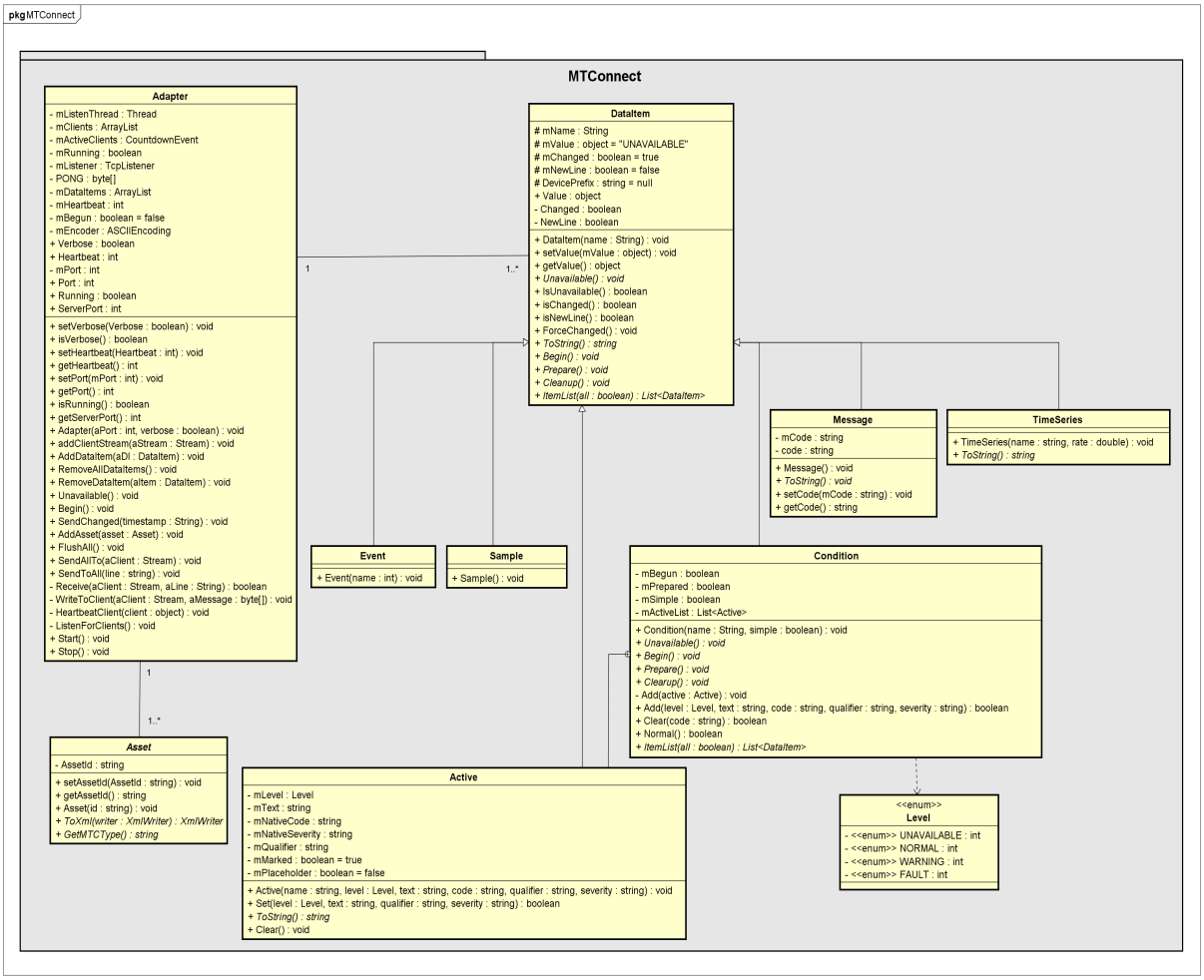

Figura 5.20: Classes do pacote MTConnect implementadas no desenvolvimento do Adaptador

A classe *DataItem* do diagrama representa é superclasse das classes que representam as diferentes categorias de *DataItem* : *Event*, *Sample*, *Codition*, *Message* e; as subclasses *Active*, classe interna da classe *Condition*, e *TimesSeries*, responsável por associar data e hora aos valores dos itens de dados capturados do CNC da máquina. A classe *Condition* também está associada a um objeto de enumeração, o Level, que associa a classe os estados de condição de operação da máquina: *Unavailable*, *Normal*, *Warning* e *Fault*. A classe *Message* também tem a *DataItem* como superclasse e tem a função de receber as mensagens de alerta provenientes do controlador que não configuram alarmes. A atualização do estado de uma condição é preparada pela classe *Active*. Esta classe basicamente recebe o novo valor de uma instância de uma condição e gera uma sequência de caracteres entre barras (protocolo SHDR) para compor o *streaming* de dados que é enviado para o Agente MTConnect.

O *Web Service* para o servidor de supervisão OPCWeb é do tipo RESTful. Utilizando os recursos da API JAX-RS 1.1 (*Java API for RESTful Web Services* 1.1) e *framework* Jersey 1.19.3 a programação desse *middleware* é substancialmente facilitada, pois a API e o *framework*, entre outros recursos, fornecem um conjunto de anotações (@Path, @GET, @POST, etc.) que utilizam URIs ("/{recurso}/{valor}") e os detalhes do protocolo HTTP para acessar os métodos da aplicação Web de forma mais prática. Dessa

forma, a estrutura de classe do *Web Service* é significativamente enxuta, conforme demonstra o diagrama de classes da Figura 5.21.

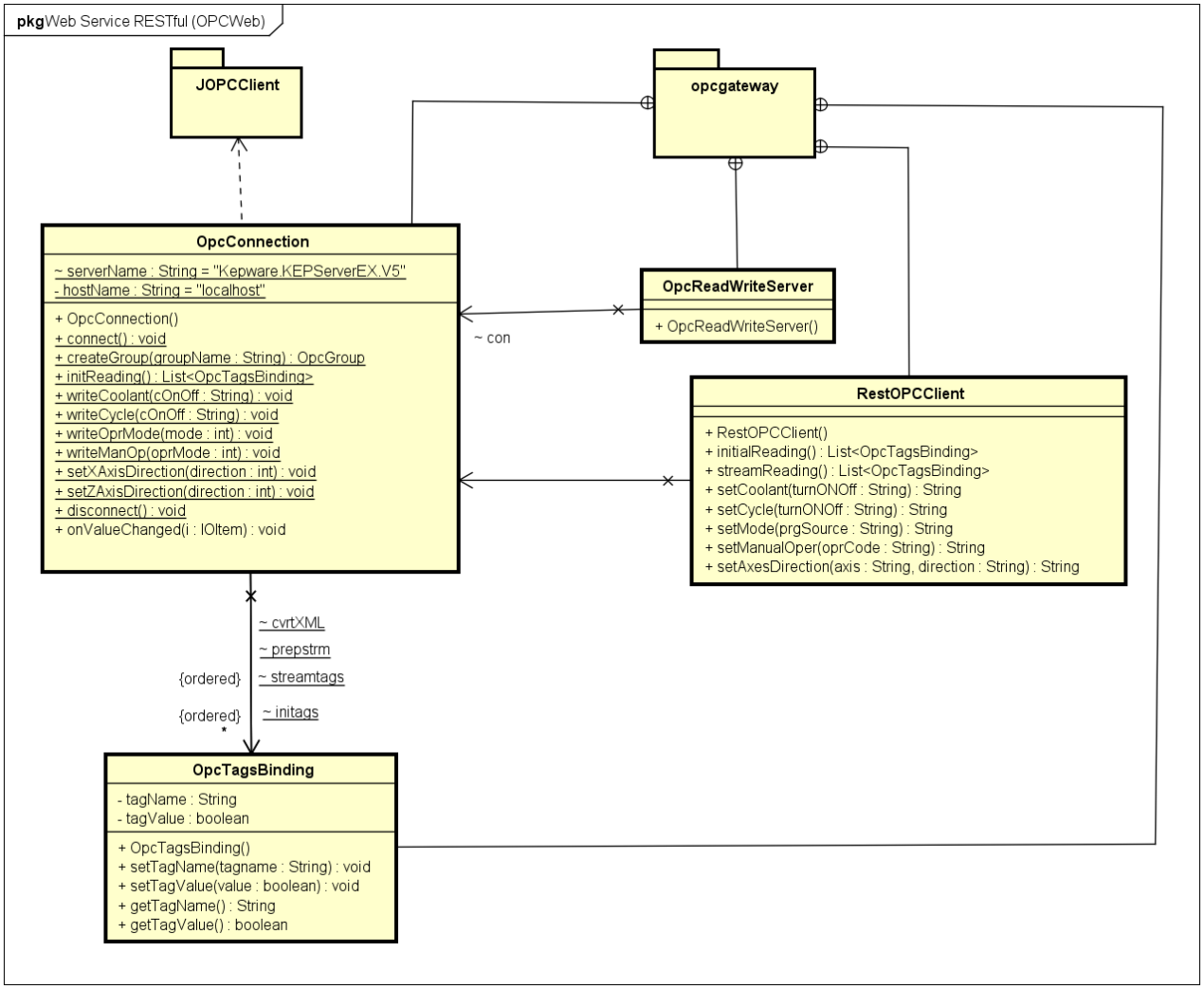

Figura 5.21: Estrutura de Classes da proposta do *Web Service* RESTful do Servidor OPCWeb

Nesse diagrama as classes concebidas foram alocadas em um *namespace* chamado *opcgateway*. Neste está a classe *OpcConnection* que concentra as rotinas para conexão e interação com o servidor OPC-DA, isso inclui geração da estrutura de itens (*tags*) do servidor no cliente (no caso o *Web Service*), e execução da atividades de leitura e atualização do status dos parâmetros de CLP no servidor OPC. Para isso, a classe *OpcConnection* usa as classes e interfaces da API *JOPCClient*, um cliente de acesso a dados preferido para comunicação com servidores OPC especificação DCOM.

A classe *OpcReadWriteServer* tem a função de instanciar a classe *OpcConnection*, e assim iniciar a conexão com o servidor, criação de grupos, e a esses adicionar itens OPC que são o elo com as *tags* do servidor OPC. No mesmo pacote está a classe *OpcTagsBindings* que é uma classe POJO (*Plain Old Java Objects*) e que tem como função principal serializar os itens (OPC *tagname*) e seus valores (*true* para ativado ou *false* para desativado) em formato XML. A classe *RestOPCClient* é um intermediário, recebendo requisições do cliente Web (*browser*) e enviando para o servidor OPC através de métodos da

classe *OpcConnection*, bem como transmitindo dados provenientes do servidor para o cliente Web em uma resposta XML, em um intervalo de atualização fornecido pela aplicação cliente.

Com o propósito de garantir o adequado funcionamento do serviço OPC através da internet, diante de possíveis dificuldades na avaliação desse servidor, uma abordagem alternativa é utilizada para o desenvolvimento do elemento *gateway* para a acesso Web do servidor OPC. Programação não orientada a objeto de *scripts* escritos em linguagem Python para execução em um servidor Web utilizando protocolo CGI em requisições HTTP em um *browser*. Esses programas são lidos pela API *OpenOPC* que faz a comunicação com o servidor OPC-DA. As *tags* OPC manipuladas são as mesmas, tanto na abordagem com *Web Service*, quanto por meio de programas em Python.

5.4.3 Diagramas de Casos de Uso

A análise dos casos de uso está focada nos principais clientes do sistema: Clientes de teleoperação e monitoramento e, cliente de monitoramento e supervisão. Este último caso é apresentado apenas como uma proposta para implementações futuras. Os casos de uso que caracterizam proposta está definido na Figura 5.22.

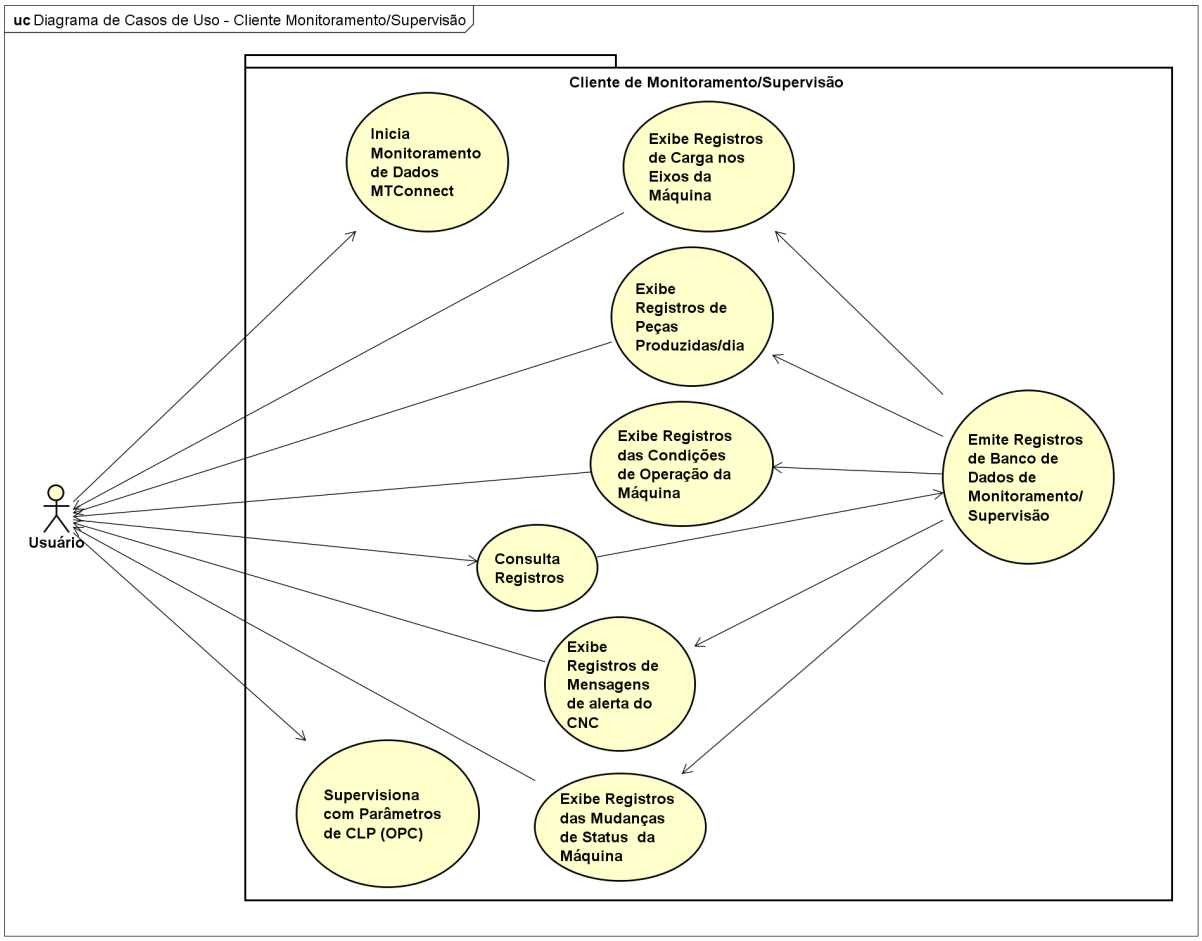

Figura 5.22: Casos de uso do cliente de monitoramento/supervisão

Nesse cliente os casos de uso dividem-se em atividades de monitoramento, supervisão com informações provenientes de bancos de dados, que podem estar na Nuvem, e supervisão em tempo real ou quase tempo real. O monitoramento envolve apenas a tarefa "Iniciar o Monitoramento de Dados MTConnect", que consistiria em clicar em um botão para iniciar a transmissão de dados pelo servidor MTConnect.

A abstração para as tarefas de supervisão tendo como servidor um sistema de banco de dados estão relacionadas às consultas de dados e análises gráficas que envolvem: registro de variação da carga nos eixos da máquina, mudanças registradas nas condições (*Condition*) de operação da máquina, status de execução, número de peças produzidas/dia e registros de mensagens de alerta.

A atividade "Supervisiona com Parâmetros de CLP (OPC)"estão associada à recepção e acompanhamento dos sinais dos controles do painel do controlador da máquina, fornecido por CLP, e transmitido por meio do servidor OPCWeb.

Os casos de uso do cliente Web de Teleoperação e Monitoramento para implementação (Figura 5.23) incluem atividades de comando remoto (DNC), que nesse protótipo de aplicação são representadas por: *Download* de Programa NC, *Upload* de Programa NC, Envia comando MDI, Lista Todos os Programas NC e Deleta Programa NC. A proposta é a execução desses comandos utilizando requisições HTTP com programas CGI.

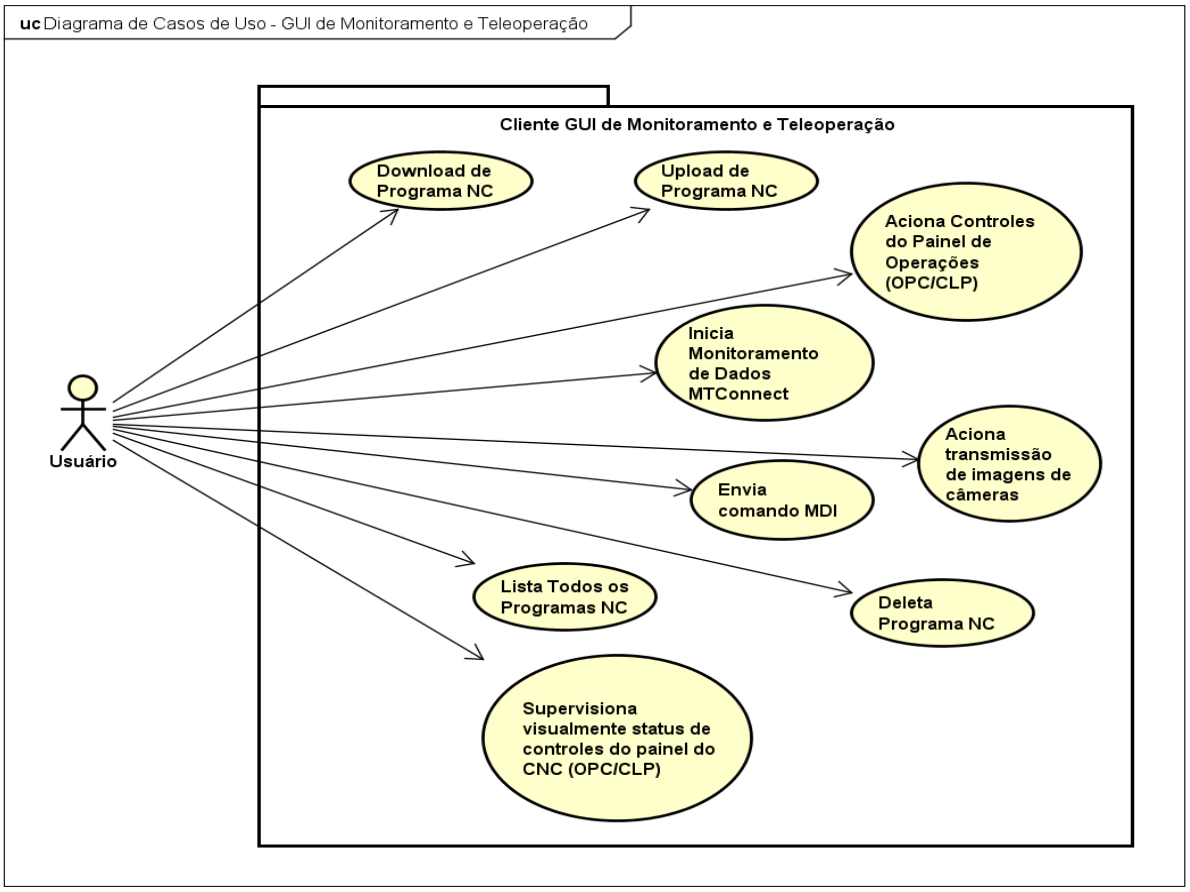

Figura 5.23: Casos de uso do cliente de Teleoperação e Monitoramento

A atividade "Download de Programa NC"correspondente à seleção de um programa (código G/M) em algum diretório do PC e envio para a memória do CNC. O *Upload* é a requisição de visualização de um programa NC no *browser* pelo envio do nome do programa como parâmetro de entrada e a resposta em HTML.

A opção MDI (*Manual Data Input*) é fornecida pelo controlador e permite o envio programas NC ou linhas de comandos em código G/M para acionamento dos eixos da máquina-ferramenta. As duas últimas outras opções comando remoto para implementação é a requisição da lista de programas NC armazenadas na memória do controlador e a exclusão de um programa NC da memória do CNC ("Deleta Programa NC"). Esta função é executada de forma semelhante ao *Upload*, através do envio do nome do programa a ser excluído como parâmetro e a confirmação.

O caso de uso "Inicia Monitoramento de Dados MTConnect", conforme indica o próprio nome, envolve o evento de inicialização *streaming* de dados provindos do servidor MTConnect e o subsequente monitoramento de dados da máquina. O "Acionamento de Controles do Painel de Operações (OPC/-CLP)"representado o clicar nos botões da GUI associados ao comando de escrita (POST) no servidor OPC por intermédio do Web Service. Também associada ao servidor OPC está a atividade "Supervisiona visualmente status de controles do painel do CNC (OPC/CLP)", que para o caso especifico do centro de torneamento Romi Galaxy 15M, estudado neste trabalho, corresponde ao acompanhamento dos sinais dos LEDs dos botões do painel do CNC (0-desligado/1-ligado).

O serviço de *streaming* de vídeo na GUI de teleoperação e monitoramento está ligado ao caso de uso "Aciona transmissão de imagens de câmeras", que é realizada ao clicar em uma área exclusiva para a exibição das imagens ou em botões específicos para cada câmera.

5.4.4 Diagrama de Atividades

O diagrama de atividades mostra a sequência do fluxo das atividades. Geralmente é utilizado para detalhar as atividades executadas por uma operação específica do sistema. Corresponde a estados de ação, que contêm a especificação de uma atividade a ser desempenhada para que uma operação seja efetuada no sistema.

Foram construídos os diagramas de atividades para GUI de teleoperação e monitoramento, desenvolvida no contexto deste trabalho. As sequências de atividades para a execução das operações de aplicação foram detalhadas desde a ação efetuada pelo usuário remoto até a comunicação com o servidor e a mensagem de resposta do mesmo.

Utilizando a sequência das atividades descritas na seção anterior, a atividade de *download* de programa NC, apresentada na Figura 5.24, é a primeira a ser analisada.

Essa atividade é acionada pelo clique do usuário no botão que abre a janela de *download* em um *frame* específico na GUI, para que o arquivo com o programa NC possa ser selecionado em um diretório qualquer do PC do usuário e carregado para envio, para que CNC da máquina possa baixar o programa (*download* na perspectiva da máquina CNC como o cliente). O programa é enviado e é recebido pelo servidor WebDNC, que também é um servidor Web, que por sua vez submete a requisição ao servidor do CNC que salva o programa em sua memória.

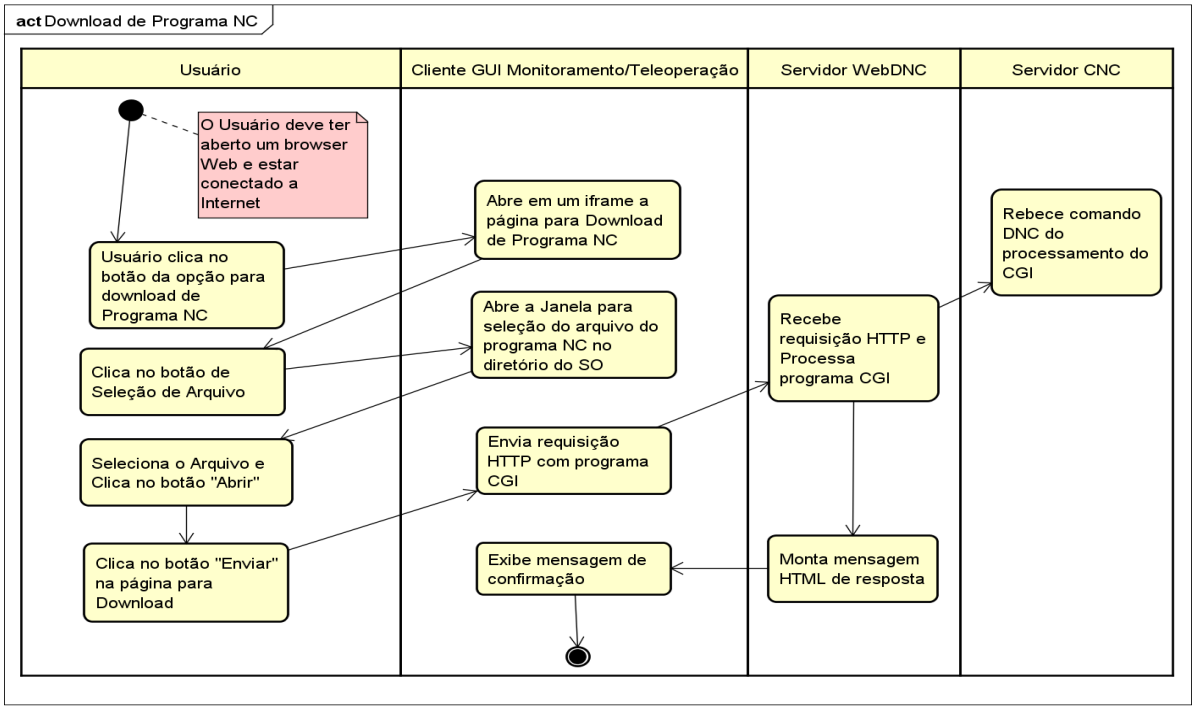

Figura 5.24: Atividade de *Download* de Programa NC

A operação *Upload* de programa NC, que é exibida na Figura 5.25, possui uma sequência de atividades semelhante a operação de *download*, a diferenças são: a necessidade de anexar um parâmetro que identifique o programa e o fato de o servidor CNC junto com o WebDNC gerarem uma resposta HTML para a GUI com o programa NC requisitado.

O caso de envio de um programa por comando MDI é efetuado quando o usuário, após abrir a página da opção MDI, digita o programa ou comando no formulário da página e envia ao servidor WebDNC, utilizando mecanismo CGI. Este CGI é processado e o comando MDI é enviado para o CNC. A Figura 5.26 mostra o diagrama associado a essa função.

A outra opção vinculada aos comandos DNC é a operação de listar os programas NC gravados na memória do CNC, que está descrita na Figura 5.27. A operação resume-se a selecionar a opção para listar programas NC na GUI. Após isso, uma requisição é enviada ao servidor Web, também utilizando mecanismo CGI. Esse pedido é processado no servidor WebDNC, que através da função DNC de visualização do diretório de programas do CNC (API Focas 1), gera no servidor do controlador da máquina a estrutura de dados que será incorporada a resposta HTML, que é exibida na página do cliente Web.

A ação de deletar um programa possui uma sequencia atividades semelhante a de *Upload* de programa, começando pelo usuário que necessita enviar como parâmetro o identificador do programa a ser excluído. A requisição é efetuada aos servidores que executam a função no CNC e enviam uma mensagem HTML de confirmação. O diagrama de atividades desse procedimento é ilustrado na Figura 5.28.

No diagrama de casos de uso da Figura 5.29 a atividade "Aciona Controles do Painel de Operações (OPC/CLP)"compreende eventos relacionados ao acionamento de uma ou mais entre as principais funções

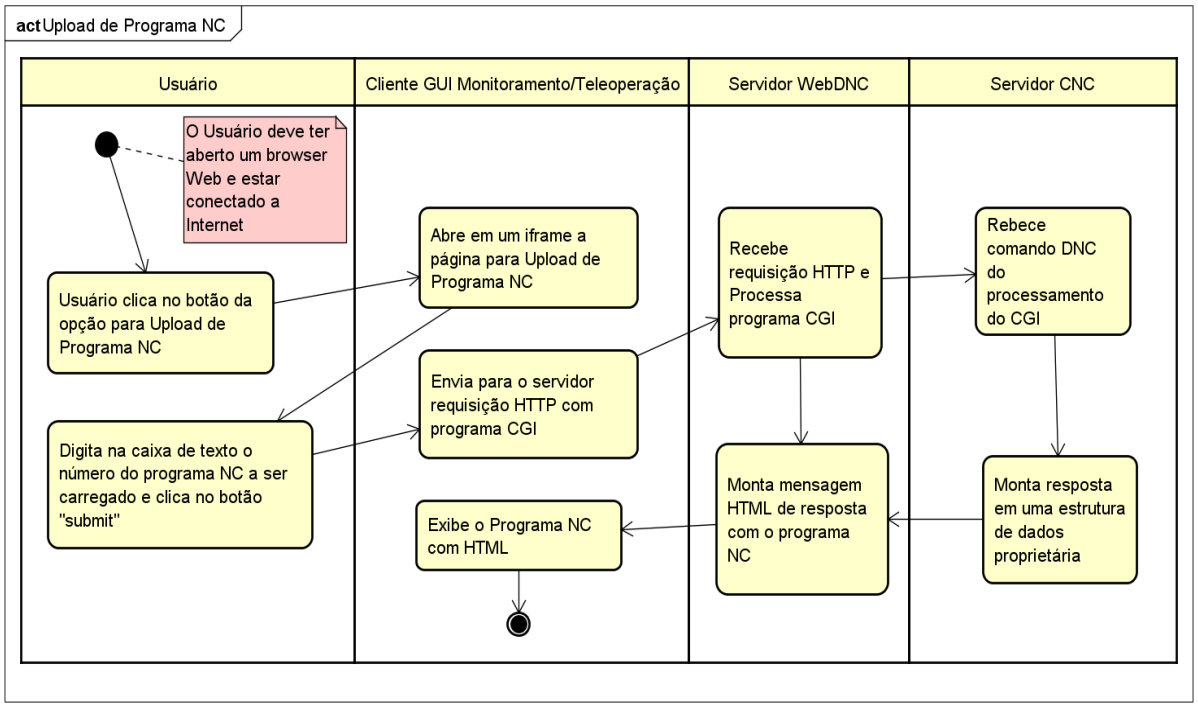

Figura 5.25: Atividade *Upload* de Programa NC

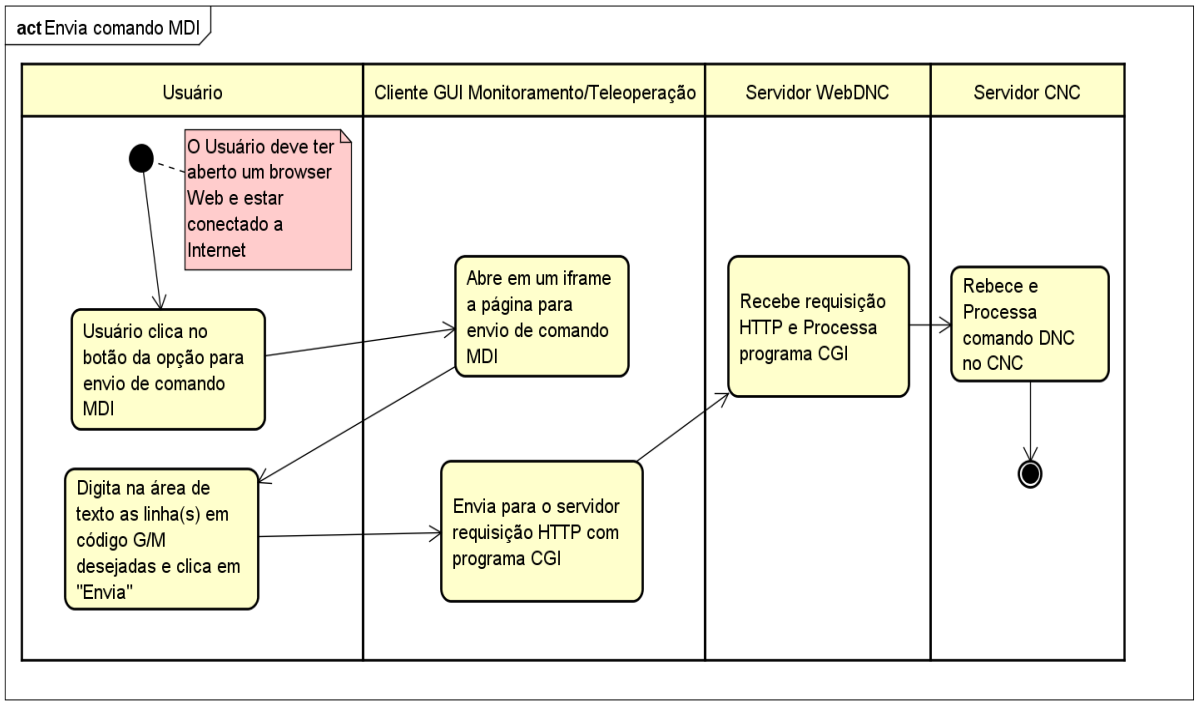

Figura 5.26: Atividade para envio de comando MDI

fornecidas pelo painel de operação do CNC. Essas funções são ativadas logicamente por CLP e para acessá-las remotamente é utilizado um servidor OPC, que neste trabalho recebeu o nome de OPCWeb. O diagrama

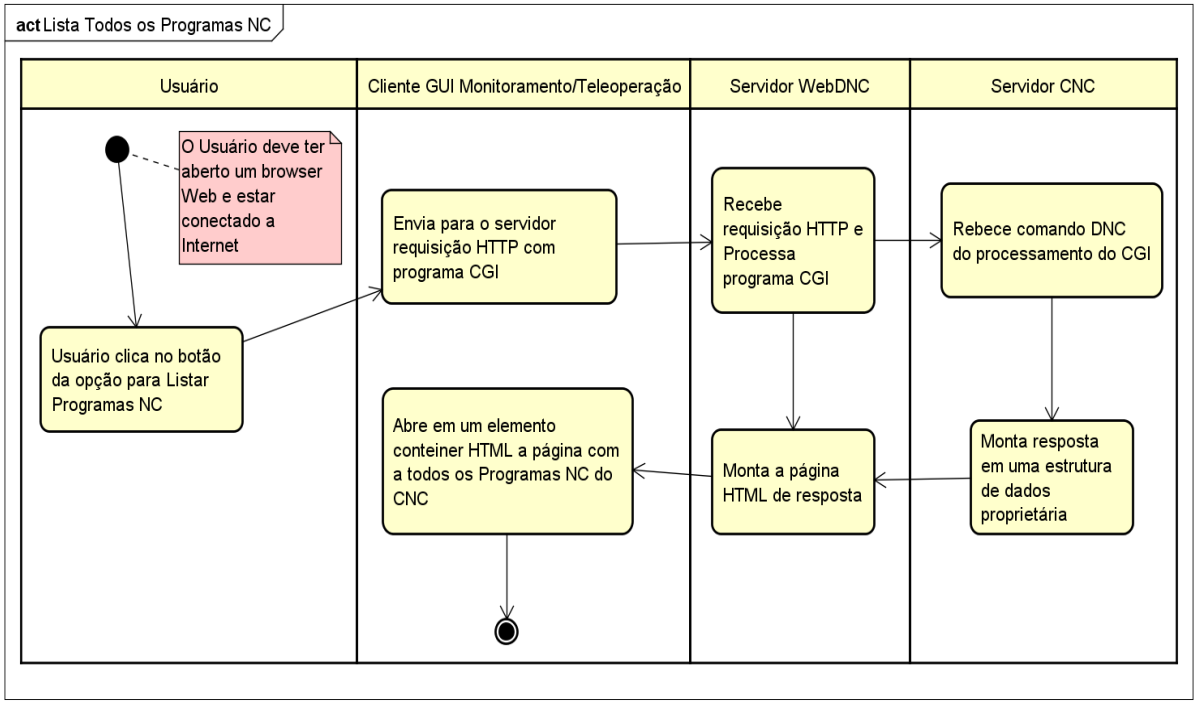

Figura 5.27: Atividade "Listar todos os programas NC"

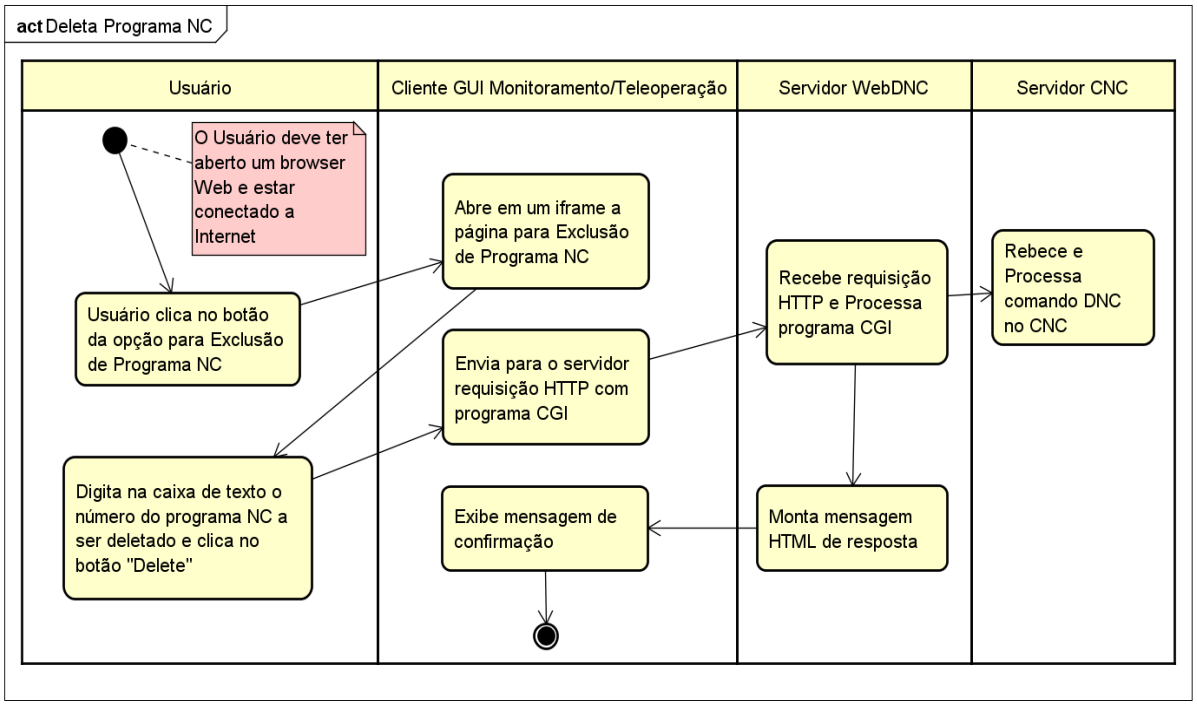

Figura 5.28: Atividade de deletar um programa NC

da Figura 5.29 representa o fluxo de atividades da operação de acionamento de controles OPC(CLP) via Web.

