ENG. MÁRIO LUCIO ROLOFF

Desenvolvimento de uma Ferramenta de Configuração, Monitoração e Testes para uma Rede de Sensores/Atuadores em uma Célula Autônoma de Manufatura

FLORIANÓPOLIS

2002

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

Desenvolvimento de uma Ferramenta de Configuração, Monitoração e Testes para uma Rede de Sensores/Atuadores em uma Célula Autônoma de Manufatura

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a obtenção do
grau de Mestre em Engenharia Elétrica.

MÁRIO LUCIO ROLOFF

Florianópolis, Dezembro de 2002.

Desenvolvimento de uma Ferramenta de Configuração, Monitoração e Testes para uma Rede de Sensores/Atuadores em uma Célula Autônoma de Manufatura

Mário Lucio Roloff

'Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Engenharia Elétrica, Área de Concentração em Automação e Sistemas, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.'

_____

Marcelo Ricardo Stemmer, Prof. Dr. –Ing.
Orientador

_____

Edson Roberto de Pieri, Prof. Dr.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

_____

Marcelo Ricardo Stemmer, Prof. Dr. –Ing.
Presidente

_____

Frank Siqueira, Prof. Dr.

_____

Ricardo José Rabelo, Prof. Dr.

_____

Rômulo Silva de Oliveira, Prof. Dr.

Nada mais difícil de manejar, mais perigoso de conduzir, ou de mais incerto sucesso, do que liderar a introdução de uma nova ordem de coisas; pois o inovador tem contra si todos os que se beneficiavam das antigas condições e apoio apenas tíbio dos que se beneficiarão da nova ordem.

Nicolau Maquiavel, 1459-1527

À esposa amorosa, linda e maravilhosa,
à mulher de minha vida,
à companheira de todas as horas,
Micheli Cristina Starosky Roloff.

AGRADECIMENTOS

Agradeço ao meu orientador, Prof. Dr. –Ing. Marcelo Ricardo Stemmer, pela total dedicação à arte de educar e pesquisar, pelo apoio nas tomadas de decisões, pela compreensão dos problemas que de uma maneira ou outra se fizeram presentes e acima de tudo, pelo laço de amizade criado desde as disciplinas da graduação em Engenharia de Controle e Automação Industrial até os dias de hoje, como orientador de mestrado. Espero que este relacionamento seja cada vez mais produtivo e satisfatório.

Agradeço, também, a Universidade Federal de Santa Catarina (UFSC), especialmente ao Depto. De Automação e Sistemas (DAS), que proporcionou um ótimo ambiente para o meu desenvolvimento profissional e humano. Agradeço ao Instituto WZL (*Laboratorium für Werkzeugmaschinen und Betriebslehre*), especialmente ao Prof. Tilo Pfeifer, ao Reinhard Freudenberg e ao Dominic Sack; pelo projeto de cooperação com a UFSC, viabilizando o desenvolvimento deste trabalho. Agradeço a amizade sincera destes irmãos.

Agradeço aos membros do projeto Pollux-UFSC, especialmente ao Felipe e ao Eduardo, que foram meus amigos e companheiros durante todo o mestrado.

Agradeço aos meus amigos do S2i (Sistemas Industriais Inteligentes); especialmente, ao Charles, ao Cabral, ao Fábio, ao Minasi, ao Deschamps, à Denise e ao Alexandre. Obrigado Fernando pelo apoio no processo de correção e apresentação do mestrado e pela força nos momentos mais complicados destes últimos tempos. Obrigado ao Alexandre, um grande amigo que me ajudou em todas as dificuldades deste mestrado e da vida cotidiana. Amigos, este trabalho é o resultado do esforço de uma equipe. Devo este resultado a vocês.

Agradeço aos meus pais Wolfgang (*in memorium*) e Genoveva, que nunca mediram esforços para fazer com que eu chegasse onde eu desejava chegar; e ao meu irmão André. Obrigado pela força e pelo amor. Agradeço, assim, a toda a minha família. Agradeço ao apoio da minha nova família, Lourival, Margarida e Ana. Agradeço, do fundo do meu coração, a minha esposa Micheli que esteve comigo todos os minutos desta etapa da minha vida. Obrigado pela compreensão, pelos seus conselhos; em suma, pelo seu amor.

Agradeço a Deus, meu caminho, minha verdade e minha vida.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica.

Desenvolvimento de uma Ferramenta de Configuração, Monitoração e Testes para uma Rede de Sensores/Atuadores em uma Célula Autônoma de Manufatura

Mário Lucio Roloff

Dezembro / 2002

Orientador: Marcelo Ricardo Stemmer, Prof. Dr. –Ing.
Área de Concentração: Controle e Automação Industrial
Palavras-chave: Informática Industrial, Engenharia de Software, C/C++ & Java, Sistemas Autônomos.
Número de Páginas: 93.

RESUMO: O presente trabalho aborda o desenvolvimento de um sistema para configuração, monitoração e testes de uma rede de sensores e atuadores numa célula autônoma de manufatura. O sistema desenvolvido é um aplicativo híbrido que utiliza os melhores pontos de duas das principais linguagens de programação OOP existentes e dominantes do mercado, C/C++ e Java. Como um dos objetivos primordiais deste sistema é a possibilidade de monitoração e controle da rede de sensores/atuadores remotamente, a técnica de utilizar a independência de plataforma da linguagem Java, rodando em qualquer lugar, em qualquer sistema operacional, até mesmo nos Web Browsers comerciais foi relevante. Além disso, a existência de uma arquitetura cliente/servidor desenvolvida em linguagem C/C++ (a linguagem dominante no setor industrial) e a possibilidade de integração entre Java e C/C++ através de JNI (Java Native Interface) satisfez todos os requisitos de autonomia, modularidade, padronização, intercambiabilidade, evolutividade

exigidos pelo projeto APZ (Autonomous Production Zell). O projeto APZ consiste no desenvolvimento de uma célula de produção totalmente autônoma. Assim, o sonho do fim da dependência das empresas que utilizam máquinas-ferramentas em relação aos desenvolvedores destas máquinas estaria se tornando realidade. O sistema desenvolvido é peça integrante do projeto da Célula Autônoma de Manufatura.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

Development of a configuration, monitoring and test tool for a sensors and actuators network in an Autonomous Production Cell

Mário Lucio Roloff

March / 2002

Advisor: Marcelo Ricardo Stemmer, Prof. Dr. –Ing.
Area of Concentration: Control and Automation
Keywords: Industrial Informatic, Software Engineering, C/C++ & Java, Autonomous Systems
Number of Pages: 93.

ABSTRACT: The present work proposes the development of a configuration, monitoring, and testing-system for sensors and actuators in an autonomous manufacturing cell. The system was developed using the best features of two commercially-available Object-Oriented Programming tools: C/C++ and Java. With remote control and monitoring of a network of sensors and actuators being one of the main objectives of this work, the platform-independent Java was chosen to allow operation over any Operating System, and even through Web-browsers. The presence of a client-server architecture built on C/C++ (the leading programming language in industries) and the possibility of integrating it with Java, through the Java Native Interface, satisfied all autonomy, modularity, standartization, intercambibiality, and evolutivity required by the APZ (Autonomous Production Zell) project. This project consists on the development of a completely autonomous production cell. Industries' dependence on tool-machine developers is closer to ending. The developed system is a part of the larger Autonomous Production Cells project

SUMÁRIO

LISTA DE FIGURAS

GLOSSÁRIO

ANSI – *Americam National Standard Institute*

AO – *Architecture Object*

APC – *Autonomous Production Cell*

API – *Application Programming Interface*

APZ – *Autonome Production Zell*

AWT – *Basic Interface Java Classes*

CAD – *Computer-Aided Design*

CAM – *Computer-Aided Manufacturing*

CAPP – *Computer-Aided Process Planning*

CIM – *Computer Integrated Manufacturing*

CORBA – *Common Object Request Broker Architecture*

COM – *Component Object Model*

CVS – *Control Version System*

DAS – Departamento de Automação e Sistemas

DCOM – *Distributed Component Object Model*

DFG – *Deutsche Forchungsgemeinschaft*

DLL – *Dynamic-Link Library (Windows)*

EPROM – *Eletronic Programable Read-Only Memory*

FMS – *Flexible Manufacturing System*

FTP – *File Transfer Protocol*

IEEE - *Institute of Electrical and Electronics Engineers*

IPC – *Inter-Process Communication*

JDK – *Java Development Kit*

JNI – *Java Native Interface*

JVM – *Java Virtual Machine*

LIB – *Library (Unix, GNU/Linux)*

MAP – *Manufacturing Automation Protocol*

MFC – *Microsoft Foundation Classes*

MMS – *Manufacturing Message Specification*

MTQ – *Messtecnik und Qualitate Gruppe*

OOP – *Object-Oriented Programming*

OSACA – *Open System Architecture for Controls within Automation Systems*

OSI – *Open Systems Interconnection*

RMI – *Remote Method Invocation*

RWTH-Aachen – *Rheinisch-Westfallische Technische Hochschule von Aachen*

S2i – Sistemas Industriais Inteligentes

SAM – *Sensor/Actuator Module*

SAMWeb – *SAM Web Configuration Tool*

SFB - *Sonderforschungsbereiches*

SFB368 – *Sonderforschungsbereiches 368*

SWING – *Interface Java Classes*

TCP/IP – *Transport Comunication Protocol/Internet Protocol*

TOP – *Technical Office Protocol*

UML – *Unified Modelling Language*

UFSC – Universidade Federal de Santa Catarina

VME – *VERSAmodule Eurocard*

VRML – *Virtual Reality Modelling Language*

WZL – *Werkzeugmaschinenlabor der RWTH-Aachen*

1 INTRODUÇÃO

Nos dias atuais, as exigências de personalização, de funcionalidade e de qualidade dos produtos são cada vez mais elevadas. Além disso, o tempo de produção cada vez menor é uma exigência do mercado, o que leva a um modelo de produção voltado para o atendimento das necessidades do cliente no menor tempo possível e com o mínimo de desperdício [2][42]. Estas condições requerem que os processos de manufatura tornem-se cada vez mais complexos e cada vez mais precisos. Além disso, os processos têm que ser robustos e altamente automatizados por razões técnicas e econômicas. Assim sendo, mais e mais o processo – e com isto o local de produção – deve fornecer um grau elevado de flexibilidade para suprir as necessidades dos clientes de uma maneira rápida, completa e econômica.

O planejamento e execução de projetos de manufatura envolvem um número muito grande de procedimentos que necessitam de tempo e recursos para serem realizados.

Em muitos casos, partes do processo de manufatura já estão totalmente automatizados, mas é sabido que quando se aumenta a complexidade e a automação do processo o número de paradas e interrupções na produção aumenta violentamente quando não se tem um sistema bem adaptado às exigências [2][43].

Soluções técnicas para a situação descrita acima surgem sempre acompanhadas da integração de funcionalidades na adição de um novo componente. A integração destes componentes conduziu às novas estruturas de controle de sistemas e redes de sensores/atuadores que podem cumprir com a expansibilidade e intercambiabilidade necessárias aos locais de produção. Por tudo isto, também a complexidade destas estruturas aumentou pesadamente e agora necessita ser controlada. A mudança de máquinas-ferramenta é hoje limitada a mudança de ferramentas ou de atividades que sejam similares. Mudar e expandir a funcionalidade de uma máquina-ferramenta não é uma tarefa fácil com os sistemas de controle e suporte que são dados para o usuário atualmente.

Além disso, hoje em dia, as etapas de controle de qualidade e escalonamento da produção são tarefas que são cumpridas ainda pelo usuário da máquina. Em muitos casos, o usuário não dispõe de conhecimento sobre os critérios de avaliação de qualidade e desempenho. Conseqüentemente, os critérios de avaliação da qualidade dos processos de produção e do produto são subjetivos, embora existam métricas bastante objetivas para avaliação da qualidade e desempenho.

Atualmente, a Informática Industrial é um dos campos que mais tem crescido na cadeia produtiva. Cada vez mais, a presença de equipamentos controladores, sensores, atuadores esta

presente no dia a dia dos funcionários. Com tudo isso uma questão que está sempre em discussão nesta área é a integração de todos estes dispositivos de uma maneira a garantir a intercambiabilidade e a interoperabilidade destes. Várias propostas surgiram no decorrer dos anos com o objetivo de tornar o chão-de-fábrica padronizado. Dentre as propostas, o Projeto MAP (*Manufacturing Automation Protocol*)[1] que busca seguir o modelo OSI (*Open Systems Interconnection*)[2]. Como propostas secundárias os sistemas de redes de campo, Fieldbus (ou Barramento de Campo), onde se destacam as propostas[36]: **FIP**, **PROFIBUS**, **INTERBUS-S**, **CAN**, dentre outras.

1.1 CONTEXTO DO TRABALHO

Neste contexto, vários institutos (Fig. 1.1) de pesquisa da Universidade Técnica de Aachen (**RWTH-Aachen**) e especializados em diferentes campos de atuação, desde da mecânica clássica, passando pelo controle de processos, planejamento da produção, indo até a mecatrônica juntaram-se dentro do projeto de pesquisa **SFB368** intitulado de "*Autonomous Production Cells*" (**APC**), onde buscam desenvolver os pontos primordiais para o desenvolvimento de uma célula autônoma de produção.

[1] MAP – iniciado em 1980 pela GM, tem como finalidade definir dentro do modelo OSI um ambiente de comunicação voltado para a automação da manufatura. MAP define mecanismos de comunicação entre equipamentos de chão-de-fábrica, tais como Robôs, CNCs, CLPs, computadores, etc[29].

[2] OSI - Open Systems Interconnection (OSI) é um modelo conceitual de protocolo com sete camadas definido pela ISO, para a compreensão e o projeto de redes de computadores. Trata-se de uma padronização internacional para facilitar a comunicação entre computadores de diferentes fabricantes[28].

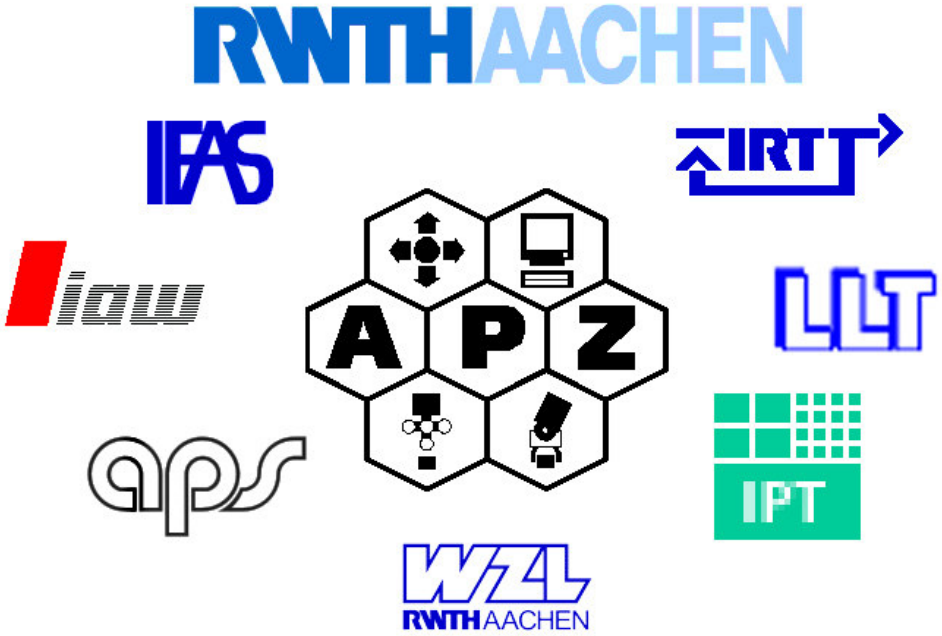


Fig. 1.1: Institutos alemães integrantes do projeto.

O projeto **SFB368** visa desenvolver uma célula de manufatura que seja totalmente independente e totalmente padronizada do ponto de vista industrial. Dentro do **SFB368** tem-se uma equipe que trabalha no desenvolvimento de uma interface padronizada para a instalação, manutenção e remoção de sensores e atuadores em uma célula de manufatura, o projeto **SAM** (*Sensor/Actuator Module*) [6].

Além dos institutos alemães citados acima, um convênio de cooperação entre a **Universidade Federal de Santa Catarina** (**UFSC**), através do **Departamento de Automação e Sistemas** (**DAS**), com a ***Rheinisch-Westfalische Technische Hochschule von Aachen*** (**RWTH-Aachen**), através do ***Werkzeugmaschinenlabor der RWTH Aachen*** (**WZL**), foi proposto. O convênio nasceu da longa troca de experiência de pesquisadores brasileiros que desenvolveram trabalhos na **RWTH-Aachen**. Neste âmbito, a equipe brasileira é responsável pelo desenvolvimento de duas partes da rede de Sensores/Atuadores (SAM) da Célula Autônoma de Produção:

- Desenvolvimento de um sistema inteligente para medição e classificação do desgaste de flanco de ferramentas de corte em máquinas-ferramenta.

- Desenvolvimento de um sistema ergonômico para configuração, monitoração e testes da rede de sensores e atuadores através de uma rede Intranet/Internet. Este tópico é o fruto desta dissertação de mestrado.

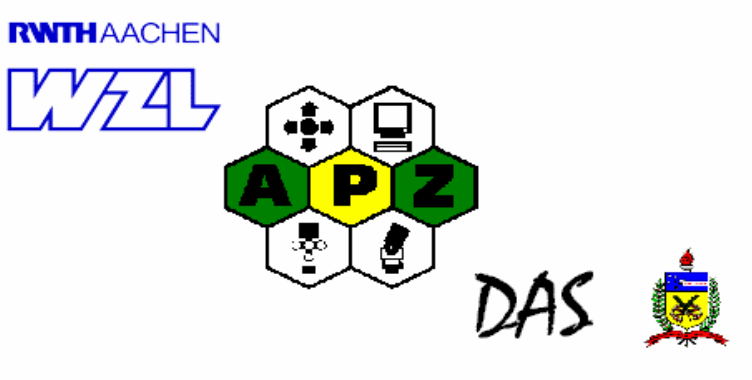


Fig. 1.2: Projeto de Cooperação Brasil-Alemanha.

Os próximos tópicos deste capítulo apresentam-se os objetivos traçados no desenvolvimento desta dissertação de mestrado.

1.2 MOTIVAÇÃO

A motivação serve como um relato histórico dos fatos acontecidos desde o surgimento da oportunidade até a sua completa solução através desta dissertação. O item sobre o sistema de visualização descreve as vantagens da utilização de um sistema gráfico com o objetivo de diminuir a complexidade do sistema. E os objetivos específicos destacam os pontos que justificam esta dissertação e fazem uma breve introdução sobre os temas que serão detalhados nos próximos capítulos.

A idéia do desenvolvimento da ferramenta ergonômica, visual e remota para a configuração, monitoração e testes de uma rede de sensores/atuadores em uma célula autônoma de manufatura surgiu da experiência vivida durante a execução do Projeto de Final de Curso da graduação em Engenharia de Controle e Automação entre o agosto de 1999 e março de 2000 em Aachen, Alemanha.

No Instituto de Máquinas-Ferramentas e Engenharia de Produção (**WZL**) da Universidade Técnica de Aachen (**RWTH-Aachen**) existe o projeto que consiste, de uma maneira resumida, no desenvolvimento de um sistema totalmente autônomo de manufatura. Dentro deste contexto do projeto da Célula Autônoma de Produção existe a parte que tem como objetivo o desenvolvimento de uma rede de Sensores/Atuadores para a célula de produção autônoma. Nexte contexto, independentemente do tipo, modelo e fabricante dos dispositivos que compõem a rede, estes dispositivos necessitam ser controlados e monitorados.

O projeto tem como foco principal à "**autonomia**" com relação aos fabricantes, fornecedores e protocolos dos dispositivos. A idéia de acabar com a dependência do sistema da presença humana junto à máquina para a configuração, monitoração e testes da rede de Sensores/Atuadores também vai de encontro com o objetivo do projeto. Assim, surgiu a idéia do desenvolvimento de uma ferramenta para controlar e monitorar os Sensores/Atuadores existentes numa Célula Autônoma de Manufatura remotamente, ou melhor, através da rede Intranet ou até mesmo da Internet.

Através da experiência adquirida durante o Projeto de Final de Curso com a arquitetura de rede industrial MAP e seu protocolo para a camada de aplicação MMS (*Manufacturing Message Specification*) em redes industriais e em programação em Java e C/C++ para sistemas do Cliente/Servidor para comunicação em redes, se propôs que seria possível o desenvolvimento de tal sistema.

1.2.1 VISUALIZAÇÃO, CONFIGURAÇÃO E TESTES

O sistema para visualização, configuração e testes que foi originalmente desenvolvido para a rede de Sensores/Atuadores é baseado em uma tela MS-Dos de comandos onde o usuário possui pouca interatividade com o sistema e somente serve como um sistema de *debug* do sistema como um todo. Para utilizar o sistema era necessário conhecer em detalhes o funcionamento e a filosofia utilizada dentro do módulo de Sensores/Atuadores (**SAM)** [6] do projeto da Célula Autônoma de Manufatura[9]. Além de tudo isto, o sistema era pouco robusto e com uma interface nada amigável.

Analisando o estado da arte descrito acima optou-se pelo estudo e desenvolvimento de um sistema que permitisse, de uma maneira fácil e ergonômica, a visualização dos Sensores/Atuadores existentes em uma determinada máquina-ferramenta, e que estes dispositivos pudessem ser acessados e seus valores alterados pelo usuário especializado. Além disso, procurou-se que um *debug* ou teste dos valores configurados para os Sensores/Atuadores pudesse ser feito previamente de maneira a não permitir que falsos valores pudessem ser usados.

1.3 OBJETIVOS

Neste item serão apresentados os principais objetivos para o desenvolvimento desta dissertação de mestrado e do sistema para configurar, monitorar e testar a rede de Sensores/Atuadores da Célula Autônoma de Produção:

- Migração do sistema para configuração, monitoração e testes para um aplicativo que desempenhe todas as tarefas pertinentes à rede de Sensores/Atuadores em uma interface ergonômica, fácil e clara e também que este *setup* possa ser realizado remotamente.

- O aplicativo resultante da dissertação deve ser capaz de ler dados dos arquivos de configuração dos SAM *servers*. E também, o aplicativo deve ser capaz de escrever novos valores para os sensores e atuadores, ou seja, deve gerar um novo arquivo de configuração SAMConfig.dat e com isto deve realizar o debug dos novos valores para os sensores e atuadores, ou seja, observar se estão ou não dentro dos valores requeridos. Um outro requisito para o aplicativo é que deve ser possível realizar a monitoração *in time* de valores de determinados sensores/atuadores através de um gráfico.
- O aplicativo deve ser um sistema que seja totalmente independente da plataforma, do sistema operacional e também deve ser construído utilizando-se apenas os recursos padronizados das linguagens de programação utilizadas.
- O aplicativo fruto da dissertação é um sistema híbrido que une as pontencialidades das duas linguagens mais conhecidas e estudadas atualmente, Java e C++. Aonde a comunicação entre o *front-end* do aplicativo (desenvolvida em Java) com a rede de Sensores/Atuadores (desenvolvida em C/C++) será feita através de JNI (*Java Native Interface*).
- A dissertação tem como objetivo também apresentar o método de trabalho para o desenvolvimento de sistemas usado durante a elaboração da dissertação e que preocupa-se com cada passo no desenvolvimento de um novo aplicativo e também com os passos para a integração de novas funcionalidades e soluções para sistemas já existentes.

O desenvolvimento do projeto ocorreu no Grupo S2i (Sistemas Industriais Inteligentes), que faz parte do DAS (Departamento de Automação e Sistemas). Para tal, foi utilizado um computador PC Pentium III 550 Mega Hertz com 128 MbRAM. O computador estava totalmente equipado com o hardware e o software necessários para a criação, desenvolvimento e implementação do sistema.

Os testes de funcionamento e integração do sistema à Célula Autônoma de Manufatura [9], foram realizados no Instituto WZL na Universidade Técnica de Aachen (**RWTH-Aachen**) na Alemanha. Esta etapa foi realizada durante uma visita técnica realizada durante os meses de janeiro e fevereiro de 2002.

1.4 ESTRUTURA DO DOCUMENTO

Após uma breve introdução e apresentação dos objetivos da dissertação, apresentados neste Capítulo 1, o Capítulo 2 faz uma descrição do problema abordado, contextualizando e justificando a dissertação dentro dos conceitos que envolvem este trabalho, que são: Sistemas Autônomos de

Produção, rede de Sensores/Atuadores em Sistemas Autônomos de Produção, Controle, Monitoração e Integração da Produção. Enquanto no Capítulo 2 a ênfase foi dada mais para a parte de manufatura e informática industrial, o Capítulo 3 busca uma contextualização da dissertação dentro dos conceitos de computação envolvidos na elaboração do sistema fruto desta dissertação. Os tópicos abordados neste capítulo são: um abordagem rápida sobre linguagens orientadas a objetos e sua utilização em ambiente industrial, também uma discussão sobre as linguagens C/C++ e Java utilizadas no desenvolvimento do sistema, com um aprofundamento dado à técnica JNI (*Java Native Interface*)[25] e a estratégia usada para o desenvolvimento do sistema. O Capítulo 4 trata especificamente do projeto e implementação do sistema resultante desta dissertação de mestrado, o ***SAM Web Configuration Tool***, ou simplesmente **SAMWeb**. O capítulo apresenta uma introdução, as justificativas, módulos, testes e os resultados alcançados. O Capítulo 5 preocupa-se basicamente em apresentar os resultados experimentais obtidos durantes simulações do sistema em funcionamento. E finalmente o Capítulo 6 apresenta as conclusões do trabalho e também as perspectivas para trabalho futuros [9] [9] [9].

2 SENSORES/ATUADORES EM SISTEMAS AUTÔNOMOS DE PRODUÇÃO

Neste capítulo serão apresentados os conceitos referentes ao sistema de configuração, monitoração e testes da rede de Sensores/Atuadores. Busca-se aqui uma contextualização e justificação do porque desenvolver tal dissertação de mestrado. Levanta-se deste as razões históricas pela busca do ser humano pelo total controle da produção até questões que levam em conta a melhoria da qualidade da produção.

2.1 EVOLUÇÃO DOS SISTEMAS DE MANUFATURA

Neste tópico descreve-se brevemente a evolução dos sistemas de manufatura através dos tempos e o estado da arte da manufatura atualmente.

O que se entende por manufatura? Segundo o dicionário Michaelis: "Processo ou trabalho de fazer artigos ou quaisquer produtos a mão ou com maquinaria; especialmente quando prosseguido sistematicamente e com divisão do trabalho; fabricação"[1]. Em outras palavras: "manufatura é a arte de transformar artigos e/ou conhecimento em alguma coisa que tenha utilidade e que desempenhe uma determinada função!".

Com a revolução industrial o ser humano possui o desejo pelo total controle e conhecimento do sistema de produção. Até algumas décadas atrás, este "controle" era conseguido com a presença humana em cada etapa da produção. Neste caso, o ser humano atuava diretamente no controle do processo ou então se restringia a mero expectador do funcionamento do processo.

Atualmente vive-se o que se convencionou chamar de **Terceira Revolução Industrial** [41], que apresenta um novo período no mundo industrial.

A **Primeira Revolução Industrial**[41] começou com o surgimento das máquinas-ferramenta, com a criação das fábricas e com o movimento das pessoas das fazendas para as cidades. Nos dias de hoje, apenas uma pequena parcela da população trabalha nas fazendas. Todavia, a produtividade continua a aumentar a cada dia. A quantidade de mão-de-obra direta em fábricas tem continuamente decrescido da mesma maneira. No entanto, isto não significa que menos pessoas estão trabalhando somente na agricultura ou na indústria de produção de alimentos; isto significa que a mão-de-obra foi significativamente reduzida, devido à introdução de um equipamento dedicado, freqüentemente automatizado e também com a evolução e descoberta de novos métodos e tecnologias.

A **Segunda Revolução Industrial**[41] começou no início do século XIX, com o advento das linhas de montagem e do conceito Ford da produção em massa. Conceito muito bem sedimentado em uma das mais famosas frases de Henry Ford [8]: "Faço carros de todas as cores, desde que sejam pretos!". Que relata o desejo da produção em massa e em série. Sistemas grandes e caros, chamados de **linha *transfer***, fizeram parte daquela tendência. Estes sistemas, que tiveram o desenvolvimento completo nos anos 40, eram caracterizados por mecanismos automáticos de manuseio de material a partir dos quais o termo **automação** evoluiu. Este tipo de automação recebe hoje o nome de **fixa**, em contraste com a automação **flexível**, que inclui máquinas programáveis. Esse tipo de automação ainda encontramos sendo aplicada em muitos casos, como por exemplo, nas montadoras automobilísticas.

A **Terceira Revolução Industrial** ocorrida na segunda metade do século XX até os dias de hoje, é tão dramática quanto as anteriores. Esta revolução consiste no uso de computadores para controlar tanto processos como sistemas inteiros, incluindo sistemas de informação. O controle da cadeia de produção não necessita da presença direta do ser humano no setor de produção. Com a evolução da computação e em especial das redes de computadores, hoje em dia, um ser humano detém o controle de dezenas, centenas e até mesmo, milhares de processos remotamente através de uma Manufatura Integrada por Computador (**CIM**). Todavia, aplicar o conceito de Manufatura Integrada por Computador (**CIM**)[43] na indústria, ou até mesmo começar uma nova planta utilizando-se a filosofia do CIM não é nada trivial e vários projetos e estudos vem trabalhando com o objetivo de reduzir os "traumas" causados por esta constante busca pelo sistema de produção perfeito.

2.2 AUTOMATIZAÇÃO E INFORMÁTICA INDUSTRIAL

Nos últimos anos, os esquemas de produção têm sofrido mudanças drásticas. Por um lado isto ocorre devido à crescente competição entre as empresas e por outro lado devido ao enorme desenvolvimento das tecnologias que envolvem microprocessadores, robôs, controladores lógico programáveis, redes de comunicação, inteligência artificial, máquinas de controle numérico, etc.

Com esta abstração todas as estratégias de produção visam um conjunto de objetivos: redução dos custos, aumento da qualidade, disponibilidade e inovação. Na realidade, e se for entendido nas suas diversas vertentes, esse conjunto de objetivos corresponde a um objetivo maior: **o aumento da competitividade[42].** Não é correto dizer que o objetivo primordial é o aumento da produtividade. De que serve uma alta produtividade se não existem clientes para os produtos?

Fig. 2.1: Busca pelo melhor caminho para atingir o sucesso

É também comum confundir os objetivos com os meios. Sendo o objetivo principal o aumento da competitividade (em razão dos fatores citados anteriormente) os meios são eventualmente automatizar, informatizar, robotizar, equipar, racionalizar, organizar, etc.

Com esta busca constante pela competitividade, a tendência é o crescimento da informatização das empresas e organizações que por um lado permite acelerar os processos de fabricação ou de oferecimento de um serviço e, por outro lado, cria uma nova necessidade de padronizar como as informações serão trocadas [2]. Assim a informática industrial busca permitir que as informações possam ser trocadas, sem problemas, por todo o setor produtivo.

A revolução da informática tem sido responsável pelo elevado grau de sofisticação no qual todos os setores se encontram atualmente. Com o surgimento dos computadores no final da década de 40 do século XX e sua extraordinária evolução nas décadas seguintes, a migração desta novidade para o chão-de-fábrica não poderia demorar, nem muito menos ficar à parte. A informática industrial surgiu com o objetivo primordial de adaptar os computadores, a computação, que era realizada em ambientes ditos "perfeitos", para ambientes que não poderiam receber esta denominação. Um ambiente industrial está extremamente sujeito à sujeira, ruídos de todos os tipos, interferências eletromagnéticas, etc.

2.3 SISTEMAS AUTÔNOMOS DE PRODUÇÃO

2.3.1 MOTIVAÇÃO E JUSTIFICATIVAS

Com a revolução industrial, a busca pela criação e desenvolvimento de sistemas totalmente autônomos se fez presente. No início a razão era a falta de mão-de-obra especializada e a dificuldade em operar máquinas inseguras, imprecisas, nada robustas e que exigiam um grande esforço dos operadores. Com o passar dos anos, séculos, as máquinas tornaram-se cada vez mais

robustas, eficientes, rápidas, mas a complexidade foi aumentando juntamente com a produtividade, e o operador passou a precisar compreender e se especializar cada vez mais no funcionamento de uma máquina específica de um fabricante projetada para uma determinada função[9].

As justificativas para a pesquisa no desenvolvimento de sistemas autônomos de produção também são muito importantes economicamente. Até os dias atuais, um chão-de-fábrica poderia, e ainda pode, ser visto como uma miscelânea de protocolos, máquinas, redes e sistemas parcialmente ou completamente diferentes que não conseguem se comunicar de maneira simples, rápida e eficiente. Através de um esforço internacional reunindo várias empresas e instituições, se vêm, nos últimos anos, trabalhando de maneira a padronizar os sistemas de produção e a comunicação no setor industrial. Muitos interesses por parte de todos os lados estão envolvidos. Com isso, o surgimento de uma política padronizada para a indústria ainda é vista como uma utopia. Entretanto, no dia em que esta política tornar-se comum, os fabricantes e fornecedores de equipamentos estarão livres para poderem escolher os melhores equipamentos que se ajustam às suas necessidades e não estarão mais presos à problemas de compatibilidade entre este ou aquele sistema. Isto acarretará sobretudo uma redução nos custos dos equipamentos, uma vez que a concorrência se tornará muito mais efetiva.

Além do aspecto econômico apresentado no parágrafo anterior, o aspecto técnico também é importante no desenvolvimento de sistemas autônomos. Com a total independência quanto a marca e fornecedores de equipamentos, o fabricante estará livre para a escolha do melhor sistema que se adapta às suas necessidades e que apresente e melhor relação custo/benefício e produtividade.

