Escola Politécnica da Universidade de São Paulo

PMR – Departamento de Engenharia Mecatrônica e Sistema Mecânicos

MONOGRAFIA

Controle de Múltiplos Veículos de Transporte Autônomos baseado no conceito de Sistemas Distribuídos

Aluno: João Alfredo Anastácio Vieira Júnior

Orientador: Prof. Dr. Diolino J. Santos Filho

São Paulo, dezembro de 2010

JOÃO ALFREDO ANASTÁCIO VIEIRA JÚNIOR

Controle de Múltiplos Veículos de Transporte Autônomos baseado no conceito de Sistemas Distribuídos

Monografia apresentada como documentação necessária para diploma de graduação em engenharia mecânica com ênfase em mecatrônica pela Escola Politécnica da USP

Orientador: Prof. Dr. Diolino J. Santos Filho

São Paulo, dezembro de 2010

FICHA CATALOGRÁFICA

Vieira Júnior, João Alfredo Anastácio
Controle de múltiplos veículos de transporte autônomos baseado no conceito de sistemas distribuídos / J.A.A. Vieira Júnior. -- São Paulo, 2011.
67 p.

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

1. Sistemas de produção 2. Sistemas autônomos 3. Sistemas distribuídos 4. Robôs I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II. t.

ÍNDICE geral

ANEXOS

LISTA de figuras

LISTA de tabelas

SIGLAS e NOMECLATURAS

API	– application programming interface
ASP	– active Server pages
ATO	– assemble to order
BD	– base de dados
CP	– controle de processos
CR	– controle de recursos
CVGD	– centróide virtual de demanda global de transporte
DER	– diagrama entidades e relacionamentos
DO	– distributed objetcts
EPUSP	–Escola Politécnica da Universidade de São Paulo
ETO	– engineering to order
FTP	– file transfer protocol
HTTP	– hyper text transfer protocol
IaaS	– infrastructure as a service
IP	– internet protocol
ISS	– internet information services
MKS	– make to stock
MTO	– make to order
MVC	– model view controller
NDDJSS	– non deterministic dynamic job shop system
PaaS	– Plataform as a service
SaaS	– Software as a service
SAP	– sistemas antropocêntricos de produção
SED	– sistemas produtivos a eventos discretos
SOAP	– simple object access protocol
SP	– sistema produtivo
SPF	– Sistema Produtivo Fabril
SQL	– structured query language
StP(s)	– Stored Procedures (procedimentos armazenados)
TCP	– transmission control protocol
UC	– casos de uso
URI	– uniform resources identifiers
URL	– uniform resource location
VT	– veículos de transporte
W3C	– world wide web consortium
WS	– web services
WSDL	– web service description language
WWW	– world wide web
XML	– extensible markup language

RESUMO

Nos dias de hoje é necessário que as empresas adéquem-se a inúmeros mercados e diversifiquem sua linha de produção, atendendo simultaneamente a diferentes demandas, sempre de forma eficaz. Um sistema de produção tende a lidar com inúmeras variáveis de acordo com a ramificação de produtos inerente ao setor em que atua. Para atender esses requisitos, a tecnologia vem oferecendo diversos avanços tal que Sistemas Produtivos Inteligentes que aceitem múltiplos processos é o objetivo a ser alcançado. Para isso, é necessário que um sistema possa integrar e comandar máquinas completamente distintas e que nem sempre estão próximas fisicamente, tanto entre si, quanto de seus operadores. Sob essa ótica, e com o crescente avanço que podemos ver nas tecnologias envolvendo redes e internet, temos a possibilidade de atingir a meta de ter sistemas produtivos capazes de adaptar a indústria a atual realidade através do desenvolvimento de ambientes virtuais que além de conversarem com arquiteturas distintas, possam ser controlados remotamente e com segurança. Neste contexto, aplicam-se soluções baseadas em Web Services / Web Servers.

Neste trabalho, por meio de um ambiente virtual, deseja-se simular o controle do processo de integração de veículos de transporte autônomos (ROBOTINO® - FESTO) com um modelo de Sistema Produtivo fabril através da implementação de um Web Service com controle de múltiplos veículos de transporte, solucionando assim o problema de flexibilidade nas rotas de movimentação exigida por essa produção com a possibilidade de integração de diferentes recursos de forma remota. Desta forma, pretende-se contribuir para oferecer maior flexibilidade e controle de processos para se obter maior produtividade do conjunto.

Palavras chave: Sistemas Produtivos Antropocêntricos Dispersos, Veículo de Transporte Autônomo, Teleoperação Remota, Robotino, Web Service.

ABSTRACT

Nowadays, the enterprises need to adapt themselves to a many distinct markets and diversify their production to attend different demands. A production system work with many variables according to the products in the sector in which it operates. In order to attend these requirements, the technology is being offering many advances: Smart Production Systems able to accept multiply process are the main target. To do this, the systems need to integrate and command different machines, in their constructions or closeness, among them or the operators. With this optic, and with the large growth of the net and internet technologies we can hit the target of to develop that production systems able to adapt the industry into this new reality, designing virtual environments that can talk with different architectures and be controlled remotely and safety. In this context we can apply solutions based in Web Services / Web Serves.

With this work, through a virtual environment, we wish to simulate the process control of integration of an autonomous transport vehicle (ROBOTINO® - FESTO) with Factory Production System models through the implementation of a web Service with multiply transportation vehicle control , solving the moving flexibility routes problems inherits of this type of production with the possibility of integrate two different systems and controlling everything in a remote way. In this context, the solution offers more flexibility to the process and increases the productivity.

KeyWords: Disperse Anthropocentric Production Systems, Autonomous Transportation Vehicle, Remote Teleoperation, Robotino, Web Service.

1.

Introdução

As indústrias que desejam crescer nos dias de hoje necessitam investir na flexibilidade das suas linhas de produção para enfrentar a concorrência oferecida pelo mercado frente a uma demanda cada vez mais forte e exigente. Aliado a esse fator, vemos um crescente avanço tecnológico tanto para a criação de produtos para consumo final quanto para desenvolvimento de ferramentas de produção. Um produto que se enquadra nessas duas demandas é a internet. Ela favorece tanto aos consumidores, fornecendo mais opções para atender às suas necessidades, quanto aos produtores, oportunizando ferramentas para atender desejosa dinâmica de cadeias produtivas dispersas.

Dos desafios que a indústria contemporânea precisa enfrentar, pode-se destacar a necessidade de fazer com que diferentes aplicativos possam intergrar-se entre si. Esta é uma forma de flexibilizar as rotinas de produção e adaptar a linha para diferentes necessidades, de maneira que o custo de produção ainda permaneça viável e competitivo. Soma-se a este cenário a necessidade de poder desvincular fisicamente a proximidade de vários processos envolvidos na fabricação de um produto devido a menores custos e/ou questões administrativas ou logísticas.

Web Services, um dos objetos de estudo desse trabalho, são sistemas capazes de integrar diferentes dispositivos e aplicativos, organizando e controlando a interação entre esses e o operador/cliente e ainda, de maneira segura, já que se pode controlar o quanto de ação humana poderá interagir no processo.

Para esse trabalho, objetivou-se integrar um robô móvel, denominado Robotino, adequado para desempenhar as funções de um veículo autônomo de transporte a um modelo de sistema produtivo de ambiente fabril (SPF). O sistema integrado simula a dinâmica de um ambiente de produção que cuidam das tarefas de produzir um determinado item, de acordo com processos pré-programados e de realizar o transporte de insumos de produção e dos produtos acabados para estocagem, por exemplo.

Essa integração será feita através de um Web Service integrado a um Banco de Dados que poderá ser acessado pela internet e possibilitará a execução de rotinas de operações do robô para locomoção de forma inteligente entre as unidades que compõem o layout fabril, simulando o transporte de insumos e produtos do SPF. O Web Service possibilitará também, com acesso ao Banco de Dados, o histórico das operações do robô, possibilitando estudo para melhorias nas rotas de movimentação, otimização do trabalho e backup para possíveis erros e problemas que venham a aparecer com o decorrer do uso da solução.

2.

REVISÃO BIBLIOGRÁFICA

2.1 SISTEMAS PRODUTIVOS

2.1.1 Contexto

Os Sistemas Produtivos (SPs) podem ser entendidos como uma classe de sistemas cuja finalidade é a produção/prestação de itens/serviços, com determinadas peculiaridades em seu comportamento dinâmico (SANTOS FILHO, 2000).

SPs pertencem a classe de Sistemas a Eventos Discretos (SEDs) (HO, 1989 . SANTOS FILHO; MIYAGI, 1991 . SANTOS FILHO, 2000) e é uma classe de sistemas feitos pelo homem e para o homem, denominados *"man made systems"* (ITO, 1991).

Considerando a evolução atual dos recursos computacionais, há um forte desenvolvimento de sistemas artificiais em que a lógica dos processos não é trivial. Esta característica dificulta a obtenção de soluções de controle totalmente automatizadas, necessitando de operadores humanos especialistas. Neste contexto, torna-se adequada uma abordagem antropocêntrica para o tratamento destes sistemas (KOVÁCS; MONIS, 1995 . SANTOS FILHO; MIYAGI; MARUYAMA, 1999). Atualmente, esta técnica pode ser implementada com maior facilidade em virtude da evolução das ferramentas de conectividade sendo possível monitoração remota dos sistemas para tomar-se decisões para intervir-se em seu comportamento.

2.1.2 Comportamento

O sistema evolui por eventos discretos se suas transições de estado acontecem por meio de eventos durante intervalos de tempo finito (Figura 2.1). Assim, pode ocorrer paralelismo e/ou conflito de eventos causando até indeterminismo quanto a ocorrência em função do tempo (SANTOS FILHO, 2000).

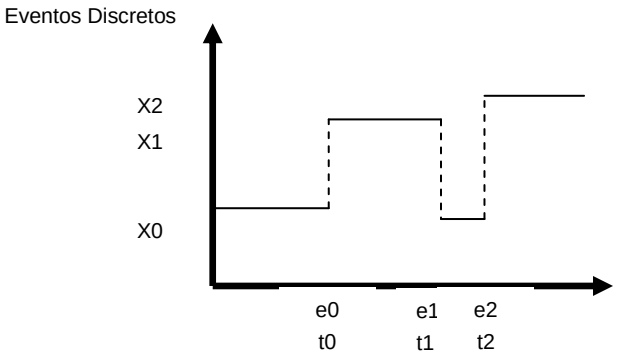

Figura 2.1 - SP a eventos discretos

2.1.3 Controle

Controle de um sistema é definido como a aplicação de um conjunto de ações pré-estabelecidas para que, aquilo que se considera como objeto de controle, atinja objetivos determinados, refletindo o comportamento dinâmico desejado para o sistema (MIYAGI, 1996; SANTOS FILHO, 2000; NAKAMOTO, 2008).

O comportamento dos sistemas e a evolução da sua dinâmica são definidos pelo projetista a partir de relações de causa-efeito diretamente vinculadas à ocorrência de eventos discretos e disponibilidade de recursos (SANTOS FILHO, 2000). Na Fig. 2.2 apresenta-se a arquitetura de um sistema de controle antropocêntrico.

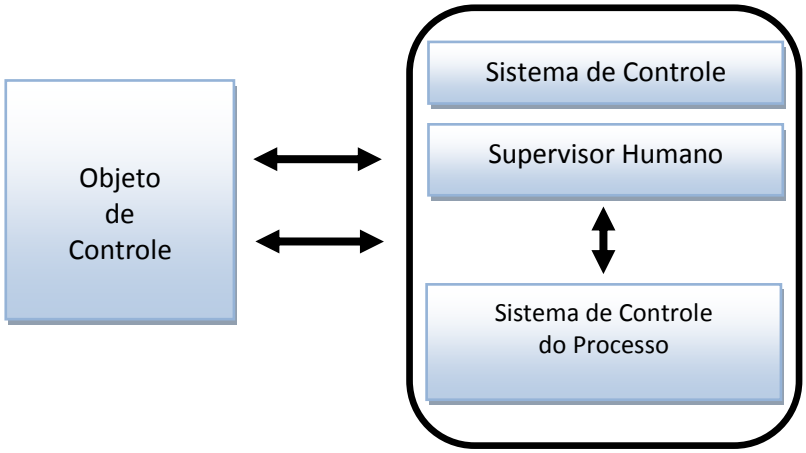

Figura 2.2 - Arquitetura de Controle de Sistema Antropocêntrico de Produção

MYIAGI (1996) criou um diagrama básico para sistemas de controle, que SANTOS FILHO, 2000, aperfeiçoou nos seus trabalhos, propondo uma divisão do sistema em níveis de controle hierárquicos, colaborativos e de semânticas distintas (NAKAMOTO, 2008).

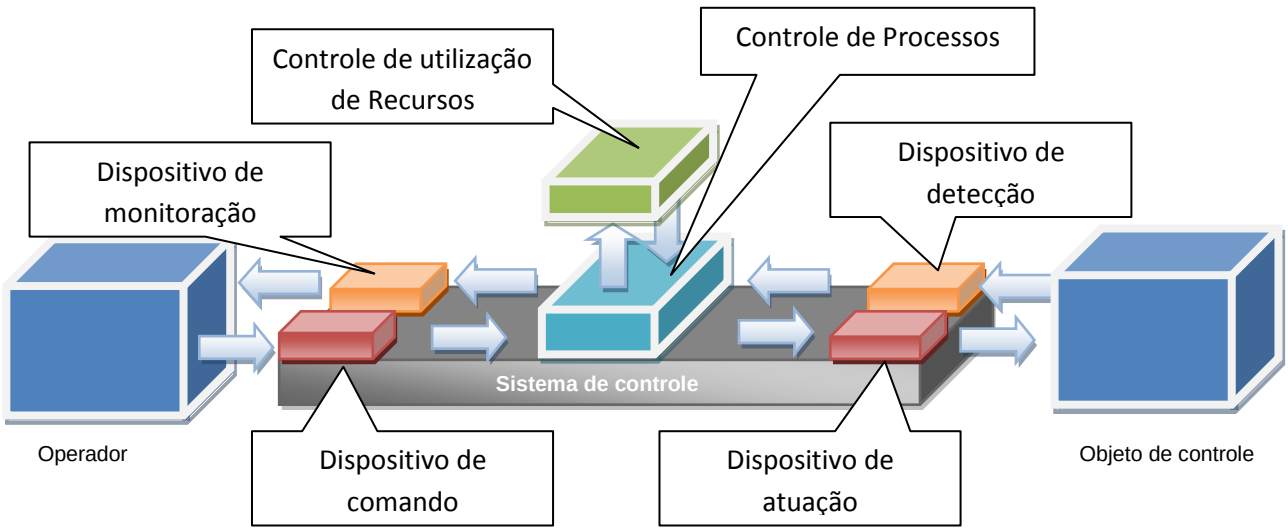

Figura 2.3 – Arquitetura do sistema de controle por Santos Filho (2000)

De acordo com SANTOS FILHO (2000) e NAKAMOTO (2008), o sistema de controle é dividido em dois níveis denominados controle de processos (CP) e controle de recursos (CR). O CP é responsável pelo seqüenciamento das atividades do processo, ou seja, é o controle que garante a execução das atividades de transformação do processo, enquanto o CR é responsável pela utilização dos recursos entre as atividades de cada um dos processos. O CP

executa cada uma das atividades, requisitando o recurso de transformação, alocando o respectivo recurso e monitorando a sua execução.

Quando é necessária a alocação de um recurso, o CP requisita o recurso para o CR que possui a visão global quanto à utilização dos recursos. Salienta-se que o controle de cada processo possui apenas uma visão de controle local. De acordo com as regras pré-estabelecidas, o CR autoriza ou não a utilização de recursos, resolvendo desta forma, os conflitos entre os processo quanto à alocação de recursos.

2.2 Sistemas Dispersos

Sistemas dispersos é a solução para uma arquitetura de ambiente colaborativo distribuído, visando diversificar e aperfeiçoar a produção e os custos envolvidos.

Podemos dividir a estrutura de um sistema disperso em dois modelos: serviços orientados e sistemas produtivos dispersos com base na internet (service-oriented and Internet-based disperse productive systems) sendo os últimos divididos em e-engineering e e-manufaturing, todos fazendo referencia ao conjunto de sistemas de eventos discretos (MIYAGI et AL, 2009).

Ainda de acordo com MIYAGI et al (2009) o chamado e-business (internet business) proporcionou maior velocidade para os processos de manufatura relacionando melhor clientes, fornecedores e parceiros. Diversas áreas técnicas sofreram melhorias devido às aplicações baseadas na internet e a maneira como ocorre essa troca de informações. Como exemplos temos:

- Síntese de sistemas, modelos, simulação e controle de processos de manufatura;
- Tecnologias que convertem informações em conhecimento para tomada de decisões;

- Métodos de treinamento que permitem maior assimilação do conhecimento;
- Softwares para sistemas colaborativos inteligentes;
- Produtos e métodos de desenvolvimento de processos que permitem ampla gama de requerimentos;
- Sistemas que permitem mudar de configuração/setup rapidamente ;
- Interface homem-máquina aprimorada;
- Processos de manufatura que permitem minimizar as perdas na produção e no consume de energia;
- Processos inovadores de desenvolvimento e manufatura de novos materiais e componentes.

Os principais desafios no desenvolvimento de sistemas dispersos podem ser expostos como: organizações colaborativas interligadas, modelos de referencia, validação e verificação, qualificação desses modelos e interoperabilidade.

Esses desafios podem ser alcançados individualmente considerando diferentes conceitos e teorias. Eles podem ser agrupados em dois grupos complementares: e-engeneering e e-manufacturing.

E-engeneering (internet based engeneering) - permite que grupos de engenheiros possam conversar para resolver problemas e desenvolver produtos por meio de equipes multidisciplinares em um ambiente colaborativo de execução de atividades ainda que em diferentes organizações. O advento da e-engeneering proporcionou mudanças eficazes no desenvolvimento de processos produtivos. O desenvolvimento de produtos está cada vez mais remoto sendo executado em diferentes partes de um

mercado globalizado em expansão. São equipes de projetos alocadas em diversas partes do globo e até de diversas companhias trabalhando juntas no mesmo assunto. Todo esse processo garante que haja transferência e ou compartilhamento de conhecimento e informação entre ambientes e centrais de tecnologia, distribuídos e dispersos geograficamente.

- **E-manufacturing ou Internet-based manufacturing** - entendido como o uso da internet para troca de informações em um ambiente distribuído e disperso. O conceito emergiu durante os ultimo anos como conseqüência da popularização das aplicações de internet no ambiente empresarial. De acordo com MYIAGI et AL (2009) e Lee (2003) e-manufaturing é um conceito recente de desenvolvimento que permite ter contato com as necessidades que as estratégias derivadas dos processos de e-business requerem, integrando completamente atores e recursos de um ambiente de produção através de ferramentas e tecnologias próprias da internet. Ele permite o gerenciamento de recursos, analise de performance da produção e manutenção das operações.

A complexidade de todo esse processo exigiu que o conceito de Arquitetura Orientada a Serviços (SOA) surgisse. Com ela, tanto serviços internos quanto externos a produção podem ser requeridos de maneira simples e integradora, trabalhando em conjunto, mas com independência para acesso, introduzindo novas funcionalidades para atender ao processo produtivo.

2.3 Sistemas Tele-operados

Os sistemas tele-operados são tipos de sistemas capazes de serem gerenciados por controle remoto. Operadores e máquinas não precisam estar geograficamente no mesmo local, sendo possível operá-las como localmente, a fim de realizar a tarefa. Basicamente, esses sistemas

são compostos por sensores, atuadores, controladores, interface homem-máquina, canal de comunicação e sistema suplementar para realização de tarefas remotamente. A unidade de comando faz a interface homem-máquina com a unidade comandada e por meio de comandos enviados através do link de comunicação executa as ordens dadas, tipificando-se uma arquitetura de natureza cliente-servidor.

A evolução da internet vem contribuindo e propiciando o desenvolvimento desse tipo de aplicação através de tecnologias com menor custo de implementação/manutenção e maior flexibilidade.

Dentre os diversos fatores de sucesso da tecnologia de controle remoto pode-se citar: redução de custos com viagens, operação em condições mais seguras para o operador (contra ambientes perigosos ou desfavoráveis) e melhoria do layout fabril. Como contra-ponto, uma estrutura tecnologia da informação necessita ser implementada com clareza para que todo o processo ocorra de maneira controlada e ótima.