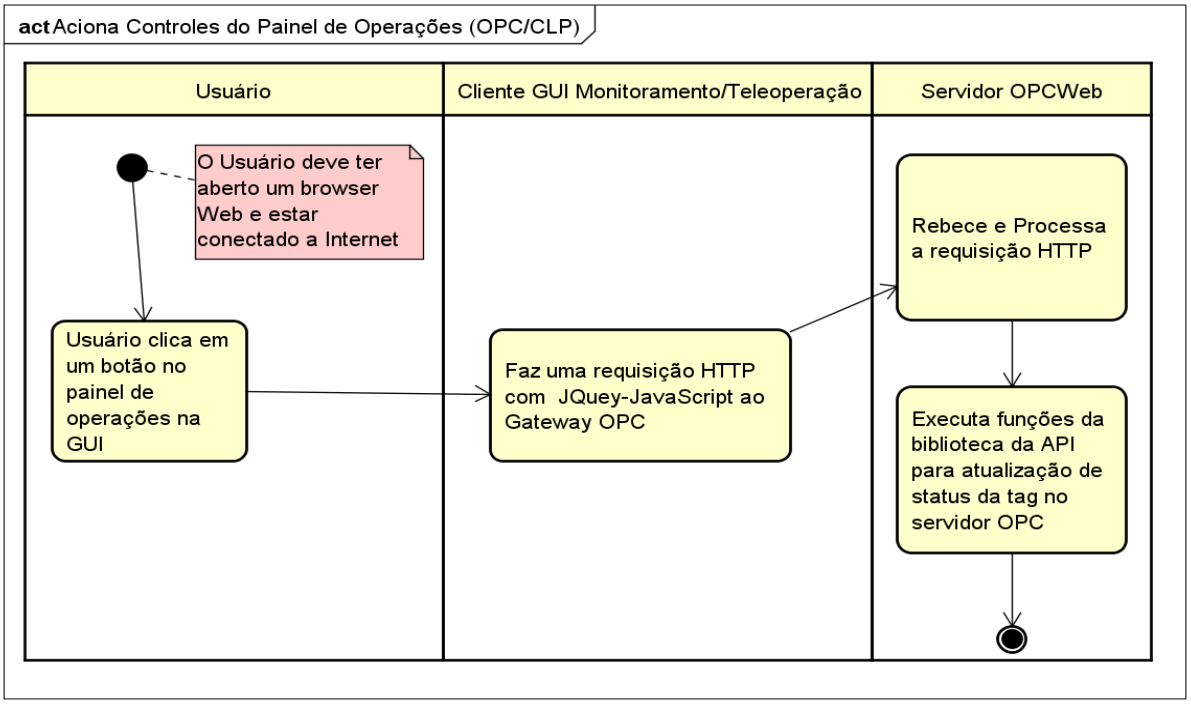

Figura 5.29: Atividade de Acionamento de Controles do Painel de Operações (OPC/CLP)

No *browser* essas atividades são executadas em uma requisição HTTP (PUT) por meio de um *script* com programação JQuery que é enviada ao *Web Service* do servidor OPCWeb para modificar o status do controle no CNC.

Um procedimento semelhante ao da atividade de escrita no servidor OPC, que foi descrito na Figura 5.29, é a operação de leitura do status dos controles de CLP através dos valores das *tags* do servidor OPC, que na GUI de teleoperação e monitoramento representa o caso de uso "Supervisiona visualmente status de controles do painel do CNC (OPC/CLP)"(Figura 5.30).

Nessa atividade a resposta do servidor OPC é convertida no cliente Web em informação visual semelhante ao esquema de LEDs do painel do controlador.

Na GUI o monitoramento é realizado através do protocolo MTConnect. A atividade de iniciar o monitoramento MTConnect é efetuada pelo usuário ao acionar o botão para inicialização da transmissão de dados. Uma requisição utilizando programação *Ajax* é enviada para o Agente MTConnect que monta o XML de resposta com dados de fabricação. Esses dados são recebidos pelo cliente Web, que trata-os utilizando recursos *JQuery.Ajax* e exibe esses dados em HTML. A Figura 5.31 descreve graficamente essa atividade.

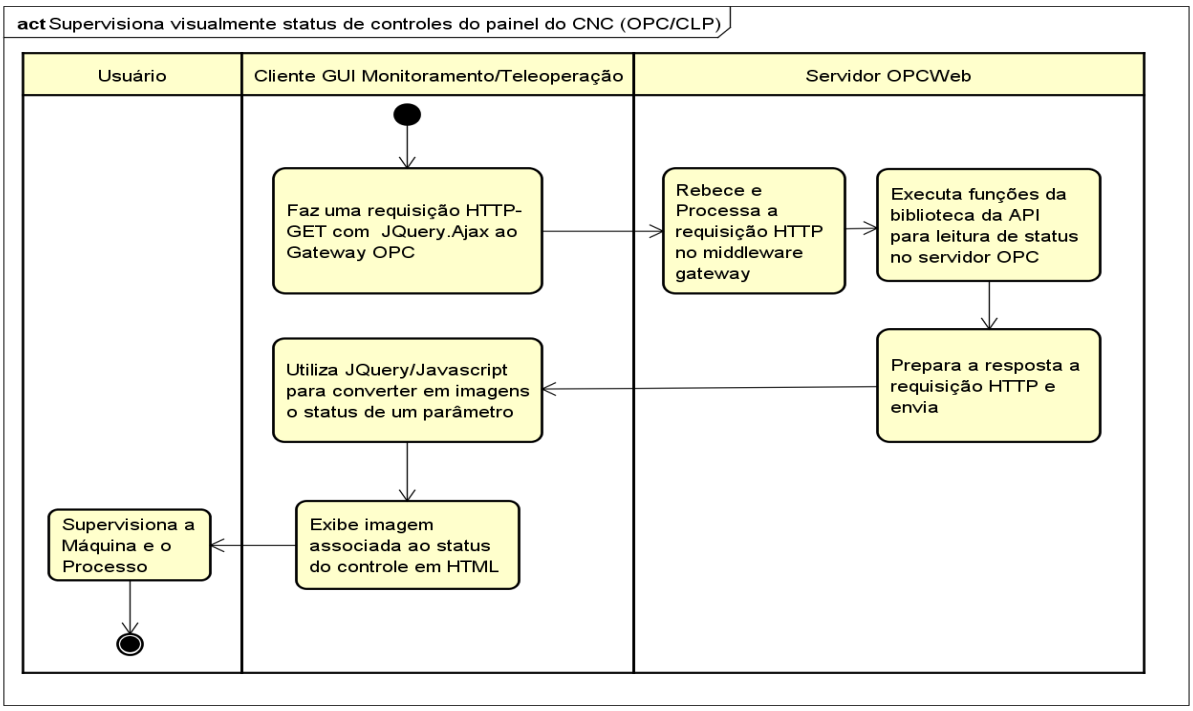

Figura 5.30: Atividade de supervisão com o status dos controles do painel do CNC

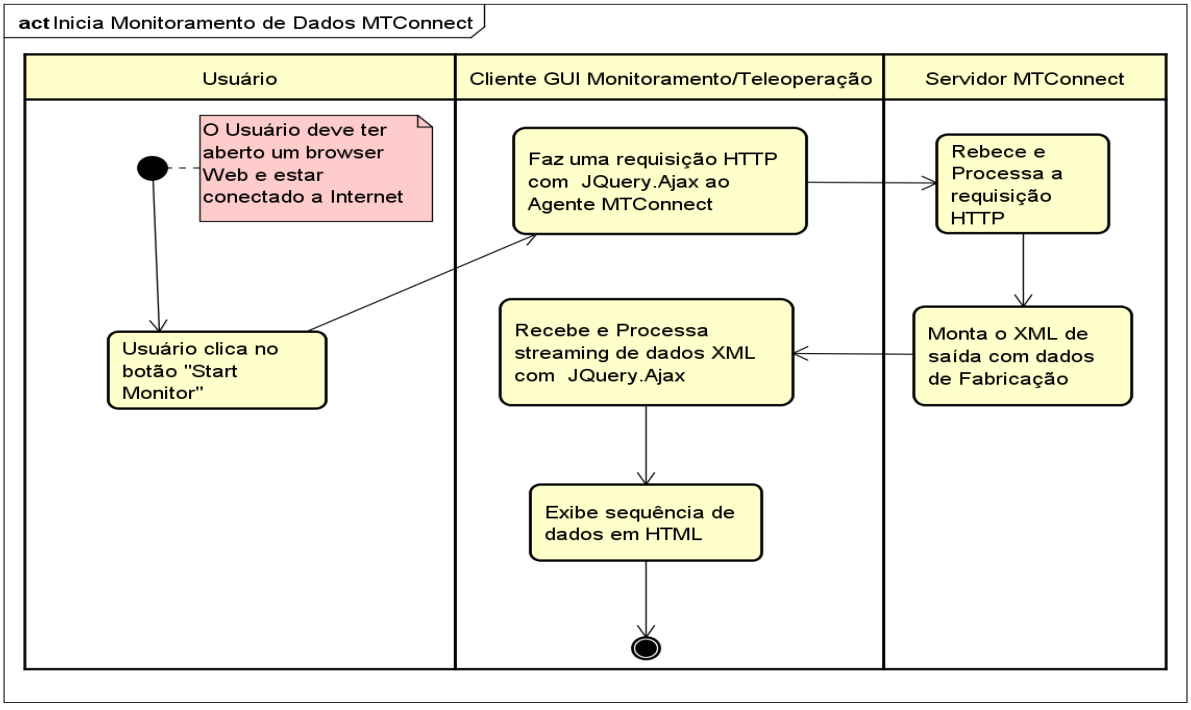

Figura 5.31: Atividade "Inicia Monitoramento de Dados MTConnect"

Capítulo 6

Implementação Computacional do Sistema

As saídas obtidas com a metodologia de projeto axiomático, como o diagrama de fluxo e a representação da arquitetura para implementação do sistema ilustrada na Figura 6.1. Acrescido a isso, a documentação gerada com a modelagem UML, como os diagramas de classe, orientam todo o trabalho de implementação computacional.

O sistema projetado possui uma arquitetura Cliente-Servidor com foco na integração de serviços baseados em padrões aderentes ao paradigma da Indústria 4.0, como o MTConnect e o OPC para internet. Para a implementação dessa integração foi desenvolvido um conjunto de servidores para a comunicação com máquinas-ferramenta CNC e um sistema Web cliente, acessível através de um navegador de internet. O sistema tem a função de teleoperação e monitoramento via internet e tem como elemento de teste e validação um centro de torneamento da marca Romi, modelo *Galaxy 15M*, provido com CNC Fanuc 18i-Ta.

O monitoramento e supervisão baseiam-se essencialmente na aquisição de dados através de um servidor MTConnect, composto do servidor do CNC, software Adaptador e software Agente; e de um servidor OPC, projetado para receber e transmitir dados através da Web por meio da associação entre um software servidor OPC-DA e um *middleware* com função de *gateway*.

A teleoperação é fundamentada na associação do servidor de *streaming* de vídeo com o serviço de comandos DNC (Comando Numérico Distribuído) para comando remoto da máquina via internet por meio de requisições HTTP com *scritps* CGI (*common gateway interface*). É possível considerar que funções do servidor OPCWeb também se enquadram entre os serviços de teleoperação. Esse servidor foi projetado para supervisionar os controles e intervir na operação da máquina-ferramenta, a exemplo da movimentação dos eixos da torre.

Foi implementada também a proposta de uma interface cliente Web que integra os recursos fornecidos pelos servidores em um mesmo ecrã, desenvolvido essencialmente em linguagem HTML com CSS (*Cascade Style Sheet*) e *Javascript* utilizando *plugin JQuery* e metodologia *Ajax*. A parte da arquitetura do sistema efetivamente implementada é representada na Figura 6.1 .

Os funções disponibilizados com a arquitetura de implementação representam o mapeamento virtual de recursos importantes da máquina-ferramenta. A máquina-ferramenta CNC pode ser monitorada e sofrer atuação via internet independente do meio físico (cabo ou *wireless*). O sistema disponibiliza dados em

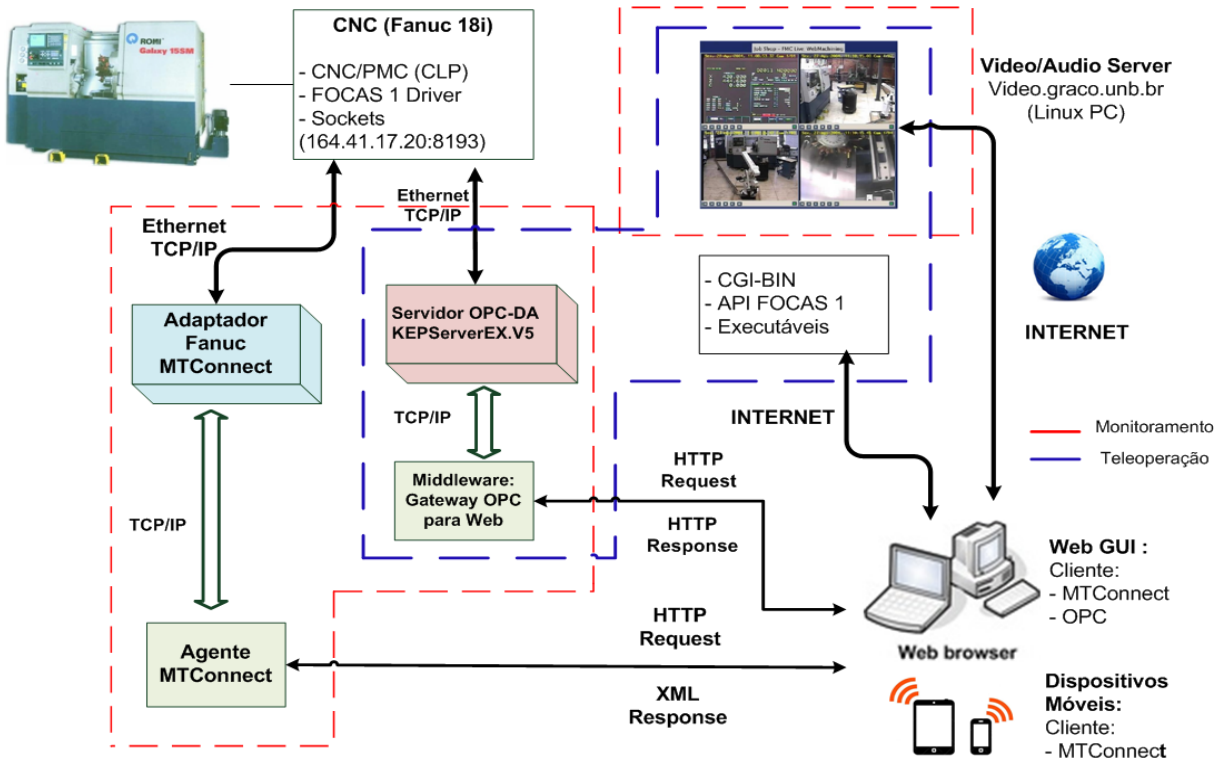

Figura 6.1: Arquitetura implementada

formato universal, como o XML, o que garante maior integração vertical e horizontal em um sistema de produção. Apenas uma infraestrutura de internet robusta com uma largura de banda que garanta elevadas taxas de transmissão e recepção de dados permitirá a realização do potencial da aplicação em operar em tempo real. Os dados fornecidos pelo sistema, em propostas de desenvolvimentos futuros, podem ser salvos e tratados para apoiar tomadas de decisões mais precisas relacionadas a fabricação. Esse conjunto de atributos posiciona o sistema como uma aplicação para Internet de Serviços (IoS), e portanto, na Indústria 4.0

Durante o trabalho de implementação houve a necessidade de aproveitar os benefícios de um variado número de linguagens de programação. Começando pelo servidor de monitoramento, utilizou-se da versatilidade e robustez da linguagem C# na plataforma *.Net* para a programação do adaptador do CNC para o agente MTConnect. Este último teve seu código fonte escrito em C++ , também na *Microsoft.Net Framework*, que foi estudado antes da sua instalação e configuração. O servidor de supervisão OPC, chamado de OPCWeb, contou com um módulo *gateway* que teve duas versões, uma em *Web Service* RESTful desenvolvido em Java com *framework* Jersey, a outro com *scripts* desenvolvidos em python utilizando mecanismo CGI. Os serviços de teleoperação utilizados na implementação computacional da arquitetura, em sua maioria, não foram necessariamente desenvolvidos, mas foram recursos herdados de outros projetos e sofreram ajustes para a sua implementação neste trabalho. Diferentemente, o serviço de atuação remota via servidor OPCWeb foi programado no âmbito deste projeto antes de sua utilização. Entre os serviços herdados de projetos anteriores, há o servidor de *streaming* de vídeo (WebCam) (ALVARES, 2005) que utilizou *Applets* Java e os *scripts* CGI para comandos DNC (ALVARES, 2005), ambos são parte do serviço de teleoperação.

O cliente Web para integração e teste é desenvolvido em HTML, utilizando CSS, PHP e *JavaScript* com biblioteca *JQuery*. Na programação desse módulo optou-se pelo uso de linguagens interpretadas (não compiladas), mais leves, a fim de demonstrar a versatilidade da arquitetura implementada.

6.1 Servidor CNC-Focas1

O Servidor FOCAS1 (*Fanuc Open CNC API Specifications* 1) está localizado no CNC Fanuc 18i-ta do centro de torneamento *Galaxy* 15M. O acesso a esse servidor é possível através de uma rede Ethernet utilizando protocolo TCP/IP e API *FOCAS1*/Ethernet. Esse servidor é localizado na rede via *socket* configurado com IP 164.41.17.20 e porta 8193. A partir desse *socket* tem-se acesso às funções do protocolo de aplicação FOCAS1/DNC1 que são executadas no CNC.

O Adaptador do servidor MTConnect é compilado junto com a API FOCAS 1 para que possa conectar-se ao controlador e execute as funções de DNC. No desenvolvimento do Adaptador algumas dessas funções foram encapsuladas a fim capturar informações relevantes para diferentes finalidades, que incluem parâmetros de fabricação e de status da máquina. A interface gráfica (GUI) desenvolvida no contexto deste trabalho também comunica-se com esse servidor através de um servidor Web (WebDNC), e um *browser* de Internet que executa requisições HTTP e que via *socket* envia uma estrutura de dados para o CNC, implementando as funções disponibilizadas pelo protocolo FOCAS1/DNC1. Esse servidor é proprietário da GE Fanuc, sendo instalado e configurado apenas no CNC. A API e *driver* de desenvolvimento *FOCAS1*/Ethernet está disponível apenas para a plataforma Windows, um dos motivos pelos quais optou-se pelo desenvolvimento do Adaptador do servidor MTConnect em IDE Microsoft Visual Studio 2010 com linguagem C#.Net, no sistema operacional Windows XP (a API Focas 1 não é compatível com versões superiores do sistema operacional da Microsoft).

No Adaptador do servidor MTConnect cada função do protocolo FOCAS1/DNC corresponde a um método, cada um com a tarefa de requisitar informações, que podem ser: Posição atual e comandada dos eixos (X, Z e C), Avanço, Modo de operação (Auto, Edit, MDI e JOG), Velocidade do eixo rotacional (*Spindle*), entre outros.

Na conexão com o servidor CNC-FOCAS1 através de um servidor Web pelo cliente (GUI), cada função FOCAS1/DNC corresponde a um programa CGI, que possui rotinas que executam tarefas como: leitura de dados enviados por formulário, montagem da *streaming* de dados a ser enviado; inicialização do *socket* (IP: 164.41.17.20, e porta do CNC: 8193); estabelecimento da comunicação via estrutura de dados FOCAS1; envio e recebimento de *streaming* de dados, fechamento do *socket*; verificação do *streaming* de dados recebido e; montagem da página HTML que é apresentada ao usuário com a resposta. Esses recursos foram utilizados na implementação do sistema *Webturning* (ALVARES, 2005) que foi uma das principais referências de implementação para este trabalho. Faz-se maiores considerações acerca do fluxo de informações entre o servidor *FOCAS 1*/Ethernet, o Adaptador MTConnect e o Cliente Web implementados no contexto deste trabalho nas próximas seções deste capítulo.

6.2 Servidor MTConnect

A implementação de um servidor MTConnect completo é o principal elo da arquitetura do sistema com a Industria 4.0. Esse nova era da manufatura baseia-se no uso da Internet e em aplicações software baseadas na Web com capacidade de processar fluxos de dados de fabricação. O MTConnect é um padrão bastante versátil construído sobre outros padrões abertos e protocolos consolidados (XML, TCP/IP, HTTP) para a Internet, facilitando o esforço de integração de dados em um ambiente industrial e permitindo maiores de níveis de interoperabilidade entre dispositivos e operadores humanos, e entre dispositivos. Consequentemente, essas características fazem do MTConnect um padrão crítico para sistemas de IoT Industrial.

O servidor MTConnect compõe um serviço para transmissão de parâmetros de fabricação e de dados sobre o status da máquina provenientes do controlador da máquina-ferramenta. Toda a estrutura do serviço é composta por um dispositivo, que no contexto deste trabalho é representado pelo Servidor CNC Fanuc 18i-Ta, além do software Adaptador para esse controlador com API Focas 1 e pelo software Agente, estes dois últimos instalados em uma mesma máquina, mas separados pelo servidor CNC-Focas1.

O Adaptador foi desenvolvido em linguagem C# na plataforma .Net (Dot Net) da Microsoft utilizando IDE Visual Studio 2010. Parte da aplicação foi programada sobre um *toolkit* para Adaptador MTConnect desenvolvido na MC2 Technical Session, conduzida por William Sobel, *Chief Strategy Officer* na *System Insights,inc*., durante a MC2 Conference de 2013, e disponibilizado no repositório do MTConnect Institute (*https://github.com/mtconnect/dot_net_sdk*) sob licença livre. O principal tarefa na implementação do Adaptador foi mapear as funções da biblioteca *fwlib32.dll* (*driver* Focas1), escritas em linguagem C, e encapsulá-las em métodos de mais simples compreensão e manipulação na linguagem C# (C-Sharp). O quadro da Tabela 6.1 lista algumas funções da biblioteca *fwlib32.dll* e os métodos sob os quais essas funções foram encapsuladas.

Tabela 6.1: Encapsulamento das funções da biblioteca *fwlib32.dll* (Focas 1)

Funções da biblioteca *fwlib32.dll*	**Métodos da classe FocasGateway (Adaptador)**
FWLIBAPI short WINAPI cnc_allclibhndl3(const char ipaddr, unsigned short port, long timeout, unsigned short FlibHndl);	public bool connect(ref ushort aFlibHndl)
FWLIBAPI short WINAPI cnc_freelibhndl(unsigned short FlibHndl);	public void disconnect()
FWLIBAPI short WINAPI cnc_statinfo(unsigned short FlibHndl, ODBST *statinfo);	public string getControllerMode()
FWLIBAPI short WINAPI cnc_actf(unsigned short FlibHndl, ODBACT *actualfeed);	public int getActualFeedRate()
FWLIBAPI short WINAPI cnc_rdsvmeter(unsigned short FlibHndl, short data_num, ODBSVLOAD loadmeter);	public int[] getAxesLoad(ref short[] dec)

A partir das funções mapeadas no Adaptador são produzidos os seguintes resultados e parâmetros:

conexão e desconexão com a máquina;

programa NC atual em execução (*program*)

bloco do programa em execução (*block*)

linha atual do programa (*line*)

a) status de execução (*execution*)

b) modo de operação controlador (*mode*)

c) status de parada de emergência (*emergency stop*)

d) ferramenta (*tool ID*)

d) número de peças produzidas (*Part Count*);

e) Avanço Atual dos Eixos (*Actual Path Feedrate*);

f) Avanço Atual em porcentagem (*Actual Path Feedrate Override*);

g) Posição atual dos eixos (Xact, Zact e Cact);

h) Posição comandada dos eixos (Xcomm e Zcomm);

i) Carga dos eixos (Xload, Zload e Cload);

j) Velocidade do eixo rotacional principal (*Spindle Speed*-S1speed);

k) Carga do eixo rotacional (*Spindle Load* - S1load) e;

l) Alarmes (relacionado ao elemento *Condition* - *system*, *servo*, *communication*, *logic control*, *motion*, *hardware*, *temperature*, *overload* e *overtravel*).

Os códigos fonte das classes do Adaptador em que esses parâmetros são capturados está descrito nos Apêndices C (Classe *FocasGateway*) e D (Classe *FanucPath*).

A operação do Adaptador é simplificada através de uma interface gráfica básica programada em *Windows Forms*, conforme pode ser visualizado na Figura 6.2 abaixo.

A interface gráfica de inicialização do Adaptador é formada pelos campos de endereço IP e porta do CNC da máquina, para inclusão do *socket* de comunicação com a máquina, além de campos para a inclusão do endereço onde o software Agente localiza a transmissão de dados provenientes do Adaptador, a taxa de atualização é definida internamente de modo fixo no próprio código fonte do Adaptador. Independente do tempo de ciclo de transmissão definido no nível de aplicação, o que irá determinar se a serviço MTConnect operará em tempo real é a largura de banda do canal, nesse caso o serviço de Internet. Conforme mencionado anteriormente, para a comunicação com a máquina-ferramenta o Adaptador é associado a API FOCAS1/Ethernet da Fanuc através da instalação da biblioteca *Fwlib32.dll*. Esta biblioteca dá acesso a

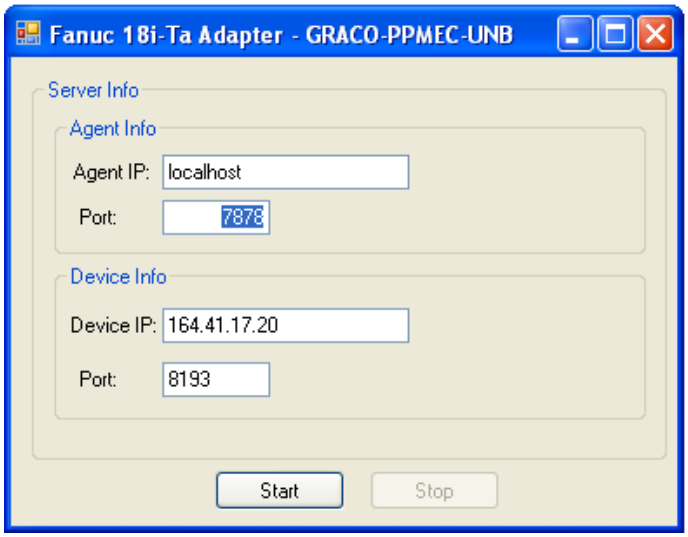

Figura 6.2: Interface gráfica do Adaptador para GE Fanuc 18i

cerca de 300 funções para controle CNC/DNC. Nos Apêndices C e D é detalhado o código fonte associado a programação do Adaptador do servidor MTConnect.

Após a conexão com a máquina-ferramenta, o Adaptador captura dados de parâmetros do CNC no formato proprietário do fabricante do CNC e os converte em formato SHDR (*Simple Hierarchical Data Representation*) , um protocolo de leitura simples, rápido e de baixa latência que é representado por barras (|), e é associado ao formato para data/hora estabelecido pela ISO 8601, conforme é exemplificado a seguir:

2016-11-04T19:51:46843700Z | mode | MANUAL | Cload | 0,05 |

Nesse exemplo o adaptador lança na porta de rede especificada os valores dos parâmetros *controller mode* (*mode*) e carga do eixo rotacional (*Cload*), juntamente com o valor de *timestamp* (data e hora). O Agente extraí os nomes dos parâmetros e o valores nas barras, e a partir da estrutura de dados do CNC definida no arquivo *Devices.xml*, constrói o *streaming* dados de saída em uma resposta XML do tipo *MTConnectStreams*.

As informações são enviadas do dispositivo para o Adaptador e, em seguida, para o Agente de forma contínua.

O software Agente possui código aberto que é disponibilizado pelo MTConnect Institute em sua conta na plataforma *Github* (*https://github.com/mtconnect/cppagent*). A instalação foi realizada no mesmo PC (*host*) que o Adaptador. A execução do instalador do Agente ocorre em duas etapas, a primeira consiste nas rotinas básicas de transferência dos arquivos essenciais para o diretório de programas do sistema operacional e a introdução de parâmetros essenciais para a configuração do Agente (Figura 6.3). O segunda etapa é quando os parâmetros de localização e funcionamento do Adaptador são vinculados ao Agente (Figura 6.4).