2.3.2 ATIVIDADES NA INTEGRAÇÃO DA MANUFATURA

Atualmente existem grupos, comunidades, centros de pesquisa e projetos em diversos países que buscam o desenvolvimento de um sistema de manufatura totalmente autônomo, como: o projeto DECOR – 8486 *Decentralised and Collaborative Production Management via Enterprise Modelling and Method Reusage* que tem como tópicos fundamentais o desenvolvimento de sistemas de tomada de decisões, modelagem empresarial, gerenciamento da produção e células autônomas de produção. Este projeto está sob a supervisão do Prof. Luis Tadeu Almeida da FORDESI de Lisboa [44]. Um outro exemplo de esforço internacional na busca pela evolução dos sistemas de manufatura é o consórcio internacional para avanço da manufatura (http://www.cam-i.org), ou *Consortium for Advanced Manufacturing Internacional (CAM-I)* formado por uma parceria entre empresas, universidades, institutos e acadêmicos que tem como objetivo solucionar os problemas que são comuns ao grupo.

O NUMA (Núcleo de Manufatura Avançada) é um centro de excelência em pesquisa fomentado pelo Ministério de Ciência e Tecnologia, através do PRONEX - Programa de Apoio à

Núcleos de Excelência, e agrega pessoas de diferentes áreas de conhecimento, permitindo o desenvolvimento de trabalhos interdisciplinares. Seu foco está na realização de pesquisas integradas em manufatura, utilizando o potencial da Tecnologia de Informação como ferramenta. O NUMA visa contribuir para que a indústria instalada no Brasil possa competir a nível mundial, fazendo frente à abertura de mercado e a globalização da economia. No NUMA as empresas encontram pessoas que podem auxiliá-las nas tomadas de decisão. Este centro conta também com a parceria do instituto WZL da RWTH-Aachen.

O Projeto SFB368 [9], ou *Autonomous Production Cells* – **APC** que é um projeto de pesquisa alemão financiado pelo DFG (*Deutsche Forschungsgemeinschaft*), fundação de pesquisa alemã. O tópico seguinte trata especificamente sobre o projeto **APC** o qual é o projeto onde se encontra inserida a dissertação de mestrado.

O objetivo aqui não é apresentar ou se alongar na discussão sobre sistemas autônomos de produção. A discussão deste ponto exige um estudo complexo e mais aprofundado do tema e isto não é o foco principal desse trabalho. Maiores informações sobre este tema dentro do projeto **APC** podem ser conseguidas na leitura de outras dissertações de mestrado ou artigos produzidos tanto pelo S2i-DAS como pelo WZL.

2.4 NECESSIDADES DE UM SISTEMA DE CONTROLE DA PRODUÇÃO

O sistema de controle para a máquina-ferramenta deve ser um sistema aberto que disponha da capacidade de integrar controles adicionais e também funções de planejamento. Conseqüentemente as seguintes características estruturais para o sistema são requeridas:

- **Suporte aos sistemas distribuídos:** no caso de algumas tarefas, que exigem muita capacidade de processamento (como no controle de processo) para um sistema de controle simples trabalhar sozinho, será necessário dividir esta macro-tarefa em tarefas menores, ou múltiplos processos, que possam ser executados paralelamente. Isto significa, que um sistema distribuído deve ser suportado pela estrutura do sistema de controle.
- **Sistema de Hardware Flexível:** expansão do sistema para desempenhar tarefas adicionais freqüentemente significa adicionar novos componentes de hardware (por exemplo sensores). Por esta razão o usuário deve ser capaz de adicionar, remover ou mudar os componentes de hardware da maneira mais fácil possível.
- **Sistema de Software Seguro e Flexível:** o usuário deve ser capaz de adaptar o sistema de software para a tarefa de manufatura atual, isto é, ele deve ter a possibilidade de adicionar, remover ou mudar componentes do sistema de software (por exemplo,

módulos de comando ou controle). O sistema também deve controlar os direitos de acesso às informações para prevenir "deadlocks" e divergências.

Com respeito a estes requisitos é necessário construir um sistema aberto que siga o significado proposto pela definição da IEEE [3] onde: "Um sistema aberto fornece as potencialidades que permitem aplicações corretamente desenvolvidas funcionarem em uma variedade de vendedores múltiplos, além disso, interoperar com outras aplicações dos sistemas e ainda mais, apresentam um estilo consistente de interação com o usuário".

2.5 A REDE DE SENSORES/ATUADORES EM UM SISTEMA AUTÔNOMO DE PRODUÇÃO

Neste tópico apresenta-se uma breve introdução sobre as redes industriais. Em seguida passa-se para uma descrição sobre Sensores/Atuadores, e também sobre Sensores/Atuadores inteligentes.

2.5.1 REDES DE COMUNICAÇÃO

Quando se trabalha na fabricação de um certo produto, põe-se em jogo uma série de etapas e processos da cadeia produtiva, dedicadas à manutenção ou ao aprimoramento do produto. A implementação dessas etapas através de processos com maior ou menor grau de automatização fica a critério da empresa. Esta tendência de informatização da empresa traz uma nova necessidade com respeito ao modo como as informações fluirão através dela.

Dentro deste contexto, nos últimos anos um esforço considerável tem sido feito com o objetivo de definir arquiteturas de comunicação que satisfaçam as características e os requisitos de compartilhamento de recursos, evolutividade, garantia de tempo de resposta, robustez, flexibilidade, etc. Como resultado deste esforço, nos Estados Unidos da América, a General Motors lançou em 1980 a iniciativa MAP (*Manufacturing Automation Protocol*) [34][33][28]que, obedecendo ao modelo de referência OSI (*Open Systems Interconnection*) [34][29], pretende definir uma norma de comunicação para ambientes industriais de produção.

Paralelamente à iniciativa MAP e também nos Estados Unidos, um consórcio liderado pela BOEING lançou a iniciativa TOP (*Technical Office Protocol*) com os mesmos objetivos do MAP, todavia, especialmente direcionado para as áreas técnicas e administrativas da empresa [12].

Esta diversidade de tentativas de padronizações para a comunicação em ambientes fabris, deve-se em muito à diversidade das características de cada classe ou nível da empresa. Em um

nível administrativo há um menor número de estações[3] por setor, enquanto que o seu custo individual é alto. Em um nível de componente tem-se o contrário, o número de estações por setor é maior, contudo, o seu custo é menor devido à menor complexidade do equipamento [36][35].

Assim, é preciso assumir a realidade de que não existe uma rede única que poderia atender a todas as necessidades de todas as classes ou níveis de atividades existentes em uma fábrica. A solução, de fato, é a adoção de várias redes interconectadas, cada rede servindo de suporte à comunicação no contexto de uma ou diversas atividades [37].

Além do mais, assim como os equipamentos precisam se adaptar ao meio industrial, as redes de computadores também precisam ser projetadas de maneira a não sofrerem danos causados pelo ambiente industrial.

Com isto, pode-se imaginar que o objetivo de interligar os equipamentos no chão-de-fábrica deve ter surgido poucos segundos após a idéia de migrar estas máquinas para substituírem os seres humanos nas tarefas repetitivas de controle, monitoração e configuração do processo. A pesquisa com o objetivo de desenvolver redes industriais que satisfaçam as mais diferentes exigências nos mais variados ambientes continua exaustivamente até os dias atuais. Muitas tentativas tiveram sucesso em determinados casos, e muitas outras fracassaram. Hoje alguns padrões estão presentes no mercado e apresentam-se até o momento como a solução para os problemas de interligação de equipamentos no ambiente industrial, desde sensores/atuadores, CLPs, até computadores paralelos de grande capacidade, passando inclusive por equipamentos *wireless*.

2.5.2 SENSORES/ATUADORES

Nos dias atuais sensores programáveis inteligentes para medição, aquisição de dados, processamento de dados e comunicação fornecem uma plataforma satisfatória para a modularização de hardware/software. A tendência é passar de um modelo centralizado para o processo de medição para módulos de computação e aquisição distribuídos. O que pode ser conseguido com esta modularização é que grandes tarefas em grandes projetos possam ser processadas em menores intervalos de tempo. Adicionalmente, funcionalidades para adicionar, remover ou modificar tornam-se mais fáceis. Esta é uma importante vantagem para o projeto da Célula Autônoma de Produção: a fácil capacidade de modificação.

De maneira a definir os requisitos para a integração de Sensores/Atuadores, deveremos analisar o problema do "ponto de vista do cliente". Do ponto de vista do usuário, ou do configurador do sistema, as seguintes características devem ser identificadas: [6]

[3] Estação é caracterizada por ser qualquer dispositivo usado para realizar o trabalho. No setor administrativo entende-se por estação o computador que o contador utiliza para fazer os balanços da empresa. Já no setor de

- '***Plug-And-Play***': o crescimento e o número de variáveis dos componentes do sistema tornam desejável que a adição ou troca de equipamentos seja feita de modo '*plug-and-play*'. Desta forma, o usuário ou configurador do sistema pode instalar de uma maneira simples e fácil sensores e atuadores.
- **Livre escolha do hardware**: o desenvolvedor do sistema deve ser livre na escolha dos componentes por preço e funcionalidade, não tendo nenhuma preocupação quanto à maneira como estes dispositivos irão se comunicar.
- **Suporte ao usuário**: o usuário deve possuir um suporte adequado durante a configuração, teste e debug do sistema.

Do ponto de vista do sistema de controle é necessário que os seguintes requisitos possam ser identificados:

- **Interface Padronizada**: uma interface padronizada sobre as informações dos Sensores/Atuadores deve ser desenvolvida, escondendo a diversidade da tecnologia de comunicação atrás de uma série de parâmetros transparentes para o usuário.
- **Requisitos de tempo-real**: o sistema de Sensores/Atuadores não deve somente fornecer logicamente resultados adequados e corretos, mas eles devem fornecer os resultados também no tempo certo. Em outras palavras, os resultados devem ser fornecidos antes do *deadline*, especialmente no caso de serviços para controle e comando de sistemas.
- ***Watchdog***: o sistema de sensores/atuadores (S/A) deve controlar a si mesmo e também notificar o usuário quando uma falha ocorre.

Tendo como base as justificativas apresentadas acima, tanto do ponto de vista do usuário ou cliente, como do ponto de vista do sistema de controle, o emprego de Sensores/Atuadores torna-se primordial no desenvolvimento de uma Célula Autônoma de Produção. Essencial porque é através desses dispositivos que se pode efetivamente conhecer o "estado do sistema" em determinado instante. Além disso, estes dispositivos permitem uma melhoria na qualidade e na produtividade do sistema de maneira a facilitar o manuseio da máquina-ferramenta e ainda mais, atuam na segurança não somente do equipamento, como dos trabalhadores, avisando em caso de emergência e atuando de maneira a corrigir erros.

2.5.2.1 SENSORES & ATUADORES INTELIGENTES

A introdução de microprocessadores e microcontroladores em sistemas de sensoriamento e atuação tornou possível a realização de uma série de funções de forma distribuída. Tais funções eram resolvidas anteriormente em uma central de processamento. Com a evolução e a distribuição

produção, uma estação pode ser entendida como um torno CNC.

da "inteligência" para os dispositivos periféricos estas funções podem atualmente ser desenvolvidas no próprio dispositivo, que deve apenas possuir uma interface de comunicação para envio e recebimento de dados referentes ao estado do sistema. Tais dispositivos microcontrolados ou microprocessados recebem o nome de Sensores & Atuadores Inteligentes.

2.5.2.1.1 Sensores Inteligentes

A complexidade requerida de um sensor é muito dependente da função a que ele está destinado. Nos modernos sistemas de sensoriamento uma configuração dos dispositivos do sensoriamento é possível. De uma maneira geral, as principais funções desempenhadas por sensores inteligentes são [33]:

- Aquisição de dados utilizando transdutores já disponíveis.
- Digitalização e codificação de dados.
- Filtragem digital de sinais.
- Compactação de informações e eliminação de redundâncias.
- Calibração do sensor.
- Supervisão de valores limite.
- Correção automática de erros sistemáticos (não linearidade, dependências de temperatura (*Drift*), etc...).
- Cálculo de grandezas derivadas do valor medido (valor médio, desvio padrão, etc.).

Para se programar estas funções nos sensores inteligentes pode-se utilizar dois meios: pode-se carregar as funções gravando em uma EPROM ou fazendo o *download* do programa através de uma interface. Nos dias atuais já é comum a existência de sensores inteligentes não só microprocessados, mas que possuem um sistema operacional que possibilita a comunicação de uma maneira muito mais fácil [34][37].

2.5.2.1.2 Atuadores Inteligentes

Enquanto os sensores trabalham com o objetivo de leitura e tratamento das informações do ambiente os atuadores são responsáveis por agir sobre o ambiente. Os atuadores são dispositivos que transformam sinais elétricos de tensão ou corrente em outras grandezas físicas como [33]:

- Velocidade.
- Força.
- Pressão.
- Rotação/Translação...
- Temperatura, etc...

Assim como foi feito para os sensores, é possível apresentar de uma maneira didática a estrutura de um atuador sem levar em consideração um alto nível de detalhamento.

A utilização de Sensores/Atuadores inteligentes vem crescendo cada dia mais devido a busca pela automatização completa do chão-de-fábrica e a necessidade de fazer isto com baixo custo.

2.5.3 REDE DE SENSORES/ATUADORES

As redes de Sensores/Atuadores estão sendo cada vez mais utilizadas nos sistemas industriais por sua tecnologia reduzir drasticamente os custos, pesquisas indicam que cerca de 15% a 40% das despesas atribuídas a cabos, comissionamento, montagem e manutenção dos sistemas de controle, podem ser diminuídos ou eliminados [37]. A figura 2.2 exemplifica a arquitetura tradicional.

A tecnologia *fieldbus* [12] [fig. 2.3], que trata da integração desses dispositivos, baseia-se na passagem da arquitetura tradicional de controle [36] para uma arquitetura mais enxuta e descentralizada no que diz respeito à conexão dos elementos de campo aos seus controladores [33].

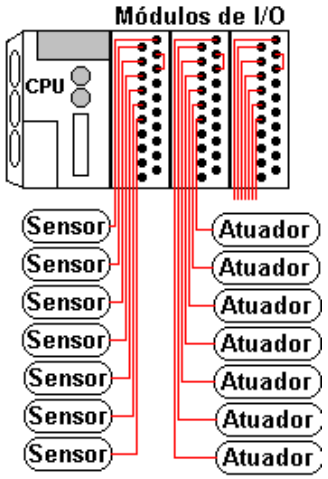


Fig. 2.2: Arquitetura Tradicional

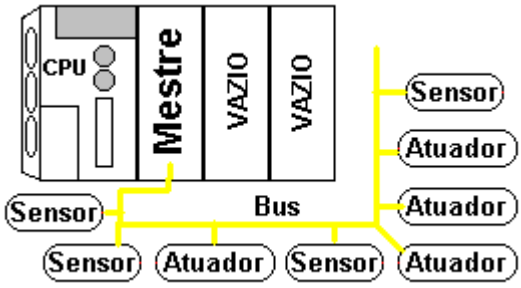


Fig. 2.3: Arquitetura ideal para o *Fieldbus*

A tecnologia *fieldbus* possui uma enorme flexibilidade de aplicações sendo empregada nos setores automobilístico, alimentício, madeireiro, petroquímico, construção civil, metal-mecânico, farmacêutico, eletro-eletrônico, entre outros.

2.6 CONTROLE, MONITORAÇÃO E INTEGRAÇÃO DA PRODUÇÃO

2.6.1 CONTROLE DA PRODUÇÃO

Com o advento de todos estes conceitos, ideologias e técnicas, em síntese, com toda esta revolução, aspectos como a monitoração, a configuração e o teste de elementos do setor de produção totalmente automatizado tornaram-se viáveis.

A evolução das técnicas de comunicação entre processos, em especial, o modelo de comunicação OSI [9] [9], permite que um processo, ou até um sistema como um todo, possam ser configurados através de um aplicativo remoto que possua uma interface de comunicação com o sistema como um todo ou com cada processo deste. A possibilidade de realizar esta configuração é um dos fatores primordiais do desenvolvimento desta dissertação e do sistema fruto dela.

2.6.2 MONITORAÇÃO DA PRODUÇÃO

A monitoração de processos tem como objetivo o controle de qualidade preventivo, ou seja, o processo de fabricação é monitorado com o intuito de evitar que problemas causem danos graves no sistema. O processo de monitoração pode ocorrer tanto de forma *online* quanto *offline*.

Os testes são tão necessários quanto às outras funcionalidades. Antes de um novo valor ser enviado para um processo ou então um novo *setup* para um sistema a ser carregado no sistema real é necessário que os dados sejam confiáveis e que não causem qualquer problema no sistema.

2.6.3 INTEGRAÇÃO DA PRODUÇÃO

Através do controle e da monitoração da produção consegue-se chegar a uma produção totalmente integrada. Neste contexto o operador terá todo o controle e saberá o que está ocorrendo na planta a qualquer instante de tempo. Com o sistema totalmente integrado é muito mais fácil atuar de maneira a evitar que um problema se propague e venha a prejudicar a produção.

Com o objetivo de disciplinar e auxiliar nesta integração existem várias filosofias que atuam neste sentido como: FMS (Flexible Manufacturing Specification), CIM (Computer Integrated Manufacturing), etc.

3 O PROJETO SFB368 (AUTONOMOUS PRODUCTION CELL - APC)

3.1 MOTIVAÇÃO

A Célula Autônoma de Produção, do inglês, "*Autonomous Production Cells*", ou simplesmente "**APC**" como será referida de agora em diante, consiste em um projeto de pesquisa que inclui o desenvolvimento de soluções nas áreas de: planejamento, controle da máquina-ferramenta, interface homem-máquina, controle das ferramentas de trabalho e controle do processo [9]. Para o desenvolvimento deste projeto foram selecionados dois dos processos mais comuns na Engenharia Mecânica clássica, que são: a usinagem e a soldagem. Estas duas funções incluem e definem os processos selecionados para cada campo de trabalho dos grupos que fazem parte deste centro de pesquisa em cooperação, conforme mostra a figura 3.1.

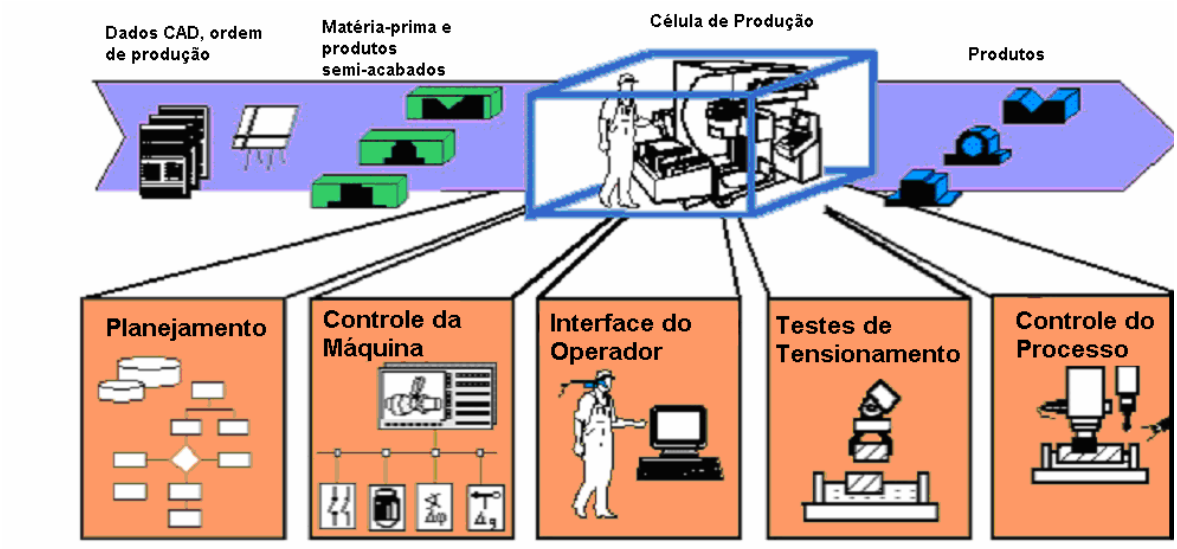


Fig. 3.1: Partes da Célula Autônoma de Produção

Na figura 3.1 se pode observar o processo completo de transformação da matéria-prima em produto. O sistema para transformar matéria bruta em algum produto com valor agregado é composto por uma série de outros componentes tangíveis e intangíveis. Antes do produto final ser fabricado ocorrem algumas fases. O *planejamento* é a primeira e talvez uma das principais etapas. Sem um planejamento adequado corre-se o risco de produzir algo que não tenha utilidade nenhuma. Um sistema de *controle de máquinas* também é importante, uma vez que garante a boa operacionalidade entre as diversas etapas de produção em diferentes equipamentos. Para monitoração, visualização, controle e atuação sobre o processo é preciso existir uma boa interface

com o usuário humano do sistema. Na *interface* é essencial existir o fácil acesso a todas as variáveis do sistema a qualquer momento. Para se realizar um teste antes da produção em grande escala ou até mesmo para garantir a segurança na produção, uma fase de *testes da montagem* se faz necessária. A "última fase" acontece durante a produção do produto, onde é preciso se ter total e irrestrito *controle do processo* de produção. Após tudo isto, tem-se um produto que atende a todas as necessidades do cliente e que utiliza todas as técnicas e equipamentos indispensáveis da célula autônoma de produção.

3.1.1 AUTONOMIA

Um dos objetivos primordiais do projeto APC consiste na base do termo "**Autonomia**" [9], que para um melhor entendimento foi dividido em 3 aspectos principais:

3.1.1.1 AUTONOMIA NA INTEGRAÇÃO DE TAREFAS ADICIONAIS

A autonomia em uma área de produção é conseguida com o aumento das capacidades das máquinas que compõem o ambiente de produção. Conseqüentemente, a inserção de novas funcionalidades, que normalmente eram desempenhadas por máquinas com interfaces com o usuário e agora são integradas e incluídas em um processo autônomo de produção resultam em uma célula autônoma de produção. O objetivo é dar ao projeto APC a habilidade de realizar o processo de manufatura de maneira completa, o que idealmente seria controlar a produção desde o projeto do produto totalmente novo até a sua efetiva produção na máquina-ferramenta. Tudo isto ocorre baseado na idéia de uma caixa-preta, onde as informações provenientes de modelos CAD, a matéria-prima correta e a entrada dos dados corretos para a produção seriam as entradas do sistema e a saída seria o produto acabado, como ilustra a figura 3.2 [6][9].

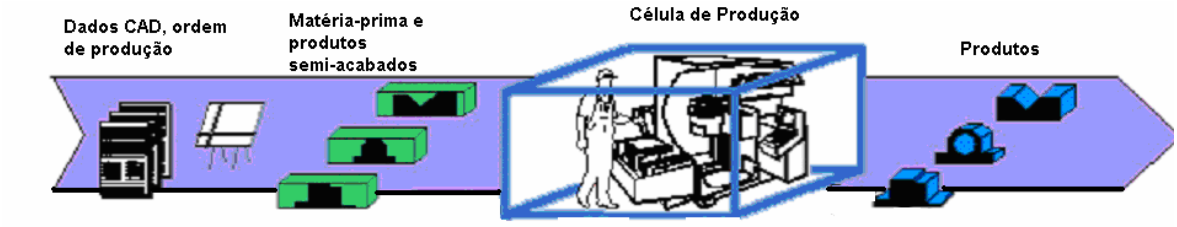


Fig. 3.2: Entradas para o APC

A realização disto tudo de maneira autônoma requer uma integração do controle de máquina com outras funções auxiliares como:

- funções de planejamento (CAPP/CAM), geração de geometria, tecnologia e processamento de dados para o sistema de controle da unidade;
- assistência estendia para o usuário;

- funções de *setup* e teste, que permitem um controle da quantidade de matéria-prima a ser usada e produtos automatizados e sobre ele próprio;
- funções de controle do processo, que permitem detectar e analisar falhas durante o processo (como parâmetros do processo, ferramentas e produtos) e gerar estratégias de reação.

3.1.1.2 AUTONOMIA NO CONTROLE DE FALHAS

Um importante aspecto para a autonomia é a habilidade da execução de processos complexos independentemente combinados com alta estabilidade quanto à falhas, baseados em reações independentes, dependendo do estado do processo e da existência de estados de erros. Em adição, uma elevada precisão no processo de manufatura tem sido conseguida com o aumento das capacidades das máquinas e dos processos.

A alta precisão e a tolerância a falhas podem ser conseguidas somente se as limitações dos componentes da mecânica clássica forem superados pela integração através de componentes eletrônicos e informações de processamento. Uma Célula Autônoma de Produção (**APC**), conseqüentemente, deve conter um alto número de Sensores/Atuadores inteligentes.

3.1.1.3 AUTONOMIA PARA O USUÁRIO

Uma parte muito importante do projeto APC é a integração homem-máquina. O projeto **APC** fornece funcionalidades poderosas para a realização e controle das tarefas de manufatura. Conseqüentemente, a interface homem-máquina do projeto **APC** deve ter um projeto ergonômico para permitir que o usuário realize de maneira simples e fácil desde tarefas complexas até as suas tarefas rotineiras. Neste contexto, a interface homem-máquina deve suportar as tarefas de planejamento e controle dos processos de uma maneira compreensível pelo usuário. Isto significa, que a interface homem-máquina deve permitir o controle permanente do estado dos processos a qualquer tempo. Também deve ser possível realizar uma intervenção no planejamento e na seqüência de manufatura.

3.2 VANTAGENS DO PROJETO APC

Seguindo estas definições quanto à autonomia do projeto **APC** pode-se relacionar as seguintes vantagens: [9]

- essencialmente o **APC** aumenta a habilidade de adicionar novos módulos nos campos de: planejamento, controle de processos, suporte ao usuário e no controle de material;
- habilidade de realizar um processo de manufatura de maneira completa sozinho, idealmente desde a escolha e seleção da matéria-prima e construção do modelo de dados até o produto final;

- da realização do trabalho e de um sistema de configurações de maneira autônoma que sejam completamente transparentes para o usuário.

3.3 UM SISTEMA DE CONTROLE - OSACA

A base para qualquer sistema de controle aberto é uma plataforma para o sistema padrão. OSACA (*Open System Architecture for Controls within Automation systems*) é uma iniciativa européia para definir uma arquitetura de controle aberta para aumentar a competitividade e a flexibilidade dos fornecedores e usuários de sistemas de controle como: construtores de máquinas-ferramenta, vendedores de controladores e usuários finais[10].

Hoje existe uma ampla demanda por uma grande variedade e flexibilidade de máquinas-ferramenta. De maneira a satisfazer estes desejos do mercado os vendedores de controladores e de softwares para estes sistemas buscam implementar estruturas de controle cada vez mais modulares. Para conseguir isto é necessário que também os softwares de interface sejam bem desenvolvidos e permitam a extensão, a mudança e o reuso de componentes do software em uma unidade de controle.

3.3.1 A PLATAFORMA OSACA

A arquitetura básica para todo sistema de controle modular consiste de uma plataforma de sistema contendo o hardware, que é composto pelo componentes eletrônicos (PC, Máquinas-ferramenta, Sensores/Atuadores, redes Fieldbus, etc), o software do sistema como o sistema operacional, que garante a independência quanto a determinado hardware e também disponibiliza as funções básicas (como: gerenciamento de tarefas, acesso a arquivos de sistema e uma execução pseudo-paralela das aplicações), e um conjunto de módulos de aplicações (software), os quais contêm as funcionalidades básicas do sistema de controle. A plataforma OSACA é baseada em uma arquitetura que segue os princípios da Orientação a Objetos e tem como principais características [11][10]:

- O Sistema de Comunicação garante a cooperação entre os diversos módulos de uma maneira padrão, definindo uma interface do sistema independente que troca informações entre os diferentes módulos de software.
- A Arquitetura de Referência define os componentes básicos do controle e também define uma lista de objetos e mecanismos para interagir com os componentes. Conseqüentemente, define quais módulos podem e devem ser encontrados no sistema de controle, quais tarefas eles desempenham e como eles interagem uns com os outros.

- O Sistema de Configuração serve para construir uma topologia de software que apresenta os módulos disponíveis para inicializar e conectar com diferentes módulos durante o *startup* do sistema.

Estes elementos, ver figura 3.3, integrados dentro da plataforma do sistema, são acessíveis através de uma Interface de Programação da Aplicação ou em inglês *Application Programming Interface* (**API**). A API também permite a escolha de uma maneira otimizada das soluções de maneira a não violar os critérios de sistemas abertos. Por causa dessa natureza orientada a objetos, que inclui encapsulamento e herança múltipla, os módulos de aplicação são também chamados de Objetos Arquiteturais ou do inglês *Architecture Objects* (**AO**).

Como um sistema aberto é composto por uma coleção de AOs, a configuração do sistema é necessária, que inicializa os AOs e desempenha a sequência de inicialização. O sistema coopera de maneira tal a garantir que os AOs necessários sejam inicializados corretamente. Isto é conseguido com o uso de arquivos de sistema que contêm informações que são interpretadas durante a configuração do sistema.

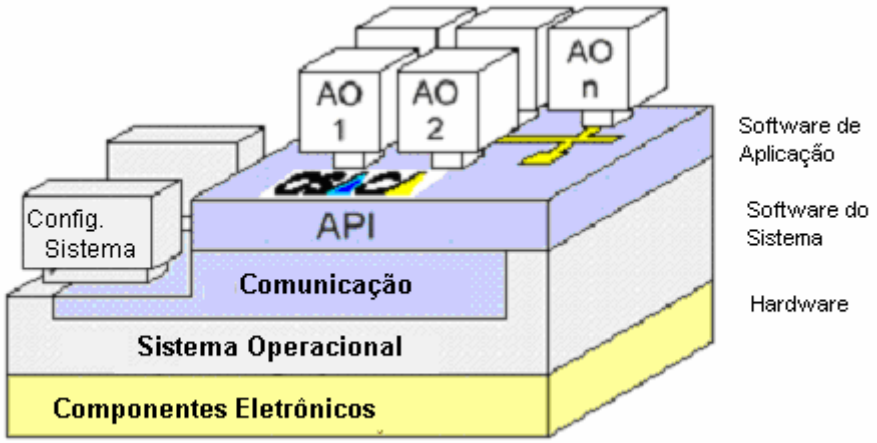


Fig. 3.3: Estrutura da plataforma OSACA

O módulo de sensores/atuadores (SAM), que é o tema central do presente trabalho, e que será descrito na próxima seção, foi implementado como sendo um AO, portanto ele possui uma interface padrão para o acesso aos Sensores/Atuadores através da plataforma **OSACA**. A figura 3.4 exemplifica o modelo da arquitetura adotada. O módulo SAM aparece como um componente AO que possui uma interface de comunicação com o **OSACA** e que por sua vez acessa a outros componentes do sistema.

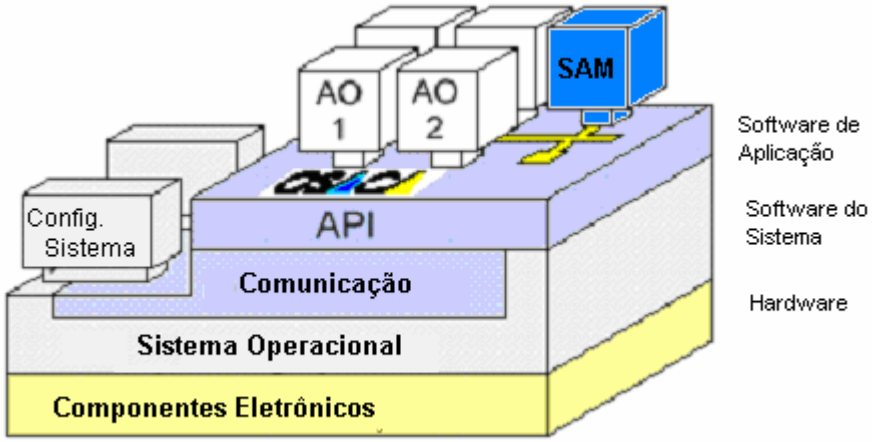


Fig. 3.4: Plataforma OSACA-SAM

Como o objetivo máximo do projeto APC é a "**autonomia**" foi necessário que cada objeto de arquitetura (AO) do módulo **SAM** pudesse ser executado nos sistemas operacionais mais conhecidos e também no **OSACA**. Assim cada AO do **SAM** possui duas partes: uma interface para o **OSACA** propriamente dita e uma interface para o Sistema Operacional (SO) no qual o **SAM** está trabalhando (figura 3.5).

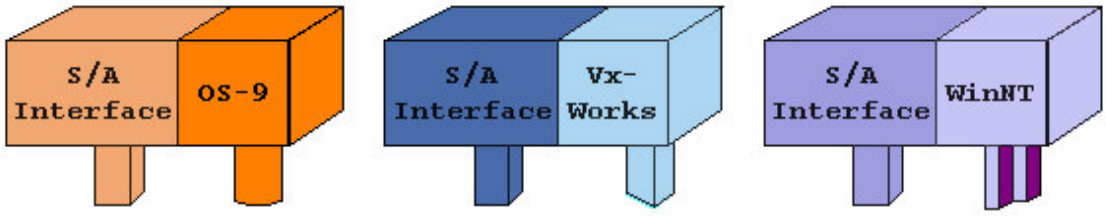


Fig. 3.5: Interface SAM para os diferentes sistemas operacionais

3.4 PROJETO SAM (*SENSOR/ACTUATOR MODULE*)

Após a discussão sobre redes industriais, sensores & atuadores pode-se apresentar o projeto **SAM,** que trata do desenvolvimento de um módulo que possibilite a inserção, retirada e configuração de sensores e atuadores na Célula Autônoma de Manufatura.