2.3.1 Middleware para sistemas tele-operados

O termo trata da geração de aplicativos computacionais com complexidade para analisar e distribuir informações de maneira inteligente e que promova a integração de sistemas que nem sempre possuem a mesma arquitetura. É a camada de funcionalidade situada entre o software/aplicativo e o sistema operacional. Exemplos desse tipo de sistemas são os DO (distributed objects) como Corba, Java e DCom (MIYAGI et AL 2009).

Outra vertente são os Web Services (WS), tema principal desse trabalho, que através de aplicativos computacionais autônomos podem usar URLs (Uniform Resource Location) para transmitir padrões de dados em pacotes/mensagens em XML (eXtensible Markup Language).

A mensagem é passada, identificada, encapsulada e navega na internet por HTTP (Hyper Text Transfer Protocol), para ser lida através de interfaces com outras unidades.

O assunto será tratado em maiores detalhes mais adiante no capítulo 3.

2.4 CONTROLE E ALOCACAO DE MÚLTIPLOS VEÍCULOS DE TRANSPORTE

Define-se conceitualmente sistema como "Conjunto de partes integrantes e interdependentes que, conjuntamente, formam um todo unitário com determinado objetivo e efetuam determinada função." (OLIVEIRA, 2001). O conceito é muito genérico e pode ser aplicado em diversos campos do conhecimento. Mesmo na engenharia, podemos nos referir a sistema como um conjunto de idéias ou equações para se atingir um objetivo, ou um conjunto de peças que formam uma máquina. Nesse projeto, trabalha-se com dois tipos de sistemas, como já mencionado: um sistema produtivo fabril e um veículo de transporte autônomo.

A denominação comum de veículo de transporte autônomo e que pode ser aplicada para esse caso é chamá-lo simplesmente de robô (especificamente denomina-se Robotino). O Robotino, produzido pela empresa FESTO, é um robô móvel didático equipado com sensores e câmeras. Pode ser usado em diversos tipos de tarefas que vão desde uma simples operação de locomoção, controlada por um cenário, até a realização de tarefas como aprender um caminho e realizá-lo em menor tempo possível, envolvendo desvios de rotas além de não colidir com objetos durante o trajeto.

Basicamente, o robô em questão foi projetado para fins didáticos, aceitando diversos tipos de programação e entrada de dispositivos (como garras), com interface amigável e pode ter suas tarefas simuladas em software gráfico e ser controlado remotamente.

No projeto, o Robotino simulará um sistema de transporte inteligente para locomoção das peças produzidas no miniCim para o estoque de produção ou mesmo reabastecimento de insumos. O objetivo será fornecer o destino do robô através de hardware com acesso remoto e, através de um caminho otimizado pelo layout da fábrica, esperar que ele se desloque para o destino, realizando a tarefa de transporte.

Como dito anteriormente, os sistemas produtivos devem apresentar um comportamento baseado na maximização da produtividade e flexibilidade da produção. De acordo com SANTOS FILHO (1998) e DOUMEINGTS et al. (1995) são três os princípios fundamentais dessa estrutura: AUTOMACAO, FLEXIBILIDADE e INTEGRAÇAO.

Em linhas gerais, esses aspectos visam aumentar a produtividade reduzindo os custos de produção controlando a qualidade do produto final e o tempo de produção tendo em vista a adaptabilidade necessária nas linhas devido as diferentes demandas.

Considera-se assim necessário um controle inteligente do fluxo de materiais demandados por essa produção suficiente para cobrir um diferente plano de rotas de acordo com o processo.

Modelando o Sistema Produtivo como um conjunto de estações de trabalho independentes distribuídos em um arranjo físico adequado à linha de produção, a integração destas unidades ocorre a partir do momento em que há, entre elas, um fluxo inteligente e otimizado de matéria-prima e produto acabado, respeitando os estoques, seqüências da produção e gargalos e demandas de cada estação.

Assim, o Sistema de Transporte associado também necessita ter autonomia e flexibilidade suficiente para entender a logística da linha de produção e adaptar-se a cada situação (necessita flexibilidade).

Como visto em SANTOS FILHO (1998), analisando a dinâmica de um sistema de transporte, existe uma variedade de rotas que os veículos necessitam cumprir. Um sistema de tomada de decisões (humano ou artificial) é necessário para se cumprir essa etapa, composta por:

- Controlar-se o seqüenciamento das operações que pertencem a cada processo;
- Controlar-se o que será transportado;
- Controlar-se como será feito esse transporte.

De acordo com o grau de flexibilidade, temos um maior numero de restrições que devem ser respeitadas, aumentando a complexidade da administração do sistema. Dentre as abordagens que podemos dar para esse problema vamos considerar a arquitetura antropocêntrica de produção e portanto de Sistemas Antropocêntricos de Produção - SAP (ITO, 1991). Nela, temos o elemento humano como parte importante do sistema intervindo na evolução dinâmica do processo e tomando decisões de otimização e flexibilização. (SANTOS FILHO, 1998 . KOVÁCS; MONIZ, 1995).

Determinamos assim a necessidade do homem como elemento de participação no sistema. Isso se faz, pois considera-se ele como especialista, com potencial de aprimorar as soluções de tomada de decisão em sistemas construídos para seu próprio beneficio. Soma-se a isso a necessidade de alterações na dinâmica do processo com relevância subjetiva para o andamento do mesmo, como aspectos socioeconômicos e culturais do ambiente no qual se insere o sistema, além da descentralização na estrutura de controle desse ambiente.

2.4.1 DEADLOCKS

Nos SAP temos a necessidade de controlar o comportamento dinâmico do sistema. É inerente o uso de recursos concomitantemente, ou de maneira compartilhada ou competitiva para o avanço da produção. Assim , se faz necessário estudar métodos de controle de fluxo de materiais nos SAPs para poder-se alocar de maneira adequada os recursos entre as várias tarefas e etapas no ciclo do produto.

Encontramos na bibliografia especializada estudos e técnicas feitos no sentido de dar subsídios para o melhor desenvolvimento de projetos em SAPs de acordo com a necessidade. Nesse trabalho será abordado um pouco do histórico desses estudos e adaptar-se-á os aspectos que mais convém com a abordagem e a realidade do projeto diretamente, visando ajustar modelos de solução e problema de maneira prática.

Para modelar a dinâmica do processo, a modelagem do sistema é feita de maneira a obter-se uma fila de eventos sendo que o próximo evento realizar-se-á somente após o anterior liberar o recurso (veículo de transporte). Assim se estabelece uma relação de acordo com o fato observado de que são quatro as condições necessárias para ocorrência de deadlock em sistemas onde há processos concorrentes. A saber (BANASZAK; KROGH, 1990):

- Mutua exclusão;
- Retenção enquanto espera;
- Não preempção;
- Espera circular .

A ação humana pode alterar a ordem dos eventos privilegiando um ou outro por motivos convenientes.

Pode-se também criar uma especificação sobre o tipo de evento que se esta realizando (busca de matéria prima, entrega de produto, avanço no processo) e estabelecer uma ordem padrão

entre a importância desses eventos. Essa ordem seria então respeitada salve intervenção humana no processo.

Um aprimoramento nesse sistema consiste em você fazer um sistema de informação mais sofisticado que o proposto nesse trabalho integrando não somente uma unidade produtiva com a de transporte, mas também todas as informações das unidades de armazenamento e outras unidades de produção.

Assim, modela-se o sistema impondo relações de ordem e prioridade entre as unidades e assim faz-se posteriormente uma regra de controle dos veículos de transporte cada vez mais inteligente para o sistema. A intervenção humana nesse caso seria para aprovar a seqüência de operações tendo o privilegio de alterar, por qualquer motivo relacionado a demanda de produção ou peculiaridade, a ordem das operações. Nesse caso o sistema exigiria que o processo fosse devidamente documentado para posterior analise ou simples arquivamento, prevendo, no futuro melhorias com as informações obtidas no passado.

2.4.2 Alocação de Veículos de Transporte em SAPs

Existem dois tipos de transportadores, de acordo com a aplicação:

- Para Rotas Fixas (como rolos, correias, camadas de ar etc)
- Para Rotas genéricas (veículos autônomos ou por operadores que realizam rotas alternativas dos materiais)

Existe um substancial ganho em operacionalidade com o tempo no uso de transportadores de rotas genéricas nos sistemas produtivos, pelo fato da dinâmica do sistema e experiência adquirida pelos operadores para aquelas situações.

Em SAPs o controle antropocêntrico do sistema é uma garantia de elevado nível de flexibilidade operacional.

Sabe-se que, com o passar do tempo de operação, ocorre uma polarização dos veículos de transporte (VT) em torno de regiões de grande demanda, causando queda de desempenho do sistema. Problemas de congestionamento de rotas e espera por recursos são fatores que comprometem a produtividade.

Não cabe ao controle de navegação do VT dirigir isso, sendo necessário entrar com uma lógica que supervisione essa tarefa numa camada acima do sistema para que ocorra uma distribuição organizada dos recursos disponíveis.

Dependendo do modelo adotado, o problema de representação das rotas é minimizado pois os operadores possuem conhecimento necessário para decidir qual o percurso mais adequado para determinado par de origem/destino. Assim, o controle dos VTs restringe-se em especificar esse par de origem / destino que os transportadores devem percorrer para realizar o fluxo de materiais. Cabe a estratégia de controle estabelecer o melhor procedimento para alocar os veículos na melhor ordem considerando a variabilidade da demanda e alterações na dinâmica do processo (como restrições por falhas, manutenção, alterações na produção etc).

Existem duas filosofias que podem ser adotadas para a seleção do VT, de acordo com a requisição (SANTOS FILHO, 1998):

- Rotas fixas pré-definidas para cada VT;
- Autonomia na escolha da rota.

Ambas possuem estratégias próprias de controle sendo que a segunda, por ter variáveis dinâmicas envolvidas na escolha, tem um custo computacional mais elevado e maior cuidado para análise. Erros na abordagem podem causar, com a evolução do sistema, custos maiores

para o transporte com tempo, deslocamento dos veículos ou congestionamento de rotas, diminuindo o desempenho global do sistema.

Para contornar os problemas inerentes aos SAPs e o uso de múltiplos veículos e rotas de transporte, deve-se levar em conta os seguintes aspectos no desenvolvimento do controle (SANTOS FILHO, 1998):

- O sistema deve apresentar elevado nível de flexibilidade operacional ;
- O sistema deve ser capaz de alocar os transportadores automaticamente segundo uma distribuição homogênea em função da demanda;
- O sistema deve ser capaz de eliminiar os congestionamentos dos veículos, ocasionando filas;
- O sistema deve ser capaz de adaptar-se a imprevistos.

Encontram-se, na literatura, técnicas aplicadas para a solução desse tipo de problema. Elas se baseiam em regras como *canto noroeste* e *método de Vogel* (SANTOS FILHO, 1998; NOVAES, 1978; BRONSON, 1985 e PUCCINI, 1987), que consistem em:

- Determinação de uma solução inicial;
- Teste da solução para condição de ótimo;
- Melhoria da solução caso não seja ótima.

Elas constituem-se sobre algoritmos de programação inteira e o problema de designação consiste em atribuir tarefas aos veículos de transporte de modo que o custo total seja mínimo.

Embora não se chegue à solução do problema, obtemos um grande auxilio com o uso desses métodos. Para o problema de alocação então é necessário uma metodologia de controle que designe a alocação dos veículos de acordo com as condições iniciais.

A organização do layout de produção definindo a melhor localização dos pontos de estacionamentos de acordo com a demanda e principais rotas é uma boa forma de começar o planejamento para otimização de todo o processo.

2.4.3 As Rotas de transporte

Os veículos estudados nesse trabalho tem capacidade de fazer desvios no caminho contra objetos ou outros veículos que possam os interromper.

As rotas de transporte devem satisfazer a questão referente ao menor consumo de recursos, tempo e disponibilidade de infraestrutura para locomoção de maneira organizada dos veículos.

Assim, podemos fazer a análise dos trajetos realizados pelos transportadores considerando rotas muito longas (entre as zonas produtivas) ou menores (dentro de cada zona produtiva).

Pelo ambiente simulado nesse trabalho, considerar-se-á a idéia de transporte sempre dentro de uma zona produtiva, fazendo-se assim o modelo de uma única rota para cada estação de trabalho.

Usar-se-á esse fato e a idéia de otimização das rotas de transporte para caminhos pré-definidos, com simples capacidade de desvios ou ultrapassagens para elaboração do layout fabril.

Atualização dos pontos de estacionamento

Os pontos de estacionamento devem ser atualizados com freqüência, dependendo da dinâmica do sistema, independentemente da existência ou não de falhas.

Considerando a ação humana atuando na dinâmica do sistema, podemos atribuir a função de atualizar os pontos de estacionamento ao operador humano, que possui experiência e condições de analisar a situação e de quando deve haver mudanças.

Centróide virtual

O posicionamento dos VTs é dado de acordo com sua importância. Assim, a vazão de material é interpretada como a taxa media de requisição de transporte de produtos. Define-se assim o CVGD (Centróide Virtual de Demanda Global de Transporte).

2.4.3 Algoritmo de controle

Após alocar corretamente os VTs é necessário fazer o algoritmo de controle para esse sistema. É interessante ressaltar (SANTOS FILHO, 1998):

- O tempo gasto de maneira produtiva será o intervalo contado a partir do instante em que o item a ser transportado foi carregado em um VT até o instante em que a estação de carga/descarga correspondente ao destino deste item é alcançada e a descarga é completada.

- Todos os intervalos de tempo gastos para o VT se locomover até a origem, carregar o material a ser transportado, enfrentar possíveis congestionamentos e descarregar o

material correspondem a perdas de eficiência no que se refere ao tempo de utilização do VT envolvido no processo.

Como premissa básica, deve-se definir:

- *O que* deve ser transportado: definindo a prioridade de transporte, para reduzir ou anular problemas de gargalo.
- *Quem* deve transportar: para reduzir os custos de transporte, define-se o VT que estiver mais perto da estação de trabalho.

A seguir, analisam-se as possíveis ocorrências sobre as rotas de transporte (par origem/destino pertencendo ou não a mesma região). Deseja-se que os pares pertençam sempre à mesma região. Dependendo do caso, adota-se um algoritmo de controle específico (extraído de SANTOS FILHO, 1998).

3.

IMPLEMENTAÇÃO e ANÁLISE DA SOLUÇÃO

3.1 Método Aplicado

Para cumprir os objetivos descritos anteriormente aplicou-se o seguinte procedimento:

- Desenvolvimento de layout de produção fabril identificado por Sistema de Produção, Flexível (SPF), Veículo de Transporte Robotino (VT) e três Estações de Parada, representando estoques de insumos/matéria-prima e armazém ou estoque de produtos fabricados e saída da fábrica além de um Ponto de Estacionamento para cada VT atuante. A Fig. 3.1.a descreve o modelo virtual deste SPF.

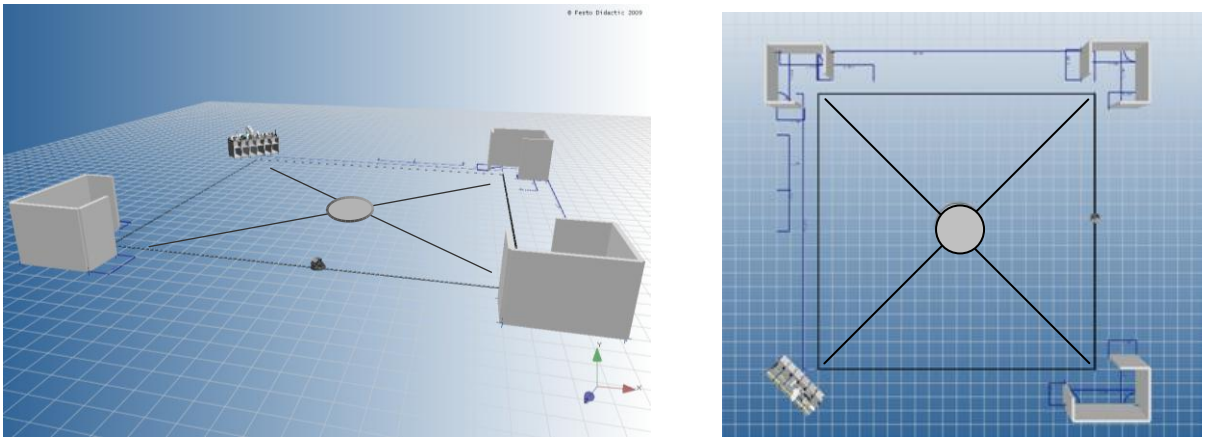

Figura 3.1a – Layout de Produção (esboço virtual)
[imagem do programa Lab Creator – FESTO]

- A construção dos pontos correspondentes às estações será feita através de totens coloridos.

- Ponderando a importância dos pontos de parada, teremos o modelo de layout ilustrado nas Fig. 3.1.b e 3.1.c.

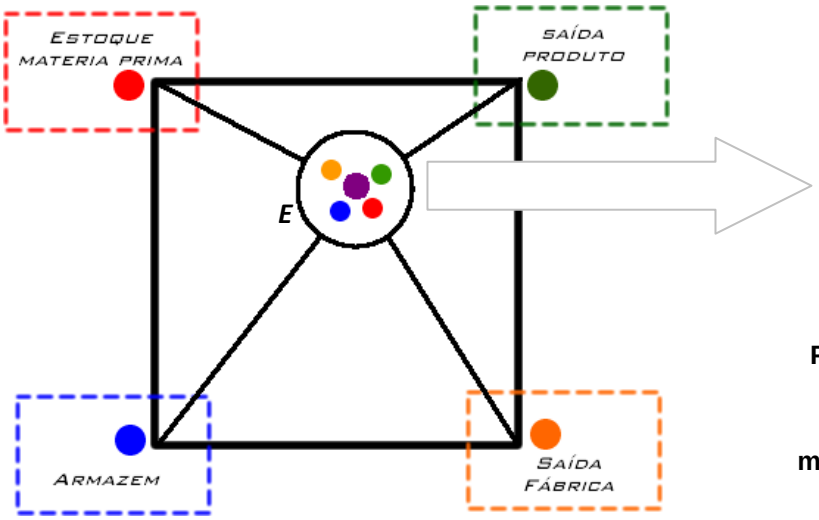

Figura 3.1b – Desenho da perspectiva superior do layout

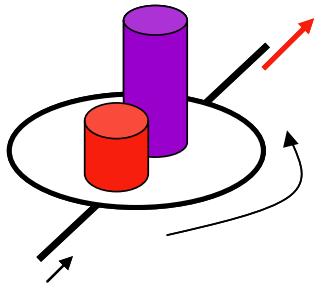

Figura 3.1c – Detalhe do ponto de estacionamento do VT com indicação de 2 rotas. As setas mostram a possível trajetória do VT ao passar pelo local

Construído o layout a tarefa será realizada da seguinte maneira:

- Os VTs ficaram no ponto de estacionamento aguardando a chamada do sistema. Após a rotina de trabalho do SPF, identificando a necessidade de transporte de seus produtos acabados ou de carregamento de matéria-prima um usuário (o próprio SPF ou um operador humano) dispara uma mensagem através da internet para o sistema de integração abrindo o chamado. O Sistema processa a mensagem e envia a requisição para os VTs, escolhendo o robô que estiver melhor disponível (no caso, livre) para realizar o trabalho. O VT é acionado e se desloca para realizar a tarefa. Ao chegar ao destino ele envia uma mensagem ao sistema pedindo autorização para partir para a

próxima localidade (indicando que já foi carregado e necessita partir). Após realizar todo o trajeto, volta para seu Estacionamento e fica aguardando nova tarefa.

Feito isto, é dado baixa no sistema da requisição liberando o VT e reinicia o processo.

3.1.2 Definições

Foram definidos os seguintes termos para a rotina de trabalho:

- Localidades – lugar pré-definido para qual o VT deve se descolar para realizar uma tarefa.
- Rota – caminho entre localidades (origem e destino) realizado pelo VT no deslocamento. É o percurso pré-definido que ele deve seguir.
- Tarefa – par origem / destino de localidades designados para um trabalho de transporte do VT. É um tipo de operação.
- Operação – tipo de serviço disponível no sistema de integração. Ele terá operações de consulta ao VT e/ou Base de Dados, realização de tarefa de transporte e edição de informações na base de dados.
- Usuário – é quem requisita alguma coisa no Sistema de Integração. Pode ser tanto uma pessoa quanto uma máquina, sendo indiferente ao sistema sua natureza.
- Operador – usuário humano do Sistema de Integração
- Servidor – local de armazenamento do Sistema de Integração e Base de Dados.