Após a execução do programa de instalação, houve a necessidade de configurar o Agente a fim de associá-lo ao Adaptador do CNC Fanuc 18i-Ta. Com isso, no diretório de instalação do Agente, foi in-

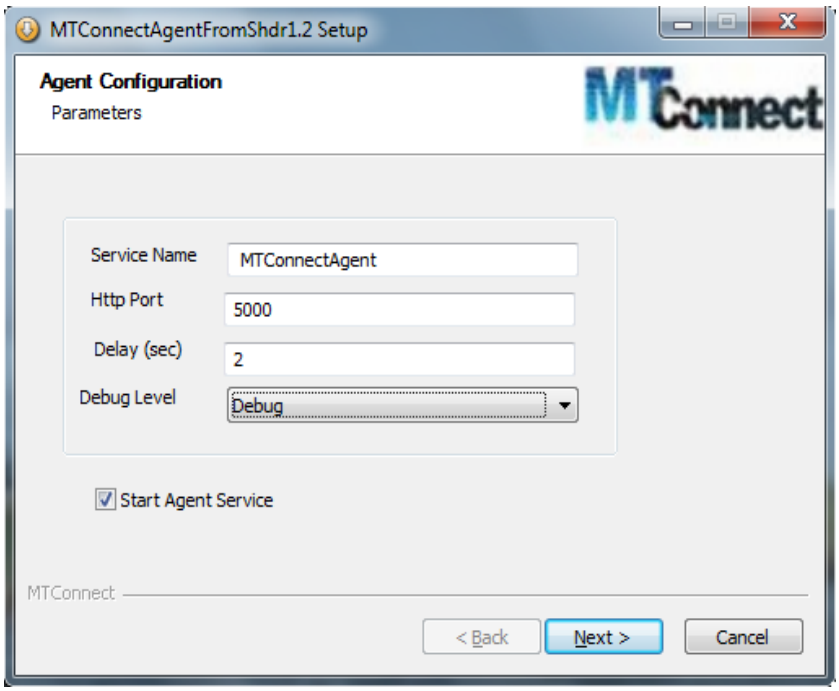

Figura 6.3: Instalação do Agente: configuração do Agente

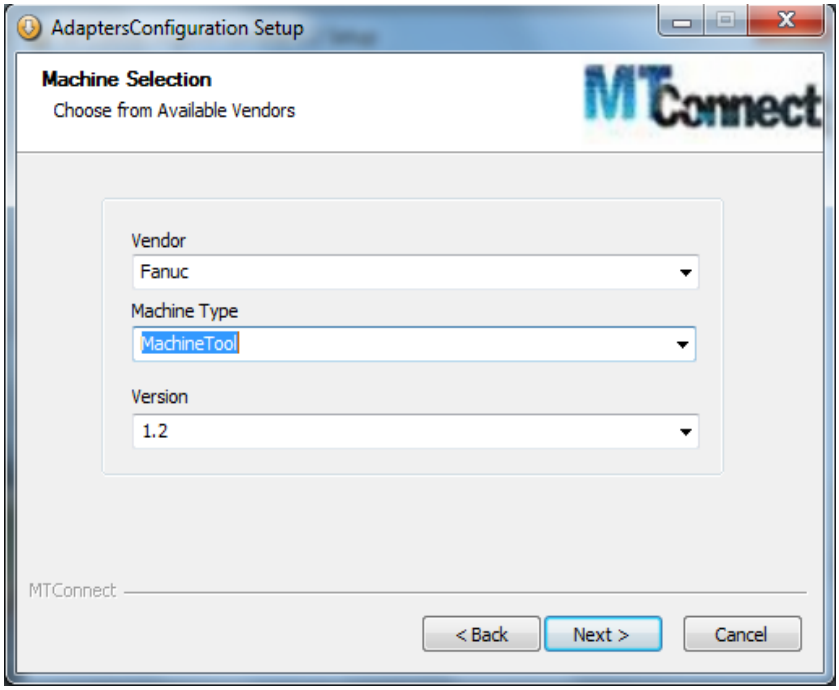

Figura 6.4: Instalação do Agente: configuração do Adaptador

cluído o arquivo *Fanuc-CNC3axis.txt* e modificado o arquivo *Agent.cfg*. O primeiro é um arquivo *template* com o modelo dos dados que serão capturados do dispositivo e que irá preencher automaticamente o arquivo *Devices.xml*, localizado no diretório principal da instalação. O segundo é o arquivo de configuração do Agente, onde são definidos parâmetros como localização do Adaptador na rede (*host* e *port*), dire-

tório de esquemas XSD (*XML Scheme File*) (arquivos que baseiam-se na especificação, são fornecidos pelo *MTConnect Institute* e suas versões são referenciadas em todos os cabeçalhos dos XMLs de resposta do Agente), configurações de *log*, porta em que o Agente está acessível, frequência de *checkpoint* e *ServiceName*, que foi configurado com o nome *MTConnectAgent*. Os conteúdos completos dos arquivos *Devices.xml* e *Fanuc-CNC3axis.txt* estão, respectivamente, nos Apêndices A e B, e os detalhes do arquivo *Agent.cfg* na Figura 6.5.

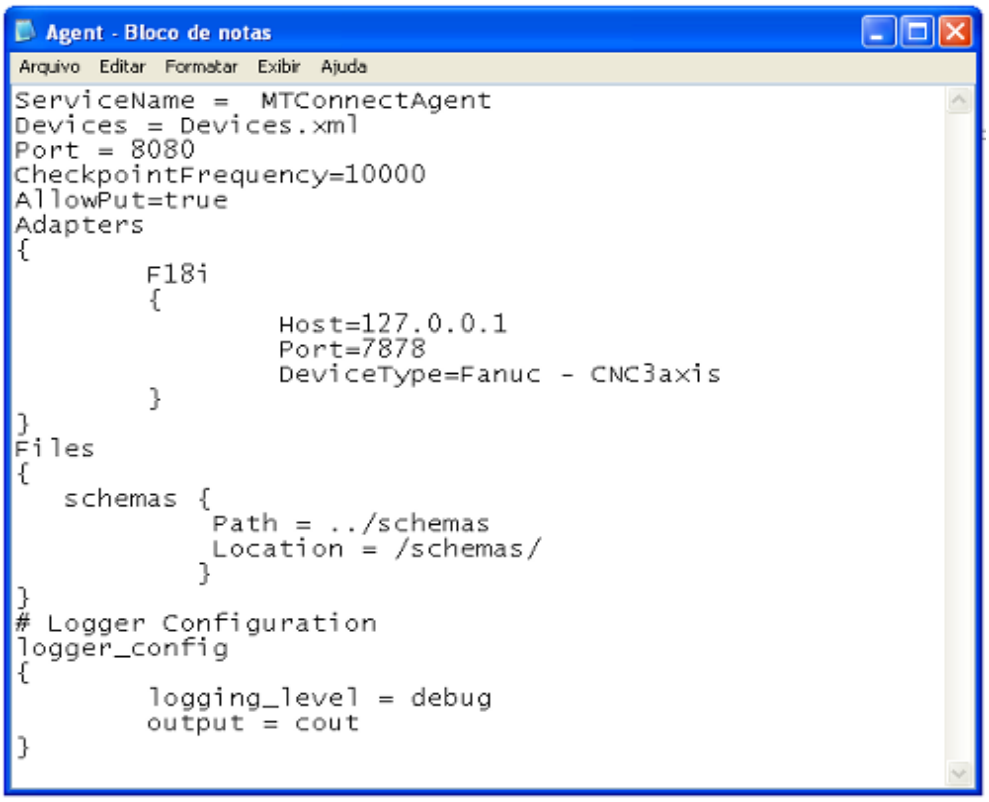

Figura 6.5: Configurações do arquivo *Agent.cfg*

Após a instalação e a devida configuração do Agente, e a inicialização do Adaptador, é possível monitorar os parâmetros de CNC do centro de torneamento Romi Galaxy 15M. O Agente *MTConnectAgent* é executado como um serviço no sistema operacional Windows XP da Microsoft, e fica disponível para ser utilizado sempre que o sistema operacional é iniciado, atendendo as requisições de dados da aplicação cliente. Testes bem-sucedidos foram realizados com o Agente sendo executado em outro PC da rede separado do Adaptador. No entanto, optou-se pela praticidade de ter as duas aplicações sendo executadas no mesmo computador. Durante a operação do servidor MTConnect os dados fluem na forma que ilustra a Figura 6.6.

Assim,através da Figura 6.7 é possível ter a visão detalhada da arquitetura do serviço MTConnect que foi implementado.

O serviço pode ter como cliente remoto tanto estações de trabalho com acesso da Internet por meio de um *browser*, como aplicações para dispositivos móveis.

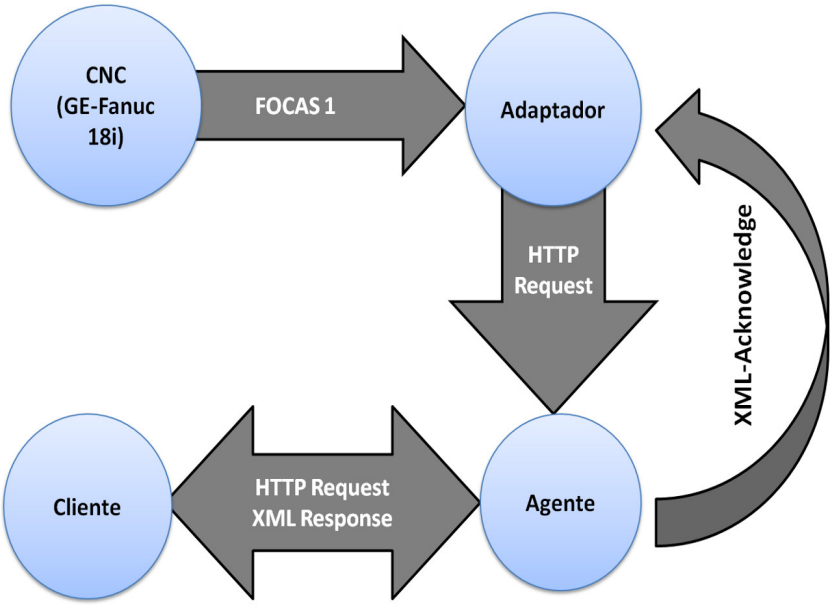

Figura 6.6: Servidor MTConnect - Fluxo de informação

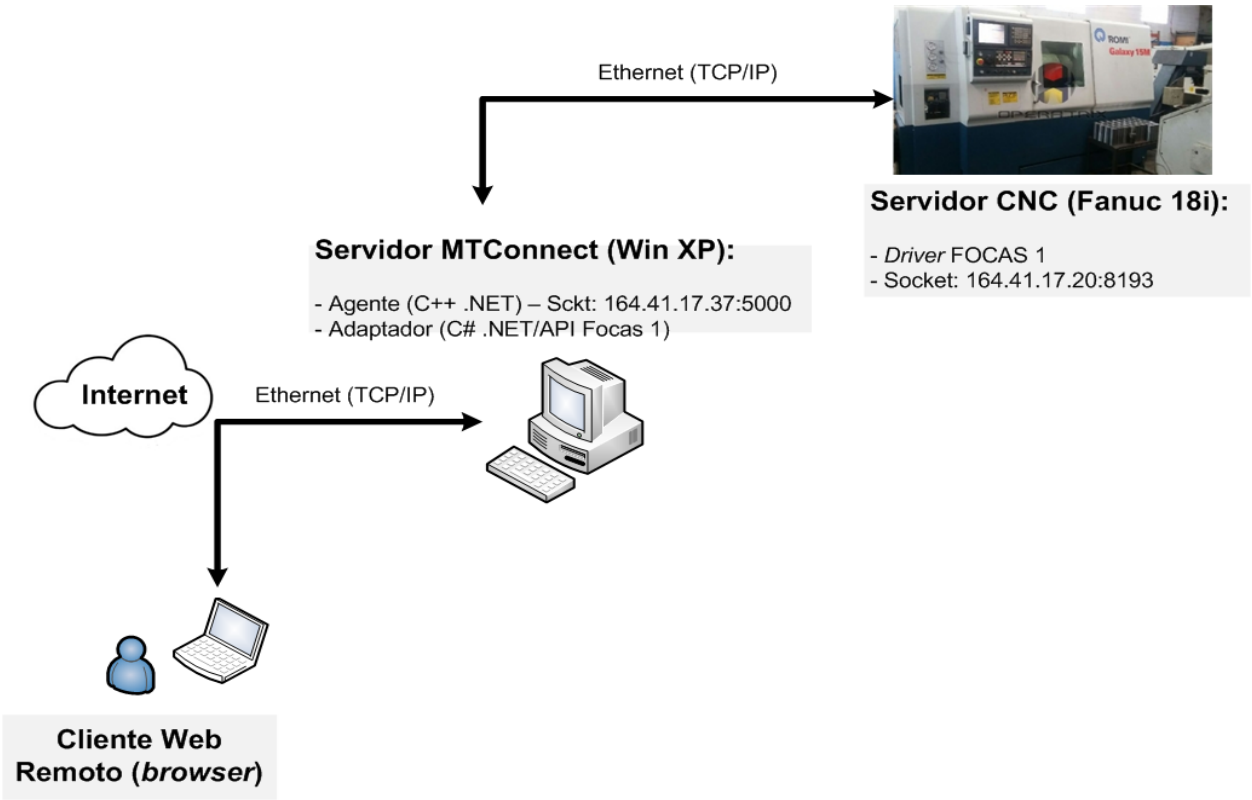

Figura 6.7: Arquitetura detalhada do serviço MTConnect implementado

6.3 Servidor OPCWeb

O paradigma da Industria 4.0 tem como um dos padrões promissores com potencial para impulsionar essa nova era da manufatura o OPC-UA (*Open Platform Communcations-United Archtecture*). Este protocolo evoluiu em relação ao seu predecessor, o OPC clássico, para possibilitar a independência de plataforma e a possibilidade de prover a qualquer dispositivo um servidor OPC embarcado. No entanto, no contexto deste trabalho, OPC-UA não foi tecnicamente utilizado, pois ainda é uma especificação em desenvolvimento, possui um menor número de trabalhos como referências em relações ao OPC clássico, sendo dessa forma um padrão ainda em processo de consolidação.

O padrão OPC-UA foi especialmente projetado, entre outros fatores, para melhorar o desempenho do transporte de dados através de redes Intranet e Internet, uma vez que provê a utilização do protocolo TCP otimizado, e a capacidade de aceitar padrões como *Web Service*, XML e HTTP. Para a proposta deste trabalho utilizou-se como referência produções acadêmicas relacionadas a manipulação de servidores OPC-DA clássico através da Web (TORRISI et al., 2007; PANTONI; TORRISI; BRANDãO, 2007; BALIEIRO, 2008; SAHIN; BOLAT, 2009; ABBAS; MOHAMED, 2011; TORRISI, 2011; MAHMOUD; SABIH; ELSHAFEI, 2015) e o trabalhos que implementaram a especificação OPC XML-DA (TORRISI et al., 2007; PANTONI; TORRISI; BRANDãO, 2007; BALIEIRO, 2008; TORRISI, 2011; YIN; ZHOU, 2012), para criar uma arquitetura que funcionalmente fosse similar a um servidor OPC-UA, que é um padrão e uma tecnologia que pactua com a estratégia Indústria 4.0.

O servidor OPCWeb recebeu esse nome por integrar uma tecnologia existente e amplamente implementada, servidores OPC-DA, com um software *gateway* entre o OPC e a Web. Juntos esses módulos fazem monitoramento, supervisão e o controle via internet de uma máquina-ferramenta CNC.

No que se refere a servidor OPC-DA, optou-se pela utilização de uma tecnologia existente. Foi selecionado o KEPServerEX.V5 (versão 5.17.495.0) desenvolvido pela *Kepware Technologies* motivado pela sua robustez e facilidade de uso. A interface gráfica da janela principal dessa ferramenta é exibida na Figura 6.8. Essa ferramenta foi configurada para operar como um servidor OPC-DA para execução em ambiente Microsoft Windows XP.

Com o software servidor OPC-DA é possível a criação de entidades denominadas Canais (*Channel*), Dispositivos (*Devices*), e aquelas previstas na norma, os Grupos (*Group*) e as *Tags*, itens dos grupos. Dessa forma, para instanciar o servidor na rede, e posterior associação ao cliente OPC, foi criado o canal *Galaxy*. Neste canal foram adicionados dois elementos *Devices*, denominados *rfanuc* (*read*) e *wfanuc*(*write*). Nestas duas entidades foram criadas para representar as *tags* de leitura separadamente das de escrita do CLP (Controlador Lógico Programável) da máquina-ferramenta CNC, respectivamente. Os endereços vinculados as *tags* foram mapeadas visualmente dos elementos da programação *Ladder* do CLP na máquina-ferramenta CNC estudo de caso deste trabalho. As *tags* incluídas na entidade *rfanuc* correspondem a endereços do CLP, que em sua maior parte, representam os parâmetros dos controles associados aos LEDs dos botões painel de operações do CNC Fanuc 18i-Ta. Todos os endereços identificados representam *tags* do tipo *boolean*, em que o estado dos controles alternam entre *true* ou *false*, equivalente aos estados binários "0"(zero), geralmente correspondendo a desligado, ou "1"(um), representando o estado ligado.

As *tags* de pertencentes ao *Device wfanuc* são utilizadas para a atuação nos controles do painel de

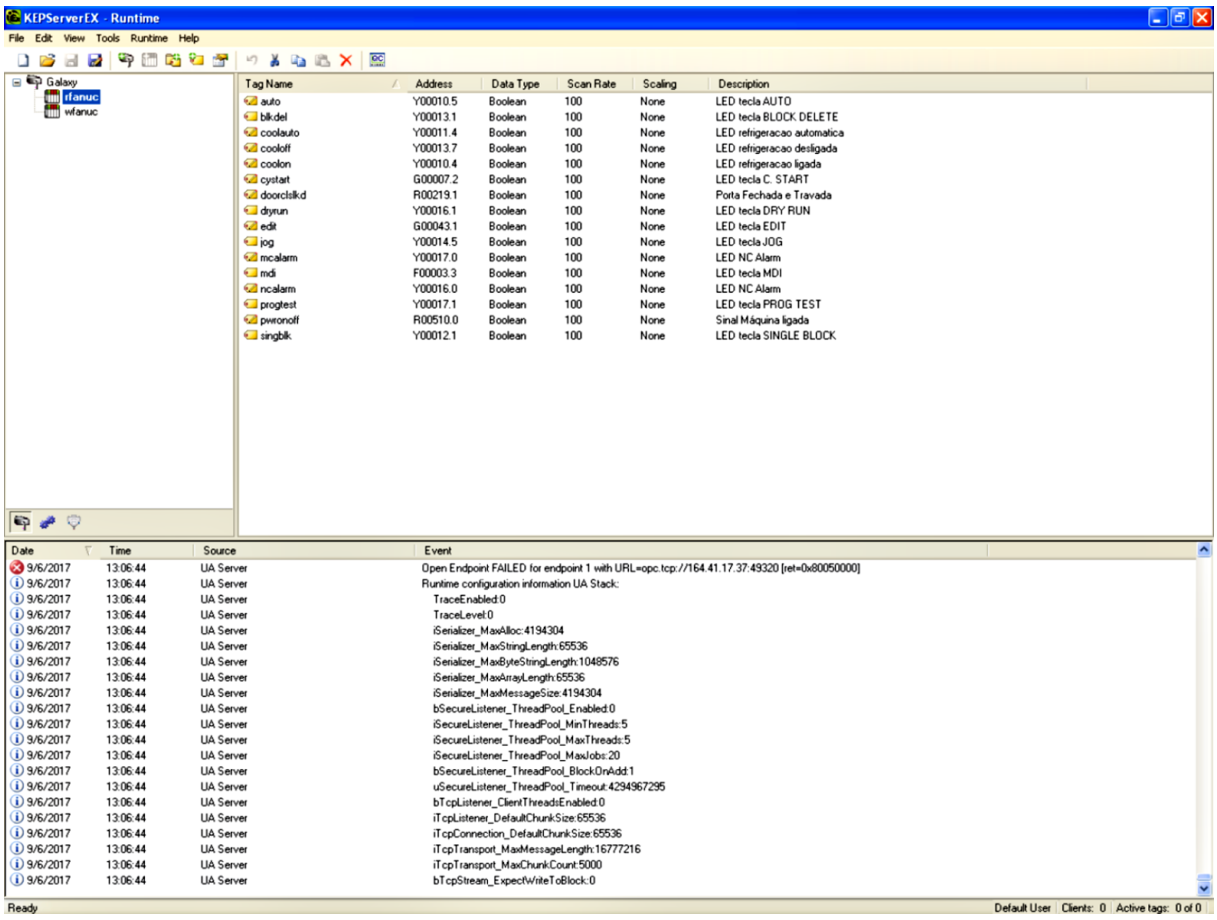

Figura 6.8: Janela inicial do software servidor OPC: KEPServerEX.V5 da Kepware Tecnologies.

operações do centro de torneamento, cujo acionamento é controlado por CLP. Todas também são do tipo *boolean*, e tem os seus estados alterados pela aplicação cliente em "0"(desativado) ou "1"(ativado).

O conjunto de endereços mapeados do *Ladder* do CLP do controlador do centro de torneamento Galaxy 15M, vinculados aos respectivos identificadores das *tags* OPC são representados no quadro da Tabela 6.2.

As funções de CLP/PMC mapeadas não esgotam todas as opções de controle CLP da *Galaxy* 15M. Os itens adicionados ao servidor OPCWeb tem a finalidade de demonstrar o controle e a supervisão dos parâmetros controláveis por CLP da máquina-ferramenta de forma remota via Internet. Tanto na entidade *rfanuc* quanto na *wfanuc* foram adicionados os controles associados às teclas do painel de operações e sinais como:

- Modo de operação do controlador

  Auto - Execução Automática;

  Edit - Acessa edição de programas; *MDI* - Entrada de dados manual, usando para inserir um ou mais blocos de dados manualmente;

  Jog - Ativa a movimentação dos eixos por teclas direcionais.

- Execução do programa CNC

Tabela 6.2: *Tags* mapeadas do *ladder* do controlador da máquina-ferramenta Romi Galaxy 15M

Grupo	Tag Name	Endereço	Data Type	Descrição
rfanuc	auto	Y00010.5	Boolean	LED tecla AUTO
	blkdel	Y00013.1	Boolean	LED tecla BLOCK DELETE
	coolauto	Y00011.4	Boolean	LED refrigeração automática
	cooloff	Y00013.7	Boolean	LED refrigeração desligada
	coolon	Y00010.4	Boolean	LED refrigeração ligada
	cystart	G00007.2	Boolean	LED tecla C. START
	doorclslkd	R00219.1	Boolean	Porta Fechada e Travada
	dryrun	Y00016.1	Boolean	LED tecla DRY RUN
	edit	G00043.1	Boolean	LED tecla EDIT
	jog	Y00014.5	Boolean	LED tecla JOG
	mcalarm	Y00017.0	Boolean	LED MC Alarm
	mdi	F00003.3	Boolean	LED tecla MDI
	ncalarm	Y00016.0	Boolean	LED NC Alarm
	progtest	Y00017.1	Boolean	LED tecla PROG TEST
	singblk	Y00012.1	Boolean	LED tecla SINGLE BLOCK
	pwronoff	R00510.0	Boolean	Sinal de Máquina Ligada
wfanuc	wauto	R00030.0	Boolean	Aciona tecla AUTO
	wblkdel	Y00013.1	Boolean	Aciona tecla BLOCK DELETE
	wcoolauto	R00033.4	Boolean	Aciona tecla COOL. AUTO
	wcooloff	R00033.3	Boolean	Aciona tecla COOL. OFF
	wcoolon	R00033.2	Boolean	Aciona tecla COOL. ON
	wcystart	R00033.5	Boolean	Aciona tecla C. START
	wcystop	R00033.6	Boolean	Aciona tecla C. STOP
	wdryrun	Y00016.1	Boolean	Aciona tecla DRY RUN
	wedit	R00030.1	Boolean	Aciona tecla EDIT
	wjog	R00030.3	Boolean	Aciona tecla JOG
	wmdi	R00030.2	Boolean	Aciona tecla MDI
	wprgtest	Y00017.1	Boolean	Aciona tecla PROG TEST
	wsingblk	Y00012.1	Boolean	Aciona tecla SINGLE BLOCK
	wxm	R00031.6	Boolean	Aciona tecla -X (MINUS)
	wxp	R00031.5	Boolean	Aciona tecla +X (PLUS)
	wzm	R00032.0	Boolean	Aciona tecla -Z (MINUS)
	wzp	R00031.7	Boolean	Aciona tecla +Z (PLUS)

Single Block - Ativa/desativa a execução de programa bloco a bloco;

Block Delete - Ativa/desativa a eliminação de bloco. Qualquer bloco precedido de barra (/) é eliminado.

- Acionamento ou não do eixo árvore

  Dry Run - Ativa/Desativa teste de programa sem ligar eixo árvore;

  Prg Test - Ativa/Desativa teste de programa sem movimentação da máquina.

- Ativação/Desativação de refrigeração

  Coolant ON - Liga sistema de refrigeração;

Coolant OFF - Desliga sistema de refrigeração.

Coolant AUTO - Refrigeração a acionada automaticamente durante a execução do programa.

- Ativação/interrupção do programa CNC

  Cycle Start - Ativa execução de programa;

  Cycle Stop - Interrompe execução de programa.

- Movimentação dos Eixos da Torre

  +X/-X - Movimentação do eixo *X* na direção positiva ou negativa;

  +Z/-Z - Movimentação do eixo *Z* na direção positiva ou negativa;

- Alarmes e Alertas

  NC Alarm - Indicação de alarme de software;

  MC Alarm - Indicação de alarme da máquina;

  Door Closed-Locked - Sinal de porta fechada e travada.

Com a função de um *gateway* entre o servidor OPC-DA e os clientes OPC na Web, foram desenvolvidas duas soluções. A primeira utiliza-se um *Web Service* (RESTful) desenvolvido em linguagem Java, utilizando API JAX-RS 1.1 (*Java API for RESTful Web Services* 1.1) e implementação de referência Jersey 1.19.3. A API JAX-RS fornece suporte ao desenvolvimento de *Web Services* baseado na arquitetura REST (*Representational State Tranfer*) e provê um conjunto de anotações, classes e interfaces para expor uma classe POJO (*Plain Old Java Objects*) como um RESTful, a fim permitir uma programação simplificada e de alto nível. A Apêndice D mostra a classe POJO implementada no contexto deste trabalho com anotação "@xmlRootElement", utilizada para serializar dados provenientes do servidor OPC em listas de dados em formato XML.

O *Jersey* é um conjunto de APIs *open source* que representam uma forma de implementação, estendendo JAX-RS com características adicionais e utilidades a fim simplificar ainda mais o desenvolvimento de clientes e *Web Services* RESTful.

O *Web Service gateway* OPC foi implementado em IDE Eclipse Luna (4.4.2), com JDK 1.7, sendo executado sobre o servidor Tomcat 7. Para comunicação com o servidor OPC-DA previamente instalado e configurado, o *Web Service* utiliza API JOPCClient (versão 2.103). Esta API foi selecionada entre outras opções existentes no mercado como a *EasyOPC* e a *Open Scada*, por ser a mais robusta para a programação de aplicações cliente OPC. Essa biblioteca foi instalada e instanciada no código do *Web Service*, e auxiliou na implementação da classe de leitura e de escrita para acessar as funções CLP/PMC mapeadas para as *tags* definidas na implementação do servidor OPC-DA.

Uma classe na especificação JAX-RS é determinada como um recurso, na codificação do *Web Service* uma classe que ilustra bem esse conceito o elemento *RestOPCClient*. Esta classe mapeia métodos HTTP (PUT e GET) em operações CRUD (*Create*, *Retrieve*, *Update* e *Delete*), utilizados para ler e atualizar o status dos controles no servidor OPC. A Figura 6.9 mostra um método dessa classe com a anotação "@PUT", cuja função é atualizar o valor da *tag* do servidor OPC associada ao modo de operação do CNC,

enviando como parâmetro um código (*progSource*) que representa a *tag* OPC, e em último nível do fluxo de dados, a própria tecla do painel do CNC associada ao modo de operação.

```
@PUT
@Produces("text/plain")
@Consumes("text/plain")
@Path("/write/program/{progSource}")
public String setMode(@PathParam("progSource") String prgSource) throws RbxIOException,
InterruptedException{
    {

        String status = "";
        int prgsource = Integer.parseInt(prgSource);
        if (prgsource > 9 && prgsource < 14){
           OpcConnection.writeOprMode(prgsource);
            status = "Mode Changed!";
        }else{
            status = "Mode Not Changed!";
        }
        return status;
   }
}
```

Figura 6.9: Método da classe RestOPCClient com função de atualizar (*write*) o *status* de operação do controlador

No método representado na Figura 6.9 apresentam mais três anotações: "@Produces", "@Consumes"e "@Path". A primeira define o tipo de retorno da URI solicitada, que no caso do método é uma resposta do tipo texto ("text/plain"). A anotação "@Consumes", contrariamente, tem a função de definir o tipo de mídia que o recurso consome, que no caso desse método é do tipo *text/plain*, mas poderia ser definido com outras extensões MIME como *application/XML* ou *application/json*, por exemplo. No "@Path"define-se o caminho onde o recurso (método) pode ser localizado. Com isso, para acessar o recurso *setMode(...)* requer um requisição da com a seguinte URI: http://{host}/opcServerRestGalaxy15M/write/program/{progSource}. O Apêndice E apresenta maiores detalhes da programação da classe *RestOPCClient Web Service* RESTful. A arquitetura de operação do serviço OPCWeb implementado com o *Web Service* RESTful ficou como ilustra a Figura 6.10.

As segunda solução para o *gateway* do servidor OPCWeb foi desenvolvida utilizando a API *OpenOPC* para Python, que é um *toolkit* para o desenvolvimento de clientes OPC baseado em linguagem Python. O *OpenOPC* independe de plataforma, trabalhando tanto em *Windows* quanto em plataformas não *Windows*. A API é instalada junto com um modulo que faz a comunicação com o servidor OPC-DA e funciona com um serviço no sistema operacional, chamado *OpenOPC gateway service*.

Com a API foram desenvolvidos programas em Python para comunicação com o servidor OPC e execução das mesmas funções de leitura e escrita no CLP do centro de torneamento desenvolvidas para a abordagem com *Web Service*. Nessa solução os programas em python são processados em um servidor web Apache por meio do protocolo CGI (*common gateway interface*) através de requisições HTTP. Os resultados dessas requisições representam mudanças no status de LEDs e funções do centro de torneamento. Um exemplo de código python para efetuar o acionamento da função *Cycle Start* do controlador da máquina-ferramento é exibido na Figura 6.11.

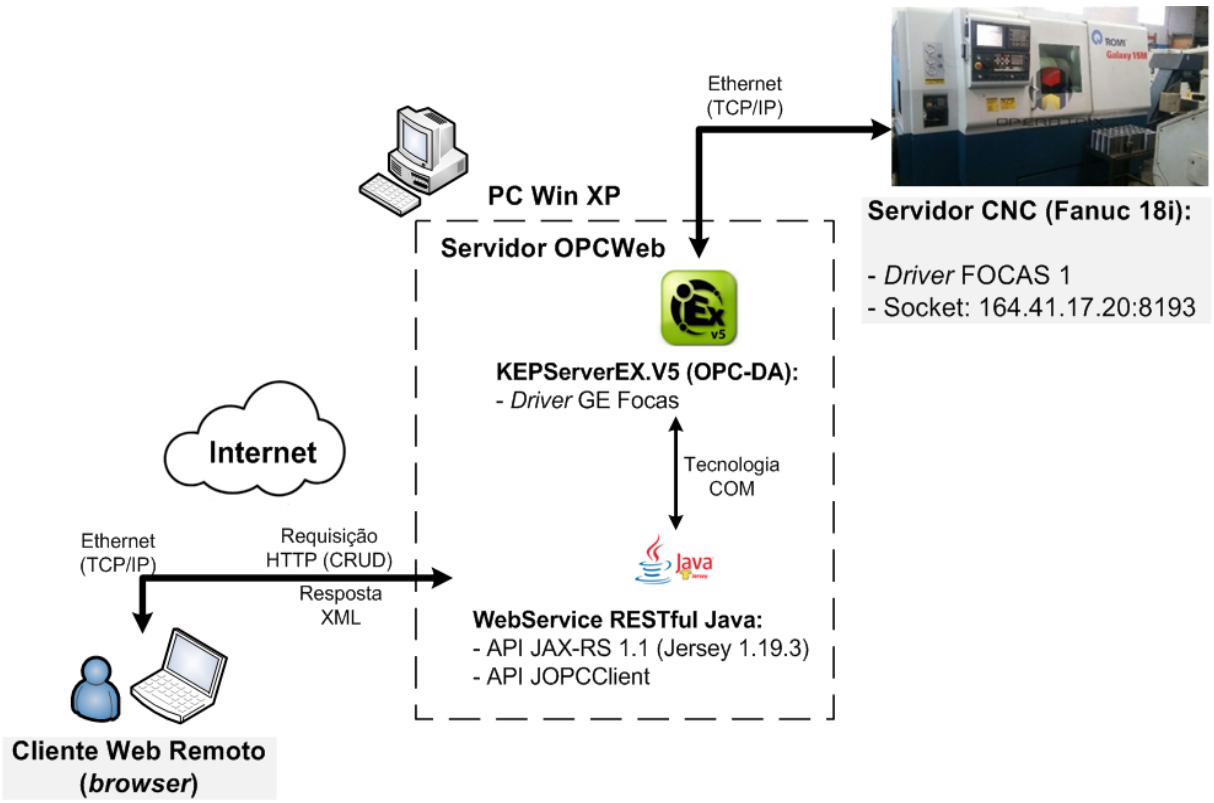

Figura 6.10: Servidor de supervisão OPC via internet (OPCWeb) com Web Service RESTful

```
#!C:\Python27\python.exe

import OpenOPC
opc = OpenOPC.open_client('localhost')
opc.connect('Kepware.KEPServerEX.V5')
opc.write(('Galaxy.wfanuc.wcystart',True))
opc.write(('Galaxy.wfanuc.wcystart',False))
opc.close()
```

Figura 6.11: Programa python a acionar a tecla *Cycle Start* por meio da API OpenOPC

A programação utilizando python é simplificada, apenas algumas linhas código são suficientes para conectar-se e interagir com o servidor OPC. A API e os programas foram executados no mesmo PC Windows XP onde foi instalado o servidor OPC-DA (Kepware KEPServerEX.V5). Por essa alternativa de desenvolvimento, o servidor OPCWeb recebeu a configuração apresentada na Figura 6.12. O Apêndice G apresenta os detalhes dos códigos dos programas desenvolvidos em python para o servidor OPCWeb.

6.4 Servidor WebCam: Transmissão de Vídeo

Este servidor foi criado durante o desenvolvimento do sistema de teleoperação *Webturning* que faz parte da metodologia de manufatura distribuída via internet *Webmachining* (ALVARES, 2005) e foi herdado por

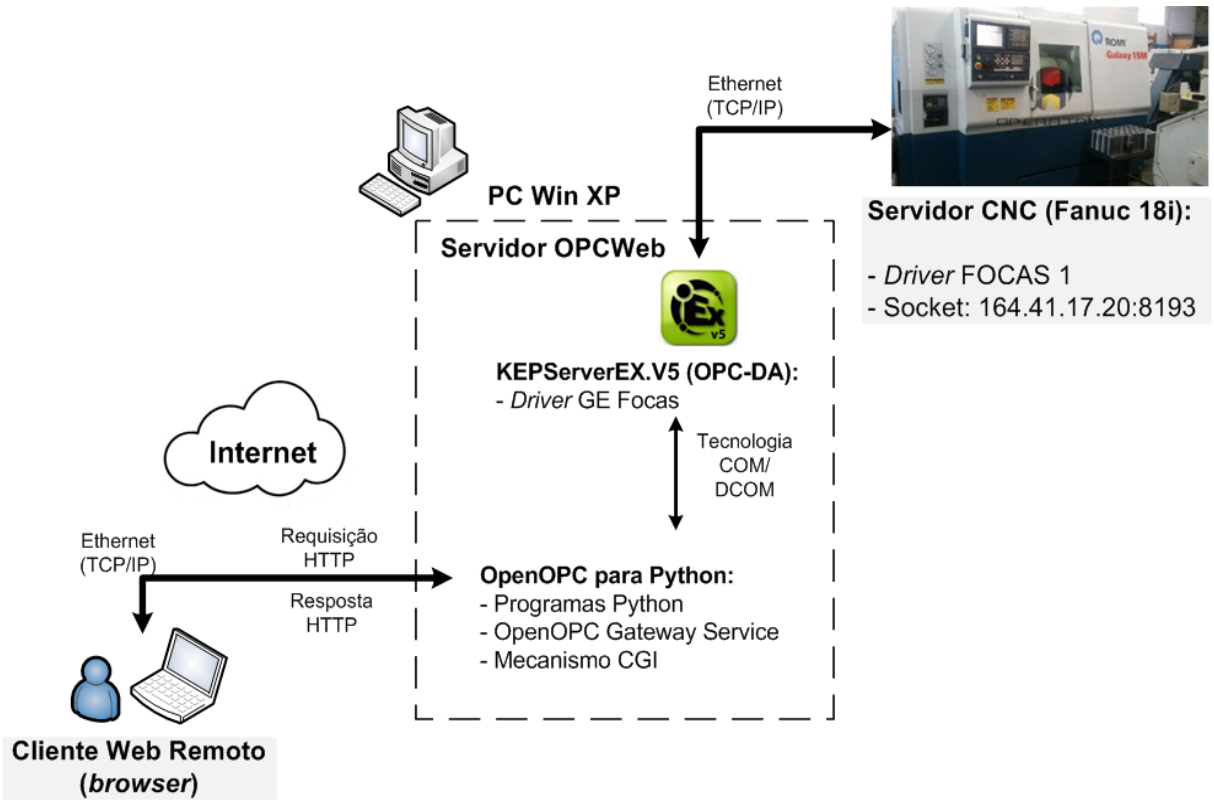

Figura 6.12: Servidor de supervisão OPC via internet (OPCWeb) com OpenOPC para python e protocolo CGI

este projeto para compor os serviços de teleoperação da implementação do *framework*. O serviço como um todo faz a captura de vídeo e áudio, e o envio de imagens e sons através da internet. Neste trabalho aproveitou-se a função de transmissão de vídeo. O sistema (Figura 6.13) foi implementado em plataforma Linux e é constituído por *Applets* Java, que monitoram a tela do CNC, o interior da máquina e o exterior. As imagens são obtidas por uma placa de captura de imagens e convertidas em para o formato JPEG, e depois enviadas ao cliente por *streaming*, via *socket* TCP/IP.