Os tópicos seguintes irão contextualizar e justificar o projeto do módulo de Sensores/Atuadores (SAM). O primeiro tópico define o módulo de Sensores/Atuadores dentro do projeto da Célula Autônoma de Manufatura enquanto que o segundo tópico busca apresentar a estrutura do módulo SAM.

3.4.1 Definição do Módulo de Sensores/Atuadores para a Célula Autônoma de Produção

Os requisitos do sistema implicam numa arquitetura para o controle da Célula Autônoma de Produção como pode ser observado na figura 3.6. Os pontos listados a seguir definem a arquitetura:

- Sistema de hardware baseado em VME;
- Sistemas de Fieldbus diferentes para conectar os Sensores/Atuadores;
- Serviços de acesso e de integração de Sensores/Atuadores através de TCP/IP [9];
- Sistema de comunicação baseado no modelo OSACA.

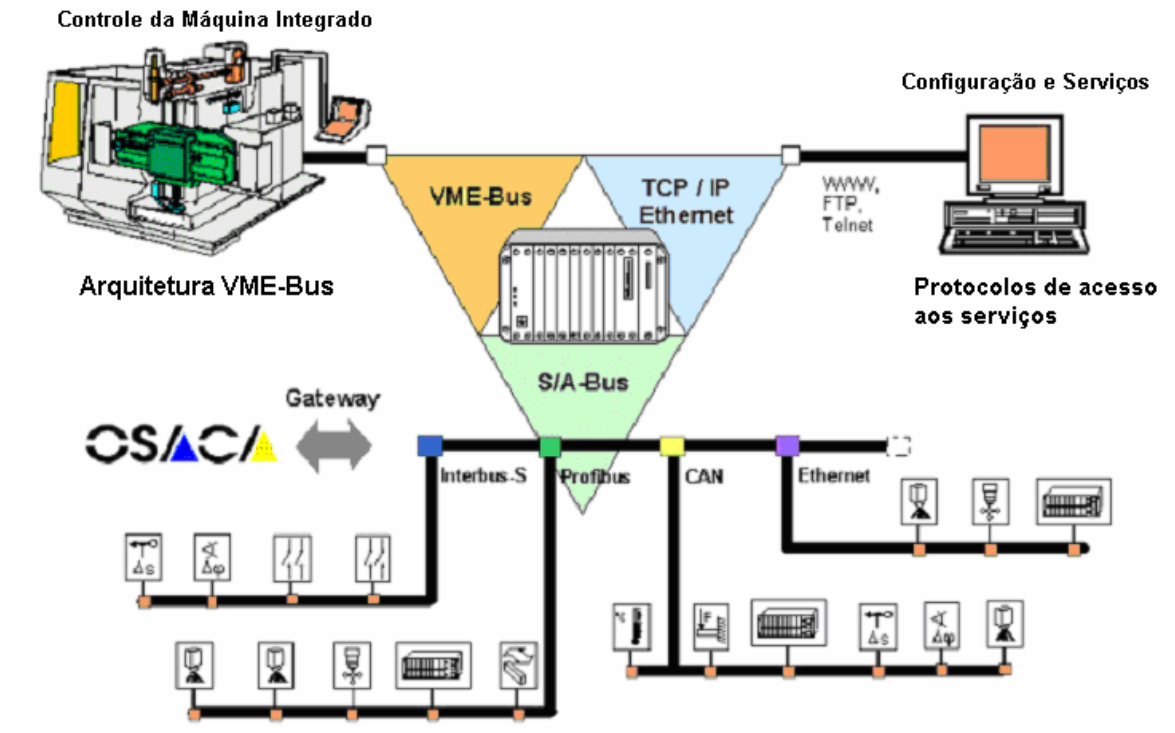


Fig. 3.6: Arquitetura da rede da Célula Autônoma de Produção

O *VME-bus* é uma das arquiteturas líderes para aplicações industriais, especialmente para sistemas embutidos, e isto fornece robustez, modularidade, flexibilidade e expansabilidade[6].

Os sistemas *Fieldbus* fornecem uma ampla expansibilidade e tornam a conexão de novos Sensores/Atuadores muito mais fácil que em sistemas convencionais. Adicionalmente, o sistema *Fieldbus* reduz consideravelmente a quantidade de fios no chão-de-fábrica e também é um sistema mais robusto, especialmente contra as interferências eletromagnéticas do meio. O uso de diferentes sistemas de barramento permite a escolha de Sensores/Atuadores através da funcionalidade e preço sem se preocupar com os requisitos para a comunicação, o que dá uma grande flexibilidade para o

sistema. Por esta razão os sistemas mais usados de *Fieldbus* foram escolhidos para o projeto da Célula Autônoma de Produção: ***Profibus***, **CAN** e ***Interbus*** [12][13].

Um dos mais importantes requisitos do projeto **APC** é uma interface padrão para a comunicação. Uma placa Ethernet em conjunto com o protocolo TCP/IP fornece uma interface padrão e uma maneira fácil para comunicação entre módulos. Ela também tem muitas aplicações como FTP, HTTP ou Telnet. Por causa destas vantagens, o protocolo TCP/IP é usado como serviço de interface do SAM.

Nos dias atuais o protocolo TCP/IP é cada vez mais usado para a integração de Sensores/Atuadores, porque possui grande facilidade quanto ao acesso e instalação. Ele trabalha muito bem em sistemas que não possuem grandes restrições de tempo-real (*soft real time systems*) e também quando as interferências eletromagnéticas não têm grande influência sobre a qualidade dos dados.

3.4.2 A ESTRUTURA DO MÓDULO DE SENSORES/ATUADORES

Como conseqüência, a estrutura do software mostrada na figura 3.7 é proposta e adotada. O componente principal é o **SAM**-*Server*, que implementa o protocolo de barramento que permite o acesso às informações dos S/A com os requisitos de tempo-real. O **SAM**-*Server* é dividido em três partes principais: a primeira (*Bus*) para a implementação do protocolo do barramento, outra (*Data*) para a manipulação da informação e a terceira parte (*Man*) que cuida da interface e do gerenciamento do **SAM**-*Server* com as *threads* das aplicações [6][7].

Outros componentes importantes dentro do **SAM** são os **SAM**-*Clients*. Um desses é o *OSACA-Application* que é uma interface para outro objeto de arquitetura (*Architeture Object*) **OSACA**. É ainda interessante fornecer interfaces para outras aplicações. Para um suporte apropriado para o usuário, foram desenvolvidos dois outros aplicativos para o **SAM**: o primeiro é uma ferramenta para configuração, monitoração e debug da rede de sensores/atuadores na Célula Autônoma de Produção (**SAMWeb**) que é o sistema resultante desta dissertação de mestrado . A outra é o *Command Line* que é uma aplicação em MS-DOS que permite o usuário testar e supervisionar o sistema. Baseado nestas idéias, um **SAM**-*Client* foi desenvolvido com duas partes principais, uma é a interface genérica para acessar os serviços de muitos **SAM**-*Servers*, e outra implementa os procedimentos específicos da aplicação. Por exemplo, no caso de aplicações **OSACA**, ela implementa a interface **OSACA** para os serviços e informações do **SAM**.

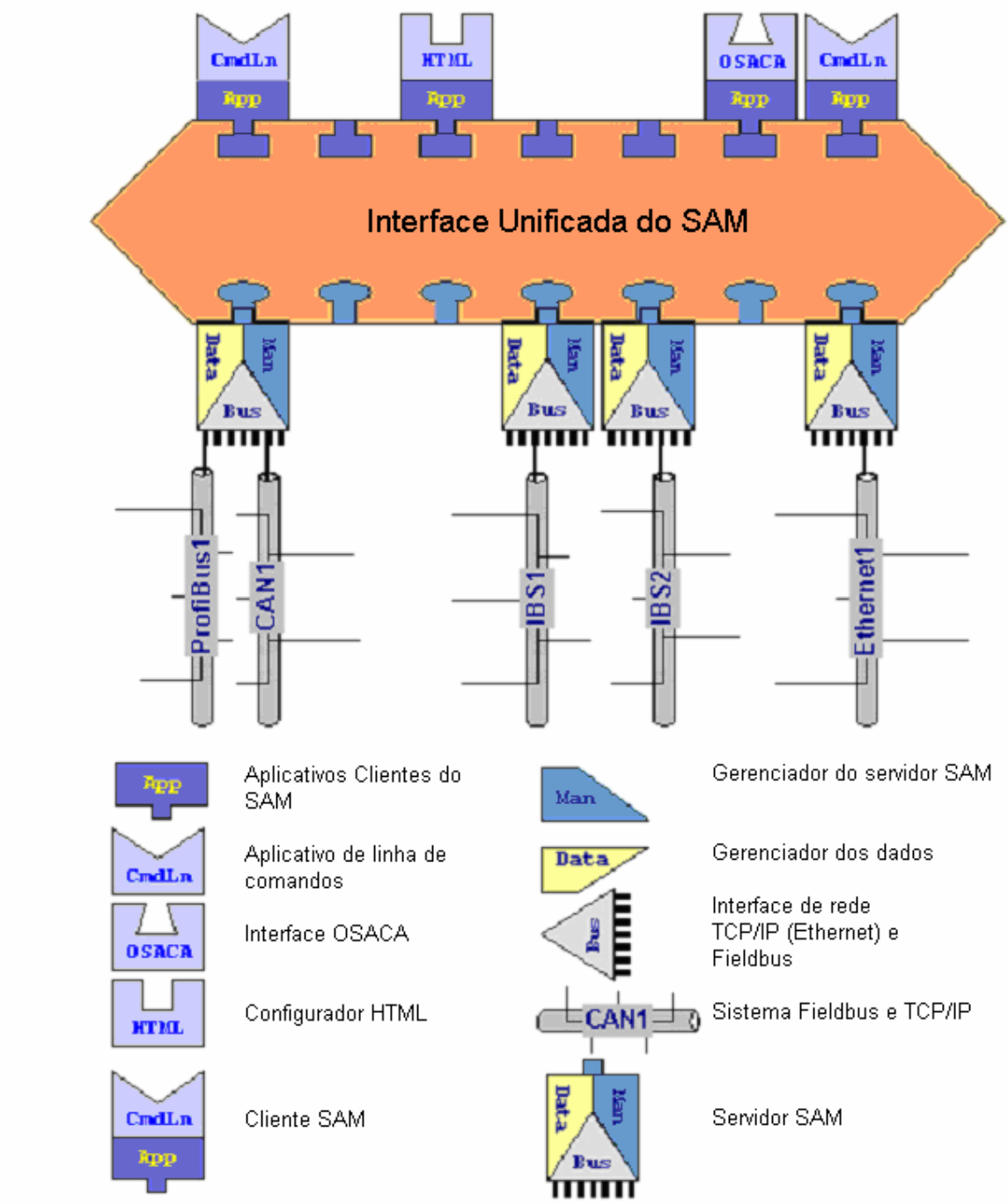


Fig. 3.7: Estrutura do Módulo de Sensores/Atuadores (SAM)

As interfaces dos **SAM**-*Clients* e **SAM**-*Servers* são implementadas de uma maneira padronizada baseada em um grupo de classes de comunicação entre processos. Estas classes tornam viável a comunicação entre muitos **SAM**-*Clients* e **SAM**-*Servers*. Neste ponto é importante salientar que um **SAM**-*Server* pode ser ligado a diversos tipos de sistemas de barramento, se

necessário, mas por razões de performance é interessante que cada **SAM**-*Server* trabalhe somente com um tipo de barramento.

4 FERRAMENTAS DE APOIO PARA IMPLEMENTAÇÃO DA SOLUÇÃO

No capítulo anterior a ênfase maior foi dada na apresentação dos conceitos de engenharia que justificam esta dissertação. Tratou-se desde a apresentação dos conceitos mais globais de manufatura até a apresentação final da estrutura do Módulo de Sensores/Atuadores (SAM) da Célula Autônoma de Produção (APC), o qual é o módulo onde esta dissertação de mestrado encontra-se inserida. Agora, este capítulo dará maior destaque às técnicas de computação que foram utilizadas para desenvolver o sistema para configuração, monitoração e teste dos Sensores/Atuadores da Célula Autônoma de Produção.

O capítulo é estruturado da seguinte forma: em primeiro lugar se apresenta muito brevemente um conceito de orientação a objetos e apresenta-se um estado da utilização de tais linguagens no setor industrial. Após esta apresentação cita-se algumas características das linguagens Orientadas a Objetos utilizadas no desenvolvimento do sistema. O maior destaque deste capítulo fica por conta do tópico onde é abordada a técnica JNI (*Java Native Interface*) que tem como principal objetivo permitir a integração de sistemas desenvolvidos em diferentes linguagens com a linguagem Java. Sem esta técnica, o desenvolvimento desta dissertação de mestrado ficaria comprometido uma vez que, como será explicado, a técnica JNI é a única que além de possibilitar esta integração dos sistemas atende ao requisito principal do projeto APC, que é a **autonomia**.

4.1 LINGUAGENS OOP

As linguagens estruturadas, como C, foram e continuam sendo linguagens amplamente utilizadas na área da informática. Apesar do seu ótimo desempenho em termos de programas otimizados, velozes e portáteis, a linguagem C sofreu duras críticas com relação a qualidade do código gerado considerando-se outros aspectos de relevância da engenharia de software como: legibilidade do código, sua reusabilidade e facilidade de manutenção.

Um programa de computador sempre busca representar uma dada realidade (mundo real ou modelo abstrato: matemático) através de uma linguagem de programação. Quando criamos qualquer programa simples de computador, estamos na verdade criando um modelo da realidade e usando este modelo para analisar e estudar esta realidade. O nível de detalhe com que criamos um programa depende diretamente do nível de detalhe necessário para modelar o ambiente de acordo com as necessidades impostas pelo usuário.

Modelar um determinado ambiente complexo utilizando-se somente as estruturas e funcionalidades disponíveis pelas linguagens estruturadas, como C, é uma tarefa árdua. Por esta razão, alguns programadores deste tipo de linguagem não conseguem muitas vezes entender determinados códigos desenvolvidos, pois não conseguem entender o modelo da realidade gerado por este programa ou a sua estrutura.

Neste sentido, criou-se uma forma totalmente nova e revolucionária para se modelar e simular um ambiente dado (mundo real ou sistema abstrato): a Orientação a Objetos. A Orientação a Objetos busca modelar um ambiente utilizando-se dos próprios elementos presentes neste ambiente, ou seja, os objetos. Todo ambiente pode ser modelado e simulado a partir de uma descrição dos objetos que o compõe e das relações entre eles[38].

Por exemplo, suponha que você esteja criando um programa para permitir que um arquiteto crie um modelo gráfico de uma sala de estar e apresente este modelo ao seu cliente. Empregando o paradigma de Orientação a Objetos, primeiramente o arquiteto terá que identificar os objetos que compõem a sala de estar que, considerando o nível de abstração do seu modelo, devem ser descritos. Neste caso, podemos facilmente listar alguns objetos que podem ser definidos: janela, porta, parede, sofá, mesa, quadro, tapete, vaso de flores, etc.

A quantidade de objetos e o nível de detalhe da sua descrição dependerá necessariamente do nível de abstração do modelo, ou seja, daquilo que o arquiteto considera importante que esteja no modelo. Por exemplo, representar o objeto "mesa" pode significar para um marceneiro definir a sua geometria, o tipo de material, o tipo de verniz requerido e o tipo do seu acabamento, enquanto que para um arquiteto significa definir a textura da mesa, o tom da sua cor, o seu custo, a durabilidade desta, etc. Por isso, é importante ter em mente as necessidades do seu cliente antes de criar um modelo super detalhado e ineficiente.

Após a definição dos objetos que compõem um dado ambiente, precisamos definir as relações entre estes. Por exemplo, precisamos definir que um objeto "porta" e um objeto "janela" estão posicionados sob a mesma parede, ou seja, são propriedades de um terceiro objeto chamado "parede". Assim, como outro objeto "parede" pode não conter uma "janela" mas um objeto do tipo "quadro". Podemos até criar um objeto chamado "sala" que contém todos os objetos presentes na sala descrita pelo arquiteto e permitir que este arquiteto venha criar futuramente outros objetos como "cozinha" ou "quarto" e possa, assim, modelar toda uma residência.

A modelagem orientada a objetos fornece uma estrutura bem clara para o código fonte do programa. Isto permite que outro programador possa entender o programa criado, reutilizar partes em outros programas e, ainda, facilmente localizar erros no código fonte. Por exemplo, suponha que na hora em que o programa estiver desenhando o objeto "mesa" na tela para o cliente do

arquiteto apareça algum defeito na mesa. Poderemos, então, facilmente concluir que: ou o objeto mesa esta mal representado (erro dentro da definição do objeto mesa), ou a relação deste objeto com os demais objetos esta mal definida (por exemplo, a posição deste objeto na sala), ou o mecanismo responsável pelo desenho deste objeto na tela não está operando corretamente. Isto nos permite que, rapidamente, possamos localizar, isolar e, em seguida, corrigir este erro.

A programação orientada a objeto se baseia no encapsulamento de uma estrutura de dados com as próprias rotinas de tratamento dos mesmos e na capacidade de herança destes dados e rotinas por outros objetos derivados.

A programação orientada a objeto carrega consigo uma série de vantagems e também algumas desvantagems. Como vantagens pode-se citar [39]:

- existem muitas ferramentas de apoio ao desenvolvimento de programas;
- os programas têm uma estrutura altamente modular, o que permite um mais fácil controle e expansão dos mesmos. Por exemplo, basta alterar as características de um objeto "mãe" para que todos os objetos "filhos" (que herdaram propriedades) sejam também automaticamente alterados de forma correspondente;
- a programação orientada a objeto se baseia fortemente na própria forma de pensar humana, ao contrário da forma algorítmica e procedural da programação convencional.

Por outro lado, há também desvantagens relativas a esta forma de programação:

- grande necessidade de memória;
- grande complexidade de gerenciamento interno das estruturas dos objetos, o que implica em velocidade de execução menor.
- difícil otimização de tempo de execução dos programas. A otimização de programas freqüentemente requer uma violação das próprias regras de programação orientada a objeto.

Estas desvantagens têm se tornado menos críticas nos últimos anos devido ao grande aumento da velocidade de processamento dos computadores bem como ao aumento de sua capacidade de memória.

Com este breve esclarecimento já se pode observar algumas das propriedades vantajosas fornecidas pela programação Orientada a Objetos.

4.2 A LINGUAGEM C ORIENTADA A OBJETOS (C++)

A linguagem C++ foi criada a partir da linguagem C, acrescentando novas estruturas e mecanismos que possibilitam a geração de programas segundo o paradigma de Orientação a

Objetos. Algumas linguagens como Smalltalk refletem muito mais a "cultura" ou a "metodologia" Orientada a Objetos, favorecendo a geração de programas segundo esta metodologia. C++ é uma espécie de adaptação de C a metodologia de Orientação a Objetos e, por isso, não possui todas as facilidades e mecanismos de uma linguagem puramente Orientada a Objetos como Smalltalk. Entretanto, C++ herda de C a capacidade de gerar programas pequenos, otimizados, de "baixo-nível" [24] e portáveis. Estes motivos propiciaram a grande difusão que a linguagem C++ vem sofrendo nos últimos anos.

A geração de códigos em C++ requer normalmente um tempo maior de desenvolvimento que um programa em C normal. Entretanto os ganhos em reusabilidade e em diminuição dos tempos de manutenção do programa fazem com que C++ seja mais atrativo para a geração de médios e grandes sistemas. Emprega-se C basicamente quando temos que fazer um programa em pouco tempo, com grandes restrições na dimensão do código e no tempo disponível para o processamento do programa (requisistos de tempo-real).

A reusabilidade advém do fato que objetos definidos para representar um dado ambiente, podem ser empregados para representar um outro ambiente. Por exemplo, para representar o ambiente “sala de estar”, havíamos utilizado o objeto "mesa". Podemos, entretanto, utilizar o mesmo objeto "mesa" no ambiente cozinha, sem precisar redefini-lo novamente. Neste aspecto, ganhamos em tempo de desenvolvimento.

No tópico seguinte do trabalho, busca-se apresentar a utilização de C/C++ dentro do projeto da Célula Autônoma de Manufatura.

4.3 C/C++ NO PROJETO DA CÉLULA AUTÔNOMA DE PRODUÇÃO

No chão-de-fábrica a linguagem C/C++ é a linguagem mais utilizada no desenvolvimento e instalação de sistemas. Hoje em dia, grande parte dos sistemas que existem na indústria foram programados em C/C++. Isto deve-se ao fato de C/C++ ser um linguagem bastante robusta e otimizada para aplicações de tempo crítico e que exigem grande segurança e confiabilidade dos dados[18].

A grande maioria dos sistemas que compõem uma célula de produção são desenvolvidos utilizando-se Assembly ou C/C++, por serem as duas linguagens mais conhecidas e dominantes no mercado de fornecedores de equipamentos para o setor produtivo.

Tendo como base à afirmação anterior, o projeto da Célula Autônoma de Produção utilizou-se da linguagem C/C++. No contexto do módulo de Sensores/Atuadores do projeto APC toda a estrutura de servidores e clientes foi desenvolvida utilizando-se C++.

4.4 JAVA

Este tópico é dedicado a uma pequena introdução à linguagem de programação Java e sua interação com C/C++. Uma atenção especial é devotada ao JNI (*Java Native Interface*) que é uma parte da linguagem Java dedicada a promover a troca de dados entre a linguagem Java e outras linguagens, como por exemplo a linguagem C/C++.

Nesta dissertação a relação acontece da seguinte maneira: a interface do sistema aplicativo foi desenvolvida em Java, o modelo de uma comunicação foi escrito com os métodos de JNI, e todo o modelo do módulo de Sensores/Atuadores do projeto APC, como todos os outros módulos, foram desenvolvidos em linguagem C/C++.

Quando se escreve um programa na maioria das linguagens de programação, é preciso decidir sobre que processador e que sistema operacional a aplicação finalizada irá funcionar. Com isto, a chamada de funções específicas é incluída em uma biblioteca associada com o sistema operacional da plataforma alvo. Por exemplo, se a aplicação for escrita para o ambiente Windows, pode-se fazer referência às classes da Microsoft através do pacote MFC (*Microsoft Fundation Classes*). Porém, se máquina alvo for um Macintosh, a aplicação chamará funções no *toolbox* do MAC OS (*Operating System*)[40].

Quando a aplicação está finalizada, o código fonte será emitido através de um compilador que transforma o código em um jogo de instruções nativas para que o processador da máquina possa utilizar. Como exemplo, os sistemas Windows funcionam geralmente com um processador da Intel tal como o Pentium, por outro lado os Macintosh utilizam os processadores da Motorola 680x0 ou PowerPC.

Quando a aplicação é escrita em Java, não se decide sobre o tipo de sistema operacional em que o sistema irá ser executado, como Windows, LINUX, UNIX, MAC, ou qualquer outro sistema operacional.

Java possui suas próprias bibliotecas, chamadas *packages* - pacotes, o que torna JAVA independente da plataforma. Similarmente, a aplicação não é desenvolvida sabendo se o produto terminado funcionará em um Pentium da Intel, em um IBM PowerPC, ou em um processador SUN SPARC. O compilador de Java não gera instruções nativas. Pelo contrário, o compilador escreve *bytecodes* para uma máquina que não existe realmente, a máquina Java ou JVM (*Java Virtual Machine*).

A Java Virtual Machine (JVM) não existe realmente no sentido físico, conseqüentemente o código de Java irá funcionar em qualquer máquina que tenha a JVM instalada. A SUN implementou uma versão de software da JVM para a maioria das plataformas comuns. Quando o arquivo de *bytecodes* (chamado arquivo de classe) é carregado na máquina alvo, a JVM interpreta

estes arquivos e faz com que o sistema funcione perfeitamente. A JVM lê o arquivo da classe e faz o trabalho especificado pelo código original de Java [40].

Sendo a máquina virtual Java (JVM) fácil de mover de uma máquina a outra, pode-se esperar que todos os processadores novos ou sistemas operacionais possuirão em um curto tempo uma *Java Virtual Machine* dentro (JVM *inside*). Uma vez que a aplicação é escrita no código de Java que funciona em uma máquina, pode funcionar em todas as plataformas que possuam uma JVM.

4.4.1 DIFERENÇAS ENTRE JAVA E OUTRAS LINGUAGENS

Java é uma linguagem de programação simultânea, orientada a objetos e com potencialidades *Client/Server*.

Nesta seção, as qualidades distintivas principais de Java são apresentadas [23].

Linguagem de Programação: Na última década do século XX, o mundo dos softwares era similar em muitas maneiras à maneira que as coisas se realizavam nos anos 70. Naqueles dias, os PCs tinham recém saído e o software disponível não acompanhava a demanda. Entretanto, a maioria dos modelos de PCs vinham com um interpretador BASIC.

Muitas pessoas que não se consideravam programadores aprenderam o BASIC e começaram a escrever programas. Freqüentemente compartilhavam seus programas pelo disco flexível, ou, mais atrasado, por BBS (*Bulletim Board System*), e a indústria de softwares *shareware* começou a se desenvolver.

Como o SmallTalk, o C, e o C++ (e ao contrário do BASIC), Java é projetado para o uso por programadores profissionais. Hoje a linguagem HTML e as linguagens de *Scripting* tais como o *Javascript* da Netscape ocuparam o nicho ocupado anteriormente por BASIC. Muitas pessoas que não se consideram programadores profissionais não podem escrever Java *applets*, todavia podem usar os *applets* escritos por outros para adicionar vida à suas páginas.

Java é OOP (Object-Oriented Programming): Em geral, os engenheiros de software trabalham em cinco atividades durante o desenvolvimento do software. A orientação a objetos é usada em algumas fases deste desenvolvimento, na parte de projeto, o conceito de objeto é muito importante para facilitar a posterior modelagem e implementação das classes e objetos na linguagem OOP que será utilizada. As fase são [20]:

- Análise do sistema: processo de identificar as exigências do usuário.
- Projeto do sistema: processo de desenvolver uma solução que satisfaça às necessidades e exigências do usuário.
- Implementação: codificação do projeto em uma linguagem de programação de computador tal como Java.

- Teste: assegura-se que o software resultado e terminado satisfaça as exigências do projeto.
- Manutenção: fixar erros da fase de implementação ou projeto, adicionar novas funcionalidades e manter o software sempre em dia com o ambiente (tal como sistemas de gerenciamento de base de dados e sistemas operaionais).

Linguagens Orientadas a Objetos foram introduzidos em torno de 1967. Durante o ano de 1983, Bjarne Stroustrup do laboratório da AT&T Bell introduziu uma nova versão da linguagem de programação mais popular no momento a linguagem C, que suportasse classes. Este "C com classes", como foi primeiramente chamado, foi transformar-se em C++, possivelmente a linguagem orientada a objetos mais popular.

A equipe da SUN Microsystems que projetou Java, era composta por programadores de C++. Eles compreenderam as características de C++ que a tornaram a linguagem OOP mais difundida no mundo. Eles compreenderam também suas limitações. Com isto tudo, eles projetaram Java, copiaram a sintaxe de C++ e reutilizaram as melhores partes de projeto de C++, incluindo o fato de ser fácil de codificar projetos orientados a objetos.

Java suporta modelo Cliente/Servidor: Java estendeu as bibliotecas padrão para incluírem comunicações via rede. Em Java, por exemplo, você pode abrir uma conexão a um *WebSite* ou à outra aplicação na Internet e ler ou escrever dados, da mesma maneira que um programador de C ou de C++ lê ou escreve em um terminal local. Esta decisão de projeto tornou fácil de desenvolver aplicativos clientes na Internet em Java. Por exemplo, o **HotJava**, [21] o *web browser* cliente Java escrito pela SUN Microsystems, é escrito inteiramente em Java.

Java suporta concorrência: No mundo real, objetos diferentes fazem seu trabalho simultaneamente. Em um computador, ao menos, em um computador com um único processador, somente um conjunto de instruções pode ser executado em um dado momento. Para ajudar os programadores na construção de aplicações que refletem mais exatamente uma determinada tarefa do mundo real, surgiu o conceito de *multitasking* ou multitarefa, em português.

Em um sistema multitarefa, duas ou mais aplicações podem compartilhar um único processador, sendo que cada um tem a ilusão de que tem o completo controle do processador. Cada aplicação (normalmente chamada processo) tem sua própria parte protegida da memória onde armazena seus dados. Na maioria dos sistemas operacionais um processo não pode interferir em outro processo. Desta maneira, estes sistemas operacionais incluem chamadas especiais de funções (*Inter-Process Communication*, IPC) que permitem que um processo emita ou receba dados de outro. No nível de sistema operacional, há uma quantidade significativa de tarefas que são requeridas para iniciar um processo ou para comutá-lo com outros processos. Este tipo de funções

recebe o nome de "*thread*". De um modo geral, as *threads* não oferecem uma proteção total aos processos, contudo elas podem ser iniciadas e usadas mais rapidamente. O problema mais grave com os processos e as *Threads* é que estas facilidades são oferecidas a nível de sistema operacional. Caso se escreva um programa que funcione em Windows NT, é necessário modificar as funções que instanciam os processos e as *threads* caso o sistema seja transferido para o UNIX, por exemplo [19].

Em alguns sistemas operacionais, como em versões mais antigas do Microsoft Windows, não existe a facilidade de programação *multitasking* como nos sistemas mais novos. A solução apresentada pela SUN para este problema foi fazer com que o suporte a concorrência fosse parte da linguagem de programação. Com isto, se uma aplicação é feita utilizando-se *threads* é garantido que a mesma aplicação funcione perfeitamente em Windows, UNIX ou MAC. Além disso, o modelo orientado a objetos de Java disciplina a maneira como é feita a comunicação entre aplicações. As *threads,* muitas vezes apresentam as mesmas proteções dos processos (*tasks*), mas com uma menor despesa quanto à programação.

4.4.2 JAVA NATIVE INTERFACE (JNI)

O JNI é um tipo de programação nativa (diretamente no kernel do interpretador), permitindo que o código Java que funciona dentro de uma Máquina Virtual Java (JVM) se relacione com aplicações e bibliotecas escritas em outras linguagens de programação, tais como C, C++, e Assembly. O benefício mais importante do JNI é que ele não impõe nenhuma limitação na execução das Máquinas Virtuais Java. Conseqüentemente, vendedores de JVM podem adicionar a sustentação para o JNI sem afetar outras partes da máquina virtual. Pode-se escrever uma versão de uma aplicação ou de uma biblioteca nativa e esperar que ela trabalhe com todas as máquinas virtuais Java que suportam o JNI. A técnica JNI é produto da especificação de JNI 1.1, da SUN Microsystems, Inc. [25][31][32]

4.4.2.1 VISÃO GERAL

Existem situações em que Java sozinho não possui todas as funcionalidades para satisfazer uma aplicação. Nestes casos, a utilização de JNI se faz necessária, e com JNI é possível fazer com que aquela aplicação que não podia ser inteiramente programada em Java possa ser desenvolvida [25].

JNI basicamente é empregado em um dos três casos citados a seguir:

- Quando a biblioteca padrão da classe Java não suportar as características dependentes de plataforma necessitadas pela aplicação.
- Quando já existe uma biblioteca escrita em uma outra linguagem, e em um desejo para fazê-lo acessível ao código de Java com o JNI.

- Quando se deseja executar uma parcela pequena do código tempo-crítico em uma linguagem de baixo-nível tal como o Assembly.

Com JNI também é possível através de métodos nativos:

- Criar, inspecionar e atualizar objetos Java;
- Chamar métodos Java;
- Realizar o intertravamento e tratamento de exceções;
- Carregar classes e obter informações sobre as classes;
- Executar e verificar tipos em tempo de execução (*runtime*).

Também é possível trabalhar com JNI através da invocação de uma **API** que pode acessar e se encaixar em uma Máquina Virtual Java. Isto permite que os programadores desenvolvam facilmente suas aplicações em Java sem ter nenhuma preocupação quanto a ligação entre a aplicação e a Máquina Virtual Java.

4.4.2.2 OBJETIVOS

Acredita-se que um sistema uniforme, padronizado e bem planejado oferece os seguintes benefícios para todos [27]:

- cada versão da máquina virtual Java pode suportar um corpo maior de código nativo;
- os construtores da ferramenta não precisam possuir diferentes tipos de relações nativas dos métodos;
- os programadores de uma aplicação poderão escrever a sua versão das funções nativas que estas funcionarão em Máquinas Virtuais (VMs) diferentes;

A melhor maneira de conseguir uma interface dos métodos nativos padronizada é envolver todas as partes que compõem as Máquinas Virtuais Java (Java VMs). Uma interface de métodos nativos padronizada deve satisfazer os seguintes requisitos[27]:

- Compatibilidade Binária: o primeiro objetivo é a total compatibilidade binária das bibliotecas de métodos nativos através de todas as implementações das JVMs para qualquer plataforma.
- Eficiência: para suportar o código de tempo crítico, a interface do método nativo deve impor um pequeno *overhead*. Todas as técnicas conhecidas para assegurar a independência da JVM (e também a compatibilidade binária) acarretam um certo aumento do *overhead*. Deve-se ter em mente este compromisso entre a eficiência e a independência da máquina virtual.
- Funcionalidade: a interface deve permitir acesso aos métodos nativos que permitam desempenhar todas as tarefas desejadas pelo usuário.

4.4.2.3 ABORDAGEM

Uma das grandes vantagens da existência da adoção de uma interface padronizada consiste em minimizar o trabalho dos programadores, que não necessitam aprender diversas interfaces de diferentes máquinas virtuais. Infelizmente, não existem soluções que satisfaçam completamente a este objetivo [40].