3.1.3 Sistema de Integração

Será composto por uma Base de Dados com função administrativa e operacional, enviando dados relevantes ao VT e aos seus Usuários quanto requisitados.

O usuário, poderá realizar uma operação no sistema podendo ser uma consulta/edição na Base de Dados ou requisitar uma tarefa ao VT (de controle manual ou autômata).

O Sistema, ao receber um chamado de tarefa autômata será capaz de identificar VTs aptos para a realização do trabalho através de uma rotina de otimização dos veículos.

As chamadas de tarefas poderão ser feitas manualmente pelo operador ou automaticamente, através de sensoriamento do SPF. Para esse trabalho, não será desenvolvido o módulo de interação de sensoriamento para realização das chamadas. Como será implementada somente a tarefa de transporte manual, não será desenvolvida as condições de disparos de funções com os devidos parâmetros para a tarefa através de computador remoto, simulando o mesmo efeito.

Qualquer problema para concluir a operação fará com que uma mensagem seja disparada a um Operador que, dependendo da situação, poderá agir da melhor maneira possível para resolver o evento.

3.2 Modelo Conceitual

Como dito, este trabalho objetiva a integração entre um sistema fabril (SPF) e Veículos de Transporte Autônomos Teleoperados (ROBOTINO) - (mais informações sobre os sistemas, consultar anexo A). Iremos definir agora como serão as bases desse sistema que integrará as operações e o que ele deverá fazer para cumprir sua tarefa (Fig. 3.2).

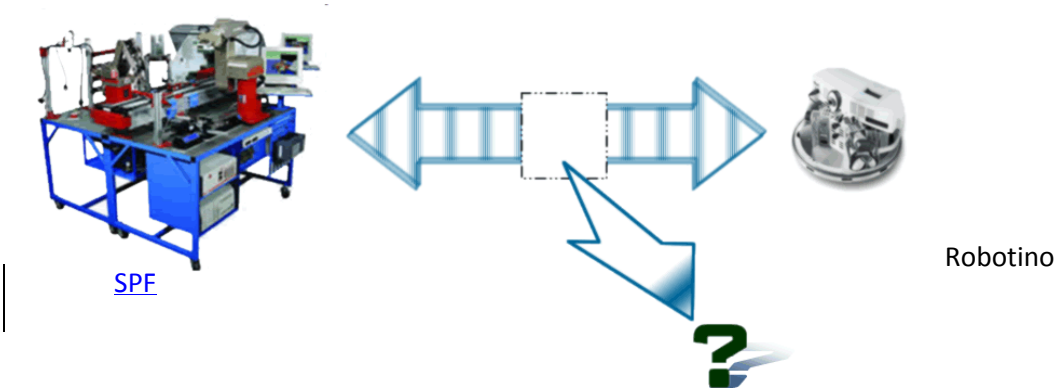

Figura 3.2 - Ilustração dos Sistemas que pertencerão a solução final do projeto

Observa-se no estudo da teoria sobre SPs e analisando o contexto mundial de produção fabril pelo qual estamos passando que é primordial que os sistemas sejam integrados de maneira que, além de não perderem em nenhum aspecto suas características principais e agregarem as suas funções ao ciclo fabril, devem permitir flexibilidade para que o sistema possa se encaixar dinamicamente às necessidades daquela indústria, seja pela mudança da quantidade do que se é produzido, pela adequação a novos produtos, nova distribuição do layout fabril ou outros percalços.

É importante para solução dos nossos problemas desenvolver um projeto sobre uma plataforma de fácil acesso e operabilidade. Pretende-se que sistemas completamente diferentes conversem entre si de maneira precisa e pensar em mudanças no maquinário para ajustar tecnologias de fabricantes diferentes costuma ser uma solução ou inviável ou custosa demais para a realidade atual.

Outro ponto importante é a consideração do elemento humano em todo o processo, e para isso podemos identificar diversas vantagens, mas também alguns problemas. Como visto no capítulo 2, considerar a atuação das pessoas na dinâmica da produção pode agregar muito para a otimização de toda a cadeia. Um operador experiente, com visão dos processos

executados e bem treinado saberá interpretar e ajustar os parâmetros da linha de produção de maneira vantajosa para o todo. Sua capacidade de processar as diversas variáveis dinâmicas envolvidas no processo produtivo, somado a experiência que se tem com a linha possibilita que as reações de tomada de decisão para o controle do sistema sejam, em muitos casos, insubstituíveis. Em contra-partida, para considerar pessoas envolvidas no processo devemos pensar nas condições de trabalho e quanto as afetarão na execução da tarefa. Um operador humano não comporta trabalhar sozinho em turnos seguidos de uma fábrica como uma máquina faria, ou mesmo muitas vezes não pode estar fisicamente no local da produção. Condições inóspitas, simples necessidade de estar em outro local, falta de algum operador por qualquer motivo são fatores que devem ser considerados e contornados para uma solução eficaz.

Nesse contexto, a internet passa a ser uma boa base de operação para o nosso sistema e permite o trabalho através do conceito de nuvem (cloud computing).

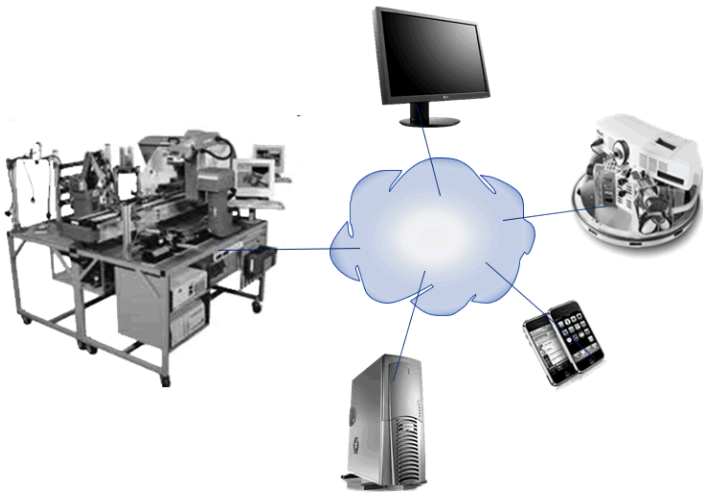

Figura 3.3 - Conceito de nuvem (cloud computing)

Este refere-se à utilização da memória e das capacidades de armazenamento e cálculo de computadores e servidores compartilhados e interligados por meio da Internet.

O armazenamento de dados é feito em servidores que poderão ser acessados de qualquer lugar do mundo, a qualquer hora, não havendo necessidade de instalação de programas, serviços ou de armazenar dados. O acesso a programas, serviços e arquivos é remoto, através da Internet - daí a alusão à nuvem.

Assim, a partir de qualquer computador e em qualquer lugar, pode-se ter acesso a informações num sistema único, independente de plataforma, tendo como requisito mínimo é um computador compatível com os recursos disponíveis na Internet (figura 3.3).

Existem quatro tipos de tipologia para computação em nuvem. A saber: [1]

- **IaaS** - *Infrastructure as a Service* ou **Infra-estrutura como Serviço**: quando se utiliza uma porcentagem de um servidor, geralmente com configuração que se adeque à sua necessidade.

- **PaaS** - *Plataform as a Service* ou **Plataforma como Serviço**: utilizando-se apenas uma plataforma como um banco de dados, um web-service, etc. (p.ex.: Windows Azure).

- **DaaS** - *Development as a Service* ou **Desenvolvimento como Serviço**: as ferramentas de desenvolvimento tomam forma no *cloud computing* como ferramentas compartilhadas, ferramentas de desenvolvimento *web-based* e serviços baseados em mashup (como com uso de APIs).

- **SaaS** - *Software as a Service* ou **Software como Serviço**: uso de um software em regime de utilização web (p.ex.: Google Docs , Microsoft Sharepoint Online).

[1] Mais referências em http://www.cloudbus.org/~raj/papers/hpcc2008_keynote_cloudcomputing.pdf - paper do Departamento de Computação e Engenharia de Software Universidade de Melbourne - AUS

Para esse trabalho, usaremos uma mistura dessas arquiteturas com o uso da Paas, SaaS, numa solução que utiliza-se de uma plataforma com softwares como serviço.

3.3 Arquitetura de desenvolvimento

Tendo a internet como base para a solução de integração o próximo passo é investigar e definir que tipo de ferramenta será desenvolvida para o processo.

Partindo do princípio que estaremos operando com elementos que possam ter linguagens e arquiteturas completamente diferentes deve-se procurar por algum tipo de serviço que consiga transitar por diferentes "idiomas" de maneira organizada, comunicando-se assim de maneira eficiente entre máquinas e operadores.

Apresenta essa característica os Web Services (WS) que tem como premissa de utilização (Fig. 3.4):

1. Um cliente, que pertence a determinado aplicativo (por cliente entende-se homem, máquina/programa) faz seu pedido ao Web Service, através da internet;

2. O Web Service interpreta o pedido e busca a resposta no Servidor (realizando uma ação em outro aplicativo, invocando um banco de dados, etc);

3. A resposta é decodificada novamente e enviada ao usuário que "lê" a mensagem.

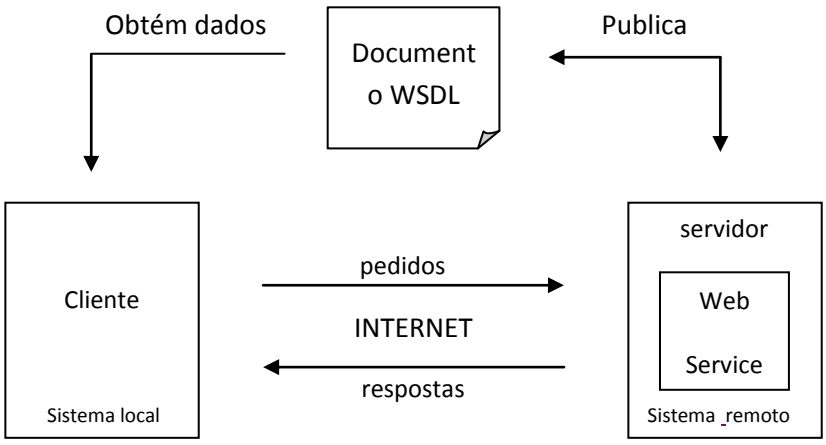

Figura 3.4 - Adaptado de A short introduction to Web Services - Chapter 1. Key Concepts

Neste contexto, a arquitetura do Web Service será definida em quatro partes:

- Processos: aos quais o Web Service está integrado e pode realizar ações.
- Descrições: base do conceito de um Web Service ser auto-descritivo. Ele pode lhe fornecer os dados de quais operações ele realiza e como invocá-las. Esse procedimento é organizado através da Web Service Descripition Language (WSDL).
- Invocações: são as mensagens passadas para um Web Service. O protocolo SOAP (Simple Object Access Protocol) especifica como deve ser o formato dos pedidos ao servidor e como o servidor deve responder ao chamado.
- Transporte: meio pelo qual os pedidos são transmitidos. O protocolo usado basicamente é o HTTP (Hypertext Transfer Protocol), o mesmo usado para acessar uma página da internet normalmente.

O Web Service é endereçado praticamente como uma página da internet, através de uma URI (Uniform Resource Identifiers), semelhante ao uso e termo URL (Uniform Resource Locator). Porém quem faz esse acesso é um programa, diferente de quando acessamos uma

página da internet onde se digita o endereço na barra do navegador. Exemplo de URI para acesso ao Web Service:

http://webservices.mysite.com/serviços/poli/Robotino

WSDL – Web Service Description Language

O cliente sabe como chamar um Web Service e quais métodos usar através de um documento. É por ele também que o Web Service identifica como processar a informação recebida. Esse documento é o WSDL, baseado em linguagem XML.

Um Web Service deve, portanto, definir todas as suas interfaces, operações, esquemas de codificação, entre outros neste documento.

Tão logo o cliente tenha acesso à descrição do serviço a ser utilizado, a implementação do Web Service pode ser feita em qualquer linguagem de programação (por isso a versatilidade da ferramenta).

No caso, a plataforma de desenvolvimento usada será a ASP.NET *****.

Basicamente, quando o cliente deseja enviar uma mensagem para um determinado WS, ele obtém a descrição do serviço (através da localização do respectivo documento WSDL), e em seguida constrói a mensagem, passando os tipos de dados corretos (parâmetros, etc) de acordo com a definição encontrada no documento.

Após, a mensagem é enviada para o endereço onde o serviço está localizado, a fim de que possa ser processada. O WS, quando recebe esta mensagem valida-a conforme as informações contidas no documento WSDL. A partir de então, o serviço remoto sabe como tratar a mensagem, sabe como processá-la (possivelmente enviando-a para outro programa) e como montar a resposta ao cliente.

ASP.NET

Plataforma da Microsoft para o desenvolvimento de aplicações Web, sucessora da tecnologia ASP (**Active Server Pages** – estrutura de programação/framework). É um componente do IIS (Internet Information Service – servidor web criado pela Microsoft) que permite através de uma linguagem de programação integrada na .NET Framework criar páginas dinâmicas. As aplicações para essa plataforma podem ser escritas em várias linguagens, como C# e Visual Basic .NET.

Embora se possa desenvolver aplicações ASP.NET utilizando somente o notepad e o compilador .NET, o ambiente de desenvolvimento mais comum das aplicações ASP.NET é o Visual Studio .NET, já que possui algumas características que facilitam o trabalho do programador, como os componentes visuais para criação de formulários de páginas Web.

Uma aplicação para web desenvolvida em ASP.NET pode reutilizar código de qualquer outro projeto escrito para a plataforma .NET, mesmo que em linguagem diferente. Uma página ASP.NET escrita em VB.NET pode chamar componentes escritos em C# ou Web Services escritos em C++, por exemplo. Ao contrário da tecnologia ASP, as aplicações ASP.NET são compiladas antes da execução, trazendo sensível ganho de desempenho.

As aplicações Web ASP.NET necessitam do Framework .NET e do servidor IIS para executar, pelo menos na plataforma Windows.

Sua utilização nesse trabalho facilitará a correspondência de informações entre o servidor (VB, SQL), o aplicativo web (HTTP) e a arquitetura de programação do Robotino (C#).

Usaremos o Framework ASP.NET na sua versão 4 nessa aplicação.

SOAP

Como dito, Web Services são identificados por uma URI (*Unique Resource Identifier*), e são descritos e definidos usando XML.

Todo esse processo é feito através de trocas de mensagens entre *cliente* ↔*Web Service*, *Web Service* ↔ *aplicativo*, etc.

O SOAP (Simple Object Access Protocol) será o protocolo padrão para a troca de mensagens entre aplicações e o Web Service. Ele é construído com base em XML e HTTP.

SOAP é projetado para invocar aplicações remotas através de RPC (*Remote Procedure Calls*) ou trocas de mensagens, em um ambiente independente de plataforma e linguagem de programação.

Uma mensagem SOAP consiste basicamente dos seguintes elementos (Fig. 3.5a):

- **Envelope**: Toda mensagem SOAP deve contê-lo. É o elemento raiz do documento XML. O *Envelope* pode conter declarações de namespaces e também atributos adicionais como o que define o estilo de codificação (encoding style). Um "encoding style" define como os dados são representados no documento XML.

- **Header:** É um cabeçalho opcional. Ele carrega informações adicionais, como por exemplo, se a mensagem deve ser processada por um determinado nó intermediário (É importante lembrar que, ao trafegar pela rede, a mensagem normalmente passa por diversos pontos intermediários, até alcançar o destino final). Quando utilizado, o *Header* deve ser o primeiro elemento do *Envelope*.

- **Body:** Este elemento é obrigatório e contém o *payload*, ou a informação a ser transportada para o seu destino final. O elemento *Body* pode conter um elemento opcional *Fault*, usado para carregar mensagens de status e erros retornadas pelos "nós" ao processarem a mensagem.

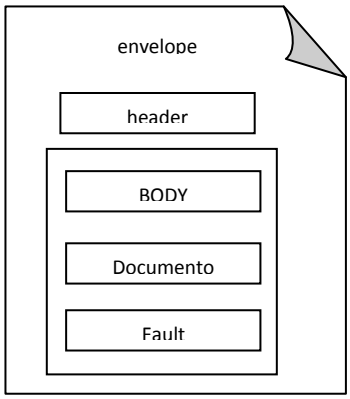

Figura 3.5 - Esquema de organização estrutural de uma mensagem SOAP

Portanto, adotou-se a seguinte especificação SOAP definida pela W3C - World Wide Web Consortium[2]:

- A URI do objeto alvo;
- O nome do método;
- Os parâmetros do método (requisição ou resposta);
- Uma assinatura do método opcional;
- Um cabeçalho (header) opcional.

3.4 Forma de Implementação

Seguindo o padrão MVC (model – view – controller) de arquitetura de softwares, temos a possibilidade de estruturar o sistema de maneira que ele cumpra, dentro do possível, os preceitos de "tão rígido quanto possível e tão flexível quanto necessário" (MASIP, 1988). O padrão MVC permite que se estabeleçam claramente funções dentro do projeto e qualquer

[2] *O **World Wide Web Consortium** é um consórcio de empresas de tecnologia feito para levar a Web ao seu potencial máximo, por meio do desenvolvimento de protocolos comuns e fóruns abertos que promovem sua evolução e asseguram a sua interoperabilidade. O W3C desenvolve padrões para a criação e a interpretação dos conteúdos para a Web. (site: http://www.w3.org/).*

alteração posterior não implica em (ou diminui) problemas de reformulação ou alteração dos dados. Com o aumento da complexidade das aplicações e funções desenvolvidas entre sistemas distintos, torna-se relevante a separação entre os dados e a apresentação das aplicações e essa reorganização que, freqüentemente, se faz necessária.

Divide-se o WS em classes que administram uma função principal dentro do serviço e que, por estarem organizadas em modelos de mensagens "universais", permite a comunicação entre os diferentes tipos de usuários/elementos.

O padrão resolve o problema de flexibilidade por meio da separação das tarefas de acesso aos dados e lógica de negócio, lógica de apresentação e de interação com o utilizador, introduzindo um componente entre os dois, o controlador (Fig. 3.6 e 3.7).

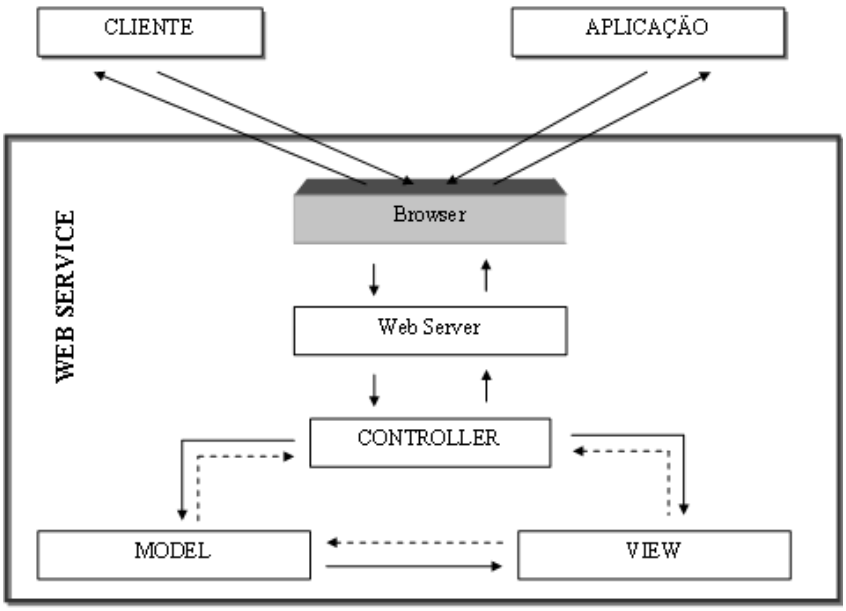

Figura 3.6 – Esquema da arquitetura do projeto + usuários

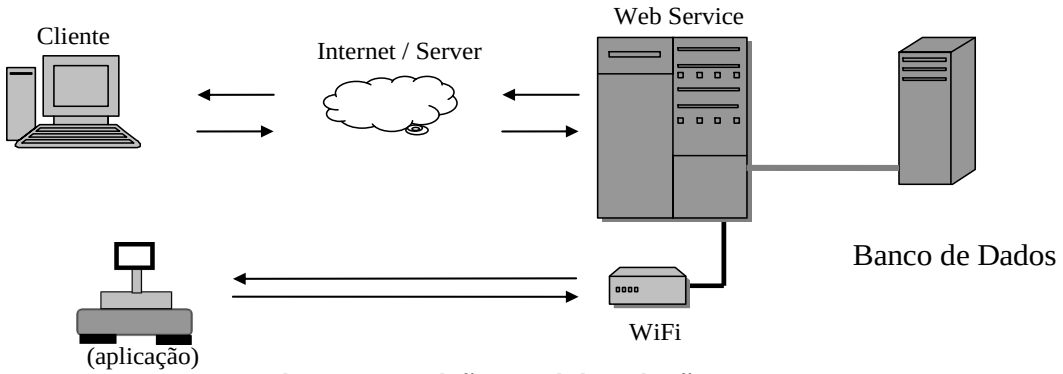

Figura 3.7 – Visão geral da solução

Para isto, serão consideradas as seguintes classes MVC:

- Model - domínio específico da aplicação que será operada pelo WS / lógica de negócios / acesso a dados. A classe MODEL (Modelagem) é usada para definir e gerenciar o domínio da informação e notificar observadores sobre mudanças nos dados. Ela é uma representação detalhada da informação que a aplicação opera. A lógica de negócio adiciona valor semântico aos dados, e quando há mudança de estado o modelo notifica seus observadores. A forma como o dado é armazenado ou acessado não é de interesse do MVC, assume-se que é de responsabilidade da modelagem.