O cliente Web conecta o WebCam através *sockets* para a captura de imagens. É possível a captura de até dezesseis entradas de vídeo, conversão das imagens em formato JPEG, gravação dessas imagens, além de detecção de movimento. As imagens são transmitas para o cliente em um *streaming* de imagens JPEG (cerca de 30 *frames*/segundo para cada processador de imagem por placa) que são animadas no lado cliente através de *Applets* Java (ALVARES, 2005). O servidor WebCam estão disponível no endereço: *http://video.graco.unb.br*.

O sistema WebCam é integrado ao sistema deste projeto por meio de sua incorporação a interface web de monitoramento e teleoperação desenvolvida. As imagens das câmeras são disponibilizadas em *frames* HTML e são alternadas pelo clique de botões que efetuam requisições HTTP/GET utilizando *Javascript JQuery*. Nesta implementação foram disponibilizadas imagens das quatro câmeras da implementação original do servidor para o Webturning (ALVARES, 2005). O aumento no número de câmeras representaria um atualização do sistema WebCam, possibilidade exequível devido a característica modular desse sistema.

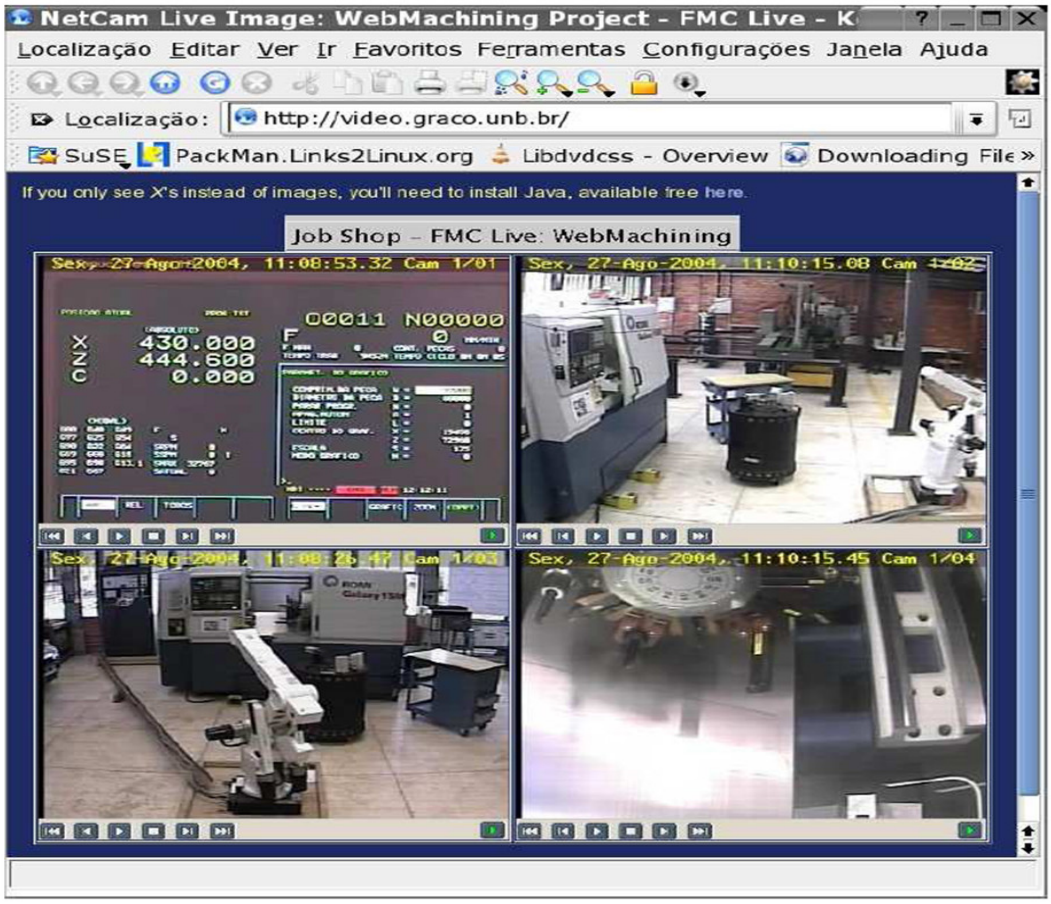

Figura 6.13: Sistema de monitoração por imagem WebCam (NetCam) (ALVARES, 2005)

O servidor WebCam mantem sistema implementado neste trabalho associado ao universo da aplicação de telemanufatura como um mecanismo de aumentar a imersão do operador no chão-de-fábrica.

6.5 GUI CyberDNC - Cliente Web de Integração

A interface Web cliente (Figura 6.14) assume a função de elemento integrador dos servidores implementados da arquitetura do sistema, agregando controles que fazem requisições e recebem respostas na forma de dados e informações através dos servidores MTConnect, OPCWeb, WebCam e WebCNC. Conforme foi citado em seções anteriores, parte dos servidores utilizados neste projeto foram herdados de trabalhos anteriores, como o WebCam e o WebCNC (ALVARES, 2005), que compõem os serviços de teleoperação, e aqueles que foram desenvolvidos no âmbito deste trabalho, como os servidores MTConnect e OPCWeb, para monitoramento e supervisão. Todos esses serviços são reunidos no cliente web desenvolvido para monitoramento e teleoperação através da internet utilizando um software *browser*. Esse cliente, que é apresentado na Figura 6.14, é formado pelos seguintes componentes:

(A) Botões representando teclas do painel de operações do CNC do centro de torneamento. Esses botões estão vinculados as requisições de escrita (HTTP/PUT) no servidor OPCWeb. As teclas representa-

das na página Web são: Auto, Edit, MDI, JOG, Single Block, Block Delete, Dry Run, Prog Test,-X, +X, -Z, +Z, Cycle Start, Cycle Stop, Coolant ON, Coolant OFF, Coolant Auto e Reset.

(B) Painel representado os LEDs das teclas do CNC e outros tipos de sinais, formado por imagens que alteram de cor, conforme o estado dos controles se alteram no CNC. Sinal verde representa uma tecla ativada, alarme acionado ou a porta do centro de torneamento fechada.

(C) Quadro para lançamento do *streaming* de dados resultante da comunicação com servidor MTConnect, e que é acionado através do clique na tecla *Start Streaming*.

(D) Opções de comando remoto DNC selecionáveis através de botões de radio.

(E) Área para seleção câmeras e exibição de imagens do processo provenientes do servidor WebCam.

(F) Painel para a exibição de mensagens ou alertas do CNC.

(G) Área para a exibição da lista de programas NC gravados na memória do controlador.

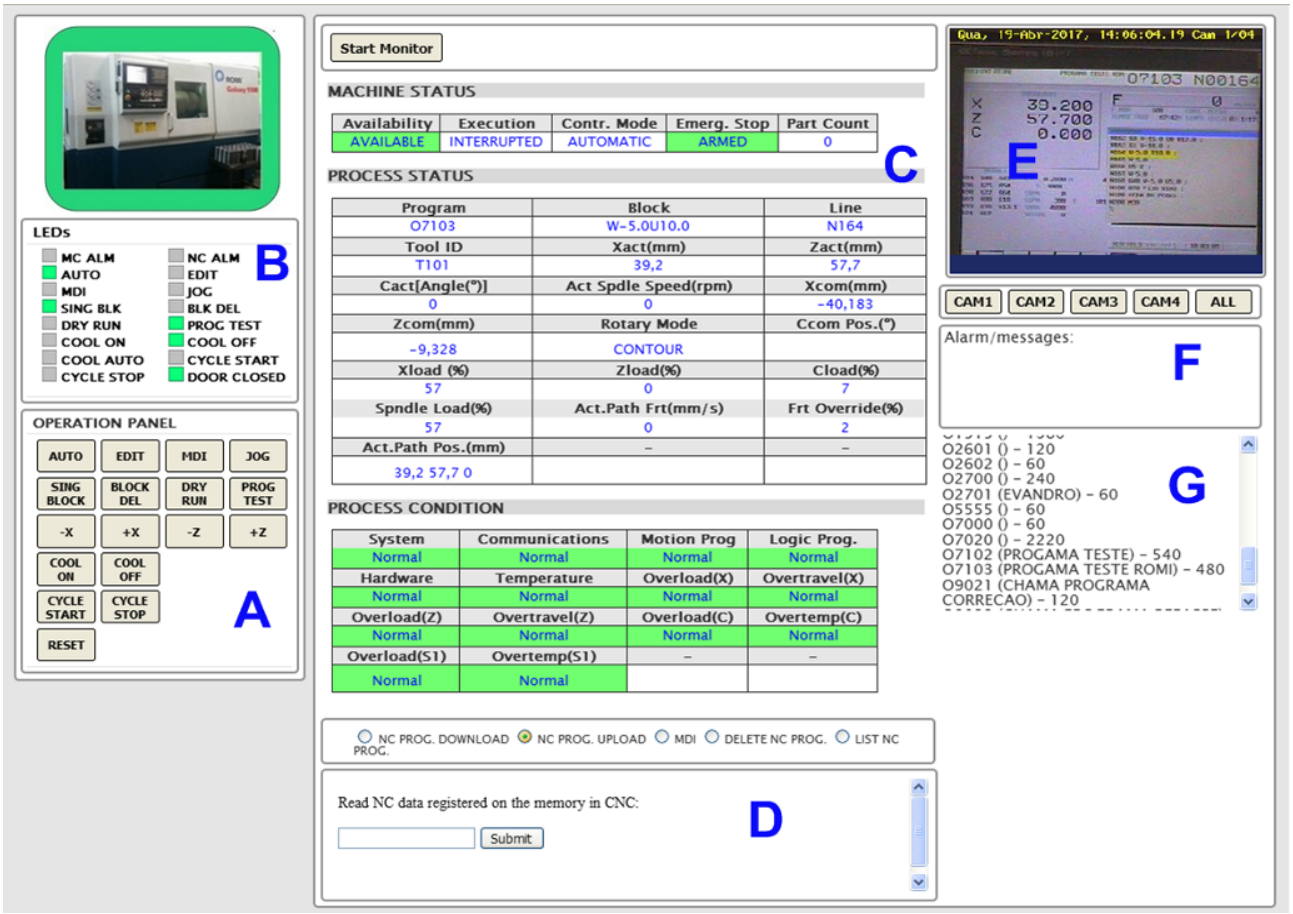

Figura 6.14: Cliente Web: Interface gráfica de monitoramento e teleoperação

A programação dessa GUI requereu o uso dos seguintes recursos: linguagens HTML, PHP, CSS, *scripts* CGI (programados em C++) e linguagem *Javascript* com *plugin jQuery* utilizando programação *Ajax*. Este último recurso permite que as requisições sejam realizadas aos servidores e informações sejam atualizadas

dinamicamente na aplicação Web sem a necessidade da página ser recarregada a cada nova requisição HTTP.

Os botões localizados no painel na parte esquerda da GUI (A) representam as teclas de acionamento dos controles do painel do CNC e utilizam programação *Ajax* para enviar requisições HTTP para o servidor OPCWeb com a finalidade de solicitar operações de atualização de parâmetros no servidor OPC, e assim atualizar o status dos parâmetros no controlador da máquina. A cada clique em uma tecla uma requisição é feita enviando um valor de parâmetro que altere o status de um controle para um dos estados binários: "0"(zero) e "1"(um), representando respectivamente *true* ou *false* no servidor OPC.

Um procedimento de requisição utilizando *jQuery.Ajax* também é utilizado para atualizar o status de cores do painel "LEDs"(B), destinado a supervisionar os controles das teclas ativas do painel de operações do controlador da máquina. Nesse caso é utilizado o método HTTP/GET para recuperar do servidor OPCWeb o valor das *tags* ("0"ou "1"ou, *false* ou *true*) que representam o estado dos LEDs das teclas do painel de operações do CNC. A taxa de atualização dessa função é de 1000ms (dois mil milisegundos), definido na programação do *script* de controle desse módulo na GUI.

O biblioteca *JQuery* é conhecida e utilizada por facilitar a programação em *JavaScript*, e essa versatilidade foi empregada no desenvolvimento da função para realizar varredura dos itens de dados (*DataItems*) provenientes do Agente MTConnect e transmitir os quadros localizados na área de principal da GUI (C) o *streaming* de dados com os parâmetros selecionados. Em um intervalo fixo o cliente web faz uma requisição ao Agente MTConnect dos dados disponíveis. Estes correspondem aos dados que sofreram alteração no CNC da máquina.

Logo abaixo do quadro de *streaming* MTConnect foram programadas com *JQuery.Ajax* as chamadas as páginas HTML implementadas para o envio de requisições HTTP ao servidor CNC utilizando mecanismo CGI (D). As páginas carregam valores de entrada que alimentam os parâmetros das funções de DNC incorporadas nos *scripts* CGI. Nesta seção foram implementadas quatro operações: *NC Program Download* (a máquina recebe um programa NC), *Upload NC program* (a máquina envia um programa NC), MDI, *Delete NC Program* e *List NC Programs*.

A operação de *Download* de um programa requer que o arquivo com o programa NC seja selecionado no sistema de diretórios do usuário. O programa é então enviado para a máquina CNC por intermédio de um servidor Web (WebDNC) que processa o CGI e envia o comando DNC para o CNC do centro de torneamento. Na operação de *Upload* o procedimento é inverso, o servidor CNC carrega o programa no cliente mediante requisição. Nesse caso o número do programa (expresso no nome do programa), que está salvo na memória do CNC, precisa ser informado na caixa de texto antes da submissão ao servidor CNC. Na opção de excluir (Delete) um programa NC também é solicitado o código do programa antes de requisitar a remoção ao servidor. O último botão tem a função de enviar um pedido para listar todos os programas gravados na memória do CNC, o uso desse botão gera como resultado uma lista de programas NC que exibida em uma área na parte inferior-direita da página (G).

A área à direita da GUI é formada pelo elemento contêiner(E) onde que são exibidos as transmissões de vídeo das quatro câmeras controladas pelo servidor WebCam. Subjacente estão as teclas para acionar as imagens de cada câmera individualmente (CAM1, CAM2, CAM3 e CAM4) e a opção "ALL"para abrir a página do servidor WebCam (*http://video.graco.unb.br*) onde todas as imagens das câmeras são exibidas

simultaneamente.

Abaixo dos botões para acionar as imagens das câmeras há um espaço (F) destinado a mostrar mensagens de alerta provenientes do CNC, mas que são distribuídas pelo serviço MTConnect. Em um posição subjacente e inferior a esse espaço de mensagens, há uma área(G) destinada a receber a lista de programa NC salvos na memoria do CNC que é exibida como resposta após a requisição HTTP/CGI efetuada com o clique no botão *LIST NC*.

Todos os serviços providos pelos servidores na arquitetura proposta ficam ativos e podem ser executados simultaneamente.

6.6 Análise da Implementação

O produto da metodologia de projeto axiomático (e projeto axiomático de software) juntamente com a modelagem UML gerada orientou a sequência de implementação e o desenvolvimento do sistema de monitoramento e teleoperação via internet, que recebeu CyberDNC. Para a implementação da arquitetura do *framework* foram utilizados padrões, tecnologias e conceitos relacionadas a Indústria 4.0 para um sistema com funções para manufatura eletrônica. O principal elemento de estudo para realização do trabalho prático foi o centro de torneamento da marca Romi model Galaxy 15M provido com CNC Fanuc 18iTa. Entre os servidores da arquitetura do sistema, uma parte foi herdada de um projeto anterior (ALVARES, 2005) e a outra foi desenvolvida para este trabalho, todos com a finalidade básica de efetuar a plena comunicação com a máquina-ferramenta remotamente através da internet.

O servidor de monitoramento implementado foi desenvolvido com a adoção de um protocolo candidato a padrão para manufatura da quarta revolução industrial, o MTConnect. Utilizou-se do benefício do livre acesso aos manuais dessa especificação e a disponibilidade de exemplos de desenvolvimentos e códigos fonte na internet para auxiliar na construção desse servidor, que consistiu na programação do Adaptador para o CNC Fanuc 18i e da instalação e configuração do Agente MTConnect pré-compilado que disponibiliza às aplicações cliente na Web dados de fabricação provenientes do CNC do centro de torneamento. O fato de a API e *driver* Focas 1 ser compatível com sistemas operacionais *Microsoft* até a versão Windows XP, restringiu o desenvolvimento do software Adaptador a esse ambiente. O software Agente foi desenvolvido para ser executado em sistemas *Windows* e o seu instalador foi compilado sob um conjunto de parâmetros básicos, isso obrigou a realização de procedimentos manuais de configuração para que o Agente disponibilizasse o *streaming* de dados de fabricação com a estrutura de dados do CNC Fanuc 18i-Ta.

Implementou-se um serviço de monitoramento e supervisão com o desenvolvimento do servidor OPCWeb. Este servidor possui um elemento *gateway* que possibilita o acesso a um servidor OPC-DA COM (*Component Object Model*) através da internet. Para desenvolvimento desse software *gateway* propôs-se duas abordagens possíveis uma envolvendo um *Web Service* RESTful associado a API JOPCClient programado em linguagem Java, na outra opção utilizou-se com programas escritos em linguagem Python associados a API OpenOPC para acesso ao servidor OPC através do mecanismo CGI com requisições HTTP. A solução utilizando *Web Service* resultou em uma maior complexidade na programação, devido própria robustez da linguagem Java com a API JAX-RS e necessidade de serialização de dados para transmissão ao cliente Web

a cada requisição HTTP de atualização. A alternativa utilizando *scripts* em python envolveu uma menor complexidade de programação, uma vez que o servidor OPC pode ser acessado e ter os valores dos seus itens alterados através de um número reduzido de linhas código.

As atividades de teleoperação implementadas integram o sistema ao contexto das ferramentas de telemanufatura. A incorporação do servidor de vídeo WebCAM (http://video.graco.unb.br) e o de comando remoto via internet, WebDNC, à arquitetura do sistema ocorreu com o desenvolvimento da interface cliente web de monitoramento e teleoperação do CyberDNC. As imagens das câmeras no servidor WebCam são alternadas e exibidas na GUI do CyberDNC através de requisições HTTP programadas em *Javascript JQuery*. As imagens de vídeo são carregas em um elemento contêiner do tipo *frame* do HTML. Para o servidor WebCNC foram utilizados botões do tipo *radio* que quando selecionados efetuam uma requisição HTTP/GET que carrega em um *frame* as páginas HTML com caixas de texto utilizadas na inclusão de parâmetros que compõem as entradas dos programas CGI que são executados no servidor Web. Os programas CGI carregam as funções de DNC (entrada manual, *download*, *upload*, exclusão e listagem de programas NC) que quando processadas efetuam comandos que são executados no controlador do centro de torneamento. A integração dos servidores de teleoperação não resultou de um trabalho com a complexidade que representou o desenvolvimento dos servidores de monitoramento/supervisão, pois são apenas requisitados em uma interface HTML através de simples métodos *JQuery* (*Javascript*).

Como um sistema de apoio a manufutura, o CyberDNC compõe o grupo das aplicações para a Indústria 4.0 devido as caracteristicas dos servidores que incopora, na foram de serviços, e pelo potencial de novos desenvolvimentos que sugere a partir de sua arquitetura. A capacidade de monitorar dados de um processo de fabricação transmitidos em um formato universal favorece uma maior integração vertical e horizontal em uma empresa de manufatura. Em futuras aplicações baseadas nessa arquitetura, dados capturados podem ser salvos em bancos de dados que possibilitarão que análises desses dados apoiem tomadas de decisões mais efetivas. O uso de teleoperação através da internet identifica a aplicação implementada com as aplicações de telepresença baseadas na nuvem citadas por Kagermann *et al.* (KAGERMANN; WAHLSTER; HELBIG, 2013). A arquitetura proposta e implementada mapeia virtualmente dispositivos de fabricação ao permitir que a máquina-ferramenta possa ser monitorada remotamente e a possiblidade de atuação na máquina. A facilidade de obter dados do status de produção em uma baixa resolução de tempo garante maior transparente do processo dentro da cadeia produtiva, isso está alinhado com potencial da IoT. Segundo Lins (LINS, 2015), a IoT utilizará o monitoramento e a possiblidade de operação remota para envolver terceiros (por exemplo, fornecedores) em novos serviços para a manufatura.

Capítulo 7

Testes e Validação da Arquitetura do *Framework*

Os testes foram conduzidos com base na avaliação de funcionamento através de casos de uso. Estes incluem aqueles definidos durante a modelagem UML na metodologia (Figura 5.22) através do cliente web de monitoramento e teleoperação implementado. Esse cliente é avaliado durante todas as etapas de teste. O procedimento de teste e validação adotado neste trabalho foca na demonstração de funcionamento do sistema com base em casos de uso, tendo como referência na comunidade acadêmica trabalhos relacionados ao desenvolvimento de metodologias no âmbito da manufatura eletrônica (ALVARES, 2005; BENAVENTE, 2007; SOUZA-JÚNIOR, 2008; BENAVENTE, 2011).

Os casos de uso do sistema permitem dividir os testes com base na avaliação por servidor, iniciando pelo servidor de monitoramento MTConnect, em que é verificada a capacidade do Agente em conectar-se com diferentes clientes Web baseados em PC, através de *browser* ou aplicação *desktop*, e aplicação para dispositivo móvel Android. Nessa etapa o CNC da máquina-ferramenta e ajustado para execução simulada (controle *Prog Test*) e modo automático, executando linha-por-linha um programa NC (Apêndice F) selecionado.

As outras fases de teste coincidem com a avaliação dos outros servidores, como o servidor OPC para Web (OPCWeb), e os servidores WebCNC e WebCam de teleoperação. Todos esses servidores foram avaliados apenas com base no funcionamento do cliente web do CyberDNC, que foi desenvolvido no âmbito deste trabalho. O WebCNC foi avaliados com o CNC da máquina-ferramenta no modo de entrada manual de dados (MDI), momento em que foram testados outros casos de uso (Figura 5.22) implementados, como: *download*, *upload*, exclusão e listagem de programa NC, e envio de comandos por entrada manual. Avaliou-se o funcionamento dos servidores OPCWeb e WebCam durante todo o fluxo de testes, pois ficaram ativos durante os testes dos outros dois servidores.

Logo abaixo, na Figura 7.1, é apresentada a FMC (Célula Flexível de Manufatura, traduzido do inglês) do GRACO/UNB (*http://www.graco.unb.br*) com o centro de torneamento Galaxy 15M, CNC Fanuc 18i-Ta (unidade de processamento), que é o elemento físico fundamental para a condução do trabalho de projeto, desenvolvimento, teste e validação da proposta deste trabalho.

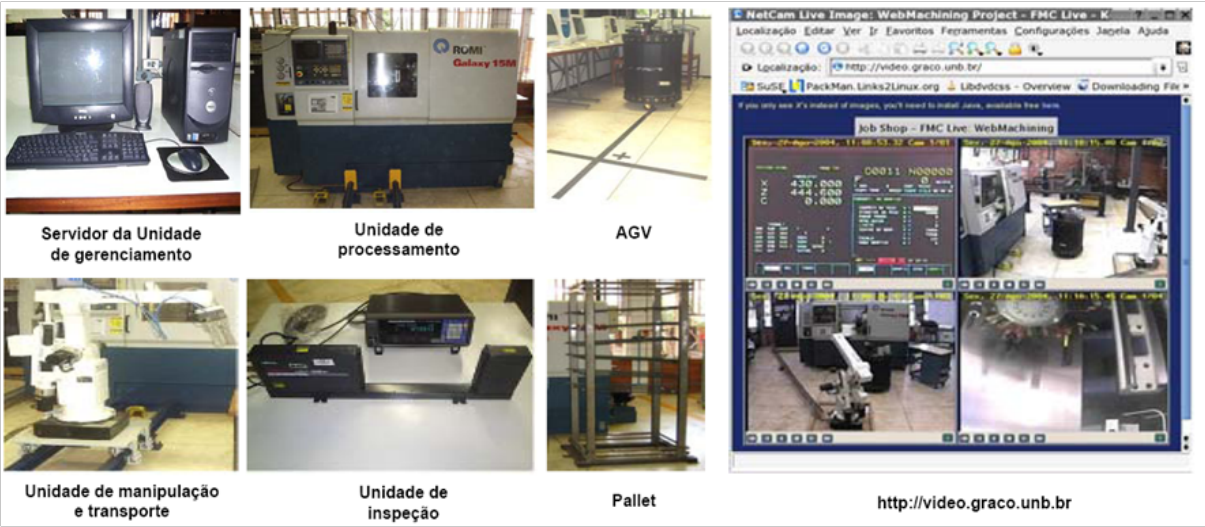

Figura 7.1: FMC com centro de torneamento Romi Galaxy 15 e CNC Fanuc 18i-Ta (TEIXEIRA, 2006)

A FMC da Figura 7.1 também é composta por uma unidade de manipulação e transporte de materiais (manipulador robótico), uma unidade de inspeção (micrômetro laser), uma unidade de armazenamento de peças (*pallet*), um AGV e por um sistema de controle (Unidade de Gerenciamento).

O propósito básico de todo o trabalho de teste é a comunicação com o controlador (Fanuc 18i-Ta) do centro de torneamento.

7.1 Servidor MTConnect: Monitoramento

A avaliação do serviço de monitoramento MTConnect procedeu através de testes de conectividade, inicialmente entre o Adaptador MTConnect e o CNC da Romi Galaxy 15M, sucedido pela demonstração da comunicação entre o Agente e diferentes clientes Web.

A comunicação entre Adaptador e o controlador Fanuc 18i é verificada através de um procedimento trivial. Com o CNC da máquina-ferramenta ligado e o Adaptador MTConnect inicializado, abriu-se o *prompt* de comandos do Windows XP em modo administrador para que a seguinte linha fosse digitada e executada:

C:\ WINDOWS \ system32> telnet localhost 7878

O protocolo de rede TELNET foi utilizado para acessar a porta 7878 do computador, em que o Adaptador transmite um *streaming* de dados no formato SHDR(*Simple Hierarchical Data Representation*), após da conexão com o CNC. Através da demonstração apresentada na Figura 7.2 foi possível constatar que o Adaptador é funcional. A conexão entre o software Adaptador e o CNC é possível graças a instalação e configuração da API Focas 1 associada ao *driver fwlib32.dll*, que deve ser instalado no diretório *system32* do sistema operacional Windows.

Com os parâmetros do Adaptador definidos para disponibilizar o *streaming* de dados para o Agente no computador local na porta 7878, e introduzindo que o dispositivo (CNC) está localizado no *socket* com IP

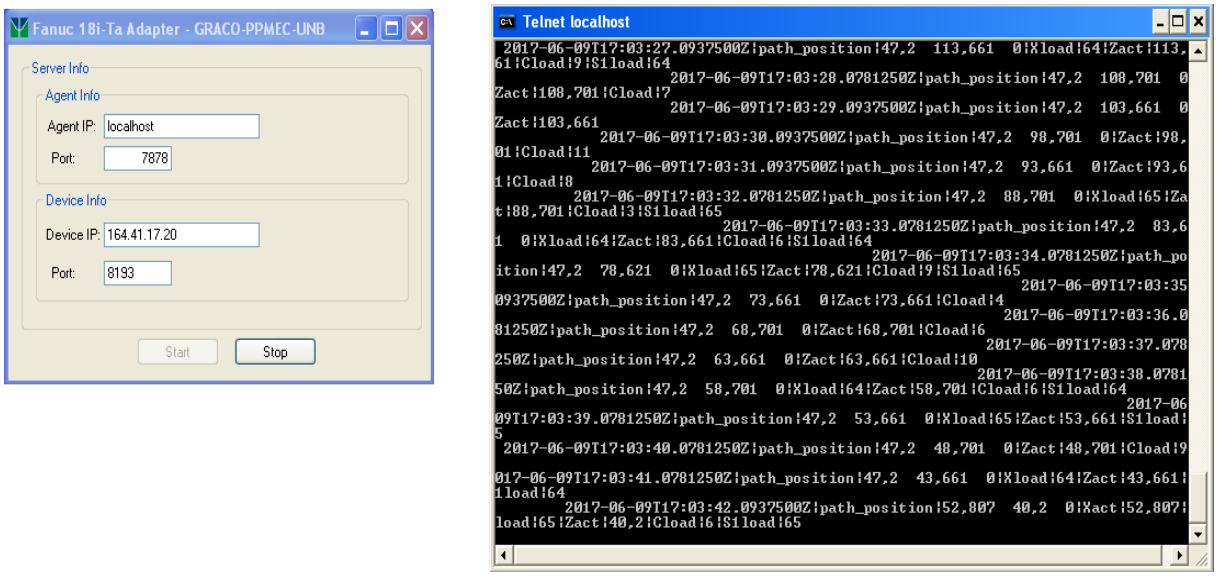

Figura 7.2: Dados de saída do Adaptador: conexão entre o CNC e o Adaptador

164.41.17.20 e porta 8193, o comando *telnet* dá acesso a transmissão de dados de forma instantânea.

O teste de monitoramento principal ocorre por meio da conexão entre o Agente MTConnect e diferentes clientes com acesso a Internet. Para isso, alguns clientes disponíveis na Internet com licenças de uso livres e a GUI de monitoramento e teleoperação desenvolvida no âmbito trabalho foram selecionados como elementos de análise dos serviços de monitoramento. Esses testes foram realizados após a seleção de um programa NC (Apêndice F) localizado na memória do CNC, para execução em modo teste (tecla *Prog Test* acionada). Abaixo são listados cada caso e os resultados obtidos:

a) MTConnect Monitor - Interface para Browser (*https://github.com/pmcoltrane/MTConnect-JS*)

Este é um cliente MTConnect criado por Phil Coltrane (https://pmcoltrane.wordpress.com/) e foi desenvolvido essencialmente em linguagem *JavaScript* com *plugins* JQuery. O procedimento de funcionamento da aplicação é simples. Em uma caixa de texto intitulada *Agent* o usuário digita o endereço URL do Agente, em uma segunda caixa de texto chamada "Proxy"é automaticamente introduzida a localização do arquivo *proxy*, que é usado na programação Ajax (*Asynchronous Javascript and XML*) desse cliente para possibilitar que arquivos XML localizados em um domínio fora do computador local possam ser requisitados e recebidos como resposta. Outro parâmetro que pode ser alterado é o intervalo (*Interval*) de atualização que é definido em segundos. Com os parâmetros introduzidos, o clique no botão "Monitor"inicia transmissão de dados de fabricação. A Figura 7.3 ilustra o resultado obtido no teste com o Agente MTConnect para a máquina Romi Galaxy 15M e cliente MTConnect.

A inicialização do *streaming* de dados nessa aplicação gera dinamicamente um conjunto de elementos contêineres alinhados horizontalmente onde são dispostos os nomes dos parâmetros com os respectivos valores, que são atualizados com base no intervalo de requisição. Semelhantemente a um sistema sinótico, as cores dos elementos são atualizadas com base nos valores dos itens de dados e condições. Parâmetros com valores indisponíveis (*unavailable*) por padrão ganham a cor cinza

escuro, valores numéricos disponíveis adquirem um tom cinza claro, parâmetros de condição (*Condition*) com status Normal recebem a cor verde, em caso de alarme no CNC a condição associada ao mesmo muda para a cor amarela, indicando o status *Warning*. Como esses testes foram realizados com a opção *Single Block* ativada, ou seja, o programa NC é executado linha por linha, necessitando de um comando de execução (*Cycle Start*) a cada nova linha, é possível visualizar os diferentes status de cores com base no estado de execução do CNC (*Execution*): *Active*, *Ready*, *Stopped* ou *Interrupted*.