4.4.2.4 PROGRAMANDO EM JNI

Os programadores de métodos nativos devem começar a programar primeiramente a declaração das funções JNI. Programando o JNI é necessário se isolar e não ter em mente qual é a Máquina Virtual Java em que o sistema irá ser executado. Prendendo-se aos métodos padrões JNI é garantido que a aplicação irá ser executada da melhor maneira e mais correta possível em qualquer máquina virtual, de qualquer tipo. Por exemplo, embora seja utilizada a biblioteca JDK 1.1 ou superiores em uma aplicação, a biblioteca JDK 1.0 continua a suportar a relação de métodos nativos da velha versão, uma vez que JNI foi desenvolvida com a preocupação de se adaptar à versão do JDK da Máquina Virtual Java onde está rodando.

Os desenvolvedores das Máquinas Virtuais Java entraram em um acordo em que garantem que as suas máquinas virtuais executem JNI conjuntamente com as outras funções da Máquina Virtual. O objetivo é garantir e assegurar que JNI não traga nenhum ônus e nem uma limitação em sua execução dentro de uma Máquina Virtual. Inclui-se aqui a representação dos objetos, o *Garbage Collection*[4] e todas as outras características da linguagem [40].

4.5 JAVA NO PROJETO NO PROJETO APC

Dentro do contexto do projeto APC outros grupos também trabalham com a tecnologia Java através do desenvolvimento de uma interface ergonômica para a visualização dos dados contidos nas bases de dados do projeto. Já no módulo de Sensores/Atuadores (SAM) do projeto APC não existia nenhum estudo que justificasse e comprovasse a eficiência da utilização da linguagem para a integração do sistema. Nem havia o conhecimento necessário sobre programação e integração de sistemas para justificar um estudo aprofundado das possibilidades e facilidades que poderiam ser encontradas com a utilização de Java com C++ no projeto SAM.

A utilização de Java juntamente com JNI[17] para o desenvolvimento da solução para o problema apresentado baseou-se em estudos desenvolvidos durante a elaboração da solução para o problema de se possuir uma ferramenta para a configuração, monitoração e testes dos Sensores/Atuadores da Célula Autônoma de Produção.

O item seguinte procura justificar a escolha do pacote Java-JNI como solução, através da descrição de vários métodos possíveis para uma solução para o problema apresentado.

4.5.1 JUSTIFICATIVAS DO EMPREGO DE JNI

Nos dias atuais, a maioria dos projetos de novos sistemas se preocupam também com uma possível interface via *Web Browser*. Dentre os sistemas já existentes, se encontram diversas alternativas de linguagens e pacotes de métodos para torná-los *OnLine*. Durante o desenvolvimento da dissertação de mestrado se realizou uma pesquisa com o objetivo de se encontrar a melhor relação entre custo & benefício para o projeto. Após uma pesquisa na Internet, em revistas, em periódicos, em livros e na troca de informações com outros estudiosos da área concluiu-se que existe uma série de possibilidades viáveis para desenvolver os sistemas com a integração de linguagens diferentes e disponibilizar uma interface do tipo *Web Browser* valendo-se do fato de se ter já sistemas prontos, testados e atualizados. Os itens abaixo relacionam algumas dessas possibilidades que foram pesquisadas e estudadas para a dissertação:

- Corpo do sistema em C/C++ com a integração da interface em Dynamic HTML
- Corpo do sistema em C/C++ com a interface em Active X
- Migração do sistema em C/C++ para a plataforma Microsoft .NET
- Interface em Java utilizando Java Native Interface para acessar o corpo do sistema em C/C++
- Desenvolvimento de um sistema modelo cliente/servidor em RMI[21] ou Corba[21] que pode ser acessado por uma interface Java.
- Desenvolvimento de um sistema cliente/servidor com a utilização de DCOM[27], Corba ou RMI.
- Utilização do pacote Cold Fusion da Macromedia/Allaire que é baseado em tecnologia Java e apresenta possibilidade de integração com outras linguagens.

Neste ponto vale salientar que os sistemas foram estudados sempre tendo como foco: o desenvolvimento de um aplicativo que possibilite a configuração, monitoração e teste da rede de sensores/atuadores na célula autônoma de manufatura. E sabendo-se que a rede de sensores/atuadores foi toda planejada e desenvolvida utilizando-se C/C++ que é a linguagem mais usada no meio industrial.

Os próximos tópicos apresentam um pequeno detalhamento de cada uma das seguintes técnicas. Assim justifica-se o porquê do emprego de cada uma das técnicas nesta dissertação.

[4] Resumidamente, trata-se da criação e principalmente a destruição dos objetos Java. A criação e a destruição são controladas por um sistema chamado de *Garbage Collection* onde nenhum objeto que não seja utilizado permanece criado no sistema operacional.

4.5.1.1 LINGUAGEM C/C++ & DYNAMIC HTML

Este foi o primeiro método estudado e foi encontrado na pesquisa no *site* do MSDN através do estudo de um artigo de autoria de Christian Gross[27] publicado na seção técnica do *site*. O método baseia-se na divisão da programação, através da separação entre a interface, os componentes e o controle da aplicação. Este método inicia-se com elaboração da interface, passando pela definição dos componentes e chegando por fim na implementação do controlador.

Os formatos de separação da interface da lógica têm as seguintes conseqüências:

Linguagem neutra: O controlador, a relação, e o componente são amarrados juntamente usando uma tecnologia distribuída de objetos, tornando mais simples adicionar uma execução ou uma nova interface do usuário (IU). Ao fazer mudanças subseqüentes deve-se usar uma linguagem apropriada a cada elemento arquitetural. Para o exemplo, a IU necessita ser dinâmica e, poder-se-ia conseqüentemente usar o HTML dinâmico, quando o componente da lógica necessitar usar uma linguagem de programação tal como o sistema de desenvolvimento visual de Microsoft® C++®.

Controlador-dependente: Um controlador mal projetado que não reflita os objetivos primários da aplicação causa problemas para os componentes que executam a relação do controlador tornando-os automaticamente obsoletos. Tal situação requer uma reescrita da lógica do sistema.

Detalhes de execução escondidos: Não é necessário que o controlador saiba onde os detalhes da interface do usuário originam. Desta maneira, o controlador pode comunicar-se como componente em uma outra máquina em uma outra Interface do Usuário (IU), se desejado. É inteiramente flexível, desde que o processo de registro seja inicializado pelo componente da lógica.

Simples: O design *pattern* oferece uma grande simplicidade uma vez que o projeto separa radicalmente o formato da interface do usuário (IU) da lógica de que a interface do usuário (IU) deve seguir.

Extensibilidade: Uma comunicação entre a interface do usuário e os componentes da lógica são um processo puramente dinâmico, que cada componente termine independentemente. O COM (Componente Object Model) faz esta comunicação dinâmica possível.

Neste formato de separação das exigências do teste padrão do projeto da lógica é encontrado tendo por resultado uma interface do usuário mais simples e como maior granularidade. Adicionalmente, a relação e o componente da lógica são independentes e fornecem meios excelentes para o melhoramento futuro de uma aplicação. Contudo, o sistema é um pouco complicado para se implementar comparando-se com os próximos métodos que serão discutidos adiante. A utilidade deste método torna-se eficaz para aplicações pontuais e não para grandes sistemas.

4.5.1.2 C/C++ COM O USO DO ACTIVEX

Os controles de *ActiveX* (anteriormente controles de OLE) são os objetos que podem ser introduzidos em páginas da *Web* ou em toda outra aplicação que for um recipiente do controle de *ActiveX*.

Os controles são uma arquitetura preliminar para desenvolver os componentes de software programáveis que podem ser usados em uma variedade de recipientes diferentes, incluindo *Web Browsers* na Internet. Todo o controle de *ActiveX* pode ser um controle Internet e pode adicionar sua funcionalidade a um aplicativo original ou ser parte de uma página na internet. Os controles em uma página da internet podem comunicar-se usando *scripts*.

Existem muitas vantagens em usar-se *ActiveX*, entre elas:

·Poucas relações são requeridas em comparação com os controles precedentes de OLE.

·A habilidade de ser um aplicativo que não depende de janelas para executar e também sempre ser executado no local onde é inicializado.

Entretando, assim como no item anterior o uso de C/C++ com ActiveX é um pouco complicado na adaptação de um sistema já existente para esta técnica. É uma técnica excelente para casos em que os desenvolvedores tenham completo domínio sobre os componentes do sistema existente e uma boa experiência com o uso dos controles ActiveX.

4.5.1.3 C# E A PLATAFORMA .NET

Nas últimas duas décadas C e C++ foram as duas linguagens mais difundidas e utilizadas no desenvolvimento de softwares comerciais e industriais. C e C++ ofereceram um grande poder aos desenvolvedores de software permitindo o acesso "ao baixo nível" das máquinas. Entretanto, este poder adquirido com a alta granularidade e flexibilidade da linguagem tornou o desenvolvimento em C e C++ muito improdutivo no que se refere ao tempo de desenvolvimento.

Com a utilização de outras ferramentas de desenvolvimento, a produtividade é muito maior se comparada com C e C++. Todavia, esta busca por uma maior produtividade custou a flexibilidade que os programadores em C e C++ tinham adquirido. A melhor solução para os programadores seria então combinar o rápido desenvolvimento com o poder de acessar todas as funcionalidades inerentes à plataforma.

A solução da Microsoft para este problema é a linguagem C# (leia-se "C Sharp"). C# é uma linguagem moderna, orientada a objetos que permite aos programadores desenvolverem um número enorme de aplicações para a nova plataforma .NET. Usando simples construtores da linguagem C#, estes componentes podem ser transformados em serviços XML, que podem ser acessados por qualquer linguagem em qualquer sistema operacional. Mais do que isto, C# foi

projetado com o objetivo de proporcionar um rápido desenvolvimento sem sacrificar o controle e o poder as funcionalidades do sistema operacional que foram as marcas registradas de C e C++[22].

C# mostra-se como sendo a solução Microsoft para o desnvolvimento de soluções onde até o momento utilizava-se predominantemente Java, em especial, em aplicações voltadas para a internet.

4.5.1.4 JAVA NATIVE INTERFACE

Java Native Interface (**JNI**) é uma interface nativa de programação. JNI permite que o código Java, que roda dentro de uma *Java Virtual Machine* (**JVM**), possa interoperar com aplicações e bibliotecas escritas em outras linguagens de programação, tais como C, C++ e *Assembly*.

O benefício mais importante de JNI é que a técnica não impõe nenhuma restrição na implementação de um aplicativo usando *Java Virtual Machine*. O suporte para uma JVM operar com JNI é feito pela simples adição do *plug-in* JNI à JVM.

Reafirmando alguns pontos importantes sobre a utilização de JNI. Utiliza-se basicamente JNI para os seguintes casos:

- Têm-se bibliotecas escritas em outra linguagem e estas são importantes para o aplicativo.
- Necessita-se implementar parte do código em linguagem de baixo nível como Assembly para melhoria do desempenho.

JNI vem destacando-se como solução porque apresenta uma alta flexibilidade quanto a integração entre Java com as demais linguagens de programação. Por Java ser uma ótima plataforma de desenvolvimento de sistemas já que possui as características de multiplataforma, cliente/servidor, suporte à concorrência, etc, já citadas e justificadas anteriormente, a solução com a utilização de JNI passa a ter grande peso. Além do mais, a grande aceitação no mercado da linguagem Java e também a existência da sua máquina virtual para a maioria dos dispositivos computacionais, desde de *palmtops*, celulares até computadores.

4.5.1.5 CORBA E RMI

Uma outra possível solução, só que mais restritiva, seria o uso de tecnologias para comunicação entre cliente/servidor. Neste tipo de arquitetura ter-se-ia um sistema distribuído onde cada elemento enviasse os dados relevantes para um servidor, que seria acessado por outros processos clientes.

Para trabalhar neste modelo temos duas arquiteturas existentes que buscam tornar-se o padrão universal para comunicação cliente/servidor, o *Common Object Request Broker Architecture* (**CORBA**), cujo maior interessado seria os desenvolvedores baseados na arquitetura C

e C++. O *Java/Remote Method Invocation* (**Java/RMI**) é uma tecnologia que tem grande suporte da SUN e dos outros desenvolvedores de sistemas em Java. O mercado tenta no momento realizar uma fusão entre as duas e produzir um padrão. No entanto, **CORBA** é a mais utilizada, porque Java já possui o suporte tanto para **CORBA** como para **RMI [22]**.

4.5.1.6 DCOM, CORBA E RMI

Esta solução seria mais para um estudo aprofundado das tecnologias de comunicação distribuídas, com o tipo cliente/servidor. Uma breve análise mostra que *Distributed Component Object Model* (**DCOM**) é um tanto complexa para implementar, porém mostrou-se bastante difundida em várias empresas e linguagens de programação. Agora, tanto *Common Object Request Broker Architecture* (**CORBA**) como também *Java/Remote Method Invocation* (**Java/RMI**) mostram-se bem adaptadas à filosofia de Orientação a Objetos. Em determinados sistemas Cliente/Servidor, o emprego deste tipo de tecnologia na implementação de sistemas poderia tornar os sistemas muito mais eficientes, observando-se pelo lado do desempenho. Com um processamento embarcado dos dados para uma resposta imediata ao Sensor/Atuador e através de uma comunicação, controle, configuração e monitoração remota de uma gama de sistemas operando em um chão-de-fábrica.

4.5.1.7 MACROMEDIA/ALLAIRE COLDFUSION

O *ColdFusion*, como ambiente de programação, suporta uma linguagem scripting poderosa, o *ColdFusion Markup Language* (**CFML**), que é extremamente fácil de aprender e integrar, clara como toda a linguagem popular da internet e suas tecnologias. *ColdFusion* trabalha com as arquiteturas mais conhecidas para a integração como **COM**, **CORBA**, e **EJB**. Pode também facilmente ser estendido com componentes novos criados em Java Servlets, ou classes de Java, ou C/C++ e outros milhares de componentes podem ser encontrados na seção *Developers Exchange* do site da Allaire[45].

4.5.1.8 COMPARAÇÃO FINAL ENTRE AS POSSIBILIDADES ESTUDADAS

Com o estudo superficial realizado sobre estas diferentes soluções para a implementação do sistema com uma interface do tipo *Web Browser* pode-se retirar algumas conclusões preliminares:

- A utilização do método com DCOM é um tanto complicada quanto ao desenvolvimento e não apresenta uma estruturação interessante. A própria compreensão do código de um simples exemplo se torna complicada pela utilização de nomes de tipos que não são "compreensíveis" logicamente. Talvez este método seja interessante para utilizar-se quando existe a necessidade de integrar sistemas legados (*legacy*).

- O uso de *ActiveX* se mostra uma boa solução. Já é um método mais estruturado com uma lógica para a implementação mais aprimorada e com uma relativa facilidade em programação. O que não se pode mensurar é o esforço que será necessário para disponibilizar os sistemas já existentes através desta metodologia.
- C# apresenta-se como a solução para o desenvolvedores em C++ para tornar seus sistemas disponíveis na internet e facilitar a integração e o *update* de aplicações ultrapassadas. No estudo realizado o sistema é grandioso, com o gigante Microsoft por trás, que apesar do sistema ser Microsoft, é talvez a melhor solução para o desenvolvimento de novos sistemas para a plataforma Windows. O porém neste caso é que o sistema apresenta alguns itens negativos:
  - Sistema totalmente novo, arquitetura, estilo, OOP.
  - Sistema que por ser novo deve possuir muitos problemas ainda.
  - A falta de dados estatísticos quanto ao desempenho, robustez e comparação com outros sistemas.
- JNI é o método que apresenta o modelo de programação mais claro entre os apresentados, o sistema para o desenvolvimento da interface é claro, conciso, robusto, flexível. Todavia JNI é Java, e com Java necessita-se de JVM que é um sistema lento (em máquinas com pouca memória RAM) e que possui alto consumo de memória, mas este problema é eliminado usando-se o sistema em máquinas com grande quantidade de memória RAM.
- Os dois últimos itens referem-se a sistemas que trabalham na arquitetura cliente/servidor, sendo o penúltimo método o uso do pacote Java & RMI a principal aposta, mas Java & CORBA não deixa nada a desejar.

Após este estudo comparativo pode-se justificar a utilização da técnica JNI como solução para o problema estudado nesta dissertação. Os pontos que justificam esta escolha estão descritos nos itens acima.

4.6 ESTRATÉGIA DE DESENVOLVIMENTO

Um fator que também merece destaque é apresentar e justificar a estratégia utilizada para o desenvolvimento da solução resultante desta dissertação de mestrado. A existência de uma resposta para a pergunta não é o suficiente para a perfeita solução do problema. Além de tudo, é necessário apresentar a solução de maneira concisa, coerente e correta acima de tudo. Para isto, baseado no estudo de metodologias de desenvolvimento sugeriu-se e adotou-se pelo grupo S2i (Sistemas Industrias Inteligentes) uma metodologia [15]. Esta metodologia busca ser genérica o bastante para

satisfazer no desenvolvimento tanto de soluções de software como de hardware. Além de uma metodologia para o desenvolvimento da solução, busca-se a utilização de ferramentas *freeware* no desenvolvimento da solução. Como resultado final, espera-se apresentar um sistema para o desenvolvimento de soluções tanto de software como hardware que seja econômica e totalmente aberta para a comunidade. O objetivo não é o domínio do conhecimento, da técnica e sim focalizar no desenvolvimento de soluções customizadas para o cliente.

4.6.1 PROCESSO DE DESENVOLVIMENTO

O processo de desenvolvimento não apresenta nada de novo que todos os autores e estudiosos de engenharia de software não conheçam. A única diferença é que a processo de desenvolvimento está adaptada às exigências do grupo S2i e do projeto de cooperação. Normalmente as metodologias de desenvolvimento têm como meta serem o mais genéricas possíveis, porém, a busca pela generalidade na maioria das vezes esbarra com a alta complexidade do sistema resultante. No processo de desenvolvimento abordado pela equipe do S2i tem como objetivo garantir um equilíbrio entre a flexibilidade e a generalidade das soluções para os trabalhos desenvolvidos pelo grupo.

Resumidamente o processo de desenvolvimento baseia-se me algumas fases macro (seguindo a metodologia clássica) que serão apresentadas a seguir: Proposta, Requisitos, Análise, Modelagem, Implementação, Testes, Documentação, Manutenção. A proposta se encarrega de apresentar uma visão superficial da solução para o problema apresentado. Na parte de requisitos são levantados todos os requerimentos, de todos os tipos para o desenvolvimento da solução. Já na fase de análise de requisitos, cada um dos pontos levantados e apresentados na etapa anterior devem ser justificados. Após a análise parte-se para a modelagem da solução, que é apresenta em forma de diagramas, fluxogramas, esquemas, etc. Com a modelagem finalizada, executa-se a fase de implementação propriamente dita da solução. A fase de testes acontece juntamente com a fase de implementação ou após ela, dependendo do tipo de solução, e nesta fase busca-se através de uma metodologia conhecida de testes colocar a solução nos mais variados “dilemas” e observar o comportamento. Durante todo o processo de desenvolvimento e inclusive no seu final a documentação do projeto é parte essencial nesta metodologia. Com ela espera-se suprimir e informar o *status* do projeto, o seu cronograma, apresentar eventuais problemas que aconteceram e como foram solucionados, etc. A última fase é a manutenção, a qual jamais se acaba e se as fases anteriores forem bem executadas esta fase não trará nenhum problema para os desenvolvedores.

Maiores informações sobre a metodologia de desenvolvimento podem ser encontradas no *site* do grupo S2i (http://s2i.das.ufsc.br)[15].

Esta metodologia foi empregada na sua totalidade nesta dissertação e a dissertação serve como exemplo da utilização da metodologia.

4.6.2 FERRAMENTAS

Como citado anteriormente busca-se utilizar ferramentas para o desenvolvimento que sejam *freeware*. Para tudo isto se tornar possível, foi feito um estudo durante meses com vários testes de ferramentas para o desenvolvimento de soluções. Procurou-se utilizar sistemas não proprietários para o desenvolvimento do sistema.

Os sistemas utilizados seguem:

- Dev C/C++ (Windows) e Kdeveloper (Linux) : Programação C/C++.
- JBuilder (Windows/Linux) : Programação Java.
- Doxygen (Windows/Linux) : Documentação software.
- S2iDoc : Documentação Online do Projeto.
- CVS (Windows/Linux) : Ferramenta para o Controle de Versão.
- ProxyDesigner (Windows/Linux) : Fluxogramas, Diagramação, Diagramas UML.

Maiores informações sobre estes sistemas podem ser também encontradas no *site* do grupo S2i (http://s2i.das.ufsc.br). Não é o objetivo deste documento apresentar cada uma das ferramentas utilizadas, somente informar, a caráter de exemplificação, que o desenvolvimento de sistemas abertos e com ferramentas livres é uma realidade atualmente.

5 PROJETO E IMPLEMENTAÇÃO DO SISTEMA *SAMWEB*

5.1 INTRODUÇÃO

Nos próximos items serão apresentados os passos executados para o desenvolvimento desta dissertação de mestrado. Cada um dos itens busca justificar o porquê do uso de tal conceito e/ou tarefa.

5.2 ETAPAS DE DESENVOLVIMENTO

O projeto do SAMWeb seguiu a metodologia de desenvolvimento proposta pelo Grupo S2i do Departamento de Automação e Sistemas (DAS)[15][38]. Nos próximos itens, cada uma das etapas de desenvolvimento será apresentada. Resumidamente as etapas são: Requisitios, Análise dos Requisitos, Modelagem, Implementação, Testes, Documentação e Manutenção. Cada um dos itens apresenta a sua estrutura característica. Isto acarreta que algumas informações são repetidas, porém o objetivo aqui é mostrar como é feito o desenvolvimento da solução.

5.2.1 REQUISITOS

Os requisitos dos SAMWeb foram levantados levando-se em consideração tanto o desenvolvimento da parte em Java como a integração entre Java e C/C++. Optou-se pela divisão nestes dois tópicos devido à complexidade de tratá-los conjuntamente.

5.2.1.1 REQUISITOS SAMWEB

Motivação: O *SAM Web Configuration Tool* faz parte do projeto de cooperação entre a Universidade Federal de Santa Catarina (UFSC) através do Departamento de Automação e Sistemas (DAS) e a Universidade Técnica de Aachen (RWTH-Aachen) através do Instituto de Máquinas-Ferramenta e Engenharia de Produção (WZL).

Objetivo: O SAMWeb objetiva o desenvolvimento de uma ferramenta para configuração, monitoração e testes dos Sensores/Atuadores existentes em uma célula autônoma de manufatura. O *Sensor/Actuator Module* (SAM) é um sistema pertencente a um projeto maior o SFB 368/APC. O projeto SFB 368 tem como objetivo o desenvolvimento de uma célula autônoma de manufatura. Maiores informações sobre os projetos podem ser encontradas no *site*: http://sfb368.rwth-aachen.de.

Situação Atual: A atuação sobre os Sensores/Atuadores na Célula Autônoma de Manufatura é atualmente feita através de aplicativos para controle através do MS-DOS, chamado *Command Line Application*, através de um aplicativo sobre a plataforma **OSACA**. A interação com

a célula autônoma de manufatura é basicamente realizada dentro de uma rede específica (TCP/IP - Ethernet), dentro de uma plataforma específica (Windows) até o momento.

Resultados Esperados: Com o desenvolvimento do SAMWeb espera-se poder contar com uma ferramenta capaz de realizar a configuração, análise, monitoração e debug dos dados de sensores e atuadores da célula autônoma de manufatura independente da plataforma operacional e do local onde se encontra a célula.

O SAMWeb será mais uma ferramenta de apoio para a realização de tais tarefas, porém de um modo mais simples, ágil e rápido. O SAMWeb deve ser independente do sistema operacional em que se encontra o usuário do sistema e também deve possibilitar a análise dos dados de um SAM *server* em qualquer lugar através de um *Web Browser*, desde que tenha-se acesso à Internet.

Requisitos Gerais: Segue abaixo a lista com os principais requisitos do sistema SAMWeb:

- Interface amigável, simples e fácil.
- Linguagem do sistema deve ser internacional, ou seja, Inglês.
- Utilização dos mais avançados recursos gráficos.
- Interface de comunicação padrão com SAM.
- Modelagem do sistema por completo e várias partes.
- Programação dentro das normas estabelecidas pelo projeto de cooperação.
- Programação da biblioteca JNI seguindo os padrões ANSI C/C++.
- Programação do aplicativo Java© utilizando apenas as bibliotecas padrão (JDK).

Requisitos Específicos: Segue abaixo a lista dos requisitos específicos do SAMWeb:

- O SAMWeb deve ser capaz de ler dados dos arquivos de configuração do SAM *servers*, arquivos estes chamados de SAMConfig.dat. Cada servidor possui o seu .dat.
- O SAMWeb deve ser capaz de escrever novos valores para os sensores e atuadores, ou seja, deve ser capaz de gerar um novo arquivo de configuração para a rede.
- O SAMWeb deve ser capaz de realizar o *debug* dos novos valores para os sensores e atuadores, ou seja, observar se estão ou não dentro dos valores requeridos.
- Com o SAMWeb deve ser possível realizar a monitoração *soft real time* de valores de determinados Sensores/Atuadores através de um gráfico.
- O SAMWeb deve ser capaz de poder enviar uma mensagem de reinício dos demais sistemas do SAM para atualização dos dados.
- Um *help* para cada campo do SAMWeb deve existir a cada vez que o usuário posiciona o mouse sobre o item (*tooltip*) e também um *help* para o sistema como um todo deve estar acessível durante todo o tempo.

- O SAMWeb deve deixar claro a todo momento em que estado que ele se encontra: estado de monitoração, estado de leitura, estado de escrita, estado de *debug*, ou em reinicialização.
- A comunicação entre o SAMWeb com o SAM será feita através de JNI (*Java Native Interface*).

Após a apresentação dos requisitos para o SAMWeb passou-se para a parte de montagem dos requisitos da interface de comunicação JNI entre o *applet* Java e a biblioteca C++.

5.2.2 ANÁLISE DOS REQUISITOS

Neste tópico, verifica-se as implicações dos requisitos, descrevendo em um documento todas as conseqüências destes requisitos. Nesta fase, também se propõe uma estrutura que atenda os requisitos descritos.

É importante descrever todos as seqüências de tarefas primordiais utilizando-se diagramas, podendo-se valer do uso de UML. Esta análise e a estrutura proposta devem ser explanadas em um documento, que será apresentado e submetido à avaliação e à aprovação de todos os envolvidos no projeto.

Como na etapa de requisitos, aqui também se optou por uma divisão na análise de requisitos para o SAMWeb e para a interface de integração JNI entre o *applet* Java e a biblioteca C/C++.

5.2.2.1 ANÁLISE DOS REQUISITOS DA INTERFACE ENTRE O SAM E O SAMWEB

Motivação: Neste documento apresenta-se os requisitos para a modelagem da interface padrão entre o SAMWeb e o sistema SAM propriamente dito.

O SAMWeb é uma ferramenta para configuração, visualização, monitoração e debug de dados provenientes de SAM *Servers*. Os SAM *Servers* são servidores que possuem informações sobre os Sensores/Atuadores presentes em um sistema autônomo de produção.

Situação Atual: O *Sensor/Actuator Module* (SAM) já possui um modelagem completa para o sistema rodar e executar as suas funções independentemente. O SAM *Modelling* exemplifica através de um diagrama de classes todas as interações internas e externas do sistema e seu relacionamento.

Entre as ferramentas externas ao SAM destacam-se: OSACA *Application* e o *Command Line Application*. O SAMWeb é mais uma ferramenta externa que vem aumentar e ampliar as funcionalidades do SAM.

Resultados Esperados: A figura 5.1 serve para exemplificar como o SAMWeb se relacionará com o sistema SAM.

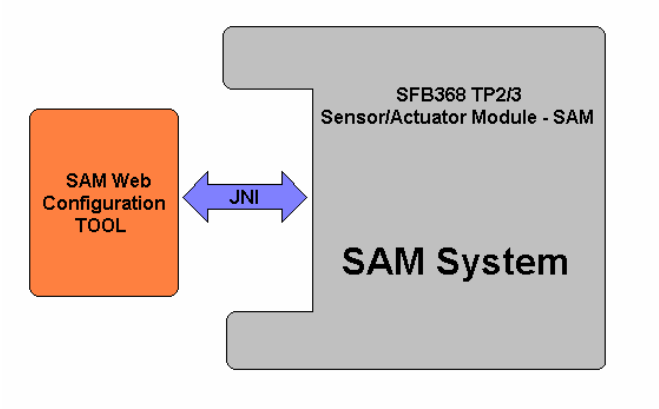


Fig. 5.1: Modelo de Interação entre o SAMWeb e o SAM através do JNI

Para que ambos os sistemas funcionem perfeitamente é preciso que o sistema cliente, neste caso sendo o SAMWeb, possua uma interface padrão com o SAM. Para o desenvolvimento desta interface utiliza-se JNI (*Java Native Interface*).

Como resultado primordial busca-se desenvolver um aplicativo que facilite a configuração, visualização, monitoração e debug de dados provenientes do sistema SAM.

Requisitos: Segue a lista de requisitos necessários para que a interface padrão entre o SAMWeb e o sistema SAM funcione perfeitamente.

- Modelagem e desenvolvimento de uma biblioteca em C/C++ que possua as funcionalidades necessárias e suficientes para acesso ao sistema SAM.
- Modelagem e desenvolvimento da interface em Java com auxílio de JNI para acessar as funções escritas em C/C++.
- As funções devem todas estar contidas nesta biblioteca, todo e qualquer acesso ao SAM deve ser realizado por esta interface.
- Programação da biblioteca deve seguir as regras de desenvolvimento estabelecidas pelo projeto de cooperação entre a UFSC/DAS & a RWTH-Aachen/WZL.
- Programação em C++ deve seguir o padrão ANSI C.
- Programação em Java deve utilizar o JDK (Java Development Kit) como base e somente este.

Após levantados os requisitos para o sistema SAMWeb e também para a interface entre o *applet* Java e a biblioteca C/C++ passou-se para a segunda etapa do processo de desenvolvimento da solução que é análise dos requisitos.

5.2.2.2 ANÁLISE DE REQUISITOS DO SAMWEB

Nos parágrafos seguintes serão apresentados os diagramas e os fluxogramas que apresentarão as bases para o funcionamento do SAMWeb. Neste ponto do processo de desenvolvimento apenas uma visão *macro* do sistema é apresentada.

Primeiramente é necessário estruturar como irá funcionar o SAMWeb, ou seja, apresentar através de um diagrama de seqüência (figura 5.2) como será o relacionamento Cliente/Servidor do SAMWeb com os servidores do SAM.

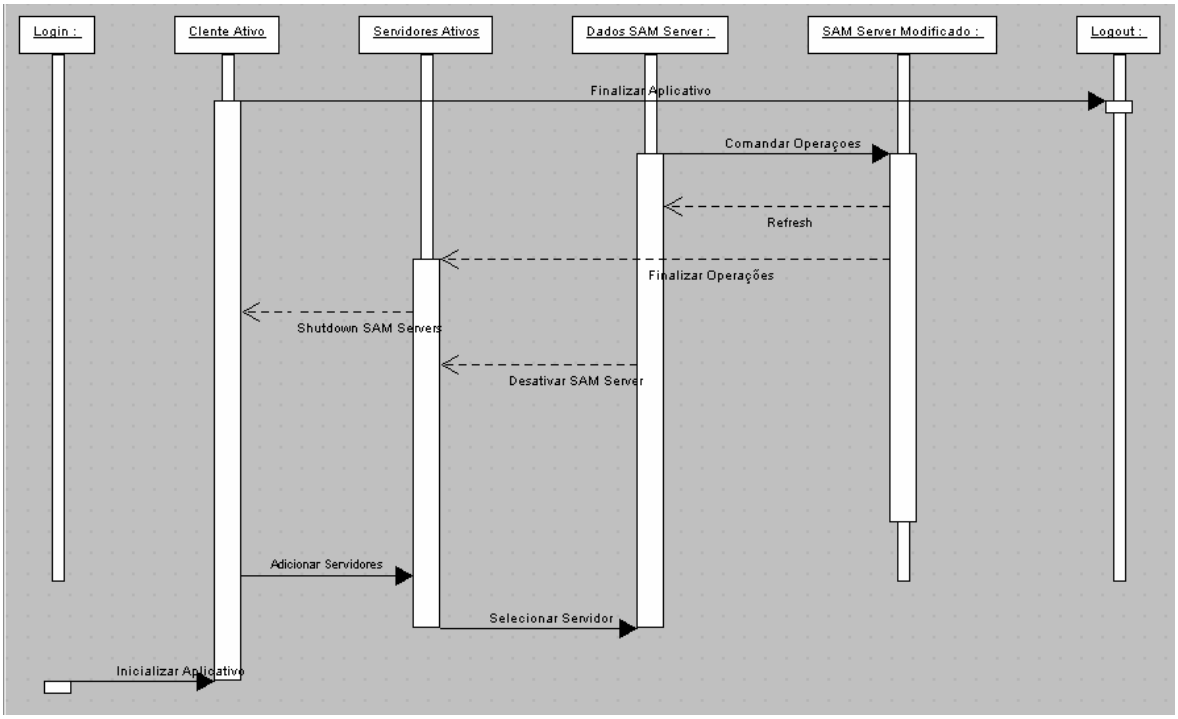


Fig. 5.2: Diagrama de sequência do SAMWeb

Agora que o modelo Cliente/Servidor já foi apresentado, passa-se a uma explicação dos diversos módulos que estão presentes no SAMWeb. Claramente, o SAMWeb apresenta um módulo para a listagem dos servidores SAM (no futuro também os clientes), um módulo para controle da ferramenta de navegação entre uma tela e outra (*Tabs containers*), um módulo individual para cada *macro* atividade do SAMWeb, ou seja, visualização dos dados, configuração (escrita e debug) de dados e monitoração. Outro módulo para controle de acesso e validação de usuários. Gerenciando e coordenando os eventos de todos os outros módulos um módulo gerenciador ou *main* do SAMWeb.