- View - renderiza a informação para, por exemplo, uma interação com o usuário. Apresenta o modelo da interface num formato adequado ao utilizador, na saída de dados, e diferentes visões podem existir para um mesmo modelo, para diferentes propósitos.

- Controller - interação entre cliente e aplicação, modifica Model/fluxo de controle lógico de negócios. Recebe a entrada de dados e inicia a resposta ao usuário ao invocar objetos da Modelagem ou direcionando as informações para as outras partes do sistema, devolvendo a resposta de maneira mais apropriada. É responsável pela validação e filtragem da entrada de dados.

Cada camada do MVC interage de alguma forma com outros componentes do sistema produtivo:

- A camada Model está diretamente relacionada com a Base de Dados do sistema;
- A camada View está diretamente relacionada com o operador, fazendo a interface das soluções com os todos os tipos de usuários.
- A camada Controller está relacionada com os sistemas integrados (nesse trabalho, principalmente o Robotino – objeto de controle, e em segunda instância ao SPF) levando e trazendo as requisições necessárias e enviando as informações a Base de Dados.

Nos capítulos 4 será exposto o desenvolvimento da camada MODEL através do desenvolvimento da Base de Dados. No capítulo 5 encontra-se como será feita a interface com os usuários, definindo assim a classe VIEW do WS. Finalizando nos capítulos 6 teremos: a lógica de negócios que irá administrar todo o fluxo de informação e a interação com os VTs.

4.

Base de Dados

Um sistema inteligente necessita de uma forma para armazenar os seus dados para formação de um histórico. Assim, tudo que acontece com ele pode ser analisado para controle da sua dinâmica.

No SPF desse trabalho, além de acesso a um histórico, os usuários necessitam carregar informações referentes à linha de produção não só para análise, mas para poderem interagir com o processo e dar ordens de comando para o SPAD. Identifica-se assim a necessidade de construção de uma Base de Dados (BD) que possa servir de "fonte de informação" para todas as ações do sistema.

A BD nesse sistema compõe parte da etapa de Modelagem (MODEL) na arquitetura do Web Service discutida anteriormente. Somada a ela, existe a lógica que irá administrar todas as alterações e consultas realizadas na BD.

Neste capítulo, será apresentado o modelo entidade-relacionamento da BD do Sistema Integrado.

4.1 Construindo a Base de Dados

Para a construção da BD primeiramente, dividiremos o sistema de maneira a identificar seus agentes/entidades principais e quais são os tipos de interação entre eles. Assim, poderá construir-se um Diagrama de Entidades e Relacionamentos que irá compor a BD. Será

apresentado o modelo de cada interação com mais detalhes, com Diagramas de Sequências dos Casos de Uso (UC). Ele servirá na especificação de toda a estrutura que compõe o WS, mostrando a lógica de negócios feitas pelo Controller, a interface com os usuários necessária para o View e com os usuários.

4.1.1 IDENTIFICAÇÃO DE NÚCLEOS AGENTES DA BD

A partir das necessidades levantadas nesse projeto, identificamos a existência de três tipos de agentes para o sistema de integração proposto (Fig. 4.1).

Figura 4.1 – Núcleos agentes no Sistema de Integração

Essas entidades irão fazer diversos tipos de requisições de trabalho entre eles para estabelecer o funcionamento do Sistema de Integração.

4.1.2 INTERAÇÕES ENTRE AGENTES

É estabelecida as seguintes formas de interação entre as entidades do sistema:

A) USUARIO / ROBOTINO

O USUARIO irá se comunicar com os VTs (ROBOTINOS) requisitando / realizando as tarefas de transporte de carga. Fará também consultas sobre a situação dos dispositivos (livres ou ocupados). Na sua implementação total, OS ROBOTINOS, por sua vez, informarão o

USUARIO (OPERADOR, no caso) quando alcançou os pontos de carga/descarga (destinos intermediários da tarefa) e requisita autorização para continuar a operação.

B) ROBOTINO / SERVIDOR

OS ROBOTINOS irão se comunicar para o SERVIDOR os registros das suas situações: quanto à chegada em localidades, se livre ou ocupado e de erros durante a operação. O SERVIDOR fornecerá aos VTs as informações de rotas (caminho entre pares origem/destino das localidades), tarefas (rotas estabelecidas para uma rotina) e localidades (informações dos pontos de estacionamento do VT – estoque, estacionamento do VT, armazém, saída da linha de produção) para realização das tarefas exigidas pelos OPERADORES.

C) SERVIDOR / USUARIO

O SERVIDOR irá se comunicar com o OPERADOR requisitando informações para liberação de acesso ao sistema e registro de tarefas. No caminho inverso, OPERADORES irão se comunicar com o SERVIDOR para acesso a consulta de informações gerais da BD (registro de ações e operadores, informações do sistema como informações de usuários, tarefas, rotas, localidades e erros).

Na Fig. 4.2 apresenta-se uma modelo esquemática das interações previstas.

4.1.3 DESENVOLVENDO MODELOS DE INTERAÇÕES

Definidas as formas de interação principal entre as entidades, é hora de especificar melhor cada modelo. Desenvolve-se o Diagrama de Atividades e esse, validado por meio de redes de Petri, permite:

- A construção dos casos de uso (UC) por Diagramas de Sequências, onde pode-se observar detalhadamente todas as iterações do processo (fazendo parte da construção de todas as camadas do WS);

- A elaboração do Diagrama de Entidades e Relacionamentos para construção da BD.

Nos anexo B, podemos observar modelos em Rede de Petri desenvolvidos para validação dos diversos tipos de interações / ações do sistema de integração aqui apresentadas.

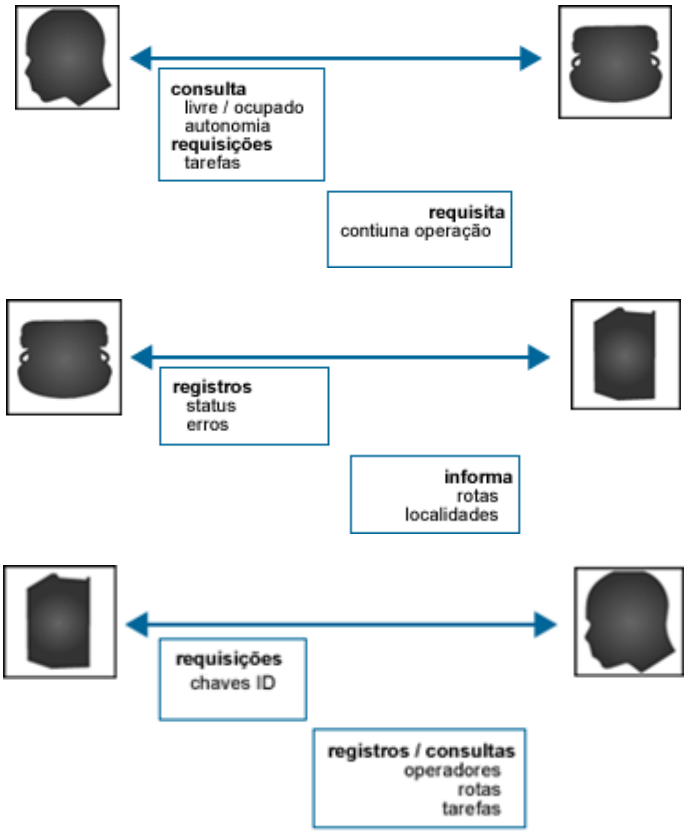

Figura 4.2 – Interação entre agentes do sistema:

(a)usuario / robotino ; (b) robotino / servidor ; (c) servidor / usuario

MODELO DE INTERAÇÕES USUARIO / ROBOTINO

Aqui, são apresentados os fluxos de dados entre USUARIO (automático ou operador humano) e VT (outro tipo de usuário do sistema, chamado ROBOTINO, no caso). Basicamente, o usuário faz as requisições de movimentação ao sistema, que envia ao Robotino, movimentando-o.

Para a tarefa autômata, O USUARIO também faria as requisições de tarefas ao VT por meio do WS. Ele registra no sistema uma operação denominada tarefa, que é composta por pares origem/destino das localidades onde o VT deve operar, marcadas no layout de produção e chamadas de rotas. O WS encaminha a informação para o BD e para o VT mais apropriado para aquela tarefa. Nesse contexto, O ROBOTINO, assim que for completando suas metas, informa o OPERADOR da situação e quando necessário, pede para que ele autorize sua partida para a próxima rota (Fig. 4.3a e 4.3b e 4.3c).

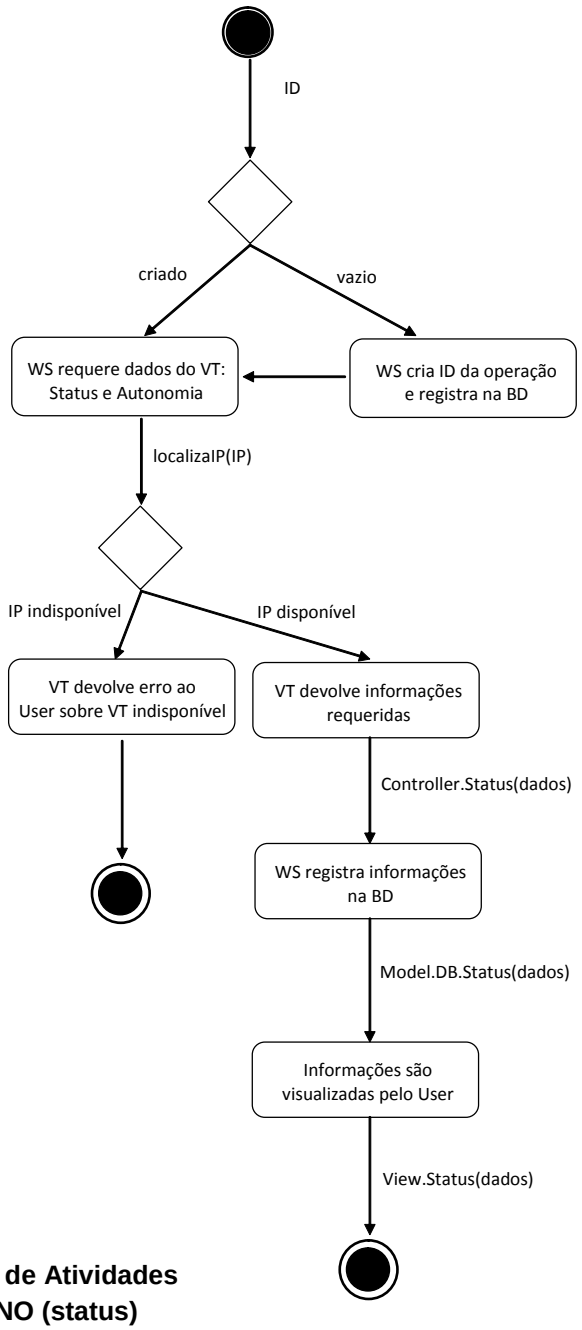

Figura 4.3a – Diagrama de Atividades USUARIO / ROBOTINO (status)

No diagrama pode-se identificar já também o processo de iteração do servidor com o Robotino, na identificação e busca pelas informações através do IP.

Tem-se um padrão nas chamadas tendo sua parte e contra-parte agindo sempre em pares.

Nas atividades definidas em um par, geralmente, observa-se o caminho de ida e volta ocorrendo como ação e reação, então se o operador chama o robotino, este chamará também o operador durante a mesma atividade, com lógica semelhante, como resposta.

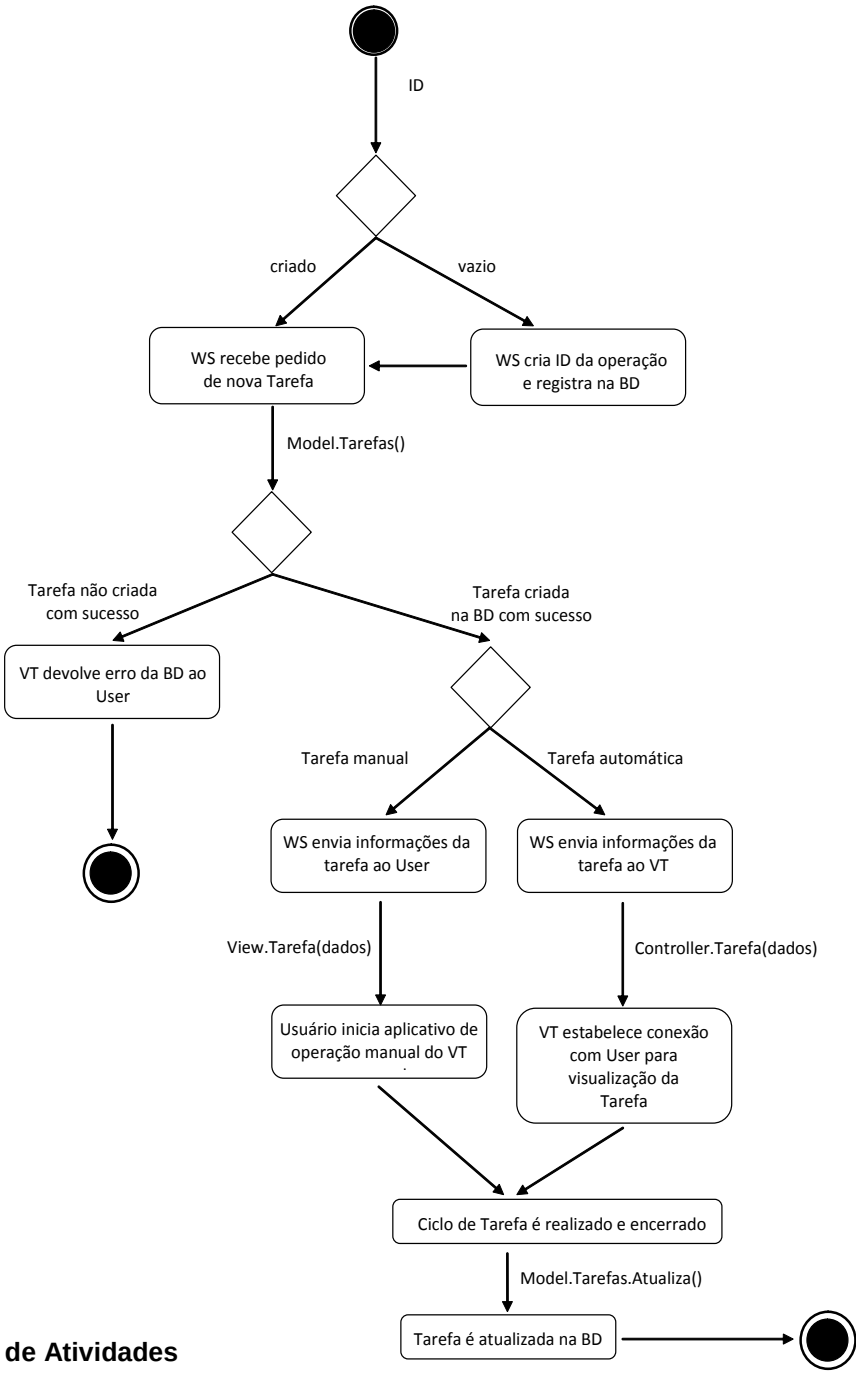

Figura 4.3b – Diagrama de Atividades USUARIO / ROBOTINO (tarefas)

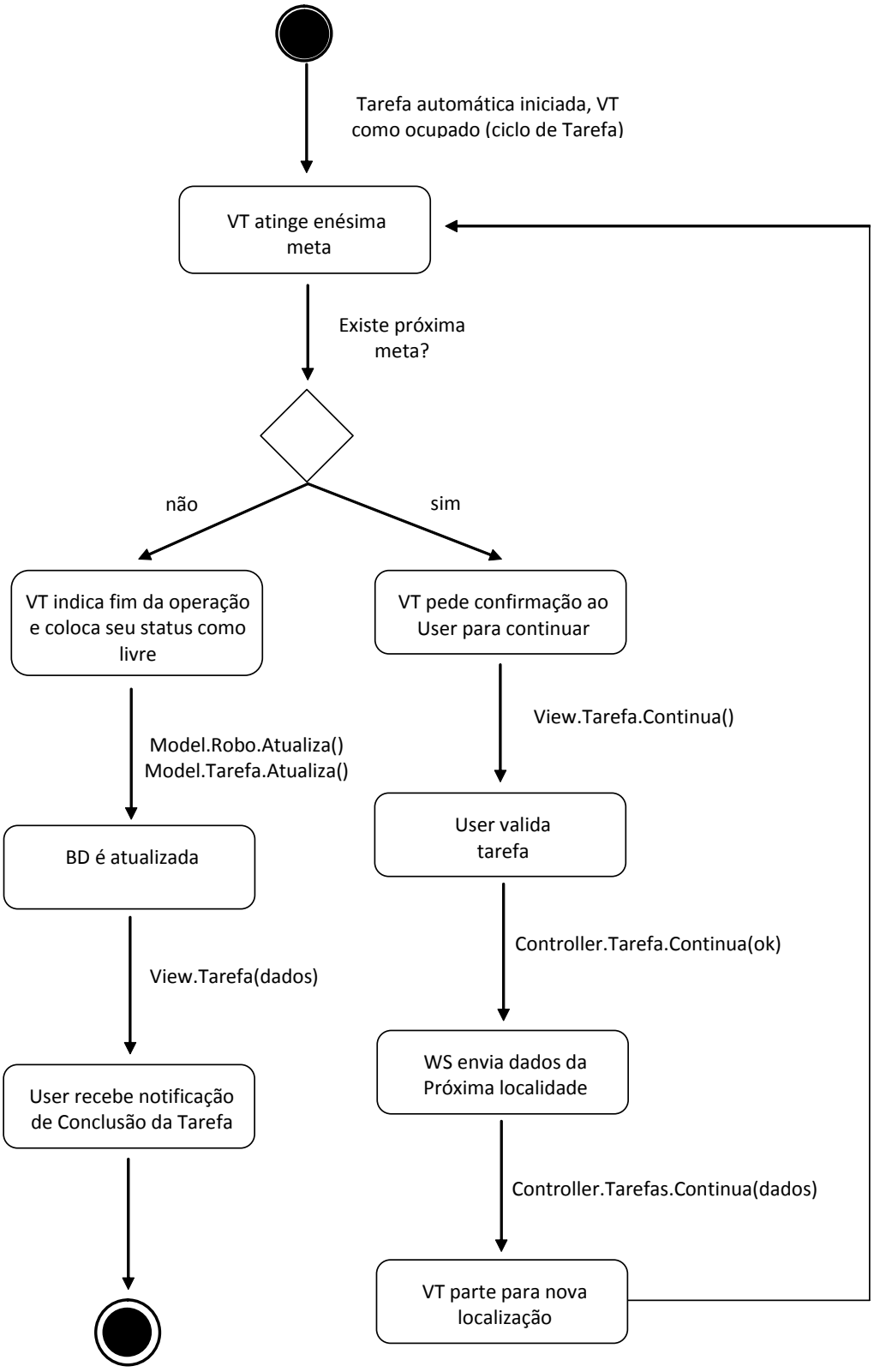

Figura 4.3c – Diagrama de Atividades ROBOTINO / Usuário

MODELO DE INTERAÇÕES ROBOTINO / SERVIDOR

Nesse tipo de interação o VT envia informações sobre o status das operações ou de erro ao SERVIDOR (Base de Dados) dependendo do caso. Essas informações são somente armazenadas ou, além de armazenadas, podem também ser remetidas ao OPERADOR, dependendo da necessidade (exemplo Fig 4.3a). O SERVIDOR também é agente da operação enviando dados da tarefa ao VT, afim de localizá-lo na respectiva ação (contra-parte do diagrama, como ocorreu no modelo anterior).