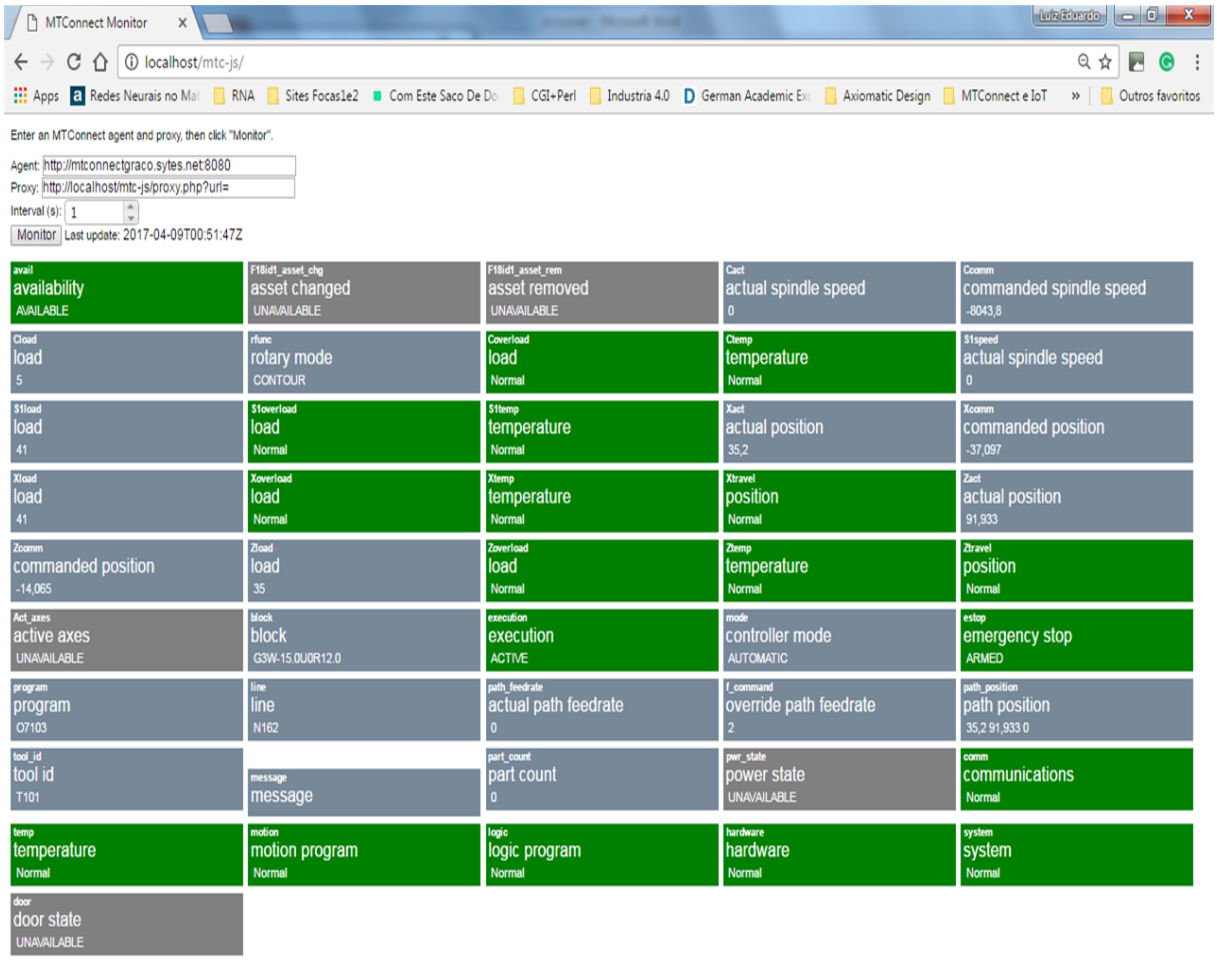

Figura 7.3: Cliente Web: MTConnect Monitor (*https://github.com/pmcoltrane/MTConnect-JS*)

b) Cliente GUI de Monitoramento e Teleoperação (*http://cyberdnc.alvarestech.com*)

Este cliente foi desenvolvido durante a implementação computacional da arquitetura do proposta neste trabalho. É acessível via *browser* e o funcionamento do seu serviço de monitoramento assemelha-se ao do cliente descrito anteriormente.

Nessa aplicação a configurações de URL do Agente são definidas internamente, sendo um cliente específico para monitoramento do centro de torneamento Romi Galaxy 15M com CNC Fanuc 18i-Ta.

Em um primeiro momento o funcionamento desse cliente MTConnect foi avaliado com o CNC ativo, em modo automático para o processamento do programa selecionado (Apêndice F). As teclas *Single*

block e *Prog Test* foram mantidas ativas, a fim de que programa fosse executado, respectivamente, em uma linha por vez e sem movimentação dos eixos.

Após a inicialização do *streaming*, os dados foram recebidos com exatidão. Os parâmetros selecionados para monitoramento foram projetados nos quadros da área principal da GUI e correspondiam aos dados monitorados através das imagens de vídeo transmitidas pelas câmeras instaladas sobre a tela do CNC. Uma vista desse cliente Web recebendo dados do CNC do centro de torneamento pode ser conferida na Figura 7.4.

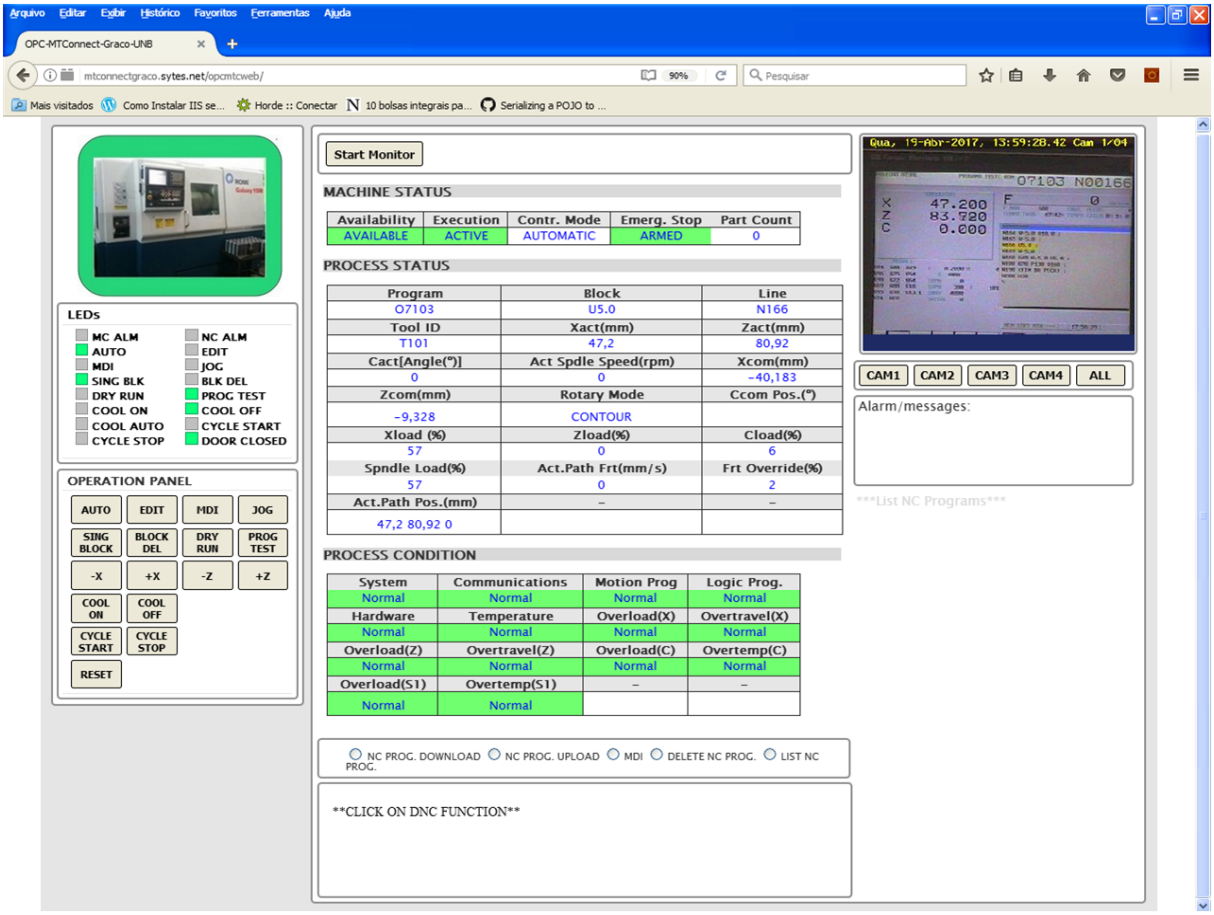

Figura 7.4: Comunicação entre o servidor MTConnect e a GUI de teleoperação e monitoramento com o CNC em *Controller Mode* automático

Uma segundo parte dos testes com esse cliente foi realizada com o centro de torneamento em modo manual (*Jog*), a fim de que através da movimentação dos eixos da máquina manualmente um alarme fosse intencionalmente provocado, para que a transmissão de parâmetros de condição (*Process Condition*) da máquina pudesse ser testada. O resultado é apresentado na Figura 7.5. É possível perceber que com a execução do programa NC parado, o parâmetro *Controller Mode* foi atualizado para *Manual*, e o parâmetro *Rotary Mode* foi atualizado para *Spindle*.

Movimentando o eixo linear *Z* até o fim de curso com auxilio das teclas direcionais, o CNC produziu um alerta de sistema, pois para o controlador não havia eixos associados ao alarme, conforme pode ser visualizado na Figura 7.5. Por uma questão de segurança, não tentou-se avaliar outros tipos con-

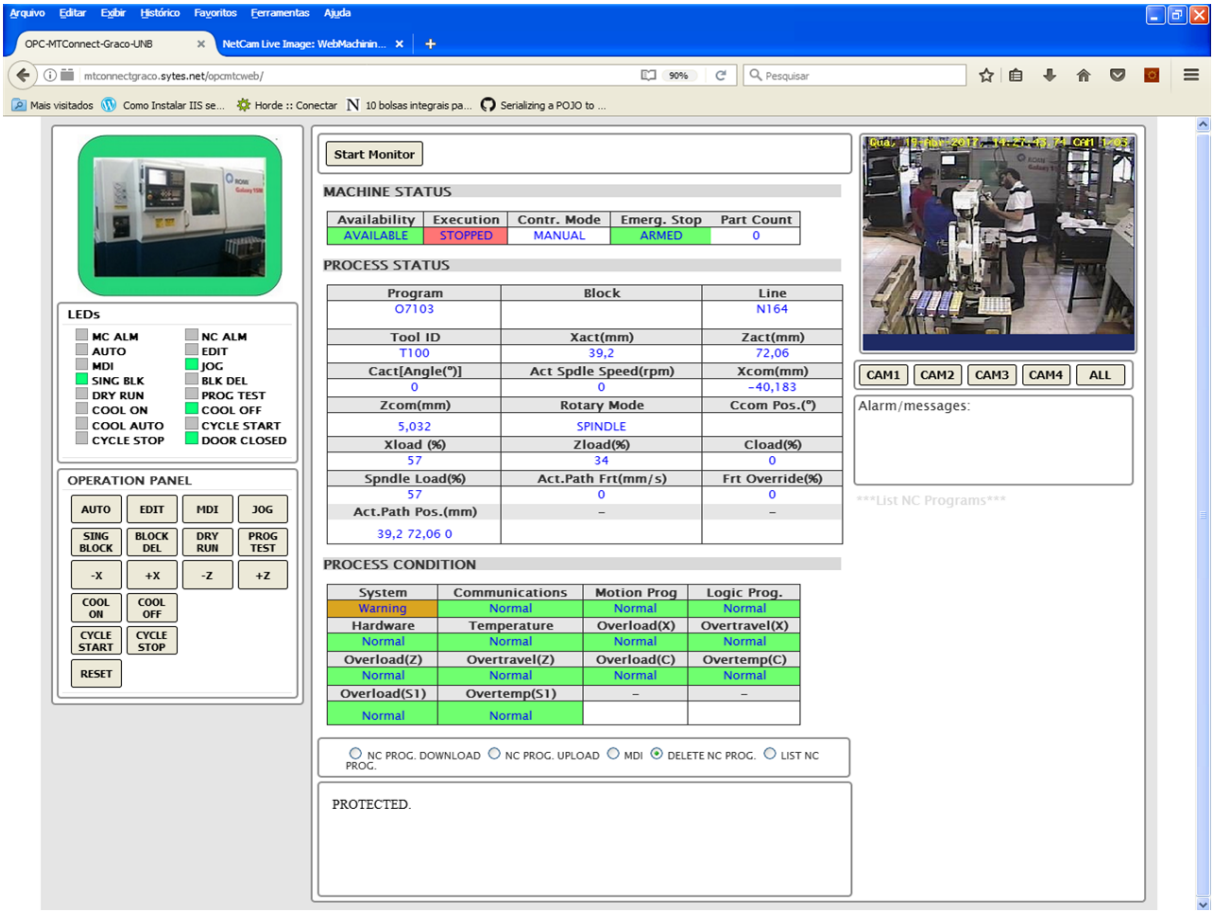

Figura 7.5: Teste da transmissão dos parâmetros de condição (*Condition*) pelo servidor MTConnect

dições e também não foram identificadas novas fontes de alarme. Apesar representar uma avaliação relativamente simplificada, o resultado pode ser considerado representativo, pois no código fonte do Adaptador há uma classe exclusiva para o tratamento de condições, que foi repetidamente depurada e até o momento desse teste não foram identificados erros de codificação relevantes.

c) Cliente GRIMA/UFSC - MTConnect Project (*http://www.grima.ufsc.br/mtc_project*)

Esta aplicação é parte projeto *MTConnect Client Application Project* , e foi desenvolvido âmbito das atividades do GRIMA (*Group of Manufacturing Integration*), o Grupo de Pesquisa em Integração Da Manufatura, do Departamento de Engenharia Mecânica da Universidade Federal de Santa Catarina. O cliente foi desenvolvido em linguagem Java e agrega funções de monitoramento e supervisão. Neste cliente sucessivos testes de conecxão foram efetuados. Verficou-se uma que o servidor dessa aplicação apresenta restrição de acesso à rede e os IPs utilizado nos testes iniciais. Uma mudança de IP permitiu a conexão e a execução do *applet* java tanto via *browser* quanto no computador local.

A Figura 7.6 apresenta o início dos testes após a inclusão dos parâmetros de entrada para conexão e o carregamento da interface principal de usuário. Em seguida, foi adicionado o Agente MTConnect do Cyberdnc (*http://cyberdnc.alvarestech.com:5000*) que no momento dos testes não possuía um domínio próprio e foi testado no *host* local na porta 5000 (*http://164.41.45.17:5000*).

Após a introdução de simples parâmetros de entrada, foi aberta interface inicial (Figura 7.6). É

importante destacar que para acessar a aplicação é necessário a instalação da versão mais atual da Java JRE(*Java Runtime Environment*).

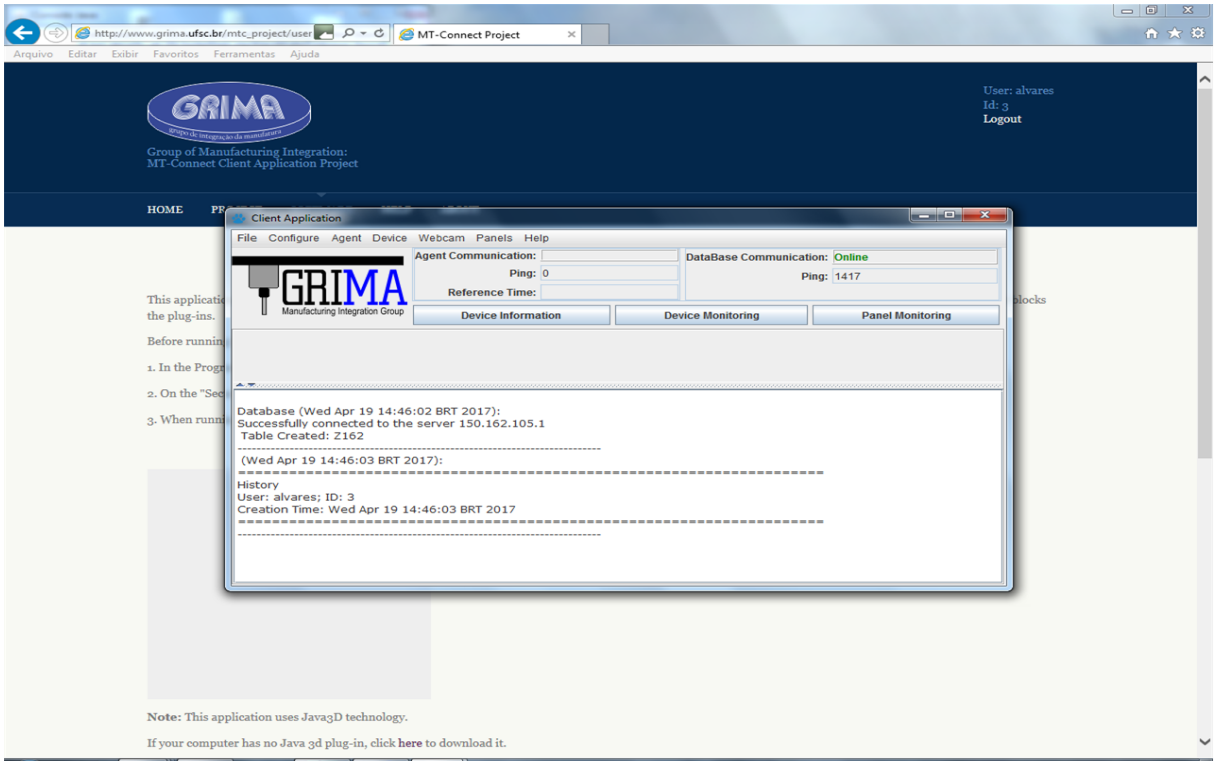

Figura 7.6: Cliente MTConnect Grima: conexão com a aplicação

A Figura 7.7 mostra o carregamento automático dos valores do parâmetros disponibilizados pelo Agente MTConnect e o início do monitoramento. A partir desse *streaming* de dados foram selecionados os parâmetros de velocidade do eixos rotacional (*Spindle Speed*) e a carga desse eixo (*Cload*) para serem projetados dois gráficos de linha, conforme ilustra a Figura 7.8. Esse recurso tem um potencial muito grande para o desenvolvimento de trabalhos futuros envolvendo a análise gráfica desses tipos de parâmetros de fabricação.

d) Aplicação *Android* - GTMTC-Lite

O GTMTC-Lite é uma aplicação para dispositivo móvel *Android* que possui como atributo capturar a velocidade do eixo rotacional (*Spindle Speed*).

Nesta aplicação para iniciar a transmissão de dados é necessário incluir a URL e clicar no botão com a descrição do Agente adicionado (Figura 7.9), o *streaming* de dados começa automaticamente.

Com auxílio dos controles de supervisão e comando remoto DNC da GUI CyberDNC os parâmetro de velocidade rotacional foi ajustado para três velocidades: 1000, 2000 e 3000 rpm. O resultado desses comandos foram transmitidos para a aplicação com uma resolução de tempo visualmente baixa. A Figura 7.10 exibe o registro de quando a velocidade nominal do *spindle* foi alterada para 2000 rpm.

É importante destacar que durante o teste para comunicação com o Agente MTConnect foram efetua-

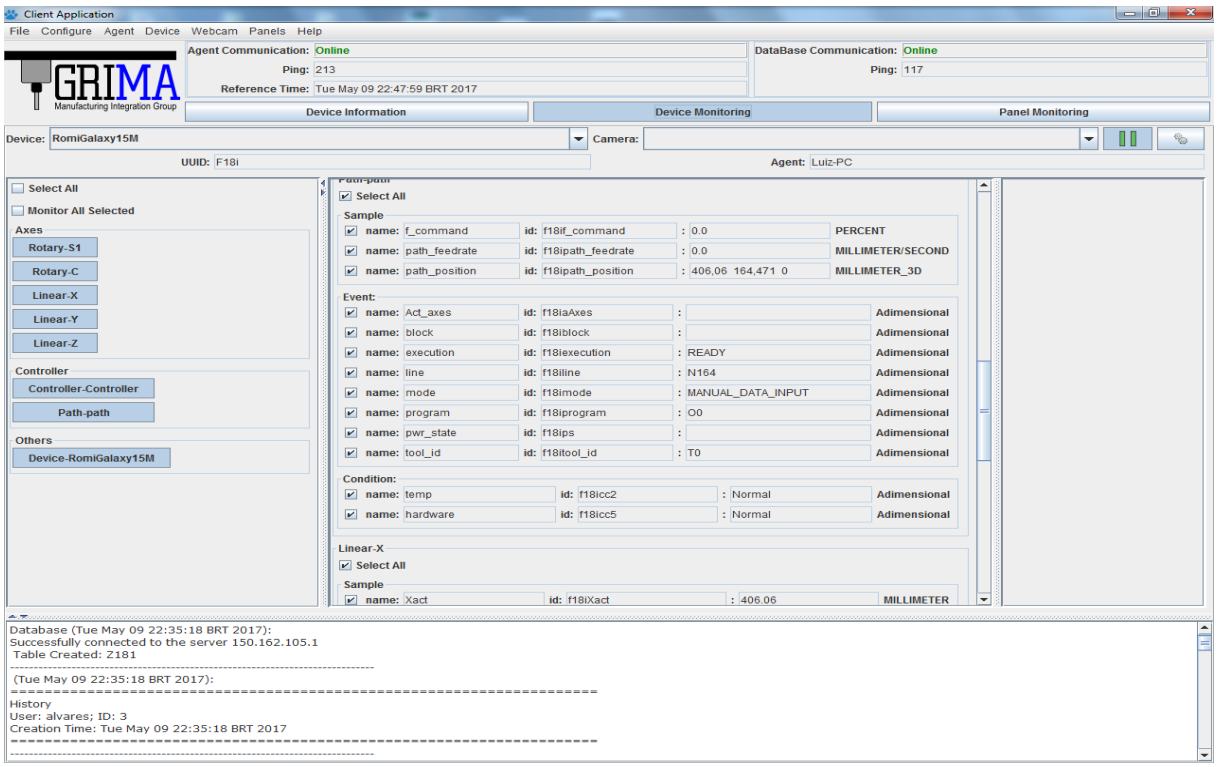

Figura 7.7: Cliente MTConnect Grima: carregamento de dados após inclusão do Agente

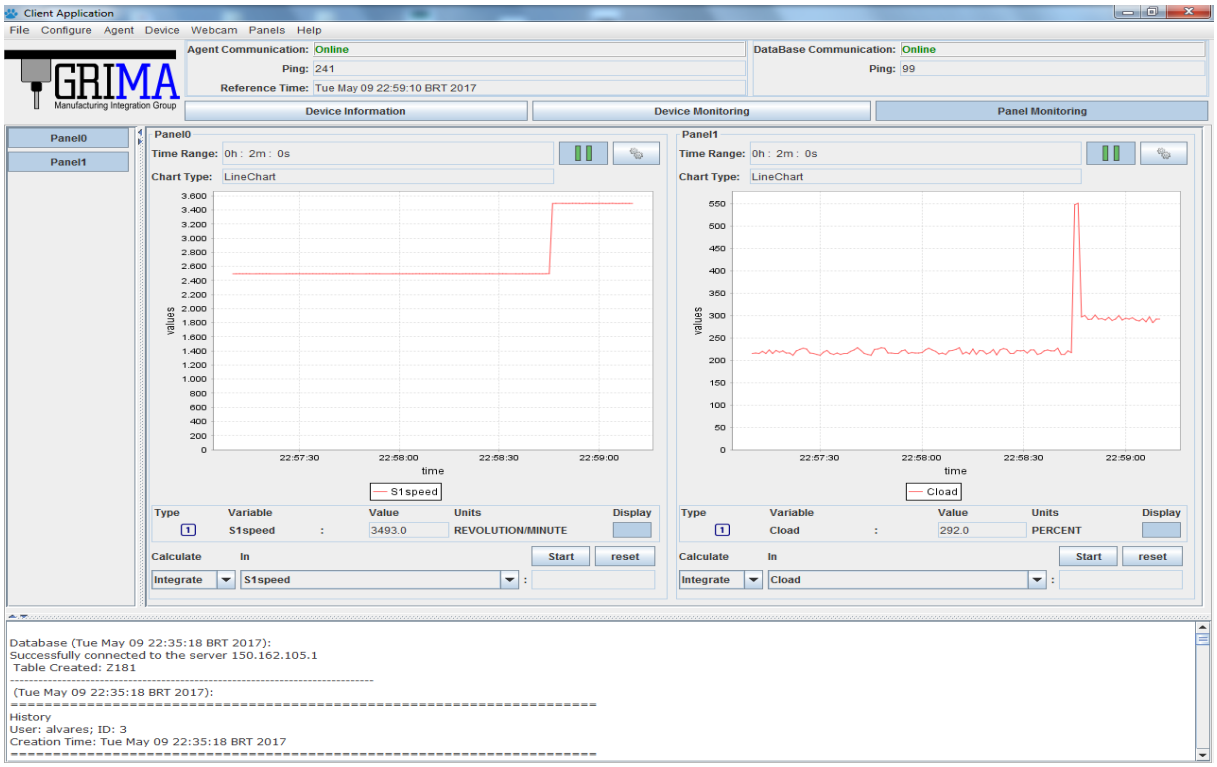

Figura 7.8: Cliente MTConnect Grima: projeção de gráficos de linha

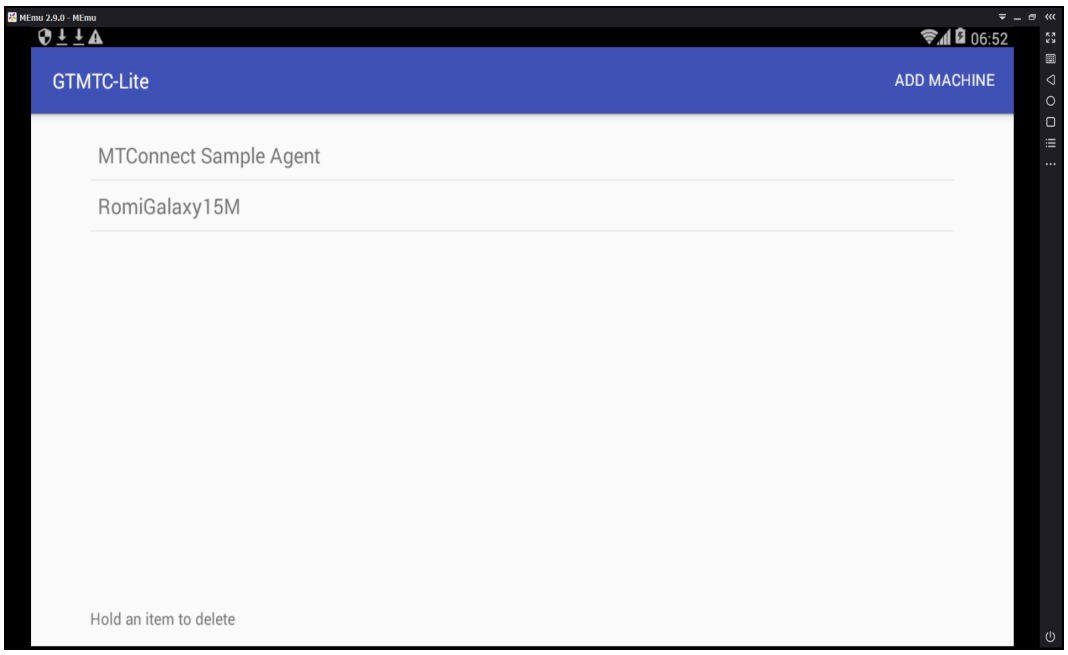

Figura 7.9: Cliente *mobile* GTMTC-Lite: Inclusão de Agente

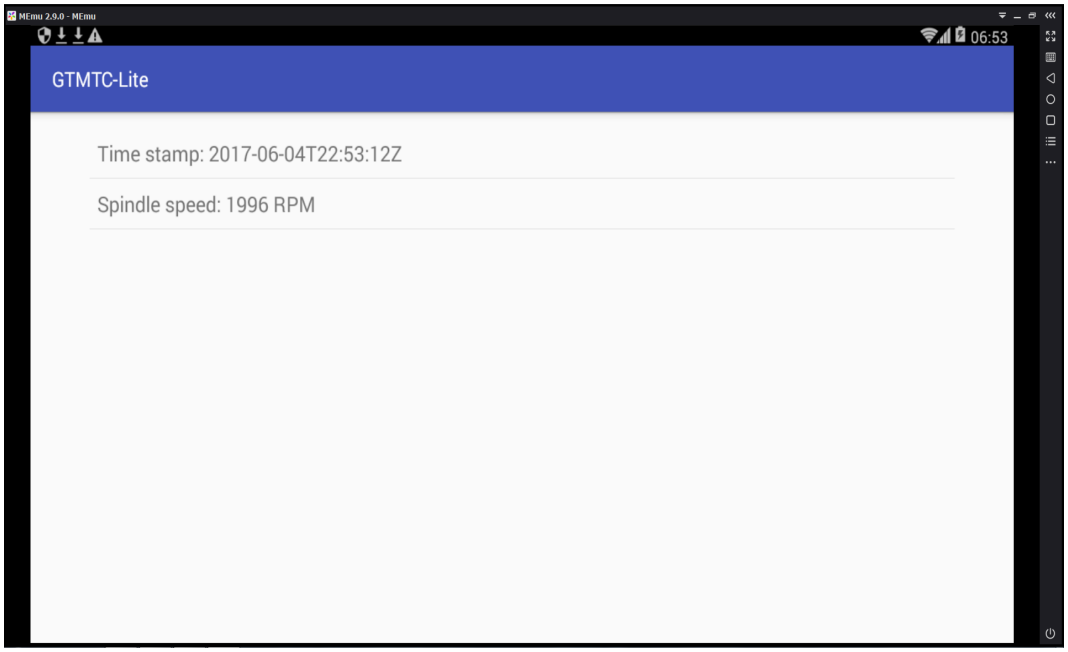

Figura 7.10: Cliente *mobile* GTMTC-Lite: *streaming* de dados da velocidade do *spindle*

das conexões simultâneas de dois ou mais desses clientes. Neste trabalho não foi avaliado a consequência do acesso simultâneo no desempenho final do serviço de monitoramento. No entanto, é importante citar que sob as condições em que esses testes foram feitos não foi possível assegurar a existência de um monitoramento em tempo real, pois não foram realizados testes com esse foco e a conexão com a internet disponível para teste possui uma largura de banda de 100 Mbps (cem megabits por segundo), velocidade abaixo dos atuais serviços de conexão com a internet disponíveis no mundo.

7.2 Servidor OPCWeb: Supervisão

Um pré-teste realizado durante a implementação desse servidor utilizando Web Service RESTful como componente *gateway* apresentou falhas no funcionamento desse *middleware*. Exceções geradas no servidor Tomcat revelou problemas na execução de classes relacionadas a API JAX-RS. Obteve-se sucesso na conexão inicial com servidor OPC-DA através da API JOPCClient, o que não foi possível constatar nas tentativas de interação com o servidor OPC através de requisições de leitura e escrita. Para esse pré-teste foi utilizado o *plugin* para navegador Mozilla Firefox chamado RESTClient. A implementação bem-sucedida do servidor OPCWeb através dessa alternativa requererá um maior esforço de análise sobre o funcionamento da API JAX-RS com Jersey e da API JOPCClient, para que novos testes possam se conduzidos e está opção possa ser validade. É importante destacar que interfaces RESTful já foram empregadas de forma bem-sucedida como extensão do OPC-UA (GRÜNER; PFROMMER; PALM, 2015; GRÜNER; PFROMMER; PALM, 2016).

AS dificuldade inerentes a implementação do servidor de supervisão OPCWeb utilizando *Web Service* abriu espaço para os testes com a implementação alternativa utilizando pequenos programas python associados a API OpenOPC (Apêndice G), que são executados a partir de requisições HTTP utilizando mecanismo CGI. Com essa opção o servidor OPCWeb foi utilizado como serviço de apoio tanto durante os testes do servidor MTConnect quanto nos teste do servidor WebCNC que será apresentado na próxima seção. Nesses testes os controles relacionados a atividade de supervisão funcionaram adequadamente, tanto na leitura quanto na atualização de parâmetros de CLP mapeados através das *tags* do servidor OPC-DA.

7.3 Servidores de Teleoperação: WebCNC e WebCam

A única opção de cliente disponível para validação desse serviço foi aquele implementado neste trabalho. O teste consistiu em comunicar-se com o servidor CNC e executar a operação de DNC requisitada. Lembrando que as opções de comandos DNC implementadas foram: *Download* de Programa NC, *Upload* de Programa NC, comando MDI (entrada manual de dados), Deletar programa e listar programas NC.

a) Listar Programas NC

Nesta opção envia-se a requisição HTTP/CGI com um simples clique no botão, conforme ilustra Figura 7.11. O resultado é apresentado na parte inferior direita da GUI.

b) *Download* de programa NC

O procedimento de teste é trivial. Na GUI, selecionou-se o botão "NC Program Download"e no formulário que foi aberto no *frame* na parte inferior da página selecionou-se a opção para carregar um arquivo. A Figura 7.12 apresenta a utilização dessa função.

c) *Upload* de Programa

Esta função trabalha com a introdução do nome programa na caixa de texto do formulário HTML, representado pelo número que consta em seu cabeçalho, e submete a requisição. O resultado aparece na própria página, conforme ilustra a Figura7.13.