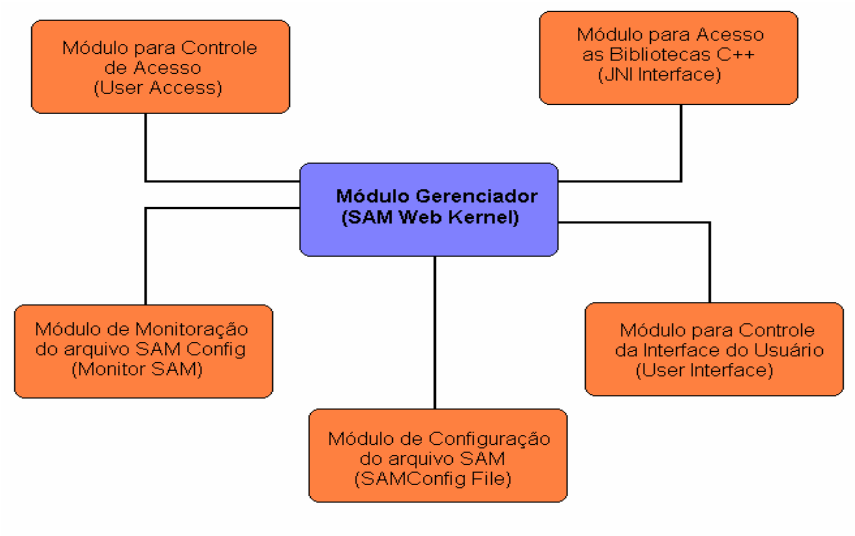


Fig. 5.3: Módulos do SAMWeb

Para um perfeito entendimento, nos passos seguintes, cada módulo do SAMWeb (figura 5.3) será apresentado individualmente. Com isto espera-se que um melhor entendimento das funcionalidades necessárias e suficientes de cada módulo.

Módulo para Controle de Acesso

Como o SAMWeb será um aplicativo cliente que além de visualizar os dados sobre processos servidores SAM, é um *applet* que realiza modificações nos valores de sensores e atuadores através da interface amigável. Por este motivo, um controle de acesso dos usuários do aplicativo é necessário para evitar problemas quanto ao bom funcionamento do sistema como um todo e também evitar que usuários indesejáveis acessem o sistema SAM. O controle de acesso será baseado em um aplicativo simples já implementado anteriormente. O módulo de segurança desde aplicativo não é primitivo, todavia, este não é um requisito fundamental para a primeira etapa. O sistema estará funcionando dentro de uma área restrita em uma rede restrita a poucas pessoas.

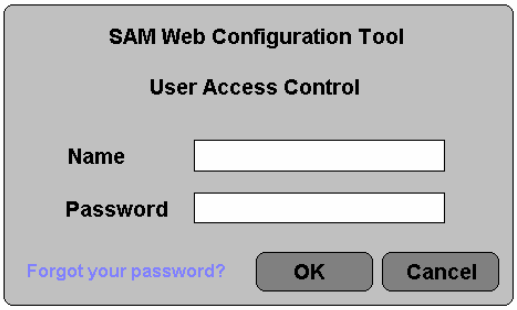


Fig. 5.4: Controle de Acesso ao SAMWeb

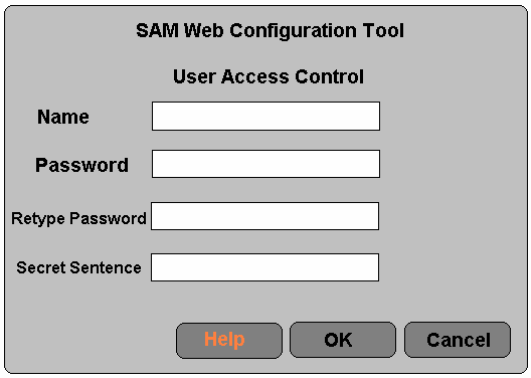


Fig. 5.5: Cadastro de Acesso ao SAMWeb

Módulo de Monitoração

Este módulo tem como objetivo principal apresentar ao usuário o desempenho do valor de um Sensor/Atuador no decorrer do tempo. Com isto, o usuário pode observar o desempenho histórico do valor de uma variável e também atuar sobre qualquer irregularidade que possa aparecer no tempo. Como o módulo anterior, este módulo também já foi implementado anteriormente através de um *applet* demo. Este *applet* lê o valor da memória que está sendo usada pelo computador no decorrer do tempo. Ele é baseado em uma *Thread* e toda a interface é construída de forma a aceitar as características *runable* do *applet*.

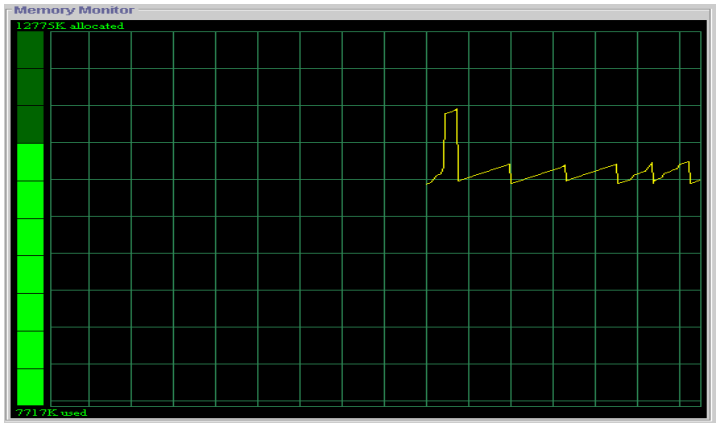


Fig. 5.6: Monitoração no SAMWeb

Este demo apresentado acima servirá de base para a montagem do módulo monitor para o SAMWeb. A mudança principal é a alteração das variáveis do gráfico que passarão a ser os limites superior e inferior para o valor do sensor/atuador escolhido. O gráfico será montado baseado nos valores constantes no arquivo de configuração do SAM. Se o arquivo for alterado, e o valor da variável sofrer alteração isto deve ser mostrado para o usuário caso o mesmo esteja nesta tela. Ao contrário o monitor ficará lendo o mesmo valor para a variável até que o usuário mude de tela.

Módulo da Interface do Usuário

A interface do sistema foi planejada a muito tempo atrás. Algumas modificações tornam-se necessárias por limitações encontradas durante este tempo e outras pelo estudo e elaboração de melhorias para um melhor funcionamento do SAMWeb. Para facilitar a navegação e entendimento, o aplicativo possui, um menu com as funções principais, na árvore com a lista dos servidores (e clientes no futuro), uma barra de *TABs* facilita a mudança de um modo para outro facilmente, além de várias instruções aparecerem na barra de status e também nos *tooltips* e com um *help* avançado é possível conseguir ajuda para cada item selecionado. O aplicativo tem dois grandes ambientes, mais à esquerda temos a tela com a árvore dos servidores SAM e no centro e à direita temos as telas dos modos de operação. A figura abaixo apresenta um modelo simples da tela do SAMWeb.

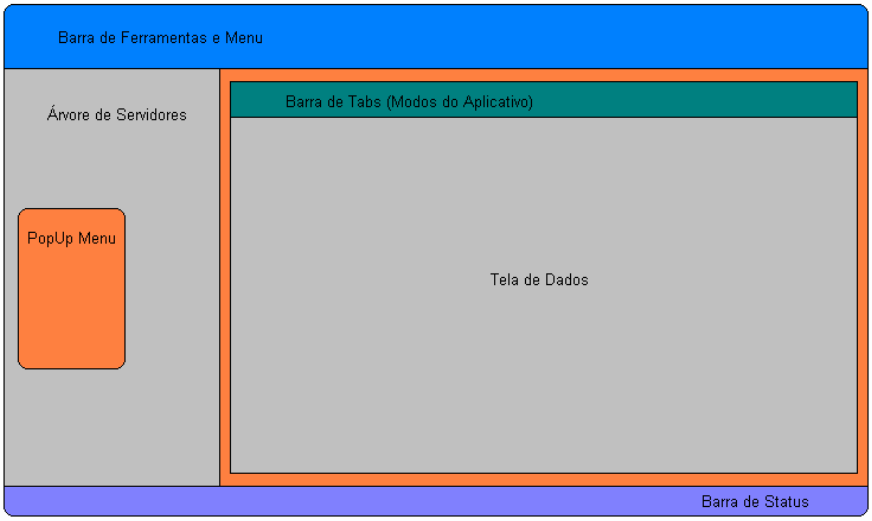


Fig. 5.7: Ambientes do SAMWeb

Módulo de Configuração

Como cada SAM Server possui um arquivo de configuração próprio, através deste arquivo é possível monitorar, visualizar, atualizar, alterar os dados dos Sensores/Atuadores de um SAM *server*. Assim no módulo de configuração temos as funcionalidades de leitura & escrita do arquivo de configuração. Este módulo é responsável por interpretar o arquivo de configuração, analisá-lo e ler os dados que são relevantes para o usuário, além de servir como base para a interpretação dos valores dos dados que ali se encontram. Uma tabela de símbolos sempre se faz presente neste arquivo o que facilita o entendimento pelo usuário de cada valor de cada Sensor/Atuador que está sendo analisado.

Módulo da Acesso às Bibliotecas Externas

Este módulo caracteriza-se por possuir as funções que realizam a comunicação efetiva com o sistema SAM para obter informações sobre os SAM *servers* e criar uma ponte de comunicação com o SAMWeb. Como todo o sistema SAM foi programado em linguagem C/C++ e o SAMWeb

é um applet Java, este módulo é programado utilizando–se da metodologia JNI (*Java Native Interface*), onde funções são declaradas de forma especial no código Java e são implementadas em C/C++ em uma biblioteca especial que fornecerá os resultados das funções.

5.2.2.2.1 Análise dos Requisitos Específicos

- O SAMWeb deve ser capaz de ler dados dos arquivos de configuração do SAM *servers*, arquivos estes chamados de SAMConfig.dat.

O arquivo SAMConfig.dat possui todas as tabelas com os dados sobre os dispositivos, tempos de *refresh*, e símbolos que encontram-se no SAM. É essencial que o SAMWeb tenha a capacidade de localizar este arquivo quando um SAM server é inicializado e possa realizar a leitura para que o conteúdo relevante seja apresentado para o usuário.

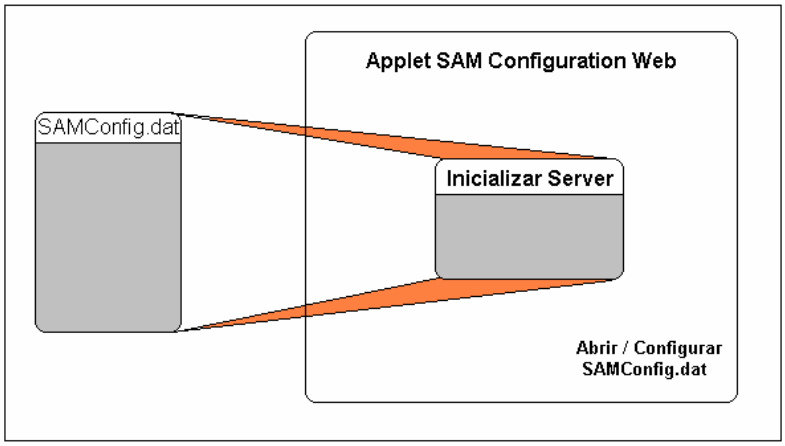


Fig. 5.8: Modelo da acesso entre o SAMWeb com o arquivo de configuração

- O SAM Web deve ser capaz de escrever novos valores para os sensores e atuadores, ou seja, deve ser capaz de gerar um novo arquivo SAMConfig.dat.

O usuário poderá além de observar os valores da tabela de dados do SAM s*erver* editar novos valores para os sensores e atuadores, estes valores serão salvos no arquivo de configuração do SAM (SAMConfig.dat) e após um período (ciclo de varredura) todos os servidores e clientes serão avisados da mudança.

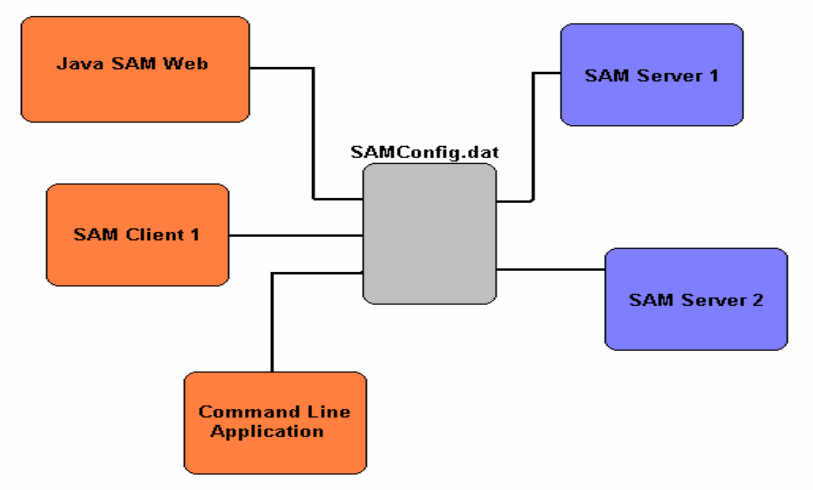


Fig. 5.9: Configuração e editoração

- O SAMWeb deve ser capaz de realizar o debug dos novos valores para os sensores e atuadores, ou seja, observar se estão ou não dentro dos valores requeridos.

Como o SAMWeb vai permitir que o usuário edite a tabela de dados do SAM é extremamente necessário que os valores que o usuário entre não comprometam o perfeito funcionamento do sistema. Uma maneira de controlar as entradas do usuário é configuração dos campos da tabela de dados do *applet* de tal forma que permita apenas a entrada de dados válidos. Este controle pode ser realizado de diferentes maneiras:

Configuração das células para que onde são necessários dados numéricos o usuário só possa editar números e em células de texto o mesmo princípio.

Para valores que se tem apenas um conjunto finito de valores possíveis apresentar ao usuário em forma de *select* as opções e aguardar pelo usuário escolher a correta.

Se possível, após as modificações realizadas pelo usuário, o sistema analisa cada campo da tabela e avalia se os valores estão dentro dos das especificações válidas, caso contrário uma mensagem de alerta e indicação da solução deve aparecer.

- Com o SAMWeb deve ser possível realizar a monitoração *in time* de valores de determinados sensores/atuadores através de um gráfico.

A monitoração dos valores dos sensores/atuadores também é essencial para que o usuário tenha noção do que vem acontecendo com o sistema no decorrer do tempo e também avaliar possíveis modificações boas ou ruins que possam ocorrer no sistema e sendo o caso resolvê-las.

- O SAMWeb deve ser capaz de poder enviar uma mensagem de reinício dos demais sistemas do SAM para atualização dos dados.

Como já foi levantado anteriormente, caso o arquivo de configuração do SAM seja alterado pelo SAMWeb os demais clientes e servidores devem saber que o SAMConfig.dat foi alterado. Isto se pode dar por duas maneiras: primeiro, o SAMWeb pode enviar um evento notificando todos os elementos do sistema que mudanças ocorreram; segunda, o sistema como um todo realiza um *refresh* do arquivo de configuração a cada *X* milisegundos, assim, apesar de um pequeno *delay* todos os componentes do sistema SAM irão atualizar os dados.

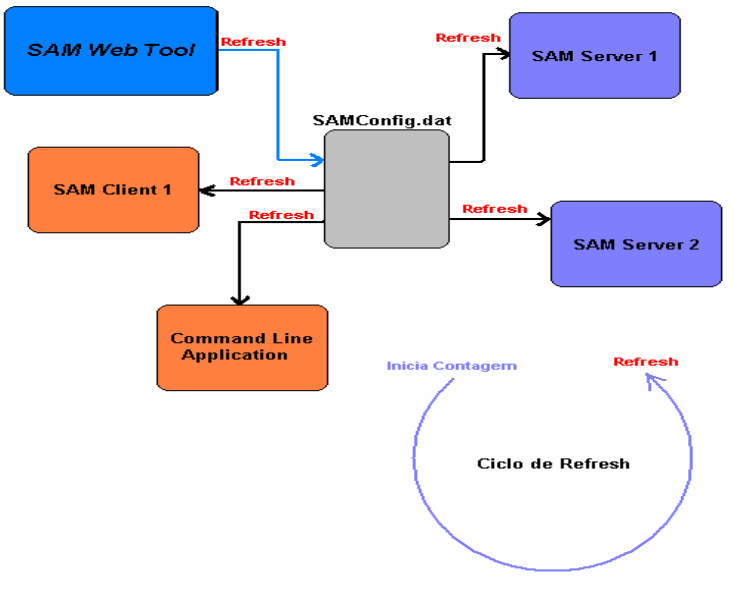


Fig. 5.10: Ciclo de Refresh

- Um *help* para cada campo do SAM Web deve existir. A cada vez que o usuário posiciona o mouse sobre o item aparece uma mensagem (*tooltip*) e também um *help* para o sistema como um todo deve estar acessível durante todo o tempo.

Como é natural em qualquer sistema moderno um *help* muito bem estruturado e com informações claras, precisas e simples é essencial para a aceitação de um novo sistema. O SAMWeb propõem-se a seguir este modelo onde a ajuda para o usuário estará disponível a todo instante para o usuário. Esta ajuda pode ser inteligente de tal forma a indicar para o usuário os tópicos relacionados a cada funcionalidade do *applet*.

- O SAMWeb deve deixar claro a todo momento em que estado que ele se encontra: estado de monitoração, estado de leitura, estado de escrita, estado de *debug*, ou em reinicialização.

O *applet* deve estar dividido em interfaces individuais para cada macro funcionalidade do SAMWeb, assim ficará fácil para identificar se o *applet* encontra-se na parte de visualização dos dados do SAM, monitoração de determinado elemento, ou *debug* de uma nova configuração. Todavia, como uma lista de servidores será visualizada em uma "árvore de SAM *Servers*" é importante que o SAM *Server* que estiver sendo analisado no momento apareça destacado na árvore e também esteja visível seu nome nas diversas interfaces.

- A comunicação entre o SAMWeb com o SAM *server* será feita através de JNI (*Java Native Interface*).

Modelo de comunicação o qual baseia-se no modelo apresentado a seguir e mais detalhes podem ser encontradas no *site* da *Sun Microsystems* (www.java.sun.com).

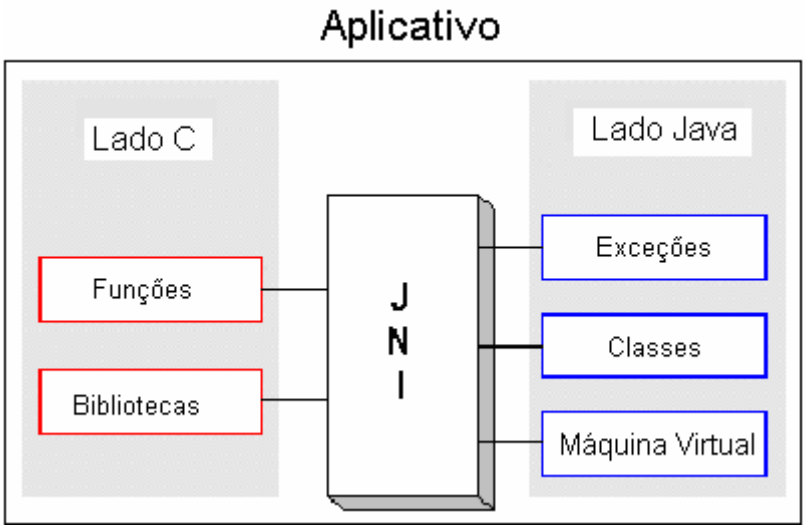


Fig. 5.11: Interação entre Java e C

5.2.2.3 ANÁLISE DE REQUISITOS DA INTERFACE JNI

Neste documento apresenta-se uma discussão sobre os requisitos para a modelagem da interface padrão entre o SAMWeb e o sistema SAM propriamente dito.

Segue a lista com a análise dos requisitos para a interface padrão entre o SAMWeb e o sistema SAM:

- Modelagem e desenvolvimento de uma biblioteca em C/C++ que possua as funcionalidades necessárias e suficientes para acesso ao sistema SAM.

A biblioteca desenvolvida em C++ deve possuir todas as funções necessárias para que o modelo Cliente/Servidor funcione corretamente. Também funções específicas para localização,

leitura de dados, configuração de arquivos e dados nos servidores, interpretação da tabela de símbolos, *help* sobre os comandos devem estar presentes na biblioteca.

- Modelagem e desenvolvimento da interface em Java com auxílio de JNI para acessar as funções escritas em C/C++.

A interface Java limita-se a uma classe. Esta classe possui declarados todos os métodos que serão necessários para o funcionamento do modelo Cliente/Servidor e das demais funções para o SAMWeb. Estes métodos são chamados de *native methods* em Java.

- As funções devem todas estar contidas nesta biblioteca. Todo e qualquer acesso ao SAM deve ser realizado por esta interface.

Este requisito é claro para que a interface com métodos nativos funcione bem. É necessário que todas as funcionalidades necessárias e suficientes do aplicativo constem declaradas na classe Java e também implementadas na biblioteca C/C++.

- Programação da biblioteca deve seguir as regras de desenvolvimento acordadas no projeto de cooperação.

Toda a análise, modelagem, implementação, testes e documentação devem seguir as regras de nomenclatura existentes no S2i e também que fazem parte do sistema de desenvolvimento no grupo responsável pelo projeto na Alemanha.

- Programação em C++ deve seguir o padrão ANSI C++.

A implementação de biblioteca deve utilizar apenas funções standard C/C++ que fazem parte do padrão ANSI C. Assim garante-se independência da biblioteca quanto ao sistema operacional e quanto à ferramenta de desenvolvimento. Uma biblioteca deve ser compilada para os dois sistemas operacionais mais utilizados no mercado. Uma compilação para a plataforma Windows (biblioteca .dll) e outra para a plataforma LINUX (biblioteca .lib).

- Programação em Java deve utilizar o JDK (*Java Development Kit*) como base e somente este.

Toda a programação do SAMWeb deve utilizar apenas os recursos existentes no pacote de desenvolvimento padrão Java, ou seja, o JDK. O SAMWeb será desenvolvido para atuar em sistemas de grande porte e com tecnologia de ponta. Assim, todo o sistema será desenvolvido usando-se o que há de mais poderoso atualmente no JDK.

5.2.3 MODELAGEM

A etapa de modelagem procura responder a seguinte pergunta: Como funciona internamente? Aqui se constrói os diagramas de classes genéricos e completos, preferencialmente em UML. Estes diagramas servirão como esqueleto para o projeto, descrevendo as interfaces que serão disponibilizadas publicamente para os demais módulos.

Esta modelagem também será apresentada para a equipe. E somente a após a aprovação deste pela equipe, é que o desenvolvimento tem seqüência.

5.2.3.1 MODELAGEM SUPERFICIAL DAS INTERFACES

O passo agora é o desenvolvimento da modelagem propriamente dita da interface JNI. Para esta modelagem utiliza-se o sistema SAM como base e também uma das suas aplicações, o *Command Line Application*, o qual é uma ferramenta que permite executar comandos no sistema SAM baseado no MS-DOS.

O primeiro passo neste processo de modelagem da interface JNI é a modelagem da biblioteca C/C++ que efetivamente servirá como ponte de ligação entre o código implementado em C/C++ e Java. Como foi dito anteriormente, no lado Java a classe apenas declara quais são os métodos que ela necessita, enquanto que no lado C/C++, estes métodos são implementados.

5.2.3.1.1 Modelagem da Interface no lado C/C++

A biblioteca é composta por um aplicativo MS-DOS simples que permite testar os comandos. Este aplicativo foi desenvolvido pelos senhores Dominic Sack e Alexandre Orth. Algumas mudanças são necessárias, mas como base para desenvolvimento do aplicativo, o *Command Line Application* serve de subsídio na modelagem do modelo de implementação.

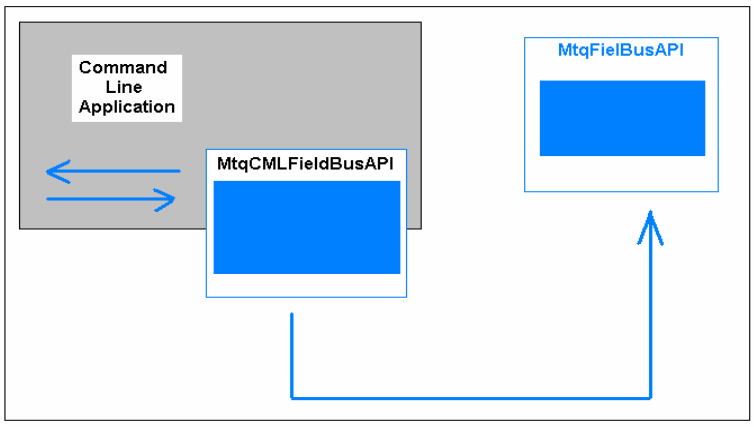


Fig. 5.12: Modelo de Interação do Aplicativo com a Biblioteca

Baseado no modelo de interação acima se pode criar o modelo de integração entre a biblioteca C/C++ e o SAM. Esta interação é possível através da classe **MtqFieldBusAPI**.

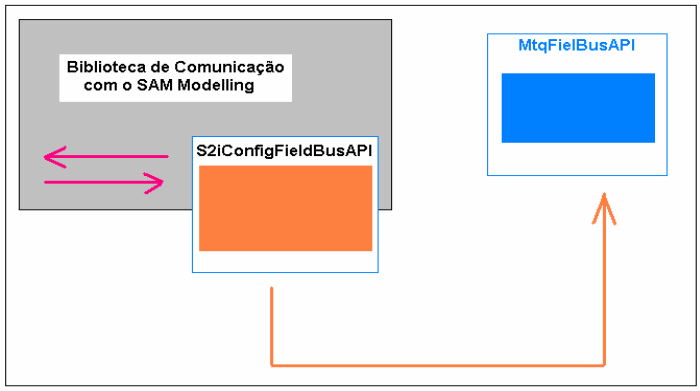


Fig. 5.13: Modelo Básico de Comunicação entre o Aplicativo e a Biblioteca

Agora se pode montar um diagrama que mostra uma visão global da biblioteca de comunicação entre a aplicação em Java (SAMWeb) com o SAM. A figura a seguir é apenas de caráter ilustrativo não levando em consideração os detalhes que serão explicados na etapa posterior (implementação).

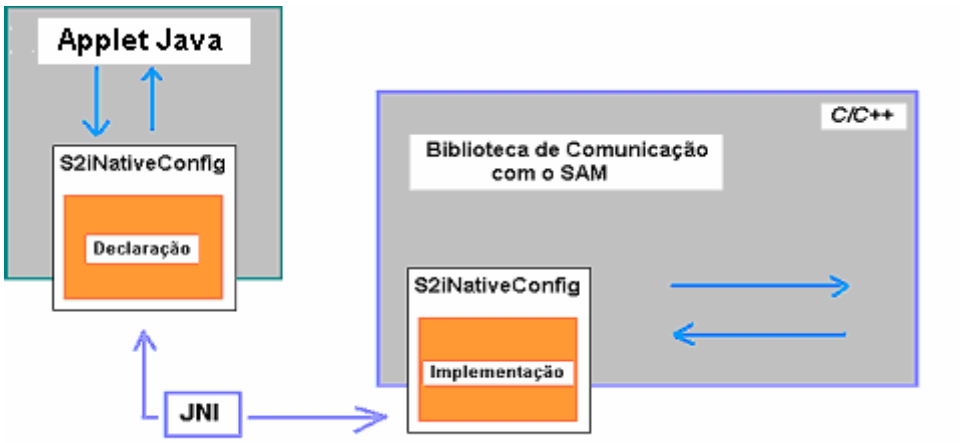


Fig. 5.14: Modelo da comunicação com entre a biblioteca em C com o aplicativo Java

5.2.3.1.2 Modelagem da Interface no lado Java

Neste ponto necessita-se definir como o aplicativo Java irá acessar a biblioteca C/C++, que funções são necessárias para acessar a biblioteca C/C++ e qual é o seu tipo. Novamente, os nomes para os componentes são apenas ilustrativos.

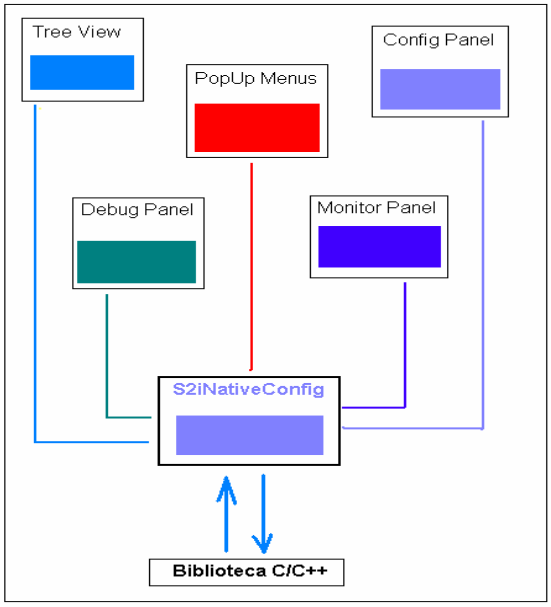


Fig. 5.15: Interface das demais classes Java com a classe Nativa (JNI)

Agora na próxima etapa será realizada a modelagem detalhada de cada um dos componentes o sistema. Esta modelagem detalhada não ficará restrita à interface JNI, mas também, realizar-se-á a modelagem do sistema como um todo.

5.2.3.2 *MODELAGEM DETALHADA*

A modelagem detalhada do SAMWeb foi baseada na modelagem usada para o desenvolvimento do SAM. Os diagramas mais usados foram o diagramas de classe, *use case* e também de seqüência, seguindo a metodologia UML. No princípio utilizou-se um software para a modelagem dos diagramas UML de uma empresa Alemã, porém o software não é mais vendido e a empresa mudou totalmente de ramo. Assim, passou-se a utilizar as ferramentas de design da TogetherSoft [14], uma empresa que fornece ferramentas para o desenvolvimento de sistemas tanto em C/C++ como Java.

Com a ferramenta de auxílio no processo de modelagem detalhada pode-se especificar cada uma das partes componentes do SAMWeb e também modelar a sua interação com a estrutura já proposta e implementada para o SAM.

O modelo detalhado do SAMWeb passou por várias seções de testes e validação pela equipe do grupo S2i antes de ser aprovada e desenvolvida. Este passo faz parte da metodologia desenvolvida pelo grupo S2i no processo de desenvolvimento dos sistemas [15].

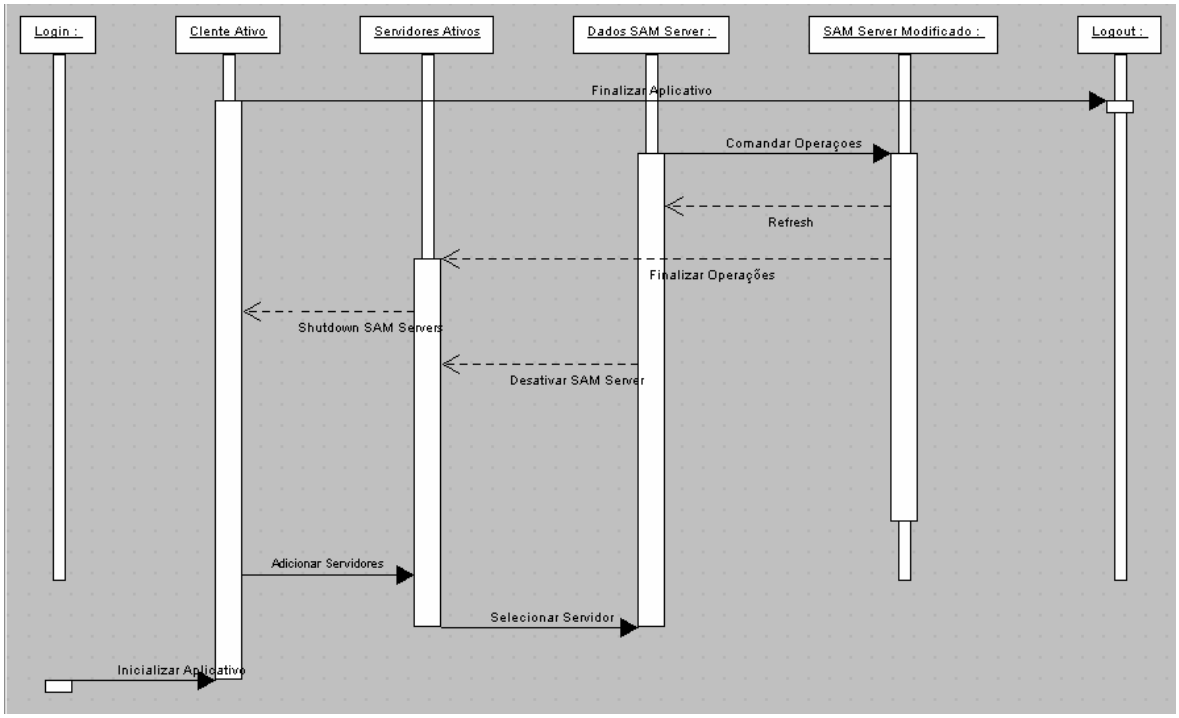


Fig. 5.16: Diagrama de Seqüência do SAMWeb

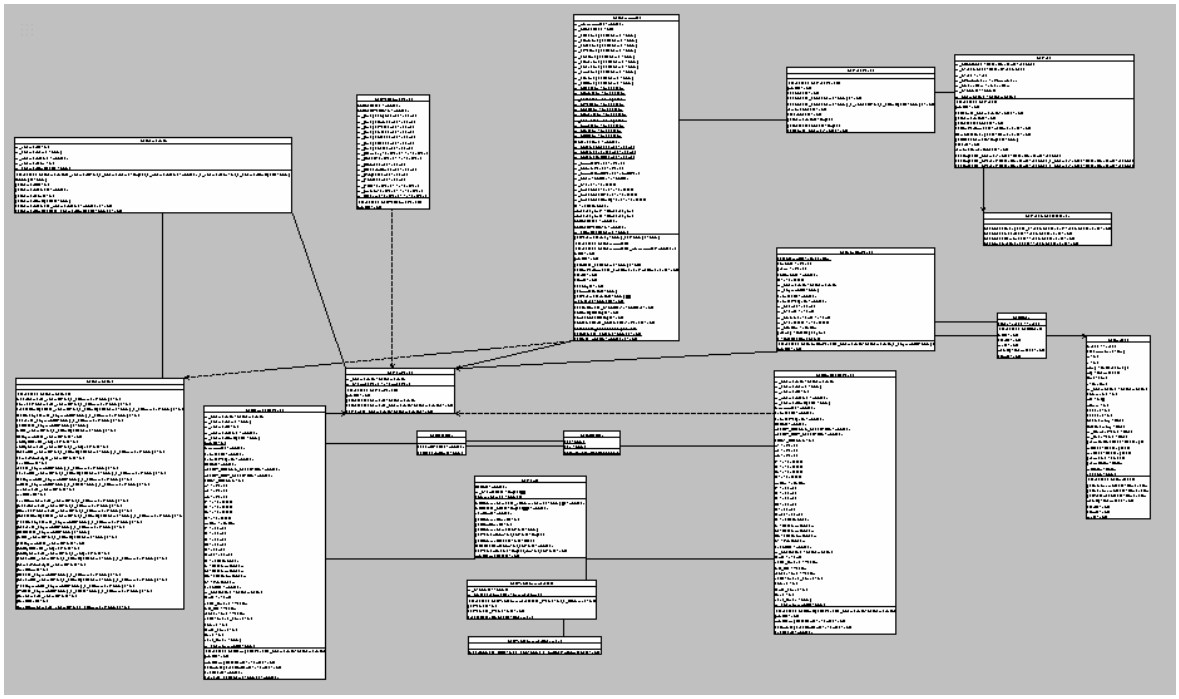


Fig. 5.17: Diagrama de Classe do SAMWeb

5.2.4 IMPLEMENTAÇÃO

Nesta etapa, parte-se para a implementação da modelagem da etapa anterior na linguagem de desenvolvimento escolhida, tomando-se os devidos cuidados e seguindo a norma de nomenclatura do grupo.