MODELO DE INTERAÇÕES SERVIDOR / USUARIO

Dos casos possíveis, basicamente, o SERVIDOR irá requisitar do USUARIO informações para registro das informações na base de dados (Fig. 4.3d).

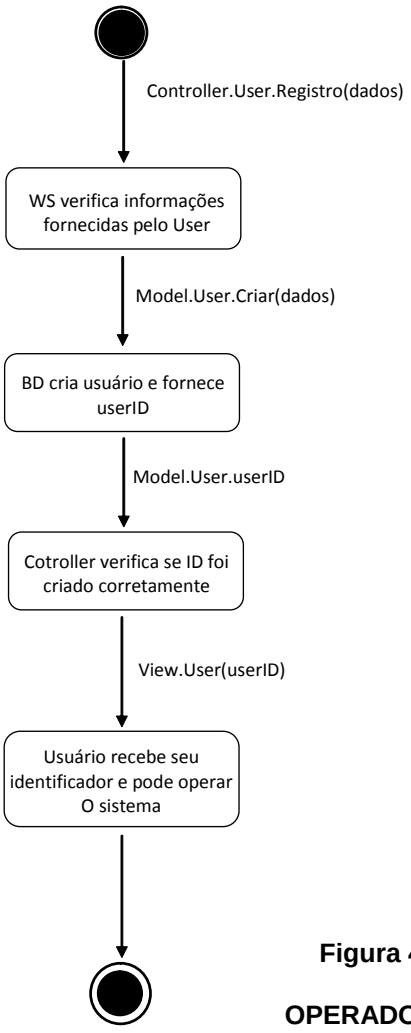

Figura 4.3d – Diagrama de Atividades OPERADOR / SERVIDOR (registro usuário)

4.1.4 Diagrama Entidades - Relacionamentos

Depois de analisado como se dará cada interação, já é possível obter-se o diagrama de entidades e relacionamentos que será de fato como modelo final para gerar a BD relacional adotada no sistema (Fig. 4.4).

Figura 4.4 – Diagrama Entidade Relacionamento da BD do sistema

Entidades da BD

- **USERS –** tabela que conterá as informações dos usuários (operadores) do sistema de integração. Terá como chave primária o atributo userID, que identificará o usuário para o banco de dados. Deverá ser inserido os campo nomeSobrenome, e-mail, dia do registro.

- **OPERAÇÃO –** tabela que conterá os registros de operações realizadas no sistema de integração. Identificada pelo campo opID (mesma função do userID, sendo uma espécie de ticket fornecido pelo sistema identificando aquela ação do usuário). Conterá identificações do VT usado, tipo da operação (manual ou automática), data do registro, status (andamento ou finalizada) e nome da operação.

- **ROTAS –** tabela que contem informações referentes as rotas que um VT deve executar na tarefa.

- **LOCALIDADES –** indica as localidades que cada rota deverá passar.

- **ROBOTS –** mostrará dados do VT armazenado como ID na BD, IP, nome, data do registro e se ocupado ou não.

5.

INTERFACE

Existem basicamente dois tipos de usuários que poderão interagir com o sistema. Uma máquina ou rotina de trabalho pré-programa e um operador humano. Dependendo de qual está utilizando o sistema, existe um tipo de necessidade para as saídas dos dados: em formato de logs ou com telas de interação ou resposta.

Pelos motivos citados nos capítulos iniciais, respeitando as características antropocêntricas do no nosso sistema produtivo, privilegiar-se-á o desenvolvimento de telas para interação do sistema com o operador, facilitando o uso da ferramenta.

Esse conjunto de objetos faz parte da classe VIEW do nosso WS, ponte de integração entre o sistema e os diversos tipos de usuários que dele se utilizam.

O padrão de desenvolvimento da nossa interface será o HTML, com elementos que permitam a integração das páginas com a BD e os VTs, como a estrutura em ASPX .

5.1 ESTRUTURA DE INTERFACE DO SISTEMA

De acordo com as necessidades levantadas para o sistema de integração, é apresentado a seguir o mapa que identifica a estrutura de visualização da camada VIEW do nosso WS (Fig. 5.1a, 5.1b):

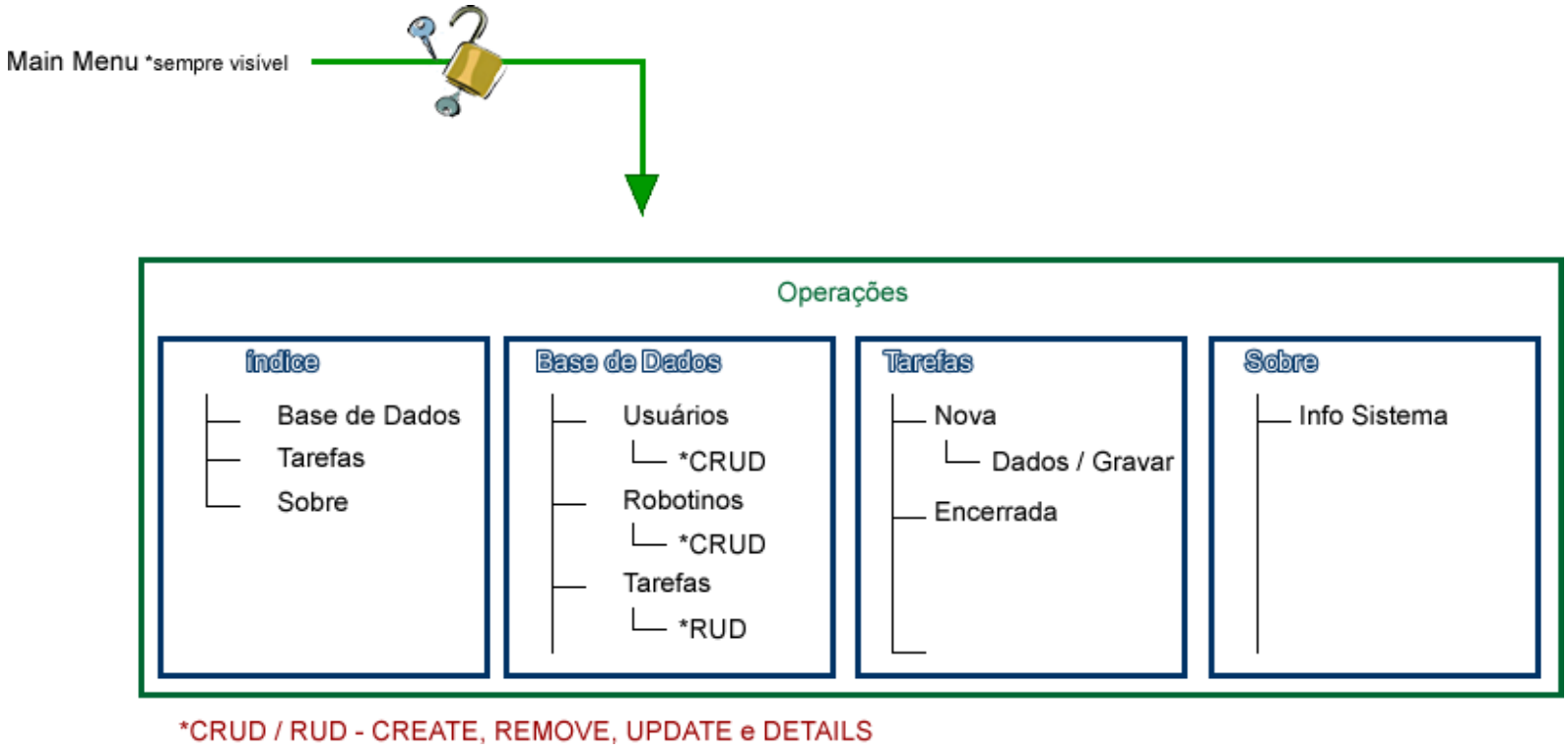

Figura 5.1a – Mapa de telas do WS

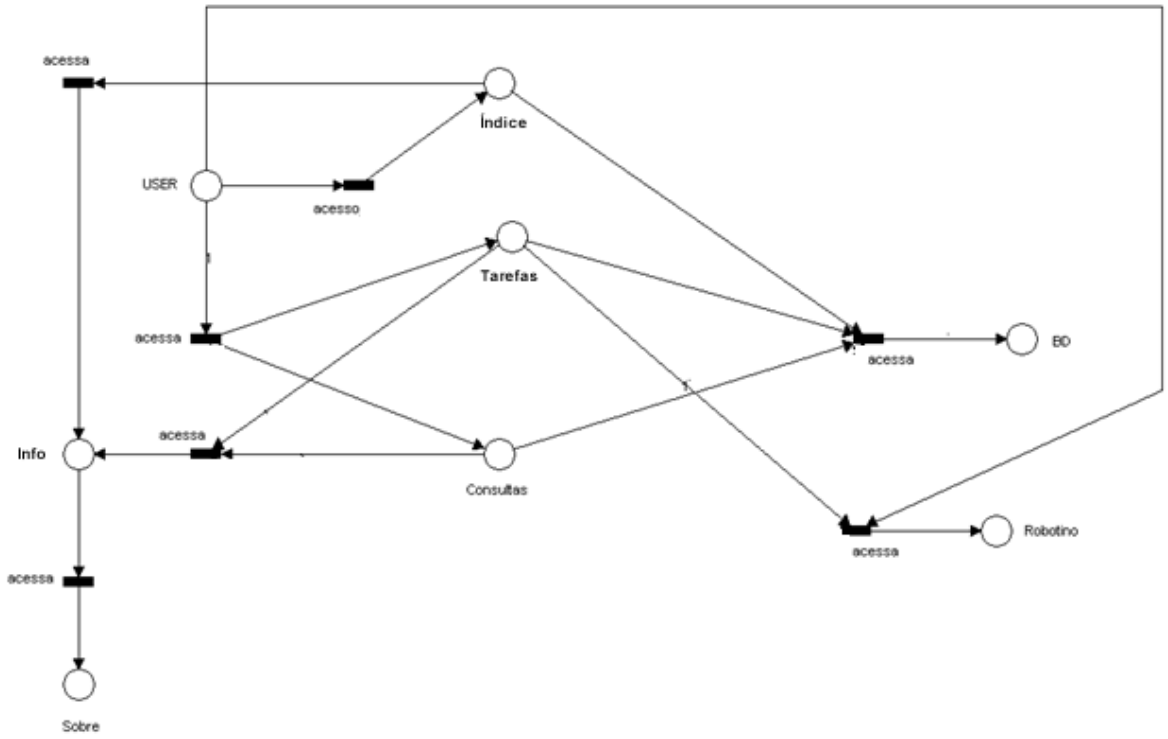

Figura 5.1b – comunicação entre telas / componentes do WS

O WebSite com o serviço de integração é inicializado na tela “HOME”, possibilitando ao usuário acessar qualquer área do site através de um menu fixo superior. Em SOBRE, são exibidas informações sobre o desenvolvimento do sistema, BASE DE DADOS, com a situação das entidades, podendo-se INCLUIR, CONSULTAR, EDITAR e EXCLUIR seus elementos. Em TAREFAS temos a possibilidade de iniciar uma tarefa manual de controle e transporte do VT.

A opção SOBRE informa sobre o desenvolvimento do WS, versão e ano entre outros aspectos.

Definidas as Views principais, existem mais algumas telas de interface de passagem, que auxiliam para melhor a compreensão do sistema.

Há também telas de ERROS que não estão inseridas no mapa anterior, mas que respondem em diversas partes do sistema, dependendo da ação do operador ou de alguma resposta indesejada do servidor.

No ANEXO C encontram-se imagens das telas de interface diagramadas para melhor compreensão.

6.

CONTROLE / LÓGICA DE NEGÓCIOS

A camada restante para construção do WS refere-se a que interpreta as requisições, valida os dados de entrada e encaminha as chamadas para suas respectivas áreas. Essas são funções da camada CONTROLLER do WS e, em linhas gerais, executa as funções de lógica de negócios entre os dados administrados pelo WS.

No sistema de integração proposto, será a CONTROLLER que irá interpretar as ações entre operadores, VTs e BD e administrar todo o fluxo de trabalho. Quando um usuário entrar com algum dado, a camada interpreta as informações e encaminha para a BD cuidar do processamento. A resposta dada pela BD passa pela CONTROLLER que encaminha para a interface responder ao operador aquela ação.

Fazendo um apanhado geral de todas as funções acumuladas pela camada de controle do WS identificam-se as principais diretrizes que, esquematizadas em diagramas de seqüências, fornecem a cadeia de ações em três blocos principais de funções (Fig. 6.1, 6.2, 6.3):

- Registro de operações.
- Requisição de rotinas.
- Verificação de status.

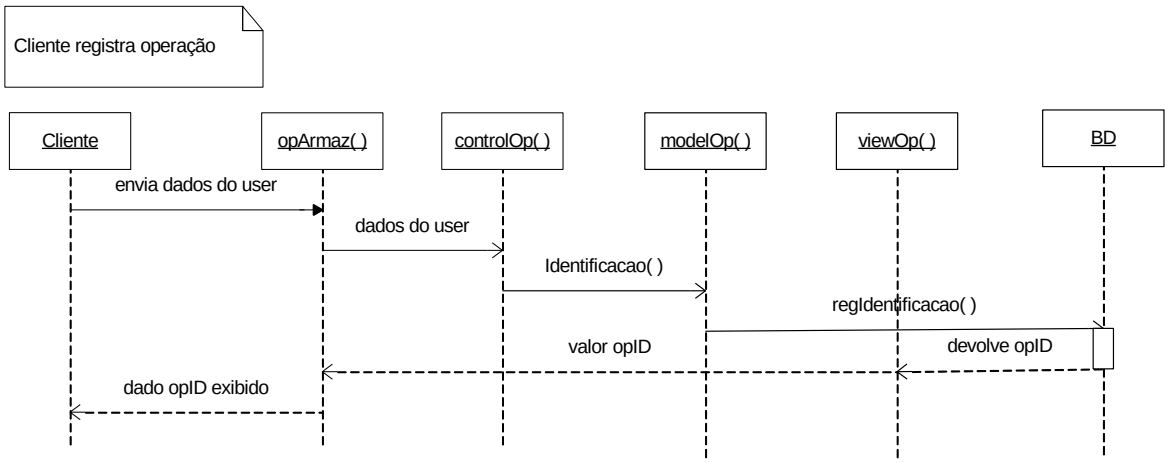

Figura 6.1 – Seqüência: registro de operação

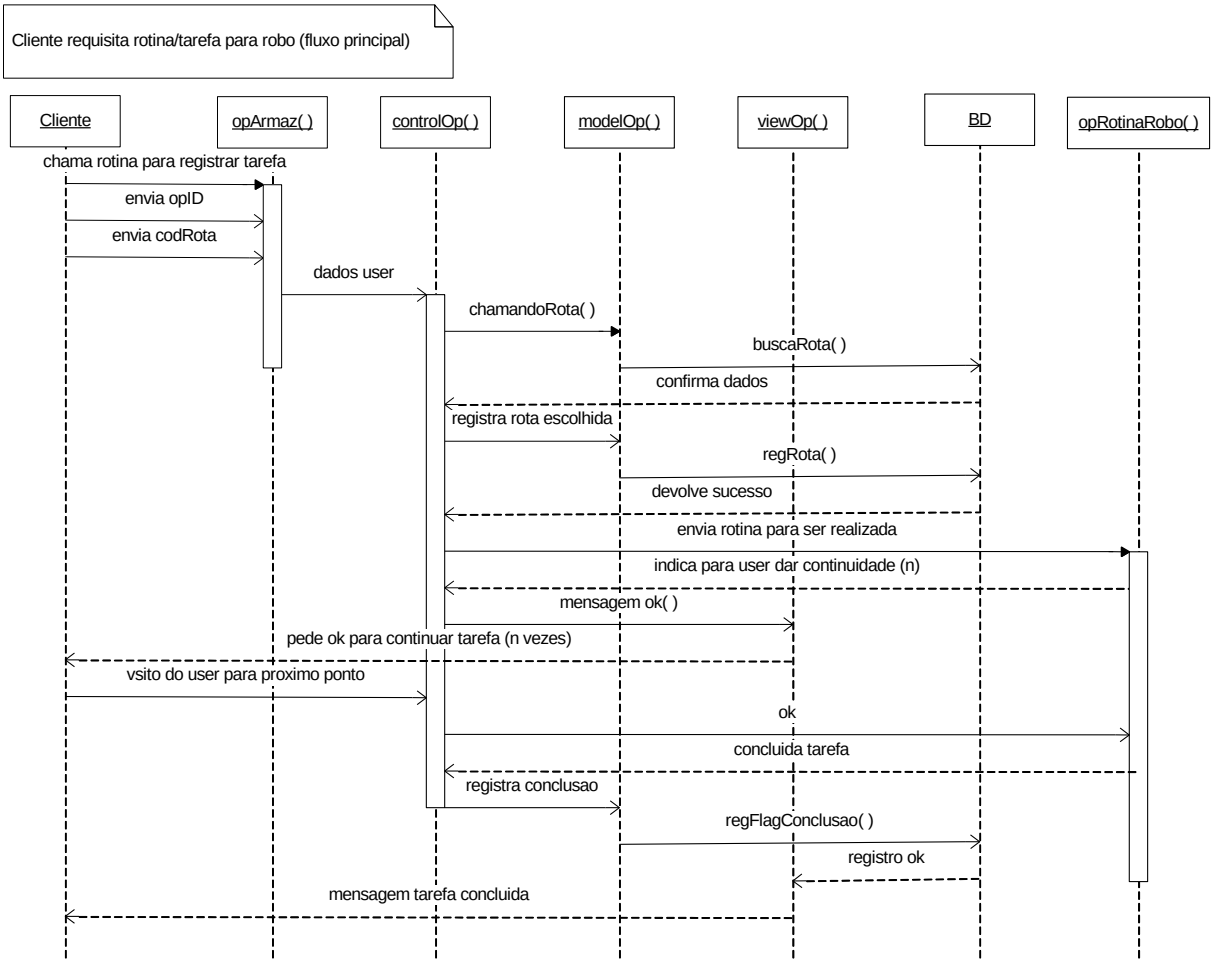

Figura 6.2 – Seqüência: requisição de rotina

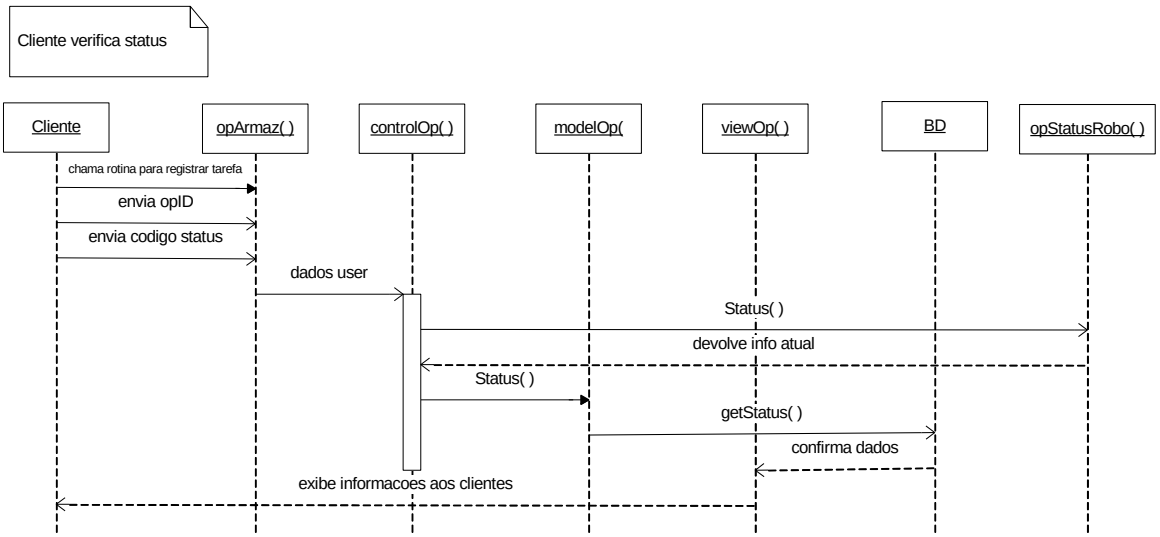

Figura 6.3 – Seqüência: Verifica Status

Assim, podemos diagramar as principais funções propostas da camada CONTROLLER, e posteriormente, em linhas gerais as demais que irão amarrar o sistema e suas funcionalidades (Tablea 6.1).