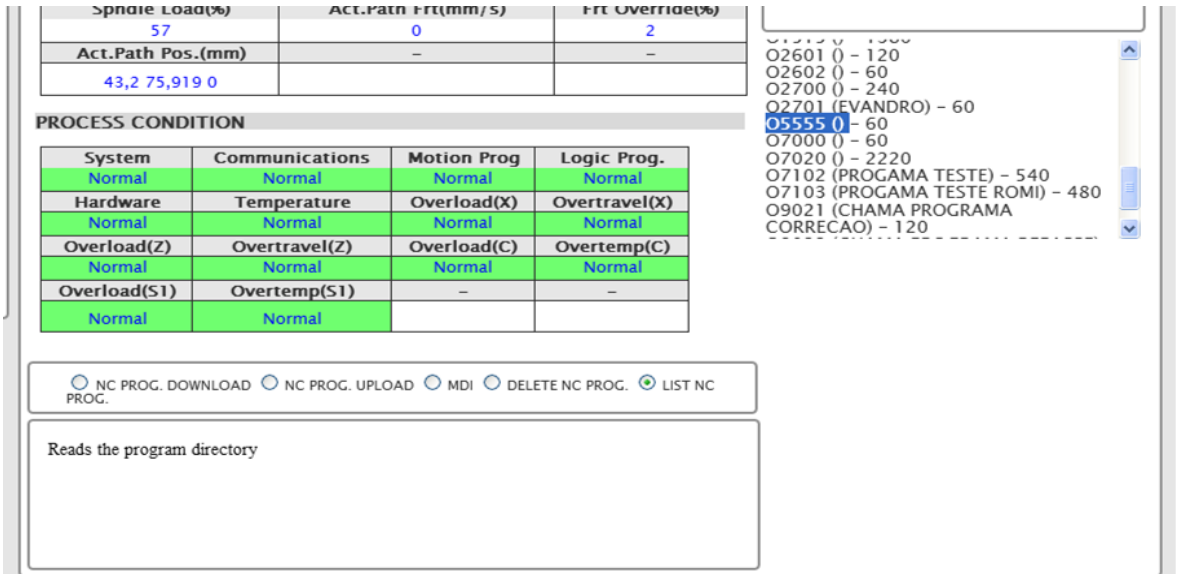

Figura 7.11: Listando programas NC salvos na memória do CNC

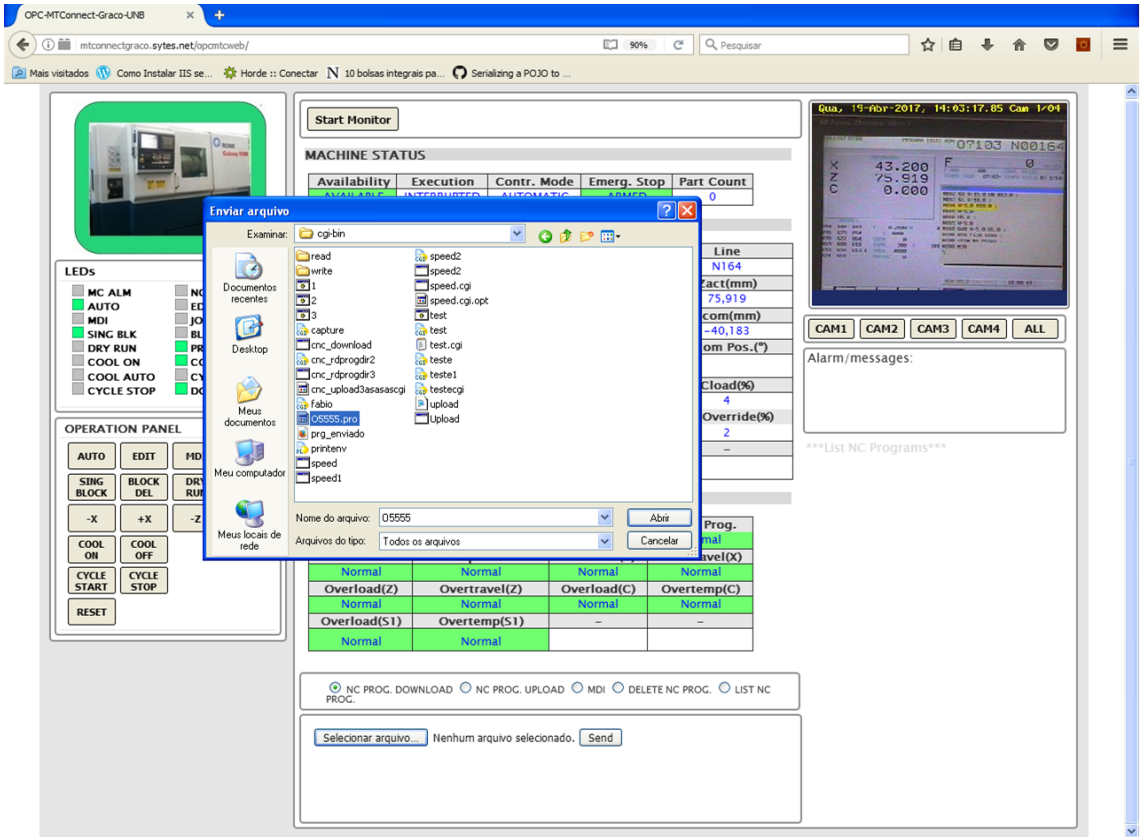

Figura 7.12: Função de *download* de programa NC no CNC

d) MDI

A opção de comando manual é aberta e na área de texto foi digitado um simples comando "S1000 M3"que tem a função de acionar o eixo rotacional principal do centro de torneamento a 1000 rpm

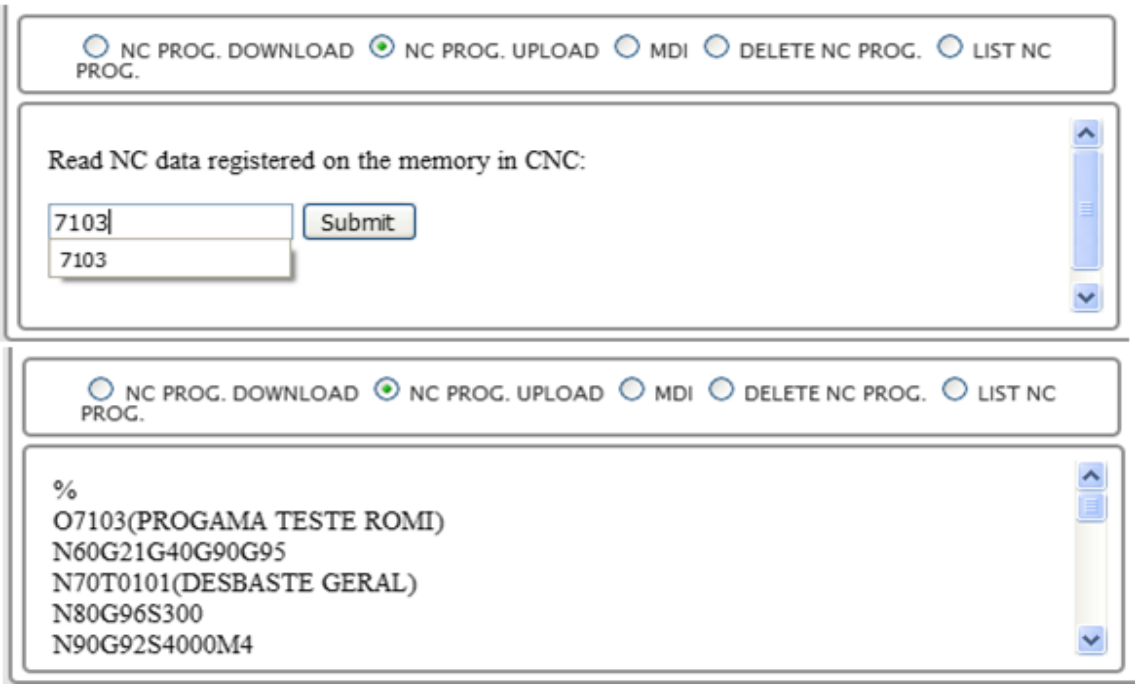

Figura 7.13: Função de *upload* de programa NC do CNC

no sentido horário. Nesse procedimento foram utilizadas as imagens da câmera em frente ao *display* do CNC, que faz parte do serviço WebCam, para a confirmação do envio do comando. Esses passos são demonstrados nas Figuras 7.14 e 7.15.

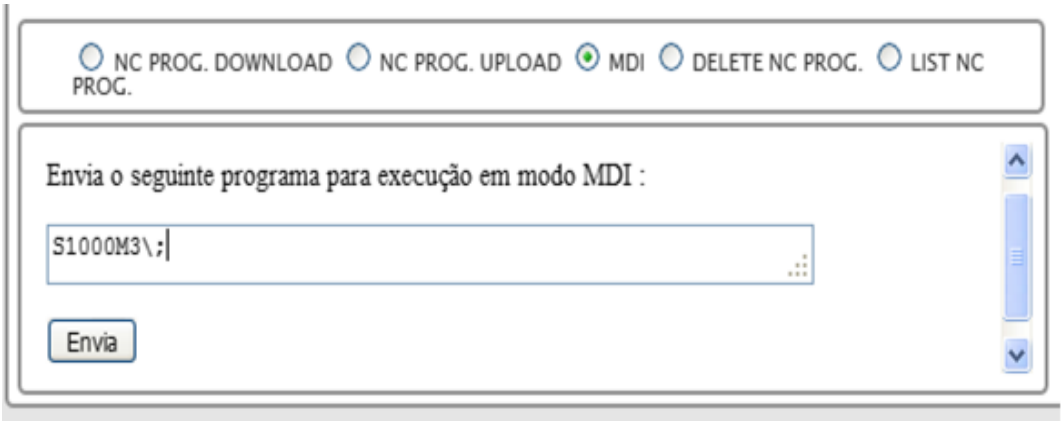

Figura 7.14: Escrevendo Comando Manual (MDI)

e) Deletar Programa NC

Este comando funcionou adequadamente, porém o CNC possui uma restrição à exclusão de programas NC através dos comandos DNC, conforme evidenciam as Figuras 7.16.

Vários casos de teste utilizaram o serviço de câmeras (WebCam), de forma que sua importância é essencial nas atividades de teleoperação (Figura 7.17) como mecanismo de realimentação para as atividades de comando remoto.

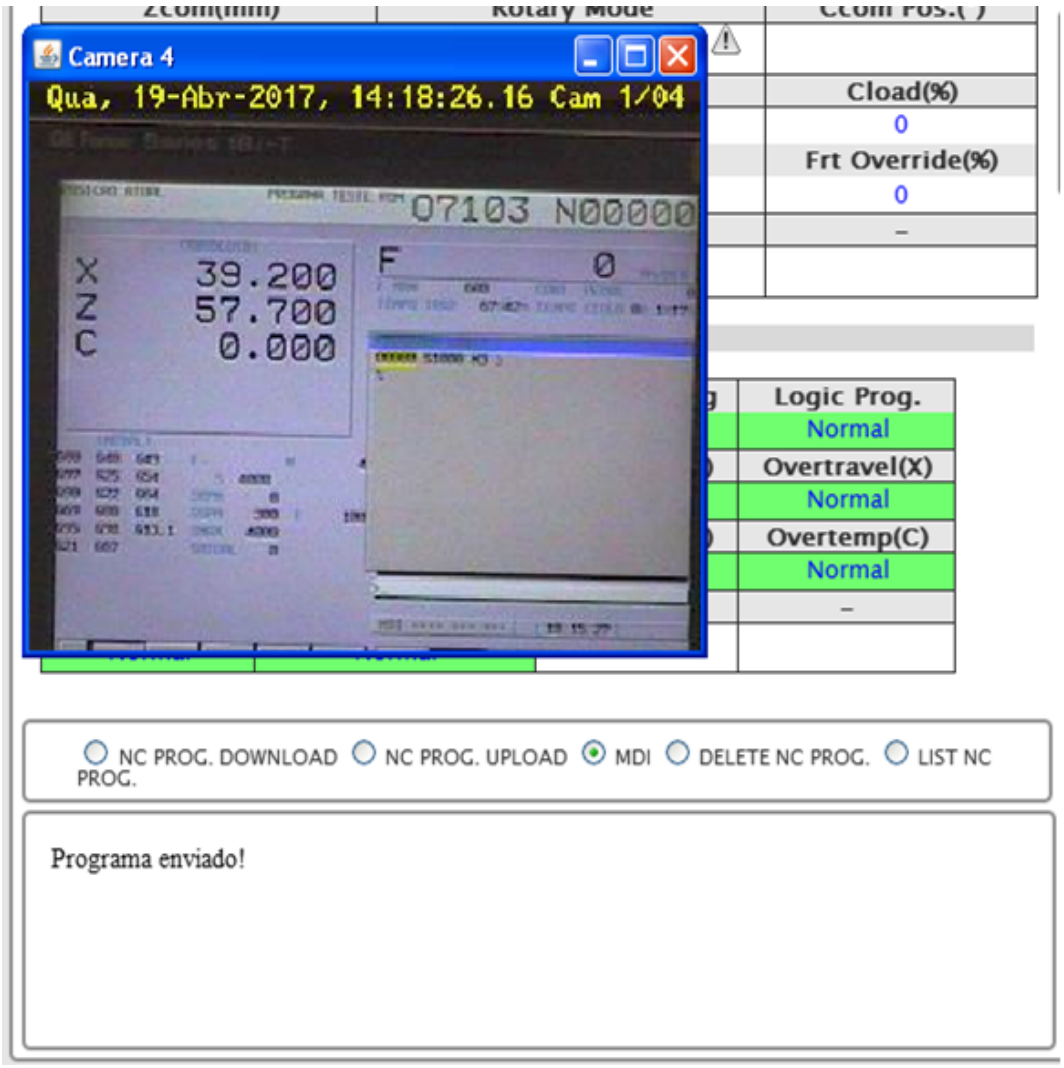

Figura 7.15: Enviando Comando Manual (MDI)

7.4 Análise dos Testes e Validação

A arquitetura do *Framework* implementado é modular isso permitiu que os testes de validação do sistema fossem executadas por serviço, representado pelos servidores de dados e comando remoto.

O servidor de Monitoramento MTConnect foi avaliado com base na disponibilidade de diferentes clientes desse serviço. Neste trabalho foram testados três casos, os dois primeiros casos foram clientes web desenvolvidos essencialmente em HTML e *JavaScript*, executando rotinas de requisição e atualização da página web com programação *Ajax*. A utilização desses recursos evidenciam a versatilidade e simplicidade do protocolo MTConnect. O último cliente testado foi desenvolvido em linguagem Java e possui recursos mais sofisticados para monitoramento e supervisão, no entanto, seu uso apresentou restrições de acesso. Contrariando o resultado desse último caso, o servidor MTConnect demonstrou sua utilidade pois, foi acessado por duas aplicações na Internet com êxito.

As características técnicas do *Web Service* RESTful, que foi especificado para interagir com fontes

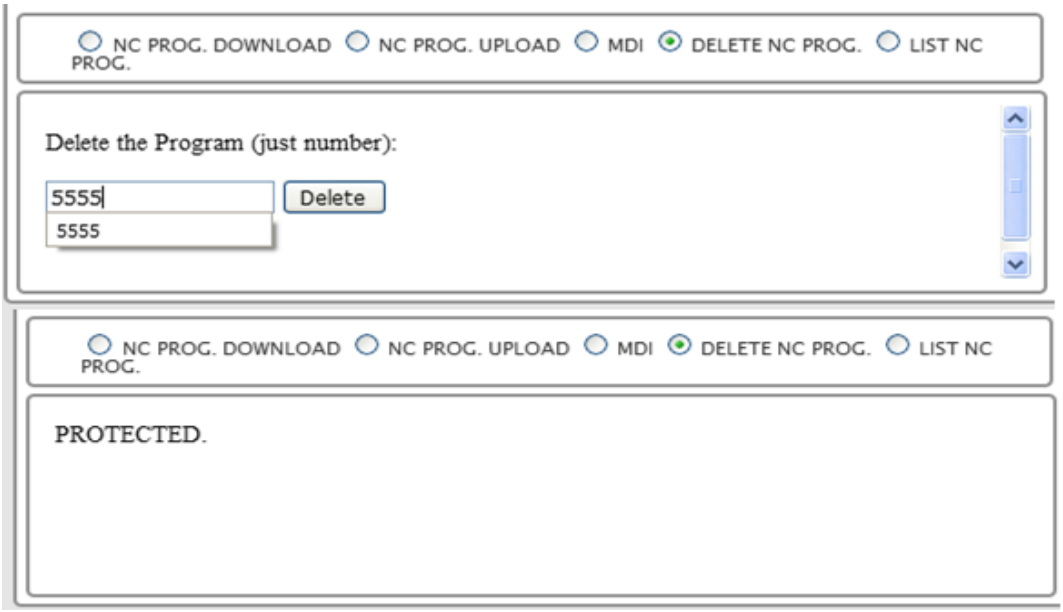

Figura 7.16: Função de Deletar Programa NC

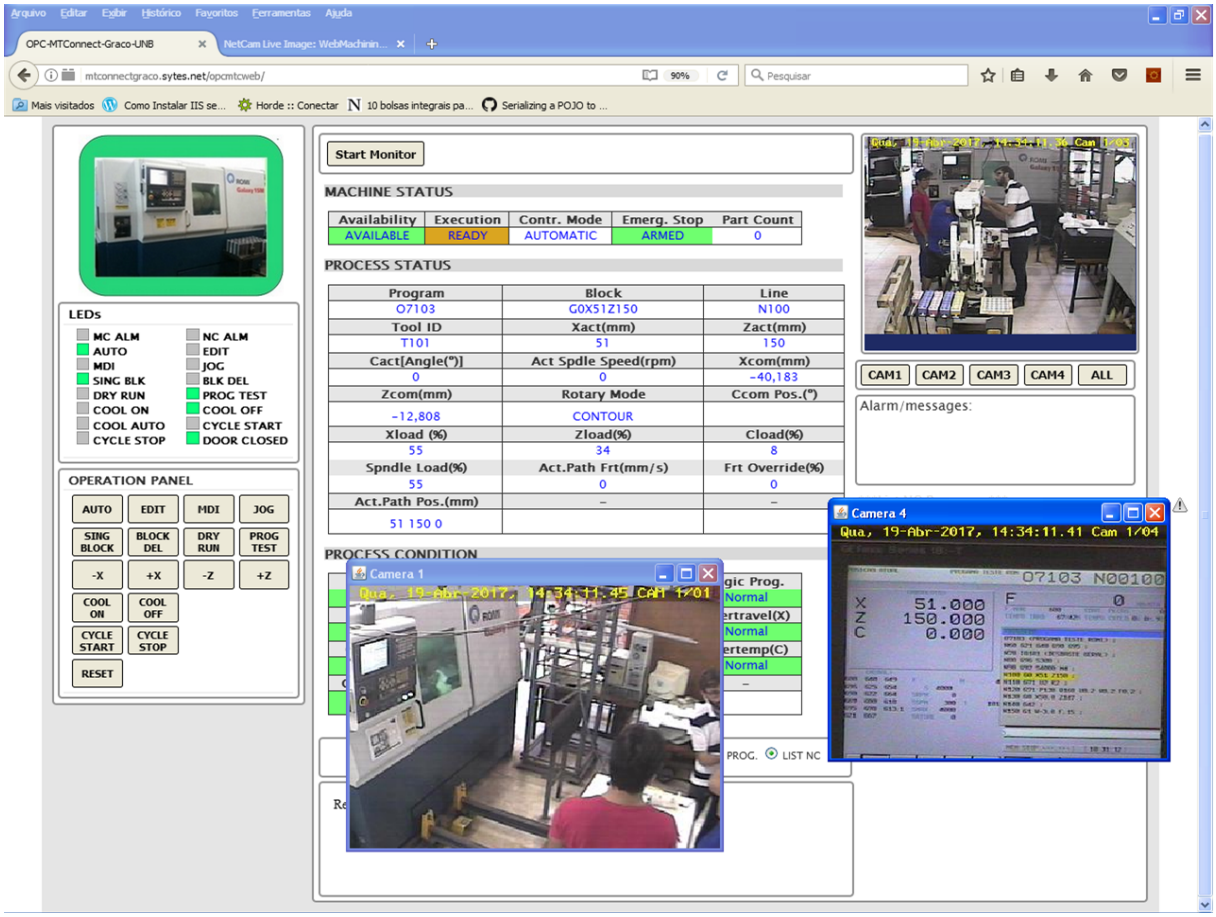

Figura 7.17: GUI CyberDNC e o funcionamento do serviço de imagens de vídeo WebCam

de dados, apresenta esse recurso como uma proposta promissora para o desenvolvimentos associados a servidores OPC para a construção de aplicações compartilhadas. No entanto, no âmbito deste trabalho, problemas de ordem técnica na utilização de *Web Service* RESTful lançaram a necessidade de se utilizar uma solução alternativa envolvendo o uso de programas python associados a API OpenOPC e mecanismo CGI para a realização via internet de supervisão e comando remoto do centro de torneamento Romi Galaxy 15M.

mas com da realização de novos testes e de um trabalho de depuração da lógica dessa aplicação, para que possa ser aperfeiçoado para que interaja de forma efetiva com um servidor OPC, e assim, fazê-lo disponível na Web de forma confiável.

As rotinas de teleoperação associadas a comando remoto (WebCNC) possuem um funcionamento baseado em requisições HTTP com programas CGI. É um serviço com uma complexidade menor em relação a outros serviços do *framework* e o tempo resposta não é um fator critico para a sua utilização, em comparação com atividades como monitoramento e supervisão. Essas fatores reduz suas as chances de falha.

A GUI CyberDNC de teleoperação e monitoramento desenvolvido como parte da arquitetura de implementação neste trabalho demonstrou bastante estabilidade durantes os testes, associado ao fato de possuir uma interface amigável, organizada de forma lógica, o que favorece a sua usabilidade.

Capítulo 8

Conclusões

Neste capítulo são apresentadas as conclusões gerais acerca da implementação da arquitetura do *framework* proposto, da metodologia empregada e a sobre a relação dos resultados obtidos com o paradigma da Indústria 4.0. Também elencou-se possíveis propostas de trabalhos futuros que gerem contribuições para o desenvolvimento dos estudos voltados para os sistemas manufatura remota dentro do contexto das fábricas inteligentes.

8.1 Conclusões do Trabalho

Este trabalho reuniu elementos do conceito de Manufatura Remota via internet, como a teleoperação, e das tecnologias de monitoramento disponíveis, associados às propostas tecnológicas para a manufatura da quarta revolução industrial (Indústria 4.0) com o propósito de conceber uma arquitetura de um *framework* que integrasse monitoramento, supervisão e teleoperação através da internet e estivesse em conformidade o esse novo paradigma da indústria de manufatura. Nesse sentido, foi implementado um sistema web com base nessa arquitetura, de forma que essa plataforma pudesse ser implementada e testes de validação pudessem ser efetuados.

Com o propósito de fornecer uma visão ampla das funções previstas na arquitetura foi empregada a modelagem IDEF0. Este recurso deu ao *framework* um aspecto modular em que cada elemento nos diferentes níveis desse diagrama correspondesse a um serviço ou atividade a ser implementada. A arquitetura geral definida pelo IDEF0 auxiliou na definição das necessidades dos clientes do projeto, informações básicas para iniciar o processo de projeto axiomático. Durante a implementação dessa metodologia aqueles conjuntos de parâmetros de projeto identificados como classes foram tratados dentro de uma abordagem de projeto axiomático de software, tendo como referência o modelo V para o projeto de software orientado a objeto.

O uso do projeto axiomático para projeto detalhado do sistema possibilitou que fosse definida uma sequência de implementação. Também foi obtido como resultado a relação entre os módulos que formam as principais classes do Adaptador MTConnect. Como produto final esses módulos foram reunidos em um diagrama classes.

O AD (*Axiomatic Design*) é geralmente aplicado em projetos complexos que envolvem muitos módulos, gerando bons resultados no projeto inicial de sistemas que envolvem apenas dispositivos, dispositivos e software, ou apenas software, mas que as informações e referências sobre o produto são muito poucas ou inexistentes. Com base nisso e nos resultado obtido com aplicação do AD no âmbito do projeto sistema implementado, é possível qualificar o projeto axiomático como adequado para o desenvolvido de sistemas no contexto da Indústria 4.0.

A implementação do serviço de monitoramento da arquitetura envolveu uma adequação dos recursos disponíveis em termos equipamentos e software, como a API e *driver* de comunicação com CNC Fanuc 18i, o Focas 1, que é compatível apenas com sistemas operacionais Windows, em versão não superior ao Windows XP. Foi necessário conciliar essas restrições com fato de viabilizar a operação de um serviço em que a velocidade de processamento e transmissão de dados são fatores-chave que influenciam na qualidade dos dados fornecidos. Observando isso, verificou-se que esse era um contexto ideal para explorar a versatilidade do padrão MTConnect. Um Adaptador pode ser desenvolvido em várias linguagens de programação, neste trabalho o Adaptador foi programado em linguagem C#.Net, escolha parcialmente motivada pelo fato do *driver* (*fwlib32*) para comunicação com o CNC Fanuc 18i ser projetado para plataforma Windows, visto que a plataforma *.Net* (*Dot Net*) foi originalmente projetada para sistemas operacionais Microsoft. Dessa forma, foi possível obter um Adaptador estável, sem problemas de compatibilidade.

Ainda tratando do serviço de monitoramento, a implementação do software Agente requereu conhecimento sobre o funcionamento e a configuração de um Agente MTConnect, pois o software foi projetado de maneira a permitir a configuração para operar com um número limitado de estruturas de dados de dispositivos CNC. Com isso, houve a necessidade de configura-lo manualmente no diretório de instalação, sendo necessário a alteração de parâmetros no arquivo *Agent.cfg*, que contem parâmetros do Agente e do Adaptador, e a modificação da estrutura do arquivo *Devices.xml*, que possui a estrutura de dados CNC do centro de torneamento. Esse esforço de configuração resultou na comunicação esperada entre Agente e Adaptador.

O procedimento de teste do serviço MTConnect envolveu diferentes clientes Web acessando o Agente. Nesse teste de conectividade verificou-se que mais de um cliente conectou-se ao software e dele requisitou e recebeu como resposta o *streaming* com dados de fabricação provenientes do CNC Fanuc 18i-Ta. Esse teste validou o sistema como uma aplicação multiplataformas, pois o Agente transmitiu dados para clientes web em um *browser*, um cliente desenvolvido com *applets* Java e uma aplicação para dispositivo móvel.

O serviço de supervisão proposto tem uma relação com o servidor de monitoramento, pois ambos fornecem dados úteis com potencial de alimentar bancos de dados em serviços de supervisão mais robustos, que envolvam processamento e análises de dados com estatísticas sobre os processos de fabricação e a planta como um todo. Neste trabalho, um dos desafios foi desenvolver um serviço em que um servidor OPC estivesse acessível via Internet e possuísse características que funcionalmente fossem similares as de um servidor OPC-UA, que é um padrão declarado como que reúne os atributos para um padrão da Indústria 4.0. Para isso, foram propostas duas soluções alternativas, a primeira envolve o uso de um *Web Service* RESTful como elemento *gateway* entre o servidor OPC e o cliente na Web. A segunda solução envolveu a utilização da API *OpenOPC* para Python, em que a comunicação com o servidor OPC e, consequentemente, com o CLP do centro de torneamento, baseia-se em requisições HTTP associadas a

programas escritos em python que são executados através de mecanismo CGI. Essa segunda alternativa foi a de implementação mais rápida e prática, e também a que garantiu o adequado funcionamento do serviço de supervisão do sistema.

No que se refere a teleoperação, a maioria dos módulos de software foram herdados de projetos anteriores, como o programas CGI utilizada na execução de comandos DNC remotamente, e o servidor de imagem (WebCam) da FMC. Nas funções implementadas usando mecanismo CGI o tempo de resposta não é considerado um fator crítico, de forma que as limitações desse protocolo nesse ponto não afetaram a usabilidade do sistema. O servidor WebCam, através de uma avaliação restritamente visual, demonstrou-se adequado para a atividade de teleoperação.

A integração de todos os servidores e serviços implementados em uma página Web requereu majoritariamente o uso de linguagens de programação para Web. O uso de HTML com CSS possibilitou a construção de um layout de interface que deu a GUI aspecto mais amigável ao usuário, fazendo com que os diferentes tipos de serviço fossem organizados em área específicas da página. O uso de *JavaScript* com *plugin* JQuery e aplicação de programação *Ajax* deu a aplicação um modo de operação mais lógico e mais usabilidade, uma vez que com *Ajax* o usuário não necessita recarregar a página para que uma atualização seja efetuado no HTML. O uso de uma arquivo *proxy* desenvolvido em PHP foi o que viabilizou a requisição e recepção streaming de dados provenientes do Agente MTConnect, uma vez que Ajax sem o auxílio de um *proxy* não faz requisições a arquivos externos ao diretório da aplicação. A opção por usar exclusivamente linguagens *script* e interpretadas, sem a necessidade de compiladores, no desenvolvimento de todo esse cliente Web serviu para ilustrar a versatilidade do sistema.

8.2 Sugestões para Trabalhos Futuros

A seguir são apresentados algumas sugestões para trabalhos futuros a serem desenvolvidas no sentido ajudar a desenvolver a manufatura remota via Internet associada às facilidades propostas pela estratégia Indústria 4.0:

- Implementar o Agente MTConnect e o Web Service do servidor OPCWeb integrados a um sistema de banco de dados, sendo executados na Nuvem. Isso produziria um serviço de supervisão baseado em análise de bases de dados disponíveis para uso por diferentes tipos de aplicações Web.
- Produzir uma metodologia que integre diferentes dispositivos CNC em uma única plataforma situada na Nuvem, com características similares de uma rede social, mas formada por máquinas CNC, onde todas as informações sobre esses dispositivos possam ser encontradas, o funcionamento dessas máquinas possa ser avaliado e intervenções possam ser efetuadas por usuários autorizados em qualquer parte do mundo.
- Desenvolver uma aplicação que utilize análises dinâmicas de dados provenientes de serviços MTConnect (CNC) e OPC(CLP) para a produção de estratégias de otimização de processos de fabricação em malha fechada.

- Utilizar um serviço MTConnect associado ao padrão STEP-NC (ISO 14649) para disponibilizar informações de fabricação de alto nível para aplicações situadas na Internet.

REFERÊNCIAS BIBLIOGRÁFICAS

ABBAS, H. A.; MOHAMED, A. M. Review on the design of web based scada systems based on opc da protocol. *International Journal of Computer Networks (IJCN)*, Computer Science Journals, v. 2, p. 266–277, 2011.

ACATECH. Cyber-physical systems. *Acatech Position Paper*, Acatech, p. 48, December 2011.

ADAMCZYK, Z.; KOCIOLEK, K. Cad/cam tecnological environment creation as an interactive application on the web. *Journal of Materials Processing Technology*, Elsevier, v. 109, p. 222–228, 2001.

AHN, S.; C., S.; P., W. *Internet-Based Design an Manufacturing*. [S.l.], 1999. 38 p.

AKYILDIZ, I. et al. Wireless sensor networks : a survey. v. 38, p. 393?422, 2002.

ALBERT, M. Seven things to know about the internet of things and industry 4.0. Modern Machine Shop Magazine, January 2015. Disponível em: <http://www.mmsonline.com/articles/7-things-to-know-about-theinternet-of-things-and-industry-40>.

ALVARES, A.; FERREIRA, J. Webmachining: System for the design and manufacture of feature-based parts through the web. *ABCM Symposium Series in Mechatronics*, v. 3, p. 688–700, 2008.

ALVARES, A.; ROMARIZ, L. Telerobotics: Methodology for the development of a through-the-internet robotic teleoperated system. *Journal of the Brazilian Society of Mechanical Sciences*, Brazilian Society of Mechanical Sciences, v. 24, p. 122–126, 2002.

ALVARES, A. et al. Um sistema de telemanufatura baseado na web orientado ao processo de oxicorte. *Encontro Nacional de Engenharia de Produção*, XXII, Outubro 2002.

ALVARES, A.; SILVA, F.; FERREIRA, J. Webturning: TeleoperaÇÃo de um centro de torneamento via internet. *Congresso Brasileiro de Engenharia de Fabricação*, III, 2005.

ALVARES, A.; TOURINO, S. Desenvolvimento de um robô móvel autônomo teleoperado via internet. *CONEM 2000 - Congresso Nacional de Engenharia Mecânica*, ABCM, p. 8, Agosto 2000.

ALVARES, A. J. Uma metodologia para integração cad/capp/cam voltada para manufatura remota de peças rotacionais baseada na internet. *Tese de Doutorado. Programa de Pós-graduação em Engenharia Mecânica da UFSC. Florianópolis, SC*, 2005.

ALVARES, A. J. et al. *Tendências e Aplicações Especiais -Capitulo 11. In: Robótica Industrial-Aplicação na Indústria De Manufatura e de Processos*. 1. ed. [S.l.]: Edgard Blücher, 2002. 256 p. ISBN 8521203152.

AMR. An e-manufacturing strategy needs to be developed from the manufacturing strater? *AMR Research*, August 2000.

ATZORI L, L. A.; MORABITO, G. The internet of things : A survey. *Computer Networks: The International Journal of Computer and Telecommunications Networking archive Volume*, Elsevier, v. 54, p. 2787–2805, October 2010.

BAILEY, M. J. Tele-manufacturing: Rapid prototyping on the internet. *IEEE Computer Graphics and Applications*, IEEE, v. 15, p. 20–26, November 1995.

BALIEIRO, R. L. Desenvolvimento de uma ferramenta computacional para aquisição via internet de dados de dispositivo de campo em ambiente fieldbus. *Dissertação de mestrado. Programa de Pós-graduação em engenharia mecânica da USP. São Carlos, SP*, 2008.

BANDYOPADHYAY, D.; SEN, J. Internet of things: Applications and challenges in technology and standardization. *Wireless Personal Communications*, v. 58, p. 49–69, 2011.

BENAVENTE, J. C. T. Um sistema para o projeto e fabricação remota de peças prismáticas via internet. *Dissertação de mestrado. Programa de Pós-graduação em engenharia mecânica da UFSC. Florianópolis, SC*, 2007.

BENAVENTE, J. C. T. Um sistema para o projeto e fabricação de peças mecânicas a distância via internet aderente à norma iso 14649 (step-nc). *Tese de Doutorado. Programa de Pós-graduação em Engenharia Mecânica da UFSC. Florianópolis, SC*, 2011.

BRETTEL, M. et al. How virtualization, decentralization and network building change the manufacturing landscape: An industry 4.0 perspective. *International journal of mechanical, aerospace, industrial and mechatronics engineering*, World Academy of Science, Engineering and Technology, v. 8, p. 37–44, 2014.

BROWN, S. M.; WRIGHT, P. A progress report on the manufacturing analysis service, an internet-based reference tool. *Journal of Manufacturing Systems*, Elsevier, v. 17, p. 389–401, 1998.

BUGHIN, J.; CHUI, M.; MANYIKA, J. Clouds, big data, and smart assets: ten tech-enabled business trends to watch. *McKinsey Quarterly*, McKinsey Global Institute, August 2010.

BURKE, J. T.; IWANITZ, F.; LANGE, J. Opc from data access to unified architecture. VDE VERLAG GMBH, 2010.

CAVALIERI, S.; CHIACCHIO, F. Analysis of opc ua performances. v. 36, p. 165?177, 2013.

CAVOUKIAN, A. Privacy in the clouds: a white paper on privacy and digital identity implications for the internet. *Information and Privacy Commission of Ontario*, p. 32, 2008.

CHEN, S. et al. A vision of iot : Applications , challenges ,and opportunities with china perspective. *IEEE INTERNET OF THINGS JOURNAL*, v. 1, p. 349–359, August 2014.

CHENG, K.; BATEMAN, R. E-manufacturing : Characteristics, applications and potentials. *Progress in Natural Science*, Elsevier, v. 18, p. 1323–1328, 2008.

CHOO, H.; LEE, I. Integrated framework of idef modeling methods for structured design of shop floor control systems. *International Journal of Computer Integrated Manufacturing*, Taylor & Francis Online, v. 12, p. 113–128, 1999.

CLAPIS P.J.AND HINTERSTEINER, J. Enhancing object-oriented software development through axiomatic design. *Proceedings of ICAD2000 - First International Conference on Axiomatic Design*, 2000.

COCHRAN D.S.AND EVERSHEIM, W. K. G.; SESTERHENN, M. The application of axiomatic design and lean management principles in the scope of production system segmentation. *International Journal of Production Research*, Taylor & Francis Online, v. 38, p. 1377–1396, 2000.

COCHRAN D.S.AND REYNAL, V. Understanding lean manufacturing according to axiomatic design principles. *The Lean Aircraft Initiative Report Series*, RP96-07-28 MIT, 1996.

COLOQUHOUN, G. J.; BAINES, R. W.; CROSSLEY, R. A state of art review of idef0. *International Journal of Computer Integrated Manufacturing*, Taylor & Francis Online, v. 6, p. 252–264, 1993.

DIKAIAKOS, M. D. et al. Cloud computing-distributed internet computing for it and scientific research. Sept-Oct 2009.