O sistema foi implementado utilizando-se duas linguagens, C/C++ para a programação da biblioteca que permite o acesso à rede de Sensores/Atuadores, e Java para a programação da interface e dos módulos do SAMWeb como um todo.

Os tópicos a seguir mostrarão apenas fragmentos do código utilizado no desenvolvimento do SAMWeb a título de exemplo. Maiores detalhes podem ser encontradas no *site* do projeto no *website* do grupo S2i (http://s2i.das.ufsc.br) ou no anexo deste documento.

O código foi implementado seguindo as regras de nomenclatura e a estrutura proposta pelo grupo. Esta norma é baseada nas normas existentes e que são mais difundidas dentre os pesquisadores. A geração do código comentado em HTML foi possível através da utilização da ferramenta *freeware* de geração de código chamada Doxygen [16]. A ferramenta já foi citada anteriormente e informações sobre a mesma podem ser conseguidas no *site* do S2i.

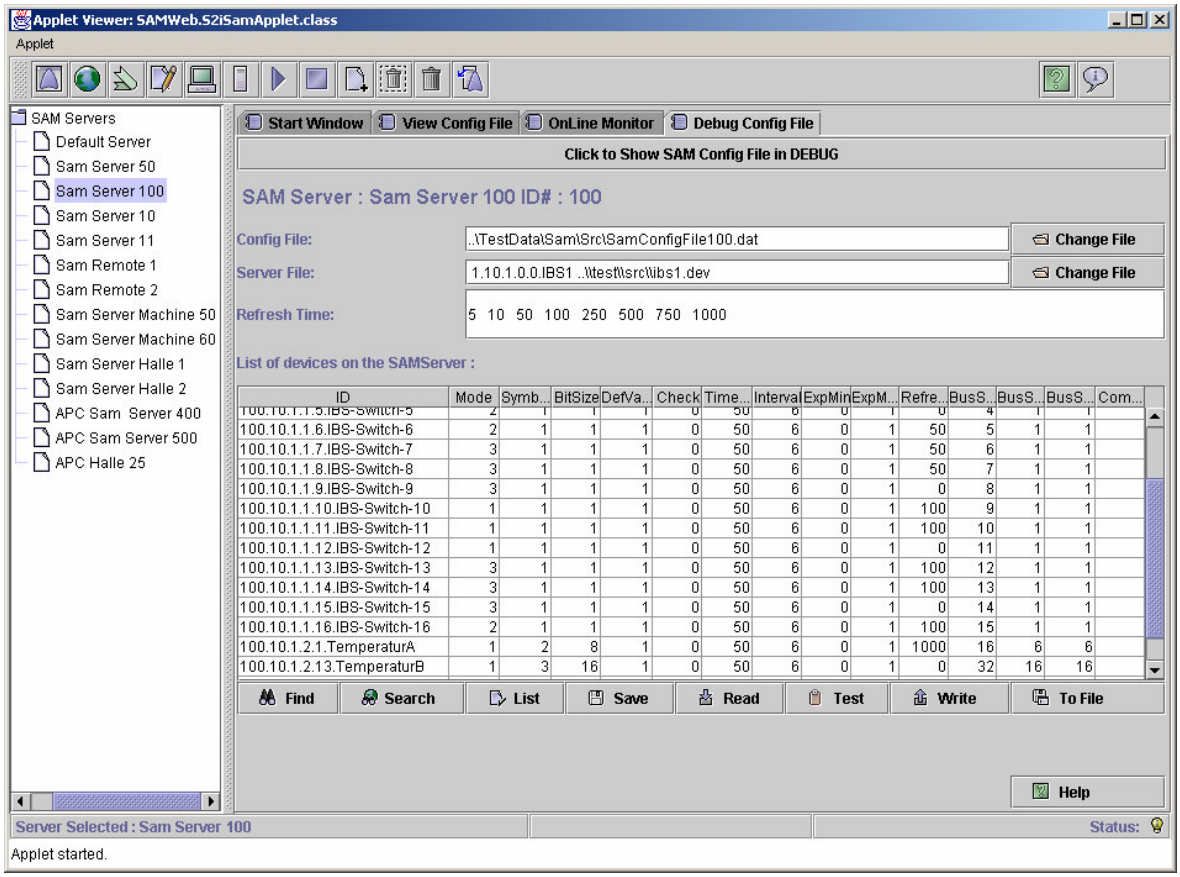


Fig. 5.18: Resultado final do SAMWeb

5.2.4.1 FRAGMENTO DE CÓDIGO C/C++

Segue abaixo um fragmento do código em C/C++ utilizado para o desenvolvimento da biblioteca que permite a comunicação entre o SAMWeb e o SAM.

```
//!-----------------------------------------------------------------------
//! NATIVE FUNCTIONS
int const S2iSamWebAPI::InsertServer( int const a_nSamID, string const & a_sComment )
{
        //Calling the insert server function in the MtqSamClient class
        int t_nStatus = MtqSAMClient::InsertServer( a_nSamID, a_sComment );

        return t_nStatus;

} //InsertServer()
```

Como observa-se no fragmento acima, todo o formato dos comentários e também a nomenclatura segue as normas ditadas pelo grupo S2i e ferramentas usadas pelo grupo.

5.2.4.2 FRAGMENTO DE CÓDIGO JAVA

Segue abaixo um fragmento do código em Java utilizado para o desenvolvimento da interface e todas as funcionalidades do SAMWeb.

```
 //---------------------------------------------------------------------    //Inline
 Class Methods Block
//-----------------------------------------------------------------------
//It calls the command to add a new server in the web config. tool
//    a_nSamID is the sam identifier
//    a_sComment identifies the application
public int insertServer( int a_nSamID, String a_sComment ) {
  int t_nStatus = jInsertServer( a_nSamID, a_sComment );
  System.out.println( "Function return : " + t_nStatus );
  return t_nStatus;
} //insertServer()
```

O fragmento acima também segue todas as normas de documentação e nomenclatura do grupo S2i e das ferramentas pelo grupo utilizadas. Outro objetivo aqui é ressaltar o empenho no desenvolvimento de sistemas totalmente modulares, facilitando desta maneira o reuso dos módulos tanto desenvolvidos em C++ como em Java.

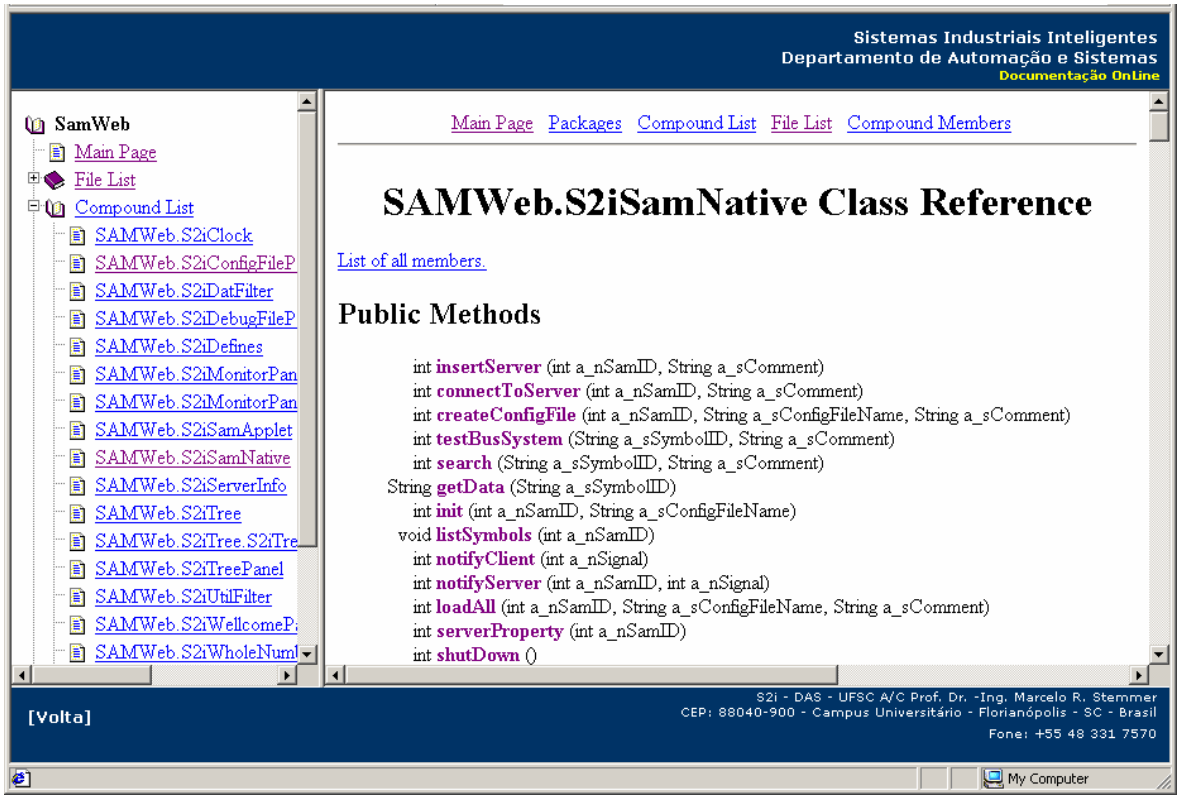


Fig. 5.19: Fragmento da Documentação em Doxygen do SAMWeb

5.2.5 TESTES

Os testes foram realizados durante o desenvolvimento dos módulos do SAMWeb. Por exemplo, durante o desenvolvimento do módulo de monitoração, em primeiro lugar construí-se um *applet* que emulava a monitoração dos valores da memória do computador com um determinado ciclo de *refresh*. Outro exemplo foi o desenvolvimento do módulo para leitura e escrita nos arquivos de configuração do SAM, que foi construído primeiro um *applet* para leitura e escrita de arquivos com determinadas características.

Uma segunda sistemática de testes ocorreu durante a integração dos diversos componentes que compõem o SAMWeb, onde a cada integração o sistema era inteiramente debugado e testado.

A terceira etapa de testes aconteceu com o sistema já integrado. Nesta etapa, vários servidores SAM eram inicializados e o SAMWeb realizava as tarefas para que foi desenvolvido. Estes testes foram exaustivos com o objetivo de atestar a confiabilidade da informações e garantir a integridade do sistema. Também estes testes contribuíram para esclarecer e solucionar alguns problemas que foram encontrados no SAM. Assim, além de testes para garantir o perfeito funcionamento do SAMWeb estes serviram para testar também o módulo de Sensores/Atuadores (SAM) do projeto da Célula Autônoma de Produção (APC).

A metodologia para os testes consistia em:

- Testar cada um dos componentes individualmente através de *demos*.
- Testar a integração sucessiva de componentes no sistema.
- Testar exaustivamente o sistema completo nos mais diversificados ambientes.

A fase de testes não foi completamente finalizada por problemas com a parte da integração do hardware da rede de Sensores/Atuadores. Com isto, todos os testes feitos com todos os sistemas da rede de Sensores/Atuadores foram feitos baseados em simulações e emulações das bibliotecas e aplicativos do SAM. Em setembro de 2002 o projeto APC deverá ser apresentado para as empresas financiadoras do projeto, entre elas, Volkswagen, Bosch e ABB. Os testes no sistema real serão feitos assim que possível.

5.2.6 DOCUMENTAÇÃO

Nesta etapa são armazenadas todas as informações relevantes a como o sistema foi desenvolvido.

Com a documentação do projeto é possível acabar com dúvidas que possam surgir na hora de instalar, configurar ou até mesmo utilizar o sistema.

Além disso, caso mudanças necessitem ser feitas por outros colaboradores, este tópico serve como uma base para solução de problemas que possam acontecer durante o desenvolvimento ou *update* do sistema.

No SAMWeb uma base de informações e *backups* do sistema foi gerada e esta armazenada em forma de *backups* (CD graváveis), na base de informações do grupo S2i e também na página do projeto no *site* do S2i e do Projeto APC (http://sfb368.rwth-aachen.de). As informações precisam estar disponíveis da melhor forma possível.

5.2.7 MANUTENÇÃO

A fase de manutenção não começa somente quando o sistema é entregue ao cliente, mas já nas fazes de desenvolvimento a manutenção de certos componentes e dispositivos já está presente. Com o SAMWeb não pode ser diferente e o sistema, embora em fase final de conclusão e testes, já passa por etapas de manutenção. Mas é uma manutenção mais preventiva do que corretiva, em função dos testes realizados. Certas peculiaridades somente foram constatadas na etapa de testes, e agora o SAMWeb passa por uma fase de manutenção preventiva contra certas características operacionais. A manutenção corretiva ocorre tanto no desenvolvimento quanto no cliente com o sistema já implantado e faz parte de todo o sistema de hardware/software desenvolvido.

6 RESULTADOS EXPERIMENTAIS EM SIMULAÇÕES

6.1 SIMULAÇÕES

O funcionamento do sistema foi simulado em computadores do instituto WZL (Aachen, Alemanha) e também nos computadores do Departamento de Automação e Sistemas (DAS). O método de simulação consiste na instalação e execução da biblioteca da rede de Sensores/Atuadores (SAM) em uma quantidade "X" de máquinas. Com o SAM instalado em cada máquina é possível emular para que cada computador comporte-se como um servidor do SAM (pode-se também simular mais instâncias do SAM em cada máquina). Assim, torna-se viável a execução do SAMWeb e pode-se simular sua conexão com os servidores e clientes SAM.

Cabe aqui ressaltar que o sistema foi somente simulado em razão de um atraso no desenvolvimento do hardware que servirá como base para testes reais em chão-de-fábrica, não somente do SAMWeb como toda a rede SAM e dos outros componentes da Célula Autônoma de Produção (APC). Estima-se que até o final de 2002 o hardware com as placas e sistemas *fieldbus* e também todo os outros componentes de hardware do APC estejam em perfeito funcionamento. Porque até o momento apenas algumas características do sistema puderam ser apresentadas no sistema real. As funcionalidades básicas de monitorar, ler e escrever dados, na rede de sensores e atuadores funcionaram perfeitamente. Funcionalidades como: procurar por um determinado dispositivo, ainda estão sendo melhoradas.

Os próximos sub-itens apresentam passo-a-passo como se pode realizar as simulações com o SAMWeb.

*6.1.1 INSTALAÇÃO DA BIBLIOTECA **SAM***

Como foi dito anteriormente, o primeiro passo consiste na instalação da biblioteca do SAM na máquina, ou nas máquinas, onde serão realizados os testes simulados, cabe aqui ressaltar que o módulo SAM estará presente somente na máquina-ferramenta onde serão realizadas as medições. O SAMWeb é um cliente que realiza acesso remoto as bibliotecas do SAM. No caso de simulação, pode-se instalar o módulo SAM também na máquina onde será executado o SAMWeb, por motivos de simplicidade. A instalação da biblioteca acontece mediante a compilação dos seus códigos fontes e geração dos arquivos .dll (*Dynamic Link Library*) e .lib (*Library*) que possibilitem a instanciação do SAM.

Com a biblioteca da rede SAM instalada na máquina é possível simular o funcionamento da rede de Sensores/Atuadores com seus clientes e servidores.

6.1.2 INSTALAÇÃO DO *SAMWEB*

Com a biblioteca SAM presente no computador local ou em um conjunto de computadores pode-se passar para o passo de instalação ou *download* do sistema para configuração, monitoração e testes da rede de Sensores/Atuadores (SAMWeb).

O SAMWeb pode acessar servidores SAM que estão presentes na máquina em que o SAMWeb está sendo executado ou em máquinas remotas. O diagrama a seguir exemplifica a afirmação anterior [Fig. 6.1].

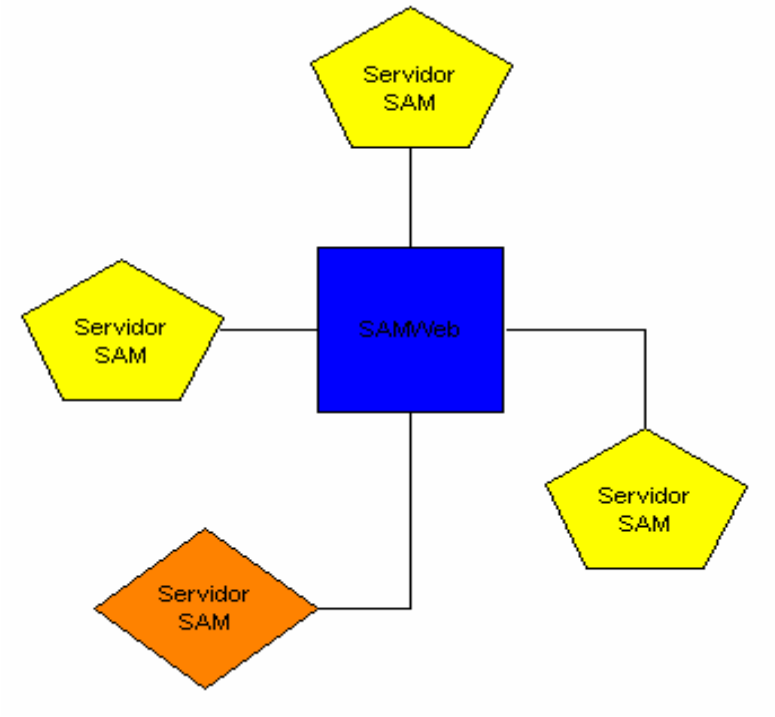


Fig. 6.1: SAMWeb acessa servidores SAM

Como o SAMWeb é um *applet* ele pode ser executado remotamente em uma máquina disponível na rede ou então se pode "baixá-lo" para uma máquina local e executá-lo localmente. Além do mais, o aplicativo pode ser executado dentro de um *browser* com suporte Java ou então em aplicativos especiais para a execução de *applets*.

6.1.3 *SAMWEB* PASSO-A-PASSO

Este tópico apresenta o funcionamento do SAMWeb passo-a-passo. Em cada um dos parágrafos seguintes serão apresentadas as características técnicas e de funcionamento de cada componente gráfico do sistema para configuração, monitoração e testes da rede de Sensores/Atuadores.

Após o usuário inicializar o aplicativo SAMWeb através de uma *Web Browser* como Internet Explorer ou Netscape Navigator, ou ainda através de uma visualizador de aplicativos Java (*Applet Viewer*) a tela de entrada do aplicativo é mostrada [Fig. 6.2].

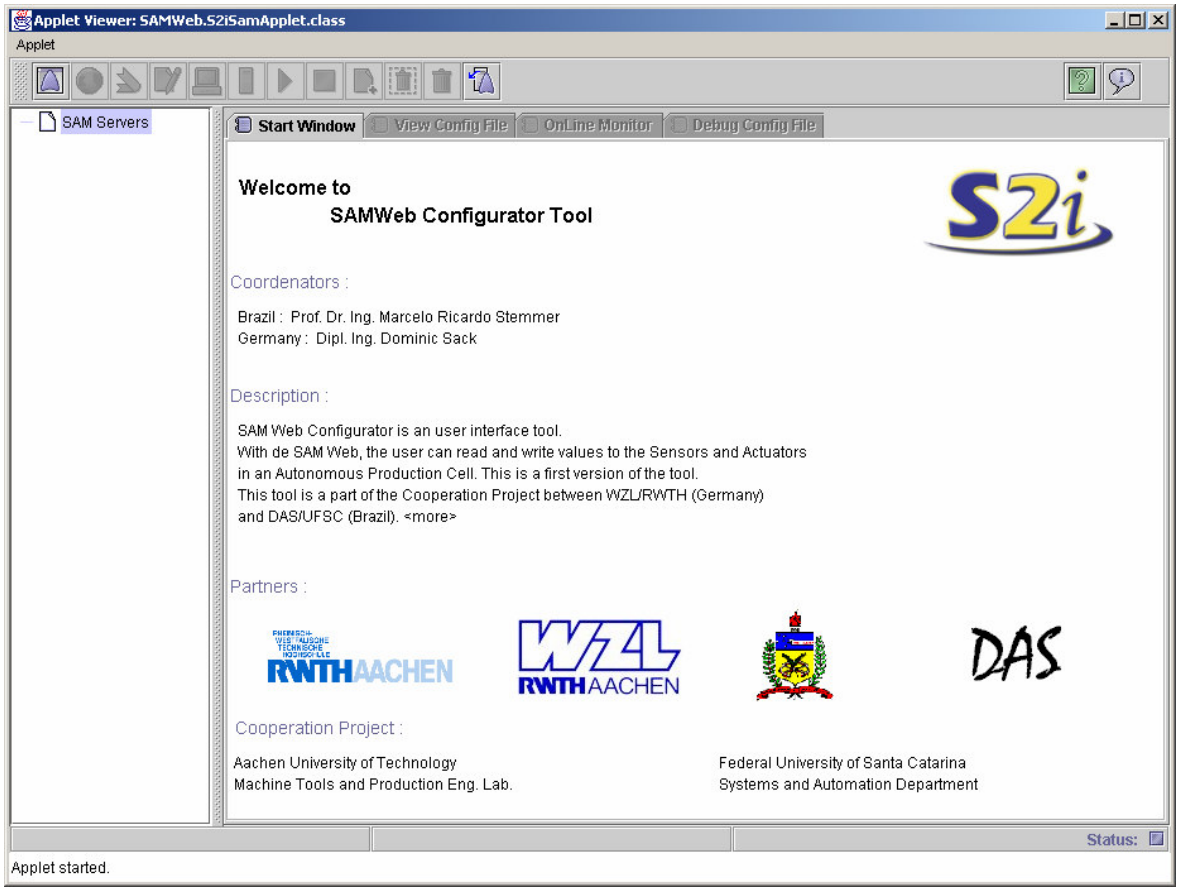


Fig. 6.2: Tela inicial do SAMWeb

A tela inicial do SAMWeb é composta por algumas regiões que ajudarão na execução das tarefas e visualização dos resultados. No topo tem-se a barra de ferramentas (*toolbox*) onde se encontram os principais comandos necessários para a execução e conexão do aplicativo com os demais componentes do Módulo de Sensores/Atuadores (SAM). A lateral direita é formada pela árvore de navegação do SAMWeb. Nesta árvore são listados todos os servidores (no futuro, os outros clientes) que estão conectados com o SAMWeb. Para acessar informações de cada um dos servidores SAM da árvore, basta clicar sobre o item desejado. Na parte inferior da janela encontra-se a *status bar* onde aparecem informações sobre o resultado de cada uma das funções executadas pelo SAMWeb (por exemplo, se a leitura do sensor de temperatura foi um sucesso ou não, retornando um erro) e também o estado de cada servidor (se está *OnLine* ou *OffLine*).

Para iniciar o uso do SAMWeb em um primeiro instante é necessário seguir uma seqüência de passos para o estabelecimento da conexão com a rede de Sensores/Atuadores (SAM). Na seqüência abaixo é possível observar o estabelecimento da primeira conexão com o **SAM** *Server*.

Após a conexão estabelecida, na árvore de servidores SAM irá surgir um novo elemento que é um servidor *default* que existe no sistema. Ele é um servidor como os outros, porém ele possui uma função a mais, é responsável pela criação do canal de comunicação da rede de Sensores/Atuadores (SAM). Como os outros ele apresenta a possibilidade de visualização, monitoração e configuração dos dados dos Sensores/Atuadores presentes neste servidor.

Agora, para a inserção de outro servidor SAM no SAMWeb é necessário clicar no botão de adição de um novo Servidor SAM e uma nova janela irá aparecer. Nesta janela, o usuário deve entrar com os dados de identificação do servidor e buscar pelo arquivo de configuração do mesmo, a figura 6.3 apresenta a janela de configuração do novo servidor SAM.

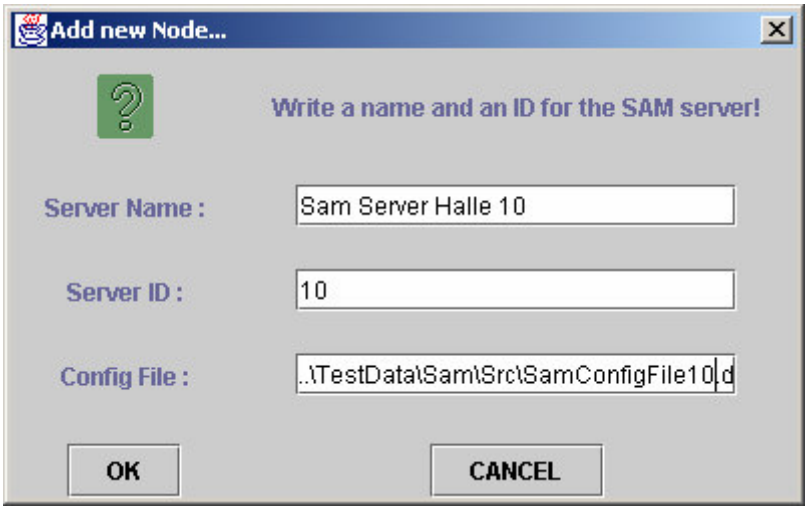


Fig. 6.3: Adicionando um novo Servidor ao SAMWeb

Após o servidor inserido, o mesmo aparece na árvore de servidores do SAMWeb. Para acessá-lo basta clicar sobre o item e o usuário terá acesso às janelas de visualização, monitoração e configuração dos Sensores/Atuadores do servidor SAM. Neste momento o cliente (SAMWeb) e o servidor inserido já apresentam um canal de comunicação criado, especificado e funcionando. A partir desse momento passa-se para as *tabs* de navegação na janela principal do SAMWeb.

A primeira *tab, Start Window*, apresenta uma janela com informações sobre os orientadores do SAMWeb, uma breve descrição dos objetivos da ferramenta, os parceiros no desenvolvimento da aplicação e uma citação do projeto de cooperação entre o DAS/UFSC e o WZL/RWTH-Aachen.

A segunda *tab,View Config File*, é a janela que apresentará ao usuário os dados do arquivo de configuração do servidor. A janela apresenta os dados referentes a localização do arquivo de configuração do servidor, a localização do arquivo que identifica o servidor, e a tabela de *refresh*

do servidor (tabela de tempo que os dados são atualizados no arquivo de configuração pelo servidor SAM). Em seguida pode-se visualizar a lista de dispositivos presentes neste servidor SAM e seus valores. Abaixo da janela com os dados dos dispositivos tem-se os comandos que podem ser realizados pelo operador nesta janela de configuração. Estes comandos servem basicamente para localizar um dado na lista de dispositivos, encontrar um dispositivo dentro do servidor, listar os dispositivos do servidor e também ler os valores de um determinado dispositivo.

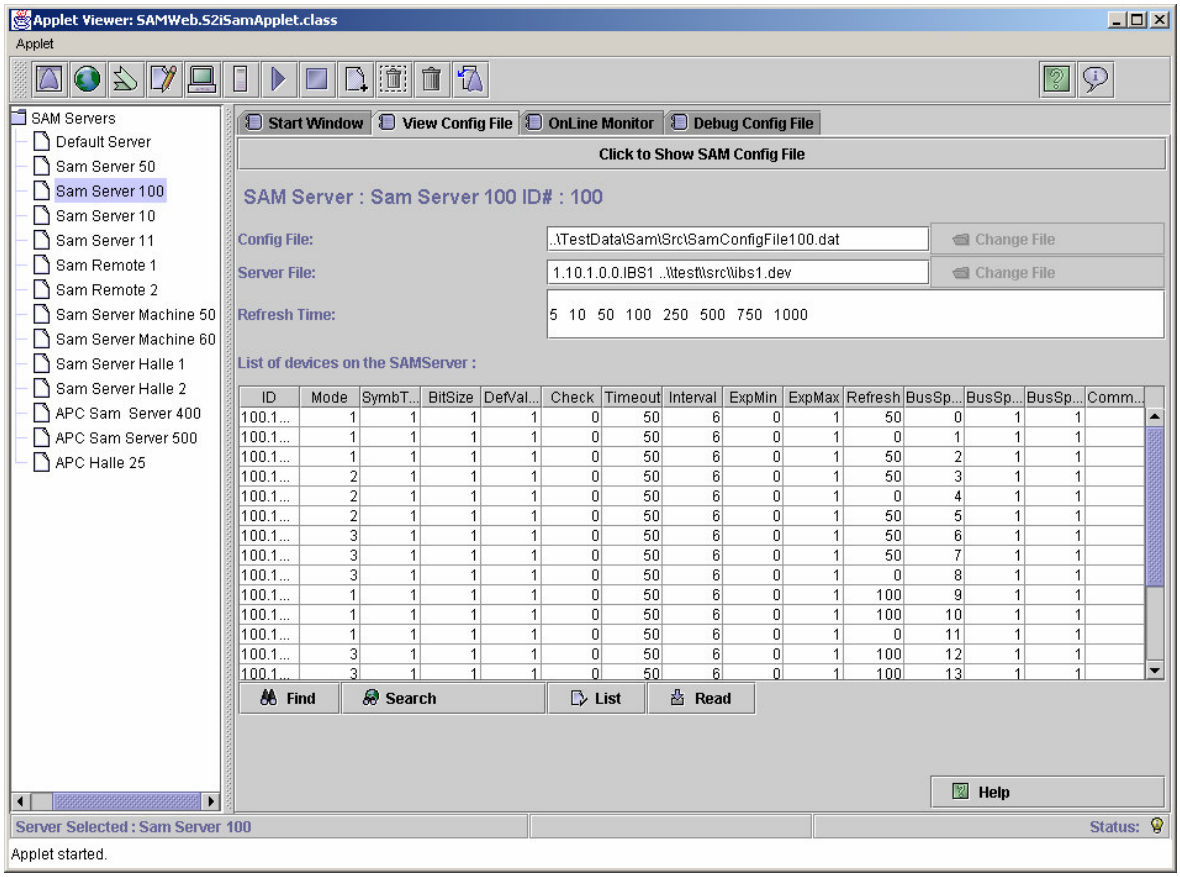


Fig. 6.4: *Tab View Config File* do SAMWeb

A terceira *tab*, *OnLine Monitor*, é utilizada para o SAMWeb fazer a monitoração em tempo real de um determinado dispositivo escolhido pelo usuário. Nesta janela, o usuário possui a lista de dispositivos presentes em um determinado servidor, selecionado na árvore do SAMWeb. Nesta lista, o usuário seleciona o dispositivo que ele deseja monitorar. Com o dispositivo escolhido um gráfico é gerado pelo monitor com os valores do dispositivo a cada determinado intervalo de tempo, definido pelo usuário.

A qualquer momento, o usuário pode selecionar um novo dispositivo na lista e reiniciar o processo de monitoração para este novo dispositivo. A vantagem do monitor é que ele permite que

o usuário fique sabendo de oscilações indesejáveis nos valores dos Sensores/Atuadores do servidor SAM.