FUNÇÕES CONTROLLER

Tabela 6.1 – classes e funções principais Controller

CLASSE	METODOS
{Home}	Index()
	Consultas()
	Tarefas()
	About()
{CONSULTAS}	UserConsultas()
	USerCreate()
	UserDetails()
	UserEdit()
	UserDelete()
	VtConsultas()
	VtCreate()
	VtEdit()
	VtDelete()
	TarefasConsultas()
	TarefasEdit()
	TarefasDelete()
{Tarefas}	Main()
	MainClose()

Com esse conjunto de objetos da CONTROLLER, somados às páginas geradas pela VIEW e o mapeamento da BD pela MODEL, podemos desenvolver o WS na linguagem de programação mais apropriada para as características das classes e execuções feitas no site. Em linhas gerais:

- **HTML** – páginas de interface e conexão entre elas.
- **ASPX** – envio e armazenamento de informações fornecidas do e para o HTML. Também nas construções das páginas, introduzindo elementos mais ricos.
- **VB.NET** - lógica de negócios entre tarefas e informações. Acesso as plataformas de trabalho (ROBOTINO)
- **C#** - desenvolvimento das funções dentro da lógica de programação do VT.
- **SQL** – acesso ao banco de dados e tarefas realizadas com ele.

Apresentaremos, no próximo capítulo, a lógica de funcionamento do VT e como ele vai operar.

7.

LÓGICA DE OPERAÇÃO PARA OS VTs

Definido o WS teremos agora que projetar o funcionamento dos VTs, determinando como será realizada, de fato, a rotina de trabalho dos Robotinos (VTs).

Para isso, usaremos as ferramentas padrão de desenvolvimento oferecidas pelos próprios VTs – Robotinos / Festo. Nos modelos do Robotino fornecidos até a data do desenvolvimento desse projeto, nos é disponibilizado o desenvolvimento de algoritmos para programação em C++ / C#, Visual Basic, Java, MatLab (.m), através de diagramas de Blocos de Função (Function Blocks) e de Controles de Processos (Control Flow Process Diagrams)[3].

Para a primeira etapa do projeto, será desenvolvido no Software de programação do Robotino - ROBOVIEW 2.0, toda a programação que o robô executará na sua tarefa através dos Diagramas de Blocos (Fig. 7.1a, 7.1b, 7.2).

No simulador ROBOTINOSIM, serão validadas as rotinas desenvolvidas através de um ambiente virtual que simula o próprio VT em um ambiente de operação que simula o SPF (Fig. 7.3, 7.4).

A implementação final será feita diretamente no Visual Studio. Através da API fornecida pelo fabricante, habilitamos o compilador a aceitar as chamadas e lógicas do Robotino através de uma biblioteca, e será desenvolvida toda a programação do controle manual do VT ali mesmo, gerando um aplicativo executável que será instalado nas máquinas clientes que executarão o WS.

[3] As linguagens requerem API específico para serem compiladas e executadas no robô, em geral.

ROTINA AUTOMÁTICA de TESTES DO ROBOTINO

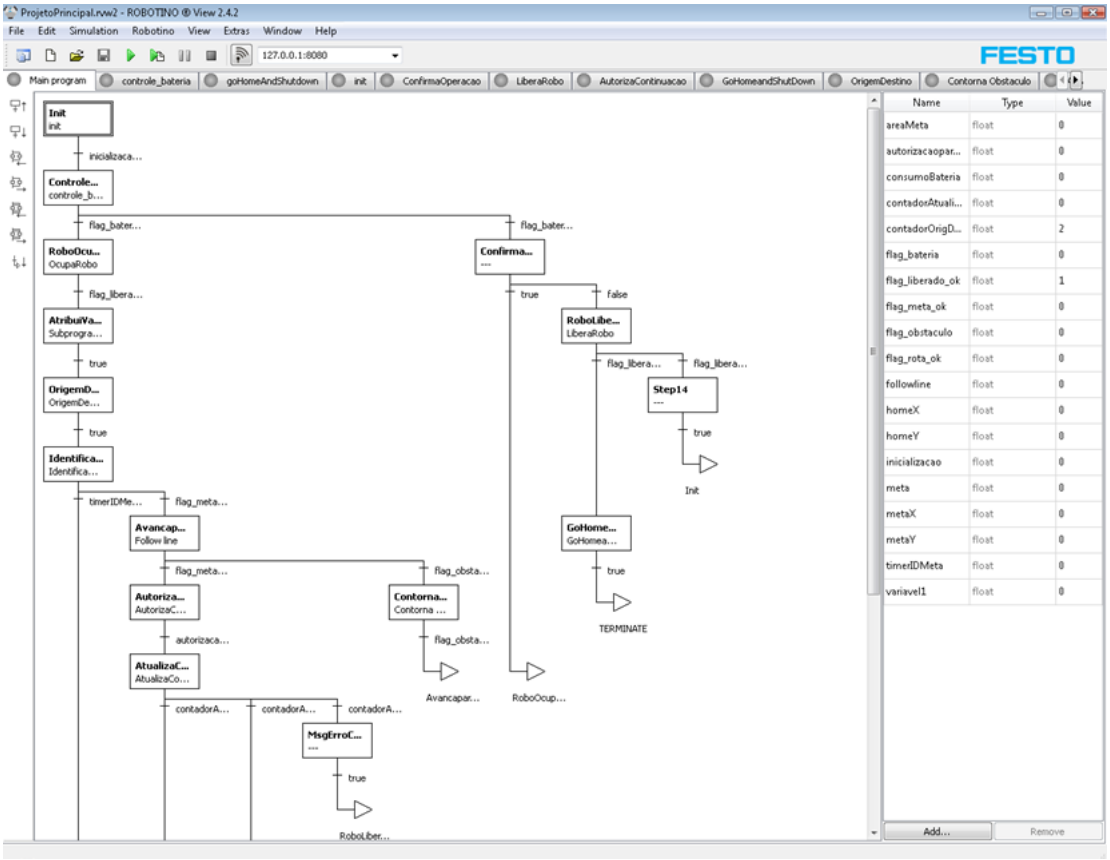

Figura 7.1a – Tela de visualização para programação do ROBOTINOVIEW (algorítmo principal)

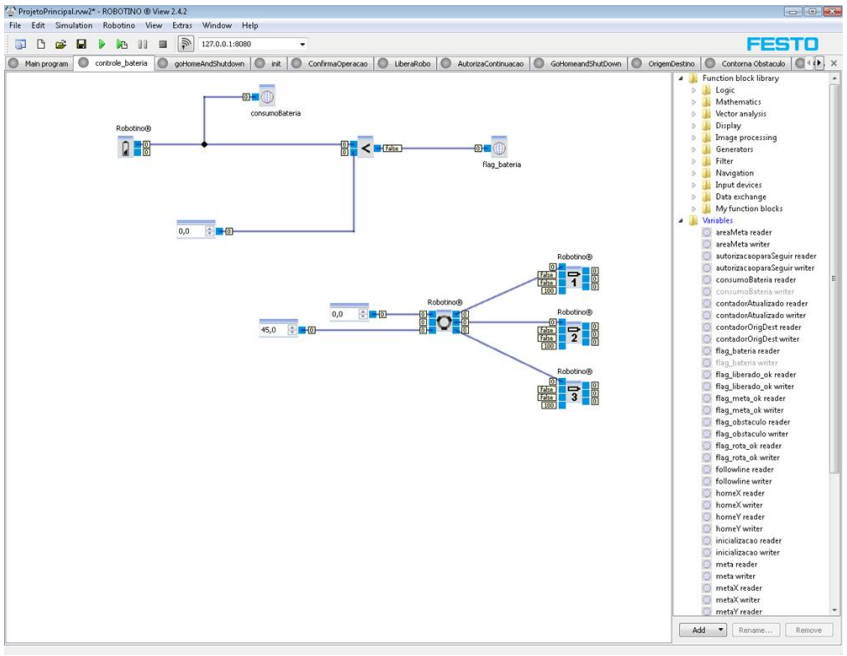

Figura 7.1b – Tela de visualização para programação do ROBOTINOVIEW (blocos de função) – RobotinoView2.0 - FESTO

Main Page | Classes | Files

Class List | Class Hierarchy | Class Members

rec_robotino_com_c Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

rec::robotino::com::AnalogInput	*An analog Input*
rec::robotino::com::Bumper	*Represents a bumper*
rec::robotino::com::Camera	*Represents a camera*
rec::robotino::com::Com	*Represents a communication device*
rec::robotino::com::ComException	***RobotinoException*** *exclusive to* ***Com*** *objects*
rec::robotino::com::ComId	*The id of a* ***Com*** *object*
rec::robotino::com::DigitalInput	*Represents a digital input device*
rec::robotino::com::DigitalOutput	*Represents a digital output device*
rec::robotino::com::DistanceSensor	*Represents an IR distance sensor*
rec::robotino::com::EncoderInput	*Represents the external motor encoder input*
rec::robotino::com::Gripper	*Represents a digital output device*
rec::robotino::com::Info	*Retrieves informational messages from Robotino*
rec::robotino::com::JPGCamera	*Represents a camera*
rec::robotino::com::Motor	*Represents a single motor*
rec::robotino::com::NorthStar	*Represents the* ***NorthStar*** *tracking device*
NorthStarCommand	*Command send to Evolution Robotics NorthStar*
NorthStarReadings	*Data read from Evolution Robotics NorthStar*
rec::robotino::com::Odometry	*Represents Robotino's odoemtry module*
rec::robotino::com::OmniDrive	*Calculates motor velocities for the omni drive*
rec::robotino::com::PowerManagement	*Represents the power management of Robotino*
rec::robotino::com::PowerOutput	*Represents a digital output device*
rec::robotino::com::Relay	*Represents a relay*
rec::robotino::com::RobotinoException	*An extended exception class used in all rec::robotino::com classes*

Figura 7.2 – Lista de classes para programação do Robotino – cd Robotino – Rec_Robotino_Com_C_Class List

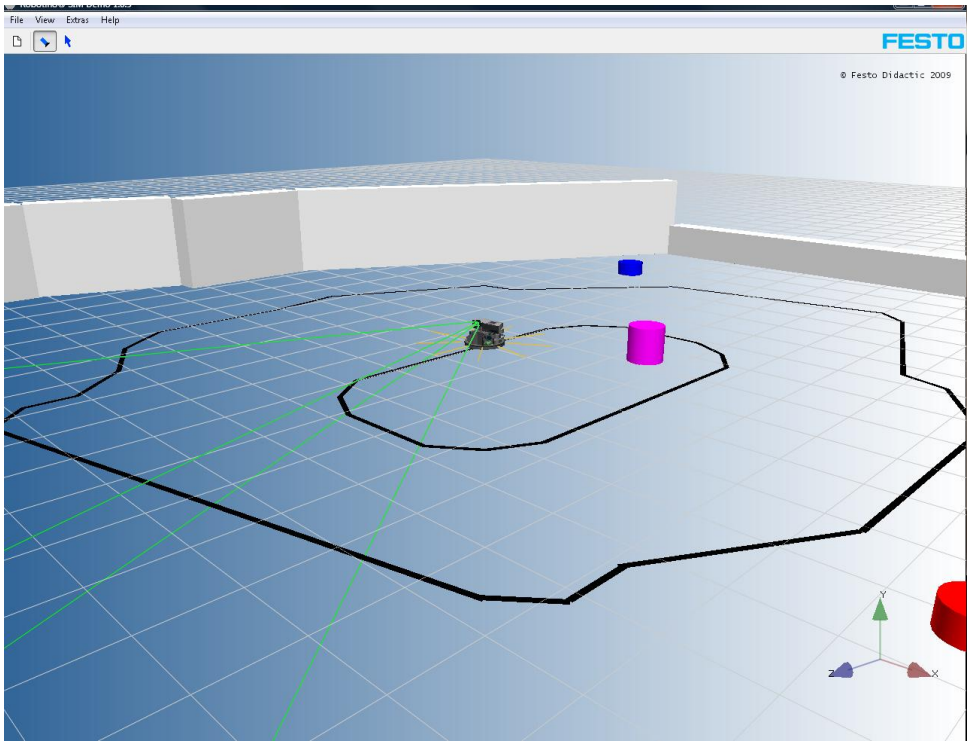

Figura 7.3 – Simulador ROBOTINOSIM – programa RobotinoSim - FESTO

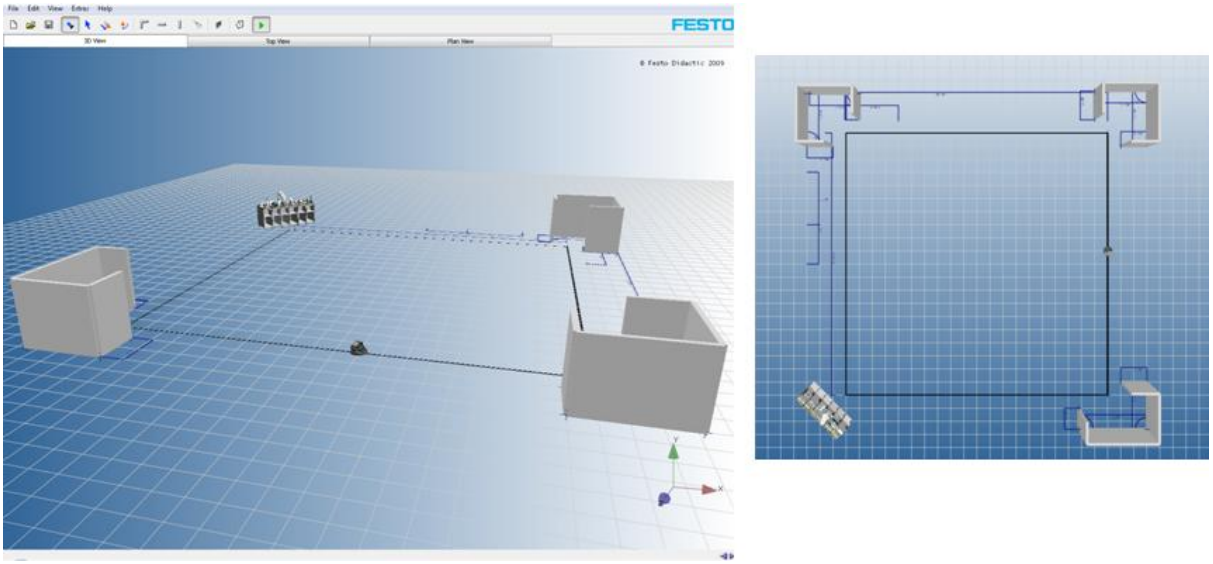

Figura 7.4 – LabView : criação de ambientes virtuais para simulação do Robotino – Programa LabCreator - FESTO

7.1 Desenvolvimento do Programa Principal

7.1.1 ROTINA MANUAL

A Rotina desenvolvida para essa aplicação será de controle manual do veículo de transporte através de cliques do mouse em uma tela de comando. Poder-se-á ter a visão do robô através da sua câmera.

Dentro das características do aplicativo temos também a possibilidade de se estabelecer conexão com qualquer VT desde que fornecido seu IP e um console para visualização de erros e demais mensagens.

Assim, é construído um formulário principal (MainForm) composto pelos seguintes controles:

- **Connect**: recebe dados de IP para conectar-se ao VT através de comando do botão conectar.

- **Console**: envia mensagens de conexão estabelecida, perda ou falha.

- **Drive**: contém os comandos para controle do VT.

- **Câmera**: contém a visão da câmera do VT.

Para realizar essas tarefas, deve ser criada também uma classe Robot, que define as propriedades de ação do robô, que é principal dentro da estrutura da lógica de programação do Robotino. Nela, instanciamos os objetos que pertencem ao Robotino, passamos os parâmetros de velocidade, posição, motores e demais características.

A rotina principal será chamada com o parâmetro de IP do VT vindo do WS.

Cabe ao usuário depois aceitar esse IP e começar a operação.

O não uso do endereço implicará em um registro de tarefas errado na BD.

7.1.2 ROTINA AUTOMÁTICA

Neste caso, deve ser criado o "main Program" no ROBOTINOVIEW contendo as tarefas de gerenciamento das operações do ROBOTINO "*in loco*". Sua estrutura indicará o fluxo de atividades que o robô deve executar tanto para receber as informações de uma tarefa, acioná-la e enviar as devolutivas ao Controller quando necessário.

Abaixo, é apresentada a estrutura do programa principal (Figura 7.5) dividida em conjuntos de atividades. Cada conjunto representa um conjunto de etapas características na rotina de trabalho do Robotino:

- Inicialização do processo
- Confirmação de início/fim de operação e outras trocas de mensagens com usuário
- Mensagens de erro ou Encerramento do processo

- Pré - Execução da operação pelo VT
- Execução da operação pelo VT

A seguir é detalhada a execução da rotina pelo programa seguido da explicação de cada bloco de função. A rotina que otimiza o uso dos VTs pode ser desenvolvida diretamente em C#.

No anexo D encontra-se os diagramas de cada função especificada, da maneira como podem ser projetadas no ROBOTINOVIEW.

7.1.1 ROTINA PRINCIPAL

Inicialização do Processo

A rotina deve começar com os controles de disponibilidade, tempo e de autonomia da operação. É iniciado um contador de tempo para a operação do VT. O segundo bloco é responsável em identificar o estado do VT e dependendo das condições de voltagem do Robotino, avisa ao usuário se ele deseja continuar a operação ou desligá-lo / enviar para recarregar as baterias. Após, deve ser feita a identificação se o VT estará realmente disponível para realizar a operação. Caso ele passe por esses pré-requisitos, é iniciada a operação de transporte, propriamente dita.