DO, S.-H. Axiomatic design software. *Unpublished paper*, MIT Axiomatic Design Group, August 1999.

DO, S.-H. Software product lifecycle management using axiomatic approach. *Proceedings of ICAD 2004 - Third International Conference on Axiomatic Design*, June 2004.

EDRINGTON, B. et al. Machine monitoring system based on mtconnect technology. *Procedia CIRP*, Elsevier, v. 22, p. 92–97, 2014.

EDSTROM, D. *MTConnect To Measure Is To Know*. 1. ed. [S.l.]: Virtual Photons Electrons, LLC, 2013. ISBN B01K3MB6HU.

ELSHAFEI, M. *Modern Distributed Control Systems: A Comprehensive Coverage of Dcs Technologies and Standards*. 1. ed. [S.l.]: Createspace Independent Publishing Platform, 2014. 478 p. ISBN 978-1535103855.

FALLER, C.; FELDMÜLLER, D. Industry 4.0 learning factory for regional smes. *Procedia CIRP*, Elsevier, v. 32, p. 88–91, 2015.

FDC. *O que seria a Indústria 4.0? Boletim: Fevereiro/2016, Pesquisa sobre digitalização*. [S.l.], 2016. 4 p.

FERNANDES, R. F.; TORRISI, N. M.; BRANDÃO, D. Remote tuning of industrial controllers using cyberopc technology. *IEEE International Conference on Industrial Informatics (INDIN)*, IEEE, p. 750–756, 2009.

FIDEL, B. Web services rest versus soap. 2015. Disponível em: <http://www.devmedia.com.br/web-services-rest-versus-soap/32451>.

FIELDING, R. The representational state transfer (rest). *PhD dissertation. Irvine: Department of Information and Computer Science, University of California*, 2000. Disponível em: <http://www.ics.uci.edu/fielding/pubs/dissertation/top.htm>.

FOSTER, I. et al. Cloud computing and grid computing 360-degree compared. *Grid Computing Environments Workshop*, IEEE, p. 10, November 2008.

FRANÇA, T. V.; TORRISI, N. M.; BOTTENE, A. C. Cnc machine tool monitoring using mtconnect communication architecture. *22nd International Congress of Mechanical Engineering (COBEM 2013)*, ABCM, p. 6660–6669, 2013.

GONÇALVES-COELHO, A. Tutorials of the fifth international conference on axiomatic design. *Proceedings of ICAD 2009 - Fifth International Conference on Axiomatic Design*, March 2009.

GONÇALVES, R. N. Desenvolvimento de servidores opc da, opc ua e wrappers para aplicação em automação. *Dissertação de mestrado. Programa de Pós-graduação em engenharia elétrica da Universidade Federal de Itajubá. Itajubá, MG*, 2012.

GORECKY, D. et al. Human machine interaction in the industry 4.0 era. *IEEE International Conference on Industrial Informatics (INDIN)*, IEEE, v. 12, p. 289–294, July 2014.

GRÜNER, S.; PFROMMER, J.; PALM, F. A restful extension of opc-ua. *Proceedings of 2015 IEEE World Conference on Factory Communication Systems (WCFS)*, IEEE, p. 25–27, 2015.

GRÜNER, S.; PFROMMER, J.; PALM, F. Restful industrial communication with opc-ua. *IEEE Transactions On Industrial Informatics*, IEEE, v. 12, p. 1832–1841, 2016.

GUBBI, J. et al. Internet of things (iot) : A vision, architectural elements, and future directions. *Future Generation Computer Systems*, Elsevier, v. 29, p. 1645–1660, September 2013.

HELANDER, M. G.; JIAO, J. Coupling in design of human computer interaction. *Proceedings of ICAD2002 - Second International Conference on Axiomatic Design*, Institute for Axiomatic Design, p. 1–5, June 2002.

HONG, X.; JIANHUA, W. Using standard components in automation industry: a study on opc specification. *Computer Standards & Interfaces*, Elsevier, v. 28, p. 386–395, 2006.

HONLE, N. et al. Benefits of integrating metadata into a context model. *IEEE Computer Society*, IEEE, v. 3, p. 25–29, 2005.

HUANG, J. et al. A novel deployment scheme for green internet of things. *IEEE Internet of Things Journal*, IEEE, v. 1, April 2014.

IVEZIC, N.; KULVATUNYOU, B.; SRINIVASAN, V. On architecting and composing through-life engineering information services to enable smart manufacturing. *Procedia CIRP*, Elsevier, v. 22, p. 45–52, 2014.

IWANITS, F.; LANGE, J. *OPC fundamentals, implementation, and application*. 2. ed. [S.l.]: Panoramix Services, LLC, 2002.

JAMSHIDNEZHAD, B. Towards a rational basis for user interface design methods. *Proceedings of ICAD2004 - The Third International Conference on Axiomatic Design*, p. 1–4, June 2004.

JUELS, A. Rfid security and privacy: a research survey. *IEEE Journal on Selected Areas in Communications*, v. 24, p. 381–394, 2006.

KAGERMANN, H.; WAHLSTER, W.; HELBIG, J. *Recommendations for Implementing the Strategic Initiative Industrie 4.0: Final Report of the Industrie 4.0 Working Group*. [S.l.], 2013. 82 p.

KIM S.-J.AND SUH, N.; S.-G., K. Design of software systems based on axiomatic design. *Annals of the CIRP*, Elsevier, v. 40, p. 165–170, 1991.

KOC, M.; LEE, J. Introduction to e-manufacturing :. *International Conference on Frontiers on Design and Manufacturing*, 2002.

KOC, M. et al. Chapter 97: Introduction to e- manufacturing. *The Industrial Information Technology Handbook*, CRS Press, 2004.

KONDOR, R. Opc tutorial. Matrikon OPC, 2008. Disponível em: <http://www.matrikonopc.com/resources/opc-tutorials.aspx>.

LASI, H. et al. Industry 4.0. v. 6, 2014.

LEE, J. E-manufacturing : fundamental, tools, and transformation. *Robotics and Computer-Integrated Manufacturing*, v. 19, p. 501–507, 2003.

LEE, J.; BAGHERI, B.; KAO, H.-A. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, v. 3, p. 18–23, 2015.

LEE, J. et al. Recent advances and trends in predictive manufacturing systems in big data environment. *Manufacturing Letters*, Society of Manufacturing Engineers (SME), v. 1, p. 38–41, 2013.

LEE, X.; HU, J. Design and research of remote monitoring system based on opc xml-da. 2008.

LI, G.; WEI, M. Everything-as-a-service platform for on-demand virtual enterprises. p. 435–452, April 2012.

LI, Z.; TATE, D. Managing intra-class complexity with axiomatic design and design structure matrix approaches. *Proceedings of ICAD 2011 - Sixth International Conference on Axiomatic Design*, March 2011.

LING, Z.; CHEN, W.; YU, J. Research and implementation of opc server based on data access specification. *Proceedings of the 5th World Congress on Intelligent Control and Automation*, IEEE, v. 6, p. 1475–1478, 2004.

LINS, T. Indústria 4.0 - desafios parte 1. 2015.

LIU, Q.; SUN, X.; MAHDAVIAN, S. Establishment of the model for flexible manufacturing system based on dorba e idef0. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 15, p. 472–483, 2008.

MACDOUGALL, W. *Industrie 4.0 - Smart Manufacturing for the Future*. [S.l.], 2014. 40 p. Disponível em: <http://www.gtai.de>.

MACHADO-JUNIOR, R. R. Desenvolvimento de um middleware para comunicação via web services e sua aplicação em sistemas de aquisição de dados industriais. *Dissertação de mestrado. Programa de Pós-graduação em Engenharia Elétrica e de Computação da UFRN. Natal,RN*, 2014.

MAHMOUD, M. S.; SABIH, M.; ELSHAFEI, M. Using opc technology to support the study of advanced process control. *ISA Transactions*, Elsevier, v. 55, p. 155–167, 2015.

MAHNKE, W.; LEITNER, S.-H.; DAMM, M. *OPC Unified Archtecture*. 1. ed. [S.l.]: SpringerSpringer-Verlag Berlin Heidelberg, 2009. 339 p. ISBN 3540688986.

MALEK, L.; WOLF, C.; GUYOT, P. Telemanufacturing : A flexible manufacturing solution. *International Journal of Production Economics*, Elsevier, v. 56-57, p. 1–12, September 2008.

MARR, B. What everyone must know about industry 4.0. *Tech - Big Data, Forbes Contributors*, Forbes, June 2016. Disponível em: <http://www.forbes.com/sites/bernardmarr/2016/06/20/what-everyone-must-know-about-industry-4-0/#327956184e3b>.

MELL, P.; GRANCE, T. The nist definition of cloud computing. National Institute of Standards and Technology, v. 53, October 2009.

MICHALOSKI, J. L. et al. Web-enabled, real-time, quality assurance for machining production systems. *Procedia CIRP*, Elsevier, v. 10, p. 332–339, 2013.

MISRA, S. C.; MONDAL, A. Identification of a company?s suitability for the adoption of cloud computing and modeling its corresponding return on investment. February 2011.

MOLINA, A.; SANTAELLA, A. Achieving e-manufacturing : multihead control and web technology for the implementation of a manufacturing execution system. *Journal of Intelligent Manufacturing*, Springer, v. 17, p. 715–724, 2006.

MONOSTORI, L. Cyber-physical production systems: Roots, expectations and r&d challenges. *Procedia CIRP*, Elsevier, v. 17, p. 9–13, 2015.

MTCONNECT-INSTITUTE-OPC-FOUNDATION. Mtconnect-opc-ua companion specification - release candidate version 1.02. MTCONNECT INSTITUTE - OPC FOUNDATION, November 2013.

NIST. Foundations for innovation : Strategic r&d opportunities for 21st century cyber-physical systems. *Report of the Steering Committee for Foundations and Innovation for Cyber-Physical*, NIST, p. 32, January 2013. Disponível em: <http://events.energetics.com/NIST-CPSWorkshop/pdfs/12_Cyber_Physical_Systems02-01-13_final.pdf>.

NOF, S. *Handbook of Industrial Robotics*. 2. ed. [S.l.]: John Wiley Sons, 1999.

NOF, S. E-work and e-mfg.: The state of the art and challenges for production and logistics managers. *IFAC Symposium on Information Control Problems in Manufacturing(INCOM)*, International Federation of Automatic Control, v. 11, April 2004.

OPC-FOUNDATION. Data access custom interface standard - version 3.00 released. OPC Foundation, March 2003.

OPC-FOUNDATION. Opc ua part 1 - overview and concepts - 1.01 specification. OPC Foundation, February 2009.

OPEN-GROUP. Service oriented architecture (soa). *The Open Group*, 2010. Disponível em: <http://www.opengroup.org/projects/soa>.

PALLIS, G. Cloud computing: The new frontier of internet computing. *IEEE Internet Computing*, IEEE, v. 14, p. 70–73, Sept.-Oct. 2010.

PANTONI, R. P.; TORRISI, N.; BRANDãO, D. An open and non-proprietary device description for fieldbus devices for public ip networks. 2007.

PIMENTEL, A. R.; STADZISZ, P. C. A use case based object-oriented software design approach using the axiomatic design theory. *Proceedings of ICAD 2006 - Fourth International Conference on Axiomatic Design*, June 2006.

PISCHING, M. A. et al. An architecture for organizing and locating services to the industry. *COBEM2015 - 23rd ABCM International Congress of Mechanical Engineering*, ABCM, v. 23, p. 48–55, December 2015.

PRADHAN, S.; HUANG, W. Virtual manufacturing information system using java and jdbc. *Computers & Industrial Engineering*, Elsevier, v. 35, p. 255–258, 1998.

RABELO, R. J. Notas de aula - avaliação de desempenho de sistemas - modelagem de processos e a metodologia idef0. *Departamento de Automação e Sistemas*, Universidade Federal de Santa Catarina, 2016.

RANGARAJAN, A.; DORNFELD, D.; WRIGHT, P. Probabilistic precision process planning - p4. *Consortium on Deburring and Edge Finishing*, Trans. North American Manufacturing Research Institution/SME, v. 31, p. 539–546, 2003.

RIMAL, B. et al. Architectural requirements for cloud computing systems: an enterprise cloud approach. *Journal of Grid Computing*, Springer, v. 9, p. 3–26, March 2011.

ROCKWELL. Making sense of e-manufacturing: A roadmap for manufacturers. *Industry White Paper*, n. 1, 2002.

RONDON, L. Desenvolvimento de sistema de supervisão utilizando padrão mtconnect. *Monografia de conclusão de curso apresentada ao Instituto de Ciências Matemáticas e de Computação ? ICMC-USP. São Paulo, SP*, 2010.

RYAN, M. D. Cloud computing privacy concerns on our doorstep. *Communications of the ACM*, ACM, v. 54, p. 36–38, 2011.

SAHIN, C.; BOLAT, E. D. Development of remote control and monitoring of web-based distributed opc system. *Computer Standards and Interfaces*, v. 31, p. 984–993, 2009.

SCHONWALDER, J.; FOUQUET M. RODOSEK, G.; HOCHSTATTER, I. Future internet = content + services + management. July 2009.

SCIENTIFIC-AMERICAN. 21st century medicine: Cybersurgery. 2000. Disponível em: <http: //www.pbs.org/safarchive/4_class/45_pguides/pguide_605/4565_cyber.html>.

SHALINI, S. Smart manufacturing with cloud computing. *One Million*, December 2010. Disponível em: <http://www.sramanamitra.com/2010/12/03/smart-manufacturing-with-cloud-computing-part-1/>.

SHERIDAN, T. *Supervisory control of remote manipulators, vehicles and dynamic processes: experiment in command and display aiding In : Advances in Man Machine Systems Research*. 1. ed. [S.l.]: Jai Pr, 1984. 317 p. ISBN 978-0892324040.

SHERIDAN, T. *Telerobotics, Automation and Human Supervisory Control*. 1. ed. [S.l.]: MIT Press, 1992. ISBN 0-262-19316-7.

SHI, J. et al. A survey of cyber-physical systems. *2011 International Conference on Wireless Communications and Signal Processing, WCSP 2011*, 2011.

SILVA, E. et al. Desenvolvimento de um sistema de monitoramento para mÁquinas-ferramentas utilizando-se o padrÃo mtconnect. *Sexto Congresso Nacional de Engenharia Mecânica*, ABCM, v. 6, p. 8, 2010.

SILVA, E. J. et al. Web based supervisory system for plunge grinding operation. *Proceedings of 21st Brazilian Congress of Mechanical Engineering*, ABCM, v. 5, 2011.

SOBEL, W. Mtconnect standard part 1 - overview and protocol - 1.1.0 - final. MTConnect Institute, April 2010.

SOUZA-JÚNIOR, J. L. N. Promme : Metodologia para gestÃo da produÇÃo via web em ambiente de manufatura. *Dissertação de mestrado. Programa de Pós-graduação em Sistemas Mecatrônicos da UNB. Brasília, DF*, 2008.

SUH, N. *The Principles of Design*. 1. ed. [S.l.]: Oxford University Press, 1990. ISBN 0-19-504345-6.

SUH, N. *Axiomatic Design of Software - Chapter 5. In: Axiomatic Design - Advances and Applications*. 1. ed. [S.l.]: Oxford University Press, 2001.

SUH, N.; DO, S. Axiomatic design of software system. *CIRP Annals - Manufacturing Technology*, Elsevier, v. 49, p. 95–100, 2000.

SUH, N.; DO, S. Object-oriented software design with axiomatic design. *Proceedings of ICAD2000 First International Conference on Axiomatic Design*, International Conference on Axiomatic Design, v. 1, p. 278–284, 2000.

TEIXEIRA, E. Desenvolvimento da unidade de gerenciamento de uma célula flexível de manufatura integrada a um sistema cad/capp/cam. *Dissertação de Mestrado. Programa de Pós-graduação em Sistemas Mecatrônicos da UNB. Brasília, DF*, 2006.

TOGAY, C.; CANIAZ, E. S.; DOGRU, A. H. Rule based axiomatic design theory guidance for software development. *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual*, p. 544–552, 2012.

TORRISI, N. et al. Design of a communication system for integration of industrial networks over public ip networks. *IEEE International Conference on Industrial Informatics (INDIN)*, IEEE, v. 1, p. 201–206, 2007.

TORRISI, N. M. Monitoring services for industrial. *IEEE Industrial Electronics Magazine*, IEEE, v. 5, p. 49–60, 2011.

VIJAYARAGHAVAN, A.; DORNFELD, D. Automated energy monitoring of machine tools. *CIRP Annals - Manufacturing Technology*, Elsevier, v. 59, p. 21–24, 2010.

VIJAYARAGHAVAN, A. et al. Improving machine tool interoperability using standardized interface protocols: Mtconnect. *Series Green Manufacturing and Sustainable Manufacturing Partnership*, International Symposium on Flexible Automation, p. 7, 2008.

W3C. W3c note - simple object access protocol (soap) 1.1. Maio 2000. Disponível em: <http: //www.w3.org/TR/2000/NOTE-SOAP-20000508>.

WARNDORF, P.; FOX, A.; SOBEL, W. Mtconnect technology. *White Paper*, MTConnect Institute, 2007.

WELBOURNE, E. et al. Building the internet of things using rfid the rfid ecosystem experience. *IEEE Internet Computing*, IEEE, v. 13, p. 48–55, May-June 2009.

WOOLDRIDGE, M. *An Introduction to MultiAgent Systems*. 1. ed. [S.l.]: John Wiley & Sons Inc., 2007. 365 p. ISBN 0-471-49691-X.

WU, L.; YANG, C. A solution of manufacturing resources sharing in cloud computing environment. *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, In: Luo Y, editor. 6240 LNCS. Berlin Heidelberg: Springer-Verlag, p. 247?252, 2010.

XU, X. From cloud computing to cloud manufacturing. *Robotics and Computer-Integrated Manufacturing*, Elsevier, v. 28, p. 75–86, 2012.

YIN, Y.; ZHOU, B. The analysis and research of opc xml-da server. *Energy Procedia*, Elsevier, v. 16, p. 1535–1540, 2012.

ZHAI, S.; MILGRAM, P. A telerobotic virtual control system. *in Proceedings of SPIE, Cooperative Intelligent Robotics in Space II*, v. 1612, p. 311–320, 1991.

ZHAI, S.; MILGRAM, P. Human robot synergism and virtual telerobotic control. *in Proceedings of 25th Annual Conference of the Human Factors Association of Canada*, v. 25, 1992.

APÊNDICES

A. ARQUIVO *DEVICES.XML*

Este apêndice apresenta a estrutura do arquivo *Devices.xml*, que corresponde a um dos elementos essenciais para a adequada configuração software Agente dentro da arquitetura de uma aplicação MTConnect. Esse arquivo descreve toda estrutura de dados que com o parâmetros do CNC da máquina que são monitorados através do servidor MTConnect.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <MTConnectDevices xmlns="urn:mtconnect.org:MTConnectDevices:1.1"
3  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4  xsi:schemaLocation="urn:mtconnect.org:MTConnectDevices:1.1
5  http://www.mtconnect.org/schemas/MTConnectDevices_1.1.xsd">
6  <Header bufferSize="130000" instanceId="1" creationTime="2014-11-06T19:21:45"
7  sender="local" version="1.1"/>
8  <Devices>
9  <Device sampleRate="10.0" name="RomiGalaxy15M" iso841Class="1" uuid="F18i" id="d1">
10 <Description manufacturer="UNB" serialNumber="F18ita"></Description>
11 <DataItems>
12 <DataItem category="EVENT" id="Id1" name="avail" type="AVAILABILITY"></DataItem>
13 </DataItems>
14 <Components>
15 <Axes name="Axes" id="A">
16 <Components>
17 <Rotary name="C" id="c">
18 <DataItems>
19 <DataItem category="SAMPLE" id="cspd" name="Cact" type="ANGLE"
20 units="DEGREE" subType="ACTUAL"></DataItem>
21 <DataItem category="SAMPLE" id="Cload" name="Cload" type="LOAD"
22 units="NEWTON"></DataItem>
23 <Constraints>
24 <Value>SPINDLE</Value>
25 <Value>INDEX</Value>
26 <Value>CONTOUR</Value>
27 <Value>UNAVAILABLE</Value>
28 </Constraints>
29 </DataItem>
30 <DataItem category="CONDITION" id="Coverload" name="Coverload" type="LOAD"/>
31 <DataItem category="CONDITION" id="Ctemp" name="Ctemp" type="TEMPERATURE"/>
32 </DataItems>
33 </Rotary>
34 <Rotary name="S1" id="b">
35 <DataItems>
36 <DataItem category="SAMPLE" id="S1speed" name="S1speed" type="SPINDLE_SPEED"
37 units="REVOLUTION/MINUTE" subType="ACTUAL"></DataItem>
38 <DataItem category="SAMPLE" id="S1speedc" name="S1speedc" type="SPINDLE_SPEED"
39 units="REVOLUTION/MINUTE" subType="COMMANDED"></DataItem>
40 <DataItem category="SAMPLE" id="S1load" name="S1load" type="LOAD"
41 units="NEWTON"></DataItem>
42 <DataItem category="SAMPLE" id="rfunc" name="rfunc" type="ROTARY_MODE">
43 <DataItem category="CONDITION" id="S1overload" name="S1overload" type="LOAD"/>
44 <DataItem category="CONDITION" id="S1temp" name="S1temp" type="TEMPERATURE"/>
45 </DataItems>
```

```
46  </Rotary>
47
48  <Linear name="X" id="x">
49  <DataItems>
50  <DataItem category="SAMPLE" id="Xact" name="Xact" type="POSITION"
51  units="MILLIMETER" subType="ACTUAL"></DataItem>
52  <DataItem category="SAMPLE" id="Xcomm" name="Xcomm" type="POSITION"
53  units="MILLIMETER" subType="COMMANDED"></DataItem>
54  <DataItem category="SAMPLE" id="Xload" name="Xload" type="LOAD" units="NEWTON">
55  </DataItem>
56  <DataItem category="CONDITION" id="Xoverload" name="Xoverload" type="LOAD"/>
57  <DataItem category="CONDITION" id="Xtemp" name="Xtemp" type="TEMPERATURE"/>
58  <DataItem category="CONDITION" id="Xtravel" name="Xtravel" type="POSITION"/>
59  </DataItems>
60  </Linear>
61  <Linear name="Y" id="y">
62  <DataItems>
63  <DataItem category="SAMPLE" id="Yact" name="Yact" type="POSITION"
64  units="MILLIMETER" subType="ACTUAL"></DataItem>
65  <DataItem category="SAMPLE" id="Ycomm" name="Ycomm" type="POSITION"
66  units="MILLIMETER" subType="COMMANDED"></DataItem>
67  <DataItem category="SAMPLE" id="Yload" name="Yload" type="LOAD" units="NEWTON">
68  </DataItem>
69  <DataItem category="CONDITION" id="Yoverload" name="Yoverload" type="LOAD"/>
70  <DataItem category="CONDITION" id="Ytemp" name="Ytemp" type="TEMPERATURE"/>
71  <DataItem category="CONDITION" id="Ytravel" name="Ytravel" type="POSITION"/>
72  </DataItems>
73  </Linear>
74  <Linear name="Z" id="z">
75  <DataItems>
76  <DataItem category="SAMPLE" id="zp" name="Zact" type="POSITION"
77  units="MILLIMETER" subType="ACTUAL"></DataItem>
78  <DataItem category="SAMPLE" id="zcom" name="Zcomm" type="POSITION"
79  units="MILLIMETER" subType="COMMANDED"></DataItem>
80  <DataItem category="SAMPLE" id="Zload" name="Zload" type="LOAD"
81  units="NEWTON"></DataItem>
82  <DataItem category="CONDITION" id="Zoverload" name="Zoverload" type="LOAD"/>
83  <DataItem category="CONDITION" id="Ztemp" name="Ztemp" type="TEMPERATURE"/>
84  <DataItem category="CONDITION" id="Ztravel" name="Ztravel" type="POSITION"/>
85  </DataItems>
86  </Linear>
87  </Components>
88  </Axes>
89  <Controller name="Controller" id="cont">
90  <DataItems>
91  <DataItem type="MESSAGE" name="message" category="EVENT" id="message"></DataItem>
92  <DataItem category="EVENT" id="estop" name="estop" type="EMERGENCY_STOP"></DataItem>
93  <DataItem type="PART_COUNT" name="part_count" category="EVENT" id="part_count"></DataItem>
94  <DataItem category="CONDITION" id="cc1" name="comm" type="COMMUNICATIONS"></DataItem>
95  <DataItem category="CONDITION" id="cc3" name="motion" type="MOTION_PROGRAM"></DataItem>
96  <DataItem category="CONDITION" id="cc4" name="logic" type="LOGIC_PROGRAM"></DataItem>
97  <DataItem category="CONDITION" id="cc6" name="system" type="SYSTEM"></DataItem>
98  </DataItems>
99  <Components>
100 <Path id="path" name="path">
101 <DataItems>
102 <DataItem type="ACTIVE_AXES" name="Act_axes" category="EVENT" id="aAxes"></DataItem>
103 <DataItem category="EVENT" id="block" name="block" type="BLOCK"></DataItem>
104 <DataItem category="EVENT" id="execution" name="execution" type="EXECUTION"></DataItem>
105 <DataItem category="EVENT" id="mode" name="mode" type="CONTROLLER_MODE"></DataItem>
```

```
<DataItem category="EVENT" id="program" name="program" type="PROGRAM"></DataItem>
<DataItem category="EVENT" id="line" name="line" type="LINE"></DataItem>
<DataItem type="PATH_FEEDRATE" name="path_feedrate" category="SAMPLE" id="path_feedrate"
units="MILLIMETER/SECOND" subType="ACTUAL"></DataItem>
<DataItem type="PATH_FEEDRATE" name="path_fdrovr" category="SAMPLE" id="path_fdrovr"
units="PERCENT" subType="OVERRIDE"></DataItem>
<DataItem type="PATH_POSITION" name="path_position" category="SAMPLE" id="path_position"
units="MILLIMETER_3D" coordinateSystem="WORK"></DataItem>
<DataItem type="TOOL_ID" name="tool_id" category="EVENT" id="tool_id"></DataItem>
<DataItem category="CONDITION" id="cc2" name="temp" type="TEMPERATURE"></DataItem>
<DataItem category="CONDITION" id="cc5" name="hardware" type="HARDWARE"></DataItem>
</DataItems>
</Path>
</Components>
</Controller>
<Systems name="syst" id="syst">
<Electric name="electric" id="el">
<DataItems>
<DataItem name="pwr_state" id="ps" type="POWER_STATE" category="EVENT"/>
</DataItems>
</Electric>
</Systems>
<Door id="d" name="door">
<DataItems>
<DataItem id="ds" category="EVENT" name="door" type="DOOR_STATE"/>
</DataItems>
</Door>
</Components>
</Device>
</Devices>
</MTConnectDevices>
```

B. ARQUIVO *FANUC - CNC3AXIS.TXT*

Este arquivo texto é um elemento que integra as configurações do Agente MTConnect quando da sua instalação. O arquivo associa um dispositivo máquina-ferramenta de três eixos com CNC Fanuc a estrutura dados descrita logo em abaixo. Seu uso é dispensável caso o arquivo *Devices.xml* tenha sido modificado manualmente.

```
<Device sampleRate="10.0" name="RomiGalaxy15M" iso841Class="1" uuid="F18i" id="d1">
<Description manufacturer="UNB" serialNumber="F18ita"></Description>
<DataItems>
<DataItem category="EVENT" id="####Id1" name="avail" type="AVAILABILITY"></DataItem>
</DataItems>
<Components>
<Axes name="Axes" id="A">
<Components>
<Rotary name="C" id="c">
<DataItems>
<DataItem category="SAMPLE" id="####cspd" name="Cact" type="ANGLE"
units="DEGREE" subType="ACTUAL"></DataItem>
<DataItem category="SAMPLE" id="####Cload" name="Cload" type="LOAD"
units="NEWTON"></DataItem>
<Constraints>
<Value>SPINDLE</Value>
<Value>INDEX</Value>
<Value>CONTOUR</Value>
<Value>UNAVAILABLE</Value>
</Constraints>
</DataItem>
<DataItem category="CONDITION" id="####Coverload" name="Coverload" type="LOAD"/>
<DataItem category="CONDITION" id="####Ctemp" name="Ctemp" type="TEMPERATURE"/>
</DataItems>
</Rotary>
<Rotary name="S1" id="b">
<DataItems>
<DataItem category="SAMPLE" id="####S1speed" name="S1speed" type="SPINDLE\_SPEED"
units="REVOLUTION/MINUTE" subType="ACTUAL"></DataItem>
<DataItem category="SAMPLE" id="####S1speedc" name="S1speedc" type="SPINDLE\_SPEED"
units="REVOLUTION/MINUTE" subType="COMMANDED"></DataItem>
<DataItem category="SAMPLE" id="S1load" name="####S1load" type="LOAD"
units="NEWTON"></DataItem>
<DataItem category="SAMPLE" id="####rfunc" name="rfunc" type="ROTARY\_MODE">
<DataItem category="CONDITION" id="####S1overload" name="S1overload" type="LOAD"/>
<DataItem category="CONDITION" id="####S1temp" name="S1temp" type="TEMPERATURE"/>
</DataItems>
</Rotary>
<Linear name="X" id="x">
<DataItems>
<DataItem category="SAMPLE" id="####Xact" name="Xact" type="POSITION"
units="MILLIMETER" subType="ACTUAL"></DataItem>
<DataItem category="SAMPLE" id="####Xcomm" name="Xcomm" type="POSITION"
units="MILLIMETER" subType="COMMANDED"></DataItem>
<DataItem category="SAMPLE" id="####Xload" name="Xload" type="LOAD"
```

```
46  units="NEWTON"></DataItem>
47  <DataItem category="CONDITION" id="####Xoverload" name="Xoverload" type="LOAD"/>
48  <DataItem category="CONDITION" id="####Xtemp" name="Xtemp" type="TEMPERATURE"/>
49  <DataItem category="CONDITION" id="####Xtravel" name="Xtravel" type="POSITION"/>
50  </DataItems>
51  </Linear>
52  <Linear name="Z" id="z">
53  <DataItems>
54  <DataItem category="SAMPLE" id="####zp" name="Zact" type="POSITION"
55  units="MILLIMETER" subType="ACTUAL"></DataItem>
56  <DataItem category="SAMPLE" id="####zcom" name="Zcomm" type="POSITION"
57  units="MILLIMETER" subType="COMMANDED"></DataItem>
58  <DataItem category="SAMPLE" id="####Zload" name="Zload" type="LOAD"
59  units="NEWTON"></DataItem>
60  <DataItem category="CONDITION" id="####Zoverload" name="Zoverload" type="LOAD"/>
61  <DataItem category="CONDITION" id="####Ztemp" name="Ztemp" type="TEMPERATURE"/>
62  <DataItem category="CONDITION" id="####Ztravel" name="Ztravel" type="POSITION"/>
63  </DataItems>
64  </Linear>
65  </Components>
66  </Axes>
67  <Controller name="Controller" id="cont">
68  <DataItems>
69  <DataItem type="MESSAGE" name="message" category="EVENT" id="####message"></DataItem>
70  <DataItem category="EVENT" id="####estop" name="estop" type="EMERGENCY\_STOP"></DataItem>
71  <DataItem type="PART\_COUNT" name="part\_count" category="EVENT" id="####part\_count">
72  </DataItem>
73  <DataItem category="CONDITION" id="####cc1" name="comm" type="COMMUNICATIONS"></DataItem>
74  <DataItem category="CONDITION" id="####cc3" name="motion" type="MOTION\_PROGRAM"></DataItem>
75  <DataItem category="CONDITION" id="####cc4" name="logic" type="LOGIC\_PROGRAM"></DataItem>
76  <DataItem category="CONDITION" id="####cc6" name="system" type="SYSTEM"></DataItem>
77  </DataItems>
78  <Components>
79  <Path id="path" name="path">
80  <DataItems>
81  <DataItem type="ACTIVE\_AXES" name="Act\_axes" category="EVENT" id="####aAxes"></DataItem>
82  <DataItem category="EVENT" id="####block" name="block" type="BLOCK"></DataItem>
83  <DataItem category="EVENT" id="####execution" name="execution" type="EXECUTION"></DataItem>
84  <DataItem category="EVENT" id="####mode" name="mode" type="CONTROLLER\_MODE"></DataItem>
85  <DataItem category="EVENT" id="####program" name="program" type="PROGRAM"></DataItem>
86  <DataItem category="EVENT" id="####line" name="line" type="LINE"></DataItem>
87  <DataItem type="PATH\_FEEDRATE" name="path\_feedrate" category="SAMPLE" id="####path\_feedrate"
88  units="MILLIMETER/SECOND" subType="ACTUAL"></DataItem>
89  <DataItem type="PATH\_FEEDRATE" name="path\_fdrovr" category="SAMPLE" id="####path\_fdrovr"
90  units="PERCENT" subType="OVERRIDE"></DataItem>
91  <DataItem type="PATH\_POSITION" name="path\_position" category="SAMPLE" id="####path\_position"
92  units="MILLIMETER\_3D" coordinateSystem="WORK"></DataItem>
93  <DataItem type="TOOL\_ID" name="tool\_id" category="EVENT" id="####tool\_id"></DataItem>
94  <DataItem category="CONDITION" id="####cc2" name="temp" type="TEMPERATURE"></DataItem>
95  <DataItem category="CONDITION" id="####cc5" name="hardware" type="HARDWARE"></DataItem>
96  </DataItems>
97  </Path>
98  </Components>
99  </Controller>
100 <Systems name="syst" id="syst">
101 <Electric name="electric" id="el">
102 <DataItems>
103 <DataItem name="pwr\_state" id="####ps" type="POWER\_STATE" category="EVENT"/>
104 </DataItems>
105 </Electric>
```

```
106  </Systems>
107  <Door id="d" name="door">
108  <DataItems>
109  <DataItem id="####ds" category="EVENT" name="door" type="DOOR\_STATE"/>
110  </DataItems>
111  </Door>
112  </Components>
113  </Device>
```

C. CLASSE DO ADAPTADOR *FOCASGATEWAY*

Está representa a principal classe do Adaptador MTConnect, ela encapsula os métodos definidos na classe que faz a comunicação com a API Focas 1, recebe os valores dos parâmetros do CNC, que são disponibilizados pela aplicação no *socket* onde o Agente MTConnect recebe os dados de fabricação e transmite via protocolo HTTP.