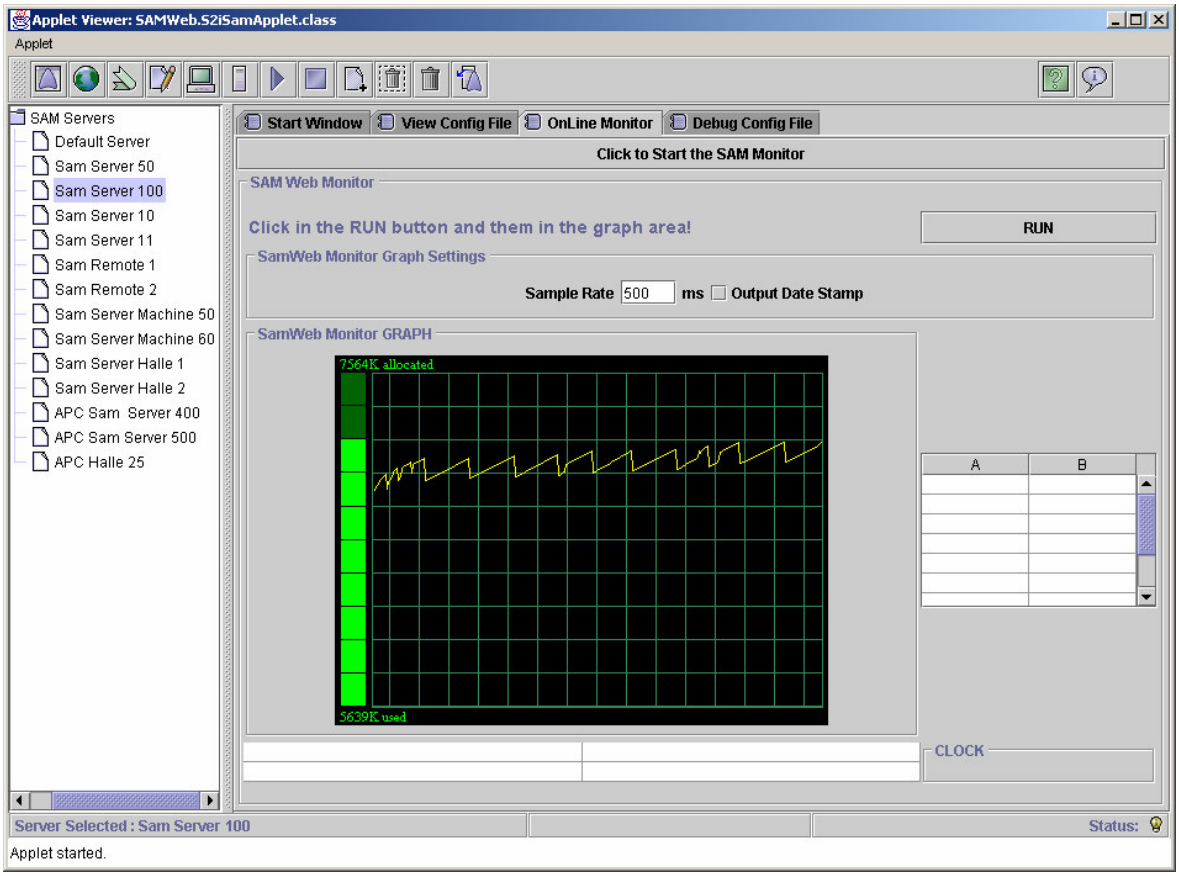


Fig. 6.5: *Tab Monitor* do SAMWeb

A quarta e última *tab, Debug Config File*, é a janela que apresenta o maior grau de complexidade do SAMWeb. Nesta janela, o usuário possui acesso a todas os comandos que o SAM permite, para edição, alteração e monitoração dos valores dos Sensores/Atuadores. A aparência inicial da janela de *Debug* do SAMWeb é muito similar à primeira janela de configuração do SAMWeb, View *Config File*.

Nesta janela, além da possibilidade de visualização dos arquivos de configuração da rede de sensores/atuadores e do arquivo de definição do tipo de rede empregada, o usuário tem possibilidade de modificá-los. Em seguida é apresentada a tabela com os tempos de *refresh* dos dados da rede. A lista com os dispositivos pertencentes ao SAM *Server* selecionado segue logo abaixo. Em seguida existe uma série de botões onde o usuário pode realizar uma série de comandos para controle e monitoração da rede de Sensores/Atuadores. A figura 6.6 mostra a janela descrita e suas partes.

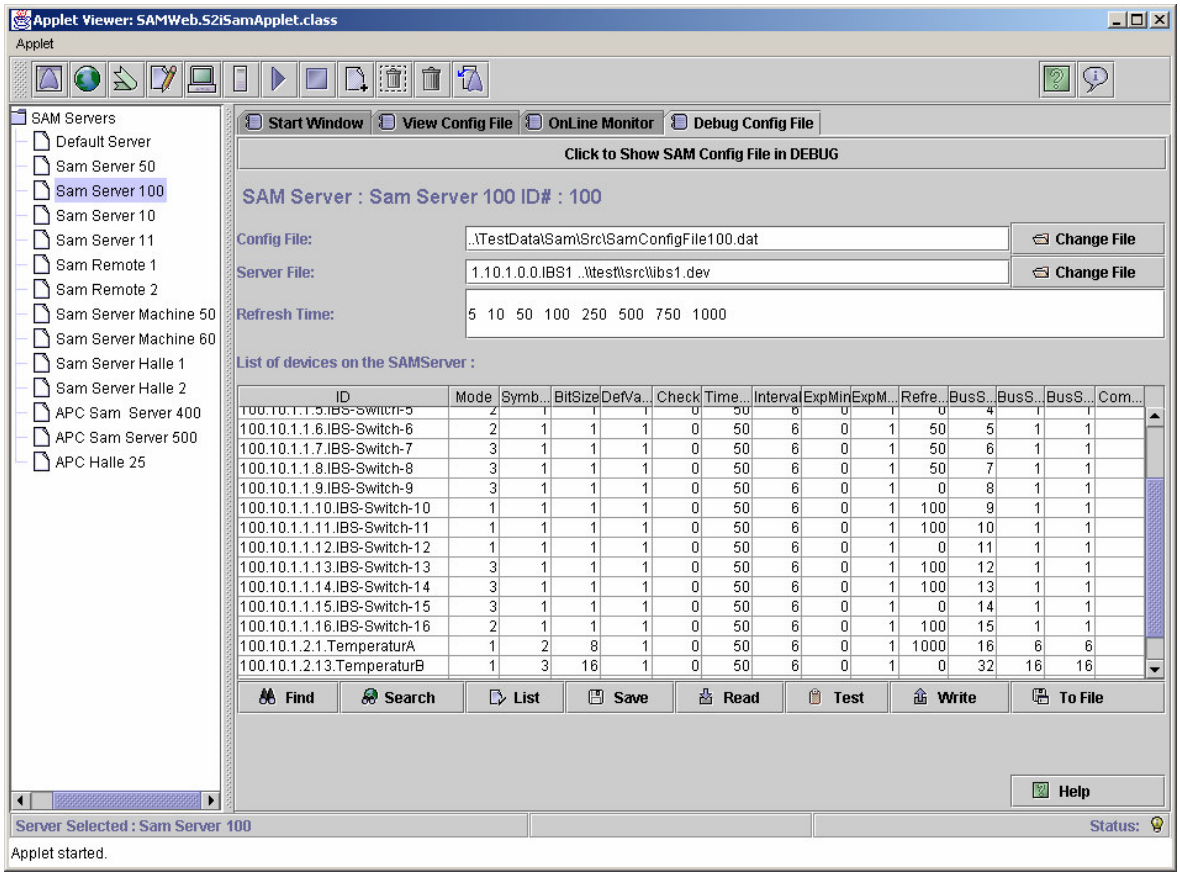


Fig. 6.6: *Tab Debug Config File* do SAMWeb

Cada um dos botões da barra de comando da janela possui uma característica específica e retorna um resultado que pode ser positivo ou negativo, se algo acontecer de errado. Em seguida, cada um dos botões será explicado suscintamente:

Find: Selecionando um elemento na lista de dispositivos o comando *Find* irá percorrer a rede de sensores/atuadores em busca do dispositivo, se o resultado for positivo, o dispositivo é selecionado e uma mensagem de sucesso aparecerá na barra de status, caso contrário, uma mensagem de erro será apresentada ao usuário.

Search: Procura por um dispositivo no servidor SAM que esta sendo analisado, o resultado aparece como no caso anterior.

List: Lista novamente todos os dispositivos (sensores/atuadores) que existem no servidor SAM selecionado. Caso o comando obtenha sucesso a lista dos dispositivos é atualizada, caso contrário, uma mensagem de erro aparece na barra de status.

Save: Salva os valores correntes no servidor SAM. O resultado aparece na barra de status, com uma mensagem de afirmativo ou de erro.

Read: Faz a leitura novamente das informações do dispositivo selecionado na lista de dispositivos do servidor SAM que está sendo analisado.

Test: Testa se um dispositivo está OK. Caso esteja tudo em ordem, uma mensagem positiva aparece a barra de status, caso contrário, uma mensagem de erro aparece na barra de status.

Write: Escreve os valores e configurações do dispositivo selecionado na lista no servidor SAM. As mensagens de sucesso ou fracasso também aparecem na barra de status.

To File: Salva todas as informações dos dispositivos para um arquivo de configuração escolhido pelo usuário através de uma janela de salvar como padrão do sistema operacional.

Estas são as operações permitidas pela arquitetura do SAM. Através destas operações básicas o operador pode acessar e configurar qualquer um dos dispositivos que fazem parte da rede de Sensores/Atuadores. Independentemente do local onde o usuário se encontra ele sempre terá acesso as informações essenciais e vitais para o bom funcionamento do SAM.

6.2 RESULTADOS

As simulações foram realizadas no Departamento de Automação e Sistemas (DAS) e no Instituto de Máquinas Ferramenta e Engenharia de Produção (*Werkzeugmaschinenlabor* – WZL) em Aachen.

Os equipamentos utilizados para a simulação foram microcomputadores PC indo desde processadores INTEL e AMD 400 MHz com 64 Mb RAM até 1 GHz com 256 Mb RAM.

A metodologia empregada para as simulações e busca dos resultados baseou-se em testes exaustivos de desempenho e confiabilidade dos dados. Primeiramente atuou-se de maneira a garantir a integridade dos dados e a validação do estabelecimento da comunicação entre os processos. A seguir passou-se para testes exaustivos na alteração destes dados sempre utilizando apenas um servidor SAM. Validado o algoritmo de comunicação entre o processo e o sistema de leitura e escrita dos dados, passou-se para testes simulados em laboratório com uma rede de computadores.

Com o auxílio de várias estações de trabalho, instalou-se os servidores SAM em cada uma possibilitando que as máquinas pudessem emular o funcionamento do SAM. Assim, é possível instanciar um ou mais servidores SAM em cada uma das estações.

No mês de setembro de 2002 o SAM foi instalado no Hall de Máquinas do WZL e isto possibilitou a demonstração do sistema SAMWeb em ambiente industrial, fazendo parte de todo o projeto APZ. A demonstração foi realizada na semana de avaliação dos resultados do projeto SFB-368 (APZ) de qual o SAM e o SAMWeb são partes integrantes. O resultado da avaliação por parte dos dirigentes do SFB, Ministério de Ciência e Tecnologia Alemão foi EXCELENTE. O

SAMWeb, fazendo parte deste projeto também foi avaliado e recebeu também o conceito EXCELENTE. O conjunto do sucesso de todos os trabalhos envolvidos no SFB368 possibilitou um novo financiamento e continuação das pesquisas por mais 3 anos em um total de € 1,5 Milhões para todo o projeto SFB-368.

Tanto os testes simulados em laboratório como os testes no ambiente industrial foram de grande valia. Com estes testes foi possível avaliar o desempenho do SAMWeb e das outras ferramentas e também buscar por algumas melhorias, tanto de caráter técnico (processo de validação da comunicação) como de caráter ergonômico, adaptando a interface para o operador.

As simulações de laboratório serviram para validar a ferramenta de configuração, monitoração e testes de Sensores/Atuadores com várias conexões, com diferentes servidores, em diversos locais. Os testes provaram a eficiência do processo de controle remoto das informações.

Os testes de desempenho e exaustão no *Machine Halle* do WZL serviram para validar a funcionalidade o sistema e a integração dele com todos os outros componentes do projeto SFB-368. Também nestes testes, o SAMWeb atendeu as expectativas tanto dos desenvolvedores e pesquisadores como dos avaliadores do SFB.

Os resultados desta dissertação foram alcançados, como se procura explicar melhor no próximo capítulo, provando que é possível desenvolver uma ferramenta para configuração, monitoração e teste dos Sensores/Atuadores em uma Célula Autônoma de Produção através de uma interface estilo WEB, remotamente.

7 CONCLUSÕES E PERSPECTIVAS

A partir dos resultados obtidos (ver item 6.2) conclui-se que o SAMWeb atendeu a todos os requisitos de modularidade, portabilidade, independência e autonomia exigidos pelos requisitos do projeto da Célula Autônoma de Manufatura.

O sistema também se mostra mais eficaz que o sistema anterior para testes dos Sensores/Atuadores que era baseado em uma janela DOS e que exigia grande conhecimento do operador para acessar os dispositivos. Através de uma interface gráfica prática e objetiva, o acesso à visualização das informações ficou bem mais ágil, simples e rápido pelo operador.

Outra vantagem que se merece destacar é a possibilidade de controlar uma rede com vários servidores SAM presentes em diversos equipamentos diferentes em locais diversos. E ainda mais que este controle pode ser realizado de uma estação remota que não esteja presente no chão-de-fábrica, por exemplo.

Em comparação com as diversas técnicas que poderiam ser empregadas para o desenvolvimento desta ferramenta para monitoração, configuração e testes dos Sensores/Atuadores (ver Capítulo 3), a técnica utilizando JNI (*Java Native Interface*) demonstrou ser a escolha correta, uma vez que facilitou a integração entre o servidor SAM e uma interface de visualização voltada para a WEB. Além do mais, pelo conhecimento de toda a estrutura SAM e também pelo conhecimento de que Java é a linguagem mais adequada para o desenvolvimento de ferramentas gráficas e para a Internet, a integração entre Java e C/C++ através de JNI mostrou-se eficiente, ágil e eficaz.

O requisito de tempo-real do sistema foi atendido. O requisito de tempo-real para o sistema é classificado como *soft*. A monitoração e atuação sobre o sistema não exigem características de tempo real *hard*, assim, a atuação nos dispositivos dos servidores SAM mostrou-se eficiente em todos os casos.

As simulações em laboratório e os testes realizados no Hall de máquinas do instituto WZL (Aachen, Alemanha) apresentaram bons resultados no acesso e monitoração dos dados. Até mesmo utilizando-se vários servidores SAM em locais e equipamentos diferentes (testes de laboratório), o resultado foi satisfatório.

7.1 PONTOS PENDENTES & TRABALHOS FUTUROS

Durante o processo de desenvolvimento da dissertação de mestrado algumas novas funcionalidades foram sendo propostas pelo acadêmico e pela equipe envolvida no projeto APZ.

Uma nova funcionalidade desejável para o SAMWeb foi elaborada, apresentada e aprovada pela equipe do projeto APZ. Esta nova funcionalidade para o SAMWeb é a integração de uma ferramenta para visualização em 3D da máquina onde se encontra a rede de Sensores/Atuadores através de VRML (Virtual Reality Modelling Language). Esta nova ferramenta está em processo de pesquisa e planejamento para ser desenvolvida também no âmbito do projeto de cooperação entre a UFSC-DAS e a RWTH-Aachen-WZL.

A figura 7.1 apresenta um exemplo do que é VRML. Com esta ferramenta será possível visualizar a localização do Sensor/Atuador na máquina. Com isto busca-se auxiliar o usuário na localização física de um dispositivo que deve ser trocado ou então que apresenta um problema, ou ainda somente por consultar a sua posição dentro da máquina.

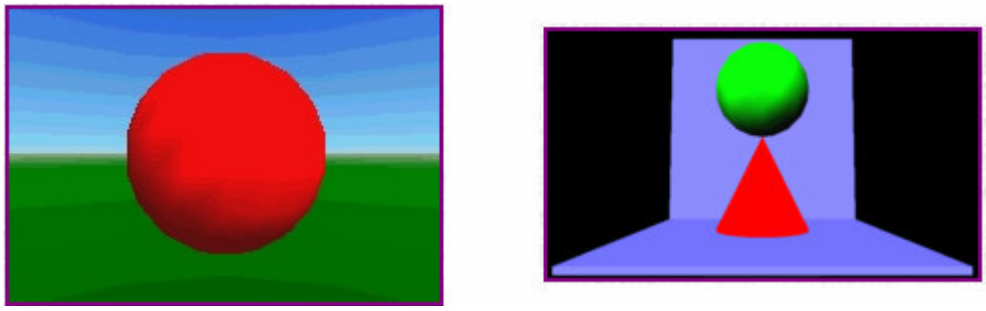


Fig. 7.1: Exemplo do uso de VRML

7.2 ARTIGOS PRODUZIDOS

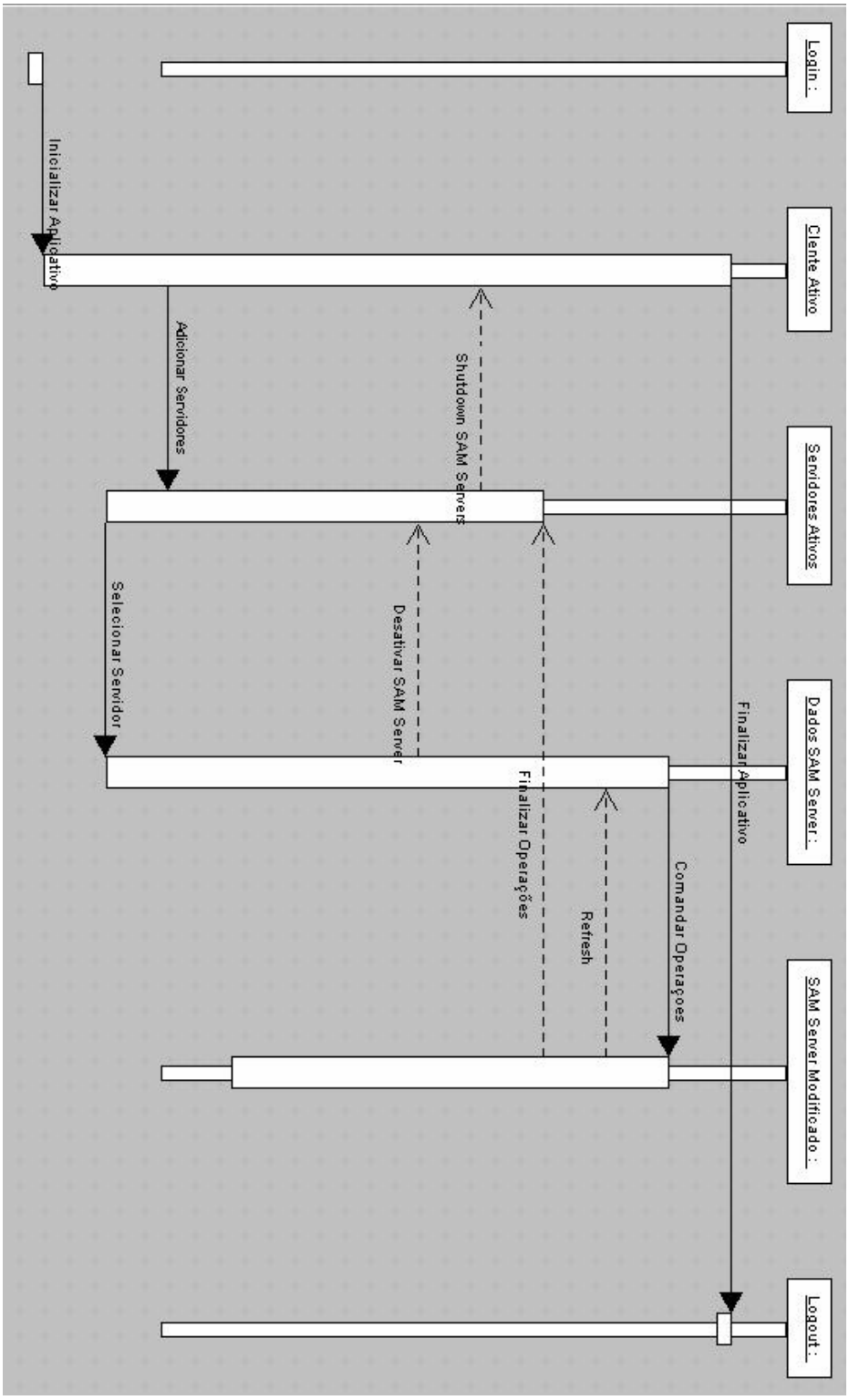
Durante o desenvolvimento desta dissertação o acadêmico participou de alguns trabalhos para a divulgação tanto do seu trabalho como do grupo S2i. A lista seguinte apresenta os artigos[9] produzidos na área da dissertação:

[1] ROLOFF, M.R., STEMMER, M.R. (2001). Sensoriamento Remoto Usando uma Interface Java Native Interface para Aplicação em Sistemas Industriais de Visão. Simpósio Catarinense de Processamento Digital de Imagens, Florianópolis, Brasil (SCPDI 2001).

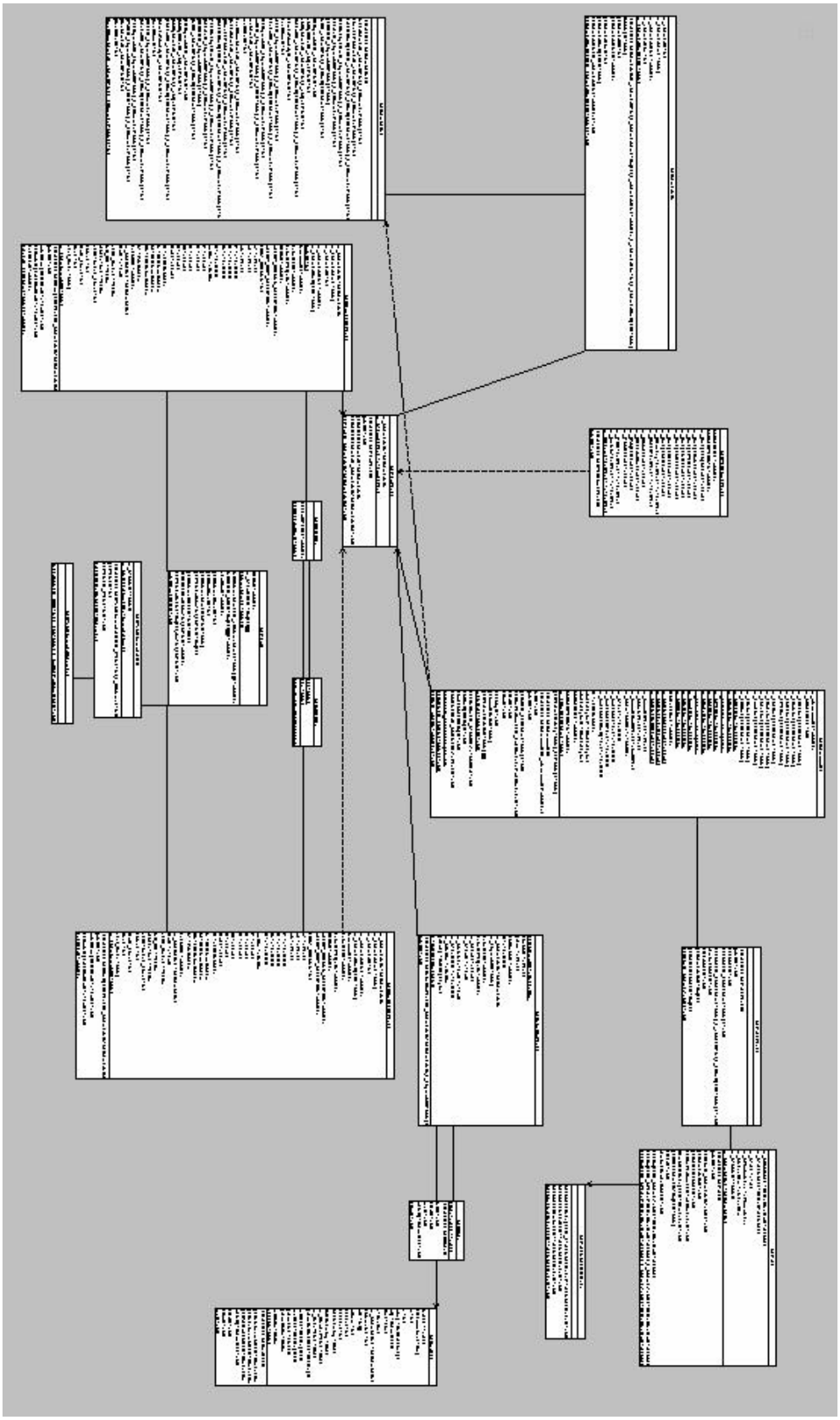
[2] ORTH, A., PFEIFER, T., SACK, D., ROLOFF, M.R., and STEMMER, M. R. (2002). Measuring Flank Tool Wear on Cutting Tools with Machine Vision – A Case Solution, Mechatronics and Machine Vision in Practice, Tailândia (M2VIP'2002).

ANEXO I

Exemplo do Diagrama de Seqüência do SAMWeb

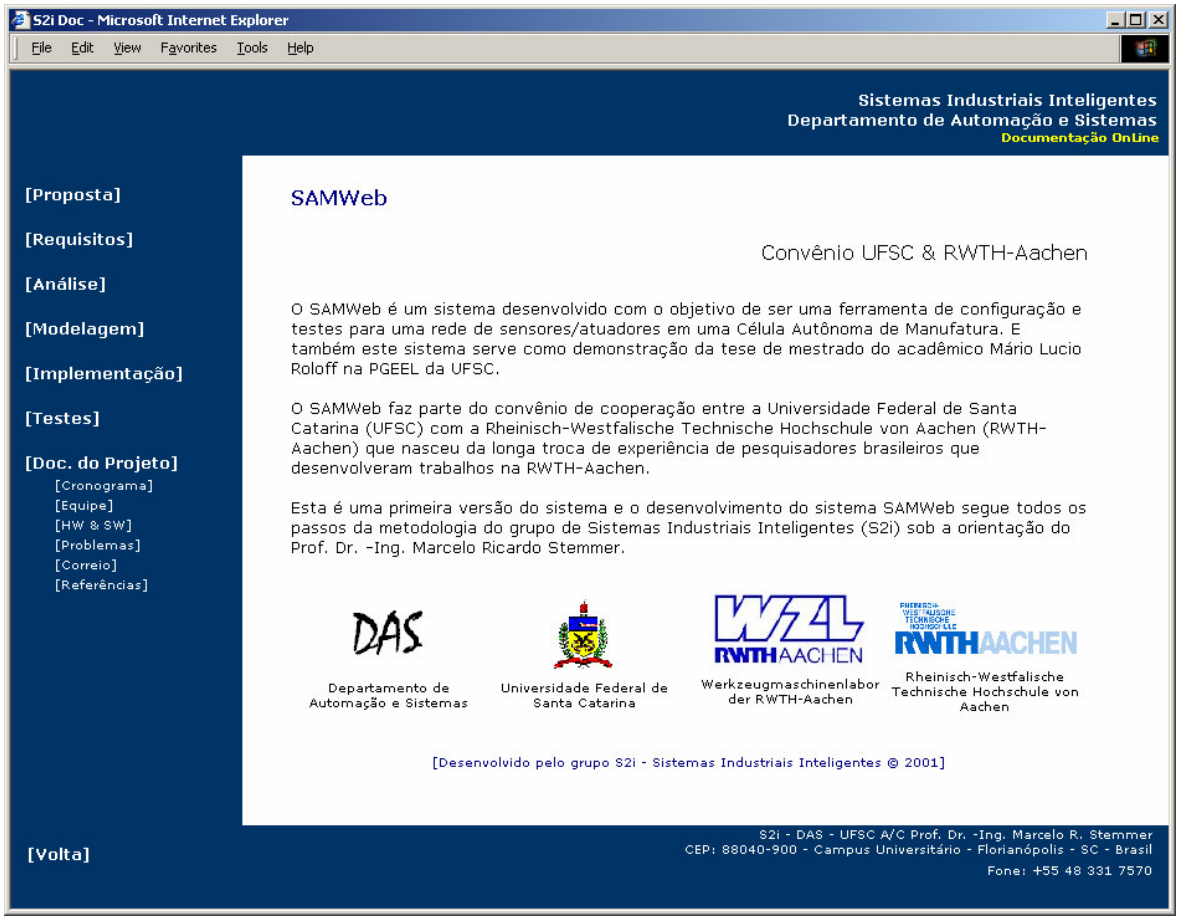


Exemplo do Diagrama de Classes do SAMWebAPI

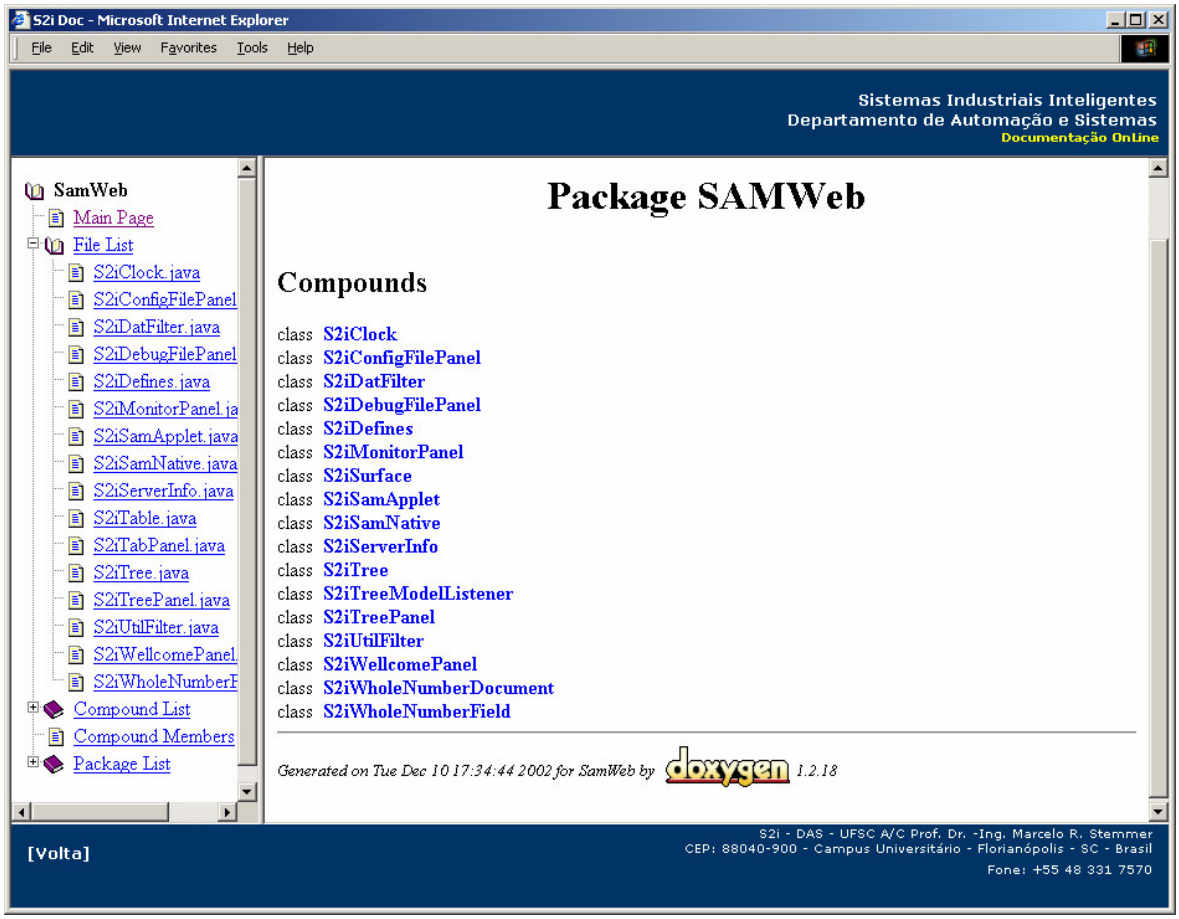


ANEXO II

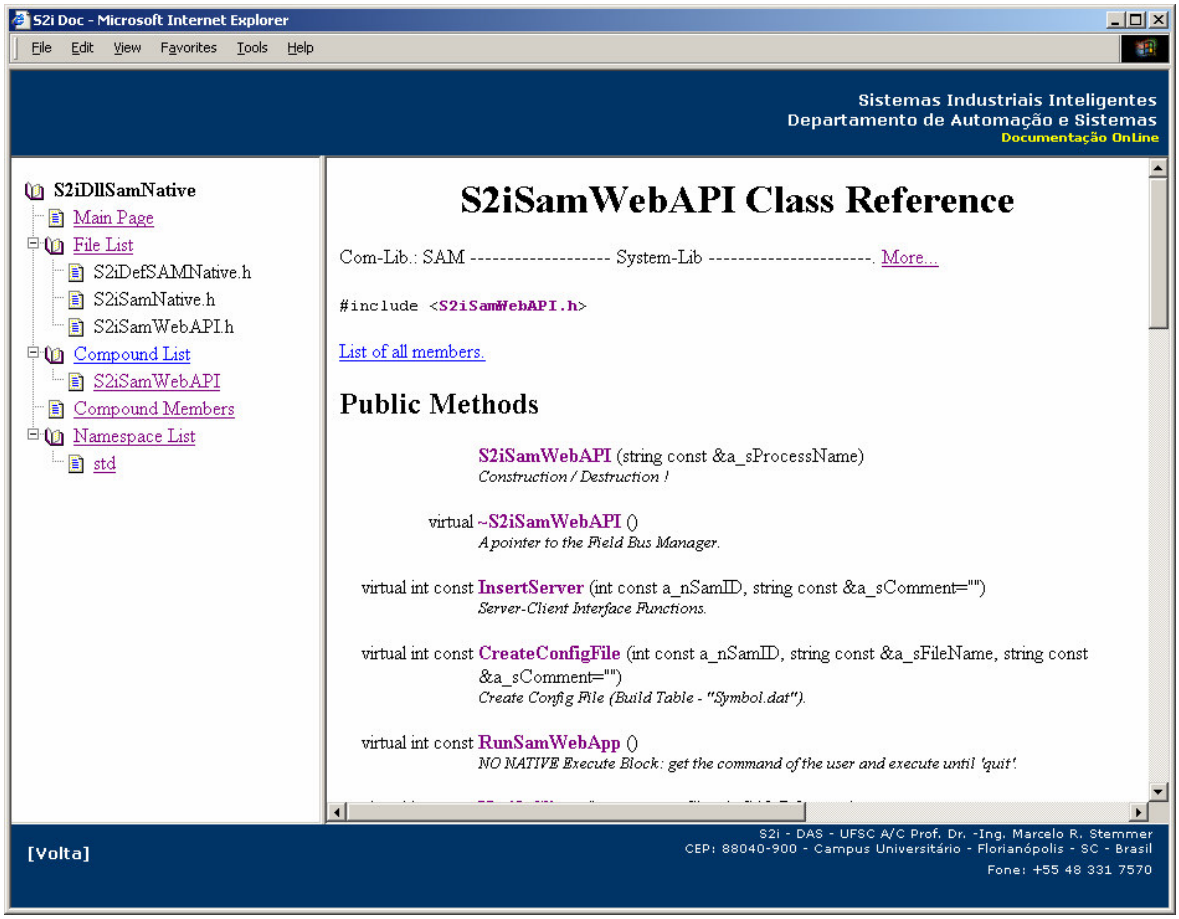
Print Screen da Tela Inicial da Documentação do SAMWeb