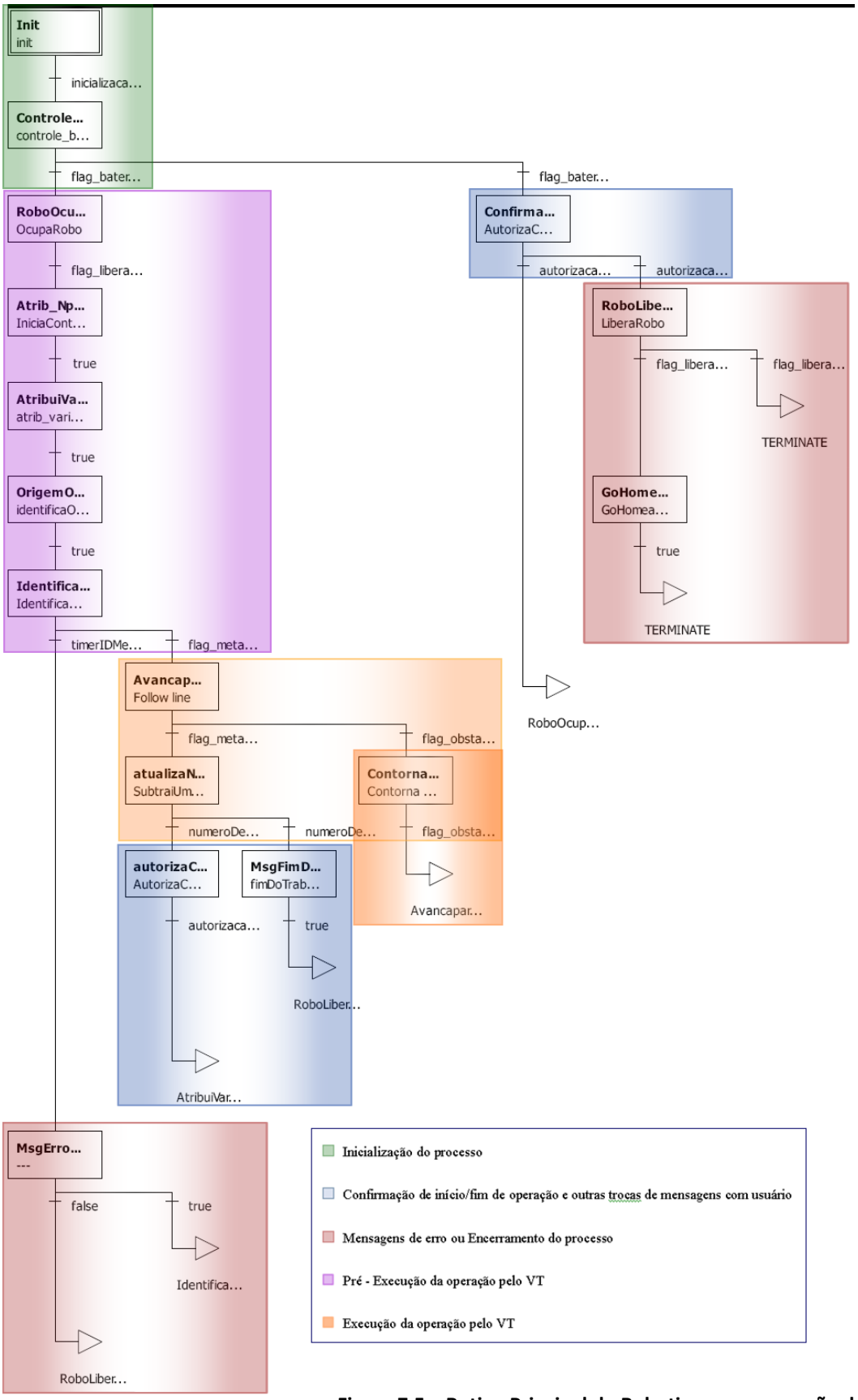

Figura 7.5 – Rotina Principal do Robotino para execução de tarefas

Pré – Execução da Operação pelo VT

Aqui identifica-se as funções que preparam o VT com informações sobre que parte da rota está sendo executada (se origem ou destino) e se informa quais os dados das localidades que o VT irá visitar, além de oficialmente ocupar o recurso no sistema, com uma flag. No primeiro bloco deve-se marcar o Robotino como ocupado no sistema. O segundo e terceiro blocos correspondem a variáveis de controle para contagem das rotas. No terceiro bloco são identificadas as características da meta (número da rota em meta) e atribuído valores para variáveis de controle. O quarto bloco deve indicar se o VT está indo para uma origem de rota ou destino. Informações sobre a localidade devem ser recebidas pelo sistema através do próximo bloco. (Posição / características da localidade).

Execução da Operação

Aqui deve-se ter as funções que correspondem à operação do robô. Ele deverá se locomover pelo layout físico do SPF através das linhas no chão identificando a localidade pela sua cor correspondente e posição de referência. Todo o processo deve ser assistido por câmeras dos robôs. A rotina consiste em detectar a meta pela sua cor. Depois, presença da linha, identificando a imagem através de um algoritmo interno. Detectada a linha, ele parte para cumprir a meta programada. Um sinal de obstáculo deve ser emitido assim que algo se aproximar do VT, reconhecido através dos seus sensores de infra-vermelho. Identificando um obstáculo ele deve processar o bloco que executa a operação de contorno do obstáculo e voltar ao traçado. Ele então deve voltar para a operação de seguir pela meta até alcançá-la.

Confirmações de início/fim e outras mensagens

Após atingir uma meta, o sistema deve atualizar suas variáveis e detectar se existe um próximo ponto na rotina. O usuário deve receber uma mensagem para autorizar que o VT parta para a próxima meta ou ser avisado da conclusão da operação. Dependendo do caso, o sistema deve voltar para as etapas de pré-execução ou parte para liberar o VT do sistema de controle.

Encerramento do Processo / ERROS

Nesse conjunto de operações, o VT deve ser liberado e ir para o estacionamento. Sendo o caso de estar com bateria em níveis mínimos, ele deverá se desligar. Possíveis mensagens de erros devem ser enviadas também nessas sequências, e a rotina também deverá ser encerrada.

7.1.2 Variáveis de projeto RobotinoView.

A tabela 7.1 apresenta as variáveis envolvidas no projeto dentro do RobotinoView. Elas irão se integrar com o WS posteriormente através de rotinas desenvolvidas em C#:

Tabela 7.1 – Lista de Variáveis do RobotinoView

Name	Type	Value
areaMeta	float	0
autorizacaoparaSeguir	float	0
consumoBateria	float	0
contadorAtualizado	float	0
contadorOrigDest	float	2
endOpMessage	float	0
errorMessages	float	0
flag_bateria	float	0
flag_liberado_ok	float	1
flag_meta_ok	float	0
flag_obstaculo	float	0
flag_rota_ok	float	0
followline	float	0
homeX	float	0
homeY	float	0
inicializacao	float	0
meta	float	0
metaX	float	0
metaY	float	0
numeroDeRotas	float	0
respostaUser	float	0
subtraendoNdeRotas	float	0
timerIDMeta	float	0
variavel1	float	0
variavel2	float	0
variavelNumeroRotas	float	0

7.2 Otimização e uso de Múltiplos VTs

Tendo a rotina de trabalho de um VT, em uma camada superior do sistema será introduzido a rotina que otimiza o uso dos VTs, atribuindo a tarefa que melhor convir para cada Robotino através de parâmetros principalmente de IP, status de ocupado / livre e distância do ponto inicial para execução da operação.

Como está admitido que todos os VTs nesse sistema possuem as mesmas características, não se fará distinção do uso entre eles para determinada rota.

Para a situação que se deseja simular nesse projeto, foi escolhido um método simples e que pelas dimensões e condições da tarefa se fará eficiente.

Para sistemas maiores, que envolvem localidades mais complexas ou que sejam mais distantes, até por divisão em zonas, o método precisa ser reavaliado e algoritmos mais complexos devem ser usados para se ter melhor eficiência, considerando centróide de massa do sistema, preferências por alocação de VTs para determinada tarefa, tipos de percursos percorridos quanto à dificuldade e tempo de locomoção, e tempo média de realização das tarefas, por complexidade .

7.2.1 Hipóteses adotadas para elaboração da rotina

Para desenvolvimento de uma rotina que trabalhe com múltiplos veículos de transporte, para esse estudo, serão adotadas algumas hipóteses que simplificam o processo de escolha:

1. Todos os VTs são semelhantes quanto às características físicas, autonomia ou demais atributos, sendo capacitados para desempenhar as mesmas funções.

    – *Com isso, pode-se admitir apto para realizar uma tarefa qualquer VT. Sendo que a escolha por um ou outro veículo será por questões particulares em determinada situação. Para esse caso seria possível implementar na classe de Controller do WS uma opção de escolha de um VT particular no momento da elaboração da tarefa.*

    – *Essa opção estará disponível na interface com o usuário do nosso WS, porém não será implementada nesse primeiro momento, por questão da premissa que todo o veículo está apto para realizar qualquer operação.*

    Ao considerá-la, pode-se introduzir no algoritmo um sistema de fila de tarefas para o robô requisitado. Ao término de uma rotina, o sistema faz a chamada da próxima tarefa pendente na fila daquele VT especificado. Na rotina do VT, o teste de autonomia deve ser automático com parâmetro pré-determinado por projeto e, caso ele não estiver apto para realizar a tarefa, uma mensagem deve ser fornecida ao usuário informando o ocorrido e a tarefa deve ser registrada como pendente no sistema. Uma nova função deve ser introduzida na interface do sistema, permitindo que o operador reenvie a tarefa pendente solicitada por ele, posteriormente.

2. O layout de produção estará situado em uma mesma região ou zona, não havendo necessidade dos VTs percorrem longas distâncias entre uma localidade e outra.

    – *Os estacionamentos serão relativamente próximos ou bem localizados com relação às rotas. Assim, existe pouca dispersão com relação à distância percorrida pelos veículos entre uma localidade e outra e o que determinará se um trabalho é mais extenso ou não será o número de rotas de uma tarefa.*

Essa conseqüência reflete no modo que o operador pode dividir as tarefas entre os VTs: ao invés de exigir que um VT faça uma tarefa muito grande (com várias rotas), ele ganharia tempo dividindo-a em partes e alocaria cada VT para realizar uma dessas partes. Essa decisão seria feita com base nas demais tarefas individuais que previamente estariam agendadas para os VTs e caberia ao(s) OPERADOR(ES), analisarem o plano de trabalho para aquele período e decidirem qual seria a melhor alternativa.

3. No caso estudado, os VTs não sobrecarregarão o ambiente de trabalho em questão de tráfego. A maior parte das rotas estará livre, mesmo admitindo todos os veículos em circulação.

   – *Podemos admitir que todos os VTs tem condições de trabalhar não tendo necessidade de rodízio entre os recursos e que tráfego entre localidades formar-se-iam em situações particulares: ou por exigência repentina da entrada de muitos insumos na linha de produção em pouco tempo, ou por alguma demanda sendo atendida em épocas de pico de trabalho. Para contornar essa situação, em casos esporádicos seria o caso do operador projetar a tarefa já considerando esse pico de tráfego e programando o VT para fazer um caminho alternativo ao destino.*

   – *Verificando que essa não seria uma exceção mas a regra, uma rotina com opções de rotas alternativas já otimizadas de acordo com a movimentação dos VTs para aquele período é interessante de ser analisada e implementada.*

7.2.2 Lógica principal de desenvolvimento

O procedimento envolve a seguinte lógica:

1. O VT será alocado através da lista de IPs disponíveis no momento da operação. Uma tabela comparativa com a lista de IPs de todos os veículos, operando naquele layout, possibilitará que manutenções sejam feitas. Por exemplo, se um IP está fora da lista o VT pode estar fora da área do roteador ou estar desligado. Assim, um operador é alocado para investigar se o problema é esse e corrigí-lo.

2. Dentro dos IPs disponíveis, será conferido na BD quais veículos encontram-se ocupados e estes serão descartados.

3. Dos veículos livres, será calculada a distância entre seu estacionamento (ponto em que eles devem se encontrar) e a primeira localidade na rota requisitada. A menor distância será considerada para escolha do VT.

4. Registra-se o IP do VT escolhido para que o sistema o chame por meio do WS e dele será inciciado o processo de execução da tarefa com sua rotina principal.

A Fig. 7.6 apresenta o fluxograma de atividades para a rotina de escolha dos VTs.

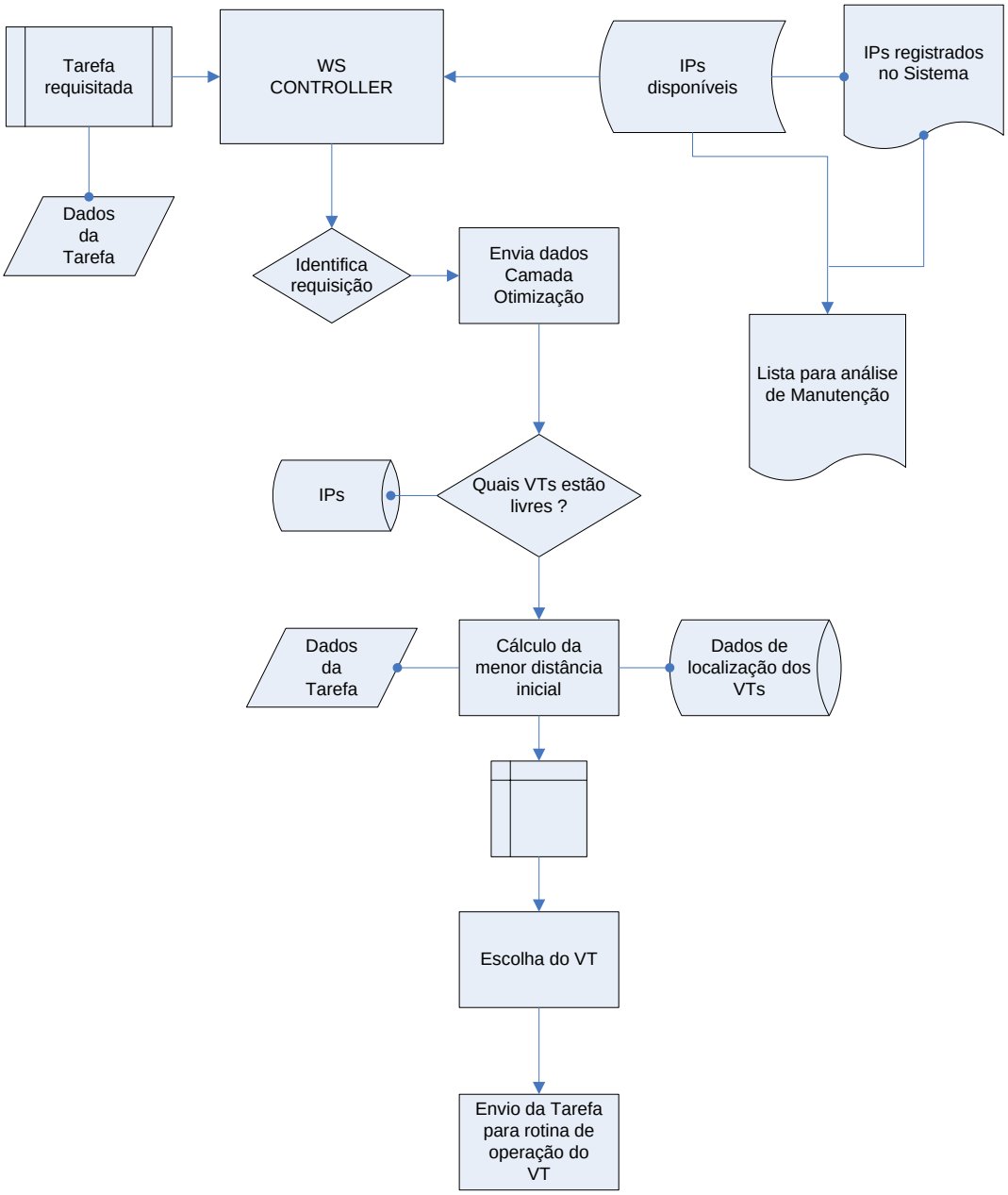

Figura 7.6 – Fluxograma da rotina de otimização

8.

Sistema Implementado

Foram apresentadas até aqui as etapas de construção da plataforma de integração entre os Sistemas Produtivos Fabril e de Transporte.

Fez-se todo o desenvolvimento teórico a partir das referências para que cada etapa pudesse ser desenvolvida criteriosamente com base em uma boa teoria que apoiasse o posterior desenvolvimento dos algoritmos por completo para implantação do sistema sem a necessidade de muitos re-projetos.

Serão apresentados agora detalhes construtivos relevantes para o desenvolvimento das etapas do projeto. Pela construção ser feita baseada em várias ferramentas e linguagens diferentes pretende-se não mostrar todos os pormenores de cada item, de cada software usado, entendendo que foge do escopo deste projeto, atendo-se ao relato de aspectos que influenciam no seu desenvolvimento.

8.1 Requisitos para desenvolvimento do sistema

As principais ferramentas para a construção do projeto foram os softwares:

- MS Visual Studio e atualizações (usada versão 2010);
- Macromedia Fireworks (suíte de aplicativos usada na versão 8)
- ROBOTINOVIEW 2.0

- ROBOTINOSIM (última atualização)
- API Robotino versão 2.41 para integração com ROBOTINOVIEW 2.0

Houve também a necessidade de instalação dos seguintes serviços no WINDOWS (plataforma de desenvolvimento – Windows Seven):

- ISS – Internet Information Service Manager para criação do servidor que hospedará o WS (versão 7).
- MS SQL Server para implementação da Base de Dados

8.2 Construção do Banco de Dados

Para facilitar o acesso aos dados da BD criou-se Stored Procedures – StP(s). As vantagens são:

- Facilidade de Manutenção devido a programação modular. Ao se alterar uma tarefa só precisar alterar o procedimento armazenado pertinente e não precisa muda o código do aplicativo.
- Como as **StPs** são analisadas e armazenadas na memória do servidor de banco de dados depois da primeira execução, sua execução é mais rápida do que usar instruções SQL pois essas devem ser analisadas a cada chamada.
- Com as **StPs** são armazenadas no banco de dados não se faz necessário enviar instruções SQL para o servidor , diminuindo o tráfego da rede.
- A segurança dos dados é mais robusta, pois se pode defini-la no nível de usuários, atribuindo permissões aos mesmos.

Assim foram criadas as StPs *listadas abaixo* para o sistema :

Com relação a Robots:

- SP_ProcuraIDRobo_Ip
- SP_ProcuraIPRobo_Id
- SP_AlteraStatusRobo
- Sp_AlteraStatusRoboId

    Parâmetros principais: robotID (int), IP (nvarchar)

Com relação à Operação

- SP_CreateOperacao
- SP_AlteraStatusOperacao

    Parâmetros principais: opID(int), userID(int), roboID(int)

Exemplo de modelo geral de um Procedimento Armazenado : (caso CreateOperacao)

A StP é criada dentro do aplicativo de desenvolvimento (Visual Studio) e entra como uma classe de desenvolvimento na Base de Dados, acessada diretamente pelo SQL Server através da chamada e seus parâmetros. Observa-se a seguir a StP CreateOperacao e a TAG SQL que propriamente usa a BD está inserida dentro da função StP.

```
ALTER PROCEDURE sp_CreateOperacao

	(
		@userID int,
		@roboID int,
		@name nvarchar(100)

	)

AS

DECLARE @param date
SET @param = current_timestamp

	INSERT INTO Operacao
					(userID, roboID, status, diaPedido, tipo, nome)
	VALUES		(@userID, @roboID, 0, @param, 0, @name)
	RETURN
```

A seguir, observamos a chamada dessa StP que será acrescentada a camada Controller recebendo os parâmetros do HTML/XML vindo das páginas de interface do usuário ou do procedimento automático do Sistema Fabril.

Modelo de chamada para a StP InsertUSer

```
Dim model As DataBaseEntity()

Try

	Model.sp_CreateOperacao(param1,  param2, param3)

Catch ex As Exception

	ViewData(“Message”) = “Falha ao criar a operação, tente novamente”

End Try
```

8.3 Modelo de Invocação automática do sistema pelo SPF

O sistema inicializado no usuário SPF deve efetuar as rotinas de maneira automática, de acordo com a requisição do Sistema Fabril através da detecção de sensores nas entradas e saídas da linha de produção.

Ao perceber a falta de insumos nas entradas ou excesso de produtos na saída da linha de produção uma rotina pré-ajustada é disparada no SPF para enviar uma mensagem ao WebService com os parâmetros de rota que o robô deve realizar.

O sistema recebe a mensagem (XML) identifica a ação a ser tomada e aciona um VT para realizar a tarefa.

Como parâmetros, são enviadas informações para:

- realizar nova tarefa,
- qual número de rotas seguidas pelo VT (1 rota = 1 par origem e destino),
- quais localidades o VT deve percorrer,
- escolha automática do VT e,
- não permitir controle manual do VT.

Para o caso do carregamento do VT ser manual têm-se:

  - O veículo de transporte se descola até o ponto de partida indicado na mensagem e fica aguardando o carregamento (sendo insumos ou produtos acabados). O carregador, através de qualquer recurso com acesso ao WebService (teclado e monitor no sistema fabril/ notebook/ palmetop +

navegador e internet)) e logado como SPF irá monitorar as mensagens enviadas pelo WebService aguardando a chegada do VT para carregamento.

- Feito o carregamento, irá dar **OK** para a janela que pede autorização para continuar o procedimento (indicando que o VT foi carregado).
- O VT caminhará para a próxima localidade onde um novo operador/descarregador estará pronto para realizar o mesmo procedimento.

8.4 Interação de usuário com Sistema de Integração

Listadas todas as características da camada de Interface, durante a construção do sistema, programaram-se mecanismos que permitiam certa flexibilidade para o usuário interferir ou não na rotina do VT.