```
1
2  using System;
3  using System.Collections;
4  using System.Collections.Generic;
5  using System.ComponentModel;
6  using System.Windows.Forms;
7  using System.Data;
8  using System.Drawing;
9  using System.Linq;
10 using System.Text;
11 using System.IO;
12 using System.Threading;
13 using System.Net;
14 using System.Net.Sockets;
15 using System.Xml;
16
17 namespace Fanuc
18 {
19 using MTConnect;
20
21 public class FocasGateway
22 {
23
24 string DeviceIP;
25 ushort DevicePort;
26 public bool mConnected = false;
27 ushort FlibHndl = 0; //uFlibHndl=0;
28 short ret;
29
30 public FocasGateway()
31 {
32
33 }
34
35 //Abertura de conexão com o controlador da máquina para um unico Path
36
37 public bool connect(ref ushort aFlibHndl)
38 {
39 if (aFlibHndl == 0)
40 {
41
42 DeviceIP = "164.41.17.20";
43 DevicePort = 8193;
44 Console.WriteLine("Connecting to Machine at {0} and port {1}", DeviceIP, DevicePort);
45 ret = Focas1.cnc\_allclibhndl3(DeviceIP, DevicePort, 10, out FlibHndl);
```

```
46  aFlibHndl = FlibHndl;
47
48  if (ret == Focas1.EW\_OK)
49  {
50  mConnected = true;
51
52  }
53  else
54  {
55  Console.WriteLine("Error " + Convert.ToString(ret) + ": CNC Connection Failed.");
56  }
57  }
58  else
59  {
60  FlibHndl = aFlibHndl;
61  }
62  return mConnected;
63  }
64
65  //Fechamento de conexão com o servidor no CNC
66
67  public void disconnect()
68  {
69
70  Focas1.cnc\_freelibhndl(FlibHndl);
71  mConnected = false;
72  }
73
74  // Ler o avanço atual do caminho
75  public int getActualFeedRate()
76  {
77  Focas1.ODBACT b = new Focas1.ODBACT();
78  ret = Focas1.cnc\_actf(FlibHndl, b);
79  int r = 0;
80  if (ret == Focas1.EW\_OK)
81  {
82  r = b.data;
83  }
84  else
85  {
86  Console.WriteLine("Error " + Convert.ToString(ret) + ": Actual FeedRate (F) Capture Failed.");
87  }
88  return r;
89  }
90
91  //Ler o avanço fornecido durante a programação da máquina
92  public int getCommandedFeedRate()
93  {
94  Focas1.ODBMDL\_3 command = new Focas1.ODBMDL\_3();
95  ret = Focas1.cnc\_modal(FlibHndl, 103, 1, command);
96  int r = 0;
97  if (ret == Focas1.EW\_OK)
98  {
99  r = command.aux.aux\_data;
100 }
101 else
102 {
103 Console.WriteLine("Error " + Convert.ToString(ret) + ": Commanded FeedRate(F) Capture Failed.");
104 }
105 return r;
```

```
106 }
107
108 //Ler a velocidade do eixo rotacional
109
110 public int[] getSpindleSpeed()
111 {
112 Focas1.ODBACT2 b = new Focas1.ODBACT2();
113 ret = Focas1.cnc\_acts2(FlibHndl, -1, b);
114 int[] r = {0,0};
115
116 if (ret == Focas1.EW\_OK)
117 {
118 r[0] = b.data[0];
119 r[1] = b.data[1];
120 }
121 else
122 {
123 Console.WriteLine("Error " + Convert.ToString(ret) + ": Actual Spindle Speed (F) Capture Failed");
124 }
125 return r;
126 }
127
128 //Ler os dados de posição da absoluta (comandada) dos eixos controlados
129
130 public int[] getMachineAxesPos()
131 {
132 Focas1.ODBAXIS bx = new Focas1.ODBAXIS();
133 ret = Focas1.cnc\_absolute2(FlibHndl, -1, 4 + 4 * 3, bx);
134 int[] r = { 0, 0, 0 };
135 if (ret == Focas1.EW\_OK)
136 {
137 r = bx.data;
138 }
139 else
140 {
141 Console.WriteLine("Error " + Convert.ToString(ret) + ": Absolute Axis Position Capture failed",
142 "Adapter Error.");
143 }
144 return r;
145 }
146
147 //Ler os dados de posição relativa dos eixos
148
149 public int[] getRelativeAxesPos()
150 {
151
152 int[] r = { 0, 0, 0 };
153 if (ret == Focas1.EW\_OK)
154 {
155 Focas1.ODBAXIS brx = new Focas1.ODBAXIS();
156 Focas1.cnc\_relative2(FlibHndl, -1, 4 + 4 * 3, brx);
157 r = brx.data;
158 }
159 else
160 {
161 Console.WriteLine("Error " + Convert.ToString(ret) + ": Relative Axis Position Capture failed.");
162 }
163 return r;
164 }
165
```

```
166 //Ler os dados de posição da máquina
167
168 public int[] getCommandAxesPos()
169 {
170 int[] r = { 0, 0, 0 };
171 if (ret == Focas1.EW\_OK)
172 {
173 Focas1.ODBAXIS brx = new Focas1.ODBAXIS();
174
175 Focas1.cnc\_machine(FlibHndl, -1, 4 + 4 * 3, brx);
176 r = brx.data;
177 }
178 else
179 {
180 Console.WriteLine("Error " + Convert.ToString(ret) + ": Machine Axis Position Capture failed.");
181 }
182 return r;
183 }
184
185 //Fornece a distancia percorrida pela ferramenta (Não utilizada)
186 public int[] getDistanceTravel()
187 {
188
189 int[] r = { 0, 0, 0 };
190 if (ret == Focas1.EW\_OK)
191 {
192 Focas1.ODBAXIS brx = new Focas1.ODBAXIS();
193 Focas1.cnc\_distance(FlibHndl, -1, 4 + 4 * 3, brx);
194 r = brx.data;
195 }
196 else
197 {
198 Console.WriteLine("Error " + Convert.ToString(ret) + ": The capture Distance to go failed.");
199 }
200 return r;
201 }
202
203 //Metodo recupera a carga do eixo rotacional
204
205 public int[] getSpindleLoad(ref short num, ref short[] dec)
206 {
207 Focas1.ODBSPLOAD sp = new Focas1.ODBSPLOAD();
208 int[] resp = new int[2];
209 ret = Focas1.cnc\_rdspmeter(FlibHndl, 0, ref num, sp);
210
211 if (ret == Focas1.EW\_OK)
212 {
213 resp[0] = sp.spload1.spload.data;
214 resp[1] = sp.spload2.spload.data;//Valor a ser testado quanto as casas decimais
215
216 dec[0] = sp.spload1.spload.dec;
217 dec[1] = sp.spload2.spload.dec;
218
219 }
220 else
221 {
222 Console.WriteLine("Error " + Convert.ToString(ret) + ": Capture of Spindle Load failed.");
223 }
224 return resp;
225 }
```

```
226
227 //Método recupera a carga do eixos lineares
228
229 public int[] getAxesLoad(ref short[] dec)
230 {
231 Focas1.ODBSVLOAD sp = new Focas1.ODBSVLOAD();
232 short num = 3;
233 int[] resp = new int[num];
234
235 ret = Focas1.cnc\_rdsvmeter(FlibHndl, ref num, sp);
236
237 if (ret == Focas1.EW\_OK)
238 {
239 resp[0] = sp.svload1.data;//Valor a ser testado quanto as casas decimais
240 resp[1] = sp.svload2.data;
241 resp[2] = sp.svload3.data;
242
243 dec[0] = sp.svload1.dec;
244 dec[1] = sp.svload2.dec;
245 dec[2] = sp.svload3.dec;
246 }
247 else
248 {
249 Console.WriteLine("Error " + Convert.ToString(ret) + ": Capture of Axes Load failed.");
250 }
251 return resp;
252 }
253
254 //Recupera os nome do programa CNC sendo executado
255
256 public string getCurrentPrgm()
257 {
258 Focas1.ODBPRO bpg = new Focas1.ODBPRO();
259 ret = Focas1.cnc\_rdprgnum(FlibHndl, bpg);
260 string resp = "";
261 if (ret == Focas1.EW\_OK)
262 {
263 resp = "O" + Convert.ToString(bpg.data);
264 }
265 else
266 {
267 Console.WriteLine("Error " + Convert.ToString(ret) + ": NC Program capture failed.");
268 }
269 return resp;
270 }
271
272 //Captura o número da linha do programa NC sendo executado
273
274 public string getLinePrg()
275 {
276 Focas1.ODBSEQ sqn = new Focas1.ODBSEQ();
277 ret = Focas1.cnc\_rdseqnum(FlibHndl, sqn);
278 string sn = "";
279 if (ret == Focas1.EW\_OK)
280 {
281 sn = "N" + Convert.ToString(sqn.data);
282 }
283 else
284 {
285 Console.WriteLine("Error " + Convert.ToString(ret) + ": NC Program Sequence Number capture failed.");
```

```
286 }
287 return sn;
288 }
289
290 //Pega o o bloco de código em execução do programa
291
292 public string getBlockPrg()
293 {
294 char[] buf = new char[32];
295 ushort len = (ushort)buf.Length;
296 short num;
297 ret = Focas1.cnc\_rdexecprog(FlibHndl, ref len, out num, buf);
298 string sn = "";
299 if (ret == Focas1.EW\_OK)
300 {
301 for (int i = 0; i < buf.Length; )
302 {
303 if (buf[i] != '\\n')
304 {
305 i++;
306 }
307 else
308 {
309 char[] lin = new char[i];
310 int n = 0;
311 for (int p = 0; p < i; p++)
312 {
313 lin[p] = buf[p];
314 }
315
316 for (int j = 1; j < i; j++)
317 {
318 if (Int32.TryParse(lin[j].ToString(), out n) != true)
319 {
320 sn = new string(lin);
321 sn = sn.Substring(j, i - j);
322 break;
323 }
324 }
325 break;
326 }
327 }
328 }
329 else
330 {
331 Console.WriteLine("Error " + Convert.ToString(ret) + ": Program block capture failed.");
332 }
333
334 return sn;
335 }
336
337 //System - Controller mode
338
339 public string getControllerMode()
340 {
341 Focas1.ODBST st = new Focas1.ODBST();
342 ret = Focas1.cnc\_statinfo(FlibHndl, st);
343 string resp = "";
344 if (ret == Focas1.EW\_OK)
345 {
```

```
346 switch (st.aut)
347 {
348 case 0:
349 case 3: resp = "MANUAL\_DATA\_INPUT";
350 break;
351 case 5:
352 case 6: resp = "MANUAL";
353 break;
354 default: resp = "AUTOMATIC";
355 break;
356 }
357 }
358 else
359 {
360 Console.WriteLine("Error " + Convert.ToString(ret) + ": Controller Mode capture failed.");
361 }
362 return resp;
363 }
364
365 //System – Execution Status
366
367 public string getExecStatus()
368 {
369 Focas1.ODBST st = new Focas1.ODBST();
370 ret = Focas1.cnc\_statinfo(FlibHndl, st);
371 string resp = "";
372 if (ret == Focas1.EW\_OK)
373 {
374 if (st.run == 3 || st.run == 4)
375 resp = "ACTIVE";
376 else
377 {
378 if (st.run == 2 || st.motion == 2 || st.mstb != 0)
379 resp = "INTERRUPTED";
380 else if (st.run == 0)
381 resp = "STOPPED";
382 else
383 resp = "READY";
384 }
385 }
386 else
387 {
388 Console.WriteLine("Error " + Convert.ToString(ret) + ": Execution Status capture failed.");
389 }
390 return resp;
391 }
392
393 //System – Emergence Stop
394
395 public string getEmgStop()
396 {
397 Focas1.ODBST st = new Focas1.ODBST();
398 ret = Focas1.cnc\_statinfo(FlibHndl, st);
399 string resp = "";
400 if (ret == Focas1.EW\_OK)
401 {
402 switch (st.emergency)
403 {
404 case 1: resp = "TRIGGERED";
405 break;
```

```
406 default: resp = "ARMED";
407 break;
408 }
409 }
410 else
411 {
412 Console.WriteLine("Error " + Convert.ToString(ret) + ": Emergency Stop capture failed.");
413 }
414 return resp;
415 }
416
417 //Ler informações sobre a ferramenta atualmente utilizada
418
419 public int[] getToolID()
420 {
421 int[] resp = { 0, 0 };
422
423 Focas1.ODBMDL\_3 command = new Focas1.ODBMDL\_3();
424 ret = Focas1.cnc\_modal(FlibHndl, 108, 1, command);
425 if (ret == Focas1.EW\_OK)
426 {
427 resp[0] = command.aux.aux\_data;
428 }
429 else
430 {
431 Console.WriteLine("cnc\_modal failed for T: {0}", ret);
432 }
433 return resp;
434 }
435
436 //Modo de operação do eixo rotacional
437
438 public string getRotaryMode()
439 {
440 short sMode = 0;
441 ret = Focas1.cnc\_rdopmode(FlibHndl, out sMode);
442 string rtym = "";
443 if (ret == Focas1.EW\_OK)
444 {
445 switch (sMode)
446 {
447 case 1: rtym = "SPINDLE";
448 break;
449 case 2: rtym = "INDEX";
450 break;
451 case 3:
452 case 4:
453 case 5:
454 case 6: rtym = "CONTOUR";
455 break;
456 default: rtym = "UNAVAILABLE";
457 break;
458 }
459
460 }
461 else
462 {
463 Console.WriteLine("Error " + Convert.ToString(ret) + ": Rotary Mode capture failed.");
464 }
465
```

```
466  return rtym;
467  }
468
469  //Fornece mensagens geradas durante a operação da máquina
470
471  public string[] getMessages()
472  {
473  string[] m = new string[2] { "", "" };
474  try
475  {
476  Focas1.OPMSG messages = new Focas1.OPMSG();
477  ret = Focas1.cnc\_rdopmsg(FlibHndl, 0, 6 + 256, messages);
478  if (ret == Focas1.EW\_OK \&\& messages.msg1.datano != -1)
479  {
480
481 m[0] = messages.msg1.data;
482 m[1] = messages.msg1.datano.ToString();
483  }
484  }
485  catch (Exception e)
486  {
487  Console.WriteLine("A error occurred on Messages capture: '{0}'", e.Message);
488  }
489  return m;
490  }
491
492  //Numero de peças fabricadas
493
494  public int getCounts()
495  {
496  if (!mConnected)
497  return 0;
498
499  Focas1.IODBPSD\_1 buf = new Focas1.IODBPSD\_1();
500  short ret = Focas1.cnc\_rdparam(FlibHndl, 6711, 0, 8, buf);
501  int pcount = 0;
502  if (ret == Focas1.EW\_OK)
503  {
504  pcount = buf.ldata;
505  }
506  return pcount;
507  }
508
509  }// End Class FocasGateway
```

D. CLASSE DO ADAPTADOR *FANUCPATH*

Está classe faz parte do *namespace* Fanuc, do Adaptador MTConnect, e tem exclusivamente a função de atualizar o parâmetros de condições (*Condition*) definidos na especificação MTConnect com base nos alarmes que eventualmente são acionados no centro de torneamento.

```
1
2  public FanucPath()
3  {
4
5  }
6
7
8  private void addDTItem(Adapter adpter)
9  {
10 adpter.AddDataItem(mTravel);
11 adpter.AddDataItem(eTemp);
12 adpter.AddDataItem(mServo);
13 adpter.AddDataItem(mComms);
14 adpter.AddDataItem(mLogic);
15 adpter.AddDataItem(mMotion);
16 adpter.AddDataItem(mSystem);
17 adpter.AddDataItem(mHardware);
18
19 adpter.AddDataItem(cOverload);
20 adpter.AddDataItem(cTemp);
21 adpter.AddDataItem(s1Overload);
22 adpter.AddDataItem(s1Temp);
23
24 adpter.AddDataItem(xOverload);
25 adpter.AddDataItem(xTemp);
26 adpter.AddDataItem(xTravel);
27
28 adpter.AddDataItem(zOverload);
29 adpter.AddDataItem(zTemp);
30 adpter.AddDataItem(zTravel);
31
32 cons[0] = mTravel;
33 cons[1] = eTemp;
34 cons[2] = mServo;
35 cons[3] = mComms;
36 cons[4] = mLogic;
37 cons[5] = mMotion;
38 cons[6] = mSystem;
39 cons[7] = mHardware;
40 cons[8] = cOverload;
41 cons[9] = cTemp;
42 cons[10] = s1Overload;
43 cons[11] = s1Temp;
44 cons[12] = xOverload;
45 cons[13] = xTemp;
46 cons[14] = xTravel;
47 cons[15] = zOverload;
```

```
48  cons[16] = zTemp;
49  cons[17] = zTravel;
50
51  this.adpt = adpter;
52  }
53
54  //Associa o Alarme a Condition
55  protected Condition translateAlarmNo(int aNum, short aAxis)
56  {
57  Condition cond;
58
59  switch (aNum)
60  {
61  case 0: // P/S alarm - motion_program
62  case 1:
63  case 2:
64  case 3:
65  case 10: return mMotion;
66
67  case 4: // Overtravel- position
68  if (aAxis > -1)
69  {
70  if (aAxis == 1)
71  cond = xTravel;
72  else
73  cond = zTravel;
74  }
75  else
76  cond = mSystem;
77  return cond;
78  case 5: // Overheat - temperature
79  if (aAxis > -1)
80  {
81  if (aAxis == 1)
82  cond = xTemp;
83  else if (aAxis == 2)
84  cond = zTemp;
85  else
86  cond = cTemp;
87  }
88  else
89  cond = mSystem;
90  return cond;
91
92  case 6: // Servo - Servo Alarm - Load Condition
93  if (aAxis > -1)
94  {
95  if (aAxis == 1)
96  cond = xOverload;
97  else if (aAxis == 2)
98  cond = zOverload;
99  else
100 cond = cOverload;
101 }
102 else
103 cond = mServo;
104 return cond;
105
106 case 8://APC Alarms - Hardware
107 return mHardware;
```

```
108
109 case 9: // Spindle - Servo Alarm - Load Condition
110 return s1Overload;
111
112 default:
113 return mSystem;
114 }
115
116 }
117
118 // Ler mensagens de Alarmes
119
120 public void getCondition(ushort mFlibHndl, ref Adapter mAdapter)
121 {
122
123 addDTItem(mAdapter);
124
125 string[] almmsg = new string[11] { "P/S 100 ALARM", "P/S 000 ALARM",
126 "P/S 101 ALARM", "P/S ALARM (1-255)", "OT ALARM", "OH ALARM",
127 "SERVO ALARM", "System ALARM", "APC ALARM", "SPINDLE ALARM", "P/S ALARM (5000-)" };
128 Focas1.ODBDY2_1 alstatus = new Focas1.ODBDY2_1();
129 //Focas1.ODBST alerta = new Focas1.ODBST();
130 int len = System.Runtime.InteropServices.Marshal.SizeOf(alstatus);
131 short ret1 = Focas1.cnc_rddynamic2(mFlibHndl, Convert.ToInt16(-1), Convert.ToInt16(len), alstatus);
132 //short ret2 = Focas1.cnc_statinfo(mFlibHndl, alerta);
133 //int statCond = 0;
134
135 if ((ret1 == Focas1.EW_OK) && (alstatus.alarm >= 0))
136 {
137 int alnum = alstatus.alarm;
138 short axs = alstatus.axis;
139 for (short i = 0; i < 15; i++)
140 {
141 if ((alstatus.alarm & (0x01 << i)) != 0)
142 {
143
144 Condition cdtion = translateAlarmNo(alnum, axs);
145 cdtion.Add(Condition.Level.WARNING, "Warning", i.ToString(), "", "");
146 Console.WriteLine("Alarm: {0}", almmsg[i]);
147 for (short j = 0; j < cons.Length; j++)
148 {
149 if (cons[j] != cdtion)
150 {
151 cons[j].Normal();
152 }
153 }
154 }
155 }
156 }
157 else
158 {
159 mTravel.Normal();
160 eTemp.Normal();
161 mServo.Normal();
162 mComms.Normal();
163 mLogic.Normal();
164 mMotion.Normal();
165 mSystem.Normal();
166 mHardware.Normal();
167 cOverload.Normal();
```

```
168   cTemp.Normal();
169   s1Overload.Normal();
170   s1Temp.Normal();
171   xOverload.Normal();
172   xTemp.Normal();
173   xTravel.Normal();
174   zOverload.Normal();
175   zTemp.Normal();
176   zTravel.Normal();
177   }
178
179   mAdapter = this.adpt;
180
181   }
182
183   }//Fim da Classe FanucPath
```

D. CLASSE POJO DO WEB SERVICE RESTFUL: *OPCTAGSBINDING*

Classe POJO do Web Service RESTful que integra o servidor OPCWeb. Tem como função serializar dados em listas em formato XML.

```
1
2  package br.unb.graco.rest.opcgateway;
3
4  import javax.xml.bind.annotation.XmlElement;
5  import javax.xml.bind.annotation.XmlRootElement;
6  import javax.xml.bind.annotation.XmlType;
7
8
9  @XmlRootElement(name = "sinal")
10 @XmlType(name = "", propOrder={"tagName","tagValue"})
11 public class OpcTagsBinding {
12
13
14 private String tagName;
15 private boolean    tagValue;
16
17 public OpcTagsBinding(){
18
19 }
20
21 public void setTagName(String tagname){
22   this.tagName = tagname;
23 }
24
25 public void setTagValue(boolean value){
26   this.tagValue = value;
27 }
28
29 @XmlElement
30 public String getTagName(){
31  return this.tagName;
32 }
33
34 @XmlElement
35 public boolean getTagValue() {
36  return this.tagValue;
37 }
38
39 }
```

E. CLASSE DO WEB SERVICE RESTFUL - *RESTOPCCLIENT*

Classe do Web Service RESTful do servidor OPCWeb que é definida como um recurso, segundo especificação JAX-RS.

```
1
2  package br.unb.graco.rest.opcgateway;
3
4
5  import java.util.ArrayList;
6  import java.util.List;
7
8  import javax.ws.rs.Consumes;
9  import javax.ws.rs.GET;
10 import javax.ws.rs.POST;
11 import javax.ws.rs.PUT;
12
13 import javax.ws.rs.Path;
14 import javax.ws.rs.PathParam;
15 import javax.ws.rs.Produces;
16
17 import dk.opi.io.RbxIOException;
18
19
20 @Path("/tags")
21 public class RestOPCClient {
22
23 public RestOPCClient() throws RbxIOException {
24
25 }
26
27 //Métodos de leitura
28 @GET
29 @Produces("application/xml")
30 @Path("/init")
31 public List<OpcTagsBinding> initialReading() throws RbxIOException{
32 return OpcConnection.initReading();
33 }
34
35 @GET
36 @Produces("application/xml")
37 @Path("/stream")
38 public List<OpcTagsBinding> streamReading() throws RbxIOException{
39 ArrayList<OpcTagsBinding> items = OpcConnection.streamtags;
40 OpcConnection.streamtags.clear();
41 return items;
42 }
43
44 //Métodos de Escrita
45 @PUT
46 @Produces("text/plain")
```

```
47  @Consumes("text/plain")
48  @Path("/write/coolant/{turn}")
49  public String setCoolant(@PathParam("turn") String turnONOff) throws RbxIOException
50  {
51  //boolean status = true;
52
53  String status ="";
54  if (turnONOff != null){
55  OpcConnection.writeCoolant(turnONOff);
56  status = "Coolant Modified!";
57  }else {
58  status = "Coolant Not Modified!";
59  }
60
61  return status;
62  }
63
64  @PUT
65  @Produces("text/plain")
66  @Consumes("text/plain")
67  @Path("/write/cycle/{turn}")
68  public String setCycle(@PathParam("turn") String turnONOff) throws RbxIOException{
69
70  String status ="";
71  if (turnONOff != null){
72  OpcConnection.writeCycle(turnONOff);
73  status = "Cycle Changed!";
74  }else {
75  status = "Cycle Not Changed!";
76  }
77  return status;
78  }
79
80  @PUT
81  @Produces("text/plain")
82  @Consumes("text/plain")
83  @Path("/write/program/{progSource}")
84  public String setMode(@PathParam("progSource") String prgSource) throws
85  RbxIOException, InterruptedException{
86  {
87  //boolean status = true;
88
89  String status = "";
90  int prgsource = Integer.parseInt(prgSource);
91  if (prgsource > 9 \&\& prgsource < 14){
92  OpcConnection.writeOprMode(prgsource);
93  status = "Mode Changed!";
94  }else{
95  status = "Mode Not Changed!";
96  }
97  return status;
98  }
99  }
100
101 @PUT
102 @Produces("text/plain")
103 @Consumes("text/plain")
104 @Path("/write/manualop/{oprcode}")
105 public String setManualOper(@PathParam("oprcode") String oprCode) throws
106 RbxIOException, InterruptedException{
```

```
{
String status = "";
int opcode = Integer.parseInt(oprCode);
if (opcode > 29 \&\& opcode < 32){
OpcConnection.writeManOp(opcode);
status = "Operation Changed!";
}else{
status = "Operation Not Changed!";
}
return status;
}
}

@PUT
@Produces("text/plain")
@Consumes("text/plain")
@Path("/write/axes/{axis}/{d}")
public String setAxesDirection(@PathParam("axis") String axis ,
@PathParam("d") String direction ) throws RbxIOException, InterruptedException{
{
//boolean status = true;

String status = "";
int dr = Integer.parseInt(direction);
if ((axis != null) \&\& (dr  > 39 \&\& dr  < 44)){
//wServer = new OpcWriteServer();
if (axis == "x"){
// wServer.setXAxisDirection(dr);
}else if (axis == "z"){
// wServer.setZAxisDirection(dr);
}
status = "Axis Direction Changed!";
}else {
status = "Axis Direction Not Changed!";
}
return status;
}
}
}
```

F. PROGRAMA NC DE TESTE - O7103

Programa NC de teste.

```
1
2  O7103(PROGAMA TESTE ROMI)
3  N60 G21 G40 G90 G95
4  N70 T0 101(DESBASTE GERAL)
5  N80 G96 S300
6  N90 G92 S4000 M4
7  N100 G0 X51 Z150
8  N110 G71 U2 R2
9  N120 G71 P130 Q168 U0.2 W0.2 F0.2
10 N130 G0 X50.0 Z147
11 N140 G42
12 N150 G1 W-3.0 F.15
13 N151 U-5.0
14 N152 W-5.0
15 N153 W-4.0 U-5.0
16 N154 W-6.0
17 N155 G3 W-5.0 U-5.0 R5.0
18 N156 G1 W-3.0 U-5.0
19 N157 W-8.0
20 N158 W-3.0 U5.0
21 N159 W-5.0
22 N160 W-5.0 U-5.0
23 N161 G2 W-15.0 U0 R12.0
24 N162 G3 W-15.0 U0 R12.0
25 N163 G1 W-10.0
26 N164 W-5.0 U10.0
27 N165 W-5.0
28 N166 U5.0
29 N167 W-5.0
30 N168 G40 W-5.0 U5.0
31 N180 G70 P130 Q168
32 N190 (FIM DA PECA)
33 N200 M30
```

G. PROGRAMAS PYTHON PARA O SERVIDOR OPCWEB

Programas em Python com a API OpenOPC. Solução alternativa para formar o módulo *gateway* do servidor OPCWeb.

```
1
2  ----------------------------------------------
3  ACTIVE AUTOMATIC MODE
4  ----------------------------------------------
5
6  import OpenOPC
7  opc = OpenOPC.open_client('localhost')
8  opc.connect('Kepware.KEPServerEX.V5')
9  opc.write(('Galaxy.wfanuc.wedit',False))
10 opc.write(('Galaxy.wfanuc.wmdi',False))
11 opc.write(('Galaxy.wfanuc.wjog',False))
12 opc.write(('Galaxy.wfanuc.wauto',True))
13 opc.close()
14
15 ----------------------------------------------
16 ACTIVE EDIT MODE
17 ----------------------------------------------
18 import OpenOPC
19 opc = OpenOPC.open_client('localhost')
20 opc.connect('Kepware.KEPServerEX.V5')
21 opc.write(('Galaxy.wfanuc.wauto',False))
22 opc.write(('Galaxy.wfanuc.wmdi',False))
23 opc.write(('Galaxy.wfanuc.wjog',False))
24 opc.write(('Galaxy.wfanuc.wedit',True))
25 opc.close()
26
27 ----------------------------------------------
28 ACTIVE MDI MODE
29 ----------------------------------------------
30
31 import OpenOPC
32 opc = OpenOPC.open_client('localhost')
33 opc.connect('Kepware.KEPServerEX.V5')
34 opc.write(('Galaxy.wfanuc.wauto',False))
35 opc.write(('Galaxy.wfanuc.wedit',False))
36 opc.write(('Galaxy.wfanuc.wjog',False))
37 opc.write(('Galaxy.wfanuc.wmdi',True))
38 opc.write(('Galaxy.wfanuc.wmdi',False))
39 opc.close()
40
41 ----------------------------------------------
42 ACTIVE JOG MODE
43 ----------------------------------------------
44
45 import OpenOPC
46 opc = OpenOPC.open_client('localhost')
47 opc.connect('Kepware.KEPServerEX.V5')
48 opc.write(('Galaxy.wfanuc.wauto',False))
```

```
49  opc.write(('Galaxy.wfanuc.wedit',False))
50  opc.write(('Galaxy.wfanuc.wmdi',False))
51  opc.write(('Galaxy.wfanuc.wjog',True))
52  opc.close()
53
54  ------------------------------------------------
55  ACTIVE -X DIRECTION BUTTON
56  ------------------------------------------------
57
58  import OpenOPC
59  import time
60  opc = OpenOPC.open_client('localhost')
61  opc.connect('Kepware.KEPServerEX.V5')
62  opc.write(('Galaxy.wfanuc.wxm',True))
63  time.sleep(3)
64  opc.write(('Galaxy.wfanuc.wxm',False))
65  opc.close()
66
67  ------------------------------------------------
68  ACTIVE +X DIRECTION BUTTON
69  ------------------------------------------------
70
71  import OpenOPC
72  import time
73  opc = OpenOPC.open_client('localhost')
74  opc.connect('Kepware.KEPServerEX.V5')
75  opc.write(('Galaxy.wfanuc.wxp',True))
76  time.sleep(3)
77  opc.write(('Galaxy.wfanuc.wxp',False))
78  opc.close()
79
80  ------------------------------------------------
81  ACTIVE -Z DIRECTION BUTTON
82  ------------------------------------------------
83
84  import OpenOPC
85  import time
86  opc = OpenOPC.open_client('localhost')
87  opc.connect('Kepware.KEPServerEX.V5')
88  opc.write(('Galaxy.wfanuc.wzm',True))
89  time.sleep(3)
90  opc.write(('Galaxy.wfanuc.wzm',False))
91  opc.close()
92
93  ------------------------------------------------
94  ACTIVE +Z DIRECTION BUTTON
95  ------------------------------------------------
96
97  import OpenOPC
98  import time
99  opc = OpenOPC.open_client('localhost')
100 opc.connect('Kepware.KEPServerEX.V5')
101 opc.write(('Galaxy.wfanuc.wzp',True))
102 time.sleep(3)
103 opc.write(('Galaxy.wfanuc.wzp',False))
104 opc.close()
105
106 ------------------------------------------------
107 ACTIVE SINGLE BLOCK FUNCTION
108 ------------------------------------------------
```

```
109
110 import OpenOPC
111 opc = OpenOPC.open_client('localhost')
112 opc.connect('Kepware.KEPServerEX.V5')
113 valor = opc.read('Galaxy.rfanuc.singblk')[0]
114 if valor == True:
115 opc.write(('Galaxy.wfanuc.wsingblk',False))
116 else:
117 opc.write(('Galaxy.wfanuc.wsingblk',True))
118 opc.close()
119
120 ----------------------------------------------
121 ACTIVE BLOCK DELETE FUNCTION
122 ----------------------------------------------
123
124 import OpenOPC
125 opc = OpenOPC.open_client('localhost')
126 opc.connect('Kepware.KEPServerEX.V5')
127 valor = opc.read('Galaxy.rfanuc.blkdel')[0]
128 if valor == True:
129 opc.write(('Galaxy.wfanuc.wblkdel',False))
130 else:
131 opc.write(('Galaxy.wfanuc.wblkdel',True))
132 opc.close()
133
134 ----------------------------------------------
135 ACTIVE DRY RUN FUNCTION
136 ----------------------------------------------
137
138 import OpenOPC
139 opc = OpenOPC.open_client('localhost')
140 opc.connect('Kepware.KEPServerEX.V5')
141 valor = opc.read('Galaxy.rfanuc.dryrun')[0]
142 if valor == True:
143 opc.write(('Galaxy.wfanuc.wdryrun',False))
144 else:
145 opc.write(('Galaxy.wfanuc.wdryrun',True))
146 opc.close()
147
148 ----------------------------------------------
149 ACTIVE PROG TEST FUNCTION
150 ----------------------------------------------
151
152 import OpenOPC
153 opc = OpenOPC.open_client('localhost')
154 opc.connect('Kepware.KEPServerEX.V5')
155 valor = opc.read('Galaxy.rfanuc.progtest')[0]
156 if valor == True:
157 opc.write(('Galaxy.wfanuc.wprgtest',False))
158 else:
159 opc.write(('Galaxy.wfanuc.wprgtest',True))
160 opc.close()
161
162 ----------------------------------------------
163 ACTIVE COOLANT ON
164 ----------------------------------------------
165
166 import OpenOPC
167 opc = OpenOPC.open_client('localhost')
168 opc.connect('Kepware.KEPServerEX.V5')
```

```
opc.write(('Galaxy.wfanuc.wcoolauto',False))
opc.write(('Galaxy.wfanuc.wcooloff',False))
opc.write(('Galaxy.wfanuc.wcoolon',True))
opc.close()

-----------------------------------------------
ACTIVE COOLANT OFF
-----------------------------------------------

import OpenOPC
opc = OpenOPC.open_client('localhost')
opc.connect('Kepware.KEPServerEX.V5')
opc.write(('Galaxy.wfanuc.wcoolauto',False))
opc.write(('Galaxy.wfanuc.wcoolon',False))
opc.write(('Galaxy.wfanuc.wcooloff',True))
opc.close()

-----------------------------------------------
ACTIVE COOLANT AUTO
-----------------------------------------------

import OpenOPC
opc = OpenOPC.open_client('localhost')
opc.connect('Kepware.KEPServerEX.V5')
opc.write(('Galaxy.wfanuc.wcooloff',False))
opc.write(('Galaxy.wfanuc.wcoolon',False))
opc.write(('Galaxy.wfanuc.wcoolauto',True))
opc.close()

-----------------------------------------------
ACTIVE CYCLE STOP
-----------------------------------------------

import OpenOPC
opc = OpenOPC.open_client('localhost')
opc.connect('Kepware.KEPServerEX.V5')
opc.write(('Galaxy.wfanuc.wcystop',True))
opc.write(('Galaxy.wfanuc.wcystop',False))
opc.close()
```