Print Screen da Tela Inicial da Documentação Do Código C/C++ do SAMWeb

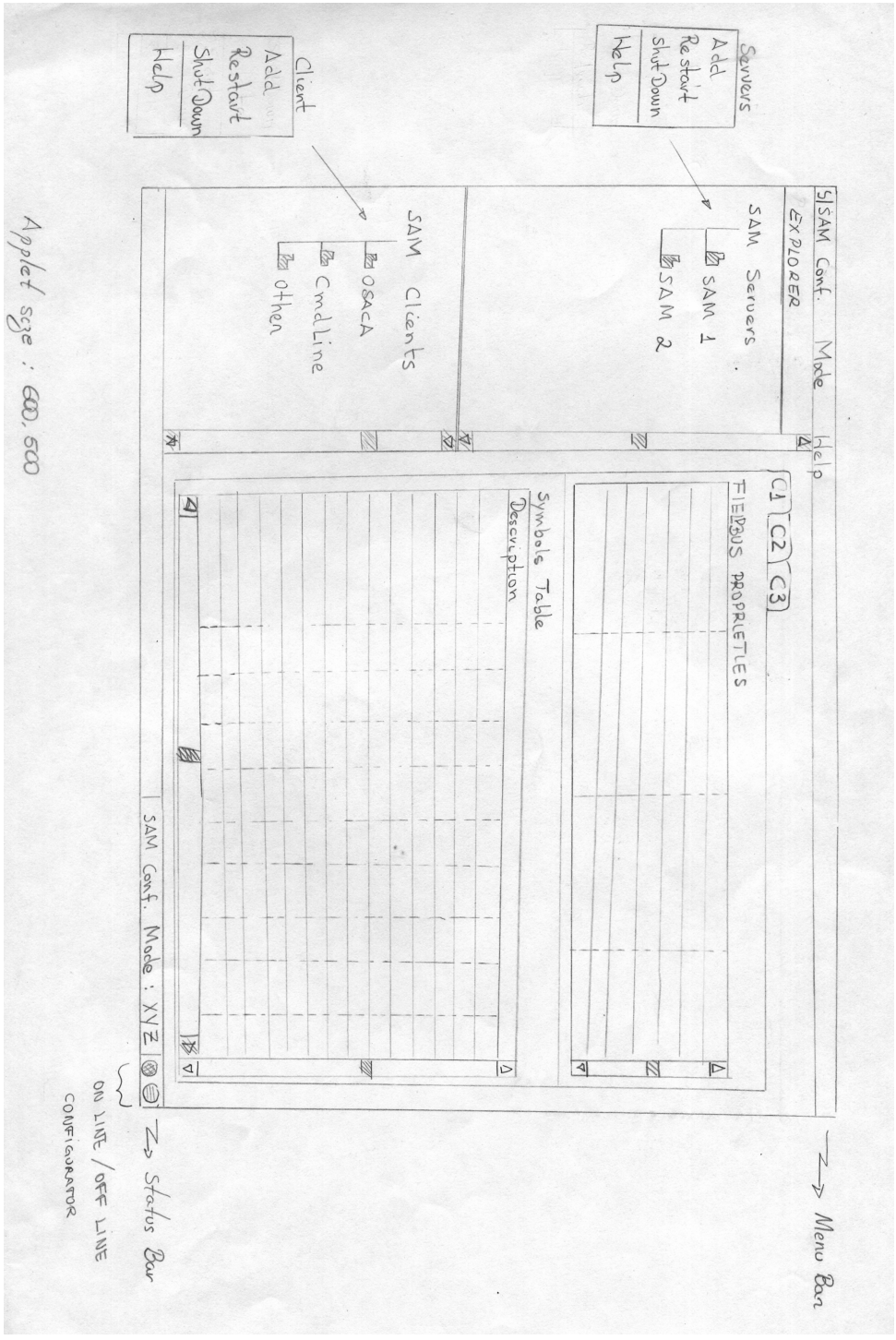


Print Screen da Tela Inicial da Documentação Do Código Java do SAMWeb

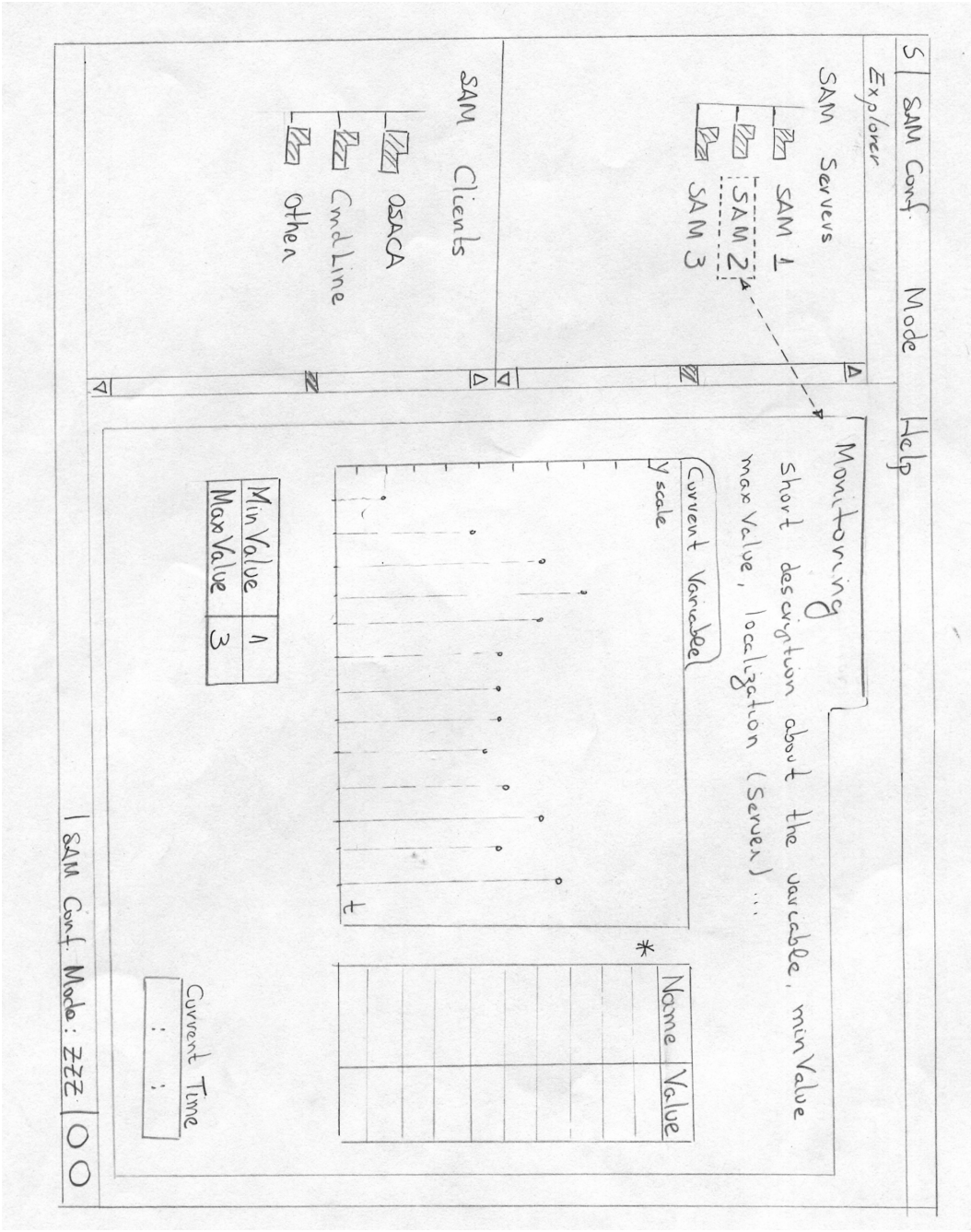


ANEXO III

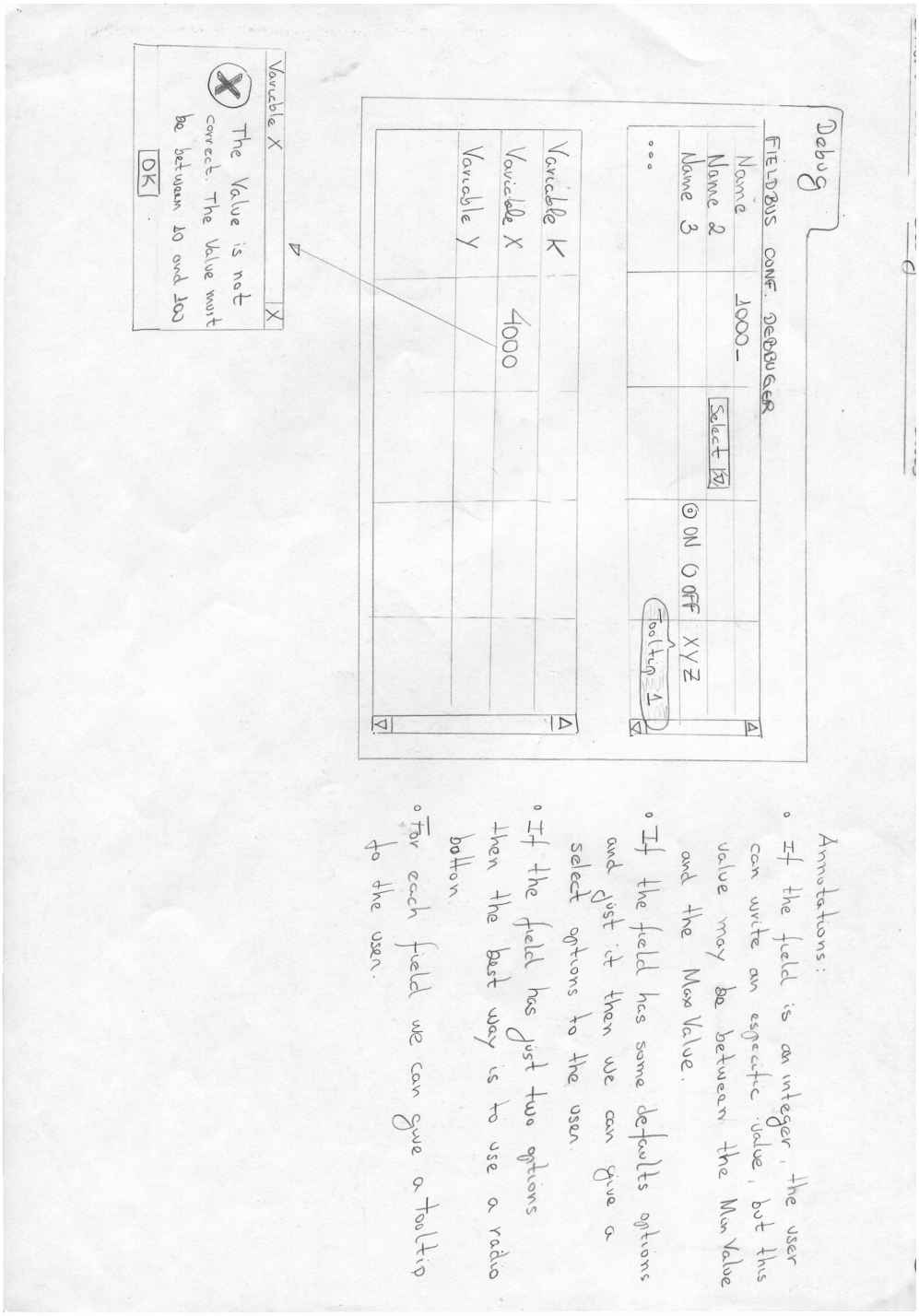
Croquis da Tela da Interface Inicial (Visualização) do SAMWeb



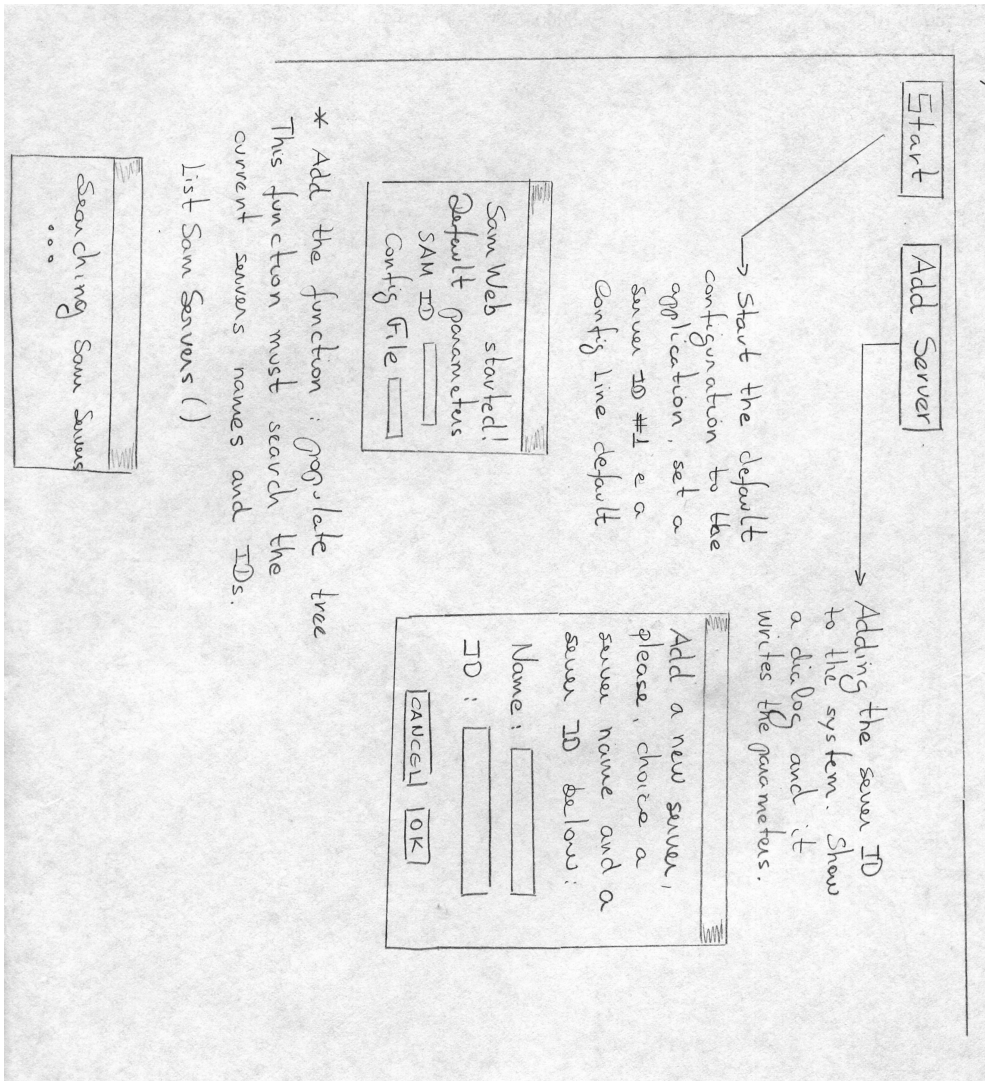
Croquis da Tela de Monitoração



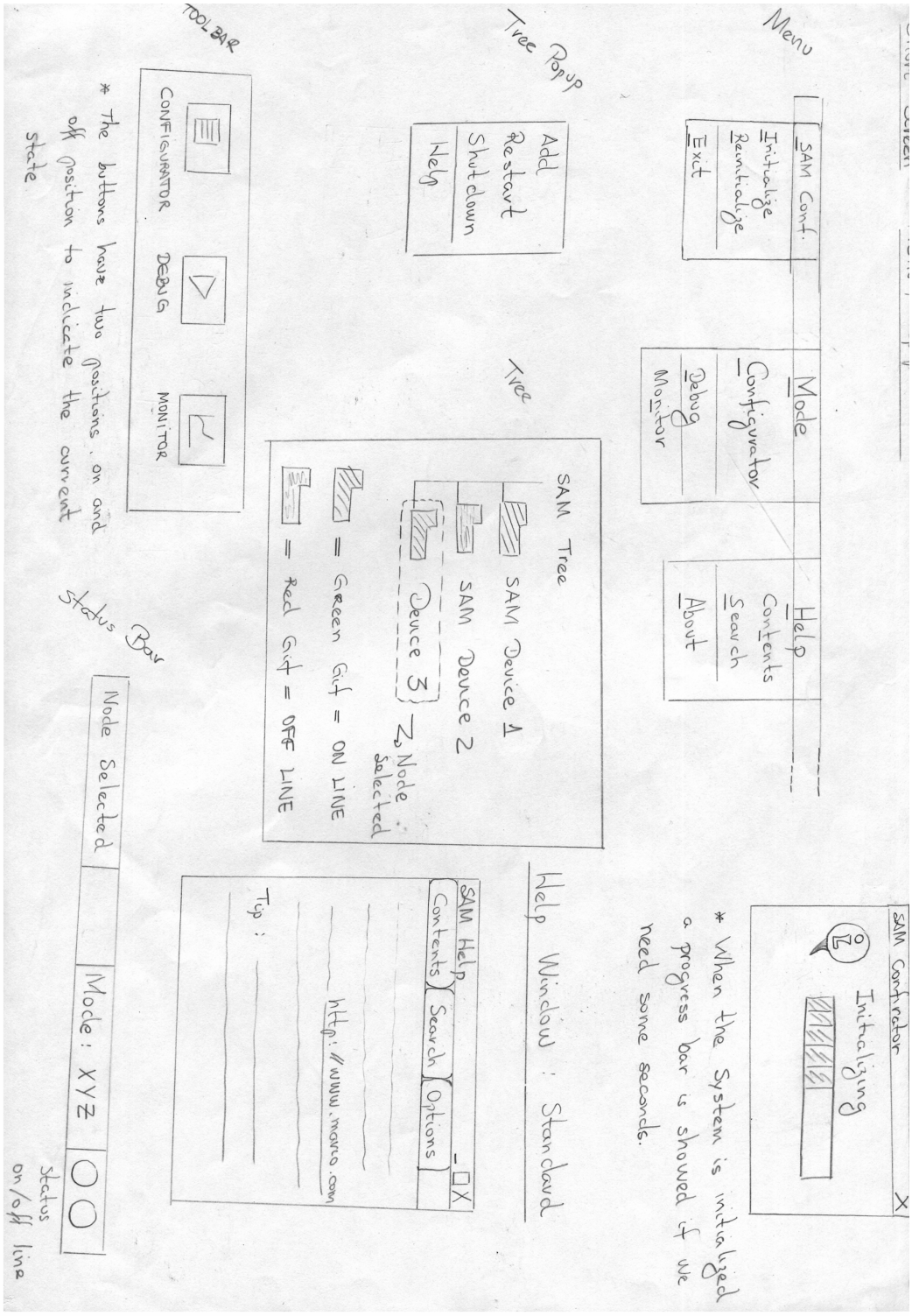
Croquis da Tela de Debug (Configuração)



Croquis do Modelo de Início do Sistema



Croquis da Tela dos Recursos Utilizados



Croquis da Lista de Requisitos

List of requirements

① SAM Configurator must read data of the SAM Conf. file.

② SAM Configurator must set news values to sensors and actuators.

③ SAM Configurator must be able to make a debug of the news values.

④ SAM Configurator must make a monitoring of the devices.

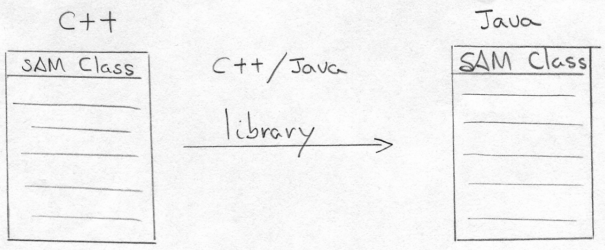
⑤ SAM Configurator must reinitialization or update of the SAM's Clients and Servers.

⑥ SAM Configurator must have a help on each screen for each field (tooltip) and one help for the whole program

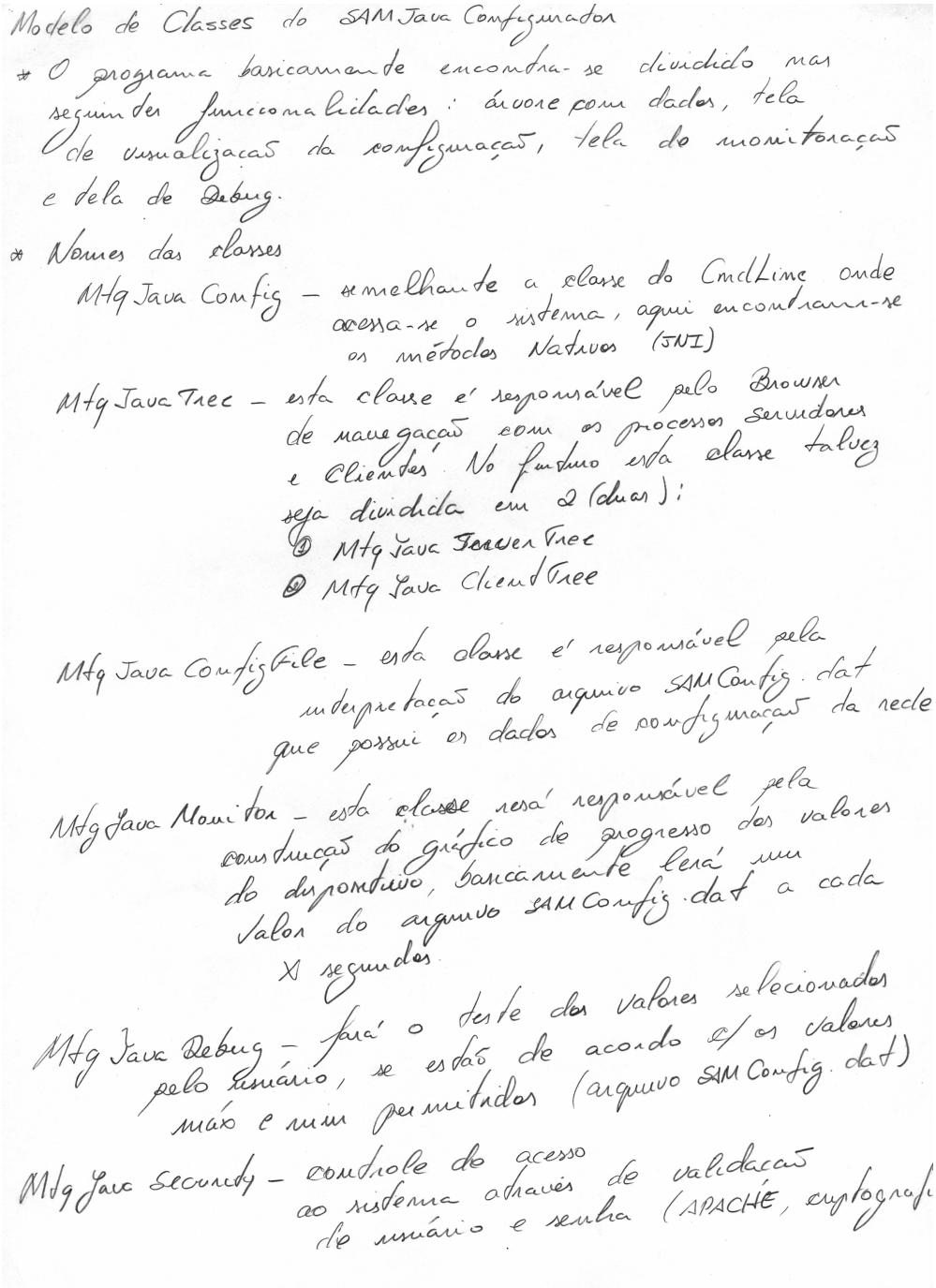
⑦ SAM Configurator must show us wich state it is working for example: monitoring state, read state, write state, debug state, reinitialize state.

⑧ SAM Configurator must be an applet that is plataform independe and must be constructed with AWT, Swing 2.0 or/and JavaBe

⑨ The communication between Java SAM Configurator and the C++ SAM classes will be done with Java Native Interface (JNI) Technology.



Croquis das classes iniciais



Modelo de Classes do SAM Java Configurador

* O programa basicamente encontra-se dividido nas seguintes funcionalidades: árvore com dados, tela de visualização da configuração, tela do monitoração e dela de Debug.

* Nomes das classes

Mfg Java Config – semelhante a classe do CmdLine onde acessa-se o sistema, aqui encontram-se os métodos Nativos (JNI)

Mfg Java Tree – esta classe é responsável pelo Browser de navegação com os processos servidores e Clientes. No futuro esta classe talvez seja dividida em 2 (duas):
① Mfg Java Server Tree
② Mfg Java Client Tree

Mfg Java ConfigFile – esta classe é responsável pela interpretação do arquivo SAM Config.dat que possui os dados de configuração da rede

Mfg Java Monitor – esta classe será responsável pela construção do gráfico de progresso dos valores do dispositivo, basicamente terá um Valor do arquivo SAM Config.dat a cada X segundos

Mfg Java Debug – fará o teste dos valores selecionados pelo usuário, se estão de acordo c/ os valores máx e mín permitidos (arquivo SAM Config.dat)

Mfg Java Security – controle do acesso ao sistema através de validação de usuário e senha (APACHE, cryptograf

BIBLIOGRAFIA

[1] MICHAELIS. Moderno Dicionário da Língua Portuguesa. Editora Melhoramentos.

[2] FERREIRA, J.C.E. Apostila de Sistemas Integrados de Manufatura (SIM) Universidade Federal de Santa Catarina. Florianópolis, 1998. p. 1-2.

[3] *IEEE Transactions on Power System*s, New York, v. 11, n. 1, p. 206-215, Feb. 1996.

[4] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *Referências Bibliográfica*s, NBR 6023. Rio de Janeiro, 1989.

[5] UFPR.STL. Formatos para Edição de Textos. Emílio Carlos Boschilia. Universidade Federal do Paraná, Biblioteca Central. Curitiba, 1990. 1 disquete 5 ¼ pol. Word 5.0.

[6] PFEIFER, T., SACK, D. Sensor/Actuator-Network: A nervous system for the Autonomous Production Cells, Sensors & Systems Journal, Moscow, Edition 6/2001.

[7] ORTH, A., PFEIFER, T., SACK, D. e STEMMER, M. R. (2000). Sensor/Actuator Network - The Nervous System Of A New Production Concept: The Autonomous Production Cells, Congresso Brasileiro de Automática, Florianópolis, Brasil (SBA'2000).

[8] GOURLEY, C., Wheels of Time : A Biography of Henry Ford, Millbrook Pr; ISBN: 076130214X - 48 pages, October 1997.

[9] RWTH Aachen, SFB 368 - Arbeits- und Ergebnisbericht, 1999.

[10] SPERLING, W; LUTZ, P. Enabling open control systems - Introduction to OSACA system platform, in Robotics and Manufacturing, Volume 6 - ISRAM '97, Montpellier, France, edited by M. Jamshidi, et al. ASM Press New York, 1996.

[11] OSACA Association. OSACA Handbook, Version 1.0.1, http://www.osaca.org, 1997.

[12] BONFIG, K. W. Feldbus-Systeme, Expert, Ehningen, 1992.

[13] DIN EN 50170/1- EN 50170/3, Universal Fieldcommunicationsystem; Beuth Verlag, Berlin, Edition: 1997-07.

[14] TogetherSoft, http://www.togethersoft.com. Site da ferramenta de UML, 2002.

[15] Metodologia de Documentação do S2i, http://s2i.das.ufsc.br/s2iDoc, 2002.

[16] Doxygen, Sistema para Documentação de Códigos C/C++/C#, Java, PHP. Site: http://www.doxygen.org, 2002.

[17] ROLOFF, M.R., STEMMER, M.R. (2001). Sensoriamento Remoto Usando uma Interface Java Native Interface para Aplicação em Sistemas Industriais de Visão. Simpósio Catarinense de Processamento Digital de Imagens, Florianópolis, Brasil (SCPDI 2001).

[18] ORTH, A., PFEIFER, T., SACK, D., ROLOFF, M.R., and STEMMER, M. R. (2002). Measuring Flank Tool Wear on Cutting Tools with Machine Vision – A Case Solution, Mechatronics and Machine Vision in Practice, Tailândia (M2VIP'2002).

[19] ROLOFF, M.R. (2000). Desenvolvimento de um Browser MAP/MMS usando JNI. Projeto de Final de Curso – ECAI – DAS – UFSC, Florianópolis, Brasil.

[20] HORSTMANN, C.; CORNEL, G.: "Core Java II – Volume I - Fundamentals", Person, 1998.

[21] HORSTMANN, C.; CORNEL, G.: "Core Java II – Volume II – Advanced Features", Person, 1999.

[22] ECKEL, B.: "Thincking in C++", Second Edition, Prentice Hall Inc., 2000.

[23] ECKEL, B.: "Thincking in Java", Prentice Hall Inc., 1998.

[24] STROUSTROUP, B.: "The C++ Programming Language", Addison Wesley Longmann Inc., 1997.

[25] LIANG, S.: "Java Native Interface: Programmer's Guide and Specification", Addison Wesley Longmann Inc., June 1999.

[26] HEAP, Nicholas, "An Introduction to OSI" Computer Science Texts, Blackwell Scientific Publications, 1993 Printed and bound in Great Britain by Hartnolls Ltd, Bodmin, Cornwall.

[27] MORGAN, Michael, "JAVA 2 for Professional Developers – The Authoritave Solution" SAMS – A Division from Macmillan Computer Publishing, 201 West 103rd. St. , Indianapolis, Indiana, 46290 USA.

[28] PISCITELLO, David m. and CHAPIN, A. Lyman, "Open Systems Networking TCP/IP and OSI", Addison-Wesley Publishing Company, 1993, Massachusetts, USA.

[29] STALLINGS, William, "Networking Standards – A Guide to OSI, ISDN, LAN and MAN Standards" Addison-Wesley Publishing Company, 1993, Massachusetts, USA.

[30] STEMMER, Marcelo Ricardo, "DAS5331 – Sistemas Distribuidos e Redes de Computadores para Controle e Automação", Universidade Federal de Santa Catarina – UFSC, Departamento de Automação e Sistemas – DAS, Engenharia de Controle e Automação Industrial – ECAI, 2001.

[31] LIANG, Sheng, "The Java Native Interface – Programmer's Guide and Specifiction" , Addison-Wesley Publishing Company, 1999, Palo Alto, California, USA.

[32] "Java Native Interface – Realease 1.1 (Revised, May 1997)", SUN Microsystems Inc. 2550, Garcia Avenue, Mountain View, California, 94043-1100 USA.

[33] STEMMER, Marcelo Ricardo, "DAS5302 – Informática Industrial I", Universidade Federal de Santa Catarina – UFSC, Departamento de Automação e Sistemas – DAS, Engenharia de Controle e Automação Industrial – ECAI, 2001.

[34] STEMMER, M. R., "DAS5303 – Informática Industrial II", Universidade Federal de Santa Catarina – UFSC, Departamento de Automação e Sistemas – DAS, Engenharia de Controle e Automação Industrial – ECAI, 2002.

[35] STEMMER, M. R., "DAS6606 – Redes de Comunicação", Universidade Federal de Santa Catarina – UFSC, Departamento de Automação e Sistemas – DAS, Pós-Graduação em Engenharia Elétrica – PGEEL, 2000.

[36] TANENBAUM, Andrew, "Computer Networks" 4th. Edition, Pearson, 2002, USA.

[37] TANENBAUM, Andrew, "Redes de Computadores", Editora Campus, 1997, São Paulo, Brasil.

[38] SOMMERVILLE, Ian, "Software Engineering", Addison-Wesley Pub Co, 2000, 6th edition, USA.

[39] PRESSMAN, Roger S., "Software Engineering: A Practitioner's Approach", McGraw-Hill Science/Engineering/Math, 2001, 5th edition , USA.

[40] JAVA Sun, http://java.sun.com", 2002, USA.

[41] CAULLIRAUX, H., COSTA, M. C. O., "Manufatura Integrada por Computador: Sistemas Integrados de Produção", Editora Campus, 1999, São Paulo, Brasil.

[42] SLACK, N., CHAMBERS, S., HARRISON, A., "Administração da Produção", Editora Atlas, 2002, São Paulo, Brasil.

[43] VOLLMANN, T. E., BERRY, W. L., WHYBARK, D. C., "Manufacturing Planning and Control Systems", McGraw-Hill, 1997, USA.

[44] Prof. Luis Tadeu Almeida da FORDESI de Lisboa, "http://www.newcastle.research.ec.org/esp-syn/text/8486.html", 2002, Lisboa, Portugal.

[45] ALLAIRE, "http://www.allaire.com", 2002, USA.