A seguir, apresenta-se a rotina implementada nesse trabalho:

1. O usuário, através do site/WS ira requisitar uma tarefa.
2. O sistema devolverá a ele o formulário para registro de tarefa, solicitando informações sobre usuário requisitante, robô requisitado e nome da tarefa.
3. Ao fornecer as informações, o usuário é levado a tela de inicialização da operação de transporte. O sistema devolverá também o aplicativo de controle aberto para início da operação. Ao mesmo tempo ele registra na base de dados que o VT está alocado (flag de ocupado).
4. Assim que realizada a tarefa, o usuário, no WebService precisará clicar em ENCERRAR TAREFA.

Essa operação finaliza o aplicativo e registra na BD a conclusão da tarefa, liberando o veículo de transporte utilizado (Fig. 8.1 e 8.2).

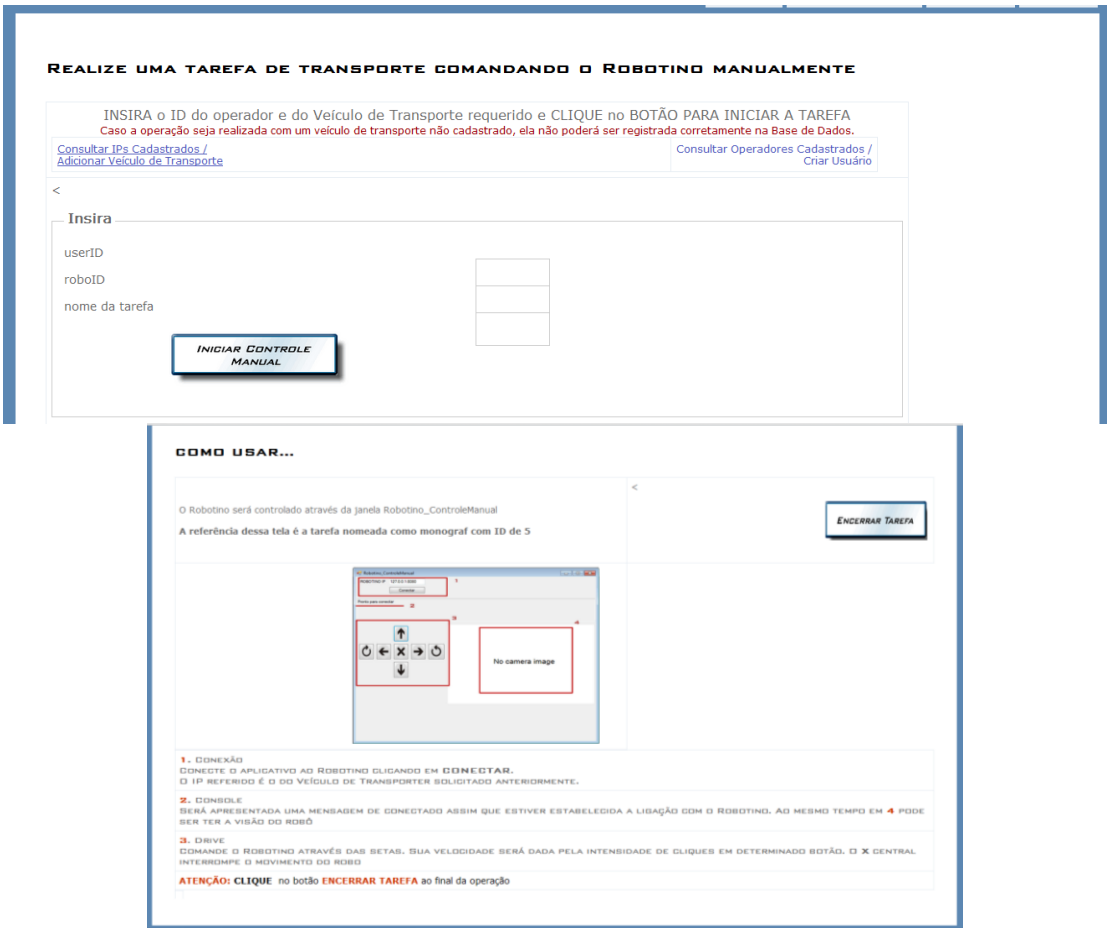

Figura 8.1 – Tela de acesso a registro de tarefas

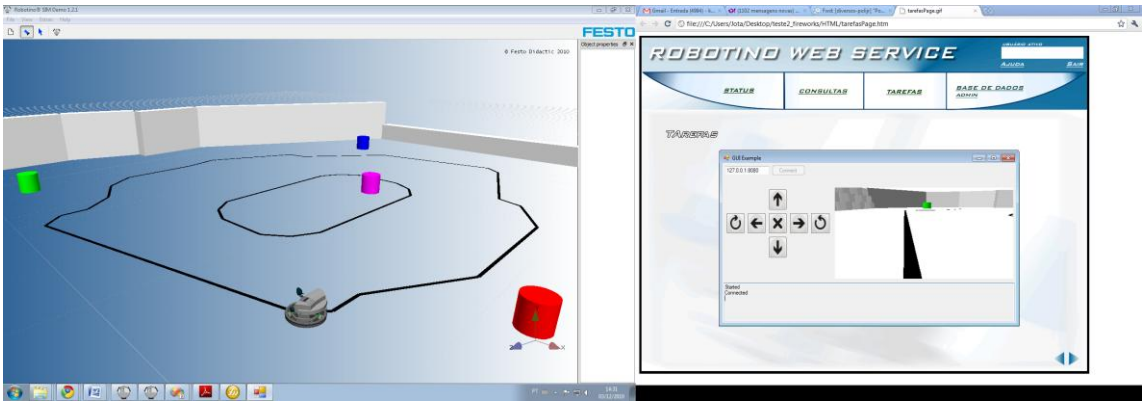

Figura 8.2 – Tela de visualização do VT sendo comandado manualmente pelo WS

É importante observar que o WS foi desenvolvido para poder acessar simultaneamente mais de um veículo de transporte. Ao dar entrada de uma tarefa, o sistema avisa, na tela de operação (“Como Usar”) qual o nome da tarefa ai requisitada e qual a sua identificação na BD. Clicando em Encerrar, o sistema fechará o processo somente daquela tarefa específica, atualizando-a na BD.

8.5 Aplicativo de Controle Manual do Veículo de Transporte

A solução de controle do veículo de transporte foi implementada, como dita no capítulo 7, utilizando um “formulário” no Windows (WinForm). Através desse objeto, pode-se criar um aplicativo com iteração com usuário com uma ampla gama de recursos de programação e visuais, gerenciados por ferramentas existentes no próprio framework do sistema operacional usado.

A utilização dessa estrutura / objeto WinForm necessita que o aplicativo seja rodado na máquina do cliente ou que o componente, no momento do desenvolvimento seja transformado em uma WebForm (padrão de formulários no mesmo principio do WinForm mas com uso através da web). Uma WebForm pode ter mais limitações de recursos, por ser gerenciada via navegador e não instalada e configurada na máquina utilizada.

No desenvolvimento do aplicativo ponderou-se o fator flexibilidade no desenvolvimento do mesmo, facilidade na disponibilização da solução para o usuário e tempo de resposta do produto, além da questão de confiabilidade na estabilidade da solução.

Ponderando-se esses fatores, chegou-se ao desenvolvimento de um aplicativo com todos os benefícios para rodar na máquina do usuário (velocidade, interface mais avançada,

estabilidade) aliado à flexibilidade para disponibilização pela internet (necessária por se tratar de um WebService).

Procurando pelo desenvolvimento desse tipo de solução chegou-se em um conceito recente de aplicativos que fazem parte das ferramentas de desenvolvimento do próprio Visual Studio denominados ClickOnce. São aplicativos que rodam no cliente (com ou sem instalação pelo usuário, dependendo das características da solução), mas que podem ser distribuídos pela internet. Entre as características possíveis para o desenvolvimento de uma solução ClickOnce, implementou-se nesse trabalho as seguintes características:

- Solução online disponibilizada através do WebService 'não sendo necessário a instalação" - o aplicativo é instalado mas fica armazenado nos arquivos temporários do cliente até o término do seu uso. Não consumindo memória em disco ou sendo necessário maiores configurações. Após fechado, o cliente fica com registro em cachê no computador das principais diretrizes usadas pelo aplicativo. Na sua próxima utilização – somente online, ele irá ser acessado mais rapidamente, caso não tenha atualizações.

- Uso de interface gráfica avançada, como com recursos de captura de imagem, sem a necessidade de passar pelo servidor ou de alguma ponte para acesso ao VT - com a aplicação rodando do lado do cliente, menos tráfego de rede é gerado tornando-a mais rápida e segura.

Em contra partida a essa solução, por ser uma tecnologia nova, somente alguns navegadores a suportam por completo. Uma preocupação menor é que o cliente deverá autorizar o uso da solução e possivelmente aceitar o uso de certificado para confiabilidade do aplicativo (tecnologias e tipo de segurança comum nos dias de hoje para desenvolvimento web).

Como navegadores mais habilitados para uso do aplicativo se mostraram o Internet Explorer e o Mozilla Firefox (com plugin para uso de aplicativos ClickOnce).

9.

CONCLUSÕES

9.1 Considerações Gerais

Com esta monografia buscou-se apresentar o desenvolvimento de uma plataforma de integração entre diferentes sistemas que compõe um ambiente produtivo.

Desde a identificação da necessidade, passando por toda a referência bibliográfica, divisão do processo de desenvolvimento do sistema e especificação final da solução com seus requisitos, apresenta-se aqui todas as etapas que compõe a solução, fazendo esse texto servir como parâmetro e referência para a implementação final desse projeto, próxima meta a ser realizada.

Entendeu-se que era preciso uma busca detalhada nas referências bibliográficas para compreender todos os componentes que iriam fazer parte do sistema projetado e assim poder desenvolvê-los de maneira satisfatória. Diversos são os requisitos para esse projeto e com as ferramentas disponíveis hoje, muitas soluções poderiam ser usadas. Identificar os pontos positivos e negativos de cada uma para a elaboração de uma solução realmente ótima, como modelo viável para aplicação em um ambiente industrial real necessita de uma boa base teórica como apoio.

No geral, muitas hipóteses simplificadoras foram adotadas no modelo e ao final do desenvolvimento do projeto da solução para integração dos sistemas de fabricação e transporte, entende-se que os procedimentos gerais adotadas podem ser estendidas para um caso real e para diferentes plataformas.

Analisando as soluções adotadas para esse projeto e o perfil das empresas que adotar medidas desse tipo para atualização do layout de produção, percebemos algumas características gerais que devem ser respeitadas como boas-práticas para se obter sucesso:

- A manutenção das instalações é fundamental para que :
    - Operadores possam saber o que esperar com as mudanças na linha de produção e assim tomarem decisões assertivas quanto à operação, de fato, das máquinas.
    - Os sistemas possam identificar como cada máquina está reagindo de maneira eficiente e consigam passar essas informações com precisão tanto para a base de dados quanto para as máquinas em operação.
- Como a solução visa à integração entre partes diferentes do ambiente fabril, a presença de uma estrutura administrativa que dirija os setores envolvidos de maneira a respeitar suas funções e integrá-las sem perda de eficiência é necessária. As informações devem ser bem organizadas e analisadas periodicamente para se obter o melhor das instalações.
- É necessário investimentos em comunicação. Para a solução adotada, uma rede eficaz de dados deve ser estabelecida e mantida, tanto para a comunicação entre os recursos humanos ou máquinas quanto para integração entre eles. Essa rede deve ser estável e deve permitir que os dados trafeguem com segurança e fluidez durante todo o funcionamento daquela linha.
- É preciso que os recursos humanos sejam bem administrados, mantendo um pessoal atualizado quanto ao uso das máquinas e administração dos processos durante a produção. As soluções adotas aqui, mesmo tendo caráter de automatização dos

recursos, envolvem sistemas onde o homem ainda é elemento principal para da produção, responsável por tomar decisões constantemente sobre o andamento da produção, influenciando assim diretamente na qualidade do produto final (tanto na visão dos consumidores, quanto da empresa).

- As soluções envolvem investimentos em tecnologias que muitas vezes são novas para a indústria. Dependendo do estado das máquinas ali instaladas, para se obter os resultados desejados, componentes necessitam ser instalados, atualizados ou melhor calibrados para se poder fazer o processo de integração da linha. Esse fator deve ser bem analisado pela gerência para não tomar medidas precipitadas como gastos em itens que não serão necessários, mas que por pouco cuidado ou desinformação, foram adquiridos antecipadamente.

9.2 Considerações Específicas

Buscou-se uma solução para o problema de integração que fosse viável mercadologicamente com ferramentas disponíveis para sua implantação. Os requisitos técnicos envolvem diversas competências trabalhas durante o curso de graduação da Escola Politécnica da USP e pertencem ao universo dos formandos aqui graduados. A solução assim se mostra como um excelente meio de demonstrar o entendimento do curso de engenharia vivido na graduação.

Toda a abordagem feita durante essa fase do projeto foi feita através de um modelo. Nas decisões tomadas, formas de apresentação e demais fatores buscou-se obedecer às características desse modelo e entende-se que alguns fatores fariam parte da matriz de decisão para implementação da solução em uma indústria real, foram estrategicamente

deixados de lado, principalmente pela escala da implementação da solução e demais objetivos que o trabalho visou.

Esse também é um dos fundamentos buscados no curso de engenharia: simplificar a realidade traduzindo a situação em um modelo que possa ser analisado e solucionado com requerida precisão, segurança e satisfação e, essencialmente viável.

Pode-se avaliar o poder das ferramentas existentes hoje para o aprimoramento da produção em indústrias e em escala geral a comunicação entre as pessoas.

Na solução aqui apresentada, foi feito com que duas plataformas construídas de diferentes maneiras interagissem de forma completamente harmônica, se complementando e agregando muito valor ao sistema. E tudo, feito ainda de maneira remota, podendo se desconsiderar a necessidade de um operador, em primeira instância.

Nesse ponto, cabe resaltar que é muito importante a intervenção do elemento humano (localmente ou a distância) para administrar todo o processo, ver suas falhas, aperfeiçoar rotinas e gerenciar custos. Isso pode ser visto no projeto através das possibilidades de gerenciamento do banco de dados e consumo das suas informações.

Que fique destacado portanto os benefícios que se pode obter trabalhando com WebServices e soluções “em nuvem”, onde diminuímos as necessidades da interação física entre os componentes do sistema e através da internet ou de redes sem fio podemos gerenciar diversos processos de maneira confiável e extremamente eficiente, tendo infinitas possibilidades de construção da solução.

Nesse projeto foi escolhido um caminho, porém existem no mercado diversos outros algoritmos, linguagens e arquiteturas para se resolver o mesmo problema, bastando que se faça as escolhas compatíveis com os custos e realidade da empresa.

9.3 Trabalhos Futuros

Acredita-se como legado desse projeto esteja no estudo e realização das demais idéias trabalhadas aqui. Criou-se o sistema de integração com a implementação de uma das soluções para extensão do ambiente fabril estudado (o aplicativo para controle manual dos veículos de transporte). Partindo-se do desenvolvimento já feito (inclusive com as rotinas de trabalho), tem-se referência dos demais passos para se chegar a uma solução completa de flexibilização e automação do ambiente fabril.

Os próximos passos para estudo, dando continuidade ao projeto, seriam de sensoriar uma planta industrial para enviar comandos ao WebService assim que tiver necessitando de matérias-primas ou com acúmulo de produtos finalizados na esteira. O comando chama o serviço em nuvem para alocar um veículo de transporte e fazer o carregamento, de fato, desse material.

Um operador poderia fazer o serviço ou interferir quando necessário, usando as ferramentas aqui desenvolvidas, ou o Robotino poderia fazer o trabalho completo, como exposto no planejamento feito nos capítulos anteriores.

Acredita-se nos inúmeros benefícios no processo de aprendizagem com esse trabalho (e aqui vai a ressalva da experiência pessoal do autor) devido ao uso de conceitos e tecnologias de vanguarda, como o desenvolvimento de aplicativos em nuvem, integração de Sistemas por WebServices, o padrão MVC (e derivados, como MVP – Model, View, Presenter, para soluções locais); que traz inúmeros benefícios para o programador do sistema, devido a sua organização, facilidade de implementação, flexibilidade, disponibilidades de ferramentas. A possibilidade de trabalhar com projetos que podem realmente ser levados a indústrias ou empresas, oferecendo novas soluções e idéias, trazem a sensação de que se tomou o rumo

certo na escolha da formação e auxilia o processo de concientização, dentro de todo o contexto do projeto em meio acadêmico, de quais áreas são aderentes para o futuro profissional.

A integração e uso de diversos conceitos estudados durante o curso de engenharia e que, por diversos fatores, necessitam ser aplicadas para serem percebidas e assimilados pelo aluno, torna-se muito claro durante o desenvolvimento de um projeto desta natureza. A satisfação final em ver que se está trabalhando com conceitos muito bem vistos pela comunidade desenvolvedora e que se pode entregar uma solução robusta e viável é gratificante e permite que o aluno aí já cresça mais nesses primeiros passos da vida profissional.

10.

REFERÊNCIAS

ITO, Y. A Diserable production structure looking toward the 21st century – Anthropocentric Intelligence-Based Manufaturing, In: XI COBEM. Anais, Brazil, S. Paulo, p.23 – 32, 1991.

Santos Filho, D. J. Apostila da matéria PMR2460. Redes de Petri, Sistemas Produtivos – informações. Oferecida no curso de PMR2460 – Modelagem e Controle de Sistemas Discretos, Escola Politécnica da Universidade de São Paulo, 2007.

Santos Filho, D.J. Aspectos do Projeto de Sistemas Produtivos. Tese de Livre Docência, Escola Politécnica da Universidade de São Paulo, 2000a.

KFEICHTNER, Cloud Computing Defined. < http://www.greenhousedata.com/2010/12/03/cloud-computing-defined-iaas-vs-saas-vs-paas/>. Acesso em: 21 março 2011

Computação Embarcada: Projeto e Implementação de Veículos Autônomos Inteligentes <http://osorio.wait4.org/oldsite/palestras/jai2005.html> Acesso em: 22 março 2011

Santos Filho, D. J., Controle de Sistemas Antropocêntricos de Produção Baseados em Redes de Petri Interpretadas. Tese apresentada a EPUSP para obtenção do título de Doutor em Engenharia, Escola Politecnica da universidade de São Paulo, 1998.

Miyagi, P. E.,Controle Programável. São Paulo: Ed. Edgard Blücher LTDA , 1996.

SUN MICROSYSTEMS Distributed Application Architecture**,** cap. 7 - <http://java.sun.com/developer/Books/jdbc/ch07.pdf>. Acesso em: 03 março 2011

Sant'Anna, M. Implantando Aplicativos WinForms com ClickOnce– Definições <http://msdn.microsoft.com/pt-br/library/ms953320.aspx>. Acesso em 21 março 2011

Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal, Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities**.** Laboratory Department of Computer Science and Software Engineering The University of Melbourne, Australia < http://www.cloudbus.org/~raj/papers/hpcc2008_keynote_cloudcomputing.pdf>. Acesso em: 22 março de 2011

Araújo Júnior, L.O. de, Método de programação de sistemas de manufatura do tipo Job Shop dinâmico não determinístico / L.O. de Araújo Junior, Ed. rev . Tese de Doutorado apresentada à Escola Politecnica da Universidade de São Paulo, 178p, – São Paulo, 2006

ROCHA, José Antonio Meira da. Modelo de Trabalho de Conclusão de Curso (TCC). Modelo de documento digital do programa OpenOffice 2.0 disponível em <http://www.meiradarocha.jor.br/uploads/1021/196/modelo_de_projeto_de_TCC-2006-06-12a.sxw>. Acesso em: 03 março. 2011.

Araujo, Sidnei Alves de. , Librantz, André Felipe H. , Filho, Oswaldo Flório. Navegação Autônoma de Robôs: Uma Implementação Utilizando o Kit Lego Mindstorms. <http://200.169.53.89/download/CD%20congressos/2006/Sulcomp/pdf/22003.PDF> Acesso em : 18 março 2011

Robotino – FESTO, referências . < http://www.festo-didactic.com/int-en/>. Acesso em: 18 março 2011

Nakamoto, F.Y. Sistematização do Projeto do controle de sistemas produtivos. Dissertação apresentada à Escola Politecnica da Universidade de São Paulo, 147p, 2002.

Web Service SOAP e Aplicações Web em <http://devedge-temp.mozilla.org/viewsource/2002/soap-overview/index_pt_br.html>. acesso em 03 março 2011

Web Services Tutorial - W3C em <http://www.w3schools.com/webservices/default.asp> acesso em 03 março 2011